Faculty of Natural,
Mathematical & Engineering Sciences
Department of Informatics

7CCSMPRJ

Individual Project Submission 2021/22

Name:	Tianshu Chu
Student Number:	21036812
Degree Programme:	MSc Computational Finance
Project Title:	Denoising Time-Series Data via EMD-CAE Model
Supervisor:	Dr De Keijzer, Bart
Word Count:	11540

Plagiarism Statement

All work submitted as part of the requirements for any examination or assessment must be expressed in your own words and incorporates your own ideas and judgements. Plagiarism is the taking and using of another person's thoughts, words, judgements, ideas, etc., as your own without any indication that they are those of another person.

Plagiarism is a serious examination offence. An allegation of plagiarism can result in action being taken under the *B3 Misconduct Regulations*.

I acknowledge that I have read and understood the above information and that the work I am submitting is my own.

Signature: Tianshu Chu **Date:** August 8, 2022

Department of Informatics
King's College London
WC2R 2LS London
United Kingdom

Denoising Time-Series Data via EMD-CAE Model

Tianshu Chu
Student Number: 21036812
Course: MSc Computational Finance

Supervisor: Dr De Keijzer, Bart

Thesis submitted as part of the requirements for the award of the MSc in Computational Finance.
7CCSMPRJ - MSc Individual Project - 2022

ABSTRACT

Effective data mining can increase market efficiency and individual profitability. Nevertheless, market trading data is filled with noise due to the globalisation of finance and the entrance of small investors. The low signal-to-noise ratio of these data is detrimental to data mining and may result in incorrect conclusions. This paper was inspired by prior research [1] that used the self-supervised learning model to denoise data.

This thesis proposes the EMD-CAE denoising model, a combination of Empirical Mode Decomposition (EMD) and Convolutional AutoEncoder (CAE). This research aims to enhance the performance of subsequent data mining and trading models by using denoised data. And the novelty of this paper consists in applying the CAE model, which is often used for image denoising, to financial data denoising and combining it with another denoising method: EMD. In this project, EMD is used to provide the desired outputs for the CAE model. The results of tests later indicate that this composite model gets better outcomes than single models and tackles the problem that pure financial trading data is hard to access.

In order to evaluate the model's validity, a multi-faceted, multi-level testing process has been established for the model's examination. Our model outperforms the competition in three tests using eleven datasets with different properties. In both in-sample and out-of-sample tests, the model shows similar good denoising performance. In addition, we prove that not only are the denoised data superior to the original data, but the model trained with the denoised data is also more reliable.

Therefore, the EMD-CAE model is appropriate and effective for denoising tasks in data pre-processing. The denoised data produced by the EMD-CAE model can enhance subsequent data mining.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my supervisor De Keijzer, Bart, who has been extremely helpful over the past six months and has guided me to timely completion of my thesis. I would also like to thank my corporate supervisors Vardan Voskanyan, Vitali Avagyan and others, for assisting me in clarifying the direction of my research and resolving a number of problems I encountered during my internship. Additionally, my friends and family provide me with moral support as I write.

I would also like to thank Yanqing Ma, the author of the key reference for this paper, whose research inspired this project. This article offers a new viewpoint on noise reduction in financial data. The modelling of this paper was greatly aided by the findings of their research.

1 Introduction

1.1 Background

The efficient market hypothesis was developed by Fame [2] in 1970. In an efficient market, the study says, share prices already represent all available information, making prior data useless for future trade. However, this theory is based on stringent hypotheses. Hence the views expressed deviate from actual financial market phenomena.

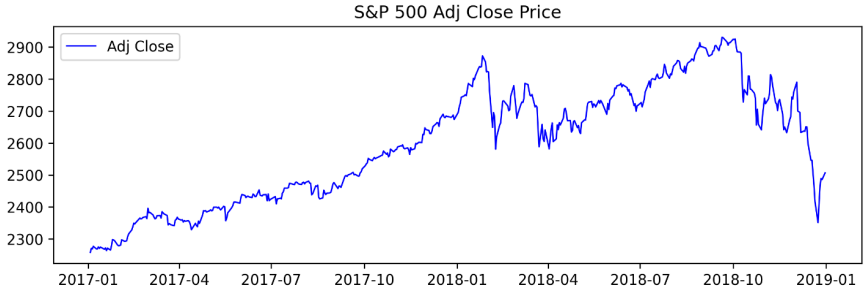
Later in "Prospect theory: An analysis of decision under risk", Daniel Kahneman [3] proposed behavioural finance, emphasising that researchers can not ignore the impact of human behaviour on share prices, which is a very different perspective compared with the efficient market hypothesis. This theory better explains certain irrational market phenomena.

Through the study of historical stock price data, Thaler [4] [5] discovered some intriguing phenomena, such as the January effect (some stocks can earn more in January) and the mean-reverting walk. In addition, Thaler's research [6] revealed that specific periods significantly impact trading prices, including the weekend, holidays, the beginning and end of the month, and the intraday effect.

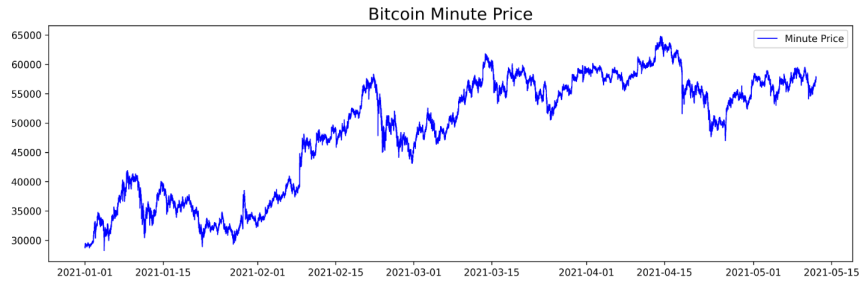
Jegadeesh [7] examined Thaler's theory and proposed "momentum" as an explanation for this type of price change. In contrast to data bias, the research demonstrates that this type of stock momentum still exists.

Behavioural finance scholars have demonstrated that mining historical data can aid in comprehending price changes in financial markets. So it makes sense to do data mining on financial data. In addition, as technology and financial markets have evolved over the past few decades, a vast amount of financial time series data have been produced. These data are typically noisy or difficult to use directly, so their effective utilisation is a crucial topic or demand.

Prior to the popularisation of personal computers, traditional time series statistical models played a crucial role in financial trading decisions. Typically, this method requires strict hypotheses, such as stationarity, white noise, Etc. They include the Auto-regressive model [8], the Auto-regressive Moving Average model [9], and others. Due to theoretical and actual data discrepancies, forecasting outcomes could be subpar. Financial data in real markets is non-stationarity and not homoscedastic random variables. As shown in the graph 1.1 below, both daily and minute financial data are non-stationary and noisy.



(a) S&P 500 Daily Price



(b) Bitcoin Minute Price

Figure 1: Price Plots

Financial trading data is one of the most prevalent types and is affected by various macro and micro factors. With the widespread availability of mobile devices, a growing number of individuals are able to trade anywhere, at any time. In addition, unpredictable news and events, as well as the ignorance of individual investors, have transformed the trading financial data into a disorderly, noisy mess. Black [10] discussed noise in financial data in his paper, which was also titled "Noise." According to him, market noise always exists and makes market transactions possible. Moreover, noise from numerous minor events may have a greater impact than significant events. As a result, individuals struggle to distinguish between signal and noise. Sometimes, people even trade based on noise, which would result in financial loss if informed traders entered the market.

The presence of noise makes market trading profitable but also makes it difficult to obtain excess returns. Not only does the low signal-to-noise ratio of financial time-series trading data affect the output of trading models, but it may also increase the time required for models to learn the pattern. Consequently, data pre-processing is crucial. Effective removal of noise from the data enables traders to gain a significant informational advantage and generate higher returns.

1.2 Motivation

This project's objective is to enhance the performance of trading models by concentrating on the design of a denoising model. Noise is unavoidable in the current global financial markets. In 1990, Long [11] showed that not only do those unpredictable events create noise but traders based on noise increase market noise. This type of traders attempted to earn more on the market by taking significantly greater risks. Their paper utilised the model with noisy traders to explain some peculiar financial market phenomena.

It is essential for general traders or those who do not wish to take excessive risks to search for the true trading signal within noisy trading data. Otherwise, this soiled information could harm standard trading models or generate false results. In contrast to other types of data, such as seismic waves or electromagnetic signals, it is difficult to define financial noise. Small fluctuations can be viewed as noise when prices are volatile. However, these may not be noise in a different period or stock. Therefore, trading data noise is dynamic and subjective, necessitating the use of adaptive algorithms or models to identify noise patterns.

Due to the difficulty in defining trading noise, it is impossible to precisely calculate the signal-to-noise ratio of trading data before and after noise reduction. Because of this, the performance of specific trading models or tests can be substituted for evaluating the denoising model. These methods may be applicable for evaluating models of financial denoising. As long as the performance of the reconstructed data after denoising is superior to the performance of the original data, this indicates that the reconstructed data contains more signal and less noise.

Ma et al. [1], who proposed a self-supervised learning model that combines Moving Average(MA) and Convolutional AutoEncoder (CAE), are the primary source of inspiration for this project. They transferred the CAE model [12], typically used for image denoising, to denoise one-dimensional financial data in that article. Their article demonstrated that they obtained favourable outcomes in tests that used denoised data to predict denoised labels and reconstruct buying signals using trading strategies. To avoid confusion, this model proposed by Ma will be referred to as the MA-CAE model from now on.

This project follows in their footsteps and attempts to improve upon their work.

1.3 Contribution

This project proposes a new self-supervised learning model for financial data preprocessing and dubs it the EMD-CAE denoising model since it combines the benefits of Empirical Mode Decomposition (EMD) [13] and Convolutional AutoEncoder (CAE) [12]. The key reference from Ma et al. [1] provided the basic framework for this paper. And the denoised data of the EMD-CAE model demonstrates superior performance compared to the data of the MA-CAE model or the original, noisy data. This project utilised three distinct assessment tests to evaluate the model's denoised results more effectively: regression tests, classification tests, and trading strategy tests. The EMD-CAE denoising model continues to perform well across multiple data sets and tests.

1.3.1 Summary results from the model

1. **Take small data sets of daily closing prices as examples:**
    - Better performance compared with original data on three separate tests. (Regression, Classification, and Trading strategy testing)
    - Stable performance on nine daily price data from four financial markets. (Index, Stock, Foreign Exchange (FX), Future)
    - Better performance compared with other models, using data at three different frequencies. (Daily, 1-minute, 20-minute)
    - Good generalization ability with data from two distinct fields. (Finance, Traffic)
2. **Observable performance on three different tests:**

    Regression test:

    We use Mean Squared Error (MSE) as an evaluation indicator to assess the mismatch between the true and predicted values.

    - In 89% of the data sets, denoised data performs better.
    - Denoised data can improve MSE values by up to 43%.
    - The results of denoised data achieve a 12% average improvement in MSE values across all daily data sets.
    - Denoised data achieve stable improvements at 89% out-of-sample tests.

    Classification test:

    We chose the F1 score as the model's evaluation criterion for the classification test.

- Denoised data performs better on 100% of the data sets.
- The F1 score of denoised data can rise by up to 200 percent under specific circumstances (some fixed labelling methods).
- Denoised data achieve stable improvements at 100% out-of-sample tests.

Trading strategy test (Moving Average Convergence Divergence):

Here, we use straightforward backtesting to calculate the trading model's simple return based on MACD.

- Denoised data performs better in 100% of the data sets.
- Denoised data has up to 40% simple return improvement.
- A positive return remains possible via denoised data even if the original data result is negative.

1.4 Structure

This article contains six chapters describing the EMD-CAE model's design and testing process.

The chapter above introduces the background and motivation of doing data preprocessing for trading data. After reading previous literature, we obtain the inspiration for this project. Afterwards, we summarise the contribution of the EMD-CAE model briefly.

We classify common denoising techniques in the following Chapter 2. In addition, we review the research procedure for the primary techniques applied in this paper. Finally, we provide a concise description of the core reference [1] for this paper in order to make comparisons in the following chapter.

In Chapter 3, some definitions are provided to help readers understand this project. We also present the models used in the following testing process.

In Chapter 4, we describe how to construct the EMD-CAE denoising model and how to test the denoised and original data. We compare the differences and improvements between the EMD-CAE model in this paper and the MA-CAE model in the reference paper [1]. Then, we present the data sets used in this paper. Eventually, we will answer questions about possible threats to this project.

In Chapter 5, we present the denoising outcomes of the EMD-CAE model and comparisons with other algorithms. We demonstrate the advantages of EMD-CAE models from multiple perspectives.

In Chapter 6, we conclude the whole project and discuss possible future directions for the EMD-CAE denoising model.

2 Literature Review

This chapter will briefly introduce denoising techniques' categories before describing in depth the empirical mode decomposition and convolutional autoencoder research processes.

2.1 Denoising method

As stated in section 1.1, Black's research [10] revealed that noise is a component of financial data and increases trading risk. Consequently, it is necessary to find a way to remove data noise. Not only the financial field but also other fields are affected by noise. To minimise noise, researchers have employed a variety of techniques.

1. Binning method
   It is a method for partial smoothing [14]. The essence of binned data is substituting the target data with nearby data, such as the mean, median, or boundary value. This method attempts to bin and smooth the noisy data for denoising. Simple moving averages (SMA) and Exponential moving averages (EMA) are two of the most popular tools.

   This method has the advantage of being calculated quickly, but it only uses data from the neighbourhood and replaces it by using fixed functions. As a result, this method can not comprehend global noise and may eliminate both noise and signal.

2. Clustering method
   This denoising method divides the data into multiple sections containing similar data. It is considered as noise if the data does not belong to a domain or is far from the nearest cluster centre. This method is also very useful in outlier detection. Most of them are unsupervised data mining techniques which are popular in the current data mining community. K-means [15] is one well-known clustering algorithm introduced in 1979. This algorithm first selects k data points as centroids and divides the unlabeled data into k parts. Then, update the centroids and assign each value to a separate part. Other clustering algorithms include k-means++ [16], DBSCAN [17], Etc.

   This class of methods has the advantage of being based on the entire data set and not requiring labels. However, this does not work for tiny noises, and the results would vary depending on the initial centroids chosen each time.

3. Regression method
   This method denoises the data by learning the data and obtaining a function or a set of mathematical equations to reconstruct the data. The reconstructed data is regarded as the denoised data. There are numerous regression methods, which can be divided into linear and non-linear approaches. The most well-known model is linear regression [18], which uses a hyperplane to fit the data. For non-linear ways, one kind of non-linear method is boosting algorithms, which is the general name for a class of models, like XGBoost [19], CATBoost [20], Etc.

   For the benefit of this method, linear and non-linear data have corresponding approaches. Its disadvantage is that the fitted results frequently lose excessive details. This method is also inappropriate for one-dimensional data denoising. However, if using more features, more noise may also be brought into the data.

4. Decomposition method
   This algorithm treats the original data as a summary of various frequencies and is primarily employed in signal processing. Using specialised functions, it can decompose noisy data into sub-data in various frequency bands. The reconstructed data are the sum of the remaining bands after removing the noise band. More information and models will be presented in section 2.2.

5. Deep learning method
   The field of deep learning denoising is expansive. The majority of these studies concentrate on image denoising. Liu Peng [21] proposed Multi-level Wavelet-CNN to denoise images, which combines the wavelet method and convolutional neural network, and obtained favourable results. In a paper published in 2019, Zhao [22] et al. proposed a pyramid denoising model. This method used the neural network properties that are good at extracting features. In addition, autoencoder-based models are an integral part of the field of deep learning denoising because they can effectively mine data features and reconstruct data. The process of autoencoder development is described in detail in section 2.3.

2.2 Empirical Mode Decomposition

Huang proposed Empirical Mode Decomposition (EMD) in 1998 [23] as the auto-adaptive method based on Fourier transform [24], which decomposes the original data into cosine and sine functions with varying frequencies. However, the Fourier transform is based on the strict requirement that the target data should be stationary. For the

traditional Fourier transform, the data can be decomposed by frequency, but it cannot be applied to time-varying data, where data vary in time and frequency. This is because the Fourier transform uses trigonometric functions to decompose the data, and trigonometric functions are infinitely long and stable in time. Later, a novel technique known as the short-time Fourier transform [25] was developed. That paper used partial stationarity to compensate for the shortage of traditional Fourier variation. Consequently, the size of the splitting window plays a vital role in this algorithm. The method performs well in the frequency domain when the window size is large. Otherwise, it performs well in the time domain. In short, it can not perform well simultaneously in the time-frequency domain. Fourier transform cannot perform well in the time and frequency domains, which is its disadvantage.

To deal with this shortage, wavelet transform was proposed in 1974 by Morlet [26]. The most significant difference between the wavelet transform and the Fourier transform is the function, which is the tool to decompose data. The wavelet transform uses wavelets as functions, and the Fourier transform uses trigonometric functions. Wavelets differ entirely from trigonometric functions, which have infinite lengths and repetitive shapes. The difficulty of decomposing data in the time-frequency domain is resolved by using wavelets of limited length and decay over time. The wavelet transform addresses some of the limitations of the Fourier transform, as it can handle non-stationary data and locate the signal in the time domain. This method has the disadvantage that the choice of wavelet functions and the number of sub-data will affect the decomposition outcome. Therefore, researchers must make manual selections based on different data.

The following well-known decomposition algorithm is EMD [23], which can also work for non-stationary data in the time-frequency domain. This method decomposes the original data into Intrinsic Mode Functions(IMF). Then, the EMD algorithm will keep doing this until no more IMF is available. A significant advantage of EMD is that it automatically decomposes the data without requiring manual parameter input. EMD has its own shortcomings as well. And for some data sets, there is a phenomenon known as mode mixing, in which, EMD cannot separate two IMF due to intermittent signals, pulse interference, or some other factor. The following figure 2 is a sample result of decomposing data of S&P 500 from 2017 to 2019 via EMD.

Figure 2: S&P 500 via EMD

To solve this phenomenon, Wu [27] put forward Ensemble Empirical Mode Decomposition(EEMD), which tried to solve the problem by adding white noise. So, they would change the original data's extreme values and help avoid mode mixing. In 2010, Complementary Ensemble Empirical Mode Decomposition (CEEMD) was raised by Yeh [28] based on EMD and EEMD. It added positive and negative white noise to minimise the noise left in IMFs.

2.3 Convolutional Autoencoder

Rumelhart first proposed the idea of autoencoder in 1986 [29]. In that paper, they argued that an essential characteristic of autoencoder is that it requires two inputs, one of which is the desired output and the other the neural network's input. The back-propagation algorithm was then used to adjust the network to minimise the loss between the network output and the desired output.

Later, Bourlard [30] (1988) gave a clear definition of autoencoder, referring to the previous study. The autoencoder is a neural network-based system with multiple hidden layers. For the autoencoder model, the input layer and output layer must have the same shape so that the model can compare the network's output to the input called desired output. Due to their simple construction, stackability, and ability to extract

deep features, autoencoders have numerous applications in various fields. For example, the model can be used to image classification [31] [32], outlier detection [33] [34] and data reconstruction [35] [36].

