Università degli Studi di Padova

Dipartimento Matematica

Business Economic and Financial Data Project Report

Time-series Analysis on Currency Exchange Rates

Student: Hieu Nguyen Minh **Student ID**: 2141573

January, 2025

1 Introduction

In this project, we are going to analyze the time-series data on currency exchange rates. We will download the data from **yahoo!finance**[1] for three pairs of currency: EUR-USD, BTC-USD, and USD-VND. We will implement different forecast methods, including basic ones: mean, naive, seasonal naive, drift; and complicated ones: ARIMA, GAM, RNN, LSTM, GRU. By comparing performance of these methods, we can see how they behave in forecasting future values and which method is optimal for financial data prediction. The source code of this project is provided here: `https://github.com/hieunm44/befd_time_series`.

About the dataset: We download the data in the last ten years, from Monday, January 05, 2015 to Friday, December 27, 2024, so there are a total of 520 weeks in the dataset. EUR-USD pair is only traded 5 days a week on the exchange, and BTC-USD is traded 24/7 every day. We focus on the `Close` rate of each currency pair. Table 1 gives the dataset summary.

Table 1: Statistics of Close rate in the dataset.

Pair	Count	Mean	Std	Min	Max	Range	Median
EUR-USD	2602	1.1191	0.0522	0.9596	1.251	0.2914	1.1147
BTC-USD	3645	20084.7	22078.6	178.1	22078.6	21900.5	9665.5

We already checked and confirmed that there is no null value. The BTC-USD data has 1043 rows more than the EUR-USD data, corresponding to 520 weekends of non-trading. Based on the range and standard deviation, it is clear that the EUR-USD data is more stable then the BTC-USD. This makes sense because BTC is a cryptocurrency that just went public in 2009 and its value has increased and decreased drastically many times in its history. In contrast, traditional currency like EUR-USD can hardly change much in price.

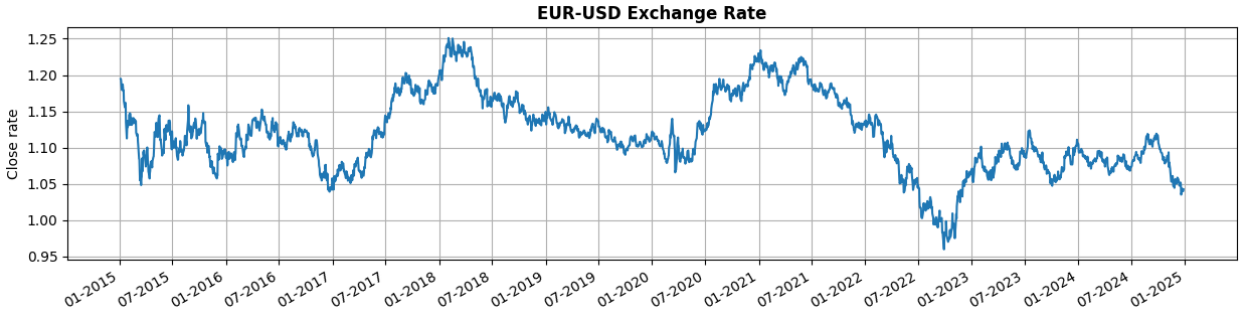
2 Exploratory Data Analysis

2.1 Data overview

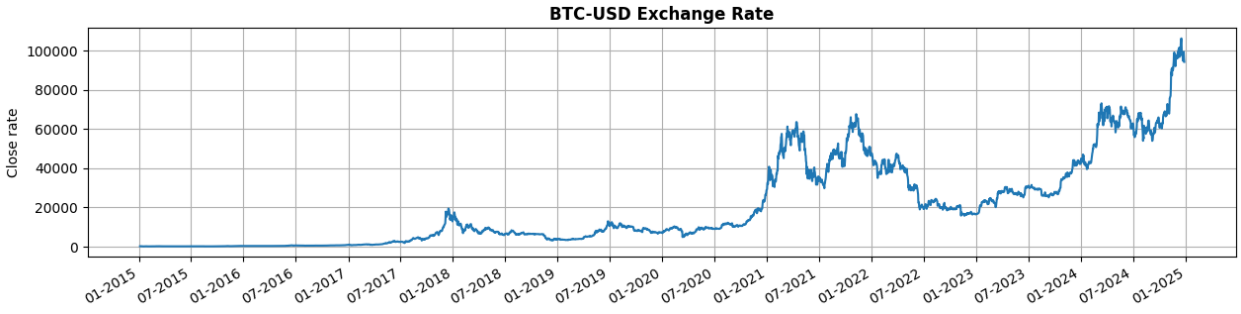
Figure 1 shows two time-series of EUR-USD and BTC-USD. For EUR-USD, the close rate did not change much from 1/2015 to 1/2017, but after that it rose constantly until 1/2020. This period is in President Donald Trump tenure, when he made a lot of changes to the US economy policy and

[1] https://finance.yahoo.com/

started the US-China trade war, affecting the USD itself. During the Covid-19 pandemic, the data witnessed an appreciation from 1/2020 to 7/2021. This was followed by a depreciation of EUR over USD when the war in Ukraine broke out. From 1/2023 to 12/2024, the data was quite stable.



(a) EUR-USD close rate time-series.



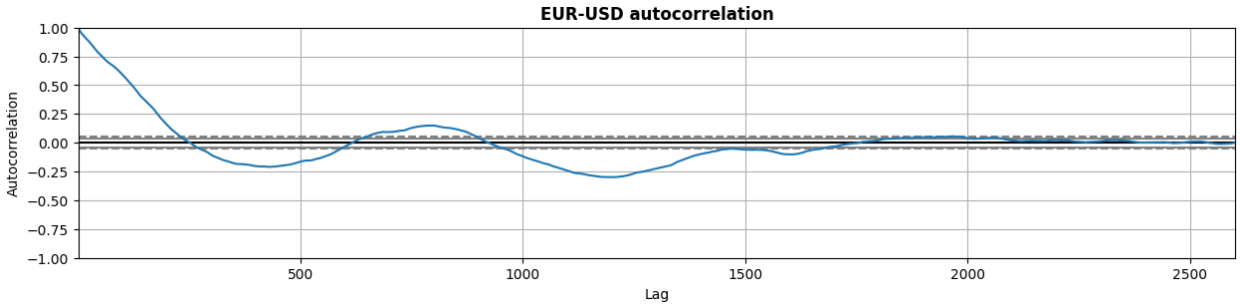
(b) BTC-USD close rate time-series.

Figure 1: EUR-USD and BTC-USD close rate time-series.

