POLITECNICO DI TORINO

Master's Degree in Data Science and Engineering

Master's Degree Thesis

Explaining contrastive models for financial time series forecasting

Supervisors

Prof. Luca CAGLIERO

Prof. Jacopo FIOR

Candidate

Stefano GALVAGNO

JULY 2023

Abstract

This thesis work is carried out within the context of financial time series forecasting, a subject that has undergone innovations over the years from the point of view of tools. Specifically, the task consists in predicting future values exploiting historical and current stock prices. For years the most common method used by traders and analysts have been technical indicators calculated from time series, but lately Machine Learning is increasingly being exploited. However, these approaches, despite being top performers, suffer from the problem of non-explainability. The goal of this thesis is to make post hoc explainability of the constrastive models to identify statistical features correlated with their predictions. In particular, we generated deep representations of financial time series by exploiting Contrastive Learning frameworks, we explained them by means of technical indicators (computed with respect to the initial time series) and we compared performances of the two approaches (contrastive vs technical indicators) on the same downstream task. In order to obtain the aforementioned explainability, we used an ensemble model that receives as input the time series having as features the technical indicators and the results of a clustering algorithm, applied to contrastive representations, as labels. Next, the most important features for label (cluster) prediction are filtered by a threshold. The downstream task is the forecasting of future values of the time series with 3-, 5-, 7-days horizons and is applied both to the latent representations and to the filtered technical indicators. The goal is to verify the capability of light-weighted models (i.e. simpler models) to achieve comparable results with respect to the ones of contrastive models.

Table of Contents

List of Tables

List of Figures

Acronyms

AI
Artificial Intelligence

ML
Machine Learning

DL
Deep Learning

MA
Moving average

NN
Neural network

ANN
Artificial neural network

CL
Contrastive Learning

MLP
Multi layer perceptron

AR
Autoregressive

fc
Fully connected

NDX

NASDAQ 100

SMA

Simple moving average

MACD

Moving average convergence divergence

RSI

Relative strength index

OBV

On balance volume

BB

Bollinger Bands

ADX

Average directional index

Chapter 1

Introduction

Time series represent a fundamental typology of data in a multitude of fields, ranging from medicine to industry. The advent of Big Data has made possible to collect this information, sometimes with very fine granularity, and consequently to use it for a wide variety of purposes. Among the many applications of time series there are forecasting, anomaly detection, trend and pattern analysis over time, signal processing, predictive maintenance and health monitoring, to name a few. Thus, these are not only applications that were historically done manually by domain experts but also complex operations that cannot be done in a reasonable time by a human being.

One of the industries most interested in time series is finance. Before Machine Learning existed, professional traders were only making forecasts based on technical indicators calculated from time series. This method is actually still used today but is risky and unsuccessful in many contexts where the trend is disrupted by external agents. These are, in particular, events of a different nature, such as the outbreak of a pandemic, the nefarious words of a CEO during an earnings call or a Tweet from an exceedingly influential person.

This is why more advanced approaches have been sought over the years. An early step in this direction was the introduction of sentiment, extracted mainly from social media. Instead, in recent years, researchers have concentrated their efforts in the intent to create representations of time series. These so-called embeddings are the output of complex Deep Learning models, projections into a latent space of a certain dimensionality. As mentioned before, in the field of Natural Language Processing (NLP), this approach has already been shown to be definitely effective for the representation of words, sentences, etc.

However, Deep learning models, despite being top performers, suffer from the problem of non-explainability, i.e., the inability to understand how a given result is produced. Therefore, in various contexts, the question arises whether it is worth using this approach, since it also requires a considerable amount of training time.

In this thesis work, a comparison is presented between embeddings generated through different Contrastive Learning approaches and features composed of technical indicators. The comparison is quantified using an error metric collected at the end of the same downstream task, specifically, forecasting.
The ultimate goal of this thesis is to make post hoc explainability of the constrastive models to identify statistical features correlated with their predictions.
The previously mentioned Contrastive Learning is a self-supervised learning method that is designed to overcome the lack of large volumes of labeled data. Indeed, there are many contexts in which the labeling phase would be too time consuming or even be impossible. More specifically, the method does not require labels but it is based on the construction of positive (similar) and negative (dissimilar) pairs which should be placed, respectively, near and far apart in an embedding space, by means of an appropriate loss. The various architectures differ in the criteria by which positive and negative samples are created and in the objective function used to incentivize their placement in the embedding space.