Traditional autoencoder can be divided into two parts: encoding layers and decoding layers. Additionally, the number of each type of layers must match to make the shapes of the input and output the same. As shown below, this figure 3 shows a brief schematic of a single autoencoder. The objective of the autoencoder model is to reconstruct the input data and make them similar to the desired output. The hidden layers are the core of this model, and its encoding layer obtains features from data. The greater the number of hidden layers, the more deeply the model can extract features from data. The model then uses the decoding layers to reconstruct data based on these features. Typically, the number of units in the hidden layer is less than that in the input layer, enabling the autoencoder model to extract knowledge and patterns from data. Because the reconstructed data is based on the features, it cannot perfectly match the desired output, even if the desired output and input are the same.

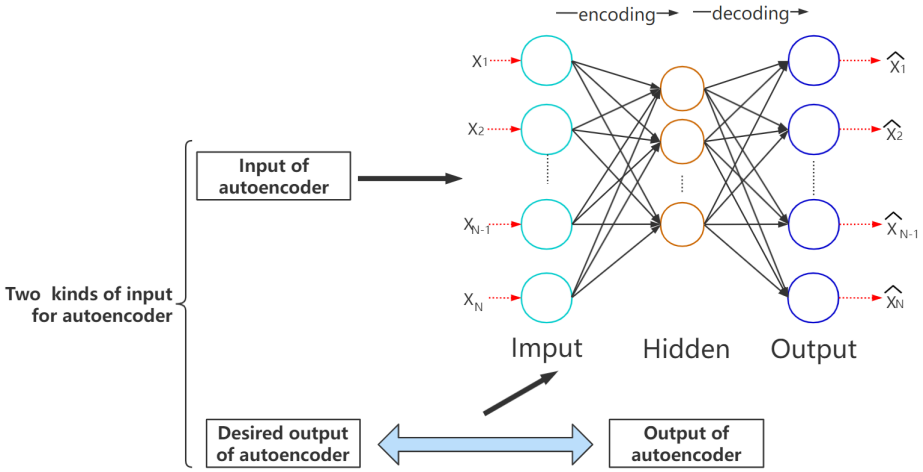


Figure 3: Traditional Autoencoder model

Many researchers proposed several new autoencoder models based on traditional autoencoder model. Vincent et al. [37] put forward denoising autoencoder. They added some noise in the original images and set the noisy data as the input of the model, and the pure data (original data) as the desired output. The denoising autoencoder model can be trained to extract clean data from noisy data.

For the convolutional autoencoder model (CAE), it was proposed by Jonathan [12]

in 2011. Due to the excellent performance of convolutional networks in recent years, they replaced the fully connected layers in the traditional model with convolutional layers. To denoise images, they set the images with noise as the input of the model and the clear images as the desired output. This model achieves excellent image denoising results.

Following in Jonathan's footsteps of research, CAE models are widely used in the field of data denoising. In 2015, Stowell [38] et al. used this model to denoise birdsong audio. For this kind of data, the article's authors emphasised that it is impossible to obtain pure sound data, as it would be mixed with environmental sounds and challenging to distinguish. Then, experimental results in that paper showed that the CAE model can effectively denoise audio data in the absence of pure data. Nishio [39] used the CAE model to denoise CT images. Their results show that the CAE model can perform well in the healthcare field. Zhou [40] (2021) used the part of noisy data as the desired input to train the CAE model. As they said in the paper, they found this model could perform better than other denoising methods and minimise signal loss while deleting noise. This article demonstrates that CAE models can effectively denoise seismic data using the same inputs.

2.4 Key reference: MA-CAE denoising model

In numerous fields, such as seismic signals, audio signals, and medical images, a vast number of denoise algorithms play a significant role. These algorithms have their own benefits and drawbacks and are suitable for various denoising applications. Hence, which algorithm is appropriate for financial trading data?

In this project, we follow an approach proposed by Ma et al. [1]. They were the first to use a self-supervised learning model for financial time data, which was the MA-CAE model combining the moving average and convolutional autoencoder. As a result of their research, their denoised data can do better in the task of predicting denoised price trends. This project replicates their research to understand their project better, and this is the brief flowchart 4 of the MA-CAE model training process.

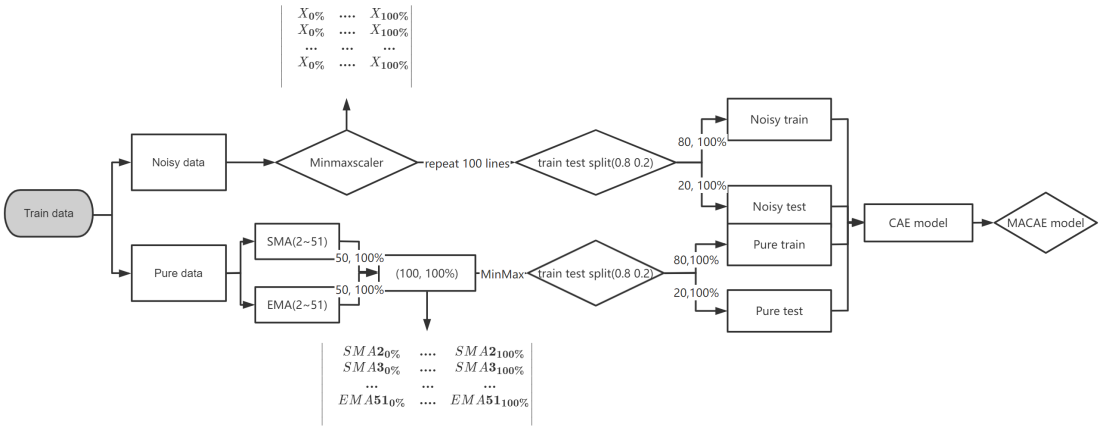


Figure 4: MA-CAE model

That paper skillfully uses the CAE model's characteristics to set the output of the moving average algorithm as the desired output and the original data as the CAE model's input. To make the following chapter more readable, the desired output will be referred to as the pure input, and the input of the CAE model will be referred to as the noisy input.

The original data for the MA-CAE model is one-dimensional. Then, they utilised various algorithms to generate both pure and noisy input. For the pure input, they utilised SMA and EMA to expand the original one-dimensional data into a 100-row matrix, while the window sizes of the moving averages ranged from 2 to 51. After which, they replicate the one-dimensional data for the noisy input one hundred times to obtain a matrix with the same shape as the pure input. With this method, the model was trained and tested with great outcomes.

3 Background

This chapter aims to introduce some project-related definitions. These definitions can aid comprehension of the paper and provide an overview of the entire project.

3.1 Time series data

Time series data are observations of a phenomenon arranged in time order to form a sequence [41]. As shown in Figure 5, the horizontal axis represents the time points, and the vertical axis represents the S&P 500 index values corresponding to the time points. The units of time can be seconds, minutes, hours and days. When compiling time, indicators within the same time series must be consistent across time to ensure data comparability. This project focuses on removing noise from such data.

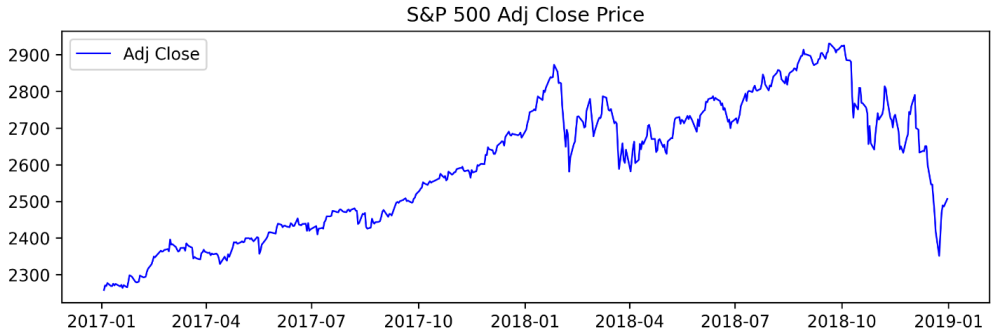


Figure 5: Time series trading data

3.2 Self-supervised Learning

Machine learning models can be categorised based on the presence or absence of labels in the input data: Supervised learning, Non-supervised learning, Semi-supervised learning, and Self-supervised learning. Both supervised and semi-supervised learning require a certain quantity of labelled data, and labelling data manually can be expensive. Self-supervised learning models have gained popularity because they can label data sets using the data sets themselves or specific algorithms. In Xiao Liu et al. [42] research, they summarised the current self-supervised learning models and divided these methods further into three categories: generative, contrastive, and generative-contrastive. In addition, the autoencoder algorithm was placed in the paper's generative section. Nonetheless, some academics consider autoencoder to be unsupervised learning. Since convolutional autoencoder requires two inputs and attempts to minimise the loss be-

tween the output and the desired output by adjusting the network, this paper views the EMD-CAE model as a self-supervised learning model, similar to the reference paper.

3.3 Regression test and classification test

As previously mentioned, defining noise in trading data is challenging, and we can not access pure data. How to test the denoised data produced by the denoising model is, therefore, a concern.

Utilizing historical data to predict future prices is the most common application of trading data. In addition, forecasting methods can be subdivided into regression forecasting and classification forecasting. The first objective is to forecast future price data, while the second is to forecast price trends in the future.

This project explores the potential of denoised data using two models as the regression and classification test. If the denoised data performs better in price forecasting than the original data in both tests, it means that the EMD-CAE model is able to separate noise from the original data without negatively affecting the signal.

3.3.1 The model for regression tests

Extreme Gradient Boosting (XGBoost) [19] is a boosting algorithm 3.3.1 that is an iterative algorithm which converts weak learners into strong learners. This method is based on the principle of gradual improvement through supervised learning. As the number of iterations increases, each weak learner is optimised based on the combination of previous weak learners, which are gradually combined to create an effective strong learner. This model is widely applicable to classification and regression tasks.

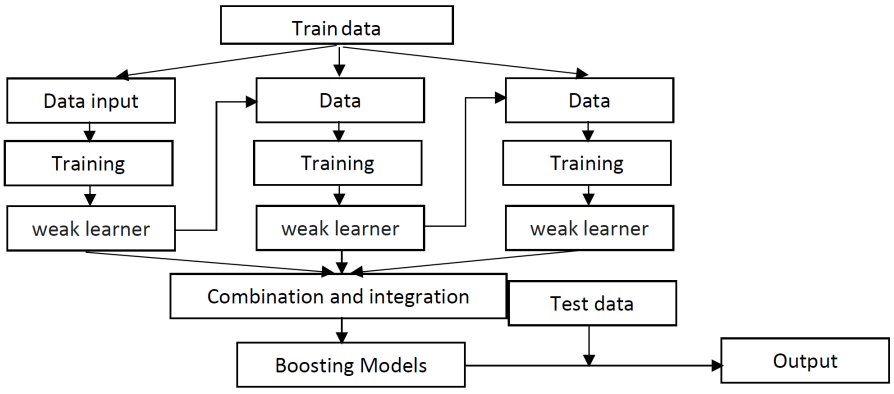


Figure 6: Flowchart of boosting algorithm

The XGBoost algorithm 3.1, which has the same basic principles as Gradient Boosting Decision Tree (GDBT), is shown below:

$$\begin{aligned}
\hat{y}_i^{(0)} &= 0 \\
\hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\
\hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\
\hat{y}_i^{(t)} &= \sum_{k-1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)
\end{aligned} \tag{3.1}$$

The above equations simulate the iterative process of XGBoost, the $\hat{y}_i^{(t)}$ is the model's prediction for round t, the $\hat{y}_i^{(t-1)}$ is the model's prediction for $t-1$ rounds before, $f_t(x_i)$ is the new weak learner and x_i is input data.

The objective function of the XGBoost algorithm model is this formula 3.2:

$$Obj^t = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) + constant \tag{3.2}$$

In this function, Obj^t means it is the objective function at time t. l represents the loss function $l(y_i, \hat{y}_i^{(t-1)})$. In l, the meaning of $\hat{y}_i^{(t-1)}$ is the predicted value from model at time $t-1$, and y_i is the actual value. $f_t(x_i)$ is the new weak learner like that in equations 3.1. And, the $\Omega(f_t)$ is penalty terms to avoid overfitting.

The core formula 3.3 of XGBoost is that it uses the second order derivative of Taylor expansion to improve accuracy and reduce errors.

$$\begin{aligned}
&Taylor\ expansion: f(x+\triangle x) \simeq f(x) + f'(x)\triangle x + \frac{1}{2} f''(x)\triangle x^2 \\
&Define: g_i = \partial_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y}_i^{(t-1)}\right), h_i = \partial^2_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y}_i^{(t-1)}\right) \\
&Obj^t \simeq \sum_{i=1}^{n}\left[l\left(y_i, \hat{y}_i^{(t-1)}\right) + g_i f_t(x_i) + \frac{1}{2} h_i f_t(x_i)^2\right] + \Omega(f_t) + constant \\
&\simeq \sum_{i=1}^{n}\left[g_i f_t(x_i) + \frac{1}{2} h_i f_t(x_i)^2\right] + \Omega(f_t) + constant
\end{aligned} \tag{3.3}$$

Here, $l\left(y_i, \hat{y}_i^{(t-1)}\right)$ can be seen as constant and divided into constant part. Then, the model needs to calculate the rest of the formula.

The preceding is an overview of XGBoost. This model is widely used in regression

forecasting in various fields. XGBoost's Python library is more complex and contains numerous useful functions. We will describe in detail how to implement regression tests later.

3.3.2 The model for classification tests

In this project, we want to use a model to predict the data changing trends: Up, Unchanged, and Down. Here, we choose the same classification approach in the key reference paper [1].

Theoretically, Support Vector Machines (SVM) is intrinsically a binary classification model, but can also be used for multi-classification problems. SVM aims to find a hyperplane which can split the data set into two parts. The desired hyperplane is the one has the longest distance to the nearest point in each part. As data is often not directly divisible in low dimensions, SVM maps data from low to high dimensions to find the desired hyperplane. However, direct mapping increases the computational cost considerably, so different kernels are used to reduce the computation. The kernels are specific functions to achieve vector inner product operations with lower computational complexity. In this project, we use the Radial Based Function (RBF) as the kernel of SVM. It is a function based on radius distance; the example formula is shown below. Assuming the shape of data $X_i = ([x_i]_1, [x_i]_2) \in \mathbb{R}^2$ looks like a circle, so the data fit this formula 3.4:

$$[x]_1^2 + [x]_2^2 = constant \tag{3.4}$$

Then, set $\phi(x) = ([x]_1^2, [x]_2^2) \in \mathbb{R}^2$, and use y to replace x^2. Then, we get the new formula 3.5, which can be seen as a line.

$$[y]_1 + [y]_2 = constant \tag{3.5}$$

By this method, the inner product of the vector can be calculated, and the formula 3.6 of RBF kernel is:

$$\begin{aligned} k(x_1, x_2) &= \langle y_1, y_2 \rangle \\ &= \langle \phi(x_1), \phi(x_2) \rangle \\ &= \langle ([x_1]_1^2, [x_1]_2^2), ([x_2]_1^2, [x_2]_2^2) \rangle \\ &= [x_1]_1^2 [x_2]_1^2 + [x_1]_2^2 [x_2]_2^2 \end{aligned} \tag{3.6}$$

This is a brief introduction to SVM based on the RBF kernel. To address the question of how to construct binary SVMs for the multi-classification problem, common

solutions are One Versus Rest (OVR) and One Versus One (OVO) [43]. For $3(k)$ classifications, the OVR will train $3(k)$ SVMs (SVM_1, SVM_2, SVM_3) to sort, then use the maximum result of (SVM_1, SVM_2, SVM_3) as the final result.

The OVO will train $3(k \times (k-1)/2)$ SVMs: $(SVM_1(AB), SVM_2(AC), SVM_3(BC))$, then vote to determine the final result based on the results of each SVM model. In this project, the OVO method is chosen for the multi-classification problem.

3.4 Trading Strategies and backtesting

In addition to machine learning trading strategies, most traders use a kind of more direct trading strategies, which are called technical indicators. These statistical indicators provide traders with buying and selling signals when the stock price follows a particular pattern.

Most technical strategies calculate a reversion period to the mean value. Theoretically, prices will revert to an average value in the long run, and technical indicators then suggest when the stocks begin to revert to mean value. Depending on different technical strategies, buying and selling signals are possible price change points.

Backtesting is the process of evaluating the viability and efficacy of trading strategies using historical data. With buying and selling signals, it is simple to calculate the strategy's returns using historical data accurately. In this project, we attempt to compare the performance of denoised data and original data using the return from backtesting, which is another way to demonstrate the potential of denoised data.

4 Model design

This chapter describes the model's design and the entire project process. The improvements to the original model are highlighted by comparing the EMD-CAE model to the MA-CAE model of the primary reference in the subsequent section. The following test setups are also described in detail in this chapter.

4.1 EMD-CAE model

This project uses two kinds of denoising methods to create a new denoising model. This section 2.3 shows that the convolutional autoencoder (CAE) needs two inputs: noisy input and pure input. Not only can it reduce irregular noise by extracting features, but it can also use back-propagation to fit and learn the results of another denoising method. In the reference paper [1], they used the moving average results as pure input and got good results in classification tasks. So the CAE model is an excellent platform for adding other denoising algorithms. Besides, Zhou[40] showed in their paper that the CAE model itself could be used to denoise. They generated the noisy and pure input from the same data and found that the CAE model can learn how to delete noise in the noisy input. This phenomenon may be caused by the fact that the CAE model reconstructs data based on the intrinsic features of the data, and noise typically lacks distinct patterns. Therefore, when the data passes through the hidden layer, it undergoes an information compression process. The model then employs features to produce data with the same length as the original data. During this period, information is lost relative to the noisy input, which could be considered noise.

Since the CAE model itself can be used to reduce noise, it may be preferable for the neural network to identify noise when the pure data is genuinely free of noise. Gondara and Lovedeep [44] presented that even the simplest CAE model can denoise the heavily damaged pictures. They utilised algorithms to add noise until the signal could not be distinguished. In this circumstance, pure data is utilised to generate noisy data. So, what to do if we can not access pure data? Stowell[38] et al. used manually labelling methods to label some data which is full of noise. However, this labelling method is time-consuming and impossible for financial time series data. Zhou's[40] research is an ideal way to solve this problem. They generated pure data from the noisy data. In addition, the key reference[1] used the moving average to generate pure input.

Following in the footsteps of these researchers, this project employs Empirical Mode Decomposition (EMD) to calculate pure input from noisy input. Before constructing the pure input matrix, the form of the noisy input must be determined. The reference article

transformed the one-dimensional data into a multidimensional matrix. They believed that this design would avoid overfitting. In our opinion, during training, it may be unreasonable for the MA-CAE model to fit different moving averages if the model always receives the same noisy input. Based on this, this paper constructs noisy input using sliding windows. Moreover, this sliding window method effectively avoids information leakage and helps the EMD-CAE model learn how to denoise from each backpropagation. The noisy input matrix of EMD-CAE model is shown in the following equation 4.1:

$$\begin{vmatrix} x_{0\%} & .... & x_{50\%} \\ x_{0\%+step} & .... & x_{50\%+step} \\ ... & ... & ... \\ x_{50\%} & .... & x_{100\%} \end{vmatrix} \tag{4.1}$$

Here, the length of the sliding window is set to 50% of the length of the whole original data set. The length of each slide is *step*, which can be adaptive depending on different data sets. Then, we keep adding the *step* to the sliding window until the end of the data set is reached.