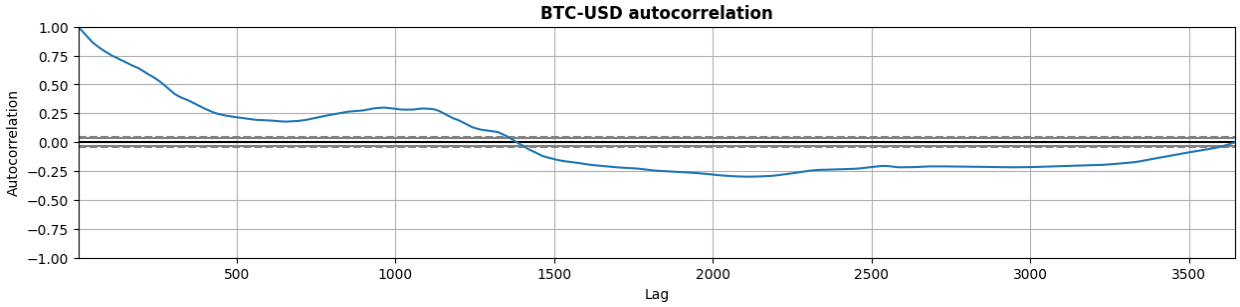
For BTC-USD, the BTC value had been increasing from 1/2015 to 1/2021 but still stayed below $20,000. Then, it went up dramatically to over $60,000 in just a few months. Then it dropped to around $30,000 in 7/2021 but it rose back to nearly $70,000 a few months later. Following that was a decrease back to $20,000 in 7/2022 and stayed stable to 1/2023. However, afterwards BTC has gained constantly and considerably in value, eaching over $100,000 in 12/2024. This behavior demonstrates the instability of cryptocurrencies.

2.2 Autocorrelation Check

Now we check the autocorrelation function (ACF) for our two time-series, which is showed in Figure2. An autocorrelation coefficient r_k measures the correlation between y_t and t_{t-k}, where k is the lag value. For EUR-USD data, the ACF gradually drops to zero after about 2000 lags, confirming the data stability. In contrast, the ACF of BTC-USD is quite large and decreases more slowly, showing the data instability. To make the BTC-USD stable, we use first differencing and show the result of time series and ACF in Figure3. Now the series looks like a white noise with ACF very close to zero, thus being a stable series.

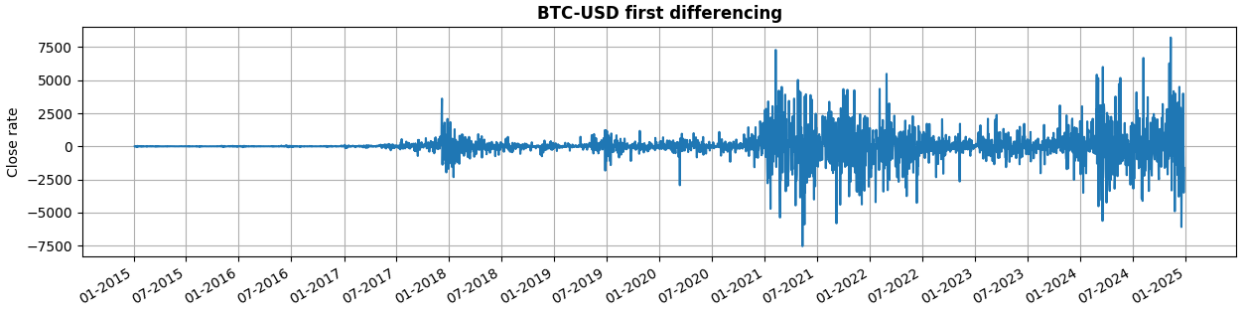


(a) EUR-USD close rate ACF.

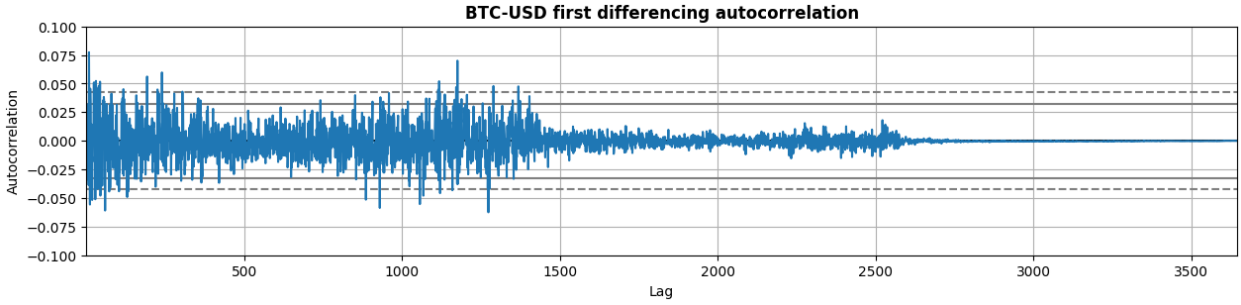


(b) BTC-USD close rate ACF.

Figure 2: EUR-USD and BTC-USD close rate ACF.



(a) EUR-USD first differencing time series.

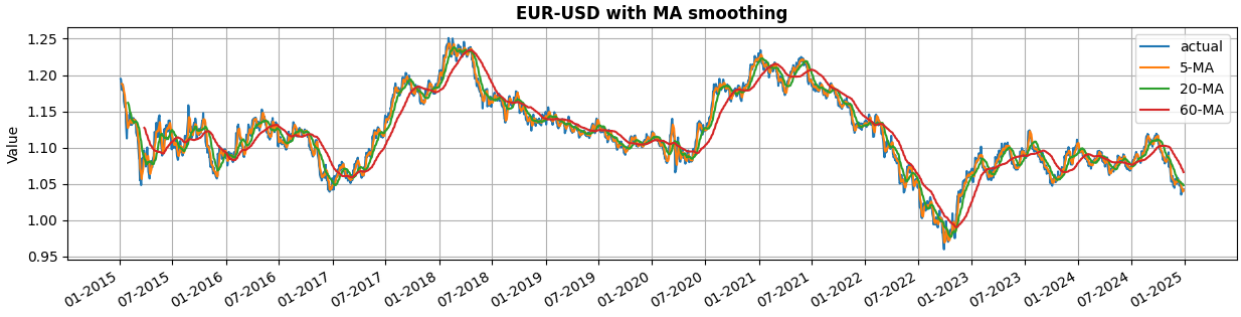


(b) BTC-USD first differencing ACF.

