Reflections - Bitcoin Price Prediction Model

1- Project's objective

This project continues the analysis initiated by the 'Bitcoin-price-prediction-using-lstm' project : https://www.kaggle.com/code/meetnagadia/bitcoin-price-prediction-using-lstm.

The purpose is to improve the predictions generated by this model. This document synthesizes my thoughts on this subject. I've structured them to make them as clear as possible. The project containing the code and data used can be found in the following repository : https://github.com/SydneyJezequel/btc_neural_network

Feel free to use it and contribute to the thinking.

Note about the calculation of predictions :
In this sandbox, the calculation of predictions (class: *DisplayResultsService*) is not complete:
- It requires corrections to provide accurate predictions.
- It has only been implemented in the training *training_with_one_dimension/price_pred_1_lstm_layer.py*.
If you want to go further, you should manage this point.

- Repository for the *DisplayResultsService* class:
https://github.com/SydneyJezequel/btc_neural_network/blob/main/service/display_results_service.py
- Repository for the *training_with_one_dimension/price_pred_1_lstm_layer.py* training:
https://github.com/SydneyJezequel/btc_neural_network/blob/main/training_with_one_dimension/price_pred_1_lstm_layer.py

2- Starting point

Prediction of Bitcoin price is a regression problem.

The initial project attempts to predict the price of Bitcoin using 2 years of daily prices and an LSTM neural network. LSTM neural networks are particularly well-suited for time series predictions. They retain information over long periods and are able to capture long-term relationships between data and trends.

The initial model has following characteristics :

- Number of LSTM layers : *10 with a RELU activation function.*
- Number of Dense layers : *1*
- Number of Epochs : *200*
- Batch size : *32*
- Loss function : *mean_squared_error*
- Optimizer : *adam*

```
# Création du modèle :
model = Sequential()
model.add(LSTM(10, input_shape=(None, 1), activation="relu"))
model.add(Dense(1))
model.compile(loss="mean_squared_error", optimizer="adam")

# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=200,
    batch_size=32,
    verbose=1,
    callbacks=[MetricsCallback()]
)
```

3- Way to improve the model

I followed 2 paths to improve the performance of this model:

- Enriching the dataset.
- Modifying the model.

3.1- Enriching the dataset

3.1.1- Integrate new data

- **Increasing the dataset size :** My first idea was to provide more data to the model. I took the BTC daily price historic from 2013 to 2025 in order to capture more relationships.

- **Adding technical indicators to the dataset :** I have integrated technical indicators' signals into the dataset. I have selected the 150, 100, 50 moving averages and RSI. They are widely used indicators in the financial market. I have chosen the 150 moving average instead of the 200 moving average. The behavior of this average is very close to the 200 but it is slightly more reactive.

3.1.2- Feature engineering

I used features engineering techniques to improve the quality of the dataset.

- **Normalization of the dataset :** I used the normalization from the initial project. I extended it over the period 2013 - 2025 to manage the significant variations in Bitcoin's price. I applied normalization only to the price of Bitcoin, not to moving averages. Moving averages are by nature less subject to volatility. I have made some tests and using normalization on the moving averages has deteriorated the training results. To normalize the price of Bitcoin, I used a MinMaxScaler with a 0-1 interval : Prices were converted to a range from 0 to 1.

- **Subsampling of the oldest data :** The first training sessions yielded poor quality results due to the important Bitcoin's price variations. I used subsampling to reduce the oldest data to 10% of the dataset and give more weight to recent data. It helped me to reduce selection bias. Data considered as « old » are data prior to 2020. 2020 is a « pivot » year in the development of cryptocurrencies: Since the 2020/2021 bull run, there has been an increase in public and institutional interest in the crypto currency ecosystem. It contributes to the growth of this sector.

- **Adding feature lags :** It is the incorporating of the lagged price values from X periods to each row of the dataset. Their addition helps the model to better identify temporal relationships. I conducted tests by incorporating lags of 1 day, 7 days (1 week), 30 days (1 month), 60 days (2 months), 90 days (3 months), 180 days (6 months), and 365 days (1 year).

- **Adding historical volatility :** It quantifies the dispersion of Bitcoin's returns. It helps the model to identify volatility patterns which precede certain price movements. Volatility is determined by the standard deviation of Bitcoin's logarithmic returns. We calculate these returns by taking the logarithm of the ratio between the BTC price on the current day (D) and the BTC price on the previous day (D-1). A key advantage of logarithmic returns is their symmetry : A 10% increase followed by a 10% decrease brings the price back to its initial level. It is useful to estimate volatility.

3.2- Modifying the model

- **Using cross-validation :** I integrated cross-validation into the original model (file : « *training_with_one_dimension/price_pred_cross_validation.py* »). It consists to divide the dataset into several subsets to test the model on different portions. My objective was to reduce the risk of overfitting. However, this technique was not relevant because it is not suitable for temporal data such as the price evolution of stock market assets.

- **Using Gradient boosting :** I used the Gradient boosting technique (file : «*Price_prediction_neural_network_with_gradient_boosting.py* ») to train the model. This method consists of building a model by successively adding decision trees. Each new tree corrects errors of the previous one. The generated predictions are the average of predictions of all the trees.The results showed that the model was overfitted. For example, R^2 and Explained variance score metrics showed significant differences between training and test phases. The model worked very well on the training dataset but was unable to generalize to the test dataset.

```
train_evs :  0.9999948470958746
test_evs :  0.02452912778165217
train_r2 :  0.9999948470958746
test_r2 :  -1.3482410056324623
```

- I have enriched the dataset and modified hyper-parameters of the model *(ex : number of epochs, batch size, etc.)*. However, it does not enhance the results.

- **Integrating drop out :** I have integrated drop out to reduce the risk of over-adjustment. Drop out consists of randomly deactivating a certain percentage of neurons during training. The objective is to reduce the risk of overfitting and make the model more robust because it is based on the entire network, not on certain neurons. This did not bring any significant improvement.

- **Integrating early stopping :** I have integrated early stopping to prevent the risk of overfitting. It degraded the obtained metrics, even by adding a patience of 10 to 20 epochs. I did not retain this solution.

4- Trainings details

First, I have trained models only with the daily closing price of Bitcoin.
Then, I have enriched the dataset with elements described in the previous section : *3.1- Enriching the dataset*.

The following table summarizes the configurations (model/dataset) used in training sessions.

File	Training characteristics	Training on a one-dimensional dataset *(BTC price)*	Training on a multi-dimensional dataset *(BTC price, moving averages, RSI, moving average crossovers, lags features)*
price_pred_1_lstm_layer.py	1 lstm layer	X	X
price_pred_2_lstm_layers.py	2 lstm layers	X	X
price_pred_3_lstm_layers.py	3 lstm layers	X	X
price_pred_3_lstm_layers_improved.py	Integration of the following elements into the price_pred_3_lstm_layers.py model : batch normalization, drop out, tanh activation function, reduction of the optimizer's learning rate, adjustment of the learning rate during the training session.		X
price_pred_3_bidir_lstm_layers.py	3 bidirectional lstm layers		X
price_pred_attention_layer.py	Addition of attention mechanisms to manage periods of high volatility.	X	X
price_pred_cross_validation.py	Addition of the cross-validation to reduce the risk of overfitting.	X	
price_pred_gradient_boosting.py	Using of gradient boosting.	X	
price_pred_gru_layer.py	Using of 1 GRU neurons layer instead of LSTM layers.	X	X
price_pred_huber_loss.py	Using of the Huber loss function.	X	X

price_pred_lags_features.py	Enrichment of dataset with lags features.		X
price_pred_lags_features_and_vol.py	Enrichment of dataset with lags features and the historical volatility of the btc price.		X
price_pred_learning_rate_optimizer.py	Modification of the optimizer's learning rate.	X	X

- Models used to train the one-dimensional dataset are available at this location : https://github.com/SydneyJezequel/btc_neural_network/tree/main/training_with_one_dimension
- Models used to train the multi-dimensional datasets are available at this location : https://github.com/SydneyJezequel/btc_neural_network/tree/main/training_with_many_dimensions

5- Selecting the best model

5.1- Training : Dataset with the BTC closing daily price

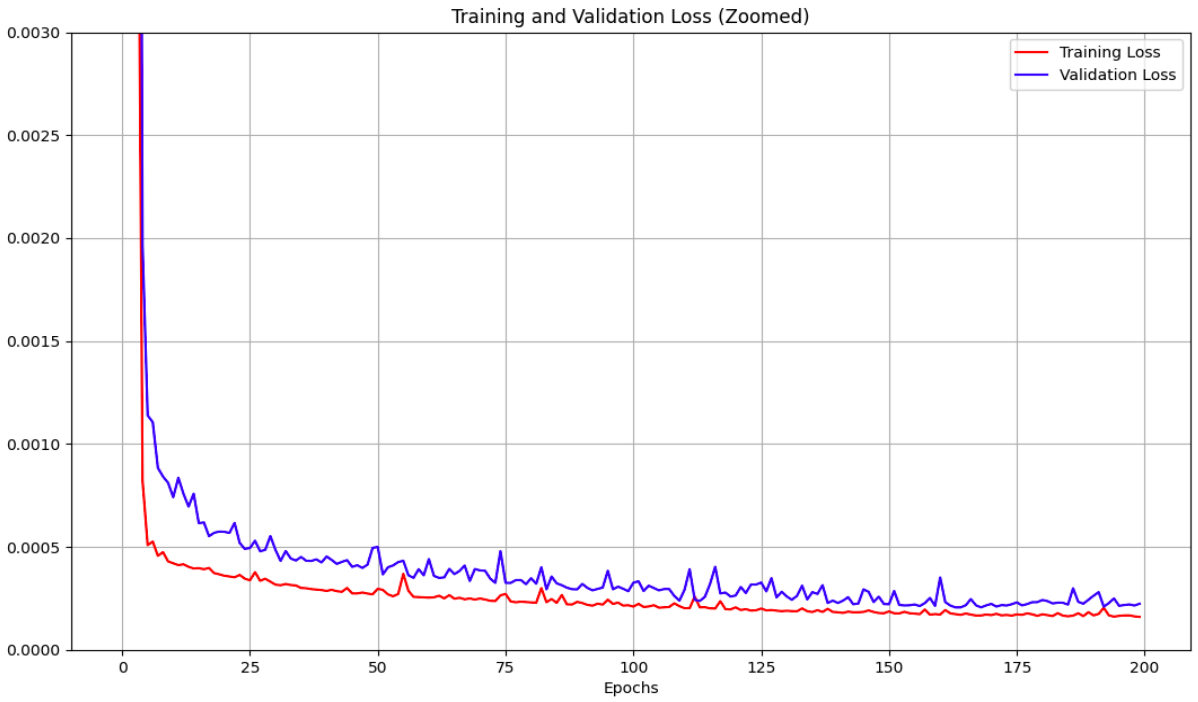
5.1.1- Selecting the most interesting models

Following table gathers characteristics of training sessions that provide the best metrics and loss curves for a one-dimensional dataset. The models which provided the best results contain only one layer of neurons. The difference between these models was in the settings of hyper-parameters *(number of neuron layers, number of epochs, batch size).*

	Neurons number	Epochs number	Batch size
Training session 1	10	200	32
Training session 2	10	300	32
Training session 3	15	300	50

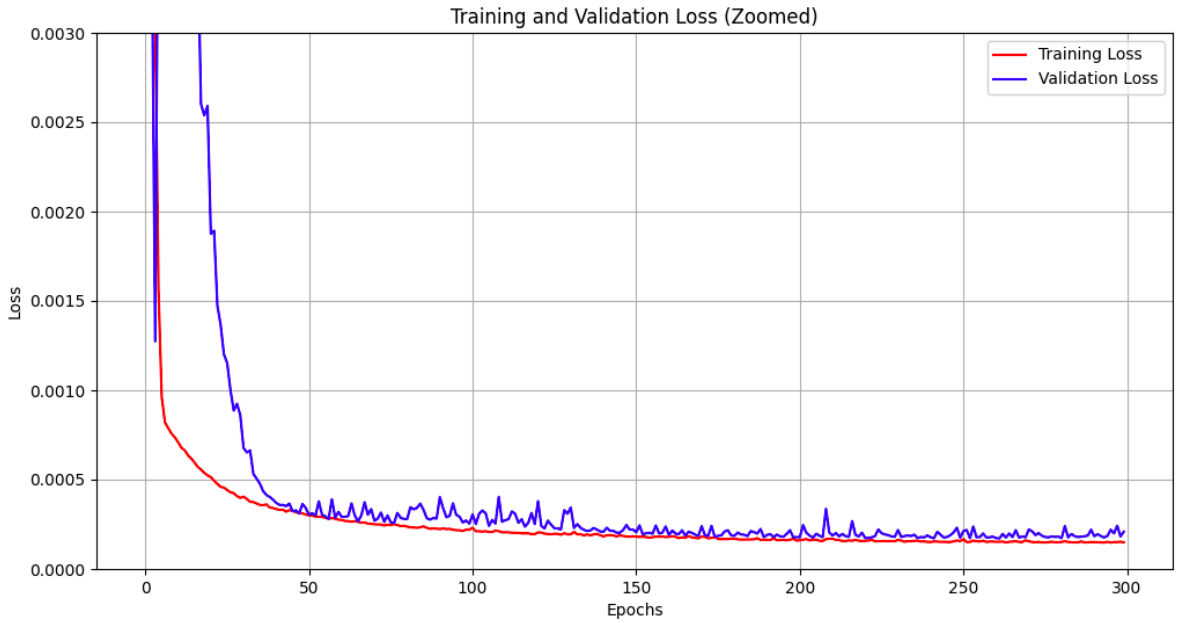
5.1.2- Comparing results of models

- Training session 1 :



Comments:
Validation and training loss curves are stable. The both curves converge.

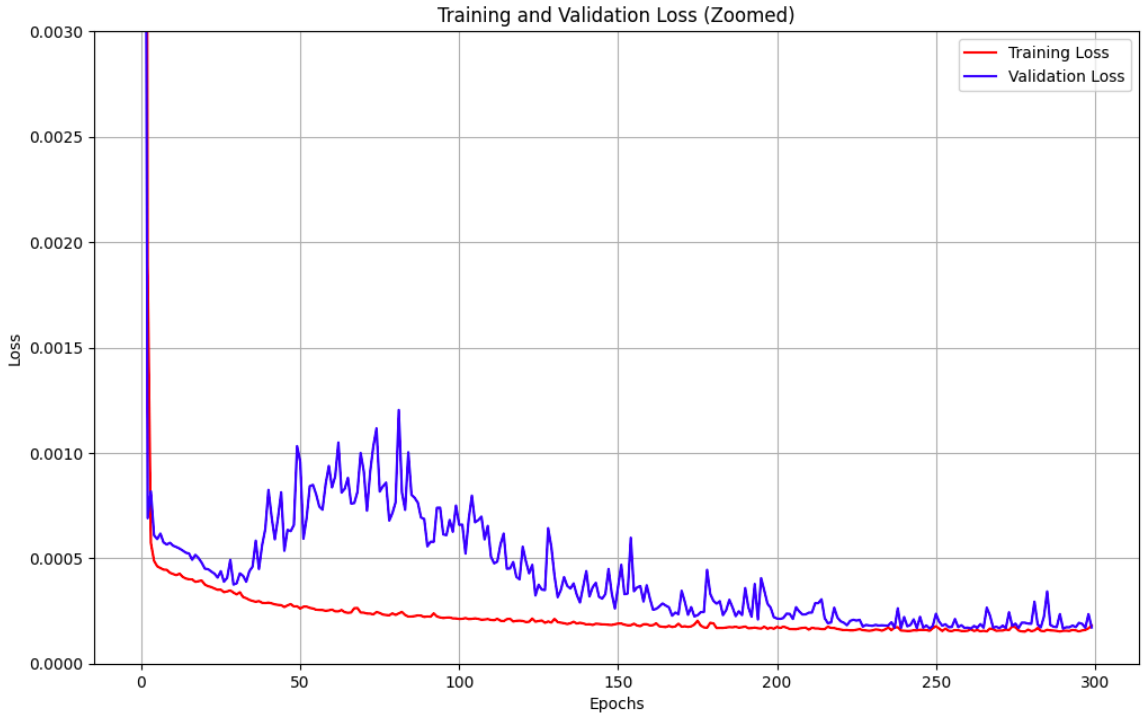
- Training session 2 :



Comments:

Validation and training loss curves are more stable compared to training session 1. They converge but diverge slightly at the end. It could indicate a potential overfitting.

- Training session 3 :



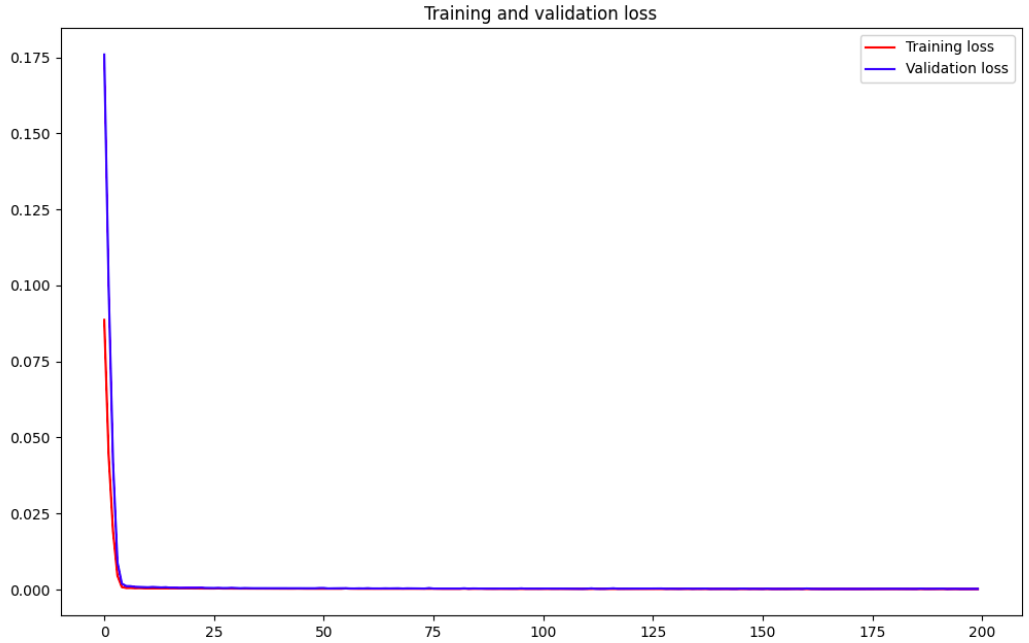
Comments:
Validation loss curve is more volatile than training sessions 1 and 2.

5.1.3- Detailed analysis of the 1st training's results

5.1.3.1- Loss curves

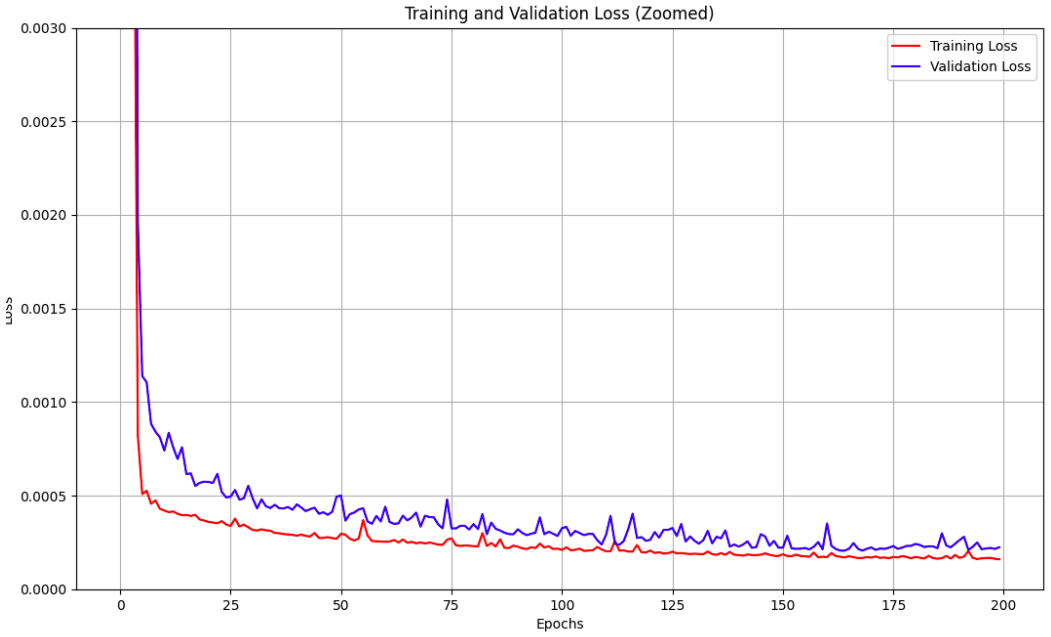
- **Loss curves :**

Training and validation loss curves converge.



- **Loss curves (Zoom) :**

The both curves do not overlap. The model doesn't appear to be overfitting.



5.1.3.2- Metrics

Following table shows the model's metrics at Epochs 50, 100, 150, and 200.
We will use it to evaluate the model's ability to generate good predictions during training and testing.

Metrics of Training 2

Epoch	train_rmse	train_mse	train_mae	test_rmse	test_mse	test_mae	train_explained_variance	test_explained_variance	train_r2	test_r2	train_mgd	test_mgd	train_mpd	test_mpd
50	0.01029	0.000106	0.00685	0.01344	0.000181	0.00866	0.99103	0.99178	0.99029	0.98918	0.01112	0.00181	0.000706	0.000520
100	0.00897	0.000080	0.00587	0.01043	0.000109	0.00667	0.99303	0.99404	0.99295	0.99377	0.00824	0.00110	0.000524	0.000311
150	0.00882	0.000078	0.00603	0.00934	0.000087	0.00662	0.99416	0.99541	0.99335	0.99513	0.01033	0.00099	0.000503	0.000264
200	0.00797	0.000064	0.00515	0.00941	0.000089	0.00600	0.99475	0.99520	0.99461	0.99510	0.00877	0.00077	0.000399	0.000234

Metrics used:

- **RMSE (Root Mean Squared Error) :** It measures the difference between predicted values and actual values. A low RMSE indicates that the model makes accurate predictions. RMSE is sensitive to large errors due to the squaring of the differences.

- **MSE (Mean Squared Error) :** It measures the average of the squared differences between the predicted values and the actual values. It gives more weight to large deviations between predictions and actual values, meaning that significant errors are penalized more heavily. A low MSE indicates that the predictions are close to the actual values, with few large discrepancies.

- **MAE (Mean Absolute Error) :** It measures the average of the absolute errors. It's less sensitive to large variations than RMSE and MSE because it doesn't square the errors. This means that large discrepancies between predicted and actual values won't disproportionately influence the metric.

- **EVS (Explained Variance) :** It measures how much the model's predictions deviate from the actual values. It quantifies the dispersion of the predictions around the mean of the actual values. An EVS close to 1 means the model explains well the variance in the data.

- **R^2 (Coefficient of Determination) :** It indicates how much the model succeeds to explain the fluctuations of observed values. It evaluates the dispersion of the data around the mean of the model's predictions. A R^2 close to 1 means the model explains a significant portion of the variance in the data.

- **MGD (Mean of Gradients of Errors) :** Prediction Spread
- This metric evaluates the stability of the model for different inputs. It indicates how the differences between predictions and actual values vary based on the input data. Constant and small errors suggest a stable model.

- **MPD (Mean of Differences between Predictions and Actual Values) :** This metric assesses whether the model tends to overestimate or underestimate the actual values. It indicates the direction (higher/lower than actual values) and trend (increasing/decreasing) of the differences between predictions and actual values. A low MPD means the model makes accurate predictions.

5.1.3.3- Analysis of metrics

We can make the following observations :

1. RMSE, MAE, and MSE : These metrics decrease over epochs. It means the model's predictions are improving. However, there's a slight increase in test_rmse and test_mse between epochs 150 and 200. It indicates the performance on the test dataset is deteriorating while the training metrics continue to improve. This could suggest the model is beginning to overfit.

2. Explained Variance and R^2 : Metrics increase over epochs. It means the model explains better and better the variance of data. There is a slight decrease in test_explained_variance and test_r2 at epoch 200 compared to epoch 150. It corresponds to a slight increase in test_rmse. It suggests a potential risk of overfitting.

3. Train MGD : Train_mgd metric reaches its lowest level at epoch 100 and then increases slightly. This is not necessarily a problem unless it is coupled with a deterioration of others test metrics.

Analysis of results after epoch 200 :

- **RMSE, MAE, and MSE :** Metrics are lower for the training set compared to the test dataset.
- **Explained Variance and R^2 :** Metrics are slightly higher for the test set compared to the training dataset. It indicates the model generalizes well.
- **MGD and MPD :** Metrics are higher for the training set compared to the test set. Model's predictions are less accurate on the training dataset. They are more stable on the test dataset. It suggests the model is not over-fitted.

5.1.3.4- Conclusions

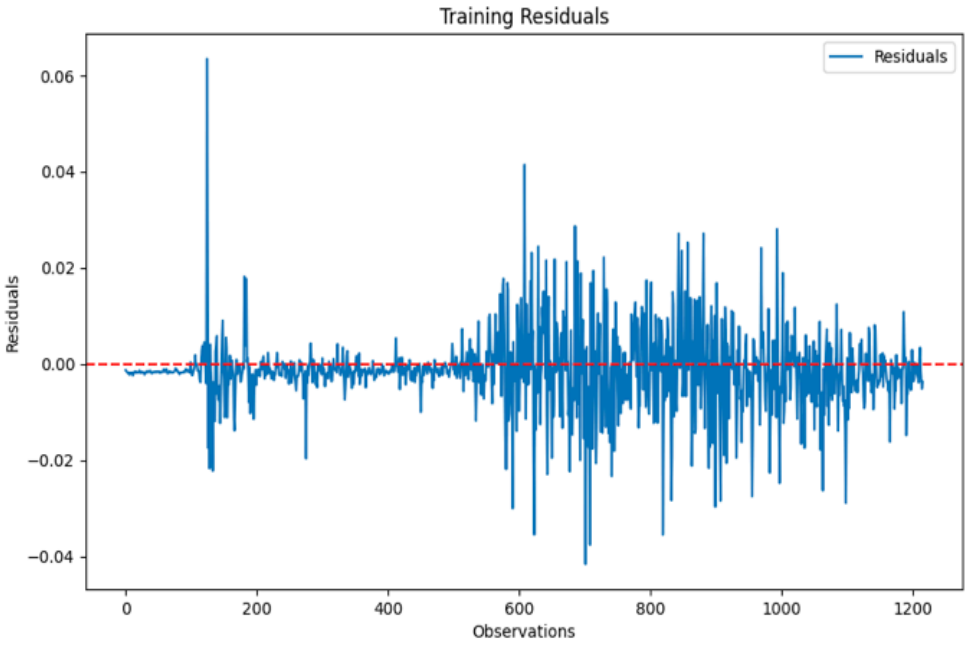
There is a small risk of overfitting. It is indicated by the deterioration of test metrics at epoch 200 compared to epoch 150.
However, slight increases or decreases in the metrics are not significant and do not alter the general trend. Techniques aimed at managing overfitting, such as dropout or early stopping, did not yield relevant results.