The shape of the noisy input matrix has now been determined. The subsequent step is the selection of methods to generate pure input. This project utilises EMD because it is applicable to non-stationary time series data and is data-adaptive as opposed to requiring manual adjustment. Despite the disadvantage of mode mixing, it is irrelevant to this project. In this task, we pay no attention to the physical meaning of IMFs or the mixing of different IMFs, because the purpose of using EMD is to generate pure input by removing some noise. EEMD and CEEMD can not be used for denoising in this project because they would add noise to original data.

EMD is a decomposition algorithm which decomposes the original data into several IMFs. The decomposition steps are shown below:

1. Find all extreme values of target data set T.
2. Use cubic splines to link all maximum and minimum values, respectively. Then, get the upper and lower envelopes.
3. Calculate the mean envelope M by using upper and lower envelopes.
4. Get new data set H: H=T-M.
5. If H fits the definition of IMF, it becomes a new IMF. Otherwise, set H as the new target data T and return to step 2.

- The diff between the number extreme and zero crossing values is less than 1.
- Its mean values of upper and lower envelopes should be 0 at any time

6. Set data set T-H as the new target data set T. Check if T is suitable for the following decomposition. If it is, return to step 2. Otherwise, stop decomposing.

   - Not monotonically or constant data

After these steps, some IMFs can be divided, which are still not the pure data wanted. Therefore, the next step is to reconstruct one-dimensional data using IMFs. This is an example 4.1 of the IMFs results of S&P 500 daily price data.

Figure 7: S&P 500 via EMD

In many reference papers [45] [46], they view the IMF with high frequency and small fluctuation range as noise. From the financial market perspective, the impact of noise on prices is also often characterised by the small range and rapid change. So, this project deletes the first IMF, which may contain most of the noise and get pure one-dimensional data by summing the rest of the IMFs. By using this way to reconstruct each raw in the noisy input matrix, we can utilise the noisy input matrix to calculate the pure input

matrix $\widetilde{X}$ 4.2.

$$\begin{vmatrix} \widetilde{x_{0\%}} & .... & \widetilde{x_{50\%}} \\ \widetilde{x_{0\%+step}} & .... & \widetilde{x_{50\%+step}} \\ ... & ... & ... \\ \widetilde{x_{50\%}} & .... & \widetilde{x_{100\%}} \end{vmatrix} \tag{4.2}$$

After getting two input matrices, the next step is to train the EMD-CAE model. This figure 4.1 shows the framework of the training process and the trained model after this process. First, we use the original data to get the noisy input matrix. After this step, we can establish the pure input matrix according to each input in the noisy matrix, as mentioned above. Then, we use the two inputs to train the CAE model. In this step, we minimise the loss between the output $\widehat{X}$ of the CAE model and the output $\widetilde{X}$ of the EMD model (pure input). To avoid information leakage, we set the first parts of each matrix as train sets and the rest as validation sets. After doing the above steps in the training process, we can get the trained EMD-CAE model.

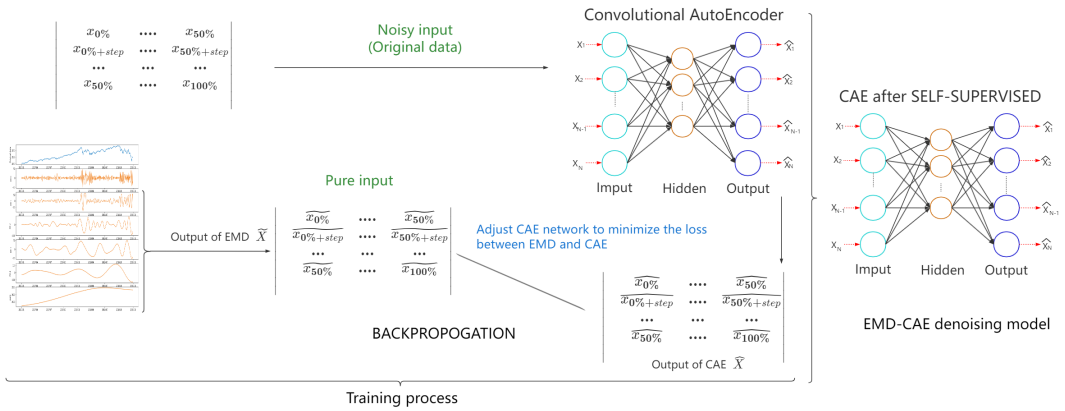


Figure 8: The framework of training process

For the CAE model part, we change some parameters in the input layer to accept different shapes of input matrices, which would be more flexible to use EMD-CAE model trained to denoise other data sets. This project's CAE network has four hidden layers. Two of the layers are encoding layers, while the remaining layers are decoding layers. Because the noisy data is one-dimensional, the encoding layers are Conv1D layers with the "relu" activation functions and convolution kernel of length three. Corresponding to these, the decoding layers are Conv1DTranspose layers with the same parameters. The output layer is a Conv1D layer with the "sigmoid" activation function. To make sure the same shape of input and output layers, padding function is used in each layer. A

summary of the constructed model is shown below. 4.1.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, None, 128)         512

 conv1d_1 (Conv1D)           (None, None, 32)          12320

 conv1d_transpose (Conv1DTra  (None, None, 32)         3104
 nspose)

 conv1d_transpose_1 (Conv1DT  (None, None, 128)        12416
 ranspose)

