Prediction Time-Series Foundation Models in Finance

by

Filip Topic

5

September 2024

Dissertation Supervisor: Ramin Okhrati Dissertation submitted in part in

fulfilment of the Master of Science degree in Banking and Digital Finance

Institute of Finance and Technology University College London

ABSTRACT

Keywords:

ACKNOWLEDGEMENTS

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Time-Series prediction

Since the beginning of financial markets, their participants have had the desire to predict the future values of instruments being traded there. And for this purposes, they have used many different techniques - majority of which have fallen short of randomly guessing the direction of the price movement. In the past few decades, many models have emerged which claim excellent capability of time-series prediction. First were the statistical models (such as ARIMA), then the Machine Learning models (such as XGBoost), then Deep Learning models (such as DeepAR).

1.2 Transformer-based Models

The revolution in the field of Machine Learning came in 2017 with the invention of the Transformer architecture (more specifically: the attention mechanism). Transformer based models have taken the whole field of ML by storm, however, their application has mostly been in the sub-field of Natural Language Processing. Recently they started seeing use in the field of time-series prediction (eg. Informer).

Time Series Foundation Models (TSFM)

Time-series foundation models are large transformer-based models which are designed for the purpose of time-series prediction and which have been pre-trained on a large amount of time-series data. This research will explore their use on financial time-series data

2. BACKGROUND

2.1 The Transformer

In 2017, Vaswani et al. [54] introduced a ground-breaking model called the "Transformer". Transformer is a model designed to solve the sequence-to-sequence mapping problem. In a sequence-to-sequence problem we have a sequence which consists of tokens[1] coming from a finite vocabulary[2] which are in a specific order, and we have an output sequence which is again a sequence of tokens in a specific order. The task of a sequence-to-sequence model is to learn the correct relationship between the input and output sequences so that when the model is presented with an unseen input sequence, it can predict what the correct output sequence is. In essence, sequence-to-sequence model's goal is to learn the "meaning" of the relationship between the input and output sequence. An example of sequence-to-sequence tasks are:

1. Language translation - where an input sequence could be a sentence in our native language and the output sequence would be the correct translation of that sentence in a foreign language.
2. Chatbots - where an input sequence could be a question and the output sequence the correct answer to that question.
3. Time-series forecasting - where an input sequence is a certain time-series and the output sequence is the future values of that time-series.

Before the Transformer, there have been many ML models which attempted to solve these tasks. Sutskever et al. (2014) [48] used multilayer LSTM[3] (type of RNN[4] model) for this task. LSTM [24] and GRU[5] [9] used to be state-of-the art sequence-to-sequence

[1]A token is a multi-dimensional numeric vector representation of an element in the sequence. Reason why sequence of tokens is used as input to a sequence-to-sequence model is because ML models can only "understand" numbers - therefore they can't work with some type of sequences (eg. human language sentences). Reason why tokens are vectors rather than simple nubers is because they are designed to represent the semantic meaning of a real-world element they represent and the semantic meaning of a real-world element could change depending on the context. Therefore different dimensions of a token account for different meaning of that element in different contexts.

[2]The vocabulary doesn't always need to be finite, as we will see later on.

[3]LSTM stands for Long Short Term Memory cell

[4]RNN stands for Recurrent Neural Network. It is a type of Artificial Neural Network architecture.

[5]GRU stands for Gated Recurrent Unit. GRU is also a type of RNN.

models, however they have some key issues. They require large memory[6], their nature is sequential[7], and they suffer from an information bottleneck due to the fixed size of the hidden state[8]. These problems were adressed by the Transformer.

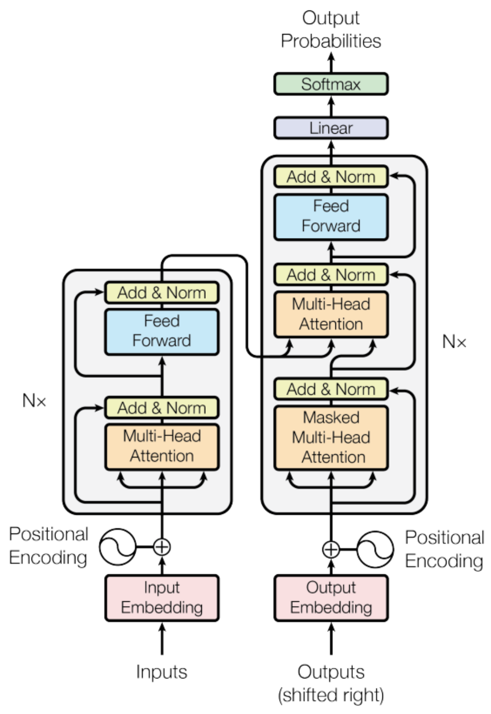

Figure 2.1: Transformer architecture schema [54]

Transformer consists of and Encoder and a Decoder[9]. Figure 2.1 is a schematic representation of the Transformer model. Left part is the Encoder and the right part is the Decoder.

Encoder

The encoder is a stack of multiple identical layers[10], each layer consisting of two sub-layers:

1. Multi-head self-attention layer

[6]This means that it is difficult to parallelize the training process across training examples and smaller batch sizes had to be used.

[7]This means that it is impossible to parallelize training within the training examples.

[8]The hidden state is an intermittent sequence within a sequence-to-sequence model. It is a product of applying the encoder on the input sequence. The output sequence is generated by applying the decoder to the hidden state. Having a hidden state of fixed size means that some information will inherently get lost when a very long input sequence is fed into the model.

[9]Similar to already mentioned RNNs.