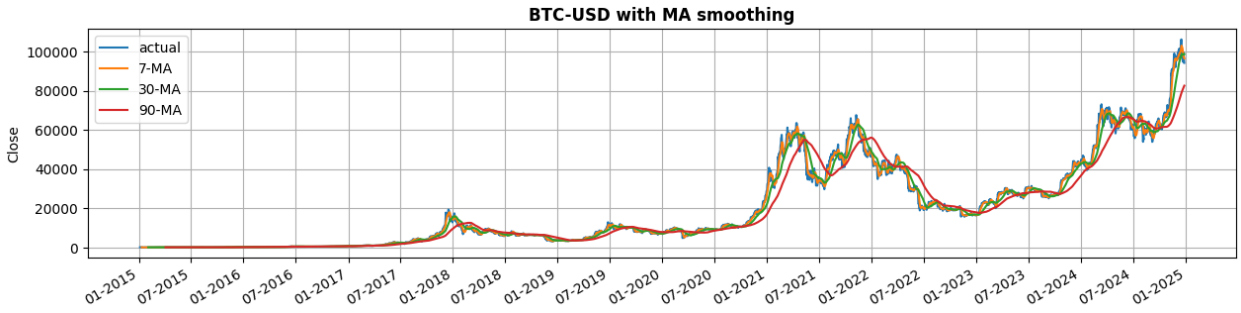
Figure 3: BTC-USD series after first differencing.

2.3 Time series decomposition

The first step in a time series decomposition is to use a moving average m-MA of order m to estimate the trend-cycle. For EUR-USD data, since it is only traded five days a week, we try with $m = 5$ (a week), $m = 20$ (four weeks, or a month), and $m = 60$ (three months). For BTC-USD data, we try with $m \in \{7, 30, 90\}$. The results are showed on Figure4. The higher m is the smoother the curve is. We see that $m = 60$ (for EUR-USD) and $m = 90$ (for BTC-USD) is sufficient for trend-cycle estimation. We then use these values of m for seasonal decomposition, whose results are showed in Figure 5.



(a) EUR-USD moving average smoothinng.

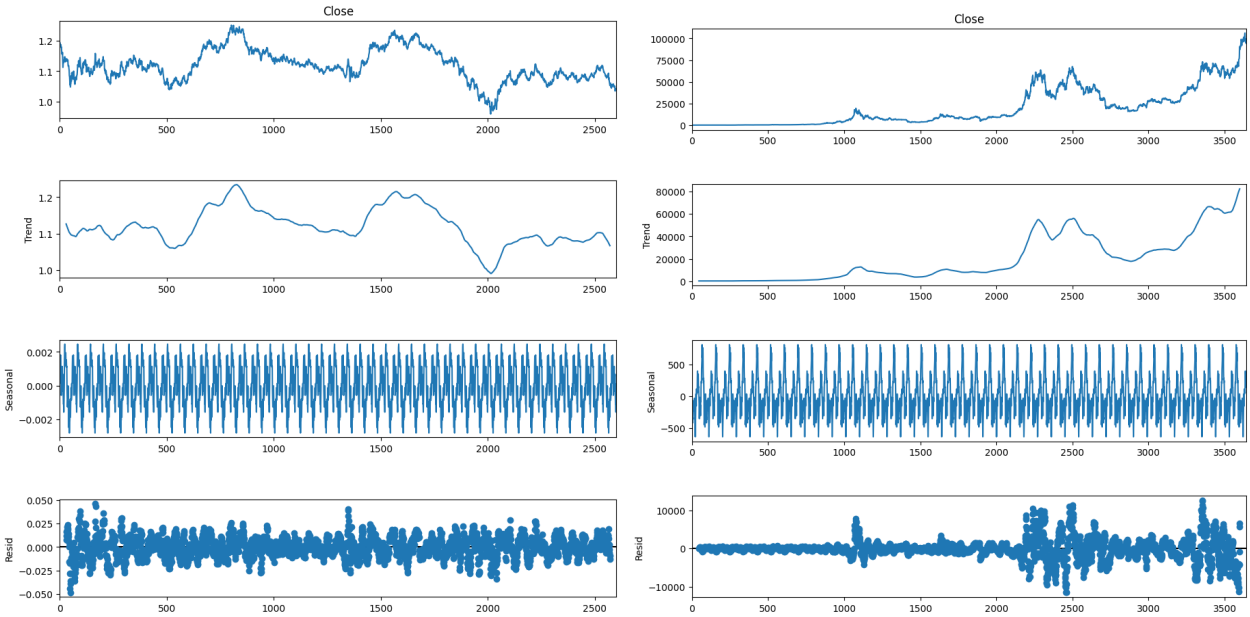


(b) BTC-USD moving average smoothinng.

Figure 4: EUR-USD and BTC-USD moving average smoothinng.

2.4 Train-test split

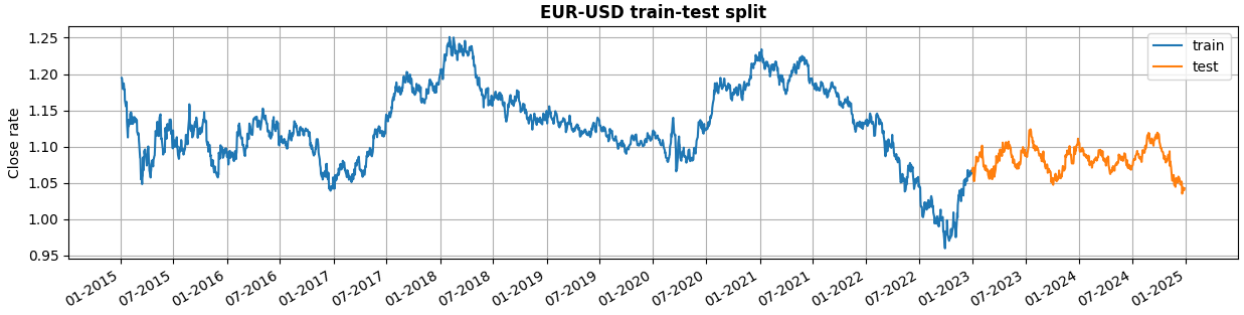
We split the series with a ratio of 80% for training and 20% for testing. That means the data from the beginning to Dec 29, 2022 (for both EUR-USD and BTC-USD) will be used in implementing different forecast methods, while the data from Dec 30, 2022 to the end will be used to check the predictions and method performance. Figure 6 illustrates the train and test time series.



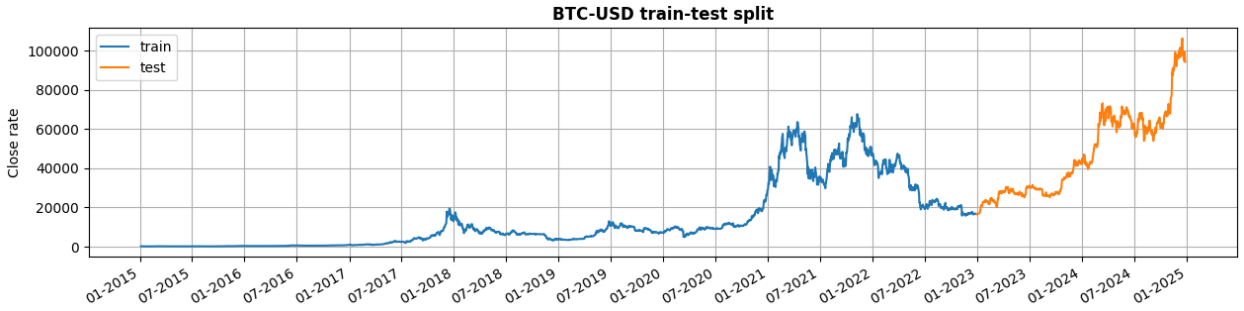
(a) EUR-USD series decomposition.