The work begins by training some state-of-the-art models: CoST, TNC, TS2Vec and Time2State, architectures that have in common the presence of an encoder that creates the latent representations intended for a downstream task. As mentioned earlier, these are models based on Contrastive Learning, therefore, they do not make use of manually generated labels. Rather, they rely on statistical assumptions to distinguish different instances of the time series and thus create appropriate representations.
The second step is to group the generated deep representations together with a certain criterion. This is done by a clustering algorithm, therefore, once again, by a non-supervised method. The experiments were conducted by choosing different values of K, number of clusters, since there is no a priori method for determining which number is appropriate. The resulting related clusters are used as labels for the technical indicators directly computed on the initial time series. From this, through a Boosting algorithm, the most relevant features are extracted and filtered by means of a threshold. These are then used for the same downstream task considered with the latent representations, namely the prediction of the same future values of the time series, with respect to different horizons (3, 5 and 7 days). In other words, the goal is to create light-weighted models trying to explain the criterion by which the deep embeddings were created and to evaluate their ability to obtain results comparable to those of contrastive models. The a posteriori gap assessment allows us to determine whether, and if so in which contexts, complex Deep Learning models can be replaced with simpler models.

The experiments were conducted on three datasets: first, a univariate dataset comprising the Nasdaq index only; second, a multivariate dataset consisting of the

prices of some stocks that compose the index itself; third, a multivariate dataset including both the stocks and the index. The effectiveness of the method has been tested with respect to three different markets conditions: bullish, bearish and mixed, providing a comprehensive overview of all possible case scenarios that can occur in the trading world.
There are many results obtained from our experiments; the most relevant ones are reported below.
First, we observed that the convenience of contrastive models decreases as the time horizon increases. In addition, horizon growth is also associated with smaller differences in performance between datasets.
Second, Time2State was the best overall model on 3- and 5-day horizons, aligning with the other models on 7 days.
Third, top performers also included models based on technical indicators, specifically related to 7-day time horizons.
Finally, the real convenience in using contrastive methods was observed during 2020, a decidedly pathological (mixed) market due to the pandemic outbreak.

Chapter 2

Machine Learning Fundamentals

2.1 Machine Learning vs Deep Learning

Since Machine Learning (ML) and Deep Learning (DL) are often used to refer to the same concept, it is crucial to comprehend the distinctions between them. Artificial intelligence includes the subfields of ML, DL, and Neural Networks (NN). However, NNs are a subdivision of ML, and DL is a subset of NNs.

Deep Learning and Machine Learning differ in terms of their learning approaches. Labeled datasets can be used to provide guidance to "deep" Machine Learning algorithms, but they are not strictly necessary. DL can take unstructured data in its original form (e.g., images) and autonomously identify the distinguishing features that separate different data categories. This reduces the required human involvement and allows for the utilization of bigger amounts of data. L. Fridman defined it as *"scalable machine learning"*, during an MIT lecture [1]. In classical ML, human intervention is greater and human-generated features (usually by domain experts) are relied on to discern instances that belong to different classes. Artificial neural networks (ANNs) consist of several layers including an input, an output and intermediate layers. Each node, or neuron, is connected to another and is characterized by a weight and a threshold. If the latter is exceeded by the output of the node, the neuron is activated and begins to transfer information to the next layer, otherwise it is inhibited.

The "deep" in Deep Learning simply refers to the amount of intermediate layers in the model. In particular, we can consider "deep" an architecture consisting of at least three layers. A Neural Network with less than four layers, including the input and output layer, is referred to as a simple NN.

The most related fields to DL and NNs are computer vision, speech recognition

and natural language processing. [2]

2.2 How it works

According to [3] the learning process can be subdivided into three main steps:

- Decision: ML algorithms are typically used to predict or classify. The algorithm will generate an estimate regarding a pattern within the data using certain input, which can either have labels or not.
- Loss function: it evaluates the prediction of the algorithm. If there are known examples, a loss function can compare them to gauge how well the model performs.
- Optimization: if the algorithm fits the data points in the training set better, the weights are adjusted to minimize the disparity between the known sample and the output of the model. The algorithm iteratively repeats this process until a level of performance, considered acceptable, is reached.

In addition, the concept of Machine Learning includes several categories. The most important ones are described in the following sections.