 conv1d_2 (Conv1D)           (None, None, 1)           385

=================================================================
Total params: 28,737
Trainable params: 28,737
Non-trainable params: 0
_________________________________________________________________
```

Figure 9: CAE model

This is reconstruction steps of the CAE model in this project.

1. **The input layer:** Each input from the noisy matrix is one-dimensional data, so we set the parameter of this layer (None, 1) to accept different data shapes.
2. **The first encoding layer:** It uses the convolutional layer to extract features from the input data.
3. **The second encoding layer:** It uses the convolutional layer to extract deeper features from the features of the first encoding layer.
4. **The first decoding layer:** It uses the transposition function of the convolutional layer to get new data from the features of the second encoding layer.
5. **The second decoding layer:** It uses the transposition function of the convolutional layer to get new data from the results of the first decoding layer.
6. **The output layer:** This layer will output the reconstructed data from the steps above.

This CAE neural network is trained by minimising Mean Squared Error (MSE) loss between the output matrix $\widehat{X}$ of CAE and the pure input matrix $\widetilde{X}$. What is more, to avoid information leakage and overfitting, the first 80% raws of matrix are divided into

train set, and the rest are validation set. When training the model by using S&P 500, its loss profile is plotted in the figure 4.1 below. This example shows that the model could fit well, without overfitting or underfitting.

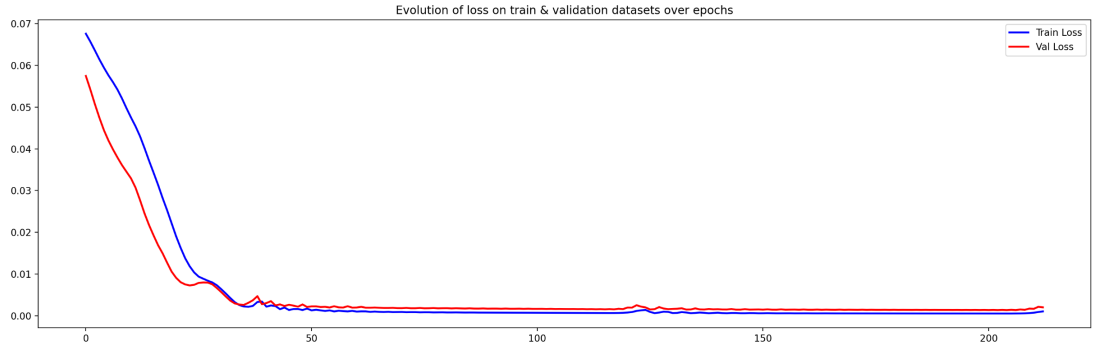


Figure 10: Loss plot of EMD-CAE model

This flowchart 4.1 summarises this section and highlight some key messages for better comprehension. The figure shows the details we mentioned before. We use the original data to generate the noisy and pure data matrices. We then divide the training test set according to temporal order. One of the new things is that we use the Minmaxscaler function. This is the formula 4.3 of the Minmaxscalter function. The specific role of Minmaxscaler is to generalise the statistical distributivity of the sample. It removes unit constraints from the data and transforms it into dimensionless, pure values that can be easily trained by neural networks.

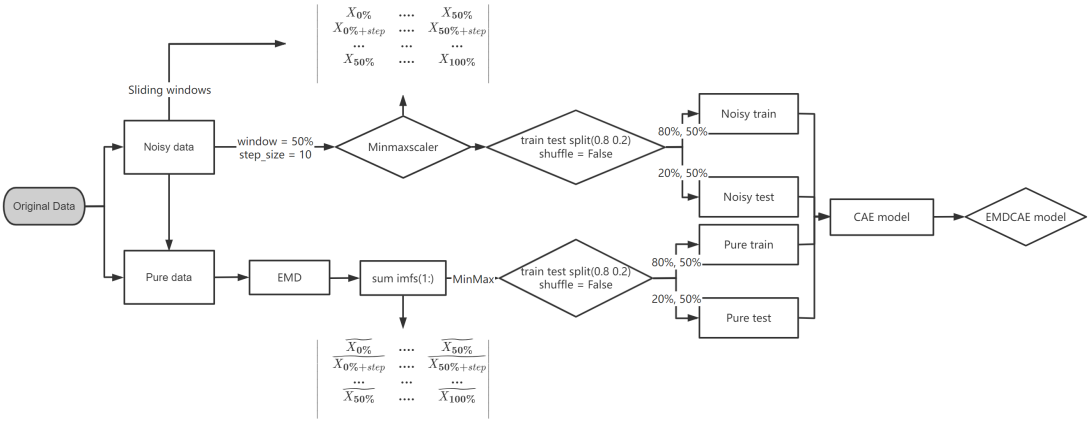


Figure 11: Flowchart of training process

$$x_{scaler} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{4.3}$$

4.2 Experiment setup

Once the EMD-CAE model has been trained, it can be employed to denoise noisy target data. The key reference paper [1] employed the same noisy data as the test set. However, this is a test within the sample. This project adds out-of-sample tests to demonstrate the generalizability of the model. We attempt to make the EMD-CAE model denoise the data it previously did not have access to. Then, the denoised data will be compared to the original data or the data from the MA-CAE model in the reference paper [1]. This project sets up three tests for evaluating the data from in-sample and out-of-sample tests for the purpose of making fair comparisons.

4.2.1 Regression test

Predicting the next day's price is typically much more complex problem than predicting the next day's trends. In the key reference paper [1], they tested denoised data from MA-CAE model in classification model. This project uses the Extreme Gradient Boosting (XGBoost) model to conduct regression forecasting tests in order to demonstrate the value of the denoised data further.

In contrast to the classification test, it may be difficult to predict the next day's price using only one-dimensional price data in the regression test. Therefore, if the XGBoost model had more features to learn, its performance could be enhanced. Before putting the denoised and original data into the model, it would be advantageous to generate features from data. By manually extracting physically and statistically significant features, it is possible to assist the model in discovering more intriguing underlying variables and generate a deeper understanding of the data.

This project begins by defining a generic feature extraction function whose output contains a total of 200 features. These features consist of 50 simple moving averages, exponential moving averages, and standard deviations with window sizes between 2 and 51. In addition, it contains 50 lag data. In summary, this project uses the 200 features extracted from the 51 data points in the denoised and original data to predict the next original data point.

The Mean Squared Error (MSE), the average squared difference between the target and predicted values, was selected as the objective function for this test in terms of data evaluation. In addition, to avoid serendipity of results, this project uses expanding

windows to calculate the MSE value each time and then obtains the average MSE value to compare the two data sets more effectively. For instance, in order to obtain two MSE values using expanding window, the data should be divided into three blocks in chronological order. Then, we utilise the first block to train the regression prediction model and calculate the MSE value of the second block, which is designated as the test set. For the second MSE value, the first two blocks are the train set, and the third block is the test set. Each time, one block would be set as the test set and the blocks before it are the train set.

According to this method, we can compare the MSE values of denoised data and original data at each expanding window period and see the average MSE as the overall performance of data. The smaller the MSE value is, the better the model prediction is.

4.2.2 Classification test

Trading time series data are continuously changing and do not simply fall into multiple categories. Therefore, if we wish to use the classification test, we must first apply a labelling method to the original data. In this project, we use a fixed-horizon labelling method to label log return values of original data. Unlike simple returns, log returns can make the data more smooth and better represent price movements in multiple periods [47]. We set the time series data X with n values: $t = 1, 2, ..., n$ and L 4.4 is the log return values of data X.

$$l_{t+1} = log\left(\frac{x_{t+1}}{x_t}\right) = log(x_{t+1}) - log(x_t) \tag{4.4}$$

Then, this project uses a simple fixed-horizon labelling method 4.5 to label the log return of original data, which is also chosen by the key reference paper [1]. If the log return is larger than the fixed threshold τ, it is labelled with "1" and means an upward price trend. If the log return is less than the minus threshold τ, it will get a "-1" label, which shows a downward price trend. In other cases, it is labelled as 0. In summary, this is a multi-label classification problem.

$$label_t = \begin{cases} 1 & \text{for} \quad l_t > \tau \\ 0 & \text{for} \quad |l_t| < \tau \\ -1 & \text{for} \quad l_t < -\tau \end{cases} \tag{4.5}$$

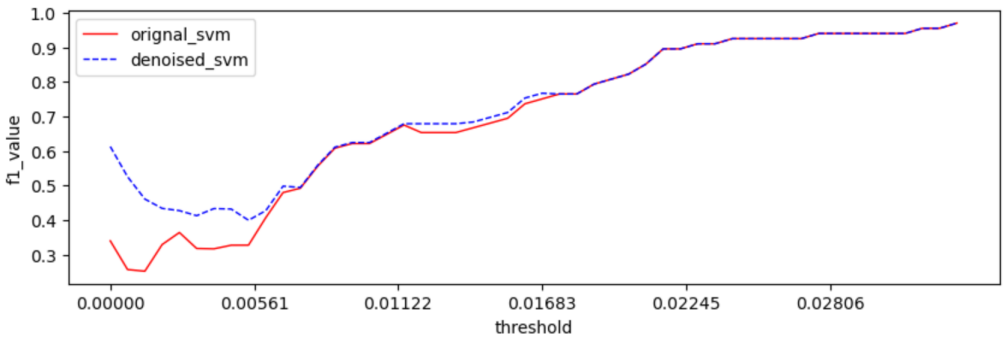
After obtaining the labels, Support Vector Machines (SVM) model is used to predict the next day's label by the previous day's log return.

For the classification forecasting test based on SVM, the log return of data is set as features, and the labels of next data point are set as the target. Moreover, this project also explores what will happen if the length of features is expanded.

We train two models to compare the denoised data and original data. The first model predicts the label of original data using the log return of denoised data from the EMD-CAE model, while the second model uses the original data's log return to predict the original data's label. Then, the F1 score of the two models can be calculated and compared based on different thresholds. The F1 score is used to evaluate the classification ability of the model comprehensively. This is the formula for F1 score, which balances the value of precision (percentage of all values predicted to be true that are actually true) and recall (percentage of all values actually true that are predicted to be true):

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{4.6}$$

In addition, in order to prevent the SVM model from reporting errors, a function implementing dynamic thresholds is added to the algorithm. We adapt the model to different data sets to prevent thresholds from being too high and labelling all data as 0. First, we determine the minimum range of log return values for the data. Then, we divide it into segments of uniform size and calculate F1 for each starting point of the segments. For a given threshold, we first label the log return of the original data according to the threshold. Then, we use the denoised and original data to train two SVM models to predict the label of original data and calculate the F1 score of their test sets. In this way, we can visually compare multiple F1 values of both data sets. The horizontal axis represents thresholds, and the vertical axis represents F1 values. As shown in the following figure 4.2.2. The higher the F1 value, the better. In this figure, we can find that the denoised data can get higher F1 scores using different thresholds.



4.2.3 Trading strategy test

Using statistical methods, investors can identify market patterns by analysing historical prices and trading data with trading strategies. They employ trading strategies to forecast future price trends and to compute buying and selling signal points. The principles of these trading strategies are relatively simple and uncomplicated, but they are effective decision-making aids for traders [48].

Moving Average Convergence Divergence (MACD) is a well-known and popular strategy and it is used in this project to compare the denoised and original data. The following steps 4.2.3 are how this strategy works. Based on the MACD theory, researchers have added numerous enhancements. This project uses the most basic MACD indicator to generate buying and selling signals.

1. Calculate the price's exponential moving average $Short, Long$ with two different lookback window sizes. The short window size is often 12, and the long window size is often 26.
2. Get new indicator DIF: $DIF = Short - Long$
3. Calculate DEA, which is DIF's exponential moving average. The window size is often 9.
4. Get a buying signal when DIF breaks upwards through DEA
5. Get a selling signal when DIF breaks downwards through DEA

MACD strategy was proposed in 1970 by Gerald Appel [49]. Although the algorithm is a bit old, it is still a good reference in the modern market. This method calculates trends using moving averages, making it a lagging indicator. However, it can work for trading to some extent because some personal traders may use trading strategies which are also based on moving averages. As a result of this, price trends have the potential to continue.

This figure 4.2.3 shows the steps of MCAD signals using the data of S&P 500 from 2017 to 2019.

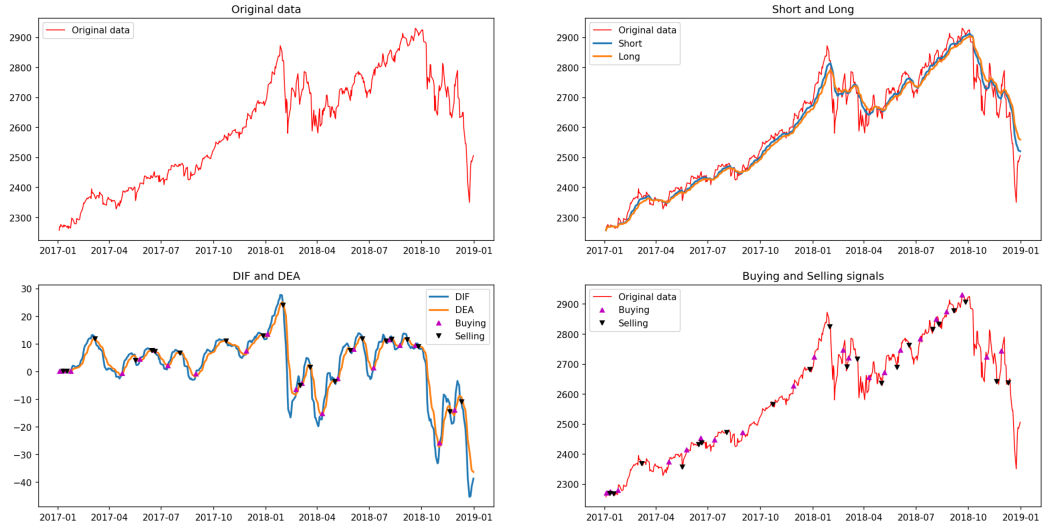


Figure 12: MACD steps

After the trading strategies is defined, it is easy to use the same way to get signals from denoised data. But, it is still a challenge to compare the signals from two data directly. The plot 13 looks like this:

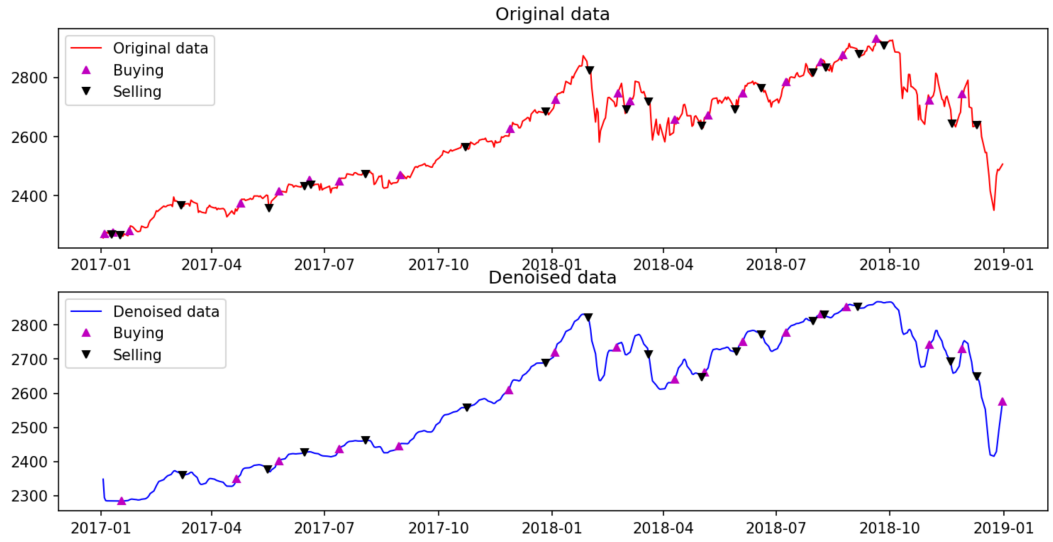


Figure 13: Comparison of signals

The signals from denoised data are similar to those from original data. To find a clear way to compare, this project uses a backtesting strategy to calculate simple returns by using these signals. Backtesting is the method of verifying the viability and