(b) BTC-USD series decomposition.

Figure 5: EUR-USD and BTC-USD series decomposition.



(a) EUR-USD train-test time series.



(b) BTC-USD train-test time series.

Figure 6: EUR-USD and BTC-USD series train-test split.

2.5 Box-Cox transformation

The BTC-USD has a significant volatility, so we apply a Box-Cox transformation[2] to the BTC-USD training data to stabilize it and to avoid model performance depreciation.

$$w_t = \begin{cases} \log(y_t) & \text{if } \lambda = 0; \\ \left(\text{sign}(y_t)\,|y_t|^{\lambda} - 1\right)/\lambda & \text{otherwise,} \end{cases} \tag{1}$$

where λ is a parameter that will be returned by the SciPy library[3] such that it maximizes the log-likelihood function. Figure 7 shows the transformed data, which will be used in model implementation. Then the model predicts future values, but these values need to be transformed back to original scale to match the actual data.

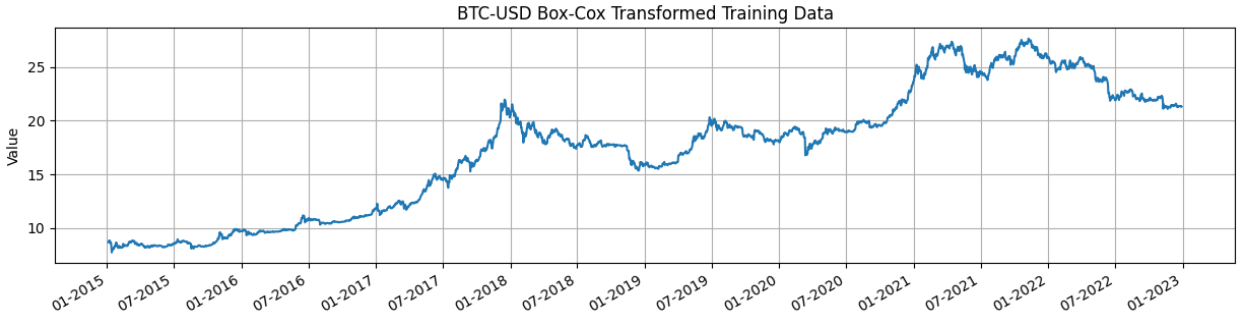


Figure 7: BTC-USD training data after Box-Cox transformation.

3 Time Series Forecasting Models

3.1 Benchmark models

We use four simple forecasting methods as benchmark models: mean, naive, seasonal naive, and drift method.

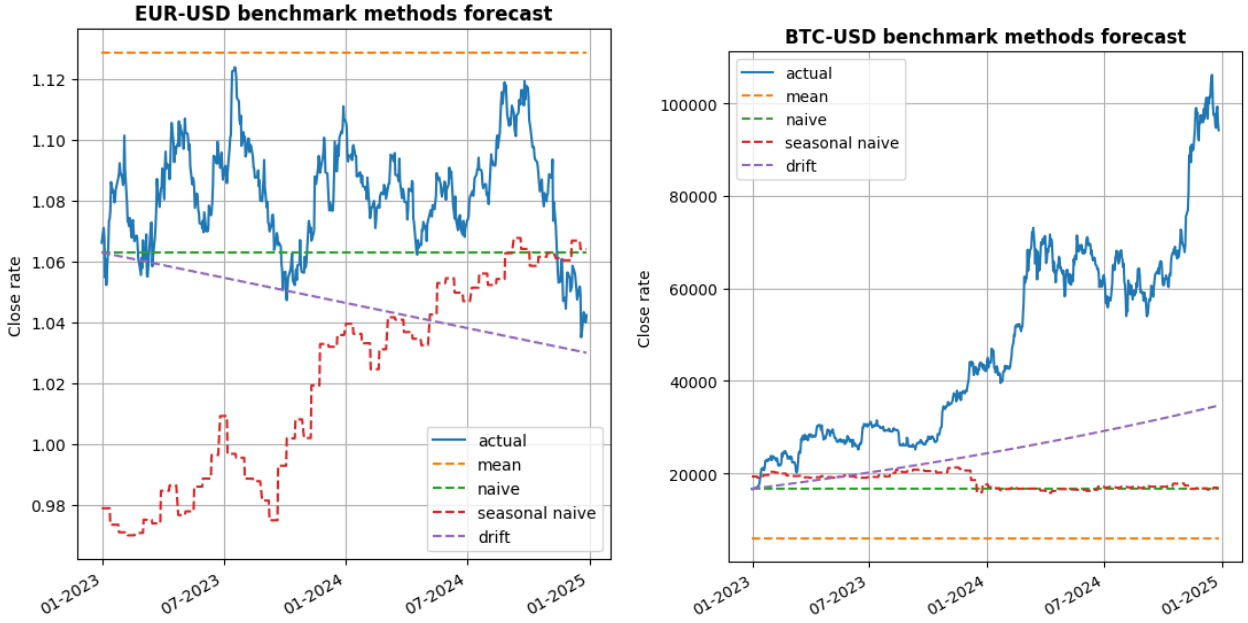
1. Mean method: All forecast values are equal to the mean of training data.
2. Naive method: All forecast values are equal to the last training value.
3. Seasonal naive method: Each forecast value is equal to the last training value in the same season. Here we set period to 60 days for EUR-USD and 90 days for BTC-EUR.
4. Drift method: Forecast values lie on a straight line between the first and last training values.

Figure 8 show the prediction results of these methods and Table 2 shows evaluation metrics, including MAE (mean absolute error), RMSE (root mean squared error), MAPE (mean absolute percentage error), and R^2 (coefficient of determination). We want MAE, RMSE and MAPE to be low, meanwhile a high R^2 is preferred. Looking at the chart and result for EUR-USD, the naive method seems to be the best one based on the metrics, and the naive line matches to the data trend better than other lines. Nevertheless, all methods are underfit to the series, and they even have negative R^2 score. This is possible if we look at the formula $R^2 = 1 - \frac{RSS}{TSS}$, so a negative R^2

[2]https://academic.oup.com/jrsssb/article/26/2/211/7028064
[3]https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.boxcox.html

means the Residual sum of squares ($RSS = \sum(y_i - \hat{y}_i)^2$) is larger than the Total sum of squares ($TSS = \sum(y_i - \bar{y})^2$). This tells us that these methods perform even worse than an average line in the test data. The seasonal naive method is the worst one: the forecast line is far from the actual line and the metrics are opposite to our expectation. This is probably because the chosen season period is not the right one, or the data itself is not seasonal.