5.1.4- Improvements suggestions

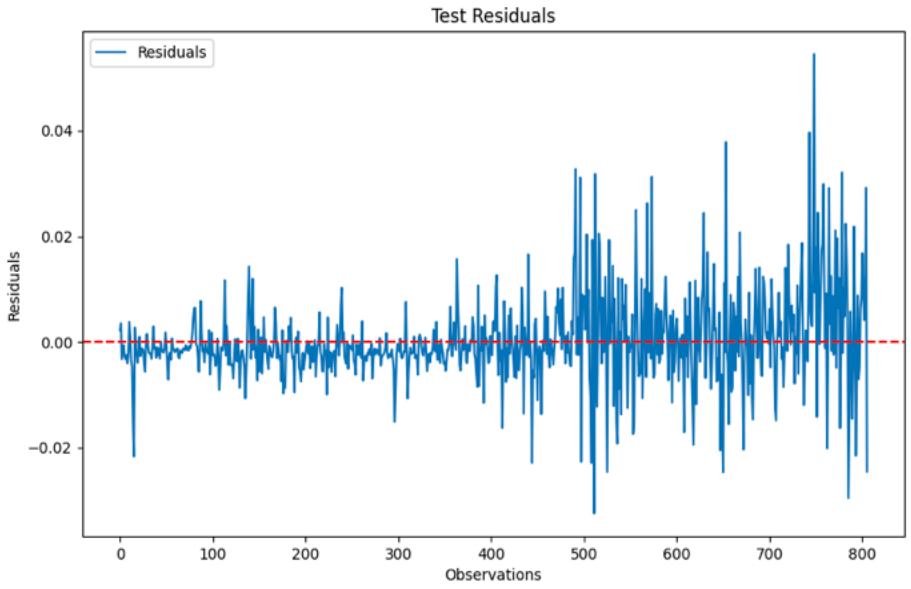
5.1.4.1- Identification of discrepancies in residuals

The curves below represent the training and test residuals from the 3rd training session. They are the differences between the values predicted by a model and the actual observed values. Residuals should be randomly distributed around zero, with no apparent pattern. A small amplitude is preferable.
The 3 training sessions encounter the same problem : They are significant discrepancies at times when we look at the residuals of the training and test phases. *(ex : Between observations 0 and 200 for the training residuals, between observations 500 and 800 for the test residuals).*

Training phase residuals



Test phase residuals



These discrepancies are probably due to volatility : Bitcoin movements are significant. Especially during periods of high volatility. Cryptocurrencies are among the most volatile assets.

5.1.4.2- Improvement paths

There are several possibilities to improve the situation :
1- Making the model more complex *(ex : By increasing the number of neural layers, using others activation functions).*
2- Test the model on a less volatile asset: It won't improve predictions for Bitcoin but testing the model on a less volatile asset would help to determine if the discrepancies are coming from the Bitcoin's volatile nature.

5.1.4.2.1- Modifying the model

In this section, we need to consider the concepts of bias and variance :

- Bias represents the difference between the model's predictions and reality. A simplistic model might ignore certain relationships within the data, leading to underfitting. We could be in this situation because of our model's simplicity.
- Variance measures the model's sensitivity to fluctuations in data. A complex model trained on a small dataset can show high variance. it can lead to overfitting.

When we increase the complexity of a model *(ex : by adding more neurons)*, the variance increases and the bias decreases. If we simplify model, the bias increases and the variance decreases.
In our case, we will increase its complexity. Then we have to pay attention to the risk of variance /overfitting. Metrics like MSE, RMSE, MAE, and R^2 are effective for monitoring this point. We just need to compare their values between the training and test datasets to identify performance differences

Initial model :

```
# Création du modèle :
model = Sequential()
model.add(LSTM(10, input_shape=(None, 1), activation="relu"))
model.add(Dense(1))
model.compile(loss="mean_squared_error", optimizer="adam")
```

5 types of modifications will be tested :

1- Adding neurons and layers to the LSTM neural network :

```
# Création du modèle :
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(None, 1), activation="relu"))
model.add(LSTM(25, activation="relu"))
model.add(Dense(1))
model.compile(loss="mean_squared_error", optimizer="adam")
```

We are increasing the complexity of the model :

- By increasing the number of LSTM neurons.
- By adding news LSTM layers to better capture relationships between variables.

2- Using the « Huber Loss » loss function instead of « Mean Squared Error » :

```
# Création du modèle :
model = Sequential()
model.add(LSTM(10, input_shape=(None, 1), activation="relu"))
model.add(Dense(1))
model.compile(loss=Huber(delta=1.0), optimizer="adam")
```

This loss function:

- Resists to extreme variations. It is particularly useful for Bitcoin.
- Reduces small discrepancies and tolerating larger. It enables better identification of significant relationships between variables.

3- Using GRU neurons instead of LSTM neurons :

GRU (Gated Recurrent Unit) neurons are a variant of LSTMs. They have a simpler architecture with a reduced number of memory gates. A memory gate decides which information is retained,. This allows to the network to maintain long-term memory.

```
# Création du modèle :
model = Sequential()
model.add(GRU( units: 10, activation="relu"))
model.add(Dense(1))
model.compile(loss="mean_squared_error", optimizer="adam")
```

In our case, we are replacing LSTM neurons by GRU neurons. It helps to filter noise to better capture important price movements.

4- Changing the learning rate of the Adam optimizer (0.001, 0.0001, 0.0005) :

```
optimizer = Adam(learning_rate=0.0001)
```

Model captures better significant relationships and large price variations with a lower learning rate. It may improve predictions during periods of high volatility.

5- Adding an attention mechanism to the LSTM neural network :

Attention mechanisms are essential for the LLMs' Transformer models. They allow the models to understand the meaning of a text by weighting the importance of each word. By integrating an attention mechanism into our LSTM neural network, we will help our model to be focused on moments of high volatility *(ex : market crashes)*. It should help us to improve its predictions.

```
# Définition d'une fonction pour ajouter un mécanisme d'attention
def attention_layer(inputs):  1 usage
    query, value = inputs, inputs
    attention = Attention()([query, value])
    return Concatenate()([inputs, attention])


# Création du modèle séquentiel
model = Sequential()
model.add(LSTM(10, return_sequences=True, input_shape=(None, 1), activation="relu"))
model.add(Lambda(lambda x: attention_layer(x)))
model.add(LSTM(5, activation="relu"))
model.add(Dense(1))
model.compile(loss="mean_squared_error", optimizer="adam")
```

5.1.4.2.1.1- Training details

5.1.4.2.1.1.1- Training session 1 : Using 1 LSTM layer / 200 iterations / 32 batch size

- Model :

```
# Création du réseau de neurones :
model = Sequential()
model.add(LSTM(10, input_shape=(None, 1), activation="relu"))
model.add(Dense(1))
model.compile(loss="mean_squared_error", optimizer="adam")
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=200,
    batch_size=32,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Initial dataset : *btc_historic_cotations.csv* (file in github project)
- File : *training_with_one_dimension/price_pred_1_lstm_layer.py*

5.1.4.2.1.1.2- Training session 2 : Using 1 LSTM layer / 300 iterations / 32 batch size

- Model :

```
# Création du réseau de neurones :
model = Sequential()
model.add(LSTM(10, input_shape=(None, 1), activation="relu"))
model.add(Dense(1))
model.compile(loss="mean_squared_error", optimizer="adam")
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=300,
    batch_size=32,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Initial dataset : *btc_historic_cotations.csv* (file in github project)
- File : *training_with_one_dimension/price_pred_1_lstm_layer.py*

5.1.4.2.1.1.3- Training session 3 : Using 1 LSTM layer / 300 iterations / 50 batch size

- Model :

```
# Création du réseau de neurones :
model = Sequential()
model.add(LSTM(10, input_shape=(None, 1), activation="relu"))
model.add(Dense(1))
model.compile(loss="mean_squared_error", optimizer="adam")
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=300,
    batch_size=50,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Initial dataset : *btc_historic_cotations.csv* (file in github project)
- File : *training_with_one_dimension/price_pred_1_lstm_layer.py*

5.1.4.2.1.1.4- Training 4 session : Using 2 LSTM layers / 200 iterations

- Model :

```
# Création du réseau de neurones :
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(None, 1), activation="relu"))
model.add(LSTM(25, activation="relu"))
model.add(Dense(1))
model.compile(loss="mean_squared_error", optimizer="adam")
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=200,
    batch_size=32,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Initial dataset : *btc_historic_cotations.csv* (file in github project)
- File : *training_with_one_dimension/price_pred_2_lstm_layer.py*

5.1.4.2.1.1.5- Training session 5 : Using 3 LSTM layers / 200 iterations

- Model :

```
# Création du réseau de neurones :
model = Sequential()
model.add(LSTM(100, return_sequences=True, input_shape=(None, 1), activation="relu"))
model.add(LSTM(50, return_sequences=True, activation="relu"))
model.add(LSTM(25, activation="relu"))
model.add(Dense(1))
model.compile(loss="mean_squared_error", optimizer="adam")
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=200,
    batch_size=32,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Initial dataset : *btc_historic_cotations.csv* (file in github project)
- File : *training_with_one_dimension/price_pred_3_lstm_layer.py*

5.1.4.2.1.1.6- Training session 6 : Using 1 GRU layer / 200 iterations

- Model :

```
# Création du réseau de neurones :
model = Sequential()
model.add(GRU( units: 10, activation="relu"))
model.add(Dense(1))
model.compile(loss="mean_squared_error", optimizer="adam")
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=200,
    batch_size=32,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Initial dataset : *btc_historic_cotations.csv* (file in github project)
- File : *training_with_one_dimension/price_pred_gru_layer.py*

5.1.4.2.1.1.7- Training session 7 : Using the Huber Loss function / 200 iterations

- Model :

```
# Création du réseau de neurones :
model = Sequential()
model.add(LSTM(10, input_shape=(None, 1), activation="relu"))
model.add(Dense(1))
model.compile(loss=Huber(delta=1.0), optimizer="adam")
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=200,
    batch_size=32,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Initial dataset : *btc_historic_cotations.csv* (file in github project)
- File : *training_with_one_dimension/price_pred_huber_loss.py*

5.1.4.2.1.1.8- Training session 8 : Improving Optimizer (0.001)

- Model :

```
# Création du réseau de neurones :
model = Sequential()
model.add(LSTM(20, input_shape=(None, 1), activation="relu", return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(20, activation="relu"))
model.add(Dropout(0.2))
model.add(Dense(1))
optimizer = Adam(learning_rate=0.001)
model.compile(loss="mean_squared_error", optimizer=optimizer)
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=200,
    batch_size=32,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Initial dataset : *btc_historic_cotations.csv* (file in github project)
- File : *training_with_one_dimension/price_pred_learning_rate_optimizer.py*

5.1.4.2.1.1.9- Training session 9 : Improving Optimizer (0.0005)

- Model :

```
# Création du réseau de neurones :
model = Sequential()
model.add(LSTM(20, input_shape=(None, 1), activation="relu", return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(20, activation="relu"))
model.add(Dropout(0.2))
model.add(Dense(1))
# Compilation du modèle
optimizer = Adam(learning_rate=0.0005)
model.compile(loss="mean_squared_error", optimizer=optimizer)
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=200,
    batch_size=32,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Initial dataset : *btc_historic_cotations.csv* (file in github project)
- File : *training_with_one_dimension/price_pred_learning_rate_optimizer.py*

5.1.4.2.1.1.10- Training session 10 : Improving Optimizer (0.0001)

- Model :

```
# Création du réseau de neurones :
model = Sequential()
model.add(LSTM(20, input_shape=(None, 1), activation="relu", return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(20, activation="relu"))
model.add(Dropout(0.2))
model.add(Dense(1))
# Compilation du modèle
optimizer = Adam(learning_rate=0.0001)
model.compile(loss="mean_squared_error", optimizer=optimizer)
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=200,
    batch_size=32,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Initial dataset : *btc_historic_cotations.csv* (file in github project)
- File : *training_with_one_dimension/price_pred_learning_rate_optimizer.py*

5.1.4.2.1.1.11- Training session 11 : Adding an attention layer / 200 iterations

- Model :

```
def attention_layer(inputs):  1 usage  SydneyJezequel *
    """ Mécanisme d'attention simple """
    query, value = inputs, inputs
    attention = Attention()([query, value])
    return Concatenate()([inputs, attention])

# Création du réseau de neurones :
model = Sequential()
model.add(LSTM(10, return_sequences=True, input_shape=(None, 1), activation="relu"))
model.add(Lambda(lambda x: attention_layer(x)))
model.add(LSTM(5, activation="relu"))
model.add(Dense(1))
model.compile(loss="mean_squared_error", optimizer="adam")
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=200,
    batch_size=32,
    verbose=1,
    callbacks=[metrics_callback]
)
```

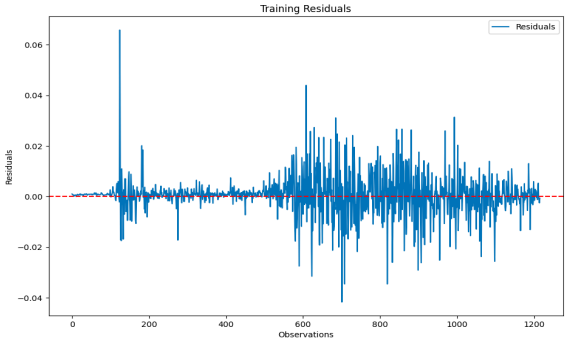
- Initial dataset : *btc_historic_cotations.csv* (file in github project)
- File : *training_with_one_dimension/price_pred_attention_layer.py*

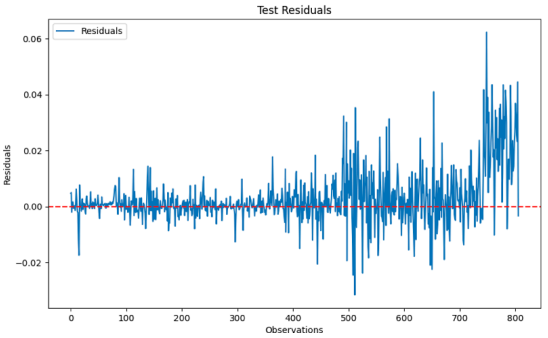
Note about models with many layers :
The progressive decrease in the number of neurons *(ex : 5.1.4.2.1.1.5- Training session 5 : Using 3 LSTM layers / 200 iterations)* allows the organized processing of the dataset's features:
- Earlier layers use more neurons to capture a large number of raw features.
- Deeper layers combine and synthesize these features for predictions.

5.1.4.2.1.2- Training results

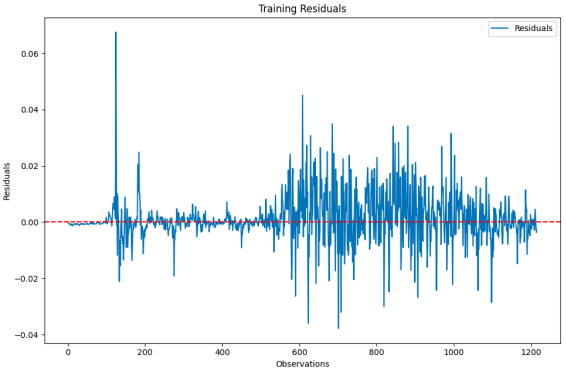
Residuals

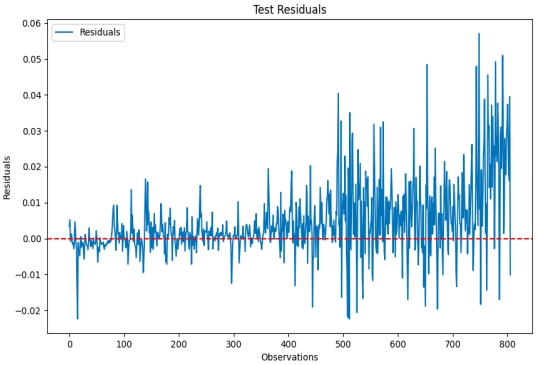
- **Adding a second layer to the LSTM neural network :**



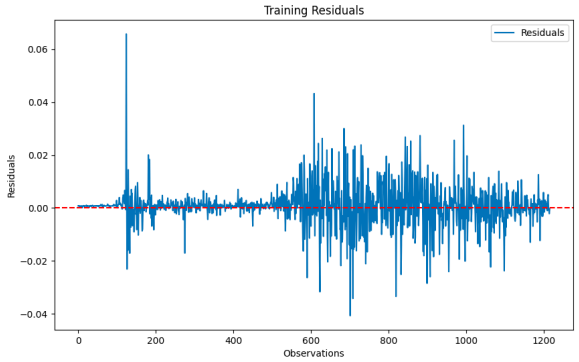


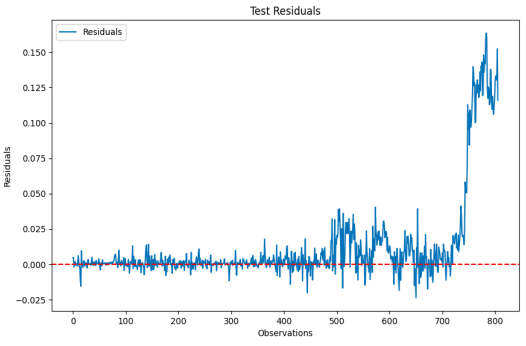
- **Using the Huber Loss loss function :**



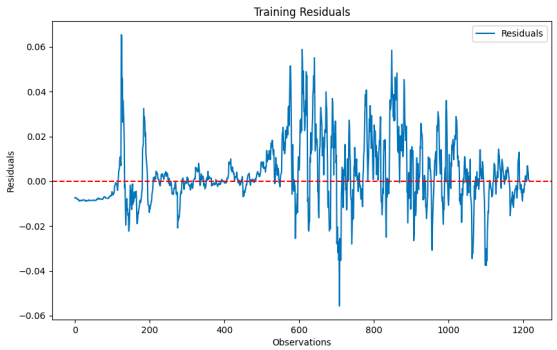


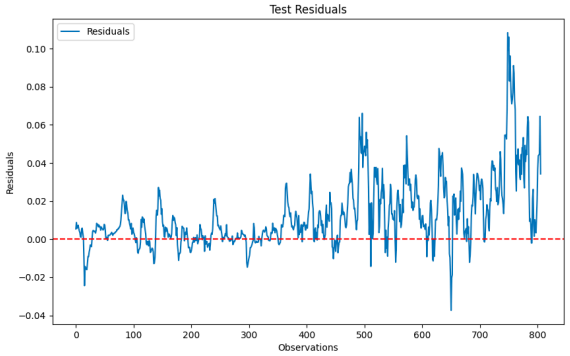
- **Using GRU neurons :**



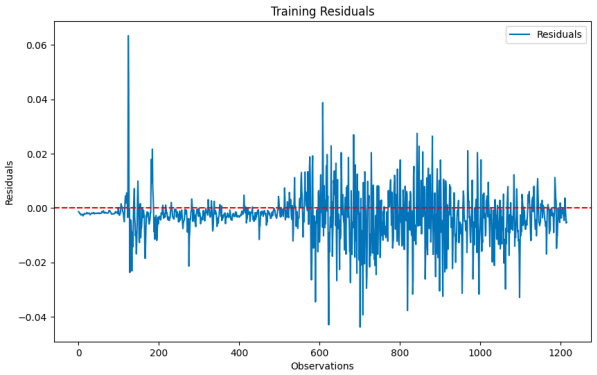


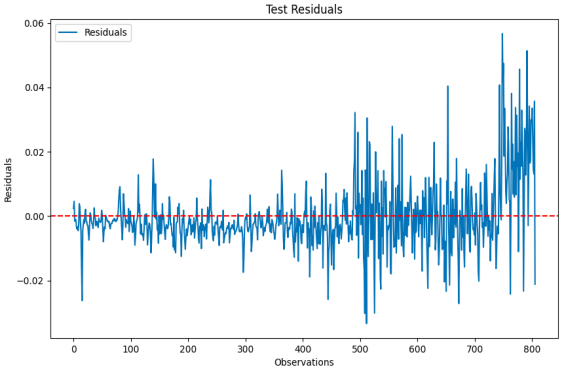
- **Modifying the learning rate of the Adam optimizer (ex : rate at 0.0001) :**





- **Adding an attention mechanism to the LSTM neural network :**





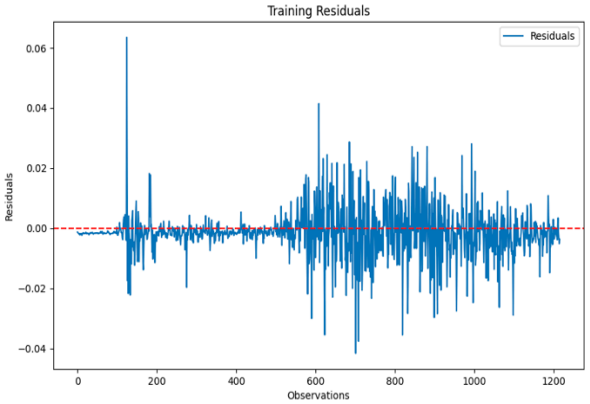
<u>Conclusion :</u>

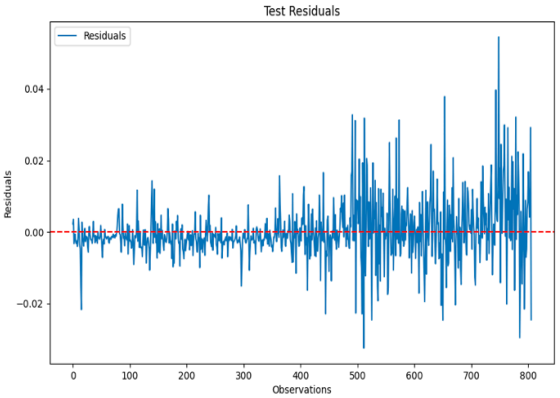
When comparing the residuals from the training sessions, we observe than :

- Test residuals deviate from the mean after observation 700.
- Differences persist between the training and test phases.

Modifications made did not improve the predictions. The 3rd training session offers the best performance.

- **Residuals of training session 3 :**



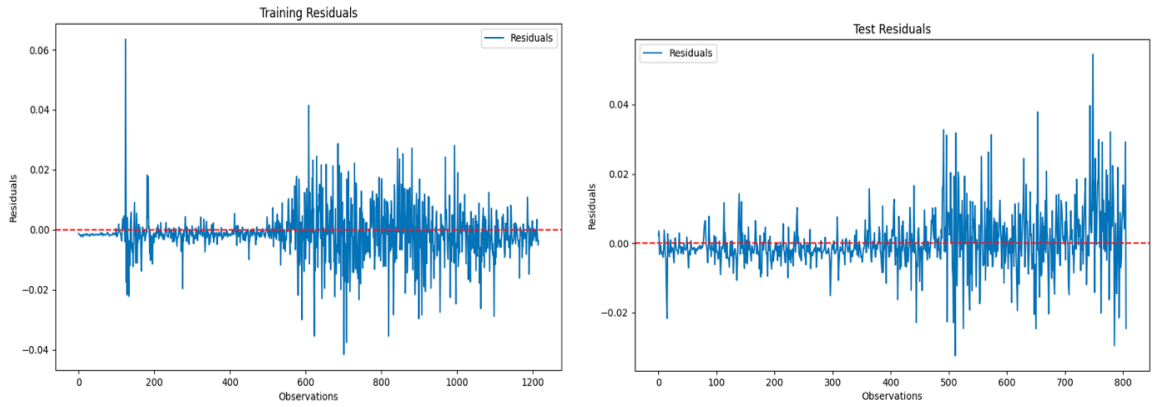


5.1.4.2.2- Testing a less volatile asset

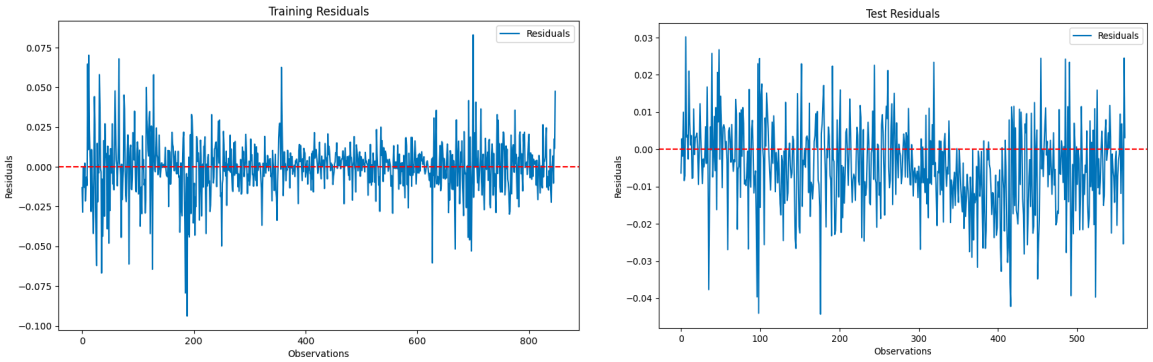
In this section, we'll test the model from the 3rd training session on a less volatile asset : the CAC 40. It is the main index of the Paris stock exchange. CAC 40 is composed of the 40 French companies with the largest market capitalizations. This index shows lower historical volatility than Bitcoin. It is a less speculative and more stable asset due to the diversification of companies.

The data are retrieved from this site: https://fr.investing.com/indices/france-40-historical-data

- **Residuals of the model from the 3rd training session on Bitcoin :**



- **Residuals of the model from the 3rd training session on the CAC 40 :**



<u>Conclusion :</u>

The model's residuals reach more extreme values on the CAC 40 than on Bitcoin during training session *(ex : 0.075 for CAC vs 0.06 for Bitcoin)* but less during the test phase *(ex : 0.03 for CAC 40 vs 0.04 for Bitcoin)*.

Then differences between predicted and actual values do not seem only related to the volatility of Bitcoin. These are discrepancies the model fails to explain.

5.1.4.3- Ranking of models

We will base our models' ranking on their metrics and residuals:

1- Training and test metrics:

- RMSE (Root Mean Squared Error) : The lower it is, better is the performance.
- MSE (Mean Squared Error) : The lower it is, better is the performance.
- MAE (Mean Absolute Error) : The lower it is, better is the performance.
- Explained Variance and R^2 : The closer they are to 1, better is the performance.

2- Residuals:

Residuals should be randomly distributed around zero, with no apparent pattern.
A small amplitude of residuals is preferable.

<u>Ranking of training sessions from best to worst :</u>

1. Training session 3 ***(1 layer, 15 neurons, 300 iterations, batch size: 50)*** **:**

- **Metrics :** Very good performance on training and test metrics, with relatively low RMSE, MSE, and MAE values.
- **Residuals :** The residuals are relatively stable and well distributed.
- This model gives the best balance between precision and stability.

2. Training session 2 ***(1 layer, 10 neurons, 300 iterations, batch size: 32)*** **:**

- **Metrics :** Good performance, with low RMSE, MSE, and MAE values.
- **Residuals :** The residuals are relatively stable and well distributed.
- The model is robust but slightly less performant than training session 3.

3. Training session 1 ***(1 layer, 10 neurons, 200 iterations, batch size: 32)*** **:**

- **Metrics :** Correct performance, but not as good as the previous trainings.
- **Residuals :** The residuals show higher variability than in the previous trainings.
- This model could be enhanced.

4. Training session with the addition of a 2nd layer of LSTM neurons :

- **Metrics :** Correct performance.
- **Residuals :** The residuals show higher variability than in the previous trainings, especially towards the end of observations.
- Adding a layer did not bring the expected improvement. The variability of the residuals increased.

5. Training session with the Huber Loss function :

- **Metrics :** Average performance. RMSE and MSE values are higher than in the previous trainings.
- **Residuals :** The residuals show higher variability than in the previous trainings, especially towards the end of observations.
- Despite its usefulness to manage volatility peaks, the Huber loss function did not improve the situation.

6. Training session with GRU neurons :

- **Metrics :** Average performance, with higher RMSE and MSE values than the previous trainings.
- **Residuals :** The residuals show higher variability than in the previous trainings.
- GRUs did not provide a significant performance gain

7. Training session with the addition of a 3rd layer of LSTM neurons :

- **Metrics :** Average performance, with higher RMSE and MSE values than the previous trainings.
- **Residuals :** The residuals show higher variability than in the previous trainings and apparent patterns, especially in the end of observations.
- Increasing the model's complexity seems leading to the degradation in performance.

8. Training session with a different learning rate for the optimizer *(0.001)* :

- **Metrics :** Average performance, with higher RMSE and MSE values than the previous trainings.
- **Residuals :** The residuals show higher variability than in the previous trainings and apparent patterns.

9. Training session with a different learning rate for the optimizer *(0.0005)* :

- **Metrics :** Average performance, with higher RMSE and MSE values than the previous trainings.
- **Residuals :** The residuals show higher variability than in the previous trainings and apparent patterns.
- Modifications in the Adam optimizer's learning rate led to degraded performance.

10. Training session with a different learning rate for the optimizer *(0.0001)* :

- **Metrics :** Average performance, with higher RMSE and MSE values than the previous trainings.
- **Residuals :** The residuals show higher variability than in the previous trainings and apparent patterns.

11. Training session with the addition of attention mechanism in model :

- **Metrics :** Average performance, with higher RMSE and MSE values than the previous trainings.
- **Residuals :** The residuals show higher variability than in the previous trainings and apparent patterns.
- Integrating an attention mechanism did not improve predictions. It even degraded the distribution of residuals.

Conclusion:

Increasing the model's complexity and tested modifications seem to increase the model's variance.