[10]In the original paper there are 6 of these layers in the encoder stack

2. Feed-forward neural network

Multi-head Self-attention layer consists of N so-called "attention heads"[11]. An attention head is essentially a series of arithmetic operations performed on the input sequence: Let's say we have a sequence of tokens $S = s_1, s_2, ..., s_n$ the dimension of input tokens is 1 x d_{model} [12]. An attention head consists of weight matrices[13] W^Q, W^K and W^V [14] of the dimension (d_{model} x d_k) where d_k=d_{model}/N. Multiplying these weight matrices by token s_i from the input sequence creates Q_i, K_i and V_i vectors respectively of dimension 1 x d_k. For each token i, dot product of Q_i and K_j (for all j, 0<j<n+1) is calculated: $dot_{i,\,1}$, $dot_{i,\,2}$, ..., $dot_{i,\,n}$. These dot-products are then scaled[15], a softmax layer is applied[16] to all these dot products. Each dot product $dot_{i,\,j}$ then multiplies the corressponding V_j to get the series of V vectors: $V_{i,\,1}$, $V_{i,\,2}$, ..., $V_{i,\,n}$. All these V vectors are then element-wise summed up to Z_i (of dimension 1 x d_k) which represents the self-attention output for the token i. This is repeated for all n tokens and creates a series of vectors Z_1, Z_2, ..., Z_n (one for each token) which are then vertically stacked into matrix Z of dimension n x d_k. This matrix Z_h is the output of the single self-attention head h. If we vectorize this process[17], we can express attention as

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

See Figure 2.2:

This process goes on in parallel in every head of the multi-head self-attention layer. Finally, the Z matrices of each self-attention head Z_1, Z_2, ..., Z_N are horizontally concatenated to produce the output of the whole multi-head self-attention layer L: Z_L (of dimension n x N*d_k. Multi-head self-attention layer has a weight matrix W_O of dimension N*d_k x d_{model} which gets multiplied by Z_L to produce the final output of the multi-head self-attention layer L.

[11]In the Original paper, N = 8.

[12]In the original paper, d(model) = 512.

[13]These weight matrices are all learnable weights.

[14]Q, K, and V stand for Query, Key and Value.

[15]Scaling factor is 1 / square root of d(k). According to the paper, this is done so that the weights are more stable during training.

[16]Applying Softmax on a series of numbers means scaling all of them so that they sum up to 1.

[17]When we need to do the same arithmetic operations multiple times with different vectors, we can simply concatenate these vectors into matrices and do the operations just once with these matrices.

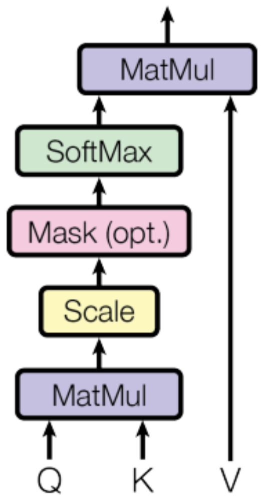

Figure 2.2: Scaled Dot product attention [54]

Before the output of the multi-head self-attention layer goes into a feed-forward neural network[18], a residual connection[19] [22] and layer normalization[20] [3] are applied. Residual connection and layer normalization are applied after every sub-layer of the Encoder and Decoder.

Decoder

The Decoder has a similar structure as the Encoder with a few notable differences.

Same as the Encoder, Decoder has N layers. Each layer has three sub-layers:

1. masked self-attention layer
2. encoder-decoder attention layer
3. feed forward layer

In principle, masked self-attention layer works similarly as the multi-head attention

[18]This network has two hidden layers with ReLU actiavation of the first layer and linear activation of the 2nd layer. In the original paper, dimensionality of hidden layers is 2048.

[19]Residual connection in this context means adding the input of the layer to the output of that layer before passing to the next layer. This is used to mitigate the vanishing gradient problem and to stabilize the training process.

[20]Layer normalization is a scaling technique that normalizes the the data across features within each data-sample indivisdually, as opposed to batch normalization which normalizes each feature across multiple data-samples.

layer in the Encoder, however, Q, K, and V come from the already generated output[21], and the attention is calculated only for the current and previously generated outputs i.e. for an already generated token i, masked self-attention head only calculates Z_1, Z_2, ..., Z_i. Otherwise, the process is the same as in the Encoder multi-head self-attention sub-layer[22]. (See Figure 2.1.)

Encoder-decoder attention layer also works similarly as the multi-head self-attention layer in the Encoder, however in this layer Q comes from the previous masked self-attention layer but K and V come from the Encoder. This allows every position in the Encoder to attend all positions in the input. (See Figure 2.1.)

Summary

Even though the Attention mechanism has been around for a while [4], the self-attention model, as implemented in the Transformer, was a revolutionary step - as it solved the issue of long range dependancies[23]. This concept enables the model to learn the intrinsic structure of the sequence it is presented with and therefore builds a certain level of actual "understanding" of the inputs rather than simple "fitting".

Another revolutionary aspect of the Transformer is the positional encoding. Positional encoding injects information about the absolute and relative position of each token in the sequence[24] thus allowing the model to learn about the importance of relative positions of tokens[25].

Finally, and probably most importantly, the way Transformer processes inputs is inherently parallelizable - allowing significant scaling benefits to be exploited and very large models to be built. This is beneficial as larger models can learn more intricate

[21]This makes the Decoder particularly suitable for autoregressive tasks as it predicts tokens sequentially based on which tokens have been predicted before, as opposed to the Encoder which would predict a sequence of tokens which it thinks would have the highest probability of being true overall without regard for the order of the sequence.

[22]i.e. Encoder also stacks the outputs of each head, applies the Wo matrix, the residual connection and layer normalization.

[23]Earlier attention models had issues relating tokens which were too far apart due to attention being applied only on hidden states (hidden states are a feature of RNN models) rather than on inputs directly.

[24]this is done by sumating each input token with a positional vector of equal dimension. Value of each number in the positional vector is a sin (or could be cos) function of the position of the token it is being added to as well as of the position inside the positional vector.

[25]This is most important on the field of NLP (Natural Language Processing) tasks.

patterns in the data and they allow for richer vector representations of the input data[26] [27]

2.2 Transformer-based models

Many have built on the success of the Transformer and have come up with their own models which retain many concepts from the original Transformer.

NLP Transformer-based models

Among the most famous NLP transformer-based models are the BERT (2018) [14], T5 (2019) [40] which themselves have been built upon many times (2019 alBERT [29], 2019 roBERTa [32], 2019 distilBERT [42], 2020 mT5 [58], 2024 flan-T5 [10]), however, OpenAI's GPT[28] series and LLaMA series [53] take the crown as the most famous Transformer-based NLP models by far.