Looking at the results for BTC-USD, all benchmark methods still perform badly with high errors and negative R^2 scores. However, in this case the drift method outperforms others as its line follows the upward trend of the signal. The reason is that although the signal is highly volatile, the BTC price in general rises year by year, so a drift line can capture this trend better than other methods. Nevertheless, those simple methods are not good enough to predict time series data and they should be only used as benchmark models.



(a) EUR-USD series.

(b) BTC-USD series.

Figure 8: EUR-USD and BTC-USD forecasts from benchmark methods.

Table 2: Performance metrics of benchmark models.

	EUR-USD				BTC-USD			
Method	**MAE**	**RMSE**	**MAPE**	**R2**	**MAE**	**RMSE**	**MAPE**	**R2**
Mean	0.0462	0.0492	4.2933	-7.4737	41,261.14	46,515.48	84.747	-3.6913
Naive	0.0218	0.0256	1.9993	-1.2954	30,456.77	37,266.58	56.5177	-2.011214
Seasonal naive	0.0630	0.0710	5.8100	-16.6655	28,927.05	36,743.97	51.3051	-1.927350
Drift	0.0359	0.0406	4.7726	-4.7726	22,369.66	27,955.43	40.9356	-6.9447

3.2 ARIMA

ARIMA (AutoRegressive Integrated Moving Average) is one of the most widely used method in time series forecasting. It combines a autoRegressive model (forecast using a linear combination of

past values) and a moving average model (forecast using a linear combination of past errors), along with differencing, so at the end we have the following model:

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q} + \epsilon_t, \tag{2}$$

where y'_t is the differenced series (to any order) and ϵ_t is the series of past errors.

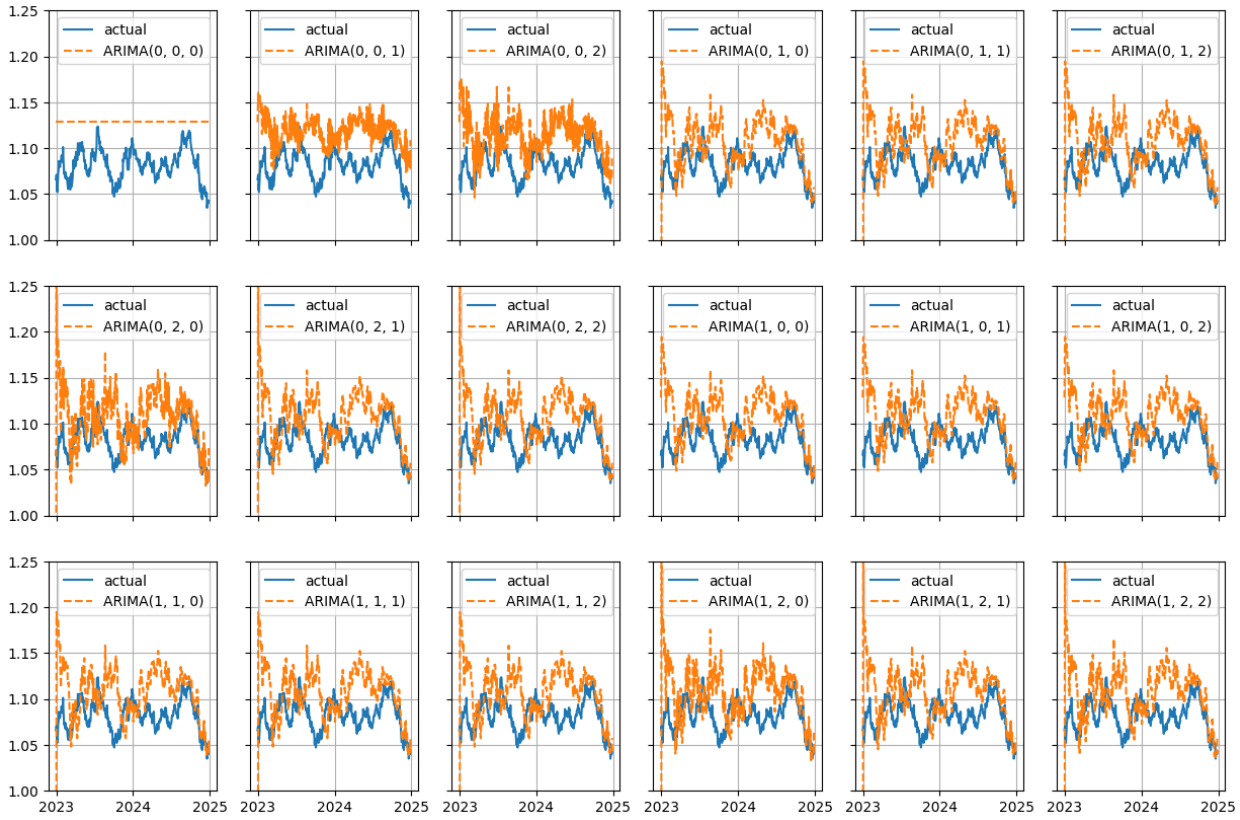
To build an ARIMA model we need to specify three parameters:

1. p: order of the autoregressive part;
2. d: degree of first differencing involved;
3. q: order of the moving average part.