5.2- Training with a multiple dimensions' dataset

5.2.1- Dataset enrichment

I enriched the dataset with elements listed in section *3.1- Enriching the dataset* :

- **Dernier (last)**
- **MA_150**
- **MA_100**
- **MA_50**
- **RSI**
- **MA_50_above_MA_150**
- **MA_100_above_MA_150**
- **MA_50_above_MA_100**
- **Lags features**
- **Historical_Volatility**

The « Dernier » (last) dimension corresponds to the BTC closing price. « Dernier » means « last » for last daily price in French. This dimension is used in training sessions with a single dimension.

The others dimensions are calculated from this one :

- **MA_150 :** 150-days moving average of the BTC closing price.
- **MA_100 :** 100-days moving average of the BTC closing price.
- **MA_50 :** 50-days moving average of the BTC closing price.
- **RSI :** The RSI indicator shows the momentum (the strength of a trend). It can also indicate reversal zones (overbought, oversold).
- **MA_50_above_MA_150 :** Situation where the 50-days moving average has crossed the 150-day moving average and is above.
- **MA_100_above_MA_150 :** Situation where the 100-days moving average has crossed the 150-day moving average and is above.
- **MA_50_above_MA_100 :** Situation where the 50-days moving average has crossed the 100-day moving average and is above.
- **Lag features :** This is the adding of the past price values from X periods into each row of the dataset. These lag features help the model to better identify temporal relationships.
- **Historical Volatility :** It measures the dispersion of Bitcoin's returns. It helps the model to identify volatility patterns which precede certain price movements.

Additional dimensions are calculated and added to the dataset before the training begins. All dimensions are not added to each training session. Several combinations of data were tested to obtain the best result.

Dataset overview :

```
Dernier,MA_150,MA_100,MA_50,RSI,MA_50_superior_MA_150,MA_100_superior_MA_150,MA_50_superior_MA_100,Lag_1,Lag_7,Historical_Volatility
0.0036534146210219563,536.9173333333333,519.5980000000001,566.99,51.461961183144524,1,0,1,602.2,597.0,
0.0036807916298099755,552.1913333333333,558.2639999999999,616.65,40.98658147596265,1,1,1,601.9,627.5,
0.003571283594657896,550.872,566.974,608.878,43.18067680201051,1,1,1,591.7,595.0,
0.0036081009513038533,550.1940000000001,572.725,608.308,45.85840439076001,1,1,1,589.8,589.5,
0.0036364219948776676,550.108,574.2270000000001,608.22,47.657380263581736,1,1,1,592.8,598.8,
0.0025857112782891 83,547.5333333333333,584.234,547.932,38.84232427318708,1,1,0,483.6,506.0,
0.0025797983730504502,548.208,582.969,545.1080000000001,39.50366022574757,0,1,0,484.5,481.8,
0.0024186171212036826,547.8833333333333,567.29,521.414,33.16008008374236,0,1,0,477.7,474.9,
0.0020174023372413236,547.1613333333333,563.145,516.114,20.97140790221846,0,1,0,461.1,478.5,
0.0018267073105109784,544.994,554.956,501.03399999999993,24.805236895644953,0,1,0,401.6,477.7,
0.0017785615364355 95,543.9393333333334,545.7919999999999,483.3639999999999,31.671862040613433,0,1,0,404.5,411.5,
0.0016256279011369014,542.1959999999999,537.235,467.398,31.566113289529866,0,0,0,388.2,423.8,
0.0010601510644464227,539.8366666666667,527.7810000000001,454.026,20.85616239889633,0,0,0,335.3,379.1,
```

Normalization, mentioned in section *3.1.2 - Feature engineering*, was not applied to the moving averages. Its utilization degraded the results during the test phase because moving averages are less subject to volatility than price.

5.2.2- New model modifications

During this phase, we have also tested several modifications to the model. This is a list of the main changes :

- **Use of Bidirectional LSTM Layers :** These layers process data sequences in both directions. It is useful for a more refined analysis. As data goes in both directions,, we needed to slightly increase the dropout rate to reduce the risk of overfitting *(ex : 20% for classic LSTM layers VS 30% for bidirectional layers)*.
- **Use of L2 Regularizer :** This component manages overfitting by penalizing too large model weights. In our tests, it was applied to the network's dense layers. It stabilizes the synthesis of features before the final prediction.
- **Use of Batch Normalization :** This technique is used to stabilize the performance of a multi-layers network. It keeps neuron activations in a reduced range of values to manage the risk of vanishing or exploding gradients.
- **Use of the Tanh Activation Function :** The Tanh function provides the representation of negative values. It is not the case of the ReLU function. This function is able to model sequences where relationships are positive or negative, which is adapted for modeling the price of a financial asset.

5.2.3- Select the most interesting models

5.2.3.1- Comparative table

Trainings	test_explained_variance	test_mae	test_mgd	test_mpd	test_mse	test_r2	test_rmse	train_explained_variance	train_mae	train_mgd	train_mpd	train_mse	train_r2	train_rmse
Training 1	-2.22e-16	0.044	0.341	0.035	0.004	-0.764	0.061	0.001	0.033	0.246	0.018	0.001	0.001	0.037
Training 2	-1.226	3.124	NaN	NaN	11.345	-14.931	3.368	-3.759	1.922	NaN	NaN	5.732	-10.992	2.394
Training 3	0.011	0.124	NaN	NaN	0.029	-0.789	0.169	0.097	0.086	NaN	NaN	0.010	0.096	0.098
Training 4	1.11e-16	0.206	0.219	0.131	0.081	-0.757	0.285	-2.22e-16	0.157	0.136	0.064	0.031	-0.000	0.176
Training 5	-116.629	20.690	NaN	NaN	783.520	-236.949	27.991	-62.427	9.013	NaN	NaN	208.733	-93.443	14.448
Training 6	-79.005	106.363	NaN	NaN	16043.910	-270.239	126.665	-162.506	77.061	NaN	NaN	10794.246	-270.880	103.895
Training 7	0.241	0.067	0.072	0.026	0.010	0.123	0.099	0.006	0.082	0.096	0.028	0.009	-0.146	0.092
Training 8	0.000	0.207	0.221	0.132	0.081	-0.765	0.285	-2.22e-16	0.157	0.137	0.064	0.031	-0.000	0.176
Training 9	-50754.434	28715.850	NaN	NaN	520643773.64	-58500.260	59335.013	-341657.116	51424.595	NaN	NaN	13820994993.088	-342157.108	117562.728
Training 10	0.000	7.30e-06	1.75e-09	4.22e-10	1.02e-10	-0.760	1.01e-05	0.000	5.56e-06	6.69e-10	1.61e-10	3.87e-11	-8.83e-07	6.22e-06
Training 11	0.000	0.206	0.220	0.132	0.081	-0.760	0.285	-2.22e-16	0.157	0.137	0.064	0.031	-3.52e-07	0.176
Training 12	0.072	0.106	4.138	0.193	0.021	-0.553	0.143	0.478	0.054	NaN	NaN	0.005	0.478	0.068
Training 13	-0.084	0.108	0.754	0.093	0.018	-0.218	0.133	0.811	0.025	NaN	NaN	0.002	0.803	0.044
Training 14	-0.193	0.119	1.177	0.139	0.022	-0.640	0.150	0.942	0.015	NaN	NaN	0.001	0.942	0.023
Training 15	0.000	0.011	0.002	0.001	0.000	-0.349	0.015	-0.001	0.010	0.001	0.000	0.000	-0.119	0.011
Training 16	0.761	0.052	0.099	0.021	0.005	0.528	0.070	0.764	0.030	0.079	0.010	0.002	0.761	0.041
Training 17	-0.920	0.576	NaN	NaN	0.418	-1.771	0.647	-1.387	0.584	NaN	NaN	0.420	-3.144	0.648
Training 18	-239952.182	94186.559	NaN	NaN	'604577180.9	-273895.567	132682.241	-555144.197	106264.556	NaN	NaN	24167022201.671	-560180.190	155457.461
Training 19	0.000	0.162	0.372	0.124	0.043	-1.543	0.208	-0.017	0.095	0.154	0.046	0.014	-0.218	0.118
Training 20	-0.536	0.157	2.401	0.242	0.036	-1.117	0.191	0.960	0.013	NaN	NaN	0.000	0.960	0.022

5.2.3.2- Training characteristics

To edit dataset in Github project :
If you wish to modify dataset used by models, edit the method *prepare_many_dimensions_dataset()* in the class *PrepareDatasetService*. Methods which generate combination of technical indicators are commented in it :

```
def prepare_many_dimensions_dataset(self, dataset, cutoff_date='2020-01-01', lags=None, add_volatility=False):
    """ Preparation of the multi-dimensional dataset before training """
    # Preparation of the dataset :
    tmp_dataset = self.format_dataset(dataset)
    tmp_dataset = self.delete_columns(tmp_dataset)
    # Saving the formatted dataset :
    self.save_tmp_dataset(tmp_dataset)
    # Adding technical indicators to the dataset :
    tmp_dataset = self.add_technicals_indicators(tmp_dataset)
    """ tmp_dataset = self.add_technicals_indicators_sma_rsi_smasignal(tmp_dataset) """
    """ tmp_dataset = self.add_technicals_indicators_sma(tmp_dataset) """
    """ tmp_dataset = self.add_technicals_indicators_rsi(tmp_dataset) """
    """ tmp_dataset = self.add_technicals_indicators_sma_signal(tmp_dataset) """
    # Displaying the entire dataset before price transformation :
```

5.2.3.2.1- Training session 1 : Using 1 LSTM layer / 400 iterations

- Model :

```
# Création du réseau de neurones :
model = Sequential()
model.add(LSTM(10, input_shape=(nb_timesteps, nb_features), activation="relu"))
model.add(Dense(1))
model.compile(loss="mean_squared_error", optimizer="adam")
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=400,
    batch_size=32,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Dataset : *Including closing price, moving averages, rsi, bullish crossover of moving averages :*

```
Dernier,MA_150,MA_100,MA_50,RSI,MA_50_supérieure_MA_150,MA_100_supérieure_MA_150,MA_50_supérieure_MA_100
0.0036534146210219563,536.9173333333333,519.5980000000001,566.99,51.461961183144524,1,0,1
0.0036807916298099755,552.1913333333333,558.2639999999999,616.65,40.98658147596265,1,1,1
0.003571283594657896,550.872,566.974,608.878,43.18067680201051,1,1,1
0.0036081009513038533,550.1940000000001,572.725,608.308,45.85840439076001,1,1,1
0.0036364219948776676,550.108,574.2270000000001,608.22,47.657380263581736,1,1,1
0.002585711278289183,547.5333333333333,584.234,547.932,38.84232427318708,1,1,0
0.002597983730504502,548.208,582.969,545.1080000000001,39.50366022574757,0,1,0
0.0024186171212036826,547.8833333333333,567.29,521.414,33.16008008374236,0,1,0
0.0020174023372413236,547.1613333333333,563.145,516.114,20.97140790221846,0,1,0
```

- File : *training_with_many_dimensions/price_pred_1_lstm_layer.py*

5.2.3.2.2- Training session 2 : Using 2 LSTM layers / 400 iterations

- Model :

```
# Création du réseau de neurones :
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(nb_timesteps, nb_features), activation="relu"))
model.add(LSTM(25, activation="relu"))
model.add(Dense(1))
model.compile(loss="mean_squared_error", optimizer="adam")
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=400,
    batch_size=32,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Dataset : *Including closing price, moving averages, rsi, bullish crossover of moving averages :*

```
Dernier,MA_150,MA_100,MA_50,RSI,MA_50_supérieure_MA_150,MA_100_supérieure_MA_150,MA_50_supérieure_MA_100
0.0036534146210219563,536.9173333333333,519.5980000000001,566.99,51.461961183144524,1,0,1
0.0036807916298099755,552.1913333333333,558.2639999999999,616.65,40.98658147596265,1,1,1
0.003571283594657896,550.872,566.974,608.878,43.18067680201051,1,1,1
0.0036081009513038533,550.1940000000001,572.725,608.308,45.85840439076001,1,1,1
0.0036364219948776676,550.108,574.2270000000001,608.22,47.657380263581736,1,1,1
0.0025857112782891 83,547.5333333333333,584.234,547.932,38.84232427318708,1,1,0
0.0025979837305045 02,548.208,582.969,545.1080000000001,39.50366022574757,0,1,0
0.0024186171212036826,547.8833333333333,567.29,521.414,33.16008008374236,0,1,0
0.0020174023372413236,547.1613333333333,563.145,516.114,20.97140790221846,0,1,0
```

- File : *training_with_many_dimensions/price_pred_2_lstm_layers.py*

5.2.3.2.3- Training session 3 : Using 2 LSTM layers / 650 iterations

- Model :

```
# Création du réseau de neurones :
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(nb_timesteps, nb_features), activation="relu"))
model.add(LSTM(25, activation="relu"))
model.add(Dense(1))
model.compile(loss="mean_squared_error", optimizer="adam")
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=650,
    batch_size=32,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Dataset : *Including closing price, moving averages, rsi, bullish crossover of moving averages :*

```
Dernier,MA_150,MA_100,MA_50,RSI,MA_50_supérieure_MA_150,MA_100_supérieure_MA_150,MA_50_supérieure_MA_100
0.0036534146210219563,536.9173333333333,519.5980000000001,566.99,51.461961183144524,1,0,1
0.0036807916298099755,552.1913333333333,558.2639999999999,616.65,40.98658147596265,1,1,1
0.003571283594657896,550.872,566.974,608.878,43.18067680201051,1,1,1
0.0036081009513038533,550.1940000000001,572.725,608.308,45.85840439076001,1,1,1
0.0036364219948776676,550.108,574.2270000000001,608.22,47.657380263581736,1,1,1
0.0025857112782891 83,547.5333333333333,584.234,547.932,38.84232427318708,1,1,0
0.002597983730504502,548.208,582.969,545.1080000000001,39.50366022574757,0,1,0
0.0024186171212036826,547.8833333333333,567.29,521.414,33.16008008374236,0,1,0
0.0020174023372413236,547.1613333333333,563.145,516.114,20.97140790221846,0,1,0
```

- File : *training_with_many_dimensions/price_pred_2_lstm_layers.py*

5.2.3.2.4- Training session 4 : Using 3 LSTM layers / 800 iterations

- Model :

```
# Création du réseau de neurones :
model = Sequential()
model.add(LSTM(100, return_sequences=True, input_shape=(nb_timesteps, nb_features), activation="relu"))
model.add(LSTM(50, return_sequences=True, activation="relu"))
model.add(LSTM(25, activation="relu"))
model.add(Dense(1))
model.compile(loss="mean_squared_error", optimizer="adam")
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=800,
    batch_size=32,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Dataset : *Including closing price, moving averages, rsi, bullish crossover of moving averages :*

```
Dernier,MA_150,MA_100,MA_50,RSI,MA_50_supérieure_MA_150,MA_100_supérieure_MA_150,MA_50_supérieure_MA_100
0.0036534146210219563,536.9173333333333,519.5980000000001,566.99,51.461961183144524,1,0,1
0.0036807916298099755,552.1913333333333,558.2639999999999,616.65,40.98658147596265,1,1,1
0.003571283594657896,550.872,566.974,608.878,43.18067680201051,1,1,1
0.0036081009513038533,550.1940000000001,572.725,608.308,45.85840439076001,1,1,1
0.0036364219948776676,550.108,574.2270000000001,608.22,47.657380263581736,1,1,1
0.002585711278289183,547.5333333333333,584.234,547.932,38.84232427318708,1,1,0
0.002597983730504502,548.208,582.969,545.1080000000001,39.50366022574757,0,1,0
0.0024186171212036826,547.8833333333333,567.29,521.414,33.16008008374236,0,1,0
0.0020174023372413236,547.1613333333333,563.145,516.114,20.97140790221846,0,1,0
```

- File : *training_with_many_dimensions/price_pred_3_lstm_layers.py*

5.2.3.2.5- Training session 5 : Using an attention layer / 800 iterations

- Model :

```
def attention_layer(inputs):  1 usage   SydneyJezequel *
    """ Mécanisme d'attention simple """
    query, value = inputs, inputs
    attention = Attention()([query, value])
    return Concatenate()([inputs, attention])

# Création du réseau de neurones :
model = Sequential()
model.add(LSTM(10, return_sequences=True, input_shape=(nb_timesteps, nb_features), activation="relu"))
model.add(Lambda(lambda x: attention_layer(x)))
model.add(LSTM(5, activation="relu"))
model.add(Dense(1))
model.compile(loss="mean_squared_error", optimizer="adam")
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=800,
    batch_size=32,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Dataset : *Including closing price, moving averages, rsi, bullish crossover of moving averages :*

```
Dernier,MA_150,MA_100,MA_50,RSI,MA_50_supérieure_MA_150,MA_100_supérieure_MA_150,MA_50_supérieure_MA_100
0.0036534146210219563,536.9173333333333,519.5980000000001,566.99,51.461961183144524,1,0,1
0.0036807916298099755,552.1913333333333,558.2639999999999,616.65,40.98658147596265,1,1,1
0.003571283594657896,550.872,566.974,608.878,43.18067680201051,1,1,1
0.0036081009513038533,550.1940000000001,572.725,608.308,45.85840439076001,1,1,1
0.0036364219948776676,550.108,574.2270000000001,608.22,47.657380263581736,1,1,1
0.002585711278289183,547.5333333333333,584.234,547.932,38.84232427318708,1,1,0
0.002597983730504502,548.208,582.969,545.1080000000001,39.50366022574757,0,1,0
0.0024186171212036826,547.8833333333333,567.29,521.414,33.16008008374236,0,1,0
0.0020174023372413236,547.1613333333333,563.145,516.114,20.97140790221846,0,1,0
```

- File : *training_with_many_dimensions/price_pred_attention_layer.py*

5.2.3.2.6- Training session 6 : Using 1 GRU layer / 400 iterations

- Model :

```
# Création du réseau de neurones :
model = Sequential()
model.add(GRU( units: 10, activation="relu"))
model.add(Dense(1))
model.compile(loss="mean_squared_error", optimizer="adam")
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=400,
    batch_size=32,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Dataset : *Including closing price, moving averages, rsi, bullish crossover of moving averages :*

```
Dernier,MA_150,MA_100,MA_50,RSI,MA_50_supérieure_MA_150,MA_100_supérieure_MA_150,MA_50_supérieure_MA_100
0.0036534146210219563,536.9173333333333,519.5980000000001,566.99,51.461961183144524,1,0,1
0.0036807916298099755,552.1913333333333,558.2639999999999,616.65,40.98658147596265,1,1,1
0.003571283594657896,550.872,566.974,608.878,43.18067680201051,1,1,1
0.0036081009513038533,550.1940000000001,572.725,608.308,45.85840439076001,1,1,1
0.0036364219948776676,550.108,574.2270000000001,608.22,47.657380263581736,1,1,1
0.002585711278289183,547.5333333333333,584.234,547.932,38.84232427318708,1,1,0
0.002597983730504502,548.208,582.969,545.1080000000001,39.50366022574757,0,1,0
0.0024186171212036826,547.8833333333333,567.29,521.414,33.16008008374236,0,1,0
0.0020174023372413236,547.1613333333333,563.145,516.114,20.97140790221846,0,1,0
```

- File : *training_with_many_dimensions/price_pred_gru_layer.py*

5.2.3.2.7- Training session 7 : Using 1 GRU layer / 800 iterations

- Model :

```
# Création du réseau de neurones :
model = Sequential()
model.add(GRU( units: 10, activation="relu"))
model.add(Dense(1))
model.compile(loss="mean_squared_error", optimizer="adam")
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=800,
    batch_size=32,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Dataset : *Including closing price, moving averages, rsi, bullish crossover of moving averages :*

```
Dernier,MA_150,MA_100,MA_50,RSI,MA_50_supérieure_MA_150,MA_100_supérieure_MA_150,MA_50_supérieure_MA_100
0.0036534146210219563,536.9173333333333,519.5980000000001,566.99,51.461961183144524,1,0,1
0.0036807916298099755,552.1913333333333,558.2639999999999,616.65,40.98658147596265,1,1,1
0.003571283594657896,550.872,566.974,608.878,43.18067680201051,1,1,1
0.0036081009513038533,550.1940000000001,572.725,608.308,45.85840439076001,1,1,1
0.0036364219948776676,550.108,574.2270000000001,608.22,47.657380263581736,1,1,1
0.002585711278289183,547.5333333333333,584.234,547.932,38.84232427318708,1,1,0
0.002597983730504502,548.208,582.969,545.1080000000001,39.50366022574757,0,1,0
0.0024186171212036826,547.8833333333333,567.29,521.414,33.16008008374236,0,1,0
0.0020174023372413236,547.1613333333333,563.145,516.114,20.97140790221846,0,1,0
```

- File : *training_with_many_dimensions/price_pred_gru_layer.py*

5.2.3.2.8- Training session 8 : Using the Huber Loss function / 400 iterations

- Model :

```
# Création du réseau de neurones :
model = Sequential()
model.add(LSTM(10, input_shape=(nb_timesteps, nb_features), activation="relu"))
model.add(Dense(1))
model.compile(loss=Huber(delta=1.0), optimizer="adam")
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=400,
    batch_size=32,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Dataset : *Including closing price, moving averages, rsi, bullish crossover of moving averages :*

```
Dernier,MA_150,MA_100,MA_50,RSI,MA_50_supérieure_MA_150,MA_100_supérieure_MA_150,MA_50_supérieure_MA_100
0.0036534146210219563,536.9173333333333,519.5980000000001,566.99,51.461961183144524,1,0,1
0.0036807916298099755,552.1913333333333,558.2639999999999,616.65,40.98658147596265,1,1,1
0.003571283594657896,550.872,566.974,608.878,43.18067680201051,1,1,1
0.0036081009513038533,550.1940000000001,572.725,608.308,45.85840439076001,1,1,1
0.0036364219948776676,550.108,574.2270000000001,608.22,47.657380263581736,1,1,1
0.002585711278289183,547.5333333333333,584.234,547.932,38.84232427318708,1,1,0
0.002597983730504502,548.208,582.969,545.1080000000001,39.50366022574757,0,1,0
0.0024186171212036826,547.8833333333333,567.29,521.414,33.16008008374236,0,1,0
0.0020174023372413236,547.1613333333333,563.145,516.114,20.97140790221846,0,1,0
```

- File : *training_with_many_dimensions/price_pred_huber_loss.py*

5.2.3.2.9- Training session 9 : Improving Optimizer (0.0001) / 400 iterations

- Model :

```
# Création du réseau de neurones :
model = Sequential()
model.add(LSTM(20, input_shape=(nb_timesteps, nb_features), activation="relu", return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(20, activation="relu"))
model.add(Dropout(0.2))
model.add(Dense(1))
optimizer = Adam(learning_rate=0.0001)
model.compile(loss="mean_squared_error", optimizer=optimizer)
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=400,
    batch_size=32,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Dataset : *Including closing price, moving averages, rsi, bullish crossover of moving averages :*

```
Dernier,MA_150,MA_100,MA_50,RSI,MA_50_supérieure_MA_150,MA_100_supérieure_MA_150,MA_50_supérieure_MA_100
0.0036534146210219563,536.9173333333333,519.5980000000001,566.99,51.461961183144524,1,0,1
0.0036807916298099755,552.1913333333333,558.2639999999999,616.65,40.98658147596265,1,1,1
0.003571283594657896,550.872,566.974,608.878,43.18067680201051,1,1,1
0.0036081009513038533,550.1940000000001,572.725,608.308,45.85840439076001,1,1,1
0.0036364219948776676,550.108,574.2270000000001,608.22,47.657380263581736,1,1,1
0.002585711278289183,547.5333333333333,584.234,547.932,38.84232427318708,1,1,0
0.002597983730504502,548.208,582.969,545.1080000000001,39.50366022574757,0,1,0
0.0024186171212036826,547.8833333333333,567.29,521.414,33.16008008374236,0,1,0
0.0020174023372413236,547.1613333333333,563.145,516.114,20.97140790221846,0,1,0
```

- File : *training_with_many_dimensions/price_pred_learning_rate_optimizer.py*

5.2.3.2.10- Training session 10 : Adding lags features and historical volatility / 400 iterations

- Model :

```
# Création du réseau de neurones :
model = Sequential()
model.add(LSTM(10, input_shape=(nb_timesteps, nb_features), activation="relu"))
model.add(Dense(1))
model.compile(loss="mean_squared_error", optimizer="adam")
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=400,
    batch_size=32,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Dataset : *Including closing price, moving averages, rsi, bullish crossover of moving averages, lags features, historical volatility :*

```
Dernier,MA_150,MA_100,MA_50,RSI,MA_50_superior_MA_150,MA_100_superior_MA_150,MA_50_superior_MA_100,Historical_Volatility,Lag_1,Lag_7
0.0036534146210219563,536.9173333333333,519.5980000000001,566.99,51.461961183144524,1,0,1,0.6607824657304868,0.0036534146210219563,0.0036534146210219563
0.0036807916298099755,552.1913333333333,558.2639999999999,616.65,40.98658147596265,1,1,1,0.6607824657304868,0.0036534146210219563,0.0036534146210219563
0.0035712835946578 96,550.872,566.974,608.878,43.18067680201051,1,1,1,0.6607824657304868,0.0036807916298099755,0.0036534146210219563
0.0036081009513038533,550.1940000000001,572.725,608.308,45.85840439076001,1,1,1,0.6607824657304868,0.0035712835946578 96,0.0036534146210219563
0.0036364219948776676,550.108,574.2270000000001,608.22,47.657380263581736,1,1,1,0.6607824657304868,0.0036081009513038533,0.0036534146210219563
0.0025857112782891 83,547.5333333333333,584.234,547.932,38.84232427318708,1,1,0,0.6607824657304868,0.0036364219948776676,0.0036534146210219563
0.0025979837305045 02,548.208,582.969,545.1080000000001,39.50366022574757,0,1,0,0.6607824657304868,0.0025857112782891 83,0.0036534146210219563
0.0024186171212036826,547.8833333333333,567.29,521.414,33.16008008374236,0,1,0,0.6607824657304868,0.0025979837305045 02,0.0036534146210219563
0.0020174023372413236,547.1613333333333,563.145,516.114,20.97140790221846,0,1,0,0.6607824657304868,0.0024186171212036826,0.0036807916298099755
```

- File : *training_with_many_dimensions/price_pred_lags_features_and_vol.py*

5.2.3.2.11- Training session 11 : Adding lags features / 400 iterations

- Model :

```
# Création du réseau de neurones :
model = Sequential()
model.add(LSTM(10, input_shape=(nb_timesteps, nb_features), activation="relu"))
model.add(Dense(1))
model.compile(loss="mean_squared_error", optimizer="adam")
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=400,
    batch_size=32,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Dataset : *Including closing price, moving averages, rsi, bullish crossover of moving averages, lags features :*

```
Dernier,MA_150,MA_100,MA_50,RSI,MA_50_superior_MA_150,MA_100_superior_MA_150,MA_50_superior_MA_100,Lag_1,Lag_7
0.0036534146210219563,536.9173333333333,519.5980000000001,566.99,51.461961183144524,1,0,1,0.0036534146210219563,0.0036534146210219563
0.0036807916298099755,552.1913333333333,558.2639999999999,616.65,40.98658147596265,1,1,1,0.0036534146210219563,0.0036534146210219563
0.003571283594657896,550.872,566.974,608.878,43.18067680201051,1,1,1,0.0036807916298099755,0.0036534146210219563
0.0036081009513038533,550.1940000000001,572.725,608.308,45.85840439076001,1,1,1,0.003571283594657896,0.0036534146210219563
0.0036364219948776676,550.108,574.2270000000001,608.22,47.657380263581736,1,1,1,0.0036081009513038533,0.0036534146210219563
0.0025857112789289183,547.5333333333333,584.234,547.932,38.84232427318708,1,1,0,0.0036364219948776676,0.0036534146210219563
0.0025797983730504502,548.208,582.969,545.1080000000001,39.50366022574757,0,1,0,0.0025857112789289183,0.0036534146210219563
0.0024186171212036826,547.8833333333333,567.29,521.414,33.16008008374236,0,1,0,0.0025797983730504502,0.0036534146210219563
0.0020174023372413236,547.1613333333333,563.145,516.114,20.97140790221846,0,1,0,0.0024186171212036826,0.0036807916298099755
```

- File : *training_with_many_dimensions/price_pred_lags_features.py*

5.2.3.2.12- Training session 12 : Using 3 LSTM layers (batch normalization, optimized rate, dropout, tanh activation function) / 800 iterations

- Model :