2.3 Supervised learning

Supervised learning is the most common and intuitive approach: an algorithm learns to make predictions or decisions by analyzing a labeled training dataset [4]. Specifically, the algorithm iteratively receives as input pairs (X,y), where X is the data made of features and y is the expected result (the label), and learns a function f such that y=f(X). In other words, it learns a function that maps the inputs to the desired outputs. The algorithm produces a certain result based on the input data and this outcome is compared with the correct expected result. In case they do not match, the algorithm receives a penalty. When an acceptable performance is achieved, the training ends and the model is likely able to generate a correct result even when it receives new data as input. This type of learning is called supervised learning because the training phase takes place in a supervised manner, the correct answers are known, and the algorithm is corrected whenever it makes an error. The ultimate goal is to obtain a model capable of satisfactory results even with data never seen in the training phase. Some of the most common applications are image recognition, fraud detection, financial forecasting, predictive maintenance, and medical diagnosis. The most common methods used in supervised learning include ANNs, logistic regression, linear regression, support vector machine (SVM) and random forest [5].

2.4 Unsupervised learning

Unsupervised learning contrasts with the approach described above since it does not rely on labels. Specifically, the dataset consisting solely of features is provided. In other words, there is no prior knowledge derived from labeling, often resulting from the intervention of a domain expert. The algorithm is tasked with discovering hidden patterns, relationships or structures present in the data itself.
Depending on the task, data can by organized by unsupervised models in several ways [6]:

- Clustering: grouping points on the basis of their similarity, finding a pattern or structure in unlabeled data. There exist both bottom up and top down approaches [7].

- Association: identify hidden insights between different objects by exploiting relationships often in the form of rules or frequent itemsets [8]. The most common application is Market Basket Analysis used to fill an online shopping cart with items frequently bought together.

- Anomaly detection: finding outliers, namely, observations that deviate significantly from the normal patterns of a variable. An example of a task is fraud or intrusion detection [9].

- Autoencoders: compress into a lower-dimensional space the representation in an input using a NN and then reconstruct the output from these embeddings. To accomplish this, the architecture exploits an encoder and a decoder [10].

Unsupervised learning is very useful because it allows us not to rely on a labeling method that is often influenced by human experience.

2.5 Semi-supervised learning

Semi-supervised learning represents a middle ground between the previous two approaches. In the training phase, a reduced labeled dataset is exploited to guide the classification and feature extraction of a large, unlabeled dataset. This type of algorithm allows for the discovery of latent patterns of unlabeled samples, mitigating the problem of having large volumes of labeled data [11].
One application is the analysis of medical images, which are lacking in terms of labeled datasets because it would require a not insignificant amount of effort from a time perspective.

2.6 Self-supervised learning

Self-supervised learning represents a hybrid approach between unsupervised and supervised learning that stems from the need to address the limitations inherent in supervised learning. In practice, it is not possible to label everything in the world. In addition, artificial intelligence systems are not yet able to exploit common sense, an ability that characterizes humans. In particular, humans during their childhood learn to recognize things. Whether it is a dog standing or lying down, the human brain is able to classify it correctly. This is done through prior knowledge about how the world works. Artificial intelligence, to achieve such accuracy, must be trained on a prohibitive amount of data, and this is often not possible [12]. Self-supervised learning is one of the most useful paradigms to overcome these problems. Specifically, the model creates its own labels in an unsupervised manner, exploiting the information inherent in the data itself. These representations can then be used for a downstream task. In other words, self-supervised learning algorithms are composed of two phases: first, task-independent features are extracted from the data, in a second phase, the knowledge gained in the first phase is transferred to carry out the main task [13]. Some of the areas that have exploited this approach the most are natural language processing (Word2Vec, GloVE, fastText, BERT etc.), image processing and time series forecasting, as we will see later. Self-supervised learning methods fall into three types:

- Generative methods
- Adversarial methods
- Contrastive methods

Methods exploiting the latter category were used within this thesis.

2.7 eXplainable AI

One of the major problems of DL is non-explainability. In other words, it is not possible a posteriori to determine what the reasons behind a given output are. The premise behind the use of AI is the fact that it will not completely replace the human being. With this in mind, it is crucial that model outputs can provide understandable feedback.
One of the most well-known cases where this issue has been raised is Amazon's recruitment system [14]. Specifically, in 2014, the use of their "black-box" models did not allow for an understanding of the criteria by which certain individuals were discarded. It emerged that the algorithm visibly favored men, and since there were policies related to gender equity, it became critical that the systems were able to

provide an explanation.
Another context is autonomous driving in which "black-box" models make decisions but it is unclear what parameters influence those choices. From the end users' point of view, being aware of these aspects would allow for a greater sense of safety and confidence, as well as being able to manually foil dangerous situations.
This need, also generalized to other cases, gave rise to eXplainable AI (XAI), a discipline that aims to build transparent artificial intelligence models to provide insights and justification to human users [15].
In the specific case of finance, the area in which this thesis is developed, it is crucial to understand what parameters influence trend or price prediction. Human beings, understandably, prefer to invest their money in something understandable, despite not having solid expertise in the subject [16].
An attempt in this direction will therefore be made in this thesis.