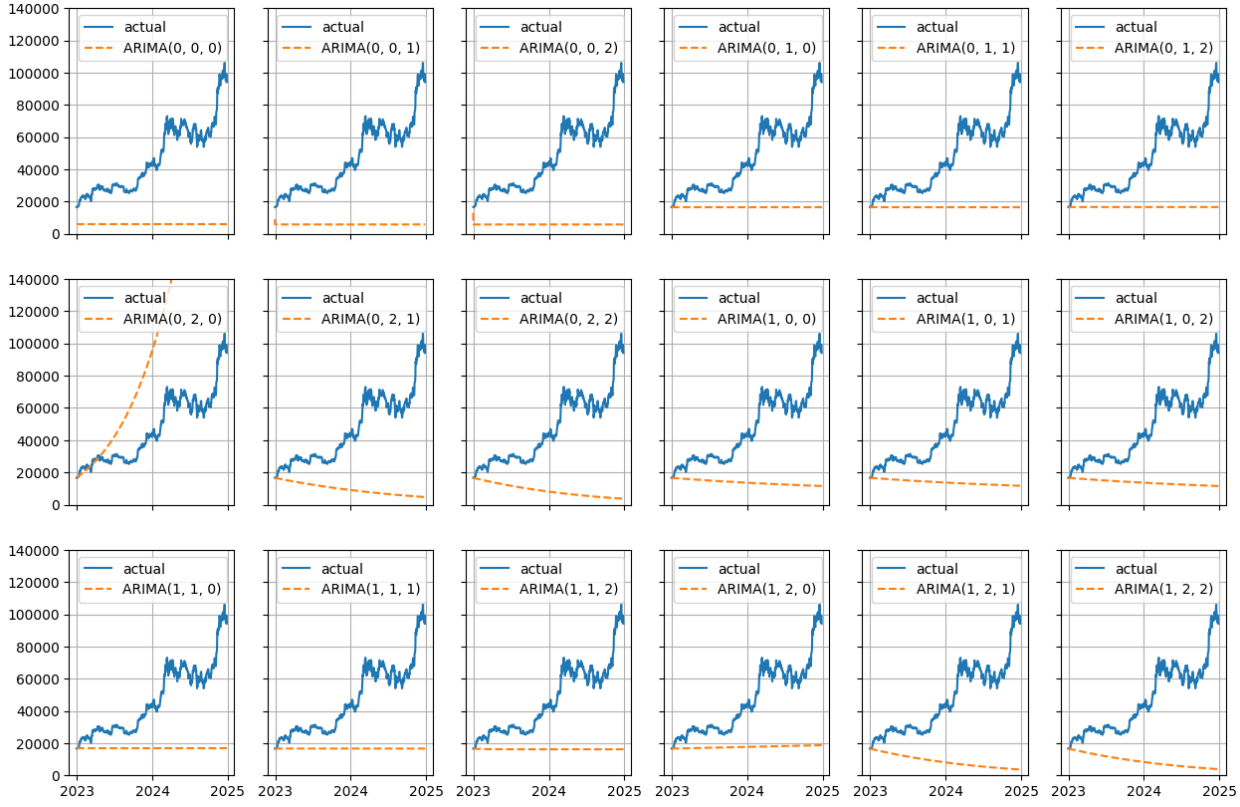
We try with $p \in \{0, 1\}, d \in \{0, 1, 2\}, q \in \{0, 1, 2\}$, so there will be 18 ARIMA models. We report the $RMSE, R^2, AICc$ in Table 3 and the forecast results in Figure 9. The ARIMA(0, 0, 0) is obviously underfit since it is just a straight line of a constant. For EUR-USD, the remaining models predict future values pretty good, among them three models ARIMA(1, 0, 0), ARIMA(1, 0, 1), ARIMA(1, 0, 2) show the lowest $RMSE$ and highest R^2. Their $AICc$ are also the lowest ones, indicating a good choice of parameters. In contrast to EUR-USD, ARIMA models perform worse in BTC-USD data as none of them can capture the movement of test data. Even the best one (ARIMA(1, 2, 0)) still has very high $RMSE$ and looks like a horizontal line rather than an upward line as expected. This performance can be explained considering the highly volatile property of BTC. This cryptocurrency kept its price stable for over a decade (2009-2021), but in recent years it has been increasing in a unpredictable way. So it is safe to say BTC data is non-seasonal, and capturing the volatility in the data is a challenging problem.

Table 3: Performance metrics of ARIMA models.

	EUR-USD			**BTC-USD**		
ARIMA	**RMSE**	**R2**	**AICc**	**RMSE**	**R2**	**AICc**
(0, 0, 0)	0.0492	-7.4721	-6242	46,515.49	-3.69	18434
(0, 0, 1)	0.0424	-5.2942	-8846	46,515.86	-3.69	14502
(0, 0, 2)	0.0412	-4.9514	-10653	46,516.23	-3.69	11060
(0, 1, 0)	0.0614	-12.1891	-15579	37,266.58	-2.01	-3061
(0, 1, 1)	0.0614	-12.1891	-15577	37,268.10	-2.01	-3061
(0, 1, 2)	0.0614	-12.1887	-15575	37,269.56	-2.01	-3060
(0, 2, 0)	0.0693	-15.8348	-14130	114,310.20	-27.33	-975
(0, 2, 1)	0.0685	-15.4475	-15559	44,824.82	-3.36	-3053
(0, 2, 2)	0.0685	-15.4476	-15557	45,660.67	-3.52	-3052
(1, 0, 0)	**0.0399**	**-4.5663**	**-15584**	40,388.98	-2.54	-3050
(1, 0, 1)	**0.0399**	**-4.5666**	**-15582**	40,288.05	-2.52	-3049
(1, 0, 2)	**0.0399**	**-4.5663**	**-15580**	40,391.48	-2.54	-3049
(1, 1, 0)	0.0614	-12.1891	-15577	37,268.19	-2.01	-3061
(1, 1, 1)	0.0614	-12.1890	-15575	37,267.71	-2.01	-3059
(1, 1, 2)	0.0614	-12.1888	-15574	37,627.05	-2.07	-3063
(1, 2, 0)	0.0691	-15.7249	-14711	**36,115.99**	**-1.83**	**-1895**
(1, 2, 1)	0.0685	-15.4476	-15557	45,670.10	-3.52	-3052
(1, 2, 2)	0.0689	-15.6177	-15366	45,468.30	-3.48	-3051



(a) EUR-USD series.



(b) BTC-USD series.

Figure 9: ARIMA forecasts for EUR-USD and BTC-USD series.

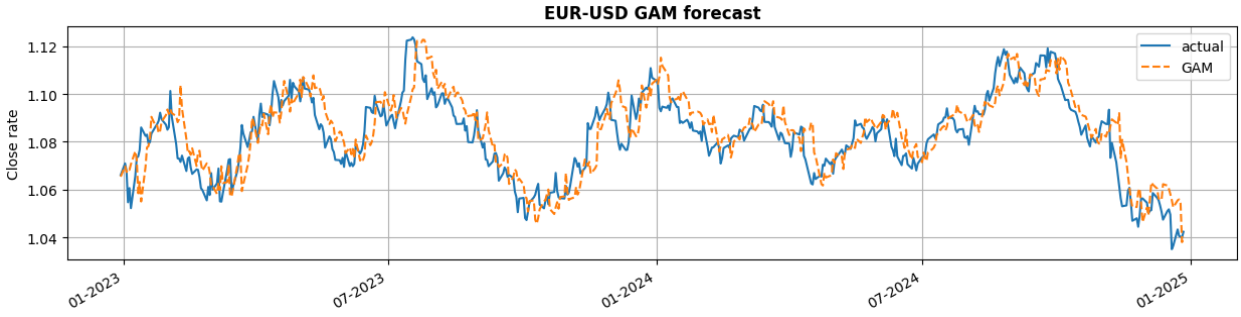
3.3 GAM

A Generalized Additive Model (GAM) is an extension of conventional a linear regression model and allows non-linear relationship between features and target values. The value at time position i is calculated as:

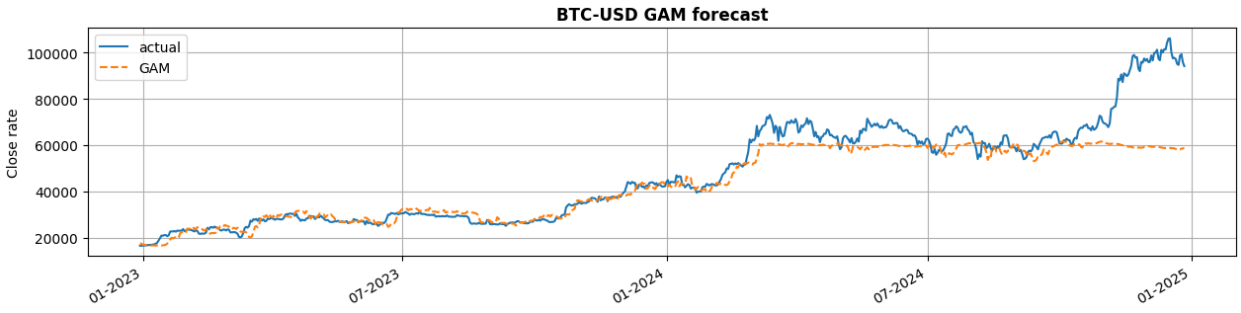
$$y_i = \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \cdots + f_p(x_{ip}) + \epsilon_i, \tag{3}$$

where $f_j(\cdot)$ is nonlinear functions applied to feature x_{ij}. For financial data, we extract year, month, day, and weekday (i.e. Monday,..., Sunday) from the date as features. We also take value at 5 days and 7 days ago in the past as a lag feature for EUR-USD and BTC-USD, respectively, assuming that the data is weekly seasonal. We then fit a GAM using spline terms on these features and show the forecast results in Figure 10 and the metrics on Table 4.

We immediately see that this method fits the data much better than previous ones. For EUR-USD, the predicted signal follows closely to the true signal, but it is delayed a bit because we used the value in the past as a feature. $MAE, RMSE, MAPE$ metrics are small, and R^2 is positive and quite large, indicating an excellent performance. For BTC-USD, GAM model captures the data very well until the beginning of 2024, after that it stays stable. This is acceptable if we consider the fact that in the last year, nobody expected the BTC price would rise fast and significantly like that, especially at the end of 2024. The GAM model has do its best in this case. The metrics are as our expectation: $MAE, RMSE, MAPE$ are low (compared to the data scale), and the $R2$ score is even higher than that for EUR-USD. So far, GAM outperforms all previous methods in forecasting our currency rate data.



(a) EUR-USD series.



(b) BTC-USD series.

Figure 10: GAM forecasts for EUR-USD and BTC-USD series.

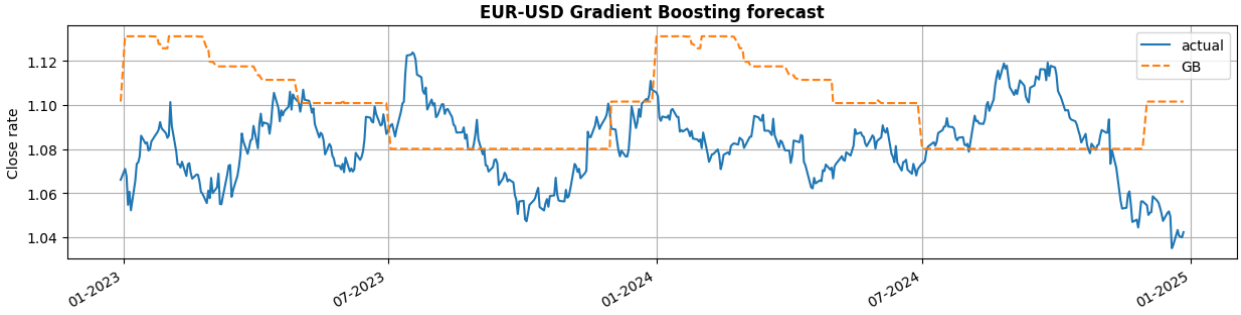
Table 4: Performance metrics of GAM models.

EUR-USD				BTC-USD			
MAE	**RMSE**	**MAPE**	**R2**	**MAE**	**RMSE**	**MAPE**	**R2**
0.0076	0.0096	0.7	0.6777	5452.390	10,530.7	8.8757	0.75955

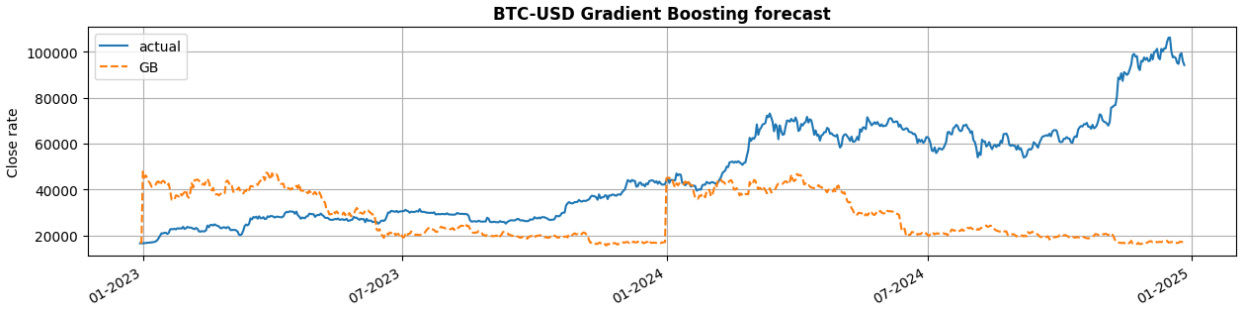
3.4 Gradient Boosting

Gradient boosting is an algorithm that build consecutive models in which the latter model(s) rectifies the errors present in the former model(s). Gradient boosting has demonstrated robustness in handling diverse feature distributions, and no Box-Cox transformation is needed for model training. Here we use an Extreme Gradient Boosting (XGB)[4] regression model, a scalable and commonly used gradient boosting model to fit our data. Features extracted from the data include year, month, day, and weekday. During the model training, we apply the grid search to find the optimal estimator hyperparameters. Our tuning hyperparameters include:

- `eval_metric`: the metric to be used for validation data
- `n_estimators`: the number of boosting rounds or trees to build
- `learning_rate`: the rate of a parameter update
- `gamma`: the minimum loss reduction required to make a node split.
- `max_depth`: the maximum step that each tree's weight estimation is allowed.



(a) EUR-USD series.



(b) BTC-USD series.

Figure 11: Gradient Boosting forecasts for EUR-USD and BTC-USD series.

[4]https://xgboost.readthedocs.io/en/stable/index.html

Table 5: Performance metrics of Gradient Boosting models.