CV (computer vision) Transformer-based models

Among the most famous CV transformer-based models are 2020 ViT [17], 2020 DETR [7], and 2021 swin-Transformer [34].

Speech and Audio processing Transformer-based models

Time-series Transformer-based models

It was a matter of time Transformer-based architectures appeared in time-series prediction tasks. Li et al. (2019) [30] wrote a pioneering work on transformers (LogSparse Transformer) in time-series forecasting where they adressed the issue of the quadratic space complexity[29] of the Transformer[30] and proposed a more task-appropriate

[26]Vector representations in the original Transformer had the dimension 512, whereas some modern transformer-based models support dimensions in the thousands.

[27]Richer representations of the input data means that the model is able to better understand more nuanced differences between some similar input tokens.

[28]GPT stands for Generative Pre-trained Transformer

[29]Quadratic space complexity of a model means that the ammount of working memory a model uses grows quadratically with the size of the input of the model. This is especially problematic for longer inputs for which the required memory could explode.

[30]They used heuristic approach to reduce the storage complexity to O(L*log(L)).

version of self-attention[31]. Zhou et al. (2021) [62] proposed a model (Informer) which also reduces computational and space complexity to O(L*log(L)) using "ProbSparse" self-attention mechanism[32] and self-attention distilling[33]. Zhou et al. (2021) [63] incorporate a seasonal-trend decomposition approach [11] [55], along with Fourier analysis into the transformer-based model (FEDformer)[34]. This approach saw great improvements in time-series prediction. Improvements were in terms of lower prediction errors as well as in terms of the distribution of predictions being closer to the distribution of ground-truth values according to the Kolmogorov-Smirnov distribution test[35] [35].

2.3 Time-series Foundation models (TSFM)

Foundation Models (FM) are a class of deep models which are pre-trained on a large ammount of data - thus being able to generalize[36] well as they have been taught many diverse patterns. FMs saw success on the fields of CV and NLP so it is only natural that development of FMs start developing on the time-series front. And so was the case: Since 2022, over 50 models, which can be classified as Time-series Foundation Models, have been released. If we analyze the TSFM landscape, according to the taxonomy proposed by Liang et al. (2024) [31], we can see that the TSFM is a very diverse class of models. TSFMs can be classified according to:

- Type of time-series it is working with, which can be:
    1. Standard time-series[37].
    2. Spatial time-series[38].

[31]They proposed so-called "convolutional self-attention" which brings local context awareness to Q-K mathcing

[32]ProbSparse self-attention uses Query Sparsity Measurement - a metric which helps determine which Qs are more "dominant" so that the attention can be calculated for only those Qs. This is an upgrade from LogSparse Transformer, which used a heuristic to determine Which QK pairs will be calculated.

[33]Distilling means that the outputs of each the self-attention layers are smaller than their inputs - thus reducing the number of calculations in every subsequent layer relative to the previous one.

[34]FED stands for Frequency Enhanced Decomposition.

[35]Authors of the FEDformer claim even though Informer predictions were good, they don't have the same distribution as the ground-truth time-series.

[36]Generalization is the ability of a model to perform well on data that it hasn't been trained on.

[37]Standard time-series is a simple sequence of any number of data-points, each associated with a time-stamp, ordered chronologically.

[38]Spatial time-series is a standard time-series but with a spatial dimension as well.

3. Trajectory[39].

4. Event sequence[40].

- Model architecture, which could be:

1. Transformer-based.

2. Non-Transformer-based[41].

3. Diffusion-based [45] [23][42].

- The nature of the models' pre-training. which could be:

1. Pre-trained LM (Large Model)[43].

2. Self-supervised[44].

3. Fully supervised.

- Capabilities to adapt to a new time-series, which could be:

1. Fine-tuning.

2. Zero-shot learning.

3. Prompt engineering.

4. Tokenization.

In this dissertation, I will only consider a small subset of TSFMs; ones that belong to a class of standard time-series, Transformer-based, Self-supervised (Generative) models

[39]Trajectory is a sequence of time-stamped locations which describe the movement of an object in space.

[40]Event sequence is a chronologically ordered sequence of events within a specific context.

[41]These models are usually MLP, RNN or CNN -bases models.

[42]Diffusion-based models are self-supervised models usually applied in image generation. They are trained by adding gaussian blur to the training-sample in a markov process manner and then applying (usually) CNN-based model to learn how to un-blur the same sample.

[43]These models use a model (usually Large Language Model) which has already been trained on some other type of sequential data, for another task and they just adopt it for time-series. Adoption strategies usually involve changing the way the inputs are tokenized in order to work better with time-series data.

[44]Self-supervised models can be further split into Generative, Contrastive and Hybrid models. Generative models are those which have been trained to reconstruct the input data. They are particularly useful for tasks that involve generating a "missing" part of the input data (such as time-series forecasting). Contrastive models are those that have been trained to distinguish between "similar" and "dissimilar" pairs of data. As such, they are more appropriate for classification tasks (In time-series analysis, they are usually used for anomaly detection). As the name suggests, Hybrid models use mix of these two strategies.

with fine-tuning capability. However, this class of TSFMs is itself very large [31] (in no particular order: PatchTST [38], MOIRAI [57], Lag-Llama [39], TimeSiam [16], Timer [33], TimesFM [13], UniTS [18], TimeGPT-1 [19], Chronos [1], MTSMAE [50]) so the efforts of this dissertation will be focused on two pioneering examples, trained on vast collection of time-series data spanning multiple domains [31]: Lag-Llama and TimeGPT-1.

3. LITERATURE REVIEW

3.1 TimeGPT-1 [19]

TimeGPT-1 is closed-source model - meaning that there is a lot of opacity to model's architecture, parameters and trainng data.

Architecture

TimeGPT-1 has encoder-decoder structure with multiple layers, each having residual connections and layer normalization. It utilizes self-attention mechanism based on the original Transformer [54].