Chapter 3

Contrastive Learning

3.1 Fundamentals

Contrastive Learning (CL) is a Machine Learning technique that allows learning a representation from data without relying on human-generated labels. This is accomplished with the idea of grouping similar entities together and separating them from those that differ. It is a very versatile method because it can also be used in contexts where the downstream task is supervised, such as a classification problem. In particular, it can be used to learn a latent representation without a label or within a two-step split approach in which, with respect to the first case, fine-tuning occurs later in a supervised manner with respect to the problem that one actually wants to solve.
More specifically, two objects representing the same thing in the world should lie close together in the low dimensional feature space while different entities should be pushed away from each other [17]. Taking a concrete example, features generated by two images of cars should be close in the feature space, far apart from those of a person. These pairs are called positive and negative, respectively, and can be generated in different ways.
If operating in an entirely unsupervised context, the so-called augmentations can be used. They consist of selecting a given item, such as an image, and then generating two variations of it. Some common operations in the images domain are represented in 3.1. These two variations of the anchor (the starting item) represent a positive pair and will have to lie close together in the feature space. The same principle applies to data with a temporal order: if two observations were recorded at close instants then they are likely to be more similar than others collected much later.

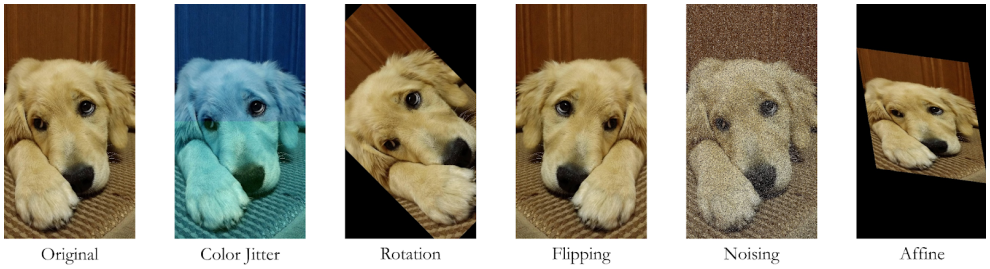


Figure 3.1: Examples of augmentations [18]

Training clearly goes through evaluation by a loss function.
The most common ones are the *Contrastive loss* and *Triplet loss*.

3.1.1 Contrastive Loss

Contrastive Loss is one of the first training objective functions in the CL context [19].

$$L_{contrastive} = \frac{1}{2N} \sum_{i=1}^{N} \Big(y \cdot d^2 + (1-y) \cdot \max(0, \epsilon - d^2) \Big)$$

where

- N represents the total amount of observations within a batch.
- d is the distance between the pairs of data points.
- y is the binary label indicating whether the pair of data points is similar (y=1) or dissimilar (y=0).
- epsilon is the margin, a hyperparameter that determines the desired separation margin between similar and dissimilar pairs.

In other words, this loss on the one hand penalizes, proportionally to the (Euclidean) distance, similar samples for being separated, and on the other hand penalizes negative samples for being at a shorter distance than the established margin.

3.1.2 Triplet loss

Triplet loss was proposed in the FaceNet paper [20]. It was used to learn face recognition of the same person at different poses and angles.

$$L_{triplet} = \sum_{\forall x} \max \Big(0, \|\theta(s_a) - \theta(s_+)\|_2^2 - \|\theta(s_a) - \theta(s_-)\|_2^2 + \epsilon \Big)$$

where:

- s_a, s_+, s_- represent, respectively, an anchor, a positive data point and a negative data point.

- $\theta(s)$ denotes the embedding (feature vector) of data point s.

- The L2 norm between two vectors is represented by the || || symbol. The distance between the anchor and the positive is denoted by the expression $||\theta(s_a) - \theta(s_+)||$, whereas the distance separating the anchor and the negative is denoted by the term $||\theta(s_a) - \theta(s_-)||$.