effectiveness of trading strategies based on historical data. Backtesting is done in the hope that the backtesting performance can be used to evaluate future performance [50].

This is the pseudo-code 4.2.3 of the backtesting strategy used in this project. Assuming we begin with 1e8 cash, we will use all cash on hand to purchase stocks upon receiving a buying signal. If we receive a selling signal, we will sell all of our stocks and get their current market value. Subtract the initial amount from the total assets and divide it by the initial amount to calculate the simple rate of return. In this way, we can summarise multiple signals into a simple return value and thus make a direct comparison. For the denoised data, we first use the MACD strategy to calculate the signals. Then, we can put the signals and original data into backtesting to get the simple return value. For the original data, we do the same thing. So, we will have two values of the denoised and original data and can compare them. The higher the simple return, the better the data.

Input :
Adj_Close: Time series data, with are the adj close prices of the share
Signal: Time series data, with are values of 1, 0, -1
Asset: Float, the default is 1e8
Output:
Simple_return

Function `Backtesting`(*Adj_Close, Signal, Asset*)**:**
- *Cash* ←Asset
- *share_number* ← 0
- **for** *i in len(Adj_Close):* **do**
  - **if** $Signal_i$==1*:* **then**
    - *share_number* + ← *cash*//(100×Adj_Close_i) ×100
    - **if** *(cash*//(100 ×Adj_Close_i)) >0*:* **then**
      - cash - ← [*cash*//(100×Adj_Close_i))]×100×Adj_Close_i)
    - **end**
  - **end**
  - **if** $Signal_i$==−1*:* **then**
    - *cash* + ←share_number×Adj_Close_i
    - *share_number* ← 0
  - **end**
- **end**
- *Asset_new*=*cash*+*share_number* ×Adj_Close_i
- *Simple_return*= (*Asset_new*-*Asset*) ÷Asset
- **return** *Simple_return*

Figure 14: Backtesting

4.2.4 Summary of testing process

This project compares the denoised and original data from two testing-related perspectives.

The first perspective includes data range. We employ in-sample and out-of-sample tests to test whether the EMD-CAE model can denoise data not seen during training.

The second perspective consists of three tests: regression forecasting tests, classification forecasting tests, and trading strategy tests. Mean Squared Error (MSE) is the criterion for evaluating the regression model; the smaller the MSE, the better. F1 score is the key value for the classification model; the closer it is to 1, the better. The simple return ratio is the judgmental measurement for testing trading strategies, and the greater it is, the better.

4.3 Improvement based on the key reference

Although this paper is based on ideas from the key reference [1], a number of improvements have been made in training process and testing process. The results in the following chapter also reflect that the EMD-CAE model in this paper has been greatly improved relative to MA-CAE model. This section will highlight the difference between the design of models.

Training process:

- The noisy and pure input matrix of EMD-CAE model is quite different from the MA-CAE model. In the project, we use sliding windows to make every single input in the noisy input matrix different.
- About the pure input, we use EMD to replace the fixed range of moving averages for the pure input matrix. EMD would be more adaptive to different data sets.
- In terms of splitting the train and test set, we used the first 80% as the train set to avoid information leakage.
- To avoid overfitting, a callback function is added to the model, which helps the EMD-CAE model stop iteration at the appropriate times. In the reference paper [1], they used a fixed number which would not be suitable for different data. In this project, we set a vast number first and let the model stop if there is no improvement in the last ten iterations.
- As a result of the above, the shapes of noisy and pure input matrices are according to the lengths of original data. When the original data is large, this model will use more input to train the EMD-CAE model.

These changes make the model automatically adjust the time cost and iteration number according to original data itself. What is more, the input layer of EMD-CAE

model is flexible, which makes the denoising model could be used to pre-process different data easily. These changes make the EMD-CAE model potentially more time-consuming during the training period. However, in subsequent tests, we find that the trained EMD-CAE and MA-CAE models take equal time to denoise the same noisy data.

Testing process:

Firstly, this project has added out-of-sample tests in terms of test data. The reference paper [1] used the same data as test data, which was used to train the MA-CAE model. The out-of-sample test may help to prove the denoising model better.

Then, the reference paper [1] compared the data by using classification and trading strategy tests. In this project, we add the regression tests to evaluate the performance of predicting accurate data values. In order to better reflect the outcome of the test, this paper also test data using the classification and trading strategy tests.

For the classification testing, we change the target. In the reference paper [1], they trained two models by using two data. One of them used the denoised data to predict the labels of denoised data. Another model used the original data to predict the labels of original data. This project changes the first model to try to make it more meaningful in practice. Because the denoised data is smoother than the original data, it is easier to get a high score on the trend prediction. In addition, this score may be hard to prove the advantage of the denoised data. Supposing the denoised data is a line in the extreme case, its F1 score will always be one. So, in this project, we change the first classification model and use the denoised data to predict the labels of the original data. In this way, we want to test data in classification tasks from another perspective.

For trading strategy testing, the reference paper [1] presented buying signals, which is better than those of the original data. They did not show the selling signals. And, it is difficult to compare the signals from two data directly. To solve this inconvenience, this project uses simple return ratio to reflect the quality of signals.

4.4 Data

In this project, we test the EMD-CAE model on eleven data sets in total. Nine small daily close price data sets (4.4) are from Yahoo finance. For the in-sample tests, we use data from 2017 to 2019 to train the model and use the trained model to denoise the same original data; for out-of-sample tests, we use data from 2017 to 2019 to train the model and use the trained model to denoise data in the period 2019-2020.

Small data sets (daily price)			
Index	Stock	Future	FX
∧FCHI (CAC 40)	LLOY.L	CL=F (Crude Oil)	EURHKD=X
∧GSPC (S&P 500)	TSCO.L	GC=F (Gold Aug)	
∧N225 (Nikkei 225)	UNH		

Table 1: Small data sets (daily price)

In addition, the project also uses bitcoin minute data as a big high-frequency data set and 20 minutes traffic volume of road congestion data. Both of them are from the Kaggle website.

4.5 Threat to validity

During the design process of EMD-CAE model and subsequent testing, much effort was made in this project to avoid validity threats. The next two parts will highlight the work for internal and external threats.

4.5.1 Internal

- All the data sets used in this project are one-dimensional data without any other features. This is not only to better focus on denoising one-dimensional trading data but also to avoid new features that bring in additional unpredictable noise.
- In the training process of EMD-CAE model, we use sliding windows to avoid information leakage and help the model access different input data.
- In the testing process, we use various methods to make comparisons as fair and comprehensive as possible, like expanding window, different thresholds and back-testing.

4.5.2 External

- We use out-of-sample tests to ensure the EMD-CAE model's validity.
- We use daily data sets from four different financial markets to avoid coincidence in the results and verify the generalizability of the model.
- We also test the model by using large high-frequency bitcoin data and traffic data, which is quite different from financial data.

5 Results and Comparisons

This Chapter evaluates the performance of the EMD-CAE model's denoised output. The benchmarks and tests are outlined in the preceding Chapter. All code is executed using Python. In addition, we replicate the MA-CAE model from the key reference paper [1]. The main comparison aspects are divided into three parts: results of daily data, further proof and results of data with different frequencies or types. The first two sections rely primarily on nine small data sets. The final one discusses the outcomes of data collected at various frequencies. The blue line in classification tests represents the denoised data, while the red line represents the original data. In regression tests, the colour green indicates that the denoised data are superior to the original data.

5.1 Results of daily data

We test the denoised data in in-sample and out-of-sample tests for the data sets in table 4.4. Here, the benchmarks are the results of MA-CAE model's denoised data and the original data. In some tests, due to the multiple data sets, we only display the results for one of them in detail. More plots or tables of the rest data results can be found in appendix A. The figure 15 below shows the denoised result of CAC 40 data in the in-sample test. The second figure 16 shows the result of same data in the out-of-sample test. The figures of the rest small data sets are in the appendix A.1.

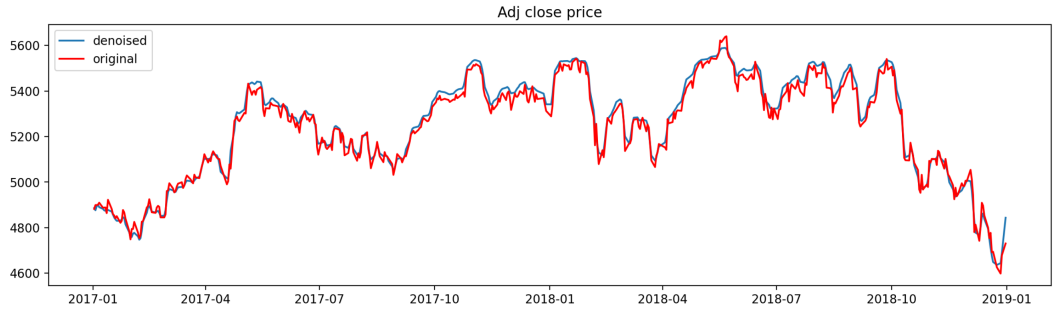


Figure 15: CAC 40 (2017-2019)

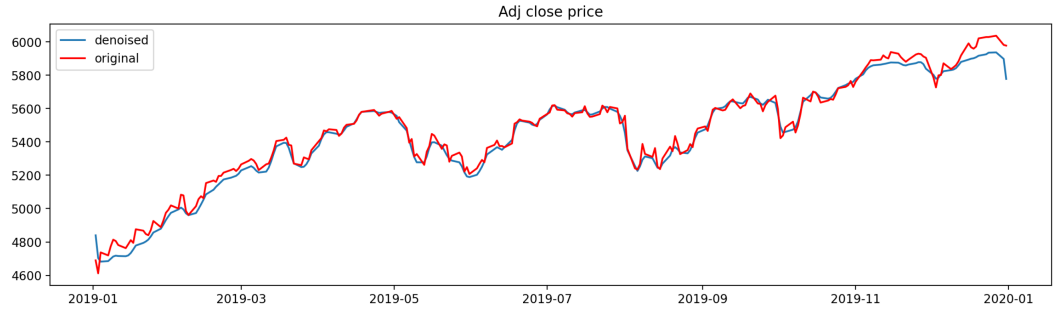


Figure 16: CAC 40 (2019-2020)

In both in-sample and out-of-sample tests, the overall look of the denoised data is smoother and consistent with the original data, as evidenced by the figures. For the remaining data sets, the outcomes are similar to these plots.

5.1.1 Regression test

In the regression test, ten expanding windows are calculated. It means that the data has been divided into 11 parts. Beginning with the second section, each subsequent section will serve as a test set. This table 5.1.1 shows the regression results of CAC 40 in the in-sample test. In most instances, the EMD-CAE denoised data can perform better than the original data. And its average MSE is indeed less than that of the original data. Similar conclusions from the other data sets are included in the appendix A.2.1.

IN-SAMPLE			
	MSE values		
Expanding window	EMD-CAE	MA-CAE	Original
1	13146	15085	7614
2	3256	4706	3289
3	16253	25335	21951
4	2494	11773	4090
5	6309	18967	5584
6	1523	9856	2180
7	10217	46775	9400
8	1492	4344	2210
9	5471	18955	9132
10	32157	60152	36211
Average	9232	21595	10166

Table 2: CAC 40 regression tests (in-sample)

This table 5.1.1 shows the regression results of CAC 40 in the out-of-sample test. In the majority of instances, the denoised data of the EMD-CAE model can perform better than the original data, such as in the in-sample test. The other data sets have the similar conclusions, which are in the appendix A.2.2.

OUT-OF-SAMPLE			
	MSE values		
Expanding window	EMD-CAE	MA-CAE	Original
1	4160	12470	8122
2	7442	23154	6972
3	3036	6070	7930
4	1046	20548	6949
5	6591	5137	5988
6	2133	6513	2679
7	8413	7746	7916
8	24235	137992	58378
9	5786	5856	16139
10	17903	10219	20730
Average	8075	23570	14180