```
# Création du réseau de neurones
model = Sequential()
model.add(LSTM(100, return_sequences=True, input_shape=(nb_timesteps, nb_features), activation="tanh"))
model.add(BatchNormalization())
model.add(Dropout(0.2))

model.add(LSTM(50, return_sequences=True, activation="tanh"))
model.add(BatchNormalization())
model.add(Dropout(0.2))

model.add(LSTM(25, activation="tanh"))
model.add(BatchNormalization())
model.add(Dropout(0.2))

model.add(Dense(1))

# Compilation du modèle
optimizer = Adam(learning_rate=0.001)
model.compile(loss="mean_squared_error", optimizer=optimizer)
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=800,
    batch_size=32,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Dataset : *Including closing price, moving averages, rsi, bullish crossover of moving averages :*

```
Dernier,MA_150,MA_100,MA_50,RSI,MA_50_supérieure_MA_150,MA_100_supérieure_MA_150,MA_50_supérieure_MA_100
0.0036534146210219563,536.9173333333333,519.5980000000001,566.99,51.461961183144524,1,0,1
0.0036807916298099755,552.1913333333333,558.2639999999999,616.65,40.98658147596265,1,1,1
0.003571283594657896,550.872,566.974,608.878,43.18067680201051,1,1,1
0.0036081009513038533,550.1940000000001,572.725,608.308,45.85840439076001,1,1,1
0.0036364219948776676,550.108,574.2270000000001,608.22,47.657380263581736,1,1,1
0.002585711278289183,547.5333333333333,584.234,547.932,38.84232427318708,1,1,0
0.002597983730504502,548.208,582.969,545.1080000000001,39.50366022574757,0,1,0
0.0024186171212036826,547.8833333333333,567.29,521.414,33.16008008374236,0,1,0
0.0020174023372413236,547.1613333333333,563.145,516.114,20.97140790221846,0,1,0
```

- File : *training_with_many_dimensions/price_pred_3_lstm_layers_improved.py*

5.2.3.2.13- Training session 13 : 3 bidirectional LSTM layers (batch normalization, optimized rate, dropout, relu activation function) / 400 iterations

- Model :

```
# Define the regularizer
l2_regularizer = L2(0.01)

# Create the model
model = Sequential()
model.add(Input(shape=(nb_timesteps, nb_features)))
model.add(Bidirectional(LSTM(100, return_sequences=True)))
model.add(BatchNormalization())
model.add(Dropout(0.3))

model.add(Bidirectional(LSTM(50, return_sequences=True)))
model.add(BatchNormalization())
model.add(Dropout(0.3))

model.add(Bidirectional(LSTM(25)))
model.add(BatchNormalization())
model.add(Dropout(0.3))

model.add(Dense(50, activation="relu", kernel_regularizer=l2_regularizer))
model.add(Dense(1))

# Compile the model
optimizer = Adam(learning_rate=0.0001)
model.compile(loss="mean_squared_error", optimizer=optimizer)
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=400,
    batch_size=32,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Dataset : *Including closing price, moving averages, rsi, bullish crossover of moving averages :*

```
Dernier,MA_150,MA_100,MA_50,RSI,MA_50_supérieure_MA_150,MA_100_supérieure_MA_150,MA_50_supérieure_MA_100
0.0036534146210219563,536.9173333333333,519.5980000000001,566.99,51.461961183144524,1,0,1
0.0036807916298099755,552.1913333333333,558.2639999999999,616.65,40.98658147596265,1,1,1
0.003571283594657896,550.872,566.974,608.878,43.18067680201051,1,1,1
0.0036081009513038533,550.1940000000001,572.725,608.308,45.85840439076001,1,1,1
0.0036364219948776676,550.108,574.2270000000001,608.22,47.657380263581736,1,1,1
0.002585711278289183,547.5333333333333,584.234,547.932,38.84232427318708,1,1,0
0.002597983730504502,548.208,582.969,545.1080000000001,39.50366022574757,0,1,0
0.0024186171212036826,547.8833333333333,567.29,521.414,33.16008008374236,0,1,0
0.0020174023372413236,547.1613333333333,563.145,516.114,20.97140790221846,0,1,0
```

- File : *training_with_many_dimensions/price_pred_3_bidir_lstm_layers.py*

5.2.3.2.14- Training session 14 : 3 bidirectional LSTM layers (batch normalization, optimized rate, dropout, relu activation function) / 650 iterations

- Model :

```
# Define the regularizer
l2_regularizer = L2(0.01)

# Create the model
model = Sequential()
model.add(Input(shape=(nb_timesteps, nb_features)))
model.add(Bidirectional(LSTM(100, return_sequences=True)))
model.add(BatchNormalization())
model.add(Dropout(0.3))

model.add(Bidirectional(LSTM(50, return_sequences=True)))
model.add(BatchNormalization())
model.add(Dropout(0.3))

model.add(Bidirectional(LSTM(25)))
model.add(BatchNormalization())
model.add(Dropout(0.3))

model.add(Dense(50, activation="relu", kernel_regularizer=l2_regularizer))
model.add(Dense(1))

# Compile the model
optimizer = Adam(learning_rate=0.0001)
model.compile(loss="mean_squared_error", optimizer=optimizer)
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=650,
    batch_size=32,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Dataset : *Including closing price, moving averages, rsi, bullish crossover of moving averages :*

```
Dernier,MA_150,MA_100,MA_50,RSI,MA_50_supérieure_MA_150,MA_100_supérieure_MA_150,MA_50_supérieure_MA_100
0.0036534146210219563,536.9173333333333,519.5980000000001,566.99,51.461961183144524,1,0,1
0.0036807916298099755,552.1913333333333,558.2639999999999,616.65,40.98658147596265,1,1,1
0.003571283594657896,550.872,566.974,608.878,43.18067680201051,1,1,1
0.0036081009513038533,550.1940000000001,572.725,608.308,45.85840439076001,1,1,1
0.0036364219948776676,550.108,574.2270000000001,608.22,47.657380263581736,1,1,1
0.002585711278289183,547.5333333333333,584.234,547.932,38.84232427318708,1,1,0
0.002597983730504502,548.208,582.969,545.1080000000001,39.50366022574757,0,1,0
0.0024186171212036826,547.8833333333333,567.29,521.414,33.16008008374236,0,1,0
0.0020174023372413236,547.1613333333333,563.145,516.114,20.97140790221846,0,1,0
```

- File : *training_with_many_dimensions/price_pred_3_bidir_lstm_layers.py*

5.2.3.2.15- Training session 15 : 3 LSTM layers, dataset without sma crossings / 400 iterations

- Model :

```
# Création du réseau de neurones :
model = Sequential()
model.add(LSTM(100, return_sequences=True, input_shape=(nb_timesteps, nb_features), activation="relu"))
model.add(LSTM(50, return_sequences=True, activation="relu"))
model.add(LSTM(25, activation="relu"))
model.add(Dense(1))
model.compile(loss="mean_squared_error", optimizer="adam")
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=400,
    batch_size=32,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Dataset : *Including closing price, moving averages, rsi :*

```
Dernier,MA_150,MA_100,MA_50,RSI
0.0036534146210219563,536.9173333333333,519.5980000000001,566.99,51.461961183144524
0.0036807916298099755,552.1913333333333,558.2639999999999,616.65,40.98658147596265
0.003571283594657896,550.872,566.974,608.878,43.18067680201051
0.0036081009513038533,550.1940000000001,572.725,608.308,45.85840439076001
0.0036364219948776676,550.108,574.2270000000001,608.22,47.657380263581736
0.002585711278289183,547.5333333333333,584.234,547.932,38.84232427318708
0.002597983730504502,548.208,582.969,545.1080000000001,39.50366022574757
0.0024186171212036826,547.8833333333333,567.29,521.414,33.16008008374236
0.0020174023372413236,547.1613333333333,563.145,516.114,20.97140790221846
```

- File : *training_with_many_dimensions/price_pred_3_lstm_layers.py*

5.2.3.2.16- Training session 16 : Adding lags features, dataset without sma crossings / 400 iterations

- Model :

```
# Création du réseau de neurones :
model = Sequential()
model.add(LSTM(10, input_shape=(nb_timesteps, nb_features), activation="relu"))
model.add(Dense(1))
model.compile(loss="mean_squared_error", optimizer="adam")
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=400,
    batch_size=32,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Dataset : *Including closing price, moving averages, rsi, lags features :*

```
Dernier,MA_150,MA_100,MA_50,RSI,Lag_1,Lag_7
0.0036534146210219563,536.9173333333333,519.5980000000001,566.99,51.461961183144524,0.0036534146210219563,0.0036534146210219563
0.0036807916298099755,552.1913333333333,558.2639999999999,616.65,40.98658147596265,0.0036534146210219563,0.0036534146210219563
0.003571283594657896,550.872,566.974,608.878,43.18067680201051,0.0036807916298099755,0.0036534146210219563
0.0036081009513038533,550.1940000000001,572.725,608.308,45.85840439076001,0.003571283594657896,0.0036534146210219563
0.0036364219948776676,550.108,574.2270000000001,608.22,47.657380263581736,0.0036081009513038533,0.0036534146210219563
0.002585711278289183,547.5333333333333,584.234,547.932,38.84232427318708,0.0036364219948776676,0.0036534146210219563
0.002597983730504502,548.208,582.969,545.1080000000001,39.50366022574757,0.002585711278289183,0.0036534146210219563
0.0024186171212036826,547.8833333333333,567.29,521.414,33.16008008374236,0.002597983730504502,0.0036534146210219563
0.0020174023372413236,547.1613333333333,563.145,516.114,20.97140790221846,0.0024186171212036826,0.0036807916298099755
```

- File : *training_with_many_dimensions/price_pred_lags_features.py*

5.2.3.2.17- Training session 17 : Adding lags features, dataset without sma crossings and RSI / 400 iterations

- Model :

```
# Création du réseau de neurones :
model = Sequential()
model.add(LSTM(10, input_shape=(nb_timesteps, nb_features), activation="relu"))
model.add(Dense(1))
model.compile(loss="mean_squared_error", optimizer="adam")
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=400,
    batch_size=32,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Dataset : *Including closing price, moving averages, lags features :*

```
Dernier,MA_150,MA_100,MA_50,Lag_1,Lag_7
0.0036534146210219563,536.9173333333333,519.5980000000001,566.99,0.0036534146210219563,0.0036534146210219563
0.0036807916298099755,552.1913333333333,558.2639999999999,616.65,0.0036534146210219563,0.0036534146210219563
0.003571283594657896,550.872,566.974,608.878,0.0036807916298099755,0.0036534146210219563
0.0036081009513038533,550.1940000000001,572.725,608.308,0.003571283594657896,0.0036534146210219563
0.0036364219948776676,550.108,574.2270000000001,608.22,0.0036081009513038533,0.0036534146210219563
0.002585711278289183,547.5333333333333,584.234,547.932,0.0036364219948776676,0.0036534146210219563
0.002597983730504502,548.208,582.969,545.1080000000001,0.002585711278289183,0.0036534146210219563
0.0024186171212036826,547.8833333333333,567.29,521.414,0.002597983730504502,0.0036534146210219563
0.0020174023372413236,547.1613333333333,563.145,516.114,0.0024186171212036826,0.0036807916298099755
```

- File : *training_with_many_dimensions/price_pred_lags_features.py*

5.2.3.2.18- Training session 18 : Adding lags features and historical volatility, dataset without sma crossings / 400 iterations

- Model :

```
# Création du réseau de neurones :
model = Sequential()
model.add(LSTM(10, input_shape=(nb_timesteps, nb_features), activation="relu"))
model.add(Dense(1))
model.compile(loss="mean_squared_error", optimizer="adam")
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=400,
    batch_size=32,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Dataset : *Including closing price, moving averages, rsi, lags features, historical volatility :*

```
Dernier,MA_150,MA_100,MA_50,RSI,Historical_Volatility,Lag_1,Lag_7
0.0036534146210219563,536.9173333333333,519.5980000000001,566.99,51.46196183144524,0.6607824657304868,0.0036534146210219563,0.0036534146210219563
0.0036807916298099755,552.1913333333333,558.2639999999999,616.65,40.98658147596265,0.6607824657304868,0.0036534146210219563,0.0036534146210219563
0.003571283594657896,550.872,566.974,608.878,43.18067680201051,0.6607824657304868,0.0036807916298099755,0.0036534146210219563
0.0036081009513038533,550.1940000000001,572.725,608.308,45.85840439076001,0.6607824657304868,0.003571283594657896,0.0036534146210219563
0.0036364219948776676,550.108,574.2270000000001,608.22,47.657380263581736,0.6607824657304868,0.0036081009513038533,0.0036534146210219563
0.002585711278289183,547.5333333333333,584.234,547.932,38.84232427318708,0.6607824657304868,0.0036364219948776676,0.0036534146210219563
0.002597983730504502,548.208,582.969,545.1080000000001,39.50366022574757,0.6607824657304868,0.002585711278289183,0.0036534146210219563
0.0024186171212036826,547.8833333333333,567.29,521.414,33.16008008374236,0.6607824657304868,0.002597983730504502,0.0036534146210219563
0.0020174023372413236,547.1613333333333,563.145,516.114,20.97140790221846,0.6607824657304868,0.0024186171212036826,0.0036807916298099755
```

- File : *training_with_many_dimensions/price_pred_lags_features_and_vol.py*

5.2.3.2.19- Training session 19 : Adding lags features and historical volatility, dataset without sma crossings and RSI / 400 iterations

- Model :

```
# Création du réseau de neurones :
model = Sequential()
model.add(LSTM(10, input_shape=(nb_timesteps, nb_features), activation="relu"))
model.add(Dense(1))
model.compile(loss="mean_squared_error", optimizer="adam")
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=400,
    batch_size=32,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Dataset : *Including closing price, moving averages, lags features, historical volatility :*

```
Dernier,MA_150,MA_100,MA_50,Historical_Volatility,Lag_1,Lag_7
0.0036534146210219563,536.9173333333333,519.5980000000001,566.99,0.6607824657304868,0.0036534146210219563,0.0036534146210219563
0.0036807916298099755,552.1913333333333,558.2639999999999,616.65,0.6607824657304868,0.0036534146210219563,0.0036534146210219563
0.003571283594657896,550.872,566.974,608.878,0.6607824657304868,0.0036807916298099755,0.0036534146210219563
0.0036081009513038533,550.1940000000001,572.725,608.308,0.6607824657304868,0.003571283594657896,0.0036534146210219563
0.0036364219948776676,550.108,574.2270000000001,608.22,0.6607824657304868,0.0036081009513038533,0.0036534146210219563
0.002585711278289183,547.5333333333333,584.234,547.932,0.6607824657304868,0.0036364219948776676,0.0036534146210219563
0.002597983730504502,548.208,582.969,545.1080000000001,0.6607824657304868,0.002585711278289183,0.0036534146210219563
0.0024186171212036826,547.8833333333333,567.29,521.414,0.6607824657304868,0.002597983730504502,0.0036534146210219563
0.0020174023372413236,547.1613333333333,563.145,516.114,0.6607824657304868,0.0024186171212036826,0.0036807916298099755
```

- File : *training_with_many_dimensions/price_pred_lags_features_and_vol.py*

5.2.3.2.20- Training session 20 : Using 3 bidirectional LSTM layers (batch normalization, optimized rate, dropout, relu activation function), dataset without sma crossings / 650 iterations

- Model :

```
# Define the regularizer
l2_regularizer = L2(0.01)

# Create the model
model = Sequential()
model.add(Input(shape=(nb_timesteps, nb_features)))
model.add(Bidirectional(LSTM(100, return_sequences=True)))
model.add(BatchNormalization())
model.add(Dropout(0.3))

model.add(Bidirectional(LSTM(50, return_sequences=True)))
model.add(BatchNormalization())
model.add(Dropout(0.3))

model.add(Bidirectional(LSTM(25)))
model.add(BatchNormalization())
model.add(Dropout(0.3))

model.add(Dense(50, activation="relu", kernel_regularizer=l2_regularizer))
model.add(Dense(1))

# Compile the model
optimizer = Adam(learning_rate=0.0001)
model.compile(loss="mean_squared_error", optimizer=optimizer)
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=650,
    batch_size=32,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Dataset : *Including closing price, moving averages, rsi :*

```
Dernier,MA_150,MA_100,MA_50,RSI
0.0036534146210219563,536.9173333333333,519.5980000000001,566.99,51.461961183144524
0.0036807916298099755,552.1913333333333,558.2639999999999,616.65,40.98658147596265
0.003571283594657896,550.872,566.974,608.878,43.18067680201051
0.0036081009513038533,550.1940000000001,572.725,608.308,45.85840439076001
0.0036364219948776676,550.108,574.2270000000001,608.22,47.657380263581736
0.002585711278289183,547.5333333333333,584.234,547.932,38.84232427318708
0.002597983730504502,548.208,582.969,545.1080000000001,39.50366022574757
0.0024186171212036826,547.8833333333333,567.29,521.414,33.16008008374236
0.0020174023372413236,547.1613333333333,563.145,516.114,20.97140790221846
```

- File : *training_with_many_dimensions/price_pred_3_bidir_lstm_layers.py*

5.2.4- Comparison of models' results

We see the 16th training session gives the best metrics :

- Model :

```
# Création du réseau de neurones :
model = Sequential()
model.add(LSTM(10, input_shape=(nb_timesteps, nb_features), activation="relu"))
model.add(Dense(1))
model.compile(loss="mean_squared_error", optimizer="adam")
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=400,
    batch_size=32,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Dataset : *Including closing price, moving averages, rsi, lags features :*

```
Dernier,MA_150,MA_100,MA_50,RSI,Lag_1,Lag_7
0.0036534146210219563,536.9173333333333,519.5980000000001,566.99,51.461961183144524,602.2,597.0
0.0036807916298099755,552.1913333333333,558.2639999999999,616.65,40.98658147596265,601.9,627.5
0.003571283594657896,550.872,566.974,608.878,43.18067680201051,591.7,595.0
0.003608100951303853 3,550.1940000000001,572.725,608.308,45.85840439076001,589.8,589.5
```

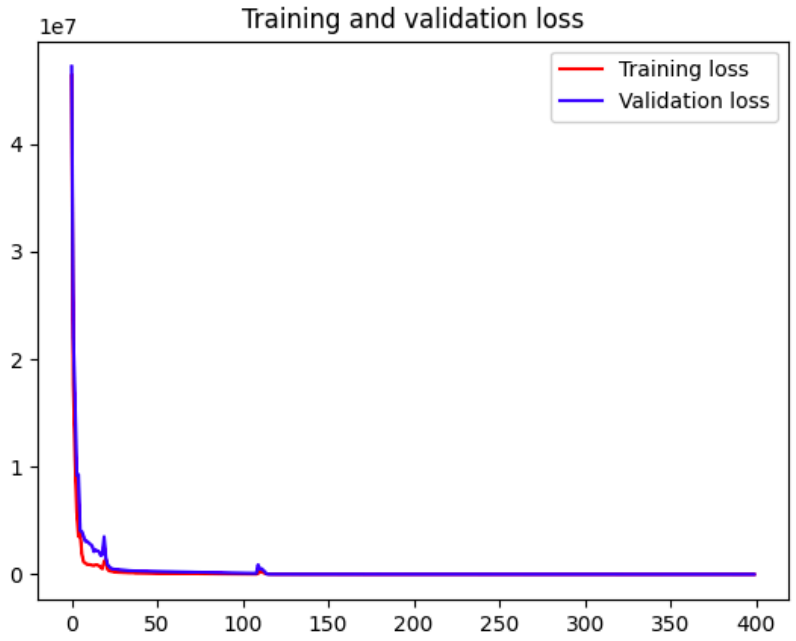
- File : *training_with_many_dimensions/price_pred_lags_features.py*

- Repository : https://github.com/SydneyJezequel/btc_neural_network/tree/main/training_with_many_dimensions/price_pred_lags_features.py

5.2.5- Detailed analysis of training results 16

5.2.5.1- Loss curves

We observe the training and validation loss curves converge :



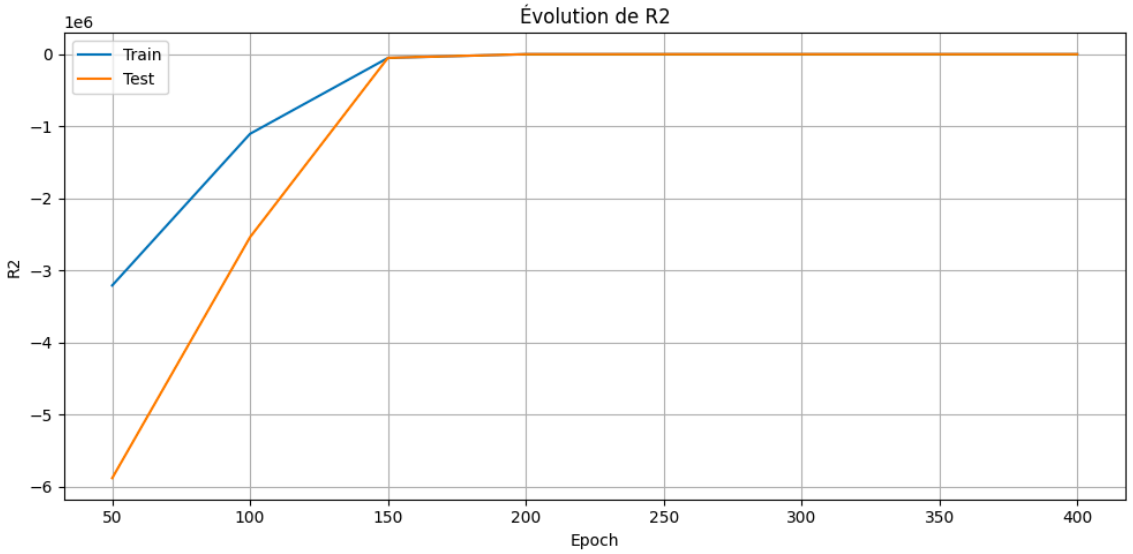
5.2.5.2- Metrics

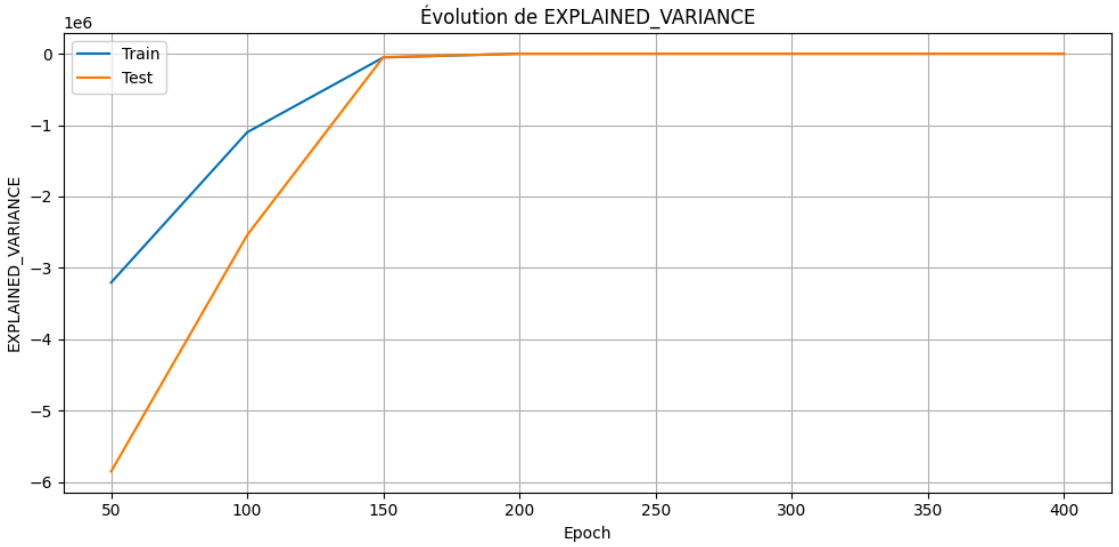
This table shows the metrics' evolution during the training session. We observe metrics degrade around epoch 350. The model is probably in overfitting after this period :

Epoch	test_explained_variance	test_mae	test_mgd	test_mpd	test_mse	test_r2	test_rmse	train_explained_variance	train_mae	train_mgd	train_mpd	train_mse	train_r2	train_rmse
50	-5.85e+06	1.05e+06	NaN	NaN	1.91e+12	-5.88e+06	1.38e+06	-3.20e+06	5.47e+05	NaN	NaN	6.99e+11	-3.21e+06	8.36e+05
100	-2.54e+06	3.71e+05	NaN	NaN	2.55e+11	-2.54e+06	5.05e+05	-1.10e+06	1.82e+05	NaN	NaN	7.45e+10	-1.10e+06	2.73e+05
150	-4.62e+04	9.45e+03	NaN	NaN	1.61e+08	-5.08e+04	1.27e+04	-5.20e+04	6.76e+03	NaN	NaN	1.11e+08	-5.22e+04	1.05e+04
200	-1.51e+02	3.02e+01	NaN	NaN	1.66e+03	-1.60e+02	4.07e+01	-1.77e+02	2.23e+01	NaN	NaN	1.22e+03	-1.77e+02	3.50e+01
250	8.52e-01	3.54e-02	2.54e-02	7.85e-03	2.57e-03	7.77e-01	5.07e-02	8.82e-01	2.54e-02	4.22e-02	5.71e-03	9.36e-04	8.79e-01	3.06e-02
300	8.80e-01	3.01e-02	1.79e-02	5.72e-03	1.97e-03	8.43e-01	4.43e-02	9.19e-01	2.39e-02	3.95e-02	4.92e-03	7.76e-04	9.08e-01	2.79e-02
350	8.96e-01	2.63e-02	1.37e-02	4.25e-03	1.48e-03	8.92e-01	3.84e-02	9.27e-01	2.87e-02	4.72e-02	6.44e-03	1.14e-03	8.75e-01	3.38e-02
400	7.61e-01	5.16e-02	9.94e-02	2.06e-02	4.85e-03	5.28e-01	6.96e-02	7.64e-01	3.04e-02	7.89e-02	9.79e-03	1.65e-03	7.61e-01	4.06e-02

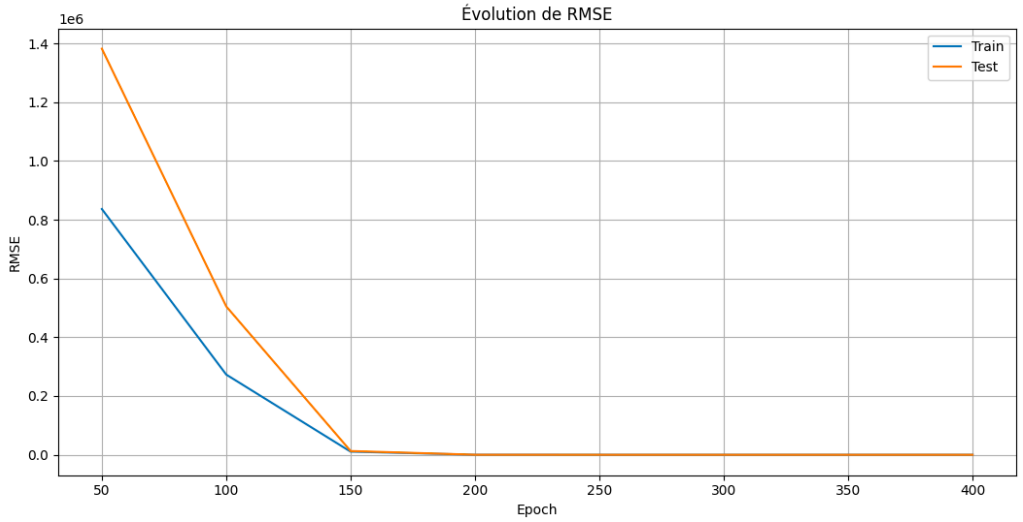
5.2.5.3- Analysis of metrics

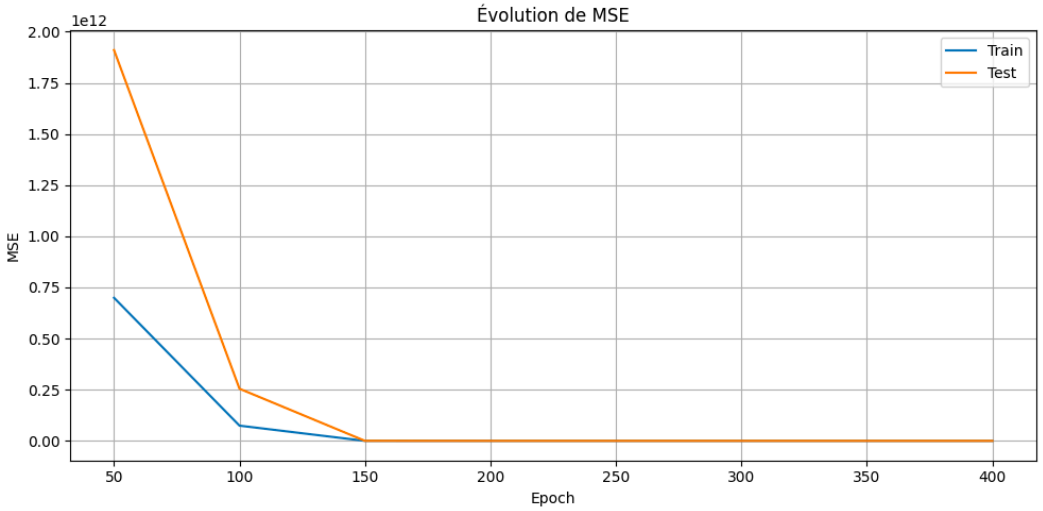
We observe that R^2 and EVS metrics converge well for training and test datasets.
The model explains the variance of the data better and better during the training session.
It does not appear to be overfitting.

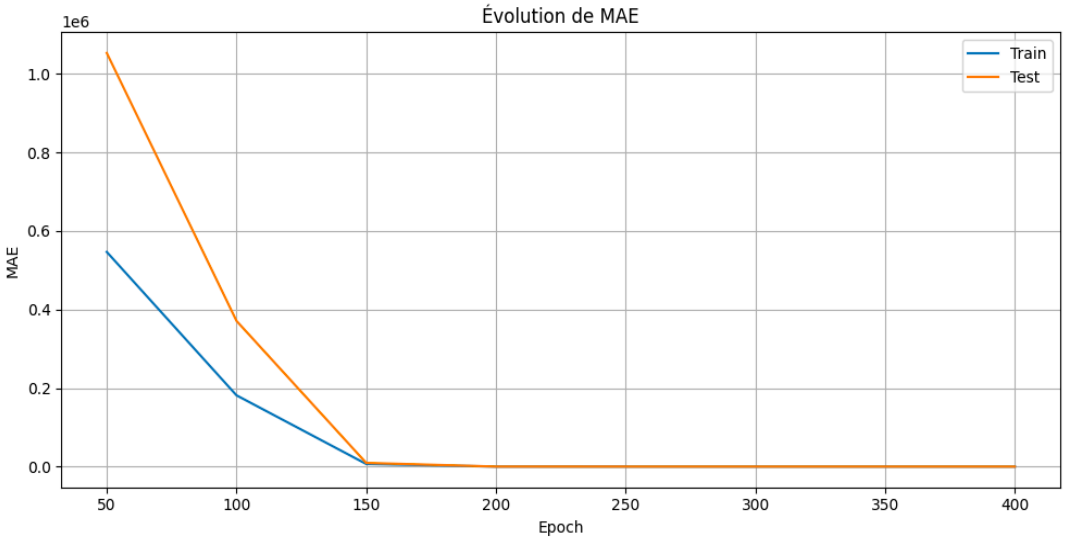




We observe RMSE, MSE, and MAE metrics converge well for training and test phases. The difference between predicted and actual values decreases more and more during the training session. The model is not overfitted.







5.3- Select the best model

Adding new dimensions and increasing the complexity of the model do not improve results. The best results are given by the 3rd training session on the one-dimensional dataset *(See section: 5.1.4.2.1.1.3- Training session 3 : Using 1 LSTM layer / 300 iterations / 50 batch size)*.

Training session details :

- Model :