- In this case as well, a hyperparameter called ϵ, the margin, controls how far apart the positive and negative pairs should be from one another.

The loss is null if the disparity between the anchor-positive and anchor-negative distances is less than the margin. Otherwise, the loss becomes positive, encouraging the network to learn embeddings that satisfy the desired separation.

3.2 Settings

Contrastive Learning can be used with different settings.

- Supervised: knowing that two items represent the same thing (by means of a label) then they will have to be close together in space. In other words, pairs are formed by relying on labels.

- Self-supervised: this is the most common case. As mentioned above, the augmentations system is used to form the pairs. One can also use the method as a way to initialize the weights of a network that will be trained for a downstream task such as image classification, semantic segmentation etc. In the latter case, the goal is to allow the network to be trained on a smaller number of labeled data.

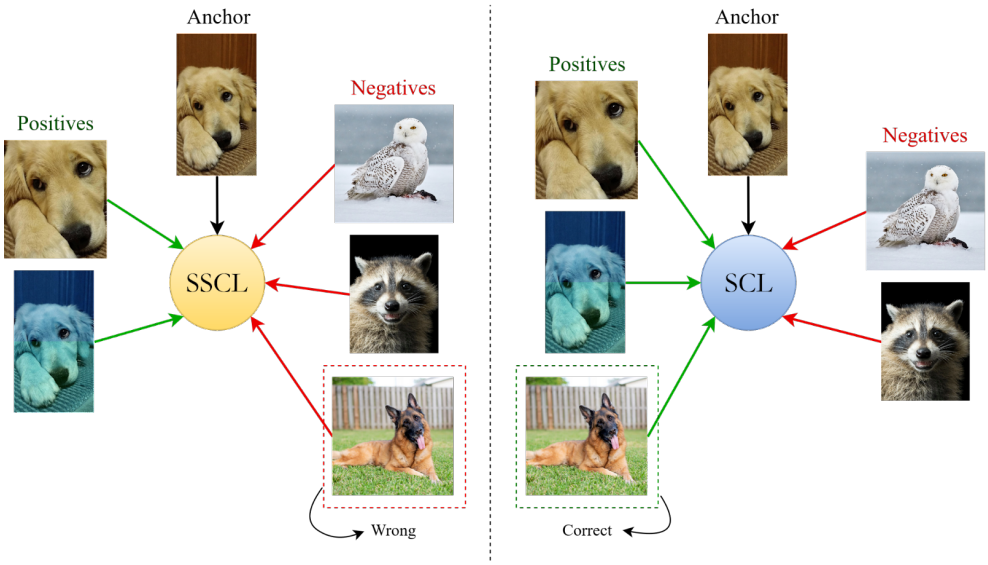


Figure 3.2: Self-supervised vs Supervised Contrastive Learning [18]

As we can observe in Figure 3.2 the self-supervised approach suffers from the problem of false negatives, namely, it may happen that the representations of two dogs are spaced in the embedding space only because one is not derived from the augmentation of the other.
Obviously, CL is applicable to different domains; in fact, in our thesis we exploited methods designed for time series. Before describing these architectures, however, it is necessary to introduce the concept of time series itself.

Chapter 4

Financial Time Series Analysis

Time Series data consist of a sequence of observations ordered over time. These can include, for example, historical records of temperatures, financial data, sales, medical diagnostic tests, sensor measurements, etc.
Time Series analysis aims at analyzing a real phenomenon observed over time in order to forecast it, detecting anomalies or classifying it, by using the set of information available up to a certain point in time t [21].

The frequency with which such observations are collected depends on the domain. One can find data at the level of second, minute, hour, day, week, month etc. To cite a few examples, there are sensors that take measurements at intervals on the order of thousandths of a second. Alternatively, one can then find historical data on companies' earnings that occur quarterly (once every three months). In general, *time series* can be divided into categories based on their characteristics [22]:

- **Univariate time series**: described by a single variable for each instant. Some examples:
  - Exchange Rates: hourly exchange rates of a currency pair, such as USD/EUR.
  - Web Traffic: hourly or daily website visitors' count over a time interval.
- **Multivariate time series**: characterized by multiple attributes. To name a few:
  - Climate data: temperature, humidity, air pressure, wind speed and precipitation.
  - Sensor data: measurements from multiple sensors (IoT), including temperature, motion, light intensity, sound levels etc.

  - Social Media data: number of followers, likes, shares, comments over time.