Special capability of TimeGPT-1 is multivariate time-series forecasting - meaning that it is able to take into account "special events" and exogenous features[45] when making predictions on a target time-series. However, in order to take full advantage of this feature, we would have to know for certain the realizations of the exogenous variables in the future which is almost never the case in time-series forecasting[46]. TimeGPT-1 solves this by separately forecasting the exogenous variables into the future and then basing the forecast of the target time-series on its own forecast of the exogenous features.

Training Data

Authors claim TimeGPT-1 has been trained on a data-set containing over 100B data-points - the largest collection of publically availible time-series according to their knowledge. Data comes from the domains of finance, economics, demographics, healthcare, weather, IoT sensor data, energy, web traffic, sales, transport, and banking. Due to diversity of data domains, the training set contains time-series with many different characteristics: seasonalities, cycles, trends, noise, outliers. Having been trained on such a diverse dataset, TimeGPT-1 has very strong robustness and generalization capabilities.

[45] Special events and exogenous features are time-series which are assumed to have some influence on the target time-series. Example of the former being bank holidays if the target time-series if number of flights per day and example of the latter being price of petrol if we are forecasting number of cars on the road.

[46] because we can never predict anything in the future with certainty.

Results

3.2 Lag-Llama [39]

Tokenization

Lag-Llama constructs "lagged features" from the past values of the time-series according to a set of frequency-dependant set of lag indices L_{indices}=1, ..., N. The set of lagged features of a point x_t at timestamp t is then $\{x_{t\text{-}L_{\text{indices}}[i]}$, for all integers i such that $0<i<N+1\}$. Lag-Llama also constructs "date-time features" F which for a point x_t in time t contain information about second-of-the-minute, minute-of-the-hour, ..., month-of-the-year. Final tokenization is done by simply concatenating the date-time features and lagged features into a single vector.

Architecture

Lag-Llama is an open-source, decoder-only TSFM based on LLaMA [53] architecture with reduced number of parameters[47] (2.5m) . Similarly as LLaMA, Lag-Llama utilizes concepts such as RMSnorm [61][48], RoPE [47][49] and SwiGLU [43][50] activation function[37]. On top of N decoder layers, Lag-Llama has a "parametric distribution head". Parametric distribution head is a module which predicts the distribution parameters of the next prediction, given the set distribution. Authors of the paper have chosen student-t distribution for the distribution head, meaning that at inference time, the model predicts the degrees of freedom, mean and scale [46] of the t-distribution from which it randomly draws n samples - idea being that this sample of n predictions gives a probabilistic insight into the model's

[47]Original LLaMA series comes in sizes of 6.7B, 13.0B, 32.5B and 65.2B parameters.

[48]RMSnorm is a layer normalization technique. Zhang and Sennrich (2019) demonstrate that centering component of the LayerNorm technique is dispensable (and computationally expensive) and propose their own layer normalization strategy called RMSnorm which delivers similar benefits (more stable training and better model convergence) but with lower computational cost.

[49]RoPE stands for Rotary Positional Encoding. This is a method for positional encoding. RoPE enables valuable properties, including the flexibility of sequence length, decaying inter-token dependency with increasing relative distances, and the capability of equipping linear self-attention with relative position encoding.

[50]SwiGLU activation function is a combination of swish and GLU activation functions. With beta hyperparameter equal to 1, it is of similar shape as ReLU but completely smooth (continuous) - therefore "behaving better" during model training.

prediction. See Figure 3.1:

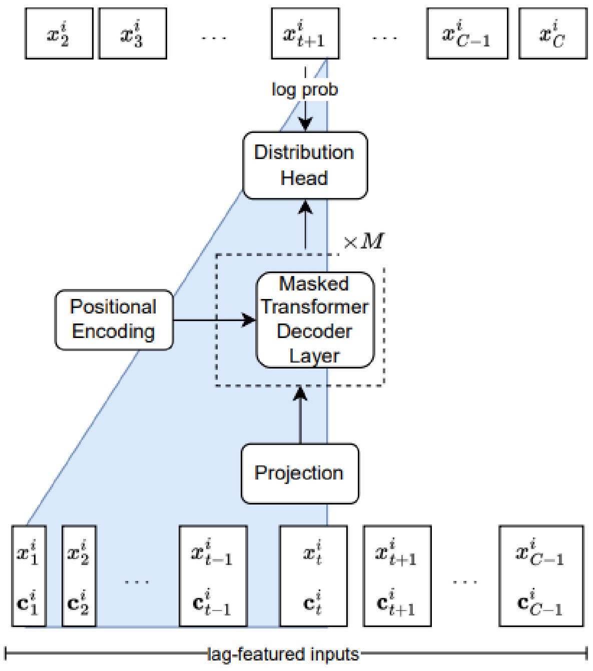

Figure 3.1: Lag-Llama architecture [39]

Training data

Lag-Llama is pre-trained on diverse set of time-series domains such as energy, transportation, economics, nature, air quality and cloud operations.

4. METHODOLOGY AND DATA

This Research aims to answer 3 main questions:

1. How good are TSFMs on financial time-series data?
2. In which cases do they perform better/worse?
3. How to improve TSFMs performance?

Methodology of this research is designed in a way to try best answer these three questions in a scientific, statistically significant manner.

4.1 Time-series prediction evaluation

Time-series predictions are evaluated using the traditional regression metrics due to the continuous nature of the target variables. All regression metrics are based on some variation of the prediction error (D.1.1). From the prediction error, following metrics are derived, and commonly used in time-series prediction tasks in Finance[25]:

- RMSE (Root Mean Square Error)
- MAE (Mean Absolute Error)
- MAPE (Mean Absolute Percentage Error)
- R2

An additional metric that will be used in this research is MDA (Mean Directional Accuracy) (D.1.2). See [12] for Directional accuracy calculation.