Table 3: CAC 40 regression tests (out-of-sample)

In this project, the average MSE is used as a representative value. Therefore, we can compare the results of the nine data sets by combining them in a single table. This is the table 5.1.1 of in-sample test results. The second table contains results from out-of-sample tests.

IN-SAMPLE			
	Average MSE values		
Name	EMD-CAE	MA-CAE	Original
^FCHI	9232.34	21595.26	10166.81
^GSPC	2730.21	7997.58	3393.83
^N225	450397.95	791182.90	475506.00
CL=F	9.93	31.73	8.11
EURHKD=X	0.016525	0.031823	0.019450
GC=F	333.67	1059.07	441.23
LLOY.L	1.35	2.97	1.76
TSCO.L	113.82	224.21	128.84
UNH	90.32	129.60	108.33

Table 4: Average MSE of regression tests (in-sample)

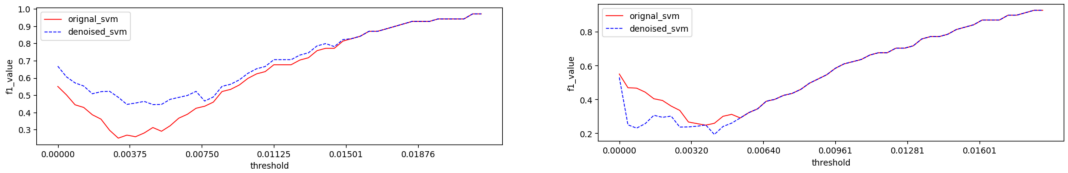
OUT-OF-SAMPLE			
	Average MSE values		
Name	EMD-CAE	MA-CAE	Original
∧FCHI	8075.08	23570.96	14180.72
∧GSPC	3532.08	7191.96	3751.05
∧N225	303752.98	828491.32	314332.89
CL=F	6.14	24.23	8.24
EURHKD=X	0.002556	0.004316	0.003917
GC=F	1477.06	1923.74	1577.80
LLOY.L	7.67	10.42	6.98
TSCO.L	45.83	155.69	50.22
UNH	154.16	203.53	154.68

Table 5: Average MSE of regression tests (out-of-sample)

The tables above show that the denoised data from the EMD-CAE model can perform better in most regression tests. No matter in in-sample or out-of-sample tests, it has similar results. In out-of-sample tests, the improvements can be up to 43%, and the average optimization is 13%. In addition, the results of the MA-CAE model are worse than the original data in regression tests.

5.1.2 Classification test

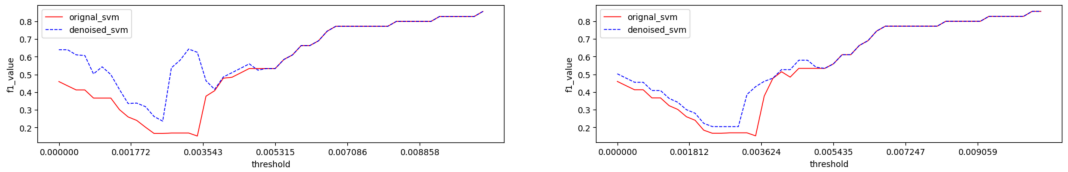
In this section, we also take the results of the F1 score of CAC 40 as an example. Because of the large number of plots, the classification results of the rest data sets are in the appendix A.3. This figure 5.1.2 shows the F1 scores in in-sample tests. Since we test the SVM model using different thresholds, we use two lines to present the values of the F1 score. The blue line is the results of denoised data, and the red line is those of original data. The higher the lines are, the better they are. Here we use the one log return value to predict the next label.



(a) EMD-CAE F1 score (b) MA-CAE F1 score

Figure 17: Classification testing (in-sample)

This figure 5.1.2 shows the results in out-of-sample tests.



(a) EMD-CAE F1 score (b) MA-CAE F1 score

Figure 18: Classification testing (out-of-sample)

The denoised data from the EMD-CAE model performs better and is more stable in both in-sample and out-of-sample tests, as evidenced by these four plots. The MA-CAE model is unstable across a variety of tests and data sets. Moreover, after adjusting the classification testing, it does not perform as well as in the reference paper [1]. In that paper, they used denoised data to predict denoised labels. That test may not work for extreme conditions, like a horizontal line.

5.1.3 Trading strategy test

As mentioned previously, we evaluate the data using simple return ratios in this section. In addition, data is also separated into in-sample and out-of-sample tests. These are the results 5.1.3 of simple return in in-sample tests.

IN-SAMPLE			
	Simple return		
Name	EMD-CAE	MA-CAE	Original
^FCHI	-0.013365	-0.165385	-0.130362
^GSPC	0.162284	-0.054440	0.042208
^N225	0.136037	-0.085391	-0.006947
CL=F	0.271139	-0.230117	0.005411
EURHKD=X	0.112123	0.030686	0.026877
GC=F	0.127385	0.096788	0.074475
LLOY.L	0.131230	-0.131063	-0.105409
TSCO.L	-0.070792	-0.232139	-0.241829
UNH	0.348277	-0.144594	-0.053926

Table 6: Simple return of backtesting (in-sample)

To facilitate a more visual comparison, we transform the table into a bar graph 5.1.3. The blue ones are the results of the EMD-CAE model, the orange ones are the results of the MA-CAE model, and the grey ones are the original data.

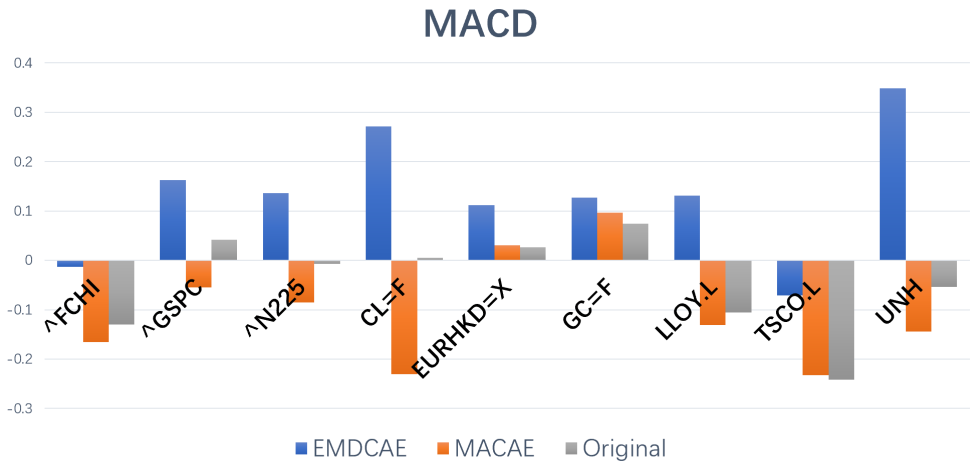


Figure 19: Simple return bars of backtesting (in-sample)

The next is the table 5.1.3 of results in out-of-sample tests and its bar chart 5.1.3.

OUT-OF-SAMPLE			
	Simple return		
Name	EMD-CAE	MA-CAE	Original
∧FCHI	0.267668	0.243742	0.216864
∧GSPC	0.270477	0.174133	0.195806
∧N225	0.142621	0.078949	0.101018
CL=F	0.339321	0.048942	0.082753
EURHKD=X	-0.038397	-0.081821	-0.064682
GC=F	0.138594	0.082184	0.079242
LLOY.L	0.104116	-0.030130	0.006078
TSCO.L	0.482657	-0.072041	0.172853
UNH	0.313525	0.169662	0.134987

Table 7: Simple return of backtesting (out-of-sample)

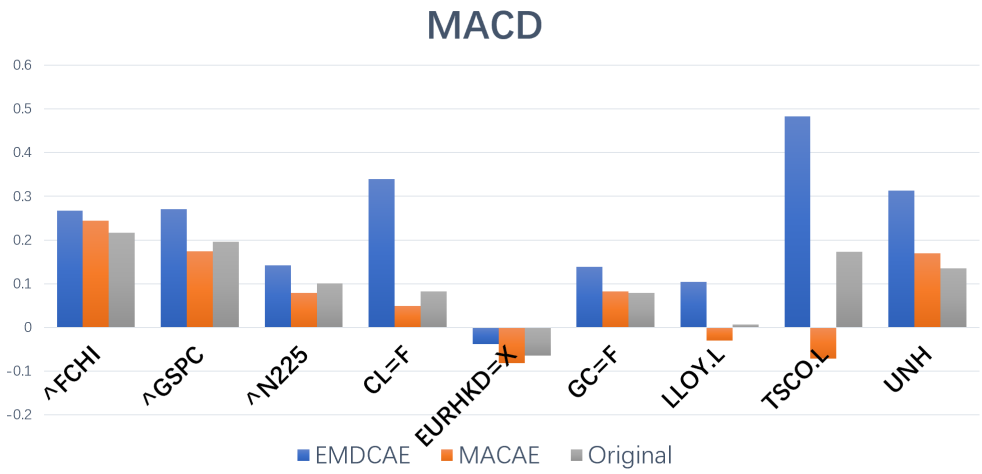


Figure 20: Simple return bars of backtesting (out-of-sample)

The tables and bar charts show that the EMD-CAE model consistently performs better in trading strategy tests, while the MA-CAE model exhibits inconsistent behaviour across data sets. The denoised data from EMD-CAE can have a significant impact on earnings. Denoised data has a simple return improvement of up to 40 % in tests, from -0.05 to 0.348.

5.2 Further proof

According to the section above, denoised data can perform well in most tests. Several experiments have been designed for this project to explore the advantages of the data further, as it may have more to provide.

We utilise the denoised data from the previous in-sample tests in the subsequent tests. It means the denoised data is from 2017 to 2019. Then, we use the denoised data and the original data to train the two models, respectively. Finally, two models are fed the same noisy original data from 2019 to 2020. If noisy data can perform better in the model trained with denoised data, this provides additional evidence of the validity of denoised data.

5.2.1 Regression test

This is the flowchart 5.2.1 for this test. We compare the results of the two models by using the same input.

Figure 21: Flowchart of regression tests (further proof)

The table 5.2.1 shows that the model trained by denoised data has a significant dominance with the input of noisy data.

Average MSE		
Name	EMD-CAE	Original
∧FCHI	35613	37487
∧GSPC	24055	30960
∧N225	87621	93885
CL=F	4.4878	5.2648
EURHKD=X	0.006787	0.010405
GC=F	15612	16138
LLOY.L	3.1452	3.1655
TSCO.L	22	19
UNH	216	245

Table 8: Average MSE in further proof

5.2.2 Classification test

This is the flowchart 5.2.2 in this test.

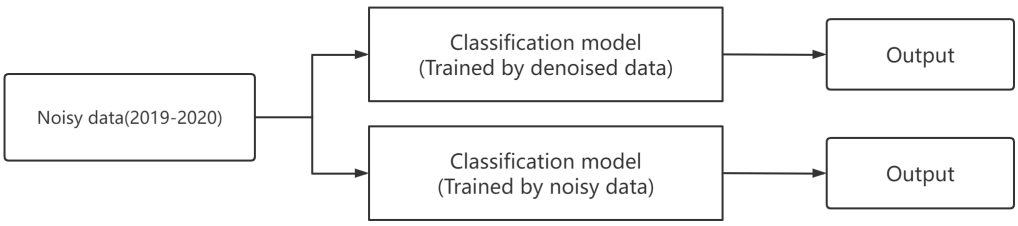


Figure 22: Flowchart of classification test (further proof)

This figure 5.2.2 is the F1 score of CAC 40, which shows the denoised data get a higher F1 score after a certain threshold. What is more, the line of denoised data grows faster than the other and reaches the extreme horizontal line in advance. This phenomenon indicates that many points in the noisy data may have been incorrectly classified as “1” and “-1” by the model trained with noisy data. However, the model trained by denoised data can ignore noise interference and divide them to “0” correctly. The rest results of other data sets have similar shapes 5.2.2.

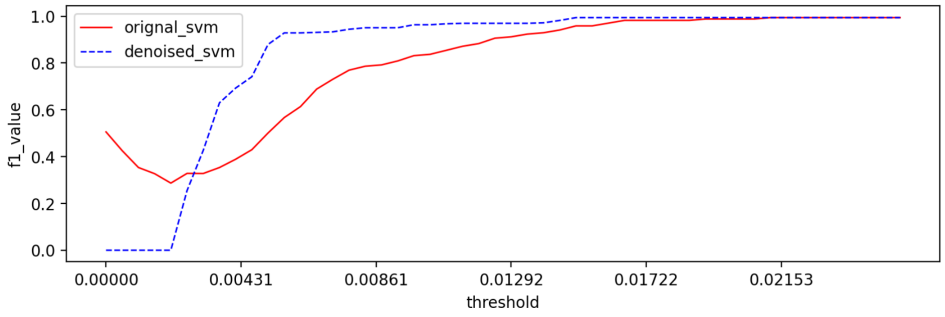
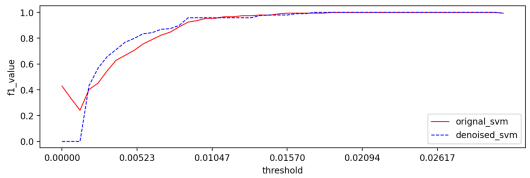
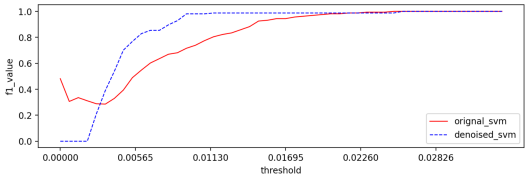


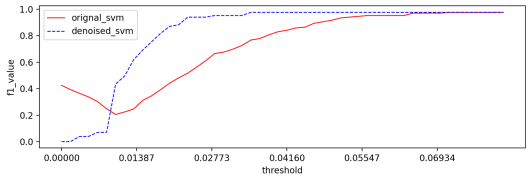
Figure 23: F1 plot of CAC 40 (further proof)



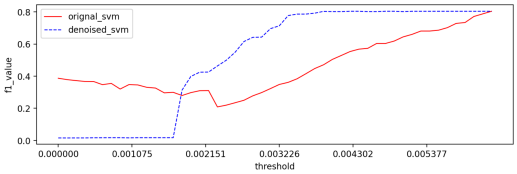
(a) ∧GSPC F1 score



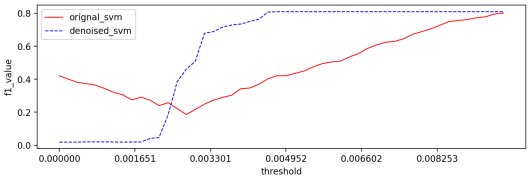
(b) ∧N225 F1 score



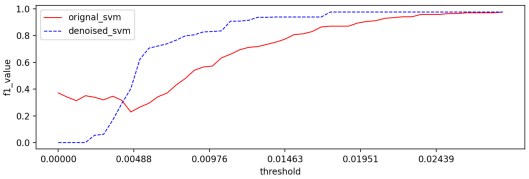
(c) CL=F F1 score



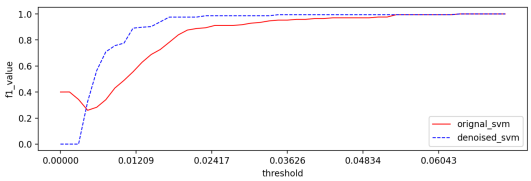
(d) EURHKD=X F1 score



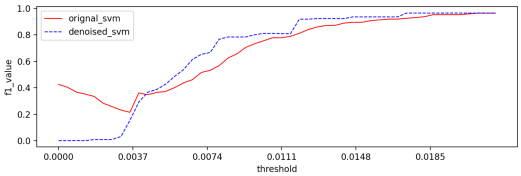
(e) GC=F F1 score



(f) LLOY.L F1 score



(g) TSCO.L F1 score



(h) UNH F1 score

Figure 24: F1 plots (further proof)

5.3 Results of data with different frequencies or types

In the sections above, we only use daily close price data to compare the EDM-CAE model to the MA-CAE model and the original. It obtains positive results for the EMD-CAE model. In this section, our model will be tested using data of various frequencies and types. In addition, we include only CAE or EMD comparisons. Here, only CAE means that the pure input equals the noisy input, and only EMD means that one-dimensional data is only denoised by using EMD. We select the S&P 500 to represent daily data, bitcoin to represent one-minute data, and traffic data to represent twenty-minute data in a different field.

5.3.1 Daily price data

We also divide tests into in-sample and out-of-sample parts. For in-sample testing, we train the model using data from 2017 to 2019 and then use the trained model to denoise data from 2019-2020. This table 5.3.1 shows the MSE values of each model and original data.

IN-SAMPLE

MSE values (S&P 500)					
Expanding window	EMD-CAE	MA-CAE	CAE	EMD	Original
1	1280	1614	1377	1721	2153
2	2281	1602	2159	1762	1824
3	6916	9763	8727	7077	9295
4	6075	5364	5718	5916	5425
5	3198	4500	3570	3498	3484
6	1215	3042	1573	1230	1631
7	256	1908	366	273	210
8	2088	33250	3514	2620	2939
9	1494	10570	1407	1571	2227
10	2496	8357	2936	2313	4746
Average	2730	7997	3135	2798	3393

Table 9: MSE values of S&P 500 in in-sample tests

This table 5.3.1 is about the results of S&P 500 in out-of-sample tests.

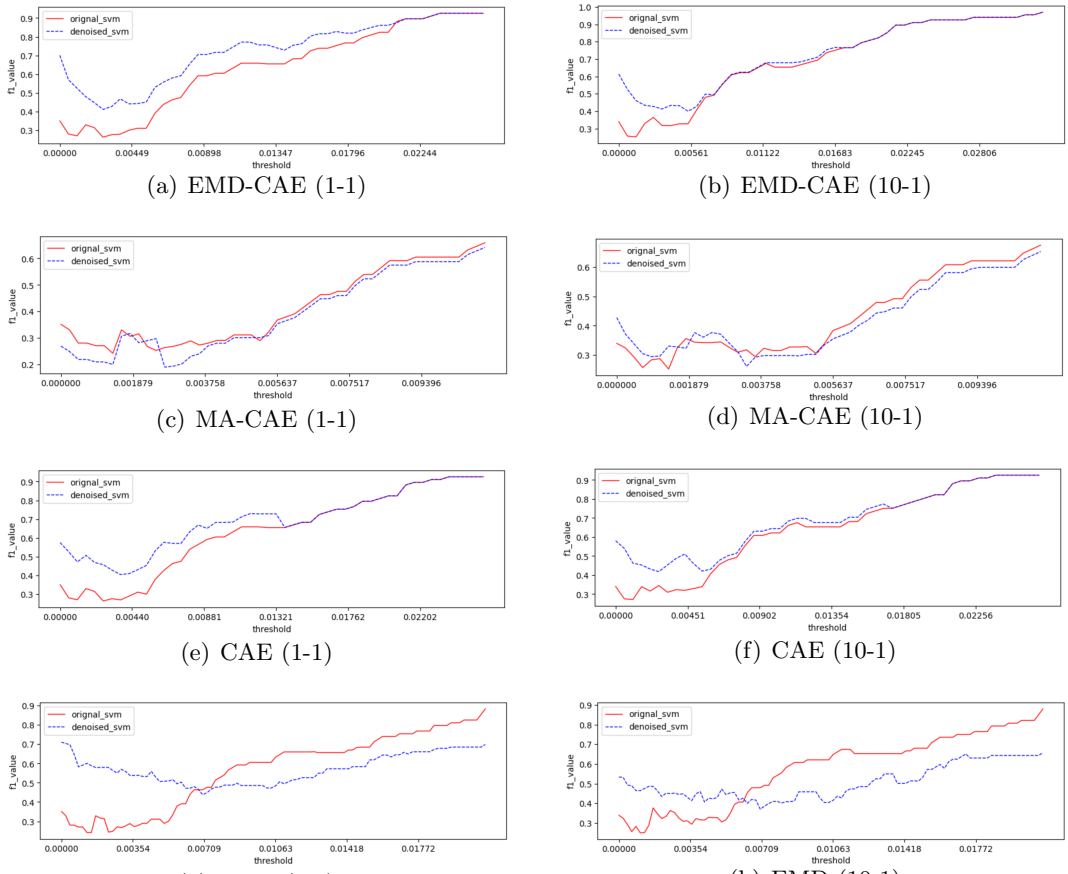
OUT-OF-SAMPLE

MSE values (S&P 500)					
Expanding window	EMD-CAE	MA-CAE	CAE	EMD	Original
1	944	625	588	764	708
2	5786	9790	8385	7373	7829
3	2010	1636	6926	1236	4808
4	5159	21782	6146	6913	5092
5	2919	2241	1987	2276	2264
6	1572	8221	914	1093	1188
7	731	2722	1590	588	851
8	4678	9667	4043	2838	5024
9	3015	4244	2007	1644	1571
10	8501	10987	3831	5327	8170
Average	3532	7191	3642	3005	3751

Table 10: MSE values of S&P 500 in out-of-sample tests

Both the EMD-CAE model and the EMD model perform admirably in the two tests. Although CAE's performance is acceptable, it is inferior to that of the other two. Similar to previous tests, the MA-CAE model may not be suitable for regression forecasting.

Next, we will examine their classification predictions' outcomes. We will test each model in two ways, using one previous data to predict the following label and using ten. This is the outcome 5.3.1 of in-sample comparisons between four models and the original data. Another figure 5.3.1 is the results of out-of-sample tests. In contrast to the regression prediction, the EMD model performs very poorly, whereas the EMD-CAE model is still reliable. Even the MA-CAE model can also outperform EMD. Therefore, the EMD-CAE model, which combines the benefits of CAE and EMD models, is the best option for denoising daily data.



(a) EMD-CAE (1-1)

(b) EMD-CAE (10-1)

(c) MA-CAE (1-1)

(d) MA-CAE (10-1)

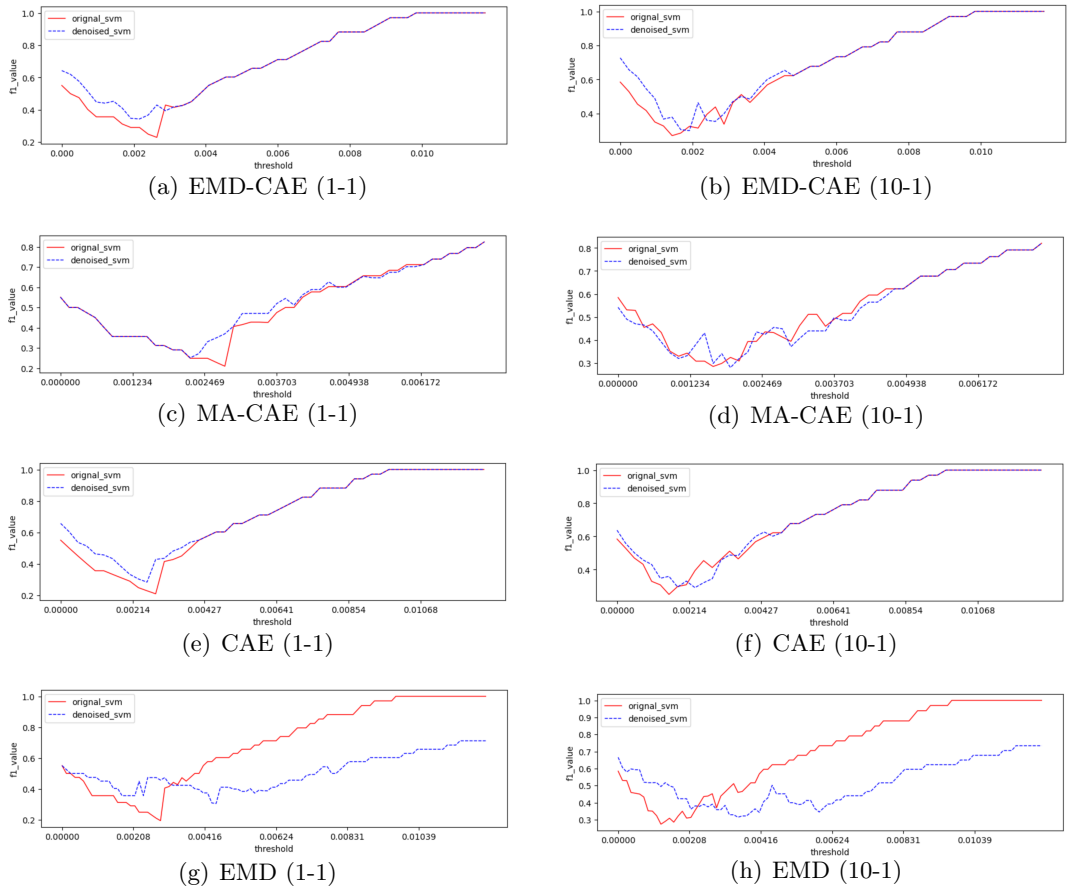
(e) CAE (1-1)

(f) CAE (10-1)

(g) EMD (1-1)

(h) EMD (10-1)

Figure 25: F1 plots of four models (in-sample)



(a) EMD-CAE (1-1)

(b) EMD-CAE (10-1)

(c) MA-CAE (1-1)

(d) MA-CAE (10-1)

(e) CAE (1-1)

(f) CAE (10-1)

(g) EMD (1-1)

(h) EMD (10-1)

Figure 26: F1 plots of four models (out-of-sample)

5.3.2 1-minute bitcoin data

In this section, bitcoin data from the Kaggle website is utilised. The data range is from 01.01.2021 to 05.12.2021. The testing procedure is similar to that of the S&P 500. In in-sample testing, the entire data set is used for training and as the test set. In out-of-sample, we designate the initial 80 percent as train data and the entirety as test data. These two tables 5.3.2 5.3.2 are the results of in-sample and out-of-sample tests.

IN-SAMPLE

MSE values (Bitcoin)					
Expanding window	EMD-CAE	MA-CAE	CAE	EMD	Original
1	4970	7929	3835	5036	6959
2	3104	3692	2279	3392	4641
3	26702459	31299286	26693579	27773179	28271068
4	6144499	6248411	6016359	6036827	6095684
5	10375	14345	8645	10274	10902
6	191519	177842	171177	196104	198253
7	3298	3925	2181	3081	3662
8	695664	661756	626055	679774	717363
9	6567	11355	4894	6659	8267
10	3798	4349	4338	3176	4237
Average	3376625	3843289	3353334	3471750	3532104

Table 11: MSE values of Bitcoin in in-sample tests