- **Seasonal time series**: data with a seasonal component that determines certain characteristics. The aforementioned Climate data are an obvious example.

- **Cyclical time series**: similar to the preceding category but with non-fixed frequency, namely, they consist of a pattern that repeats at non-periodic intervals. For example, financial data may show patterns of daily or weekly fluctuations.

- **Stationary time series**: when the observations of a time series are not time dependent, it is said to be stationary. These time series' statistical features will not vary over time, therefore they will have constant mean, variance, and covariance.

- **Non-stationary time series**: a time series is non-stationary if its statistical features, such as mean, variance, and auto-correlation, do not remain constant across time. For instance, a non-stationary time series is one that has a growing trend over time. As the sample size grows, so will the mean and variance.

- **Irregular time series**: they present random fluctuations that cannot be described with seasonal components. The most common example is audio files.

4.1 Financial Time Series

The financial market is governed by the *Law of Supply and Demand* and it consists of four basic components: stock, commodity, bond and exchange market [23]. What they have in common is that they are based on the concept of price (price of share, bond price, currency price, commodity price etc.). These prices are gathered with a certain frequency, generating *financial time series*. Unlike a generic *time series*, this type of data possesses characteristics that are the result of very complex dynamics. From these factors, macro- and micro-economic, comes high price *volatility* or, in other words, high uncertainty. By *volatility*, in the financial sphere, we mean a measure of the price fluctuations of a stock, market or index over time [24]. To get an idea of how uncertain the stock market is, just think that a Tweet by Elon Musk in 2018 caused Tesla shareholders to lose $12 billion in a matter of a few days. It is precisely because of this complexity that the study of *financial time series* has evolved rapidly over the years, increasingly involving Machine Learning techniques. However, especially in the world of trading, traditional methods are

still relied on. Particularly, there are two types of analysis that have historically been used for forecasting: *technical analysis* and *fundamental analysis*.
Technical analysis is conducted by means of various tools assuming that historical data can help in the effort to predict future values. Specifically, indicators are used to provide trend information and are calculated from past stock prices. These statistical tools and charts are widely used in the context of trading.
Unlike *technical analysis*, which uses historical data to make predictions, *fundamental analysis* also relies on external factors such as information about the company, the condition of the market in which it operates, company policies, the occurrence of events etc. Thus, both qualitative and quantitative analysis is conducted. For *fundamental analysis*, there are two approaches: top-down, which starts from the macroeconomic framework and then delves into the particular case; bottom-up, which analyzes the situation of the individual company and then investigates its global effects [25]. In the following sections, only *technical analysis* will be explored in depth, as it is the only one that was used in our work.

4.2 Technical Indicators

A *trade signal*, generated by technical indicators, is an analytical instrument that tells a trader when to buy or sell to optimize profits. There are various types of trading signals, each with its own set of goals and potential gains [26]. For a long time, traders have used *technical indicators* to lower their trading risk. They are mainly based on technical analysis but often include quantitative methods and measures of sentiment as well. Technical analysis, in particular, is based on various parameters such as price, volume etc. The actual signal is then generated through thresholds or comparisons made against the values of these *technical indicators*. Next we analyze those we have used within this thesis work.

4.2.1 Moving Average

The *Moving average* is a technical analysis tool that involves calculating average values with respect to different subsets, ordered over time, of the starting dataset. This is done by considering windows of arbitrary but fixed length that move progressively over time as observations are added. In other words, for each new time instant, the initial point in the window is excluded and the new observation is added to the queue. This indicator has several variants that differ substantially in the weight given to individual points when calculating the mean value.

Simple Moving Average

The *simple moving average* is calculated by adding the closing prices of the last n days and dividing them by the length n of the window [27].

$$SMA_t(n) = \frac{P_t + \ldots + P_{t-n+1}}{n}$$

In this case all elements have the same weight therefore, if we had a window of 4 elements, each point would have a 25% contribution. In addition, the frequency of observations can be daily, weekly, yearly but also at the minute level, etc., depending on the context. The choice of window length depends on individual preferences and trading strategies.

The *SMA* is frequently used to determine the trend of a market. Commonly, signals are extracted by comparing "fast" and "slow" moving averages. By the former we mean a type of MA calculated over a relatively short period, thus very responsive to price change. With the latter, the exact opposite. Signals are generated at crossover, or intersection, points as we can observe in Fig. 4.1.