Finally, I suggest a new metric called MES (Mean Equal Sign) to be used in scenario where the time-series being predicted is financial returns. For a set of n time-series predictions $\{y_{\text{predicted}}{}^1, y_{\text{predicted}}{}^2, \ldots, y_{\text{predicted}}{}^n\}$ and ground-truth values $\{y_{\text{actual}}{}^1, y_{\text{actual}}{}^2, \ldots, y_{\text{actual}}{}^n\}$, I define MES as mean$\{\text{sign}(y_{\text{predicted}}{}^1 \text{ x } y_{\text{actual}}{}^1), \text{sign}(y_{\text{predicted}}{}^2 \text{ x } y_{\text{actual}}{}^2), \ldots, \text{sign}(y_{\text{predicted}}{}^n \text{ x } y_{\text{actual}}{}^n)\}$. This metric is introduced because MDA alone doesn't fully capture the model's ability to predict direction of underlying asset value movement (see D.1.3).

MDA and MES aould be specially important in the field of Finance as we are not necessarily only interested in predicting the actual values of an asset, but rather we are interested in predicting the direction in which the price will go next time-step.

4.2 Data

This project uses 4 types of financial time-series data of 3 different frequencies (See D.1 and D.2 for information on which specific time-series were used) (See table 4.1):

Frequency	**Type of Data**			
	Stock Index	**Commodity**	**Exchange Rate**	**CHAPS**
Daily	YES	YES	YES	YES
Weekly	YES	YES	YES	YES
Monthly	YES	YES	NO	NO

Table 4.1: Frequency of different types of data

Data Pre-processing

In the field of Finance, there is a concept of "return". Return of an asset A at a certain point in time T: $\mathrm{A_T}$ is calculated as

$$(A_T - A_{T-1})/A_{T-1}$$

i.e. the percentage change in the value of that asset between the points in time T and T-1. If we think about each one of the fore-mentioned time-series as assets, we can represent them as time-series of returns rather than their actual values which is a common practice in financial time-series prediction [25].

Time-series data characteristics

Time-series data exhibits several key characteristics that are essential to understand for effective analysis and forecasting. **Trend** is a long-term increase or decrease in the data. It could take many shapes: none, linear, exponential, polynomial. This research uses

Kendall's Tau coefficient of correlation [28]. Kendall's Tau measures the strength of the relationship between two ordinal variables. If a time-series of length L has statistically significant Kendall Tau correlation (p-value < 0.05) with a monotonically increasing sequence $\{1, 2, ..., L\}$, this is a sign that there is a linear trend present in the data. If the square roots of every value in a time-series have statistically significant Kendall Tau correlation with a monotonically increasing sequence $\{1, 2, ..., L\}$, this is a sign that there is a quadratic trend present in the data. **Seasonality** is the presence of recurring patterns at regular intervals in the data. Chosen method for determining the presence of seasonality is by looking at partial autocorrelation values (see D.2.2 for explanation) in the data and declaring a seasonal pattern present if there is a significant positive or negative partial autocorrelation present. There is no way to determine the significance in this case. Rather, I chose values >0.3 or <-0.3 to be significant. **Stationarity**: A time-series is said to be stationary if the statistical properties such as mean and variance remain constant over time. Stationarity is determined using the Dickey-Fuller test [15]. Other characteristics of time-series are **Noise** and **Volatility** however, they are relatively difficult to classify and are scale-dependant.

4.3 Experiment Design

Benchmark

Calculating the evaluation metrics for time-series prediction is not sufficient to answer whether a time-series prediction model is good or not. The results of the evaluation need to be put into context and compared against a benchmark in order to make a judgment on how good they are. Chosen benchmark models are: autoARIMA [26] (D.3.1), Naive Simple Autoregressor (D.3.2) and Meta's Prophet model [51].

Time Series Cross Validation (TSCV)

K-fold cross-validation (CV) is a statistical technique used to evaluate the performance of a model by dividing the dataset into K equally sized subsets, or "folds." The model is trained on K-1 "train" folds and tested on the remaining "validation" fold, and this process repeats K times, with each fold serving as the test set once. The results are then averaged to provide a more reliable estimate of model performance. It is important because it helps prevent overfitting by providing multiple evaluations of the model's ability to generalize to unseen data. However, in time-series prediction, where data points are temporally dependent, standard K-fold cross-validation cannot be applied directly since randomly partitioning the data breaks the time order and dependencies. To address this, K-fold cross-validation for time-series is modified by partitioning the data in a way that respects temporal structure (See D.3.3 for details). This allows for a valid evaluation of time-series models while still leveraging the benefits of cross-validation.

5. RESULTS

5.1 Section

Sub-Section

REFERENCES

[1] A. F. Ansari, L. Stella, C. Turkmen, X. Zhang, P. Mercado, H. Shen, O. Shchur, S. S. Rangapuram, S. P. Arango, S. Kapoor, et al. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024.

[2] V. Assimakopoulos and K. Nikolopoulos. The theta model: a decomposition approach to forecasting. *International journal of forecasting*, 16(4):521–530, 2000.

[3] J. Ba. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[4] D. Bahdanau. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[5] G. Box and G. Jenkins. *Time Series Analysis: Forecasting and Control*. Holden-Day series in time series analysis and digital processing. Holden-Day, 1970.

[6] R. G. Brown. *Exponential smoothing for predicting demand*. Little, 1956.

[7] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[8] 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*, pages 785–794, 2016.

[9] K. Cho. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[10] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

[11] R. B. Cleveland, W. S. Cleveland, J. E. McRae, I. Terpenning, et al. Stl: A seasonal-trend decomposition. *J. off. Stat*, 6(1):3–73, 1990.

[12] M. Costantini, J. C. Cuaresma, and J. Hlouskova. Forecasting errors, directional accuracy and profitability of currency trading: The case of eur/usd exchange rate. *Journal of Forecasting*, 35(7):652–668, 2016.

[13] A. Das, W. Kong, R. Sen, and Y. Zhou. A decoder-only foundation model for time-series forecasting. *arXiv preprint arXiv:2310.10688*, 2023.

[14] J. Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[15] D. A. Dickey and W. A. Fuller. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American statistical association*, 74(366a):427–431, 1979.