EUR-USD				BTC-USD			
MAE	**RMSE**	**MAPE**	**R2**	**MAE**	**RMSE**	**MAPE**	**R2**
0.0267	0.0319	2.4736	-2.5602	24,901.64	32,061.15	49.16126	-1.2287

Figure 11 shows the forecast values produced by Gradient Boosting models, and Table 5 show the metrics. The predictions were made by the best estimator found by the Grid Search procedure. For both EUR-USD and BTC-USD, the predictions are badly deviated from the true data and the model is unable to capture the general trend in the test data. The predicted fluctuations are not accurately aligned with the actual data jumps and they are even opposite to the true movement. Negative R^2 score show that the predictions are worse than a simple mean line.

3.5 LSTM

So far, all implemented methods were able to match the test data closely and there is only one having a positive R^2 (GAM). However, this method still struggles in following the surprising increase at the second half of the test data. In this section, we will try a widely used neural architecture called Long-Short Term Memory (LSTM)[5], which has proved its efficiency in time-series forecast. Before training the LSTM, we need to preprocess the time series data as follows. First we normalize the data using a Min-Max scaler, which performs the operation:

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}}, \tag{4}$$

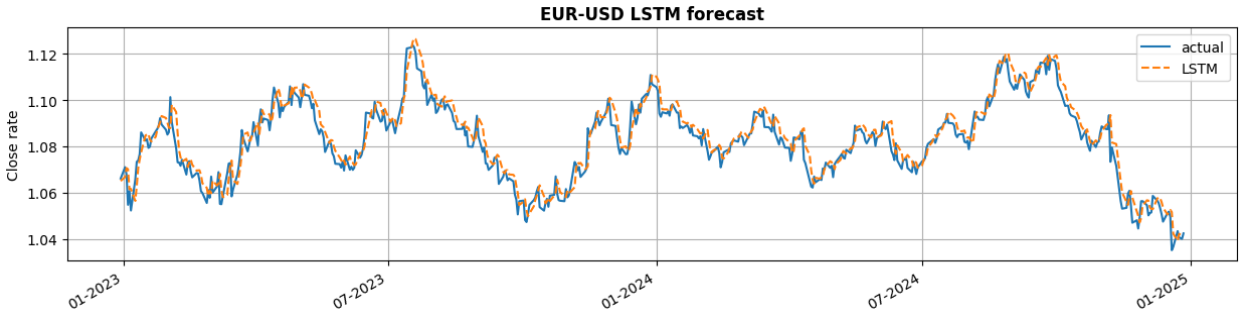
which changes the range of data to $[0, 1]$. This is necessary to guarantee the model training can work without processing too large numbers. The reason is because the model is trained using an optimizer to update the parameters based on the gradient of the loss function over them, so that after each update, the loss function decrease. Calculating gradients which involve large numbers will ruin the procedure, so we want to keep the input values in a small range, but also not too small to avoid the gradient vanishing phenomenon (occurs when multiplying many small values and the computer is not able to represent too small numbers correctly). Moreover, it should be noted that we use the training data to get the scaler, and use this scaler to normalize the test data because the test data is assumed to be unknown in advance.

Next, we choose a window size value, which equals 20 for EUR-USD and 30 for BTC-USD. We use this to create a window that slides along the data, so that all values in a window together will be a training sample and the value right next to the window will be a target. The intuition behind is that we use all values in the last 20 or 30 days as the input to the network to predict the current value. Then, we build a simple LSTM network with a LSTM layer and a Fully Connected layer with one node to compute the output. There is no activation function before the output because the target values are not limited in any range. We train the network with Adam optimizer[6] and MSE loss function. Finally, future values are predicted by the model before being transformed back to the original scale by the computed scaler.

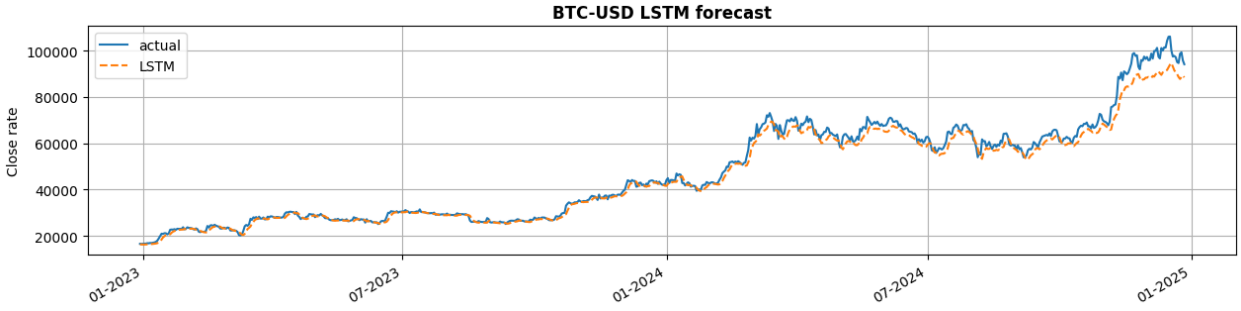
Figure 12 show the predictions made by LSTM and Table 5 provide the corresponding performance metrics. We immediately see that for both EUR-USD and BTC-USD, the prediction line is aligned closely to the actual line, confirming th efficiency of LSTM. Although we only use one LSTM layer here, the result we obtained is impressive: all errors are close to zero, and R^2 is over

[5]https://www.bioinf.jku.at/publications/older/2604.pdf
[6]https://arxiv.org/abs/1412.6980

0.9 for both data. There is still a minor gap between the predicted line and the actual line at the end of BTC-USD period, when the predictions do not reach the true value. As discussed before, in the last quarter of 2024, the BTC price increased significantly which was out of any expectation. Therefore, it is difficult for any model to predict this behavior.



(a) EUR-USD series.



(b) BTC-USD series.

Figure 12: LSTM forecasts for EUR-USD and BTC-USD series.

Table 6: Performance metrics of LSTM models.

EUR-USD				**BTC-USD**			
MAE	**RMSE**	**MAPE**	**R2**	**MAE**	**RMSE**	**MAPE**	**R2**
0.004	0.0051	0.369	0.908	195.08	3056.49	3.4533	0.9797

4 Conclusions

In this project, we implemented various models to forecast the financial time-series data. Simplest models include mean, naive, seasonal naive, and drift methods; more complicated models include ARIMA models, Generalized Additive Model, Gradient Boosting, and LSTM. We uses two pairs of currency exchange rate: a stable one (EUR-USD) and a highly fluctuated one (BTC-USD). As expected, all methods have trouble in fitting and predicting BTC-USD data due to its high volatility. The experimental results show that LSTM model, a neural network, significantly outperforms all other methods and can predict the test data nearly perfect. The second best model is GAM, which can predict EUR-USD data fairly well but is unable to catch the highly volatile BTC-USD data. In summary, neural networks should be used in practice due to its excellent performance. However, when using neural network to fit any data, we must consider the training complexity and the availability of computing resources.