```
# Création du réseau de neurones :
model = Sequential()
model.add(LSTM(10, input_shape=(None, 1), activation="relu"))
model.add(Dense(1))
model.compile(loss="mean_squared_error", optimizer="adam")
```

```
# Entraînement du modèle :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=300,
    batch_size=50,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Dataset : *Including only the closing price :*

```
Dernier
0.007826286248918912
0.0
0.004541283235544055
0.004861012211006034
0.004789332558690074
0.004832717611407629
0.003308581846373529
0.0030511009900280406
0.0036490601948743397
0.0035490859429599744
0.003286889320014752
0.0031491889353025123
0.0031737109216211307
0.003225584354218207
```

- Repository : https://github.com/SydneyJezequel/btc_neural_network/tree/main/training_with_one_dimension/price_pred_1_lstm_layer.py

6- Ideas to enhance the model

Once the model is selected, the next step is to improve the predictions' accuracy by integrating events which may impact Bitcoin's price. To do this, we can proceed as follows :

1- Collecting news data about Bitcoin.
For example, with APIs on financial news sites or with Google Finance API.

2- Using a language model like an LLM to extract market sentiment from the collected data.
Sentiment can be positive, negative, or neutral. It influences price movements. We can use pre-trained models like BERT or libraries like NLTK to generate sentiment scores from collected data.

3- Integrate market sentiment into the dataset to enhance predictions.
We can use sentiment scores generated in step 2 as news features to enrich the dataset. Simply by adding a new column.

6.1- Identifying a data source

The first point is to find a source to collect data about Bitcoin and Financial markets. Most of the data sources are paid. But some of them are free under certain conditions *(not going too far back in data history, limited number of daily calls, etc.).* It limits their usefulness, but we can do some tests.
After some research, I have selected EODHD which provide :

- Bitcoin market sentiment scores *(based on several media and social networks)*.
- News about Bitcoin.

6.2- Training model with sentiment scores from EODHD

In this section, we will train a model using the sentiment scores provided by EODHD. These scores are a synthesis of market sentiment from media and social networks.

6.2.1- Trained Model

- Model :

```
# Original neural network :
model = Sequential()
model.add(LSTM(10, input_shape=(nb_timesteps, nb_features), activation="relu"))
model.add(Dense(1))
model.compile(loss="mean_squared_error", optimizer="adam")
```

```
# Model training :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=300,
    batch_size=50,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Repository :

https://github.com/SydneyJezequel/btc_neural_network/blob/main/training_with_market_sentiment/training_with_api_scores_sentiments.py

6.2.2- Tests results (logs)

{"epoch": [50, 100, 150, 200, 250, 300],
"test_explained_variance": [
0.9940585890750591,
0.9957372011688463,
0.9961383483825833,
0.9956783860315008,
0.9955547058036254,
0.9927534400803586],
"test_mae": [
0.006664306675056148,
0.006428732323724049,
0.005272346971607222,
0.006895980779143493,
0.005592512040009276,
0.00915512354790315],

"test_mgd": [
0.0012242976234473496,
0.0010875834294053759,
0.000761049567937679,
0.0012090291017775553,
0.0008081222851385845,
0.0018554878722630613],
"test_mpd": [
0.0003128138215221504,
0.0002808667214573167,
0.00020766724254903416,
0.0003120681425433269,
0.0002307799578408687,
0.000523516504357209],
"test_mse": [
0.00010483934061420073,
0.00009140466140132898,
0.0000720477977294357,
0.00010359222219413185,
0.00008297878858110117,
0.00019313112110731906],
"test_r2": [
0.9940551494773291,
0.9948865497037093,
0.9961132147020051,
0.9942028998837653,
0.9953864756456049,
0.9889605778030464],
"test_rmse": [
0.01023910838961092,
0.00956057850767039,
0.008488097415171182,
0.010178026439056437,
0.009109269376909498,
0.013897162340108108],
"train_explained_variance": [
0.9934558324217917,
0.9948669527762658,
0.9954331396408315,
0.995487980428903,
0.9955632789645114,
0.9955227989266627],
"train_mae": [
0.005802397985168061,
0.0051409833997366275,
0.004625058234047905,
0.005328401102047372,
0.004394016523983863,
0.005732011893082087],
"train_mgd": [

0.010057296206034577,
0.008096699923356343,
null,
null,
0.00486792589012742,
null],
"train_mpd": [
0.000509129978734118,
0.00042520252421908796,
null,
null,
0.00033748101686444874,
null],
"train_mse": [
0.00007828606979976944,
0.00006637051498601084,
0.00005743489907349214,
0.00006341978239265453,
0.00005410196419851993,
0.00006677692649106448],
"train_r2": [
0.9934514956062308,
0.99452275892593,
0.995429258320547,
0.9947646059748547,
0.9955626903715468,
0.9942856369286062],
"train_rmse": [
0.008847941557208062,
0.008146810111080953,
0.007578581600371678,
0.007963653834305867,
0.00735540374136729,
0.008232208993377465]}

6.2.3- Conclusion

The model using scores provided by the EOHD API shows very good performance. EVS and R^2 metrics are very high : Around 0.99.
The MAE, MSE and RMSE are very low: between 10^{-3} and 10^{-5}.However, we observe a slight overfitting. It is indicated by a divergence in performance between test and training set : Metrics of the training phase remain excellent and are improving throughout the training process. Metrics of the testing process reach an optimum around epoch 150 and then start to degrade.

Despite this minor overfitting, the model shows good generalization. There are two ways to manage it. Either we limit the epochs to 150. Or we apply the technique of early stopping to halt training when the model starts to overfit.

6.3- Training model with our own sentiment scores

In this section, we will train our model with our own generated sentiment scores.

6.3.1- Using dedicated libraries to generate sentiment scores

6.3.1.1- Tests

Initially, I tried using well-known libraries to generate sentiment scores *(ex : Roberta which is highly-rated in Hugging Face)* :

- NLTK (vader)
- TextBlob
- Roberta (Bert)

- *Exemple with TextBlob :*
  Title : Bitcoin Would 'Moon' If US Nationalizes Strategy, Predicts Analyst Willy Woo
  Polarity (TextBlob): 0.0000
  Subjectivity (TextBlob): 0.0000

  Title : House Advances Major Crypto Bill. Industry Voices React
  Polarity (TextBlob): 0.0625
  Subjectivity (TextBlob): 0.5000

- *Exemple with NLTK (vader) :*
  Title: Bitcoin Would 'Moon' If US Nationalizes Strategy, Predicts Analyst Willy Woo
  Scores (NLTK): {'neg': 0.0, 'neu': 0.763, 'pos': 0.237, 'compound': 0.4767}

  Title: House Advances Major Crypto Bill. Industry Voices React
  Scores (NLTK): {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}

- *Exemple with Roberta :*
  text_to_analyze : Don't Trade, Buy Bitcoin: Legendary Trader Peter Brandt *sentiment_result : {'label': 'negative', 'score': 0.3767838776111603}*

6.3.1.2- Conclusion

The returned scores were not relevant. I think these results are poor because the libraries we used are designed to estimate human sentiments. They are not made to evaluate how market participants perceive current events : « market sentiment ».

Using a large language model (LLM), which is semantically richer, should be more appropriate.

6.3.2- Using a LLM to generate sentiment scores

For these tests, I used Monster API. It provides access to various LLM APIs. I decided to work with the « Llama3.3_70b » model.

I chose Llama models because they are known for their language understanding performance. Two models caught my attention, but I selected the most powerful because larger models generally give better results.

- **Llama3.3_70b :** It is a powerful model with 70 billion parameters. It is well-suited for language analysis.
- **Meta-Llama-3.1-8B-Instruct :** It is another model from the Llama family. It is also well-suited for sentiment analysis, but smaller than Llama3.3_70b.

<u>LLM Sentiment Analysis Implementation :</u>

The implementation of the sentiment analysis with an LLM *(Monster API calls, prompt, manage LLM responses, etc.)* is located in the SentimentAnalyzer class. This class can be found at the following location in the GitHub project :
https://github.com/SydneyJezequel/btc_neural_network/blob/main/BO/sentiment_analyzer.py

sentiment_analyzer.py

```python
class SentimentAnalyzer:  2 usages  SydneyJezequel

    def analyze_sentiment(self, text):  1 usage  SydneyJezequel
        """ Sentiment scoring on an article """
        response = self.monster_client.chat.completions.create(
            # model="meta-llama/Meta-Llama-3.1-8B-Instruct",
            model=self.model_name,
            messages=[
                {"role": "system", "content": "You are a helpful assistant."},
                {"role": "user",
                 "content": f"Évalue le sentiment du texte suivant sur une échelle de 1 à 10, où 1 est très négatif et 10
            ]
        )
        sentiment_score = response.choices[0].message.content.strip()
        try:
            sentiment_score = int(sentiment_score)
        except ValueError:
            sentiment_score = 0
        return {"text": text, "score": sentiment_score, "explanation": "Score de sentiment numérique"}



    def analyze_sentiments(self, articles):  1 usage  SydneyJezequel
        """ Sentiment scoring on multiple articles """
        results = []
        for article in articles:
            sentiment = self.analyze_sentiment(article['title'])
            sentiment['date'] = article['date']
            results.append(sentiment)
        return results
```

Example of scores Generated by the « Llama3.3_70b » Model:

Title : Bitcoin Vaulted to Top Macroeconomic Indicator, McGlone Says

Score (MonsterAPI) : 8.0000

Explanation : Score de sentiment numérique

Title : XRP Showing Bullish Pattern That Could Trigger 281% Surge, According to Peter Brandt – But There's a Catch

Score (MonsterAPI) : 7.0000

Explanation : Score de sentiment numérique

Title : Meme Coins Decimated Amid Crypto Massacre: BONK, FLOKI, AI16Z Hit Worst

Score (MonsterAPI) : 1.0000

Explanation : Score de sentiment numérique

Title : Peter Brandt Sounds Alarm on Bitcoin, Bulls Buckle Up

Score (MonsterAPI) : 6.0000

Explanation : Score de sentiment numérique

Title : The Truth About the Russia Bitcoin Mining Ban

Score (MonsterAPI) : 5.0000

Explanation : Score de sentiment numérique

Title : Crypto Market Bloodbath: $249 Million Worth of Longs Wiped Out in Mere Hours

Score (MonsterAPI) : 1.0000

Explanation : Score de sentiment numérique

Analysis :

The scores obtained are more relevant than those generated by the libraries.
We can see that positive events *(ex : XRP Showing Bullish Pattern That Could Trigger 281% Surge, According to Peter Brandt – But There's a Catch)* are provided a high score. Negative events receive a low score *(ex : Crypto Market Bloodbath: $249 Million Worth of Longs Wiped Out in Mere Hours)*.
News with no impact on the market has a medium/neutral score *(ex : The Truth About the Russia Bitcoin Mining Ban, Peter Brandt Sounds Alarm on Bitcoin, Bulls Buckle Up)*.

6.3.2.1- Trained Model

- Model :

```
# Original neural network :
model = Sequential()
model.add(LSTM(10, input_shape=(nb_timesteps, nb_features), activation="relu"))
model.add(Dense(1))
model.compile(loss="mean_squared_error", optimizer="adam")
```

```
# Model training :
history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=300,
    batch_size=50,
    verbose=1,
    callbacks=[metrics_callback]
)
```

- Repository :

https://github.com/SydneyJezequel/btc_neural_network/blob/main/training_with_market_sentiment/training_with_own_generated_scores_sentiments.py

6.3.2.2- Tests results (logs)


```
{"epoch": [ 50, 100, 150, 200, 250, 300],
  "test_explained_variance": [
    0.9631585484090561,
    0.9666786858863408,
    0.9662909602781427,
    0.9648017795769586,
    0.9638705942011864,
    0.9647763794737972],
  "test_mae": [
    0.009108131974394489,
    0.00795810790106103,
    0.006940567266663952,
    0.006794220101272671,
    0.007234347390190917,
    0.0068503937139544205],
  "test_mgd": [
    0.0064362075238266385,
    0.005756712293020927,
    0.006580304297946348,
    0.007555718936797279,
    0.00778346937419437,
    0.0068926460700063607],
  "test_mpd": [
    0.0019025746961965591,
    0.0017999225791371022,
    0.0019285565265034766,
    0.002110995911976805,
    0.002176746001178444,
    0.002024588402874744],
  "test_mse": [
    0.0006201980552172161,
    0.0006145303519808826,
    0.0006205063154442822,
    0.0006538557054994144,
    0.0006737602714915657,
    0.0006517075124316621],
  "test_r2": [
    0.9630234000383514,
    0.9666761121583469,
    0.9655860779354957,
    0.964035245789705,
    0.9625406084110785,
    0.9646428201414374],
  "test_rmse": [
    0.024903775922883984,
    0.02478972270883405,
    0.02490996417990765,
    0.02557060236872441,
    0.025956892562315037,
    0.025528562678530535],
  "train_explained_variance": [
    0.9923622909986508,
    0.9937719971691574,
    0.9949721882713592,
    0.9952598559642328,
```