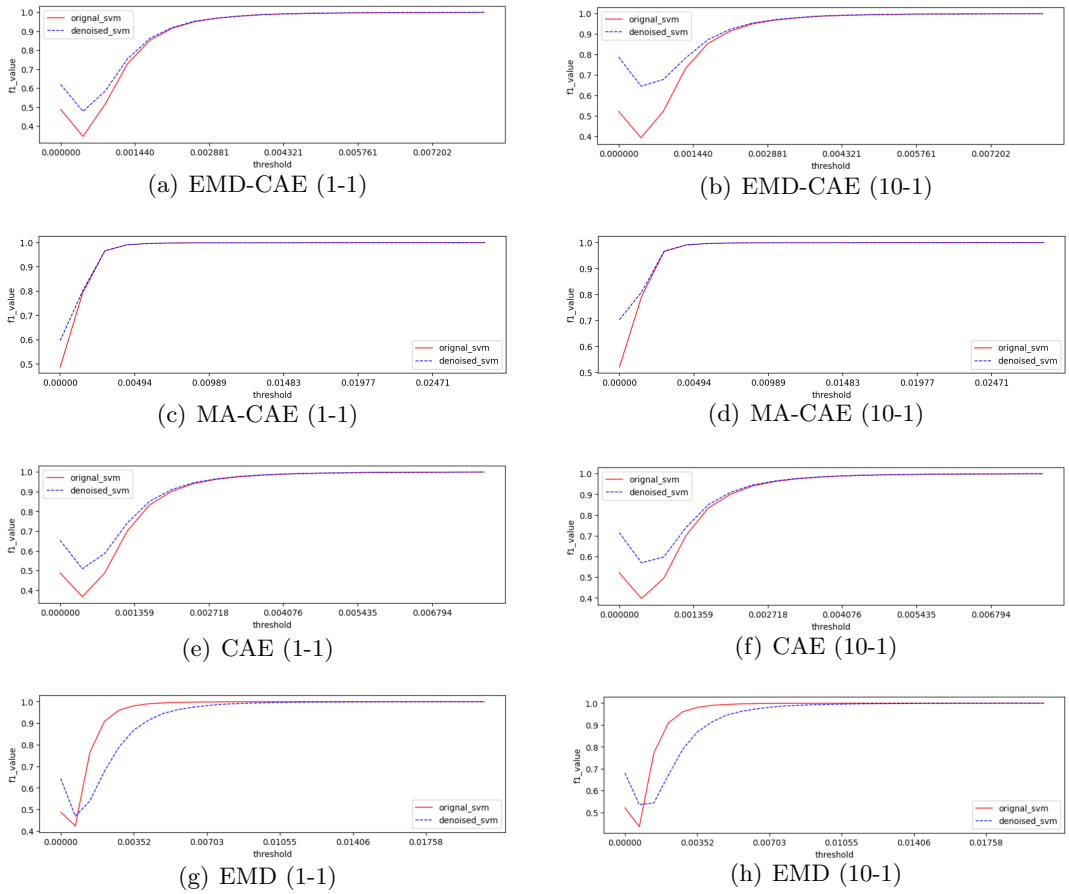
OUT-OF-SAMPLE

MSE values (Bitcoin)					
Expanding window	EMD-CAE	MA-CAE	CAE	EMD	Original
1	4616	7628	5067	5036	6959
2	3104	4974	3181	3392	4641
3	26977903	28594959	28008864	27773179	28271068
4	6303679	6266115	6458390	6036827	6095684
5	10198	14095	10435	10274	10902
6	196822	172597	183405	196104	198253
7	2552	6540	3032	3081	3662
8	670754	912124	658877	679774	717363
9	5562	12522	6973	6659	8267
10	5987	8817	4572	3176	4237
Average	3418117	3600037	3534280	3471750	3532104

Table 12: MSE values of Bitcoin in out-of-sample tests

In regression tests, the EMD-CAE model continues to produce the best results. And the model performs better with high-frequency data than with daily data, presumably

because high-frequency data contain more noise. Next 5.3.2 5.3.2 is the performance of the four models in the classification tests. Here, the plots of the MA-CAE model are different because of adaptive thresholds. As can be seen from the figures, all three models perform well on the classification task on high-frequency data, except for EMD. Regardless of different thresholds, the denoising data from these three models outperformed the original data.



(a) EMD-CAE (1-1)

(b) EMD-CAE (10-1)

(c) MA-CAE (1-1)

(d) MA-CAE (10-1)

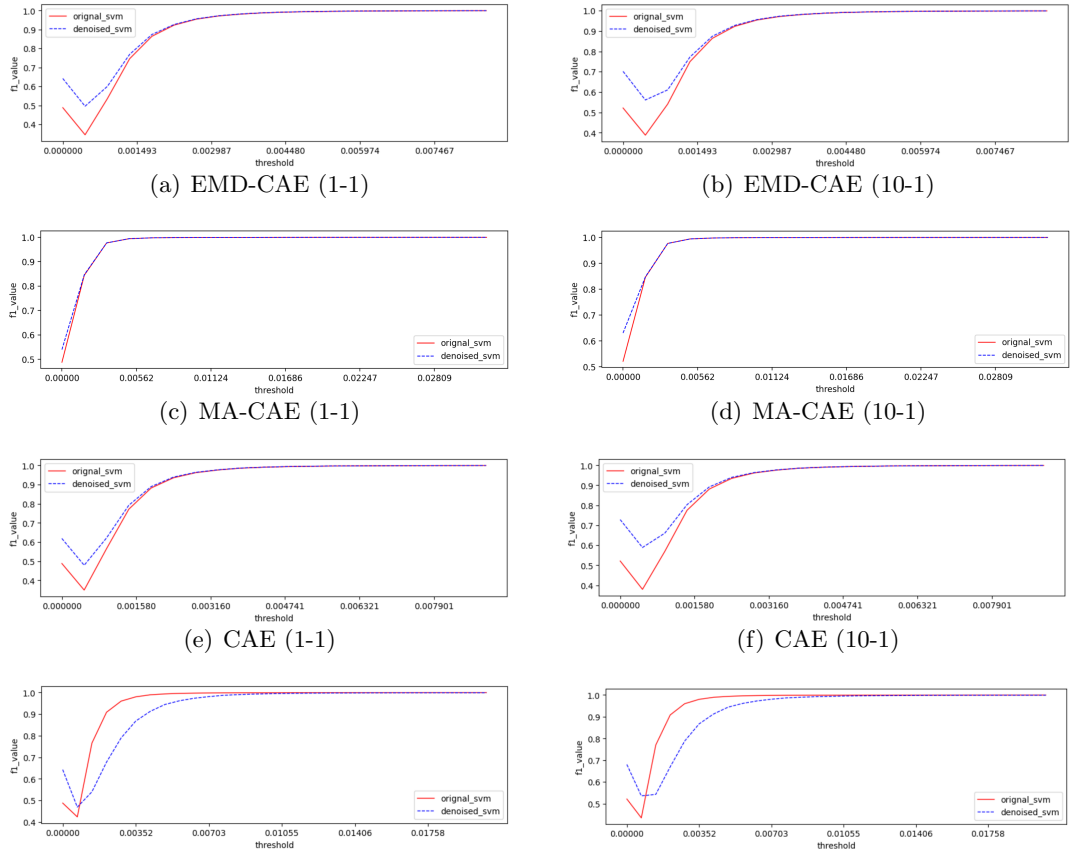
(e) CAE (1-1)

(f) CAE (10-1)

(g) EMD (1-1)

(h) EMD (10-1)

Figure 27: F1 plots of four models (in-sample)



(a) EMD-CAE (1-1)

(b) EMD-CAE (10-1)

(c) MA-CAE (1-1)

(d) MA-CAE (10-1)

(e) CAE (1-1)

(f) CAE (10-1)

(g) EMD (1-1)

(h) EMD (10-1)

Figure 28: F1 plots of four models (out-of-sample)

5.3.3 20-minute traffic data

This dataset does not contain financial trading data; instead, it describes traffic volume every 20 minutes. It is also time series data, so we test our EMD-CAE model with it. The division of this data regarding in-sample and out-of-sample tests is consistent with the division of bitcoin data. The two tables 5.3.3 5.3.3 show the results of regression tests.

IN-SAMPLE

MSE values (Traffic)					
Expanding window	EMD-CAE	MA-CAE	CAE	EMD	Original
1	74	141	137.38	105	137.76
2	72	162	151	100	144
3	62	163	132	93	150
4	61	157	131	99	153
5	65	158	125	100	150.56
6	55	133	102	89	131
7	54	129	107	84	127
8	52	130	92	79	134
9	67	149	117	107	152
10	48	126	89	78	124
Average	61	145	118	93	140

Table 13: MSE values of road congestion in in-sample tests

OUT-OF-SAMPLE

MSE values (Traffic)					
Expanding window	EMD-CAE	MA-CAE	CAE	EMD	Original
1	76	150	137.97	105	137.76
2	69	162	148	100	144
3	61	161	149	93	150
4	61	139	139	99	153
5	66	150.47	150.78	100	150.56
6	56	130	125	89	131
7	50	133	120	84	127
8	54	126	123	79	134
9	64	139	145	107	152
10	50	122	117	78	124
Average	61	141	135	93	140

Table 14: MSE values of road congestion in out-of-sample tests

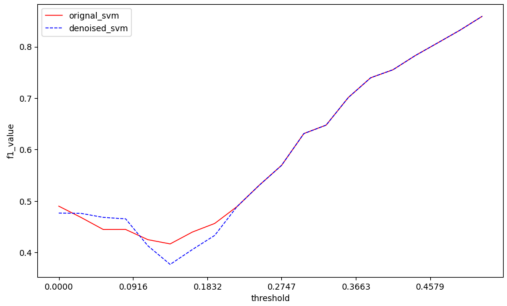
The EMD-CAE model appears to perform the best in regression tests. It may be because the noise in the traffic data set here is not as complex as it is in the financial

markets.

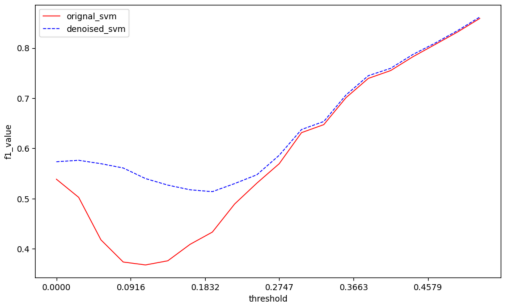
Then, the figures 5.3.3 5.3.3 show the results in classification tests. EMD-CAE and CAE model perform well in in-sample task. Their shapes appear different due to the existence of adaptive thresholds.

5.4 Brief summary

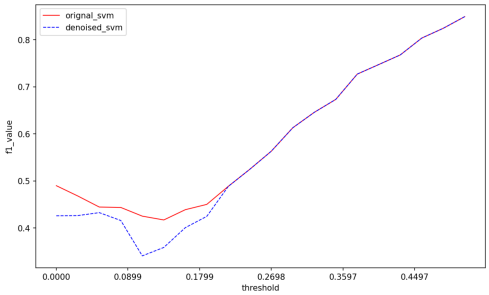
EMD-CAE performs admirably and is consistent across nine datasets. It is improved compared to the MA-CAE model and performs balanced in several tests. In subsequent comparisons, it becomes apparent that CAE and EMD have distinct advantages in the different tests. CAE performs well in the classification tasks, while EMD performs well in the regression tasks when the results of the three frequencies are combined. Importantly, EMD-CAE combines the strengths of both models and performs best and reliably through the testing process.



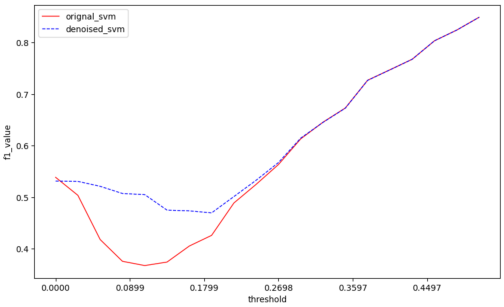
(a) EMD-CAE (1-1)



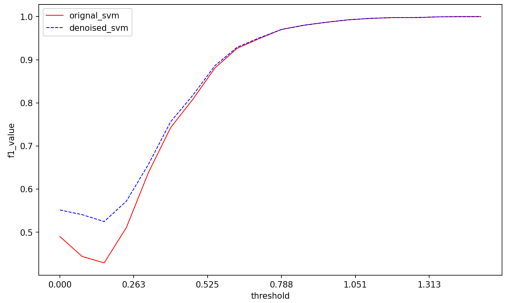
(b) EMD-CAE (10-1)



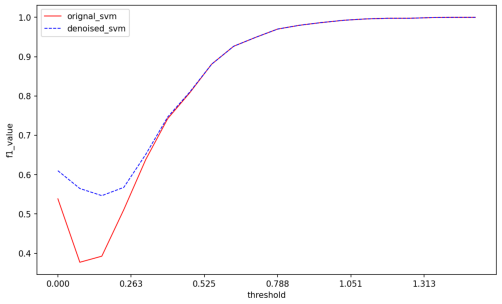
(c) MA-CAE (1-1)



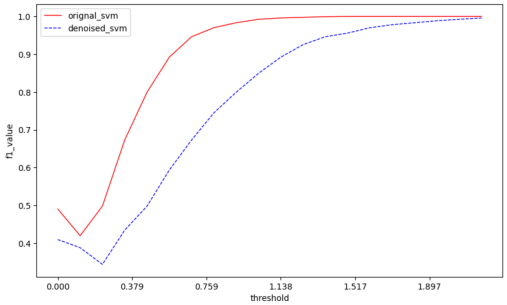
(d) MA-CAE (10-1)



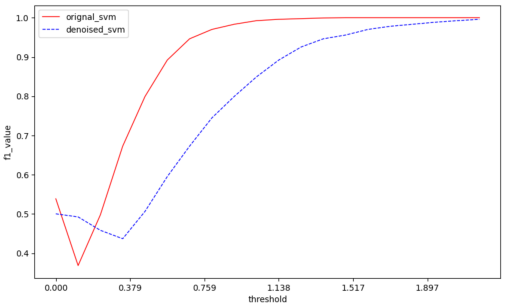
(e) CAE (1-1)



(f) CAE (10-1)

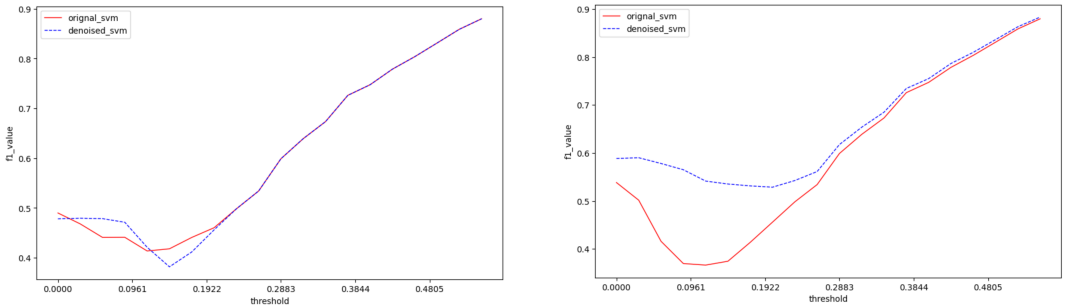


(g) EMD (1-1)



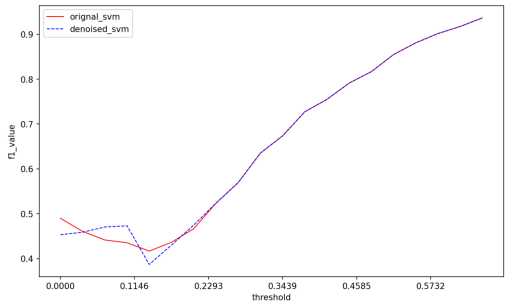
(h) EMD (10-1)

Figure 29: F1 plots of four models (in-sample)

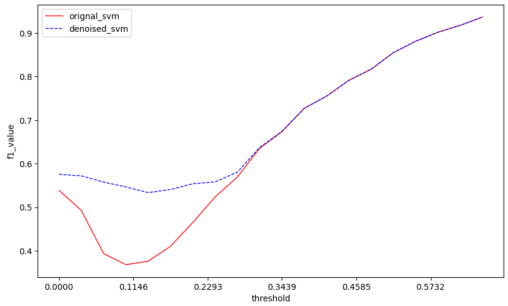


(a) EMD-CAE (1-1)

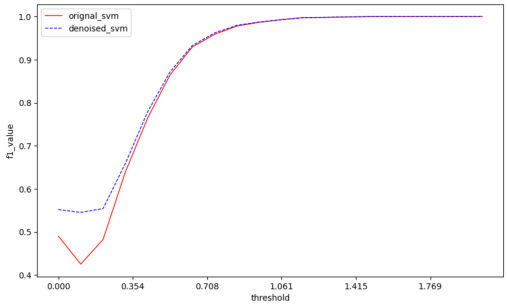
(b) EMD-CAE (10-1)



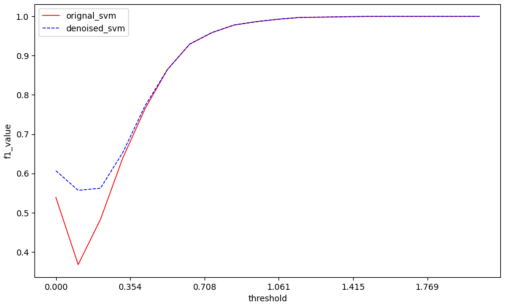
(c) MA-CAE (1-1)



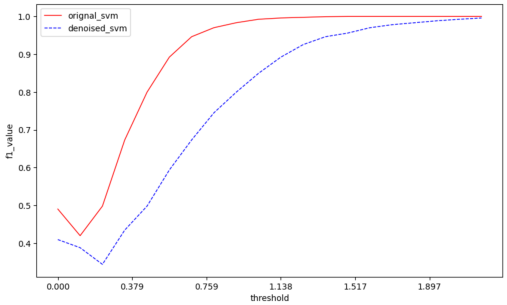
(d) MA-CAE (10-1)



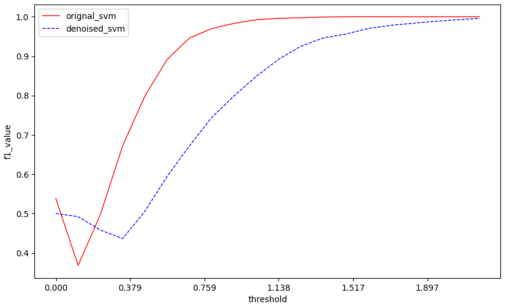
(e) CAE (1-1)



(f) CAE (10-1)



(g) EMD (1-1)



(h) EMD (10-1)

Figure 30: F1 plots of four models (out-of-sample)

6 Conclusion

In this chapter, we will briefly conclude the whole project and discuss possible future directions for the EMD-CAE denoising model.

6.1 Conclusion

This thesis was inspired by this key reference [1]. They were the first to use a self-supervised learning model to denoise financial trading data. We name their model as the MA-CAE model and further propose an EMD-CAE model along the direction of their research. The EMD-CAE model has many improvements in the design and testing compared with the original model. The central direction of this paper is denoising, which is part of data pre-processing in data mining. So, the design of the EMD-CAE model is the main contribution of this project. This model combines Empirical Mode Decomposition and Convolutional AutoEncoder models and attempts to enhance the signal-to-noise ratio in financial data.

Due to the inherent characteristics of financial trading data, it is challenging to define noise and impossible to obtain pure signal data. In order to validate the EMD-CAE model, a series of tests simulating the actual usage situations of trading data are designed. In terms of time, they can be categorised as in-sample and out-of-sample tests. From the perspective of trading model, they can be categorised as regression forecasting, classification forecasting, and trading strategy testing. In addition, we develop a distinct proof method for the regression and classification tests. The design information is in Chapter 4.

What is more, eleven data sets are selected to demonstrate the generalizability of the EMD-CAE model. They are from various financial markets and have various frequencies. We also test the effect of the EMD-CAE model on non-financial data.

Numerous comparisons demonstrate that the EMD-CAE model's denoised data is significantly superior to the MA-CAE model and the original data. Therefore, the EMD-CAE model is effective and practical for denoising. In addition, it combines the benefits of EMD and CAE models. It achieves better results in multiple tests than EMD or CAE alone. The EMD-CAE model's denoised data can provide a boost to subsequent data mining projects.

6.2 Future work

Although the EMD-CAE model is currently performing well, there are still several research avenues to be explored in the future.

- The simplest autoencoder model with two encoding layers and two decoding layers was chosen for this project. Perhaps other autoencoder models could be utilised to denoise better.
- It is interesting to find out what will occur if the convolutional network is replaced with a long-short-term memory network or a transformer network.
- Perhaps we could use other denoising methods in conjunction with CAE rather than EMD.
- In this project, we do not tune the EMD-CAE model's hyperparameters. It would be helpful if there was an algorithm to tune the model and make it more adaptable to various data sets.

References

[1] Y. Ma, C. Ventre, and M. Polukarov, "Denoised labels for financial time-series data via self-supervised learning," *arXiv preprint arXiv:2112.10139*, 2021.

[2] E. F. Fama, "Efficient capital markets: A review of theory and empirical work," *The journal of Finance*, vol. 25, no. 2, pp. 383–417, 1970.

[3] D. Kahneman and A. Tversky, "Prospect theory: An analysis of decision under risk," in *Handbook of the fundamentals of financial decision making: Part I*, pp. 99–127, World Scientific, 2013.

[4] W. F. De Bondt and R. H. Thaler, "Do security analysts overreact?," *The American economic review*, pp. 52–57, 1990.

[5] R. H. Thaler, "Anomalies: the january effect," *Journal of economic perspectives*, vol. 1, no. 1, pp. 197–201, 1987.

[6] R. H. Thaler, "Anomalies: weekend, holiday, turn of the month, and intraday effects," *Journal of Economic Perspectives*, vol. 1, no. 2, pp. 169–177, 1987.

[7] N. Jegadeesh and S. Titman, "Profitability of momentum strategies: An evaluation of alternative explanations," *The Journal of finance*, vol. 56, no. 2, pp. 699–720, 2001.

[8] B. E. Hansen, "The grid bootstrap and the autoregressive model," *Review of Economics and Statistics*, vol. 81, no. 4, pp. 594–607, 1999.

[9] E. J. Hannan, "The estimation of the order of an arma process," *The Annals of Statistics*, pp. 1071–1081, 1980.

[10] F. Black, "Noise," *The journal of finance*, vol. 41, no. 3, pp. 528–543, 1986.

[11] J. B. De Long, A. Shleifer, L. H. Summers, and R. J. Waldmann, "Noise trader risk in financial markets," *Journal of political Economy*, vol. 98, no. 4, pp. 703–738, 1990.

[12] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *International conference on artificial neural networks*, pp. 52–59, Springer, 2011.

[13] G. Rilling, P. Flandrin, P. Goncalves, *et al.*, "On empirical mode decomposition and its algorithms," in *IEEE-EURASIP workshop on nonlinear signal and image processing*, vol. 3, pp. 8–11, Citeseer, 2003.

[14] P. Newbold, *Statistics for business and economics*. Pearson, 2013.

[15] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the royal statistical society. series c (applied statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[16] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," tech. rep., Stanford, 2006.

[17] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "Dbscan revisited, revisited: why and how you should (still) use dbscan," *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 3, pp. 1–21, 2017.

[18] G. A. Seber and A. J. Lee, *Linear regression analysis*. John Wiley & Sons, 2012.

[19] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

[20] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: unbiased boosting with categorical features," *Advances in neural information processing systems*, vol. 31, 2018.

[21] P. Liu, H. Zhang, K. Zhang, L. Lin, and W. Zuo, "Multi-level wavelet-cnn for image restoration," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 773–782, 2018.

[22] Y. Zhao, Z. Jiang, A. Men, and G. Ju, "Pyramid real image denoising network," in *2019 IEEE Visual Communications and Image Processing (VCIP)*, pp. 1–4, IEEE, 2019.

[23] N. E. Huang, Z. Shen, S. R. Long, M. C. Wu, H. H. Shih, Q. Zheng, N.-C. Yen, C. C. Tung, and H. H. Liu, "The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis," *Proceedings of the Royal Society of London. Series A: mathematical, physical and engineering sciences*, vol. 454, no. 1971, pp. 903–995, 1998.

[24] R. N. Bracewell and R. N. Bracewell, *The Fourier transform and its applications*, vol. 31999. McGraw-Hill New York, 1986.

[25] D. Griffin and J. Lim, "Signal estimation from modified short-time fourier transform," *IEEE Transactions on acoustics, speech, and signal processing*, vol. 32, no. 2, pp. 236–243, 1984.

[26] C. Torrence and G. P. Compo, "A practical guide to wavelet analysis," *Bulletin of the American Meteorological society*, vol. 79, no. 1, pp. 61–78, 1998.