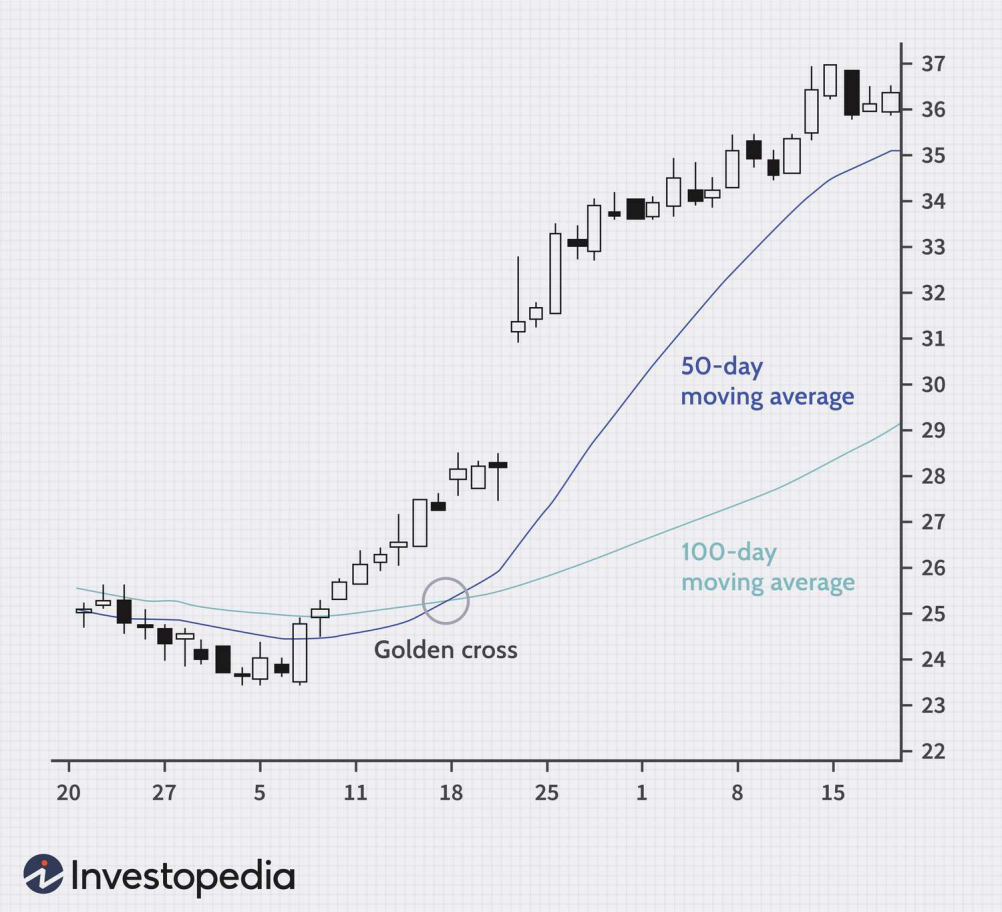


Figure 4.1: Crossover points for SMA [28]

Exponential Moving Average

SMAs are characterized by a problem called *lag*. It consists in the fact that changes at the trend level are displayed with a certain delay. Specifically, this *lag* is estimated to amount to $n/2$, where n is the length of the window. If, for example, we use a dataset with daily frequency and employ a 25-day *SMA*, we are lagging by about a dozen days, and this can be a problem within a dynamic context such as financial markets. One method to mitigate the effect of *lag* is to give more weight to more recent data. This variant, is called *exponential moving average* and is defined as follows [29]:

$$\begin{aligned} EMA_t(n,\beta) &= \beta P_t + \beta(1-\beta)P_{t-1} + \beta(1-\beta)^2 P_{t-2} + \cdots \\ &= \beta P_t + (1-\beta)EMA_{t-1}(n,\beta) \end{aligned}$$

were β is the smoothing parameter, usually defined as:

$$\beta = \frac{2}{n+1}$$

On a practical level, the first MA is an *simple moving average*.

Moving Average Convergence Divergence

Moving averages are improved by the *Moving Average Convergence Divergence*, which sheds light on market buying and selling pressure. In other words, the *MACD* is a momentum indicator that can help indicate when a market is overbought (time to sell) or oversold (time to buy) [30].

Moving Average Convergence Divergence is calculated using different *EMAs* with respect to different periods. It consists of three components: the *MACD Line*, the *Signal Line* and a value representing the Convergence/Divergence between them.

- **MACD Line**: it represents the difference between two *EMAs*, the former having a longer lookback period than the latter. This is referred to as a *"slow" signal line*. The standard lookback period values are 26 and 12 respectively.