```
  0.9952525877579278,
  0.9953259888437984],
 "train_mae": [
  0.006967508782283471,
  0.005782715104165469,
  0.004868558715716161,
  0.004636245901265284,
  0.004604908131900829,
  0.004786489860424868],
 "train_mgd": [
  0.016029785172726078,
  0.009495573331501153,
  0.008117416381524972,
  0.01667796748186885,
  0.004948293178033417,
  0.008265196927510628],
 "train_mpd": [
  0.0007065641573661567,
  0.0005015109635345352,
  0.0003907697417392384,
  0.0003839749126143969,
  0.0003615345514308966,
  0.0003756915673522185],
 "train_mse": [
  0.00009389466641537736,
  0.00008119186982912926,
  0.00006151732628606685,
  0.00005915374548358281,
  0.00005908154417084863,
  0.00005974371719825832],
 "train_r2": [
  0.9917419316665735,
  0.9935051990778843,
  0.9949670062569153,
  0.9952002523856028,
  0.9951543936249349,
  0.9952185666637671 ],
 "train_rmse": [
  0.0096899260273429,
  0.009010653129997251,
  0.007843298176536886,
  0.007691147215050743 4,
  0.007686451988456614,
  0.007729406005525801 5]}
```

6.3.2.3- Conclusion

Using an LLM for market sentiment analysis seems more relevant than using sentiment analysis libraries. However, generating our own scores appears to be less performant than using the sentiment scores provided by EODHD.

The performance when using our own sentiment scores remains very good :

- The absence of NaN values for the MGD/MPD metrics is an improvement compared to the previous training.
- The EVS and R^2 metrics are still very high : Around 0.96 to 0.966 for the test phase and around 0.991 to 0.995 for the training phase. It indicates the model effectively explains the variation of data.
- MAE, MSE, and RMSE metrics are low : From 10^{-2} to 10^{-5} for RMSE, and from 10^{-3} to 10^{-4} for MAE and MSE.

However, there are a some elements to note :

- There is a persistent gap between training performance (R^2 is around 0.995) and test performance (R^2 is around 0.965). A test R^2 at 0.96 is very good. But this difference could incidate the model was not trained enough to provide more precise predictions during the test phase.
- Additionally, we observe small fluctuations in the test metrics after epochs 100-150. The metrics no longer improve but also do not degrade. It suggests the model might have reached its limits.

The model is robust and generalizes well. The only point to consider is the constant difference between training and test metrics. There are two possibilities :

- The model was not trained sufficiently. It leads to less precise results during the test phase compared to training.
- The model is not complex enough to capture all the specificities of the dataset.

These elements would suggest we are experiencing underfitting.

6.4- Conclusion

Training with sentiment scores from EODHD (section: « *6.2- Training model with sentiment scores from EODHD »*) offers better performances on the test dataset, despite signs of slight overfitting.

- For correlations metrics (R^2) : The training session with EODHD scores provide a R^2 of approximately 0.996 at epoch 150 of the test phase. The training session with our own scores is around 0.966 at epochs 100 and 150. An R^2 of 0.996 is exceptional. It indicates the model has an almost perfect predictive ability on unseen data. An R^2 of 0.966 is also very good but lower.
- For the errors : The errors from the training with EODHD sentiment scores are lower. For instance, the minimum test_mae is 0.00527 at epoch 150, compared to 0.00694 at epoch 150 for the training with our own scores. It means the predictions from the EODHD's scores training are, on average, closer to the actual values.

These differences can be explained by the quality of scores provided by the EODHD API. These scores come from the aggregation of multiple sources (media, social networks, etc.). They provide a more global and comprehensive view of the market than scores generated by the LLM which come from a single information source.

However, the null metric values remain a weak point for the first training run.

By keeping the training based on EODHD sentiment scores, the next step will consist of implementing an early stopping mechanism with a patience of 20 or 50 epochs. The idea is to stop the training session as soon as performance on the test dataset no longer improves.

7- Appendices – Model metrics

This section lists metrics resulting from the training processes.

7.1- Training sessions with a one dimensional dataset

7.1.1- Training session 1 : Using 1 LSTM layer / 200 iterations

- File : *training_with_one_dimension/price_pred_1_lstm_layer.py*

- Logs :

Metrics History:
epoch: [50, 100, 150, 200]
train_rmse: [0.010843230853733114, 0.009514538975523661, 0.009027380488528525, 0.008137786761460163]
train_mse: [0.00011757565534734975, 9.052645191675884e-05, 8.14935984846655e-05, 6.62235733749963e-05]
train_mae: [0.007406600935896878, 0.0064505816854828, 0.00608482642699738, 0.005263784103946073]
test_rmse: [0.03796968915470835, 0.020068375545433302, 0.013906746362792117, 0.00908482531397436]
test_mse: [0.00144169729450517 7, 0.0004027396970325 4545, 0.0001933975943990 3175, 8.253405098542933e-05]
test_mae: [0.016186528728082957, 0.010219419597777382, 0.00816153514503032, 0.005825391941161068]
train_explained_variance: [0.9902891415225528, 0.9924440636891291, 0.993586238625506, 0.9944930604752001]
test_explained_variance: [0.9253703427678379, 0.9780034873697899, 0.9907399545118187, 0.9955719151243515]
train_r2: [0.9897125154631097, 0.9923912098868117, 0.9932763468452878, 0.9944930253239379]
test_r2: [0.9153315751360422, 0.9772793812678324, 0.9892900243342698, 0.9953933073310484]
train_mgd: [0.017243491401680296, 0.015235011674401442, nan, 0.012433598459769699]
test_mgd: [0.00865760536754331, 0.0025091213482893505, 0.0015422929516234556, 0.0008954256895759833]
train_mpd: [0.0008027703212411089, 0.0006151389399248679, nan, 0.000423503421259961]
test_mpd: [0.003417850736584203, 0.0009219833427229315, 0.0004838975499257553, 0.00024295739987387294]

7.1.2- Training session 2 : Using 1 LSTM layer / 300 iterations

- File : *training_with_one_dimension/price_pred_1_lstm_layer.py*

- Logs :

Metrics History:
epoch: [50, 100, 150, 200, 250, 300]
train_rmse: [0.010531291105316778, 0.00935683282893234, 0.008486207531271285, 0.007967201138246232, 0.007912942950143994, 0.007749906031441711]
train_mse: [0.00011090809234492427, 8.755032058858598e-05, 7.201571826380547e-05, 6.347629397727206e-05, 6.261466613223353e-05, 6.006104349617662e-05]
train_mae: [0.006887232775644011, 0.0064948562025739336, 0.005702772021457658, 0.005083202375948280 5, 0.005140895949848802, 0.004857756412237046]
test_rmse: [0.011154845092612395, 0.009836444150432801, 0.00928483033551266, 0.008418272979227063, 0.008376273698644549, 0.009119000514221464]
test_mse: [0.00012443056904017883, 9.675563352458367e-05, 8.620807435925615e-05, 7.086731995278449e-05, 7.016196107460442e-05, 8.315617037837134e-05]
test_mae: [0.007708538967957284, 0.007015485122252917, 0.006062776306283874, 0.005500536212752502, 0.005601171509890335, 0.005740447726595942]
train_explained_variance: [0.9903607330558916, 0.993246744343229, 0.9942938200745665, 0.9949387786357159, 0.9952016931725811, 0.9952805125912484]
test_explained_variance: [0.9934928303168082, 0.9945155070934226, 0.9957190522883183, 0.9961312142867642, 0.9962163170929996, 0.9960290831477341]
train_r2: [0.9901619583300239, 0.992578548904726, 0.994048508048633, 0.9948364968829846, 0.994944467732343, 0.9951390107006507]
test_r2: [0.9925915369488454, 0.9944949285531826, 0.9952180821829452, 0.9961306813259845, 0.996197684530565, 0.9954826693348362]
train_mgd: [0.014732480313727259, 0.014103056530087461, nan, 0.010802606891792207, 0.011577610811706369, nan]
test_mgd: [0.0016567252492142475, 0.0012747319635613458, 0.0010160197276688427, 0.0007990174073029787, 0.0007949399180164396, 0.0008028152898494704]
train_mpd: [0.0006990265846280359, 0.0005877492484992717, nan, 0.0003942320405104975, 0.0003944349119569563 3, nan]
test_mpd: [0.0004056001070464475 5, 0.0003069694382205148 7, 0.0002614351114345917, 0.0002107745731062047 7, 0.0002087199434767536 5, 0.0002342322847187683 8]

7.1.3- Training session 3 : Using 1 LSTM layer / 300 iterations / 15 neurons / batch size : 50

- File : *training_with_one_dimension/price_pred_1_lstm_layer.py*

- Logs :

Metrics History:
epoch: [50, 100, 150, 200, 250, 300]
train_rmse: [0.010033853288829601, 0.00897880507438017 5, 0.009230751677509377, 0.008172466476178, 0.008339637911288373, 0.007838313116867382]
train_mse: [0.00010067821182175661, 8.061894056371518e-05, 8.520677653184219e-05, 6.678920830425325e-05, 6.954956049139831e-05, 6.143915251805525e-05]
train_mae: [0.006584649862808894, 0.005751232136466137, 0.006299719535592 29, 0.005321907496667565 4, 0.005569769976414633, 0.004984881986353211]
test_rmse: [0.019339736812145323, 0.01688346396870699, 0.010177173516828762, 0.009321437635447556, 0.008469500596841326, 0.008271019185232516]

test_mse: [0.0003740254199630 4887, 0.0002850513555826 2723, 0.0001035748607916 4069, 8.688919959153811e-05, 7.173244035989557e-05, 6.840975836248436e-05]
test_mae: [0.011156155244755349, 0.009234784513091303, 0.007204432186723599, 0.006130862439257547, 0.005915665130333463, 0.005411791743090822]
train_explained_variance: [0.9915283090133853, 0.9932577201130701, 0.9941167384787846, 0.9948364808863265, 0.995057126682327, 0.9951923051680827]
test_explained_variance: [0.9826821090989585, 0.9860185487301437, 0.9944777539926672, 0.9952197953177955, 0.9963781874639287, 0.9962526582181468]
train_r2: [0.9911110122572727, 0.9932125419693846, 0.9930960316330633, 0.9945638998883993, 0.9944596389168372, 0.9950349956473291]
test_r2: [0.9776139763037311, 0.9837313241111307, 0.994310972394064, 0.9952059123156598, 0.996126368427156, 0.9962524149364171]
train_mgd: [0.011791220475613214, 0.00954274946321579, 0.012289026025974746, 0.01022686415669404, 0.011195360972393613, 0.00970498428103197]
test_mgd: [0.0026076166727096854, 0.0018371360652850153, 0.0011828373239627947, 0.0008944897258197231, 0.0008717387479275563, 0.0007477270943027585]
train_mpd: [0.0006739915220575999, 0.0005231681190034587, 0.0005603761320011761, 0.0004450902762590341, 0.0004613557892415 5925, 0.0004022386929948 0793]
test_mpd: [0.0009049364342661985, 0.0006604042574207237, 0.0003072016321186 4286, 0.0002452702329493983, 0.0002208600035070 6052, 0.000201083803482703]

7.1.4- Training session 4 : Using 2 LSTM layers / 200 iterations

- File : *training_with_one_dimension/price_pred_2_lstm_layers.py*

- Logs :

Metrics History:
epoch: [50, 100, 150, 200]
train_rmse: [0.009469228751702169, 0.007832405806022392, 0.008607939468351524, 0.007595815191528294]
train_mse: [8.966629315206303e-05, 6.134658071021328e-05, 7.409662189080393e-05, 5.769640842385201e-05]
train_mae: [0.006220177960274961, 0.004801979970313554, 0.006161492556075477, 0.004620054409988747]
test_rmse: [0.012519527970575378, 0.009529972804585535, 0.009508560343644555, 0.01011263578357973]
test_mse: [0.00015673858060601922, 9.08203816561399e-05, 9.041271980872986e-05, 0.00010226540249133723]
test_mae: [0.00793729541249432, 0.005836220796371801, 0.006889621140700348, 0.006017566202767015]
train_explained_variance: [0.9923478954532294, 0.9950992440705636, 0.9952703113483993, 0.995349398118429]
test_explained_variance: [0.9922894467381819, 0.9954256367567251, 0.9953373532007508, 0.9948503461189109]
train_r2: [0.9922467177359869, 0.9950624986541989, 0.9940215527459672, 0.9953147208216108]
test_r2: [0.9908126277285393, 0.9950448225216223, 0.995054860434776, 0.9943704399968747]
train_mgd: [0.012586897263705984, nan, 0.01372287692382917, nan]
test_mgd: [0.0014730275518917697, 0.0008362008118705646, 0.0010669859656560635, 0.0008373886643846441]
train_mpd: [0.0006051365407264417, nan, 0.0005350853063007385, nan]
test_mpd: [0.0004301136376393745, 0.00024485077080733074, 0.0002724164237139497, 0.0002609079185173583]

7.1.5- Training session 5 : Using 3 LSTM layers / 200 iterations

- File : *training_with_one_dimension/price_pred_3_lstm_layers.py*

- Logs :

Metrics History:
epoch: [50, 100, 150, 200]
train_rmse: [0.010515215128133356, 0.010405067145658914, 0.007605909973101585, 0.007465860332699156]
train_mse: [0.00011056974919092458, 0.00010826542230567054, 5.784986651892615e-05, 5.573907050737075e-05]
train_mae: [0.007219545207744362 5, 0.007465544979148417, 0.004786727362464304, 0.004736046451464648]
test_rmse: [0.016460441240417345, 0.011379160356225024, 0.014732872118296077, 0.017106281346959275]
test_mse: [0.00027094612582923206, 0.00012948529041268322, 0.00021705752085406595, 0.0002926248615213268]
test_mae: [0.010033286647867741, 0.008840866433518939, 0.007937544951938453, 0.009024936205169469]
train_explained_variance: [0.9908344018828181, 0.9937187108832589, 0.9952043481451829, 0.9952983384811371]

test_explained_variance: [0.9878162356919825, 0.9943441783772573, 0.98942610017869, 0.9850756806536429]
train_r2: [0.9905919267364298, 0.9915841999759545, 0.9951706692236256, 0.995273312926952]
test_r2: [0.9843718943799341, 0.9931768472490995, 0.9877166039290063, 0.9831783971258586]
train_mgd: [0.014110817682260801, nan, 0.006592751096036114, 0.006588739475719421]
test_mgd: [0.0022618836589239485, 0.0018029643873716495, 0.0013205487775998003, 0.001650309947631687]
train_mpd: [0.0007060830017425727, nan, 0.0003509795083636924, 0.0003471549507453385 4]
test_mpd: [0.0007015140678051856, 0.0004205700892764080 4, 0.000489539062041615 9, 0.0006456054423670947]

7.1.6- Training session 6 : Using 1 GRU layer / 200 iterations

- File : *training_with_one_dimension/price_pred_gru_layer.py*

- Logs :

Metrics History:
epoch: [50, 100, 150, 200]
train_rmse: [0.007984792252759134, 0.00869564237451769, 0.007696013475913473, 0.007440368250544646]
train_mse: [6.375690731972229e-05, 7.561419630550764e-05, 5.9228623421441785e-05, 5.5359079703712797e-05]
train_mae: [0.005162423660953451, 0.006395852414737629, 0.0051016931128350694, 0.004434947870538297]
test_rmse: [0.019796415378715817, 0.021732384863790772, 0.01822477122987262, 0.01722319672907656]
test_mse: [0.0003918980618466560 6, 0.000472296551867922 3, 0.000332142286381192 8, 0.000296638505568473 53]
test_mae: [0.010073458982660839, 0.01225902629828916, 0.009472107857170238, 0.008687721705716049]
train_explained_variance: [0.9950859446360342, 0.9953057976608428, 0.9953724642432428, 0.9953627275911889]
test_explained_variance: [0.978759129410766, 0.9803923644093137, 0.9820044676260318, 0.9851814181186708]
train_r2: [0.9946316373761604, 0.993622089133723, 0.9949577028039592, 0.9953551437749633]
test_r2: [0.9776309910587822, 0.9729947062411726, 0.980831858908019, 0.983127868009708]
train_mgd: [0.009541556730174056, nan, 0.01123667228821353, 0.019373610939554798]
test_mgd: [0.002074516981973052, 0.003011990151509001, 0.001786221664502844 2, 0.001574495753241377 2]
train_mpd: [0.0004148323645774634 5, nan, 0.000419622999948957 5, 0.000367106758930982 1]
test_mpd: [0.000841980755981953, 0.001084196768129512 2, 0.000717894694075275 3, 0.000639972317591232 4]

7.1.7- Training session 7 :Using the Huber Loss / 200 iterations

- File : *training_with_one_dimension/price_pred_huber_loss.py*

- Logs :

Metrics History:
epoch: [50, 100, 150, 200]
train_rmse: [0.011272175604553411, 0.009490546501852444, 0.008826024976733465, 0.008277674562648791]

train_mse: [0.0001270619428598890 8, 9.007047290382365e-05, 7.789871688992297e-05, 6.851989616512286e-05]
train_mae: [0.007441416605951829, 0.006161749884814625, 0.005607069169562461, 0.005146504208574214]
test_rmse: [0.02538707897167958, 0.01608681330467959, 0.009883317889538359, 0.010753943551148238]
test_mse: [0.0006445037787142956, 0.0002587855622996162 5, 9.767997250566894e-05, 0.00011564730190128277]
test_mae: [0.014622794174212703, 0.009194497171472147, 0.006386770639877577, 0.006829129440991411]
train_explained_variance: [0.9897679535808936, 0.9922435988881514, 0.9936700702440117, 0.9945072925287542]
test_explained_variance: [0.9709828962896251, 0.9866568238936559, 0.9946982709291436, 0.9946574153274017]
train_r2: [0.9883484950614027, 0.9922381125582403, 0.993669224989805, 0.9943707933526369]
test_r2: [0.9599363027189212, 0.9848823611281704, 0.9946186513574596, 0.993559419296903]
train_mgd: [0.06171927347614568, 0.011602726033662563, 0.011636486026361382, 0.007703180261555584]
test_mgd: [0.004442549615066085, 0.0018648316869639477, 0.0010201496615294143, 0.0010335072230605279]
train_mpd: [0.0009812039086994908, 0.0006349368547821565, 0.0005538010844959851, 0.0004435608373525904 4]
test_mpd: [0.0015797050119763698, 0.0006261423864817176, 0.0002772916018109998 4, 0.0003103191750027427]

7.1.8- Training session 8 : Improving Optimizer (0.001) / 200 iterations

- Optimizer learning rate modified to 0.001.
- File : *training_with_one_dimension/price_pred_learning_rate_optimizer.py*
- Logs :

Metrics History:
epoch: [50, 100, 150, 200]
train_rmse: [0.012696319711481 05, 0.011287198574108571, 0.010776636727073444, 0.012114753926241069]
train_mse: [0.000161196534216142 28, 0.000127400851651358 57, 0.000116135899147308 22, 0.000146767262693373 4]
train_mae: [0.009277193196089424, 0.007932683191663237, 0.007560172463442008, 0.008795504471389052]
test_rmse: [0.015418269085897548, 0.016090845590278586, 0.024970099072513213, 0.028244364434361232]
test_mse: [0.000237723021605144, 0.000258915311810187 83, 0.000623505847691125 3, 0.000797744122301009 7]
test_mae: [0.011415856072439567, 0.010663966900030953, 0.014878679687042616, 0.017987471788379897]
train_explained_variance: [0.9855962726693385, 0.9882198056729403, 0.9887120697175409, 0.9854938579945464]
test_explained_variance: [0.9853687928002848, 0.9860849509505839, 0.9669268841087079, 0.9630401663291256]
train_r2: [0.9851260672674488, 0.988200672550975, 0.9886646180328282, 0.9844902932864824]
test_r2: [0.9851303625200958, 0.9837444494514407, 0.9587456993552073, 0.9428525271596183]
train_mgd: [0.02159581811278286, 0.018567727343181822, 0.01714361277397074, 0.01639827295899706]

test_mgd: [0.003069629658886163, 0.002566051148492359, 0.004619410014988636, 0.005813134642361453]
train_mpd: [0.0012643706412482165, 0.0009846998727682603, 0.0008742441278797206, 0.0010556980409715288]
test_mpd: [0.0007569486868613817, 0.0007311680669758951, 0.0015784057761228272, 0.0020396445588590933]

7.1.9- Training session 9 : Improving Optimizer (0.0005) / 200 iterations

- Optimizer learning rate modified to 0.0005.
- File : *training_with_one_dimension/price_pred_learning_rate_optimizer.py*
- Logs :

Metrics History:
epoch: [50, 100, 150, 200]
train_rmse: [0.013372021124162277, 0.01231725298528413, 0.011073911975494365, 0.010345862386523639]
train_mse: [0.00017881094894504215, 0.00015171472110349078, 0.00012263152644099748, 0.000107070368685208846]
train_mae: [0.010071788841660887, 0.009466281268217797, 0.008064477088983068, 0.007322815957719237]
test_rmse: [0.03051867639226168, 0.031357918858490694, 0.0314093661323535, 0.032640802917475925]
test_mse: [0.0009313896087355904, 0.0009833190751356864, 0.000986548280836235, 0.001065422015097505]
test_mae: [0.018045279970890792, 0.017606003178461012, 0.01702117396294413, 0.016926247558597996]
train_explained_variance: [0.9808401097501364, 0.9844620664402287, 0.9878752331168151, 0.9898246596317847]
test_explained_variance: [0.9509423598128065, 0.9483252692514884, 0.946518862820849, 0.9390895540526326]
train_r2: [0.9807203534575635, 0.9844513152961876, 0.9878552602911977, 0.989604204454826]
test_r2: [0.9319237892928861, 0.9316844728590311, 0.9337686744467685, 0.9298536581966503]
train_mgd: [0.021003374641773386, 0.021377999421996852, 0.019421478886618666, 0.01837365990980938]
test_mgd: [0.006598409703465615, 0.006564180179981556, 0.006379012979629847, 0.006613030557455871]
train_mpd: [0.0014049604774145684, 0.0012795113365712113, 0.001028753384506424, 0.0009171924447363163]
test_mpd: [0.0023707563618547145, 0.0024198602571444167, 0.0023842351345039698, 0.0025310790771866315]

7.1.10- Training session 10 : Improving Optimizer (0.0001) / 200 iterations

- Optimizer learning rate modified to 0.0001.

- File : *training_with_one_dimension/price_pred_learning_rate_optimizer.py*

- Logs :

Metrics History:
epoch: [50, 100, 150, 200]
train_rmse: [0.01618455256267452, 0.01578534508937717, 0.014986021565320628, 0.014811373666831745]
train_mse: [0.0002619397416539744, 0.0002491771195907239 4, 0.00022458084235625492, 0.00021937678989851687]
train_mae: [0.01109494472251996, 0.01081199571973356, 0.010662285931012986, 0.010252295613448387]
test_rmse: [0.02488319005760237, 0.02363279876443177, 0.02208909380176205, 0.022112632240085243]
test_mse: [0.0006191731474427614, 0.0005585091774401278, 0.0004879280649830425 4, 0.0004889685045852574]
test_mae: [0.01639318681207219, 0.015244351087395303, 0.014224118020286143, 0.014707270527549911]
train_explained_variance: [0.9716470559527537, 0.9739695286846168, 0.9768413721269255, 0.9784949378850523]
test_explained_variance: [0.9665443101837253, 0.9704017893582819, 0.9722296903546235, 0.977171325419807]
train_r2: [0.9713458786825959, 0.9737647597286637, 0.9768332954396013, 0.9774784990097567]
test_r2: [0.9540847620083146, 0.9601373446771614, 0.9658802077203614, 0.965971205316273]
train_mgd: [0.02235400639470177, 0.024259840743978512, 0.02256529200844175, 0.022369019549563225]
test_mgd: [0.005968168629452848, 0.005341041074843814, 0.004504044325255973, 0.004998092812615461]
train_mpd: [0.0017239330763388525, 0.0017048147426359794, 0.0015920075751810806, 0.001521244395789562]
test_mpd: [0.0018090161219973613, 0.001612542121535953, 0.001372320966380275, 0.0014360184821930449]

7.1.11- Training session 11 : Using attention layer / 200 iterations

- File : *training_with_one_dimension/price_pred_attention_layer.py*

- Logs :

Metrics History:
epoch: [50, 100, 150, 200]
train_rmse: [0.00962696729974938, 0.007961449989456427, 0.007682311906051299, 0.007974301414665867]
train_mse: [9.267849939044387e-05, 6.338468593461574e-05, 5.9017916221857546e-05, 6.358948305194205e-05]
train_mae: [0.006451637509334341, 0.00513395201137284, 0.004811148125195726, 0.00532886550067779]
test_rmse: [0.010788180385519323, 0.013819450630459643, 0.015894029803791534, 0.010822479556789331]
test_mse: [0.0001163848360305038 5, 0.00019097721572771144, 0.0002526201834038135 7, 0.00011712606375712301]

test_mae: [0.007319260330180446, 0.008083865431325282, 0.008609318133969044, 0.007072901168058565]
train_explained_variance: [0.992494444340444, 0.9950341325945421, 0.994991876068588, 0.9953128642698484]
test_explained_variance: [0.9934350502172861, 0.9893741759886518, 0.9875681307586267, 0.9936649850108982]
train_r2: [0.9922881907491937, 0.9947265626250713, 0.9949253254364838, 0.9949228209973682]
test_r2: [0.9934350305334686, 0.989229148619743, 0.9852751324576909, 0.993660576652554]
train_mgd: [0.01121457726982406, 0.007056269307945314, 0.0078146026639907, 0.09057461470983813]
test_mgd: [0.0013802953135379947, 0.0014265215037312109, 0.0015600720662053537, 0.0011010139066908503]
train_mpd: [0.0005875577539072375, 0.0003830235330264116, 0.0003851451169154168 4, 0.0004650453914131877]
test_mpd: [0.0003524065155441724 4, 0.0004648298708097929 7, 0.0005814178233962792, 0.0003114374865834 36]

7.2- Training sessions with a multi-dimensional dataset

7.2.1- Training session 1 : Using 1 LSTM layer / 400 iterations

- File : *training_with_many_dimensions/price_pred_attention_layer.py*

- Logs :

Metrics History:
'test_explained_variance': [-11023879.171073196, -328053.23832580796, -88587.92860180166, -14748.678971060835, -1047.1439928441878, -425.8878515510033, -2.0783774288207324, 0.349893442942692],
'test_mae': [2270633.2792832367, 87045.4244993842, 39852.8595834515, 13018.458146194032, 2206.4086353609255, 850.1957769637503, 16.972178002880568, 14.973482410554723],
'test_mgd': [nan, nan, nan, nan, nan, nan, nan, nan],
'test_mpd': [nan, nan, nan, nan, nan, nan, nan, nan],
'test_mse': [7809567616985.119, 12581211701.776001, 2234255405.5140867, 213381607.81500512, 5832272.152577031, 856024.3162006092, 323.8625055899725, 243.6206657926965],
'test_r2': [-20967167.93693807, -668299.1003146156, -243542.5698283022, -66666.73041894507, -6340.131967056992, -2742.6172661289384, -2.376562438210196, -7.157372792150525],
'test_rmse': [2794560.362022105, 112166.00064982258, 47267.91094933313, 14607.587337236944, 2415.009762418577, 925.2158214171487, 17.99618030555297, 15.608352436842798],
'train_explained_variance': [-8293671.643782305, -851356.7867961829, -419352.430768093, -127703.99661665654, -28007.75030439711, -6371.533797737195, -1574.7381241376454, -401.4256766271226],
'train_mae': [1033965.1167560347, 76894.09241449945, 36864.53287565569, 11510.883695743407, 2514.689530339964, 741.8392461808809, 94.19293502746328, 34.31436922670187],
'train_mgd': [nan, nan, nan, nan, nan, nan, nan, nan],
'train_mpd': [nan, nan, nan, nan, nan, nan, nan, nan],
'train_mse': [2267414207552.1763, 10764110138.7626, 2600796338.048496, 275563037.51081204, 17493693.90455309, 1366208.4011855612, 104823.75473570406, 8067.2870856310055],
'train_r2': [-9069597.43908858, -851865.5775430726, -422370.48017085245, -128268.68880593452, -28336.068741620493, -6522.771901265238, -1627.2392354628705, -401.44643705411056],
'train_rmse': [1505793.5474533606, 103750.22958414405, 50998.00327511358, 16600.091491037394, 4182.546342188343, 1168.8491781173313, 323.7649683577642, 89.8180777217538]

7.2.2- Training session 2 : Using 2 LSTM layers / 400 iterations

- File : *training_with_many_dimensions/price_pred_2_lstm_layers.py*

- Logs :

Metrics History:
'epoch': [50, 100, 150, 200, 250, 300, 350, 400],
'test_explained_variance': [-299316.7790482659, -148846.6635830141, -62.5019106262855, -42.386415120075704, -18.217597743089446, -6.855864744353218, -0.018614436115373945, -5.698224249672812e-05],

'test_mae': [49516.44997865986, 21522.3165158517, 330.1002421387555, 93.38350933644357, 9.985866961871874, 1.282505240837545, 0.6381354884344943, 0.30644947599696587],
'test_mgd': [nan, nan, nan, nan, nan, nan, nan, nan],
'test_mpd': [nan, nan, nan, nan, nan, nan, nan, nan],
'test_mse': [4266370330.788918, 936330665.1939946, 446834.3166650742, 32748.90869001628, 280.966415054014, 4.208392074841843, 0.6630803069410649, 0.1768829629401455],
'test_r2': [-340381.3916316733, -173721.0726770322, -62.860847910670344, -43.2488534861122, -19.921422297167805, -9.042907658270458, -1.5871709729263794, -0.8255258266786345],
'test_rmse': [65317.45808578988, 30599.520669350273, 668.4566677542189, 180.9665955087189, 16.762052829352793, 2.0514365880625807, 0.8142974314960504, 0.42057456287814826],
'train_explained_variance': [-134477.80747389366, -46053.172864678134, -3261.8943910431694, -769.2106730207712, -6.644474135016157, -0.9891807415407385, -0.17394938989511655, -0.05293091135156147],
'train_mae': [20283.14815735243, 7042.117180221996, 393.80177854449704, 80.23319826263035, 5.329448434237136, 0.7052457425197411, 0.38503548523289816, 0.23003957568270209],
'train_mgd': [nan, nan, nan, nan, nan, nan, nan, nan],
'train_mpd': [nan, nan, nan, nan, nan, nan, nan, nan],
'train_mse': [1132361345.1369677, 166615211.21975094, 15363504.400717106, 386844.8334054935, 85.37653652259114, 0.8289546139875855, 0.23889488060381642, 0.06858728276164684],
'train_r2': [-134596.79985830653, -46054.849124790344, -3270.3185834291976, -777.7290593552631, -8.471520829607186, -1.9472597899358144, -0.3887057497368127, -0.05460533770771048],
'train_rmse': [33650.57718876405, 12907.951472629224, 3919.630645955956, 621.9685148023922, 9.239942452341959, 0.910469447036849, 0.48876873938890203, 0.2618917386280958]

7.2.3- Training session 3 : Using 2 LSTM layers / 650 iterations

- File : *training_with_many_dimensions/price_pred_2_lstm_layers.py*

- Logs :

Metrics History:
'epoch': [50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600, 650],
'test_explained_variance': [0.0, 0.0, 0.0, 0.0, -2.220446049250313e-16, 0.0, 0.0, 0.0, -2.220446049250313e-16, 0.0, 0.0, 1.1102230246251565e-16, 1.1102230246251565e-16],