[16] J. Dong, H. Wu, Y. Wang, Y. Qiu, L. Zhang, J. Wang, and M. Long. Timesiam: A pre-training framework for siamese time-series modeling. *arXiv preprint arXiv:2402.02475*, 2024.

[17] A. Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[18] S. Gao, T. Koker, O. Queen, T. Hartvigsen, T. Tsiligkaridis, and M. Zitnik. Units: Building a unified time series model. *arXiv preprint arXiv:2403.00131*, 2024.

[19] A. Garza and M. Mergenthaler-Canseco. Timegpt-1. *arXiv preprint arXiv:2310.03589*, 2023.

[20] R. Gençay and M. Qi. Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging. *IEEE transactions on neural networks*, 12(4):726–734, 2001.

[21] J. Hasanhodzic and A. W. Lo. Can hedge-fund returns be replicated?: The linear case. *The Linear Case (August 16, 2006)*, 2006.

[22] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[23] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[24] S. Hochreiter. Long short-term memory. *Neural Computation MIT-Press*, 1997.

[25] Z. Hu, Y. Zhao, and M. Khushi. A survey of forex and stock price prediction using deep learning. *Applied System Innovation*, 4(1):9, 2021.

[26] R. J. Hyndman and Y. Khandakar. Automatic time series forecasting: the forecast package for r. *Journal of statistical software*, 27:1–22, 2008.

[27] R. J. Hyndman, A. B. Koehler, R. D. Snyder, and S. Grose. A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of forecasting*, 18(3):439–454, 2002.

[28] M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1-2):81–93, 1938.

[29] Z. Lan. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[30] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems*, 32, 2019.

[31] Y. Liang, H. Wen, Y. Nie, Y. Jiang, M. Jin, D. Song, S. Pan, and Q. Wen. Foundation models for time series analysis: A tutorial and survey. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6555–6565, 2024.

[32] Y. Liu. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[33] Y. Liu, H. Zhang, C. Li, X. Huang, J. Wang, and M. Long. Timer: Transformers for time series analysis at scale. *arXiv preprint arXiv:2402.02368*, 2024.

[34] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[35] F. J. Massey Jr. The kolmogorov-smirnov test for goodness of fit. *Journal of the American statistical Association*, 46(253):68–78, 1951.

[36] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943.

[37] J. A. Miller, M. Aldosari, F. Saeed, N. H. Barna, S. Rana, I. B. Arpinar, and N. Liu. A survey of deep learning and foundation models for time series forecasting. *arXiv preprint arXiv:2401.13912*, 2024.

[38] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

[39] K. Rasul, A. Ashok, A. R. Williams, A. Khorasani, G. Adamopoulos, R. Bhagwatkar, M. Biloš, H. Ghonia, N. V. Hassen, A. Schneider, et al. Lag-llama: Towards foundation models for time series forecasting. *arXiv preprint arXiv:2310.08278*, 2023.

[40] A. Roberts, C. Raffel, K. Lee, M. Matena, N. Shazeer, P. J. Liu, S. Narang, W. Li, and Y. Zhou. Exploring the limits of transfer learning with a unified text-to-text transformer. *Google, Tech. Rep.*, 2019.

[41] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[42] V. Sanh. Distilbert, a distilled version of bert: Smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[43] N. Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[44] E. SLUTSKY. ön a case where the law of large numbers applies to mutually dependent quantities. the extension of a theorem by magoichirô watanabe. *Tohoku Mathematical Journal, First Series*, 28:26–32, 1927.

[45] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

[46] Student. The probable error of a mean. *Biometrika*, pages 1–25, 1908.

[47] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[48] I. Sutskever. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*, 2014.

[49] I. Svetunkov, N. Kourentzes, and J. K. Ord. Complex exponential smoothing. *Naval Research Logistics (NRL)*, 69(8):1108–1123, 2022.

[50] P. Tang and X. Zhang. Mtsmae: Masked autoencoders for multivariate time-series forecasting. In *2022 IEEE 34th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 982–989. IEEE, 2022.

[51] S. J. Taylor and B. Letham. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018.

[52] P. Temin. Price behavior in ancient babylon. *Explorations in Economic History*, 39(1):46–60, 2002.

[53] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[54] A. Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[55] Q. Wen, J. Gao, X. Song, L. Sun, H. Xu, and S. Zhu. Robuststl: A robust seasonal-trend decomposition algorithm for long time series. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 5409–5416, 2019.

[56] H. Wold. *A study in the analysis of stationary time series.* PhD thesis, Almqvist & Wiksell, 1938.

[57] G. Woo, C. Liu, A. Kumar, C. Xiong, S. Savarese, and D. Sahoo. Unified training of universal time series forecasting transformers. *arXiv preprint arXiv:2402.02592*, 2024.

[58] L. Xue. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*, 2020.

[59] G. U. Yule. On the time-correlation problem, with especial reference to the variate-difference correlation method. *Journal of the Royal Statistical Society*, 84(4):497–537, 1921.

[60] G. U. Yule. Vii. on a method of investigating periodicities disturbed series, with special reference to wolfer's sunspot numbers. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 226(636-646):267–298, 1927.

[61] B. Zhang and R. Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

[62] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.

[63] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International conference on machine learning*, pages 27268–27286. PMLR, 2022.

A. HISTORY OF TIME-SERIES FORECASTING

A.1 Brief History of time-series prediction in Finance

When the agricultural revolution took place (circa 10,000 BC), for the first time in history, humans have started producing more food and items than they required just to survive. This gave birth to the concept of "storing" things and eventually trading them for other things which were perceived to be of greater value. When sufficient amount of people started transacting goods in this manner, the concept of "market price" took hold. It did not take long before humans realized that they didn't have to work in the field the whole day to amass wealth, but that they could do it through trade alone. They could buy items from one person at a low price and sell to another at a high price (arbitrage) or buy it now at a low price and hopefully sell later at a high price (asset appreciation). Opportunities to exploit the first option became more rare and less profitable as more and more people started engaging in trade and eliminating these arbitrage opportunities. However, the 2nd option has captured the imagination of traders ever since 10,000BC as no one seemed to be able to tell with certainty whether the price of a good would indeed go up in the future or not. As humans learned to write and keep records, they starter noting down the prices of goods that were being traded along with the time when those prices were prevailing. We have records of ancient Babylonians keping historic records of enf-of-month barley, dates, cuscuta, sesame, cardamom and wool prices on clay tablets spanning some 400 years [21]. Investigating these prices, we can see that they follow a random walk pattern with exogenous shocks (such as the death of Alexander the Great) [52] and they seem almost impossible to predict with a naked eye, however, it is not impossible that someone might have tried and unknowingly became the first person in history to work on time-series prediction in the field of Finance.