[27] Z. Wu and N. E. Huang, "Ensemble empirical mode decomposition: a noise-assisted data analysis method," *Advances in adaptive data analysis*, vol. 1, no. 01, pp. 1–41, 2009.

[28] J.-R. Yeh, J.-S. Shieh, and N. E. Huang, "Complementary ensemble empirical mode decomposition: A novel noise enhanced data analysis method," *Advances in adaptive data analysis*, vol. 2, no. 02, pp. 135–156, 2010.

[29] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[30] H. Bourlard and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition," *Biological cybernetics*, vol. 59, no. 4, pp. 291–294, 1988.

[31] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, "Deep learning-based classification of hyperspectral data," *IEEE Journal of Selected topics in applied earth observations and remote sensing*, vol. 7, no. 6, pp. 2094–2107, 2014.

[32] E. Li, P. Du, A. Samat, Y. Meng, and M. Che, "Mid-level feature representation via sparse autoencoder for remotely sensed scene classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, no. 3, pp. 1068–1081, 2016.

[33] M. Sabokrou, M. Fathy, and M. Hoseini, "Video anomaly detection and localisation based on the sparsity and reconstruction error of auto-encoder," *Electronics Letters*, vol. 52, no. 13, pp. 1122–1124, 2016.

[34] M. Ribeiro, A. E. Lazzaretti, and H. S. Lopes, "A study of deep convolutional auto-encoders for anomaly detection in videos," *Pattern Recognition Letters*, vol. 105, pp. 13–22, 2018.

[35] X. Yan, J. Yang, K. Sohn, and H. Lee, "Attribute2image: Conditional image generation from visual attributes," in *European conference on computer vision*, pp. 776–791, Springer, 2016.

[36] R. Sabathé, E. Coutinho, and B. Schuller, "Deep recurrent music writer: Memory-enhanced variational autoencoder-based musical score composition and an objective measure," in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 3467–3474, IEEE, 2017.

[37] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103, 2008.

[38] D. Stowell and R. E. Turner, "Denoising without access to clean data using a partitioned autoencoder," *arXiv preprint arXiv:1509.05982*, 2015.

[39] M. Nishio, C. Nagashima, S. Hirabayashi, A. Ohnishi, K. Sasaki, T. Sagawa, M. Hamada, and T. Yamashita, "Convolutional auto-encoder for image denoising of ultra-low-dose ct," *Heliyon*, vol. 3, no. 8, p. e00393, 2017.

[40] H. Zhou, Y. Guo, and K. Guo, "Seismic random noise attenuation using a tied-weights autoencoder neural network," *Minerals*, vol. 11, no. 10, p. 1089, 2021.

[41] P. J. Brockwell and R. A. Davis, *Time series: theory and methods*. Springer science & business media, 2009.

[42] X. Liu, F. Zhang, Z. Hou, L. Mian, Z. Wang, J. Zhang, and J. Tang, "Self-supervised learning: Generative or contrastive," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[43] L. Kahsay, F. Schwenker, and G. Palm, "Comparison of multiclass svm decomposition schemes for visual object recognition," in *Joint Pattern Recognition Symposium*, pp. 334–341, Springer, 2005.

[44] L. Gondara, "Medical image denoising using convolutional denoising autoencoders," in *2016 IEEE 16th international conference on data mining workshops (ICDMW)*, pp. 241–246, IEEE, 2016.

[45] G. Han, B. Lin, and Z. Xu, "Electrocardiogram signal denoising based on empirical mode decomposition technique: an overview," *Journal of Instrumentation*, vol. 12, no. 03, p. P03010, 2017.

[46] Y. Zhang, X. Ma, D. Hua, Y. Cui, and L. Sui, "An emd-based denoising method for lidar signal," in *2010 3rd International Congress on Image and Signal Processing*, vol. 8, pp. 4016–4019, IEEE, 2010.

[47] M. Mörke, "Marcos lópez de prado: Advances in financial machine learning," 2019.

[48] J. J. Murphy, *Technical analysis of the financial markets: A comprehensive guide to trading methods and applications*. Penguin, 1999.

[49] G. Appel, *Technical analysis: power tools for active investors*. FT Press, 2005.

[50] C. R. Harvey and Y. Liu, "Backtesting," *The Journal of Portfolio Management*, vol. 42, no. 1, pp. 13–28, 2015.

Declaration

I declare that this thesis is the solely effort of the author. I did not use any other sources and references than the listed ones. I have marked all contained direct or indirect statements from other sources as such.

Neither this work nor significant parts of it were part of another review process. I did not publish this work partially or completely yet. The electronic copy is consistent with all submitted copies.

Signature and date: Tianshu Chu August 8, 2022

A Appendix

A.1 Denoised results of the rest small data sets

A.1.1 IN-SAMPLE

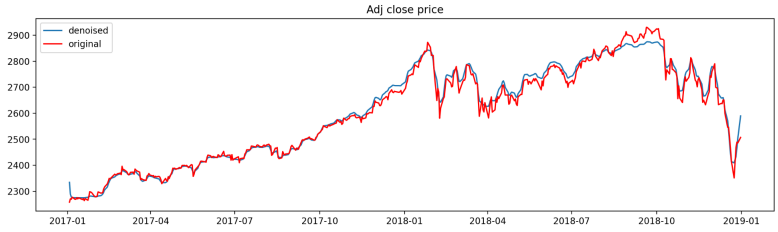


Figure 31: ∧GSPC

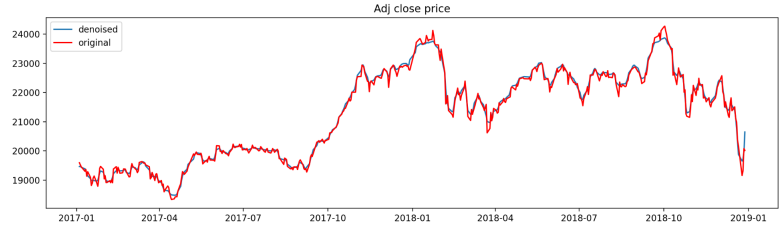


Figure 32: ∧N225

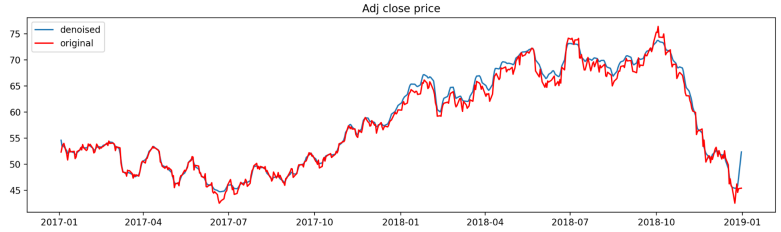


Figure 33: CL=F

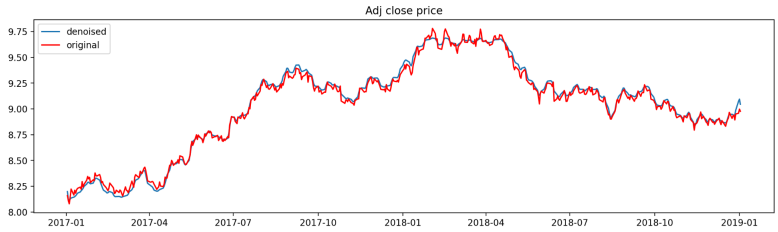


Figure 34: EURHKD=X

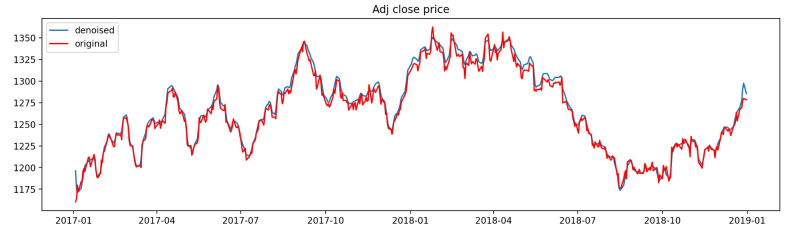


Figure 35: GC=F

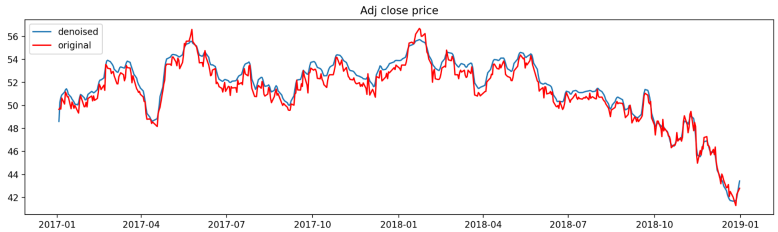


Figure 36: LLOY.L

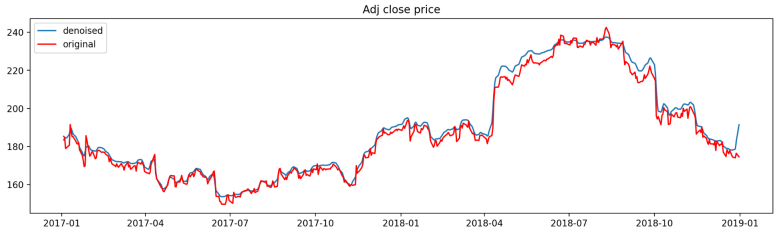


Figure 37: TSCO.L

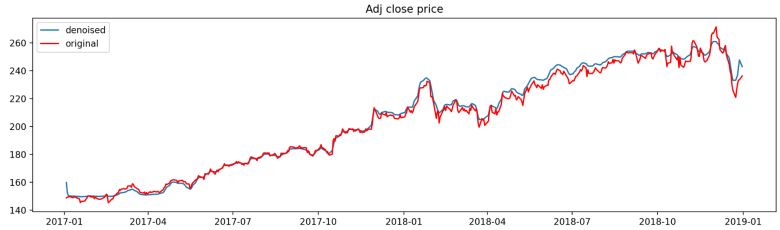


Figure 38: UNH

A.1.2 OUT-SAMPLE

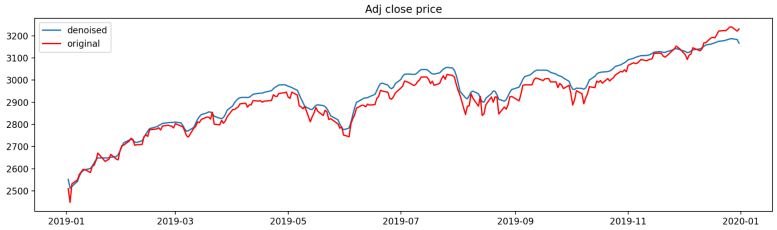


Figure 39: ∧GSPC

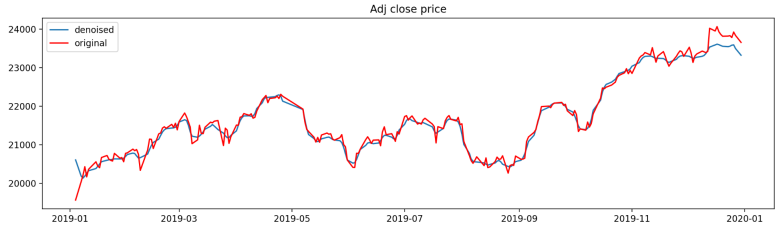


Figure 40: ∧N225

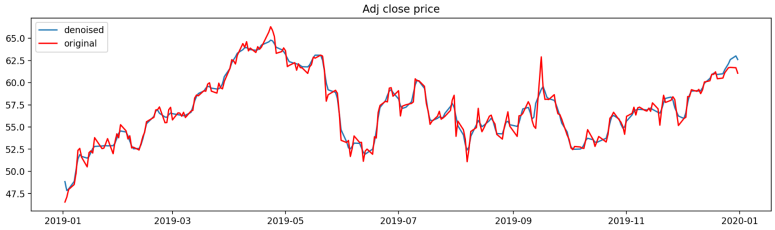


Figure 41: CL=F

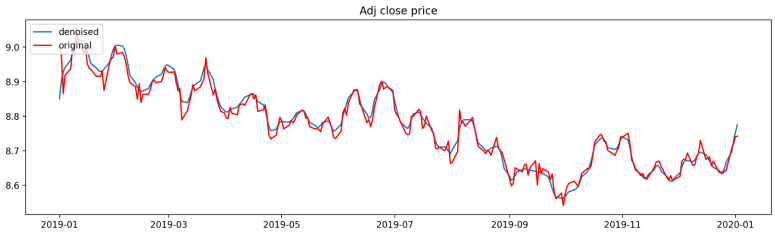


Figure 42: EURHKD=X

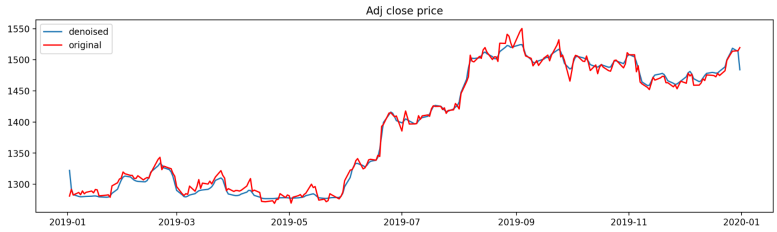


Figure 43: GC=F

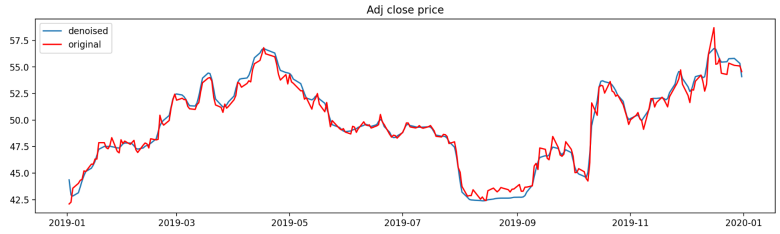


Figure 44: LLOY.L

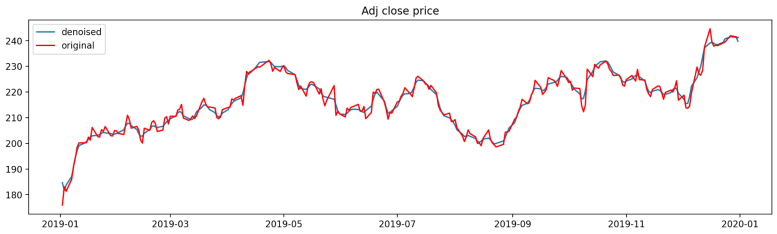


Figure 45: TSCO.L

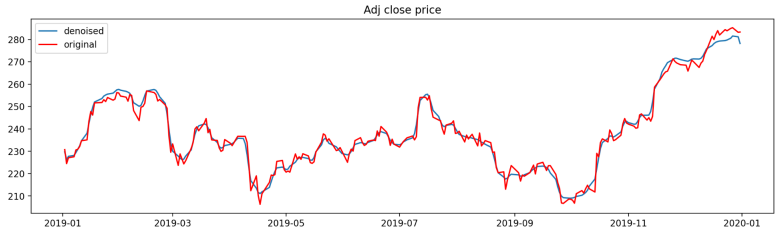


Figure 46: UNH

A.2 Regression tests of the rest small data sets

A.2.1 IN-SAMPLE

IN-SAMPLE			
	MSE values		
Expanding window	EMD-CAE	MA-CAE	Original
1	1280	1614	2153
2	2281	1602	1824
3	6916	9763	9295
4	6075	5364	5425
5	3198	4500	3484
6	1215	3042	1631
7	256	1908	210
8	2088	33250	2939
9	1494	10570	2227
10	2496	8357	4746
Average	2730	7997	3393

Table 15: ∧ GSPC regression testing (in-sample)

IN-SAMPLE			
	MSE values		
Expanding window	EMD-CAE	MA-CAE	Original
1	1137180	1483985	797903
2	44765	149007	53873
3	2400711	2728199	2381638
4	249677	338949	178107
5	240683	1079349	520150
6	75369	301567	125021
7	30037	485399	57966
8	34324	188680	57778
9	150680	506656	280074
10	140548	650034	302545
Average	450397	791182	475506

Table 16: ∧N225 regression testing (in-sample)

IN-SAMPLE			
	MSE values		
Expanding window	EMD-CAE	MA-CAE	Original
1	21	17	15
2	10	23	4
3	8	19	5
4	3	36	3
5	5	5	4
6	21	34	17
7	7	23	9
8	2	5	2
9	6	14	6
10	12	136	12
Average	9	31	8

Table 17: CL=F regression testing (in-sample)

IN-SAMPLE			
	MSE values		
Expanding window	EMD-CAE	MA-CAE	Original
1	0.017403	0.021163	0.018442
2	0.065241	0.076567	0.069586
3	0.020952	0.036004	0.019802
4	0.005792	0.013270	0.005114
5	0.031107	0.098543	0.050872
6	0.005570	0.017103	0.009474
7	0.002583	0.020940	0.003557
8	0.009550	0.010273	0.006495
9	0.003421	0.017805	0.003788
10	0.003630	0.0065654	0.007370
Average	0.016525	0.031823	0.019450

Table 18: EURHKD=X regression testing (in-sample)

IN-SAMPLE			
	MSE values		
Expanding window	EMD-CAE	MA-CAE	Original
1	307	647	312
2	1634	5081	2427
3	149	861	172
4	311	581	204
5	304	938	220
6	140	287	363
7	104	671	107
8	243	839	379
9	85	362	126
10	55	318	97
Average	333	1059	441

Table 19: GC=F regression testing (in-sample)

IN-SAMPLE			
	MSE values		
Expanding window	EMD-CAE	MA-CAE	Original
1	0.49	7.79	1.38
2	0.35	0.66	0.52
3	0.71	2.06	1.11
4	0.59	1.35	0.59
5	1.77	3.01	2.34
6	0.23	0.70	0.34
7	0.17	1.07	0.36
8	0.47	0.62	0.39
9	2.12	4.87	3.64
10	6.55	7.59	6.91
Average	1.35	2.97	1.76

Table 20: LLOY.L regression testing (in-sample)

IN-SAMPLE			
	MSE values		
Expanding window	EMD-CAE	MA-CAE	Original
1	42	53	38
2	33	61	60
3	2	61	7
4	153	312	263
5	12	15	11
6	371	445	457
7	269	665	298
8	22	19	21
9	185	501	98
10	44	106	30
Average	113	224	128

Table 21: TSCO.L regression testing (in-sample)

IN-SAMPLE			
	MSE values		
Expanding window	EMD-CAE	MA-CAE	Original
1	96	88	98
2	65	59	61
3	38	49	37
4	112	132	147
5	112	133	117
6	13	51	40
7	237	509	319
8	94	95	108
9	12	19	16
10	98	157	134
Average	90	129	108

Table 22: UNH regression testing (in-sample)

A.2.2 OUT-SAMPLE

OUT-SAMPLE			
	MSE values		
Expanding window	EMD-CAE	MA-CAE	Original
1	944	625	708
2	5786	9790	7829
3	2010	1636	4808
4	5159	21782	5092
5	2919	2241	2264
6	1572	8221	1188
7	731	2722	851
8	4678	9667	5024
9	3015	4244	1571
10	8501	10987	8170
Average	3532	7191	3751

Table 23: ∧GSPC regression testing (out-sample)

OUT-SAMPLE			
	MSE values		
Expanding window	EMD-CAE	MA-CAE	Original
1	25115	114707	109204
2	202921	261264	224166
3	248052	142890	63766
4	23100	63015	48205
5	277123	721245	353762
6	64841	386612	44737
7	104584	472384	131182
8	1011937	3172950	1228050
9	21345	2051032	44956
10	1058507	898809	895296
Average	303752	828491	314332

Table 24: ∧N225 regression testing (out-sample)

OUT-SAMPLE			
	MSE values		
Expanding window	EMD-CAE	MA-CAE	Original
1	2	2	2
2	26	26	32
3	7	108	7
4	3	20	8
5	12	42	12
6	4	8	4
7	2	16	1
8	0.8	5	1
9	1	2	9
10	0.3	8	0.8
Average	6	24	8

Table 25: CL=F regression testing (out-sample)

OUT			
	MSE values		
Expanding window	EMD-CAE	MA-CAE	Original
1	0.001564	0.002271	0.004321
2	0.000528	0.000678	0.001031
3	0.005197	0.009454	0.007004
4	0.001651	0.001918	0.001904
5	0.002616	0.014415	0.005503
6	0.005318	0.006773	0.006876
7	0.001632	0.001678	0.003566
8	0.005840	0.002579	0.007019
9	0.000786	0.002547	0.000488
10	0.000425	0.000844	0.001454
Average	0.002556	0.004316	0.003917

Table 26: EURHKD=X regression testing (out-sample)

OUT-SAMPLE			
	MSE values		
Expanding window	EMD-CAE	MA-CAE	Original
1	807	622	1081
2	422	686	533
3	4554	7514	5469
4	710	1352	961
5	5998	7001	6138
6	600	415	524
7	606	253	281
8	153	148	150
9	701	706	459
10	214	533	177
Average	1477	1923	1577

Table 27: GC=F regression testing (out-sample)

OUT-SAMPLE			
	MSE values		
Expanding window	EMD-CAE	MA-CAE	Original
1	1.5	1.6	2.5
2	8.7	17.1	6.1
3	0.6	4.4	1.1
4	0.7	3.6	0.5
5	54.7	49.5	48.2
6	2.6	2.8	2.9
7	4.0	6.4	3.2
8	1.3	6.1	1.8
9	0.7	1.0	1.0
10	1.4	11.3	2.1
Average	7.6	10.4	6.9

Table 28: LLOY.L regression testing (out-sample)

OUT-SAMPLE			
	MSE values		
Expanding window	EMD-CAE	MA-CAE	Original
1	24	87	14
2	81	95	94
3	9	19	20
4	12	23	9
5	184	402	156
6	16	97	35
7	11	234	33
8	5	127	10
9	9	37	19
10	104	431	108
Average	45	155	50

Table 29: TSCO.L regression testing (out-sample)

OUT-SAMPLE			
	MSE values		
Expanding window	EMD-CAE	MA-CAE	Original
1	80.6	96.5	57.6
2	42.8	41.4	83.5
3	27.8	17.3	29.8
4	202.3	152.2	143.6
5	16.1	36.0	71.8
6	14.2	29.3	22.6
7	112.8	105.9	75.6
8	48.7	137.5	61.8
9	336.5	520.2	285.1
10	659.3	898.6	715.1
Average	154.1	203.5	154.6

Table 30: UNH regression testing (out-sample)

A.3 Classification tests of the rest small data sets

A.3.1 IN-SAMPLE

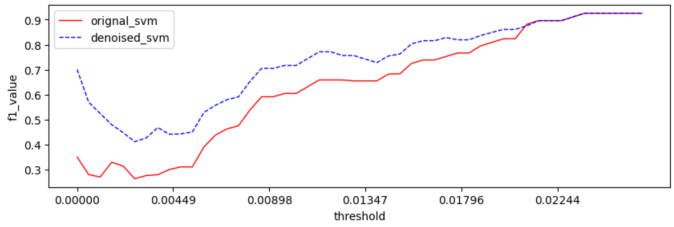


Figure 47: ∧GSPC F1

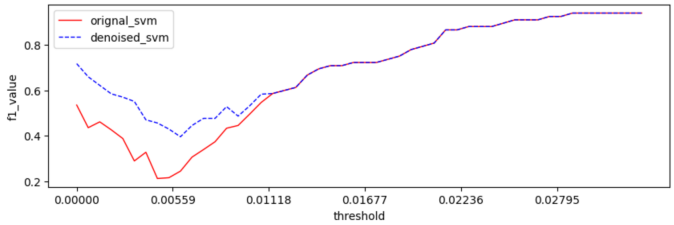


Figure 48: ∧N225 F1

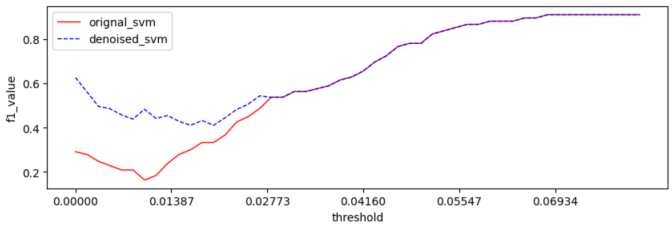


Figure 49: CL=F F1

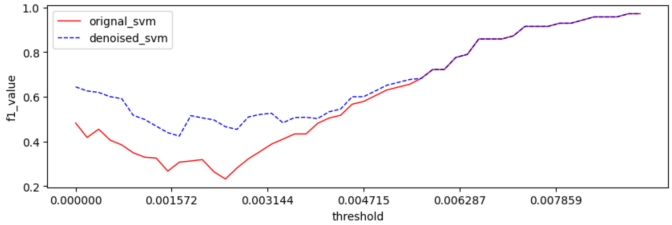


Figure 50: EURHKD=X F1

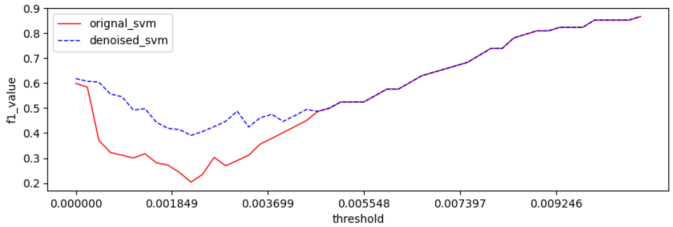


Figure 51: GC=F F1

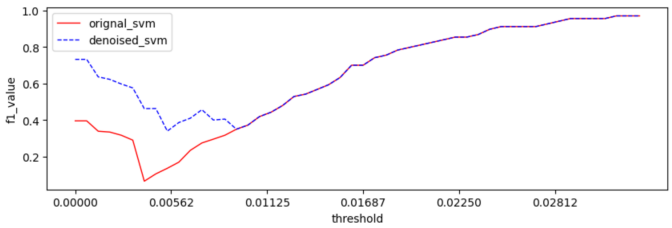


Figure 52: LLOY.L F1

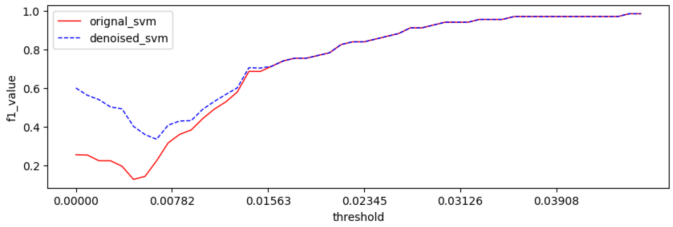


Figure 53: TSCO.L F1

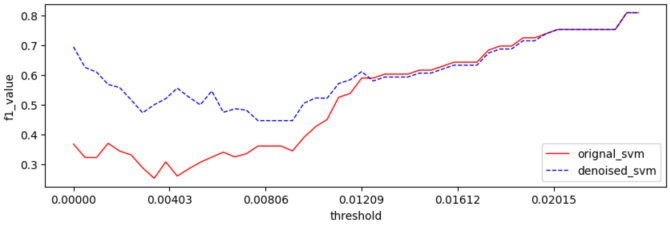


Figure 54: UNH F1

A.3.2 OUT-SAMPLE

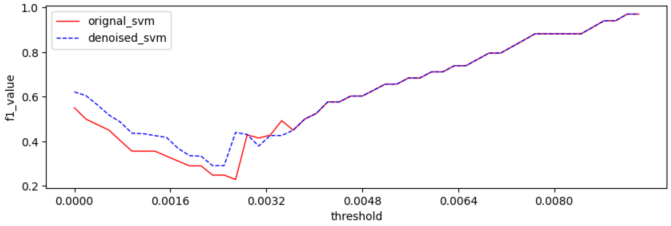


Figure 55: ∧GSPC F1

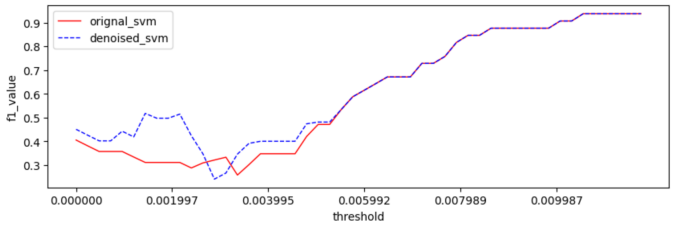


Figure 56: ∧N225 F1

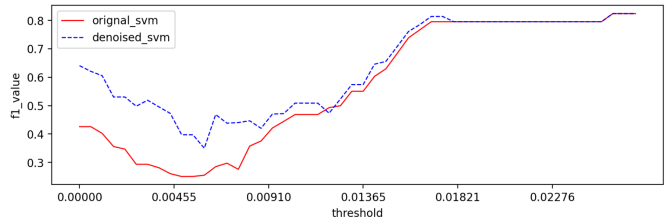


Figure 57: CL=F F1

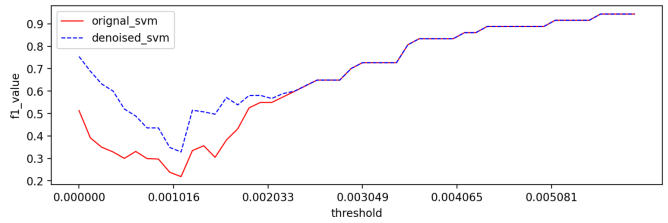


Figure 58: EURHKD=X F1

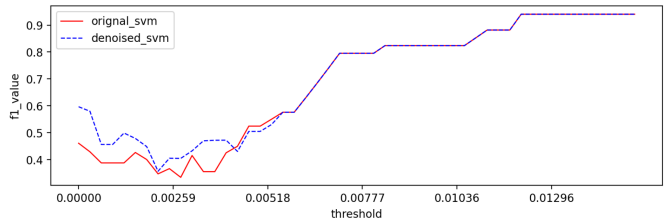


Figure 59: GC=F F1

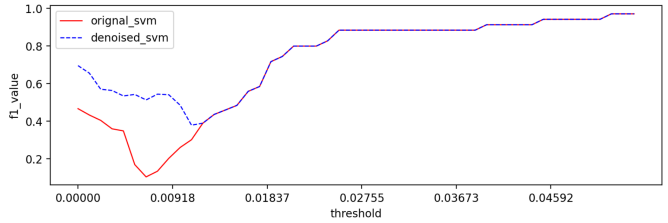


Figure 60: LLOY.L F1

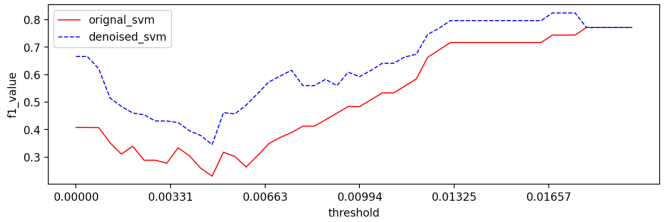


Figure 61: TSCO.L F1

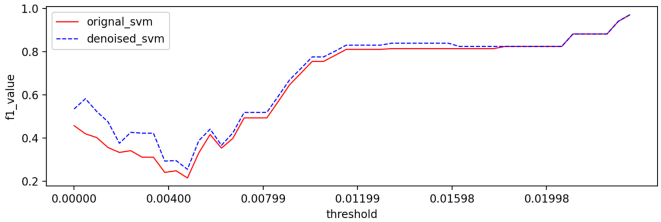


Figure 62: UNH F1