'test_mae': [21.571715017896846, 18.54198576742946, 27.86777129066857, 0.044236799508658144, 3.5475461927879716e-06, 7.228736274793937e-06, 0.000582830318474463, 0.05058308928290803, 0.015155180240594153, 0.20813823070869478, 0.20720524006084237, 0.20530829533620665, 0.2059437451139798],
'test_mgd': [nan, nan, nan, nan, 7.682211250470417e-10, 1.9153640473668078e-09, 1.1393140009752223e-05, 0.7429373156692156, 0.01133621163304216, 0.22656372981877165, 0.22307335891310018, 0.21611558067860664, 0.21844472276661459],
'test_mpd': [nan, nan, nan, nan, 1.262727829243193e-10, 4.3151828645060507e-10, 2.7161595016753083e-06, 0.05765477610640961, 0.0022280426061851706, 0.13428244205835774, 0.132826538363596, 0.12989859418508218, 0.13088260221980794],
'test_mse': [596.3308451930047, 446.0732849762755, 1025.7367148864319, 0.0027467742108378897, 2.0755502578093966e-11, 9.721808881490211e-11, 6.475423607474667e-07, 0.0048844855022418, 0.00043888760363585707, 0.082281982596056, 0.08174057930637005, 0.08064112793577059, 0.08101222658438893],

'test_r2': [-3.5524233828883682, -3.361804985154107, -3.1173735075534017, -2.477458570466364, -1.5050834760513476, -0.8899896138021524, -0.7643273863181201, -0.7574076375137846, -0.752685630397744, -0.7821872947679807, -0.7704607656569753, -0.7466471894392397, -0.7546850037432529],
'test_rmse': [24.41988626494818, 21.120447082774444, 32.02712467403891, 0.05240967669083534, 4.5558207359480205e-06, 9.859923367597849e-06, 0.000804700168228804 8, 0.06988909430119838, 0.02094964447516609, 0.28684836167573974, 0.28590309425812455, 0.2839738155812444, 0.28462646852390405],
'train_explained_variance': [-205.90258984920365, -76.13113065892306, -364.98285029096786, 0.0037455380202849886, -2.220446049250313e-16, 0.0, 4.5893332109248064e-05, 0.00351607679035304, 0.0023004253005535213, -2.220446049250313e-16, -2.220446049250313e-16, -2.220446049250313e-16, -2.220446049250313e-16],
'train_mae': [28.256753558218158, 16.03622617388491, 28.93445678122617, 0.022530703941198504, 2.094244112692533e-06, 5.190629859555092e-06, 0.0004431954887608507 6, 0.03849496078358711, 0.011576207803406572, 0.15694227057241386, 0.15710693671220693, 0.15744523358962628, 0.15733045275975102],
'train_mgd': [nan, nan, nan, nan, 2.4434181825047325e-10, 6.853385799581806e-10, 4.348263203461245e-06, nan, 0.004803048146094191, 0.13821679075603346, 0.1373126862206483, 0.13553876962083222, 0.13612829132098941],
'train_mpd': [nan, nan, nan, nan, 4.0162259231283996e-11, 1.5439914766026617e-10, 1.0349539321279007e-06, nan, 0.000896850224602356 1, 0.06421526728539755, 0.06400720241663438, 0.06361051752395959, 0.06374052079377136],
'train_mse': [18781.987731477962, 5468.696723784948, 61986.90764391833, 0.0009183992074119051, 6.601436496652143e-12, 3.478440843279812e-11, 2.463353615738689e-07, 0.001858969575368473, 0.00016769326986936571, 0.030995533283515675, 0.03099024638738825, 0.030992214544216107, 0.03098961229330705],
'train_r2': [-212.61970671205887, -78.66879710556691, -369.7052413680276, -0.7322672471834732, -0.18705758831043617, -0.0074914069091041036, 4.112091307273413e-05, 0.003515937897040744, 0.0022752739974274405, -0.0002120881510121464, -4.1482361878619756e-05, -0.0001049939082873319, -2.102042759788425e-05],
'train_rmse': [137.04739228266243, 73.95063707490928, 248.97170048806416, 0.030305102002994563, 2.5693260783038308e-06, 5.897830824362303e-06, 0.0004963218326588797, 0.043115769451193525, 0.012949643619396083, 0.17605548353719538, 0.1760404680389945, 0.17604605801953108, 0.1760386670402473]

7.2.4- Training session 4 : Using 3 LSTM layers / 800 iterations

- File : *training_with_many_dimensions/price_pred_3_lstm_layers.py*

- Logs :

Metrics History:
'epoch': [50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600, 650, 700, 750, 800],
'test_explained_variance': [-448415.7274160662, -115954.2792433329, -31834.1721678186, -11025.967276744697, -7641.04757747607, 0.0, 0.0, 0.0, 0.0, 0.0, 1.1102230246251565e-16, -2.220446049250313e-16, -2.220446049250313e-16, 3.3306690738754696e-16, 3.3306690738754696e-16, 0.0],
'test_mae': [59219.374556678835, 19923.940885033935, 5014.139876138477, 1307.0899300492995, 893.4489652966338, 0.3746197839905093, 0.3733598461080859, 5.806667759284453e-07, 0.00014032603356695046, 0.00011504809340179507, 0.00010440377272381386, 5.106575328573785e-05, 7.313200446187215e-05, 9.444303826473603e-05, 0.00010099762979135517, 0.00011472488299847393],
'test_mgd': [nan, nan, nan, nan, nan, 4.209807989099013, 4.064895848485795, 1.3525963238336658e-10, 6.190969050020034e-06, 2.5506447137618736e-06,

1.0411184870035007e-06, 1.2351960686274312e-07, 1.848417090288823e-07, 2.94910348471733e-07, 3.3715491390418073e-07, 4.391603859660309e-07],
'test_mpd': [nan, nan, nan, nan, nan, 0.7939605950747272, 0.7800144837344933, 7.817465182136108e-12, 4.060097868344159e-07, 2.1687302788332003e-07, 1.3071955260984814e-07, 2.369447443736905e-08, 4.319100256014452e-08, 7.083463366440446e-08, 8.100035230537706e-08, 1.0488877455631928e-07],
'test_mse': [13911357411.652145, 1264479389.6046817, 65418966.89747297, 4906279.85566482, 1783084.5488808723, 0.18650908208867353, 0.18556667421743128, 4.518181819448915e-13, 2.6626541719899807e-08, 1.8440015657136498e-08, 1.6412737463566916e-08, 4.545255175359329e-09, 1.0092217747948383e-08, 1.701379987720622e-08, 1.9460068021066314e-08, 2.5051567569209157e-08],
'test_r2': [-516644.5732093713, -131473.0925262145, -34561.75902662211, -11489.4157416175, -9822.371049448424, -3.0396950337120936, -3.019282942490828, -2.941044356276433, -2.8393628094592005, -2.5434639187729395, -1.9773188905874601, -1.2062730080031958, -0.8155374307742991, -0.7612371703425427, -0.760565653418297, -0.7717655853483725],
'test_rmse': [117946.41754479932, 35559.51897319031, 8088.199237004054, 2215.0123827339703, 1335.321889613464, 0.43186697267639435, 0.4307745050689877, 6.721742199347513e-07, 0.00016317641287851565, 0.00013579401922447282, 0.0001281122065361725, 6.741850766191232e-05, 0.00010046003059898192, 0.00013043695748217305, 0.00013949934774423254, 0.00015827686997539836],
'train_explained_variance': [-185002.03190766557, -62834.39212940309, -11526.407434243356, -3252.8895841755443, -2023.8942268645908, 5.022183824365811e-08, 5.80358644475254e-08, 0.0, 6.548178728260634e-05, 4.093863414555976e-05, 4.7507790512879566e-05, 1.5364460873690255e-05, 4.290155123631845e-05, 5.5041342550299355e-05, 6.431421270725046e-05, 8.16760804912553e-05],
'train_mae': [32986.77249684182, 11660.540556981041, 2641.558548903715, 688.5018793626339, 382.01017601820854, 0.19604608643424476, 0.19506125993223225, 3.0147399143549885e-07, 7.233876666448534e-05, 5.859312904130192e-05, 5.628032951332529e-05, 3.248946608393971e-05, 5.4295993238118536e-05, 7.19220207165751e-05, 7.693799328663351e-05, 8.692977204303937e-05],
'train_mgd': [nan, nan, nan, nan, nan, 1.9204327478493581, 1.8492388003606373, 4.739552486093458e-11, 2.1486706533527304e-06, 8.591934847732853e-07, 3.3444288478897567e-07, 4.044174363266135e-08, 6.845559500569467e-08, 1.1244290731839651e-07, 1.2860242886878356e-07, 1.6647015281271263e-07],
'train_mpd': [nan, nan, nan, nan, nan, 0.32999112651388973, 0.3232607608905933, 2.7392669394859037e-12, 1.4085566233159648e-07, 7.30344097614727e-08, 4.198274218176193e-08, 7.756952420949457e-09, 1.5992502511955064e-08, 2.7000612583145014e-08, 3.088762544762793e-08, 3.974700312681156e-08],
'train_mse': [3372312362.1317782, 408612106.32631254, 14666558.204010889, 986605.2788525192, 271331.9595015741, 0.06601391528671983, 0.0655439090480802, 1.5831836159606468e-13, 9.233767224891577e-09, 6.208177663923343e-09, 5.270109891555614e-09, 1.4878268362980442e-09, 3.736146645777161e-09, 6.483584611507041e-09, 7.418564716730841e-09, 9.490135661180734e-09],
'train_r2': [-186592.01785651318, -63296.13410997916, -11543.558331210052, -3441.4786127777766, -2226.0704395753746, -1.1302397171875609, -1.1150728246842543, -1.0574216062983353, -0.9836621319196621, -0.7773594055732258, -0.4243226868230421, -0.07596404777462373, -0.0013522995628834078, 5.5040529091998636e-05, 6.412246417863976e-05, 2.7247675257968673e-05],
'train_rmse': [58071.61408237056, 20214.15608741341, 3829.6942703055147, 993.2800606337164, 520.8953440966565, 0.2569317327359932, 0.2560154468935033, 3.978923995203536e-07, 9.609249307251622e-05, 7.879198984619783e-05, 7.259552253104604e-05, 3.8572358448739484e-05, 6.112402674707517e-05, 8.052070920891743e-05, 8.613109030269407e-05, 9.741732731491217e-05]

7.2.5- Training session 5 : Using attention layer / 800 iterations

- File : *training_with_many_dimensions/price_pred_attention_layer.py*

- Logs :

Metrics History:

'epoch': [50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600, 650, 700, 750, 800],

'test_explained_variance': [1.1102230246251565e-16, 1.1102230246251565e-16, 1.1102230246251565e-16, 1.1102230246251565e-16, 0.0, 0.0, -2.220446049250313e-16, -2.220446049250313e-16, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.1102230246251565e-16, 1.1102230246251565e-16],

'test_mae': [0.5098898058463406, 0.5083614384461712, 0.5043391829062771, 0.4941251894165348, 0.4691809771288703, 0.41364521269542676, 0.31562900745166045, 0.22671807669331512, 0.20682039701789187, 0.20632507720440196, 0.2071653368373278, 0.2065553463384466, 0.20690719106643962, 0.20693375909179018, 0.2058094679903101, 0.20591891925789022],

'test_mgd': [nan, nan, nan, nan, nan, 23.69813757906287, 1.2093554484792852, 0.30708631939387265, 0.22165067390260618, 0.21983347548536492, 0.22292561467985858, 0.22067589221019587, 0.22197107439525487, 0.22206926869160198, 0.21795553978953816, 0.21835418170496707],

'test_mpd': [nan, nan, nan, nan, nan, 1.6963068717595513, 0.399436190384851, 0.16579507301275695, 0.13223065919750007, 0.13146745477898641, 0.13276472339000098, 0.13182155330755319, 0.1323649808524148, 0.13240613235533233, 0.13067625969569938, 0.1308444240624227],

'test_mse': [0.30615671363759644, 0.3046004516306378, 0.30052711094614876, 0.2903288023475039, 0.26629988883117917, 0.2172714615174225, 0.14579076987649797, 0.09321454491588443, 0.08151798371274341, 0.08123202114522794, 0.08171751512675667, 0.08136481777216106, 0.08156821203060077, 0.0815835942596705, 0.08093454376489027, 0.08099785884972531],

'test_r2': [-5.631203916554592, -5.597496046512765, -5.509269489663782, -5.288379138712284, -4.76792468410699, -3.7059930499363185, -2.1577564075466427, -1.018981220375108, -0.7656394545227956, -0.7594456458841872, -0.7699612068644628, -0.7623219555432545, -0.7667273751962977, -0.7670605467163358, -0.7530024320603135, -0.7543738056733451],

'test_rmse': [0.5533142991443438, 0.5519061982172675, 0.5482035305852643, 0.538821679544823, 0.5160425261847895, 0.4661238692852175, 0.38182557519959026, 0.3053105712481709, 0.28551354383416455, 0.28501231753246725, 0.2858627557531003, 0.28524518886768463, 0.2856014916463161, 0.2856284199089273, 0.2844899712905365, 0.2846012277726948],

'train_explained_variance': [0.0, 0.0, -2.220446049250313e-16, -2.220446049250313e-16, -2.220446049250313e-16, 0.0, 0.0, -2.220446049250313e-16, -2.220446049250313e-16, -2.220446049250313e-16, -2.220446049250313e-16, -2.220446049250313e-16, -2.220446049250313e-16, -2.220446049250313e-16, -2.220446049250313e-16, -2.220446049250313e-16],

'train_mae': [0.32241957364082147, 0.3208912062406521, 0.316868950700758, 0.3066549572110157, 0.28171074492335135, 0.22795186238287243, 0.1656060653381717, 0.1543609811691997, 0.15717500175976734, 0.15726275837703213, 0.15711397934419533, 0.15722196121989465, 0.1571596243169681, 0.1571549172166893, 0.15735442928366475, 0.15733488529348055],

'train_mgd': [nan, nan, nan, nan, nan, 12.155421670509442, 0.5117600966545099, 0.16118083858439086, 0.1369468522875891, 0.13648187474246876, 0.13727462175856447, 0.13669710556499023, 0.1370291038440703, 0.1370543278000301, 0.1360041105382784, 0.1361052927274918],

'train_mpd': [nan, nan, nan, nan, nan, 0.7923903452221535, 0.1507160362081277, 0.07015826721826511, 0.06392408563644379, 0.06381939206776645, 0.06399852445479717,

0.06386771823052505, 0.06394271686576562, 0.06394843706305214, 0.06371297787556875, 0.06373541351074467],
'train_mse': [0.1349433423588274, 0.13396012713467906, 0.1313948928102977, 0.1250262236741882, 0.11034990469736783, 0.08214408269170846, 0.04741363256468929, 0.03185504978901378, 0.0309892554774648, 0.030989008972875317, 0.03099011145399103, 0.030988979113565022, 0.03098941825552124, 0.03098947521110877, 0.030989991712632038, 0.030989675904397922],
'train_r2': [-3.354561704366019, -3.3228337859123522, -3.24005481396446, -3.0345406904581944, -2.56094239757182, -1.650753052925177, -0.5300168576087705, -0.02794830391155001, -9.506138897164007e-06, -1.5515453131165202e-06, -3.712812109935015e-05, -5.879986362256062e-07, -1.4758914458345984e-05, -1.6596845963112727e-05, -3.326412064419415e-05, -2.3073129238770917e-05],
'train_rmse': [0.36734635204235716, 0.3660056381187031, 0.3624843345722649, 0.35359047452411413, 0.33218956139133543, 0.2866078901421042, 0.21774671654169506, 0.17847983020222138, 0.17603765357861595, 0.17603695342988449, 0.17604008479318292, 0.17603686862008489, 0.17603811591675605, 0.17603827768729383, 0.17603974469599767, 0.17603884771378708]

7.2.6- Training session 6 : Using 1 GRU layer / 400 iterations

- File : *training_with_many_dimensions/price_pred_gru_layer.py*

- Logs :

Metrics History:
'epoch': [50, 100, 150, 200, 250, 300, 350, 400],
'test_explained_variance': [0.5076646706015704, 0.5102175642405833, 0.4921185607138244, 0.5344406958530186, 0.5059571732309109, 0.6823620677581839, 0.7217479208603879, 0.8814019105302545],
'test_mae': [0.024467797395432547, 0.024548679077153874, 0.02408569417982809, 0.025278892154945862, 0.02479991916661909, 0.027885626310183723, 0.029546773477137717, 0.024761287090115228],
'test_mgd': [0.009361058059909841, 0.009425973924295799, 0.009623518507637159, 0.010010717082697304, 0.011567697359336613, 0.01199829689002329, 0.019337714190012206, 0.014436309535702345],
'test_mpd': [0.0030381686261097603, 0.0030591342838797962, 0.0030608034518904027, 0.0032495762927439442, 0.003532221052122079, 0.003936243548056858, 0.005708942091508876, 0.004150669575141331],
'test_mse': [0.0009934479074522875, 0.0010003944486441153, 0.0009800018793784014, 0.0010642697592733146, 0.0010862407758931453, 0.0013241154803993768, 0.0017287637483392377, 0.0013393372672373888],
'test_r2': [0.4340489709370602, 0.4371788443800374, 0.39696956077648626, 0.46728370896002847, 0.37597146155073036, 0.6589985013432396, 0.6248464488170973, 0.86753696076996],
'test_rmse': [0.031519008668615955, 0.03162901276745949, 0.0313049817022531, 0.03262314759910997, 0.032958167059063605, 0.03638839760692104, 0.041578404831585807, 0.036596957076202224],
'train_explained_variance': [0.5364908602003278, 0.5390501376030793, 0.5208601404518133, 0.5632232278100129, 0.5347778889334833, 0.7052075723389035, 0.7406835973222999, 0.8577361795441959],
'train_mae': [0.024065485827722905, 0.02416026664838443, 0.02299143618925995, 0.025061341906448042, 0.022625250846331053, 0.030914971702944757, 0.026221905789217 91, 0.02840727669804315],
'train_mgd': [0.01023388444270636, 0.010345344285630659, 0.009596682367964886, 0.011441883395219822, 0.010306884813926068, 0.020178053475152072, 0.022844982521921 14, 0.040789764956684044],

'train_mpd': [0.0027500171363187987, 0.002775184784407235, 0.0025413397681897823, 0.0030200129980584725, 0.0025779721227929832, 0.0048293451172379, 0.004377638652214144, 0.00603525329611859],
'train_mse': [0.00074291016125014 2, 0.0007484908191122791, 0.00067657325258815, 0.0008022176599258734, 0.0006495185510057772, 0.0011747387182939834, 0.0008795637084510774, 0.0011004738681799993],
'train_r2': [0.3694573850903885, 0.37262095535460094, 0.37974319036431603, 0.4017525381183852, 0.4440774382908953, 0.5492703810229892, 0.7156291298233257, 0.8378455443630433],
'train_rmse': [0.02725637835902162, 0.027358560252913146, 0.026011021752098667, 0.028323447175897805, 0.025485653827315814, 0.03427446160472814, 0.02965743934413552, 0.03317339096595341]

7.2.7- Training session 7 : Using 1 GRU layer / 800 iterations

- File : *training_with_many_dimensions/price_pred_gru_layer.py*

- Logs :

Metrics History:
'epoch': [50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600, 650, 700, 750, 800],
'test_explained_variance': [-1168026.1062801168, -100341.49791249675, -21452.468592839665, -2287.031999081503, -96.87792841644884, -50.35218020681733, -86.88820868794114, -550.3510110444972, -49.660947927576274, -30.93475820941548, -31.636194273270377, -19.266079698640713, -77.46604996972029, -15.85814681379967, -368.8139320999337, -12.478020808132909],
'test_mae': [209837.86353803123, 20830.619440591334, 4540.438207849245, 609.2329121148521, 103.2467283764213, 70.98036651457633, 254.33196587738843, 661.3144429636802, 81.77080991017925, 45.73242238087336, 50.025619809802926, 35.32184082435333, 137.27256130434037, 27.7449645599364, 332.36794873339727, 35.78588962140604],
'test_mgd': [nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan],
'test_mpd': [nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan],
'test_mse': [104646668540.3865, 914448977.8864026, 39990322.95506886, 750901.538773723, 20607.842616468635, 10362.34570348677, 79669.56498347939, 523937.7885659149, 12950.630411117652, 4693.6592086330265, 5503.348405390716, 2296.0334841813296, 26173.012363425285, 1485.2473223801258, 138764.6589154162, 1907.87965435468],
'test_r2': [-1551755.9073458547, -130983.91621525798, -22435.495052752405, -2743.045272350416, -104.59991660981754, -55.38868324075292, -465.78468794733277, -3334.684835982423, -93.60208967846094, -36.929569515958285, -47.23877093939086, -21.039098595292554, -279.20481458500564, -16.308238911059952, -1812.5683960437077, -25.731519181789565],
'test_rmse': [323491.3732086012, 30239.857438261883, 6323.790236485462, 866.545751113998, 143.55431939328275, 101.79560748621117, 282.25797594307124, 723.8354706464135, 113.80083660113247, 68.51028542221253, 74.18455638062895, 47.91694360225128, 161.78075399572498, 38.53890660592391, 372.5112869637861, 43.67928175181776],
'train_explained_variance': [-2003094.6529635151, -167278.38283141848, -41872.57604335823, -3362.170056346079, -290.1592377458277, -249.63118619094763, -335.68231151121853, -1016.7591151432772, -197.6560568320698, -166.6135139039807, -156.13125431819435, -139.39873602652705, -180.63621468537042, -114.52797589604768, -497.2917609478407, -99.25488669716837],
'train_mae': [211503.6283543705, 19956.27167635407, 4940.992013382762, 623.539086006577, 128.78645496148928, 90.48322886477771, 202.87384908760154, 445.1705005234048,

71.68909468287117, 57.92292810226538, 53.69282074040338, 50.77574168433683, 86.96966192294187, 40.61765353452205, 200.6454209714257, 40.13725486575088],
'train_mgd': [nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan],
'train_mpd': [nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan],
'train_mse': [91295478005.4308, 813052811.7023458, 50140417.393160395, 620746.5232082722, 40730.411353836134, 31026.576000069163, 70911.03066530214, 289234.9664941694, 19161.51751472872, 13921.96600305375, 12177.902666209113, 10239.364503502324, 16898.150593340088, 6802.699206464454, 63077.70026530058],
'train_r2': [-2016934.162912684, -173509.17755717412, -41910.42602439728, -3378.612938389334, -229.9526289359792, -250.54293128185046, -617.9881624281581, -2742.4716283914263, -207.53747512798637, -166.614397093696, -158.03258380830115, -145.43099190879462, -268.52896861453695, -117.10812141001654, -1227.2175761146716, -112.69195542004108],
'train_rmse': [302151.4156932428, 28514.080937360508, 7080.989859699023, 787.8746875032045, 201.81776768618795, 176.14362321716095, 266.2912515748539, 537.8056958550824, 138.42513324800782, 117.99138105409968, 110.35353490581583, 101.18974505107879, 129.99288670285034, 82.47847723172666, 251.15274289822236, 73.79999642343785]

7.2.8- Training session 8 : Using the Huber Loss / 400 iterations

- File : *training_with_many_dimensions/price_pred_huber_loss.py*

- Logs :

Metrics History:
'epoch': [50, 100, 150, 200, 250, 300, 350, 400],
'test_explained_variance': [0.0, 0.0, 0.0, 0.0, 0.0, -2.220446049250313e-16, 0.0, 0.0],
'test_mae': [21.73716514024595, 13.343932948682045, 6.30554709088982, 1.5935064257957199, 0.2909685836888379, 0.16894320173030933, 0.1022747963036264, 0.07219695092961108],
'test_mgd': [nan, nan, nan, nan, nan, nan, nan, nan],
'test_mpd': [nan, nan, nan, nan, nan, nan, nan, nan],
'test_mse': [879.1520948286146, 341.3810468819971, 73.70393691249772, 4.902874180008661, 0.16356448358399725, 0.054969954506854685, 0.020088477814318356, 0.010015753374839413],
'test_r2': [-0.4521430269681368, -0.5618398996814198, -0.4432937700132489, -0.6253653931865177, -0.6558896290301874, -0.7020146882142111, -0.7236499567687038, -0.7199310108831893],
'test_rmse': [29.6504990654224, 18.476499854734314, 8.585099703119221, 2.2142434780323192, 0.4044310615964076, 0.23445672203384293, 0.14173382734660894, 0.10007873587750503],
'train_explained_variance': [-895.1579325415429, -313.6243287477217, -385.15268327038865, -3.143380836186741, -0.03656515057144549, 0.01845349843487054, 0.0338145073256404, 0.03827416030471986],
'train_mae': [106.41534352471314, 38.25772449079538, 26.5479247244144, 1.5902002039096612, 0.2301350716801041, 0.13036844567138098, 0.07786053497826467, 0.05484983142693134],
'train_mgd': [nan, nan, nan, nan, nan, nan, nan, nan],
'train_mpd': [nan, nan, nan, nan, nan, nan, nan, nan],
'train_mse': [365595.8481055726, 46247.329178201406, 13652.046477142838, 8.39925377225689, 0.06875033288610925, 0.02127789842257995, 0.007558632677687679, 0.003759062167716836],
'train_r2': [-898.6861271650915, -314.23044480921794, -397.29623309285495, -3.148445164193898, -0.03695897550894989, 0.018453117990949286, 0.03374962794778147, 0.0382741189557505],
'train_rmse': [604.6452249919556, 215.05192205186498, 116.84197224089826, 2.8981466098623945, 0.2622028468306728, 0.14586945678441374, 0.08694039727127821, 0.06131119121104104]

7.2.9- Training session 9 : Improving Optimizer / 400 iterations

- File : *training_with_many_dimensions/price_pred_learning_rate_optimizer.py*

- Logs :

Metrics History:
'epoch': [50, 100, 150, 200, 250, 300, 350, 400],
'test_explained_variance': [-343.8716712668835, -0.2287400785106637, 0.862543274144937, 0.8001187811155255, 0.6809402450960955, 0.7170643306867667, 0.593167648582323, 0.5487460614337132],
'test_mae': [204.36793311927732, 0.39406809981189556, 0.06870779406904648, 0.05835384208961065, 0.2056222260893594, 0.046254103113744824, 0.04198069982397545, 0.037778476029621234],
'test_mgd': [nan, 0.3109262446040086, nan, 0.2012469802714287, 0.3359723176711678, 0.07846467518021913, 0.06561303250837489, 0.05096533364804061],

'test_mpd': [nan, 0.24195251051705205, nan, 0.028370096844494652, 0.13161352622570405, 0.016518183624197692, 0.013890414476694715, 0.01146123307886069],
'test_mse': [51965.20981452688, 0.27371482847639095, 0.007738475814010606, 0.004754059067557561, 0.06044993476319485, 0.0036743161340949126, 0.003020114331514865, 0.0026248124628637324],
'test_r2': [-1756.1725695933942, -1.5738228908286525, 0.652972222112885, 0.29553306604617147, -0.06036417370555869, 0.3251303017795486, 0.02467172803741635, 0.02649932313771053],
'test_rmse': [227.9587897285974, 0.5231776261236627, 0.087968606980050 6, 0.06894968504320785, 0.24586568439535203, 0.06061613757156516, 0.05495556688375496, 0.051232923622059015],
'train_explained_variance': [-401.5467382429675, -0.5385263127813764, 0.8633075450184226, 0.8196107450936988, 0.5923590563569032, 0.7275541601591036, 0.613841909063687, 0.5737385320026608],
'train_mae': [123.9403071367058, 0.28572878310111227, 0.04116893787550451, 0.029583194951591386, 0.09974416590508049, 0.025241303895483343, 0.02430172262110701, 0.02370084193640701],
'train_mgd': [nan, nan, nan, 0.1546795521652909, 0.23424643366711273, 0.04458905319084727, 0.029784539532179258, 0.023554525799876765],
'train_mpd': [nan, nan, nan, 0.013951278777084245, 0.06556212083966138, 0.006906731079704761, 0.005120043338612812, 0.004313098412783701],
'train_mse': [23210.055900156614, 0.16608825903354982, 0.0032780028645832977, 0.0016512918633975041, 0.022582807492264476, 0.0011580858418903184, 0.000910727491805613, 0.0008117789656264954],
'train_r2': [-1168.29019174822, -1.3268246514805817, 0.7809907142215573, 0.6354439826487385, 0.4098240168982321, 0.6830953939783644, 0.5618122157684007, 0.5514403929341276],
'train_rmse': [152.3484686505139, 0.4075392729953149, 0.05725384584971823, 0.04063609065101494, 0.1502757714745277, 0.034030660321103356, 0.030178261908294403, 0.02849173504064811]

7.2.10- Training session 10 : Adding lags features and historical volatility / 400 iterations

- File : *training_with_many_dimensions/price_pred_lags_features_and_vol.py*

- Logs :

Metrics History:
'epoch': [50, 100, 150, 200, 250, 300, 350, 400],
'test_explained_variance': [-365976.68797027017, -5.683067196387611, -5.480833906776719, -5.198471464948341, -4.9663893750184185, -3.1285095206267854, -1.5814083837831427, -0.16760763190471484],
'test_mae': [24461.757905704348, 1.220909203267651, 1.1914817561514124, 1.140834939398388, 1.0657751082757425, 0.7656190637792255, 0.4310595049177627, 0.1410276653328248],
'test_mgd': [nan, nan, nan, nan, nan, nan, nan, 1.3602996546120383],
'test_mpd': [nan, nan, nan, nan, nan, nan, nan, 0.2014226750251934],
'test_mse': [2418119480.1994457, 1.8786789481467725, 1.7929875030695217, 1.6494145967891551, 1.4451423834282369, 0.7707070360473222, 0.2647274930128477, 0.03390106398094026],
'test_r2': [-480399.34739244194, -15.990844445577576, -15.708592881235997, -15.053967205808679, -13.601222909276782, -10.239107618554302, -5.351196606234092, -1.4605866882249288],

'test_rmse': [49174.378289912784, 1.3706490973793302, 1.3390248328800782, 1.2842953697608488, 1.202140750256906, 0.8778992174773379, 0.5145167567852846, 0.18412241574816537],
'train_explained_variance': [-263143.81903023267, -2.9136960644767607, -2.78561922295238, -2.6073266170720917, -2.461277688870501, -1.3250773670975584, -0.4157515400905 1873, 0.294734591801582],
'train_mae': [14824.18483896877, 1.047729131133622, 1.021792025368199, 0.9744937379104713, 0.8961764829001225, 0.6278600116522618, 0.3159990828221242, 0.07111784862678791],
'train_mgd': [nan, nan, nan, nan, nan, nan, nan, 0.5027492954382321],
'train_mpd': [nan, nan, nan, nan, nan, nan, nan, 0.067336911513438],
'train_mse': [1097997544.725144, 1.267507006002324, 1.2069744900590949, 1.100072435622421, 0.9335954082093946, 0.4678160407502704, 0.1282985909353805, 0.00981112275268018],
'train_r2': [-324989.98669220397,-16.078792179078672, -15.75734056926851, -14.952112765190352, -13.053409142814855, -9.163937957504407, -3.5858810644603976, -0.06093418004310047],
'train_rmse': [33136.04600318427, 1.125836136390338, 1.0986239074674713, 1.0488433799297305, 0.9662274101935809, 0.6839707893984, 0.3581879268420148, 0.09905111181950549]

7.2.11- Training session 11 : Adding lags features / 400 iterations

- File : *training_with_many_dimensions/price_pred_lags_features.py*

- Logs :

Metrics History:
'epoch': [50, 100, 150, 200, 250, 300, 350, 400],
'test_explained_variance': [-5476839.894039199, -3403737.216905766, -955601.790962938, -265206.13473217195, -262236.1455763003, -105958.68235064059, -25310.54653109254, -3978.6145010276755],
'test_mae': [2190866.459664147, 1004218.5531067023, 396917.06672506436, 57574.29097735384, 58287.48576625268, 22007.319860467138, 7018.493877964105, 1439.719830287559],
'test_mgd': [nan, nan, nan, nan, nan, nan, nan, nan],
'test_mpd': [nan, nan, nan, nan, nan, nan, nan, nan],
'test_mse': [8350252576913.775, 1797542308752.917, 263767473453.45386, 7965289503.316261, 6052991642.063855, 956117159.4419359, 88670942.09030575, 3755883.1668571914],
'test_r2': [-6479919.013994679, -6427788.332219915, -2277649.300509074, -380018.8987543727, -329541.09639466257, -109393.11693276315, -32853.41801938239, -6060.811447933852],
'test_rmse': [2889680.3589521414, 1340724.546188708, 513582.97621071304, 89248.47059370967, 77800.97455728851, 30921.144213012816,9416.524947681377, 1938.0101049419716],