Over the centuries, methods to solve the problem of time-series prediction in finance seem to have developed universally across many different cultures. Christopher Kurz, a 16th century merchant from Antwerp, thought about the idea of "trend" and "seasonality" in agricultural prices and claimed he could predict prices 20 days in advance. In 18th century Japan, Munehisa Honma - a man dubbed "God of the markets" by his contemporaries, developed a principle stating that when prices become extremely high, they must come down again which we know today to be the principle of "mean reversion". In the late 19th century, Charles Dow, co-founder of Wall Street Journal and Dow Jones Company[51] popularized and coined the term "technical analysis" (a collection of methods that employ math and statistics to predict future prices) which later evolved into quantitative analysis - which is used today in institutions engaged in future price prediction.

A.2 Brief History of modern time-series prediction methods

Statistical models

Although the field of future price prediction has moved far beyond simply considering just the historical prices[52], there has been notable work on the general field of univariate[53] time-series prediction. It is hard to pinpoint the exact start of modern time-series prediction, but I think a good candidate would be Udny Yule's[54] 1927 [60] paper which is considered first to have used simple autoregressor model (AR)[55] to predict Wolfer's Sunspot numbers[56]. Even though Slutsky (1927) [44] and Udny (1921) [59] demonstrated the use of Moving Average as a way to show trend and cyclicality in time-series data, Herman Wold's work (1938)[57] [56] indirectly invented the Auto Regressive Moving Average (ARMA) model[58]. ARMA model is theoretically sound if the time-series being predicted is stationary[59], which is not the case with all time-series. In 1970, George Box and Gwilym Jenkins introduced probably the most famous time-series prediction model of all times: ARIMA [5]. ARIMA(p, i, q) model is an ARMA(p, q) model that employs the method of

[51]Dow Jones Company is the publisher of the prominent DOWJ stock market index

[52]Quantitative analysis now-adays considers wide palette of data and information

[53]Univariate time-series prediction is a practice of using only the past values of a certain time-series in order to predict its future value(s).

[54]Udny studied at UCL under a well-known (and controversial) professor Karl Pearson - father of mathematical statistics and generally one of the most well-known personas in statistics.

[55]An AR(p) time-series prediction model predicts future values based on a linear combination of the previous p observations in the series, assuming that the current value depends on its own past values and a stochastic error term.

[56]Certain time-series data from the field of Astronomy.

[57]Wold's Decomposition Theorem states that any stationary time series can be decomposed into a deterministic part (AR component) and a stochastic part (MA component)

[58]The MA(q) part of the ARMA(p, q) model forecasts future values by modeling the current value as a linear combination of the past q error terms (shocks) and a stochastic error term.

[59]A time series is said to be stationary if its statistical properties, such as mean, variance, and autocorrelation, remain constant over time

differencing[60] on the date before training and prediction.

Another direction of innovation in the field of univariate time-series prediction was towards "Exponential Smoothing" (ES). Concept of ES originates from the work of Brown 1956 [6] and has been thoroughly built upon since. Most notable invention was by Hyndman et al. in 2002 [27] - implementing exponential smoothing within a state-space framework, giving us the ETS[61] model. Most recent culmination in the ES family of models was by Ivan Svetunkov in 2022 [49] with the CES model[62]. Final honorable mention among the time-series prediction models of "statistical" nature is the "Theta" model. Vassilis Assimakopoulos, and Konstantinos Nikolopoulos (2000) [2] Introduced the Theta model which decomposes a time-series into its' cyclical and trend components which are then forecasted separately and finally combined[63].

Machine Learning (ML) models

As the hardware technology advanced in the 21st century, so did the time-series models as the scientists have finally gotten machines with enough computational power to run more complex algorithms. The most famous of these algorithms being the Artificial Neural Network (ANN). Although the idea (in its primitive form[64]) dates back all the way to McCulloch and Pitts (1943) [36], ANN first real ancestor was the "Perceptron"[65], invented by a psychologist - Frank Rosenblatt in 1958 [41]. The perceptron had ability to take continuous inputs and proper learning algorithm - known even to this day as the "Rosenblatt algorithm". Through the rest of the 20th century, there have been general

[60]Differencing is a technique in time-series analysis used to transform a non-stationary series into a stationary one by deducting the preceding observation from each observation. The order "i" of the differencing refers to the ammount of times this procedure is applied to a time-series.

[61]ETS stands for Error Trend Seasonality. ETS is a family of 18 models. ETS model can account for errors being additive or multiplicative, and trend or seasonality being none, additive or multiplicative. ETS models with no multiplicative errors or seasonality have their equivalent withing the ARIMA family of models

[62]CES stands for Complex Exponential Smoothing model. CES method models the time-series as a series of complex numbers where the real part is the prediction and imaginary parts are errors. CES models can also be expressed in state-space form to adress seasonality. CES advantages are that it is flexible and has been empirically shown to perform well.

[63]These components are called "theta lines". Theta model can use exponential smoothing or an AR model to predict these two lines. Predictions are finally combined using weighted average where weights could be equal or adjusted to favor either the trend line or the cycle line.

[64]The original vision of an ANN was based on simple binary inputs with fixed thresholds activation - a far cry from modern ANNs

[65]Perceptron can be thought of as a single-neuron ANN. ANNs are comprised of one or more neuron layers, each layer containing one or more neurons.

periods of great interest and disinterest in ANNs for many reasons, but ANNs finally came on their own in the 21st century when computers become powerful enough to be able to train larger versions of these models[66] in reasonable time[67]. Naturally it was not long before people realized ANN-type models can also be used for time-series prediction[68] which led to many variations of the ANN architecture to be invented. Bayesian regularization ANN has been used in Finance [20]. Another successful variation of ANN is the Radial Basis Function ANN (RBFANN).

Another

Chen and Guestrin (2016) [8] introduced the Xgboost model which has

Deep Learning models

[66]Size of an ANN matters when dealing with very complex tasks (tasks with many input features).

[67]Another reason for explosion in popularity was the beginning of the internet era and sudden availibility of large ammounts of data which are needed for ANN training

[68]It is important to note that univariate time-series prediction ANNs are simply regression ANNs where lags of the data we're trying to predict are explanatory features (inputs) to the model

B. TIMEGPT-1

C. LAG-LLAMA

D. METHODOLOGY

D.1 Time-series prediction

Prediction error

Prediction error is the difference between the value a model predicts and the actual (ground-truth) value.

MDA

MDA is time-series specific. Since time-series data is inherently ordered (as opposed to regular regression data), we can measure the direction in which a time-series is moving at each time-step. Hence we can compare the direction of movement of our predicted time-series against the actual time-series and we can calculate in what percentage of cases did the two time-series move in the same direction (up or down).

MES

Reason why MDA doesn't fully capture the model's ability to predict direction of underlying asset value movement is because the models are working with returns time-series, therefore MDA measures whether the model accurately predicts the directional change of returns and not the actual underlying values. Imagine scenario where $\mathrm{y_{actual}}^{\mathrm{T}}$ = 1%, $\mathrm{y_{actual}}^{\mathrm{T+1}}$ = 0.5% and $\mathrm{y_{predicted}}^{\mathrm{T}}$ = 0.7%, $\mathrm{y_{predicted}}^{\mathrm{T+1}}$ = -0.5%. In this case MDA would account this as a correct directional guess as the model predicted y would go down in period T+1, and y actually did go down at T+1. However, if we inspect this case in greater detail, at the moment T+1, the value of the underlying asset which y represents, went up at T+1, as the return in that period is still positive, however the model actually implicitly predicted the price to go down. This is a nuance which MDA doesn't capture, but MES does.

D.2 Data

Choice of Data

Stock Index	**Commodity**	**Exchange Rate**	**CHAPS**[69]
S&P 500	WTI	USD/GBP	Aggregate
FTSE 100			Delayable
DOWJ			Social
NASDAQ			Staple
			Work-related

Table D.1: Frequency of different types of data

Daily	**Weekly**	**Monthly**
2018-01-01 to 2020-01-01	2015-01-01 to 2020-01-01	1987-01-01 to 2024-01-01
2020-01-01 to 2022-01-01	2017-01-01 to 2022-01-01	
2022-01-01 to 2024-01-01	2019-01-01 to 2024-01-01	

Table D.2: Period

Exceptions:

- Daily CHAPS data not available 2018-01-01 to 2020-01-01.
- Weekly CHAPS data not available 2017-01-01 to 2022-01-01 and 2015-01-01 to 2020-01-01.
- Weekly CHAPS data used was available only 2020-01-01 to 2024-01-01.
- Monthly CHAPS and Exchange rate data was not available in sufficient quantity to conduct experiment.

Partial autocorrelation

Partial autocorrelation measures the correlation between a time series and its lagged values, while controlling for the influence of intermediate lags. In contrast, autocorrelation simply measures the correlation between the time series and its lagged versions without accounting for the effects of other lags. Partial autocorrelation provides a clearer picture of the direct relationship between observations and their lagged counterparts by isolating the specific influence of each lag. This makes it particularly useful for detecting seasonality in time-series data because it helps identify the direct impact of observations at seasonal intervals. By highlighting the most relevant lags for seasonality, partial autocorrelation allows for a more precise analysis of recurring patterns, which is harder to achieve using autocorrelation alone.

D.3 Experiment design

ARIMA

Originally, this package is for R. I used the equivalent python implementation: https://github.com/alkaline-ml/pmdarima.

Naive Simple Autoregressor (NSA)

Given a training time-series sequence of length n: T = {y_1, y_2, ..., y_n}, and a prediction horizon of length h, the prediction of NSA will be: P = {y_n repeated h times}, i.e. NSA predicts the future value(s) the same as the last value.

Time Series Cross Validation

Say we have a time-series *S* of length *l*: S = {y_1, y_2, ..., y_l}, we specified the length of the train sequence to be *n*, and the length of prediction horizon to be *h*. K-fold TSCV method will sample *K* time-series {F_1, F_2, ..., F_K} from S where each fold F_i (1 <= i <= K for all integer i) consists of the "train" and "validation" parts. Fold F_i^j which starts from timepoint *j* is defined as F_i^j = {y_j, y_{j+1}, ..., y_{j+n-1}, y_{j+n}, ..., $y_{j+n+h-1}$}. The model is trained on {y_j, y_{j+1}, ..., y_{j+n-1}} and validated on {y_{j+n}, ..., $y_{j+n+h-1}$} (Size constrains being *K* <= (*l*-textitn)/textith). This experiment uses the version of TSCV where *l* is fixed which is called "rolling window" TSCV. This experiment employs the following fold sampling strategy: say we have two additional cross-validation parameters: *f* and *r*. Folds we sample from *S* are split into *r* groups: {G_1, G_2, ..., G_r} such that all folds within a group are sequential i.e. the start-point of the following fold is the time-point right after the start-point of the preceding fold), each group containing *f* folds. The first group of folds G_1 is always $F_1{}^1$, $F_2{}^2$, ..., $F_f{}^f$ (i.e. first *f* folds are folds with start-points 1, 2, ..., *f* respectively). This sampling procedure is repeated *r* times in a way that the time-distance between the first fold of G_k and the first fold of G_{k+1} is equal to the time-distance between G_{k+1} and G_{k+2} for all $1<=k<=r$-2, and so that the distance between two consecutive groups is maximised subject to constraint $rk <= (l\text{-}n)/h$.

Benchmark