'train_explained_variance': [-14486829.172929153, -8385987.281740204, -3030223.051641879, -739007.2077606883, -344375.380832632, -153098.39337233288, -46012.88357164575, -9004.172417277123],
'train_mae': [2416764.2763315868, 889404.7636920445, 347516.2749178202, 76864.96498521733, 46646.680296341256, 21742.708273432, 6718.84756441723, 1472.0842342198289],
'train_mgd': [nan, nan, nan, nan, nan, nan, nan, nan],
'train_mpd': [nan, nan, nan, nan, nan, nan, nan, nan],
'train_mse': [12530964699538.373, 1578846371487.798, 235860925827.6776, 10411072691.9198, 4249830858.266051, 953154328.0870063, 83360045.30083415, 3816350.249325056],
'train_r2': [-14487681.837577207, -8411369.772882646, -3034352.646066228, -740021.0186912781, -344711.3648976544, -162475.46776000506, -46015.627667402856, -9175.624587933162],

'train_rmse': [3539910.2671590946, 1256521.5364202072, 485655.1511388277, 102034.66416821197, 65190.72678123823, 30873.197568230706, 9130.172249242298, 1953.5481179958317]

7.2.12- Training session 12 : Using 3 LSTM layers (batch normalization, optimized rate, dropout, tanh activation function) / 800 iterations

- File : *training_with_many_dimensions/price_pred_3_lstm_layers_improved.py*

- Logs :

Metrics History:
'epoch': [50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600, 650, 700, 750, 800],
'test_explained_variance': [-0.08680863832612218, -0.1625368387475845, -0.1617648947217596, -0.18097988274145393, -0.20358409772352482, -0.22178882204986983, -0.23602058358715983, -0.2525578384262077, -0.25636516434905054, -0.2576979164554072, -0.22281398025459098, -0.24100508368646945, -0.2051642926189916, -0.21301482007695616, -0.21495789797812326, -0.21265780941122459],
'test_mae': [0.1304155349560433, 0.1333686712902313, 0.1329872198228559, 0.13394311754232027, 0.1362409020717589, 0.1389997498539989, 0.14057653284431917, 0.14319733159903952, 0.1430856272469182, 0.14334008697470813, 0.14181390329988078, 0.14209491723690718, 0.1415511615340772, 0.14317066601360212, 0.1430020217903016, 0.1426333438874195],
'test_mgd': [nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan],
'test_mpd': [nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan],
'test_mse': [0.03237917556115998, 0.033232347809418944, 0.03303517297607329, 0.03340290343526441, 0.034353915351006015, 0.03566428858240744, 0.03638614259194145, 0.037701579802490254, 0.03751287941893692, 0.03770758048624617, 0.03747089812293029, 0.0373196640986528, 0.037196694286022985, 0.03809767348565679, 0.038028764353766925, 0.03783269287773197],
'test_r2': [-0.5353742992778368, -0.6046458087790227, -0.607208520787194, -0.6097385893963567, -0.6262851388098984, -0.626101333708496, -0.6373608885699913, -0.6430611456189177, -0.6510526374218246, -0.6466952608978602, -0.6381149358221416, -0.6499524830551373, -0.6389884637516099, -0.6370916876379016, -0.6325424289995689, -0.6334894983788282],
'test_rmse': [0.17994214503878736, 0.18229741580565245, 0.1817558058937136, 0.18276461209781397, 0.1853480923856677, 0.18884991019962769, 0.19075152054948724, 0.1941689465452451, 0.1936824189722364, 0.1941843981535236, 0.1935740120029811 6, 0.1931829808721586, 0.19286444536519162, 0.19518625332142833, 0.1950096519502738, 0.19450627978996454],
'train_explained_variance': [0.44518180852657485, 0.4511154881738618, 0.45218826173770443, 0.45336623591336545, 0.45551912446756815, 0.4571561143649212, 0.4581727858202421, 0.45729149845731343, 0.4600316750499648, 0.45641317313059915, 0.45860601459512085, 0.46034006303499264, 0.4621757768308131, 0.4626032447816312, 0.4634334126988644, 0.4649691584607931],
'train_mae': [0.07524054775117586, 0.074161334915077, 0.07370394150110117, 0.07383830166203535, 0.07426254528660813, 0.07549340268584925, 0.07583362341180132, 0.07719942686013294, 0.07626099358953607, 0.07697218427175904, 0.0768440682815004, 0.07604930615437462, 0.07595190306408724, 0.07707899049758549, 0.07701202946340042, 0.07653219371753031],
'train_mgd': [nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan],
'train_mpd': [nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan],
'train_mse': [0.007855798592055282, 0.007633914179857899, 0.007560892100402806, 0.007621781201283442, 0.007724282096505071, 0.007997324700690001, 0.00808851405651289,

0.008366456559529377, 0.00823932783151109, 0.008359897470080513, 0.008320975078063193, 0.008197638208466265, 0.008196465706452667, 0.00839808145140636, 0.008394975868733136, 0.00832155352167929],
'train_r2': [0.4450124881294628, 0.4508261803592155, 0.4519589726184644, 0.4527673452626344, 0.45521772532553906, 0.4567457153138601, 0.4577225688671237, 0.45677452564385534, 0.4597227361736156, 0.45608640963150626, 0.45803794717782775, 0.4600334600489743, 0.46192530531888665, 0.46235077908991984, 0.46307193455606943, 0.4646991992453453],
'train_rmse': [0.08863294304069612, 0.08737227351887954, 0.08695339039050062, 0.08730281324953648, 0.08788789505105393, 0.0894277624716732, 0.08993616656558634, 0.09146833637674502, 0.09077074325745653, 0.0914324749204598, 0.0912193788515532, 0.09054080963005724, 0.0905343344066364, 0.09164104676075215, 0.09162410091636991, 0.0912225494144912]

7.2.13- Training session 13 : Using 3 bidirectional LSTM layers (batch normalization, optimized rate, dropout, relu activation function) / 400 iterations

- File : *training_with_many_dimensions/price_pred_3_bidir_lstm_layers.py*

- Logs :

Metrics History:
'epoch': [50, 100, 150, 200, 250, 300, 350, 400],
'test_explained_variance': [-0.582484691272249, -0.2828269345454715, -0.19660946482910457, -0.3625586732818751, -0.4983843600834066, -0.6178515149629096, -0.7758123892985429, -0.6233514335213415],
'test_mae': [0.2356096394563376, 0.18507138063636952, 0.15759910478881806, 0.17526777499627902, 0.1799977551798252, 0.1618407310165886, 0.174847543193553, 0.1613663929322442],
'test_mgd': [nan, nan, nan, nan, nan, nan, 3.305687859766376, 1.8584663022612105],
'test_mpd': [nan, nan, nan, nan, nan, nan, 0.3282808510868451, 0.22929629282975356],
'test_mse': [0.0927203608467137, 0.058230986616862025, 0.04021032254421899, 0.04868512353704935, 0.048806824430923455, 0.04118926178735467, 0.04666246964604121, 0.03963455122458102],
'test_r2': [-0.7891100549784869, -1.2263472787408758, -0.8476196189799745, -1.1624152902471718, -1.4239331739343486, -1.1105868689130687, -1.5449471230589458, -1.101890982803004],
'test_rmse': [0.3045001820142538 3, 0.24131097492004383, 0.2005251169909121, 0.22064705648852276, 0.2209226661773831, 0.20295137788976617, 0.21601497551336854, 0.19908428171149278],
'train_explained_variance': [0.31451773540302486, 0.6112894177920944, 0.7801575600741351, 0.7920424803284652, 0.8284006227809051, 0.8683406008868737, 0.905456391566441, 0.940972694196498],
'train_mae': [0.16713287103773228, 0.06391940894576685, 0.04218943579456197, 0.037408548231781864, 0.03263202927839997, 0.03021553727329, 0.02324820830728074, 0.016887457415040354],
'train_mgd': [nan, nan, nan, nan, nan, nan, nan, nan],
'train_mpd': [nan, nan, nan, nan, nan, nan, nan, nan],
'train_mse': [0.041718092454068385, 0.007083070326292678, 0.0032169682460300923, 0.003146764095787179, 0.0023195737514149955, 0.0018355882680363017, 0.0011635901258310979, 0.0007479970633124587],
'train_r2': [-0.1993081251233355, 0.5965360790247716, 0.7797752260632924, 0.7917661140122455, 0.84006022546792543, 0.8598674944885913, 0.9054513220788882, 0.9409009465540403],

'train_rmse': [0.20425007332695963, 0.08416097864386249, 0.056718323723732283, 0.056096025668376716, 0.04816195335962813, 0.042843765801296015, 0.03411143687725713, 0.02734953497433656]

7.2.14- Training session 14 : Using 3 bidirectional LSTM layers (batch normalization, optimized rate, dropout, relu activation function) / 650 iterations

- File : *training_with_many_dimensions/price_pred_3_bidir_lstm_layers.py*

- Logs :

Metrics History:
'epoch': [50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600, 650],
'test_explained_variance': [-0.7071370671474921, -0.5627074344420819, -0.45620970111223325, -0.6248275504582037, -0.5761188740041143, -0.44597315168760265, -0.6290969110015401, -0.30526450891760337, -0.3086585504818662, -0.21637046773145663, -0.41977055357877435, -0.40571491645819924, -0.360678385724865],
'test_mae': [0.2519656175799133, 0.20082640641081023, 0.20350027972384935, 0.17159885323209714, 0.14716321946852273, 0.16152775859036572, 0.14402316092549608, 0.14090008397591852, 0.13368632804242359, 0.11898744825419318, 0.13559229084237054, 0.1383051647211057, 0.11715137342148577],
'test_mgd': [nan, nan, nan, nan, nan, 2.0995273270415447, 1.8751395810777063, 2.114630910363163, 2.9838194612301607, 1.7712866557002251, 1.047549113757962, 1.7721723386289059, 0.8144156616662142],
'test_mpd': [nan, nan, nan, nan, nan, 0.2510634405322648, 0.2100675329010204, 0.20634794376667465, 0.2095938552548899, 0.1566618503614067, 0.14268525930012793, 0.1857579233379893, 0.11570308546707343],
'test_mse': [0.09841415031066543, 0.06526296236634352, 0.06403391793550181, 0.04925055795545763, 0.03796112725696308, 0.04276913254575261, 0.03288595093520466, 0.031948130877124205, 0.028673453720539498, 0.022590168170012647, 0.027902980306736273, 0.029738670618420348, 0.021952586541505267],
'test_r2': [-1.3217206861165218, -1.0446484613015952, -1.1311199762744586, -1.2636192760689822, -0.9290687376281237, -1.0462575267137186, -0.881842088790538, -0.6782341901323823, -0.5650351531415057, -0.45127105914374943, -0.533970081847388, -0.7591598064035494, -0.4804446768358379],
'test_rmse': [0.3137102967877615, 0.2554661667742786, 0.25304924014013913, 0.22192466729829208, 0.19483615490191517, 0.20680699346432319, 0.1813448398361659, 0.17874040079714548, 0.16933237646870578, 0.15030026004639063, 0.1670418519615257, 0.1724490377428078, 0.1481640527979215],
'train_explained_variance': [0.11821261165900698, 0.37473741860577714, 0.6918012369130251, 0.7808323494024236, 0.8517520820151904, 0.8622674564333296, 0.8333324241532202, 0.8959175322118514, 0.9317388983940964, 0.9375878171342901, 0.8177610824267387, 0.9469547068789834, 0.9467896688285494],
'train_mae': [0.1346046129294348, 0.09173218481878619, 0.060319851976724376, 0.04196409809208181, 0.0333919301790478, 0.02648472700933887, 0.027180376865015062, 0.02299512091338802, 0.019249761987358702, 0.01624300885085381, 0.029421241922678948, 0.016595918048231652, 0.015372747874334729],
'train_mgd': [nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan],
'train_mpd': [nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan],
'train_mse': [0.026575438629527756, 0.013397975191162663, 0.0062634184553267335, 0.003252010960754027, 0.002112532925443777, 0.0019140728500595943, 0.002004303302257819, 0.0013599159915964787, 0.0008619712397031819, 0.0006534401059042674, 0.00240758116305504, 0.0006106005384306848],
'train_r2': [0.06593466393147973, 0.3746323284167433, 0.6894341899218566, 0.7773162020873958, 0.8400602254679254, 0.8616627869473956, 0.8291241824311472,

0.8935701179447441, 0.9299059335594757, 0.9374569224660906, 0.8028069411283162, 0.9461872339212086, 0.9467763638910418],
'train_rmse': [0.1630197492009105, 0.1157496228553798, 0.07914176176537097, 0.05702640582005872, 0.045962298957338686, 0.04405370019187894, 0.044769446079416915, 0.036877038812741986, 0.029359346717922417, 0.02556247456535199, 0.049067108770081816, 0.02471033262484916, 0.023015840168728298]

7.2.15- Training session 15 : Using 3 LSTM layers, dataset without sma crossings / 400 iterations

- File : *training_with_many_dimensions/price_pred_3_lstm_layers.py*

- Logs :

Metrics History:
'epoch': [50, 100, 150, 200, 250, 300, 350, 400],
'test_explained_variance': [-24674.74928703454, -60.28128380779874, -24.02599588933179, 0.0, 2.220446049250313e-16, 0.0, 0.0, -2.220446049250313e-16],
'test_mae': [20558.672239839754, 151.71461932101258, 43.60831742573993, 0.13840782462409965, 0.11144008133982675, 0.06026047597248345, 0.0015390038124062384, 0.0014165950014461872],
'test_mgd': [nan, nan, nan, nan, nan, nan, 8.769024541450676e-05, 6.510839416734082e-05],
'test_mpd': [nan, nan, nan, nan, nan, nan, 1.9581769535992123e-05, 1.5774600874995727e-05],
'test_mse': [1037509885.0425725, 137722.77424949306, 20417.149775995178, 0.028947990964626396, 0.01957374387907917, 0.006153078759979987, 4.372808399185949e-06, 3.8219579669205344e-06],
'test_r2': [-24676.99144463724, -63.03721087565968, -26.596364561765572, -1.9565118321623745, -1.7357300288779527, -1.3474364783852013, -0.9310084320335081, -0.7686958685715486],
'test_rmse': [32210.4002620671, 371.1101915193021, 142.88859218284423, 0.17014109134664204, 0.13990619671436705, 0.0784415627074065, 0.0020911261079107468, 0.0019549828559147354],
'train_explained_variance': [-39411.533350282145, -3719.056173492273, -524.797644724282, 0.004628155860740835, 0.005183176965305014, 0.005576440638439317, -0.00016439812783453078, -0.00015821155277429],
'train_mae': [9955.96372538886, 286.72207887795173, 74.58144421053696, 0.07485757544774056, 0.06271231219761537, 0.03669774710370791, 0.0010859840440239743, 0.0010750691351319648],
'train_mgd': [nan, nan, nan, nan, nan, nan, 3.1112393169880655e-05, 2.4904861786236584e-05],
'train_mpd': [nan, nan, nan, nan, nan, nan, 6.9208234246623545e-06, 6.010696369309635e-06],
'train_mse': [1112299363.5615761, 5384322.312194951, 261135.982988813, 0.009297296966681575, 0.006218219060733558, 0.001971535615312148, 1.539522107140365e-06, 1.4506697909830915e-06],
'train_r2': [-39416.007066171514, -3728.942058055046, -524.8572529885445, -0.41469326654731664, -0.2948210294124345, -0.1205994171101834, -0.012870778099693059, -0.00018471335392322707],
'train_rmse': [33351.15235732607, 2320.414254437115, 511.0146602484248, 0.09642249201654962, 0.0788556850248196, 0.04440197760586963, 0.0012407748011385325, 0.0012044375413374872]

7.2.16- Training session 16 : Adding lags features, dataset without sma crossings / 400 iterations

- File : *training_with_many_dimensions/price_pred_lags_features.py*

- Logs :

Metrics History:
{'epoch': [50, 100, 150, 200, 250, 300, 350, 400],
'test_explained_variance': [0.0, 1.1102230246251565e-16, 0.0, 0.0, 1.1102230246251565e-16, 0.0, 0.0, 1.1102230246251565e-16],
'test_mae': [0.2063804797217207, 0.20584155831493042, 0.20678853088467883, 0.20675456294267938, 0.2060359077675379, 0.20747472294167313, 0.2069758049532728, 0.20585229408002995],
'test_mgd': [0.22003580610872767, 0.2180723732478042, 0.22153321115106941, 0.22140806424797854, 0.21878089986588553, 0.22407444112851965, 0.2222248276234225, 0.21811150104515334],
'test_mpd': [0.1315525450301817, 0.13072555406436281, 0.13218139407656856, 0.1321288985180767, 0.13102430236748966, 0.13324497622667533, 0.13247130451684797, 0.13074205581919413],
'test_mse': [0.08126394894707704, 0.08095310592119681, 0.0814995515101277, 0.08147990885938317, 0.08106552932057737, 0.08189653702400639, 0.08160794460099197, 0.08095931478969122],
'test_r2': [-0.7601371863771367, -0.7534044792411088, -0.7652402220750605, -0.7648147719158889, -0.75583951480648, -0.7738387331551213, -0.7675879631391926, -0.7535389602805358],
'test_rmse': [0.2850683232964986, 0.28452259298902227, 0.28548126297557197, 0.2854468582054866, 0.28472008942218563, 0.2861757100524193, 0.28567104263644216, 0.2845335038087628],
'train_explained_variance': [6.122717111090026e-09, 6.122717111090026e-09, 5.740470765402961e-09, 6.122717111090026e-09, 6.122717111090026e-09, 3.445438823135305e-09, 1.2627460410641334e-08, 1.4926378577584387e-08],
'train_mae': [0.15725294222783648, 0.15734869731188925, 0.15718064716406785, 0.15718666529492809, 0.15731401612463644, 0.15705937469949105, 0.15714746706818902, 0.15734677992626883],
'train_mgd': [0.1365335288950041, 0.1360337635082179, 0.13691672951677916, 0.13688463567055212, 0.13621375375628222, 0.13757105720831397, 0.13709432583136127, 0.13604371629934878],
'train_mpd': [0.0638309683569555, 0.06371954703714235, 0.0639172704075373, 0.06391001435936167, 0.06375952389965028, 0.06406628471993589, 0.06395751344726787, 0.06372175289647111],
'train_mse': [0.030988978668615698, 0.030989892937122784, 0.030989204486224938, 0.03098915560200017, 0.030989402090632966, 0.03099135336830621 8, 0.0309895717857296, 0.0309898607987871],
'train_r2': [-5.736403190237382e-07, -3.0076678845247784e-05, -7.860674241522148e-06, -6.283202026891033e-06, -1.4237280695494903e-05, -7.720414428447064e-05, -1.9713266070642277e-05, -2.9039589031443214e-05],
'train_rmse': [0.17603686735628904, 0.17603946414688607, 0.17603750874806465, 0.17603736990196192, 0.1760380700037153, 0.17604361212013975, 0.17603855198714172, 0.1760393728652403]}

7.2.17- Training session 17 : Adding lags features, dataset without sma and RSI / 400 iterations

- File : *training_with_many_dimensions/price_pred_lags_features.py*

- Logs :

Metrics History:

{'epoch': [50, 100, 150, 200, 250, 300, 350, 400],
'test_explained_variance': [0.3827472743098863, 0.3784451555788073, 0.39577973204778505, 0.3673315593730532, 0.39709078425683186, 0.3818499674314828, 0.12053955243991643, -19023.71955958647],
'test_mae': [0.03897687619510479, 0.03820018085529268, 0.042307956507653836, 0.036544790158102025, 0.042954352029546936, 0.03949207818623377, 0.012259953893627072, 2421.1534248853577],
'test_mgd': [0.02229521328183342, 0.021849606880179926, 0.024448543229638072, 0.021106906141762888, 0.025716727079036585, 0.025067774606512057, 0.00515784002562314, nan],
'test_mpd': [0.0078856762651881, 0.007656252115307299, 0.008938502221936307, 0.007212481350628622, 0.009344819544477823, 0.008542549144538017, 0.0011907978398806073, nan],
'test_mse': [0.002823453110906789 5, 0.0027147065790037255, 0.003312630953688903, 0.00249149687210493, 0.0034437531691375515, 0.0029493538667170636, 0.0002752250397192932, 11200750.62703752],
'test_r2': [0.23597666813467433, 0.21879489122215623, 0.29342743960467566, 0.17285626900175, 0.2903587777063492, 0.19128308782402226, -0.7203236720561155, -39912.925950323566],
'test_rmse': [0.053136175162565, 0.05210284616989484, 0.05755545980781409, 0.049914896294642644, 0.05868349997348106, 0.05430795399126231, 0.016589907767052027, 3346.752250621118],
'train_explained_variance': [0.470995095515787, 0.4616091907905618 4, 0.5068572769661435, 0.4398028700544707, 0.5119427365171123, 0.4689788397713779, 0.12537678108194472, -12792.888581486797],
'train_mae': [0.03638586839256589, 0.035327075338869945, 0.040821409336065914, 0.03292731566639345, 0.04111744246857315, 0.035275806653926135, 0.008582294043068307, 1555.012473104624],
'train_mgd': [0.018719328941632637, 0.017818978439185548, 0.022764216768267388, 0.015941996145706776, 0.023427682221580027, 0.018722601472641064, 0.0019077228099584557, nan],
'train_mpd': [0.0055452944585015, 0.005245882372846804, 0.006896929026212942, 0.004612299768957544, 0.007032716474506922, 0.005332546338576455, 0.0004256109236412 4903, nan],
'train_mse': [0.0016556581810847875, 0.001556153298267318, 0.0021090309543121324, 0.0013439942208139598, 0.002132233474934012, 0.0015326688796770034, 9.50184737353485e-05, 4827331.7436873475],
'train_r2': [0.33251561350118086, 0.332826255196514, 0.3297897641688218, 0.3352437520871824, 0.3453849782748829, 0.3738730861645675, 0.1151397972631284, -25627.834394675192],
'train_rmse': [0.04068977981121043, 0.039448108931447116, 0.04592418702940894, 0.03666052673945043, 0.046176113683743594, 0.03914931518784209, 0.009747741981369249, 2197.118964391175]}

7.2.18- Training session 18 : Adding lags features and historical volatility, dataset without sma crossings / 400 iterations

- File : *training_with_many_dimensions/price_pred_lags_features_and_vol.py*

- Logs :

Metrics History:
{'epoch': [50, 100, 150, 200, 250, 300, 350, 400],
'test_explained_variance': [0.0, 0.0, 0.0, 1.1102230246251565e-16, -2.220446049250313e-16, 0.0, 1.1102230246251565e-16, 0.0],
'test_mae': [1.8100934751179363, 0.9082247293246359, 0.5250168770372643, 0.3195492460434813, 0.16731257481318104, 0.08851308984871933, 0.05896680489280832, 0.05294382722352449],
'test_mgd': [nan, nan, nan, nan, nan, nan, 2.464282007882169, 0.9788670064913256],
'test_mpd': [nan, nan, nan, nan, nan, nan, 0.11388075685014465, 0.06865537909408433],
'test_mse': [3.9640069500177564, 1.0124974885646507, 0.3522533010226303, 0.1461453636159228, 0.049404458915103346, 0.0149676669573708, 0.006666444555045 6055, 0.005372618090769883],
'test_r2': [-4.765253347558597, -4.396379532981401, -3.597971999669614, -2.3189478263001044, -1.1465463777167049, -0.7537217345479323, -0.7343749895593465, -0.7360634295244057],
'test_rmse': [1.9909814037347904, 1.0062293419318733, 0.5935093099713182, 0.3822896331525651, 0.22227113828633566, 0.12234241683639734, 0.08164829793109961, 0.07329814520688695],
'train_explained_variance': [-1.2341931756166682, -0.47067684362735696, -0.14109914491097042, -0.07390589082986865, -0.022665212420536207, 0.01740520063951878, 0.022013266578557955, 0.023048762599185535],
'train_mae': [1.1913855562836568, 0.5658370612868325, 0.3006700527286024, 0.16984191977014024, 0.10937891944802314, 0.06688553038703712, 0.04479175021386559, 0.04008832924194068],
'train_mgd': [nan, nan, nan, nan, nan, nan, nan, nan],
'train_mpd': [nan, nan, nan, nan, nan, nan, nan, nan],
'train_mse': [2.2768640456418785, 0.4921345906579505, 0.1424235135422394, 0.05080059052878264, 0.016781981919326804, 0.005629741884297184, 0.002523137976839482, 0.002029312851703281 4],
'train_r2': [-3.9336175337491497, -2.9078403562014756, -1.769729370906954, -0.7188126223858289, -0.08632998760823574, 0.017258158529711687, 0.02201065890959064, 0.02304847596505777],
'train_rmse': [1.5089281114890392, 0.7015230506960912, 0.37739039937740787, 0.225389863411784, 0.1295452890665145, 0.07503160590242744, 0.050230846865641054, 0.04504789508626659]}

7.2.19- Training session 19 : Adding lags features and historical volatity, dataset without sma and RSI / 400 iterations

- File : *training_with_many_dimensions/price_pred_lags_features_and_vol.py*

- Logs :

Metrics History:
'epoch': [50, 100, 150, 200, 250, 300, 350, 400],
'test_explained_variance': [0.0, -4.440892098500626e-16, -2.220446049250313e-16, -2.220446049250313e-16, 0.0, 0.0, 1.1102230246251565e-16, 0.0],
'test_mae': [0.19506911719217832, 0.3358176817823718, 0.26890059821500784, 0.2176151971866721, 0.20650212311595248, 0.20677595877497956, 0.20552497720079807, 0.20708675548926625],
'test_mgd': [nan, 1.7372763161838152, 0.590757134175802, 0.26529051971251905, 0.22048081376414377, 0.221486858871306, 0.21691534577037772, 0.22263502756655268],
'test_mpd': [nan, 0.49335852564789034, 0.2561176412893655, 0.14990287617167414, 0.1317395997475752, 0.13216195485448162, 0.13023691983074337, 0.13264309930165014],
'test_mse': [0.050621156889935355, 0.1589426149292639 8, 0.11819890925669607, 0.08788083570109341, 0.08133410154043971, 0.08149228079628429, 0.08076890468539222, 0.08167211674687047],
'test_r2': [-3.027398033061484, -2.442618906191884, -1.5601302701572775, -0.9034557007330584, -0.7616566570636873, -0.7650827419873576, -0.7494147710234165, -0.7689779002731107],
'test_rmse': [0.22499145959332625, 0.3986760776987553, 0.34380068245525064, 0.2964470200577051, 0.2851913419801515, 0.2854685285566244, 0.28419870634011024, 0.28578333881958634],
'train_explained_variance': [-0.012205682776740945, 0.0, 1.1102230246251565e-16, -2.220446049250313e-16, -2.220446049250313e-16, -2.220446049250313e-16, -2.220446049250313e-16, -2.220446049250313e-16],
'train_mae': [0.10241568288280894, 0.17154373783465998, 0.1570636167106748, 0.15534573541263488, 0.15723139086482063, 0.15718287495556724, 0.15740564280926458, 0.1571278483870146],
'train_mgd': [nan, 0.7446449973629566, 0.25966608823153137, 0.14880730075318924, 0.13664721465338875, 0.13690482951640282, 0.13574069842484934, 0.13719980484408276],
'train_mpd': [nan, 0.19065102560596314, 0.0963328516555338, 0.06684884433192084, 0.06385649522083815, 0.06391457959583854, 0.0636548311061199, 0.06398148762977324],
'train_mse': [0.018082325302486147, 0.05299592668805519, 0.037535949951415494, 0.031270001991915396, 0.03098896371224276, 0.03098918593859821 2, 0.030991092502547998, 0.03098986759845157 7],
'train_r2': [-1.1433404277423418, -0.7101550088298814, -0.21126842820297953, -0.009069071428231545, -9.100480924395526e-08, -7.2621505675662945e-06, -6.878612216576485e-05, -2.925901118300267e-05],
'train_rmse': [0.13447053693090597, 0.23020844182621797, 0.1937419674500481, 0.17683326042324 56, 0.17603682487548666, 0.17603745606716262, 0.1760428712062718, 0.17603939217814737]

7.2.20- Training session 20 : Using 3 bidirectional lstm layers (batch normalization, optimized rate, dropout, relu activation function), dataset without sma crossings / 650 iterations

- File : *training_with_many_dimensions/price_pred_3_bidir_lstm_layers.py*

- Logs :

Metrics History:

'epoch': [50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600, 650],

'test_explained_variance': [-1.3007897773037898, -1.0372352321186167, -0.4676643458615768, -0.5901960899894438, -0.4683441966999029, -0.7562104347060326, -0.6577417876727807, -0.2172871508864309, -0.5494477383216401, -0.4837811604994031, -0.39030785259915146, -0.6585957967931577, -0.3752422623669667],

'test_mae': [0.41563181179887926, 0.28094482052879804, 0.20547890908545294, 0.21073909690027612, 0.18046739504077017, 0.1790255002955427, 0.17099843942767806, 0.14991129168252296, 0.1392451211155018, 0.14054083245755788, 0.1353199732264058, 0.16859693556743433, 0.12914559722943733],

'test_mgd': [nan, nan, nan, nan, nan, nan, nan, 4.747577993749377, 2.5252074677782974, 1.5382047306958788, 1.6007378300887882, 2.3969422862250593, 1.1460161620685099],

'test_mpd': [nan, nan, nan, nan, nan, nan, nan, 0.2579700757395152, 0.21705817099014432, 0.18583131745282255, 0.17801682829504942, 0.26204857311775276, 0.1451629199266746],

'test_mse': [0.25358612818470677, 0.1208150926001284, 0.06523534269715844, 0.06537045241011459, 0.04713341972725279, 0.04761333162960942, 0.044705241208548706, 0.03286707825618174, 0.03088787272094756, 0.031874213184965916, 0.029372618760400145, 0.04122057204306511, 0.026006799853866863],

'test_r2': [-2.5709019185968485, -1.8522544359977435, -1.2014465925184146, -1.6233338594521656, -1.5204984026770862, -1.4562771094397324, -1.1520210464507823, -0.9580631796450254, -0.9157006611292102, -0.8552353171285112, -0.807088538874885, -1.37240739573961, -0.6986221610243248],

'test_rmse': [0.5035733592881049, 0.3475846553001562, 0.25541210366221573, 0.2556764604145532, 0.21710232547638172, 0.21820479286580627, 0.2114361397882318, 0.1812927970333674, 0.17574946008721495, 0.17853350717712885, 0.17138441807935792, 0.20302850056843033, 0.16126623903925727],

'train_explained_variance': [0.20754916602167062, 0.5596667830704313, 0.5984746100261152, 0.8316817323240606, 0.8612660980956474, 0.9135722747079763, 0.929273407575668, 0.9053834709385691, 0.9371440678532033, 0.9162969954733337, 0.9580480255367788, 0.9675762586934742, 0.9692284562388399],

'train_mae': [0.183367782162448, 0.08919425626567809, 0.06649057720749532, 0.041447513079906406, 0.02905976294895155, 0.023964758414464493, 0.023534729218645117, 0.022298604970587006, 0.0171294606328231, 0.019129247961333994, 0.014791491652806212, 0.013394946838475646, 0.011793208040785874],

'train_mgd': [nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan],

'train_mpd': [nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan],

'train_mse': [0.04998104443273517, 0.012564376317104163, 0.008142753543843997, 0.002855154892813733, 0.001748633098218 7956, 0.0011367129386171083, 0.001033725120897524, 0.0010716489686655518, 0.0006845095968644 1, 0.0010582923317597511, 0.0004577045209798479, 0.0003871216793748652, 0.0003168641593568141],

'train_r2': [-0.0485813916112805, 0.5580711042757639, 0.5906065534923992, 0.8292950407275769, 0.8606841398711339, 0.9126336919731513, 0.925862569044324, 0.9048820366523199, 0.9367495555165584, 0.9082281324197174, 0.9580466779771343, 0.966805423039701, 0.9691661766824948],

'train_rmse': [0.22356440779501366, 0.11209092879044298, 0.09023720709244051, 0.0534336494431527, 0.0418166605340359, 0.033715173714769855, 0.032151595930801385, 0.03273604998568935, 0.026163134308878417, 0.03253140531486076, 0.02139403003129256, 0.019675407984966035, 0.01780067862068225]