- **Signal Line**: it constitutes the so-called *"fast" line*, the result of an *EMA* having a shorter period among those characterizing the *EMAs* mentioned above. The values mentioned above are usually associated with an *EMA* of 9.

- **MACD Histogram**: result of the difference between *MACD line* and *Signal Line*, commonly represented by a histogram.

A positive signal suggests a bullish trend, whereas a negative signal indicates a bearish one. In other words, the clue that it could be time to sell is given when the *MACD* crosses below the *Signal Line*. The indicator, on the other hand, provides a positive signal when *MACD* crosses above the Signal Line, meaning that the price of the asset is likely to experience upward momentum, as shown in Fig. 4.2 [31].

Figure 4.2: Crossover points for MACD [32]

4.2.2 Relative Strength Index

The *Relative Strength Index* is a momentum indicator that compares the current price to the high and low averages over the course of a preceding trading period. This indicator determines if a market is overbought or oversold and assists in identifying trend changes, price declines, and the establishment of bullish or bearish markets [33]. Mathematically, it is defined as:

$$RSI_t(n) = 100\frac{RS_t(n)}{1 + RS_t(n)}$$

where $RS_t(n)$, the *Relative Strength*, is the relative ratio of days with upward and downward movement over the previous n days.

$$RS = \frac{up_t(n)}{down_t(n)}$$

The standard time period is 14 observations but some intraday (short-term) traders prefer using values between 9 and 11. Instead, long-term traders prefer values in the range of 20 to 30. Typically, an *RSI* reading of 70 or above suggests an overbought condition, indicating a potentially high market sentiment. Conversely, a

value of 30 or below suggests an oversold state, indicating a potentially low market sentiment.

4.2.3 On Balance Volume

The *On Balance Volume* indicator measures buying and selling pressure assuming there are buying flows, and thus incoming volumes, on bullish sessions, and selling waves, and thus outgoing volumes, on bearish days [34].

The OBV basically adds the volumes of that day to the value of the OBV index from the previous day, in case there was a price increase, to calculate the total positive or negative volume of a trading session.

In contrast, if there was a negative session, the volumes of the observation day are subtracted from the value of the OBV index from the previous day.
To recap:

$$OBV_t = OBV_{t-1} + \begin{cases} Volume_t, & \text{if } Close_t > Close_{t-1} \\ -Volume_t, & \text{if } Close_t < Close_{t-1} \\ 0, & \text{if } Close_t = Close_{t-1} \end{cases}$$

As we can see in Fig. 4.3, between February and March the indicator experienced a sharp upward trend, suggesting traders to get in before the rise.

Figure 4.3: OBV upward trend [35]

4.2.4 Average Directional Index

The *Average Directional Index* is a directional movement index that measures the strength of a trend. It is composed of three different values:

- ADX: It solely indicates the strength of the trend, regardless of its direction. Commonly, values above 40 are indicative of strong trends, below 20, of great stability.

- +DI: it indicates increasing values in case of bullish markets (increasing prices).

- -DI: indicates increasing values in case of bearish markets (decreasing prices)

At the signal level, it is often used in conjunction with the *MACD*. Specifically, when *+DI* crosses upward *-DI* and at the same time *MACD* Main is above *MACD Signal*, it is bought. Conversely, when *+DI* crosses downward *-DI* and at the same time *MACD Main* is below *MACD Signal*, one sells [36].

4.2.5 Bollinger Bands

Bollinger Bands are a useful indicator for determining whether an asset is oversold or overbought. They are, as the name suggests, in the form of two bands that cover a *simple moving average* on the top and bottom. These bands, in particular, are usually defined as +/- 2 standard deviations from a 20-day *SMA* [37]. At the signal level, if price continually touches the upper band, an overbought market is evidenced. Similarly, if the same occurs with respect to the lower band, is is an oversold state. Here is this Bollinger Band formula:

$$BOLU = MA(TP, n) + m \times \sigma[TP, n]$$
$$BOLD = MA(TP, n) - m \times \sigma[TP, n]$$

where BOLU and BOLD are the Upper and Lower Bollinger Bands, respectively. MA is a Moving average, TP is the typical price, defined as (Close+Low+High)÷3. Commonly, n, number of days in smoothing period, is set to 20, whereas m, the number of standard deviations is typically 2. Finally, σ[TP,n] is the standard deviation over last n periods of TP.

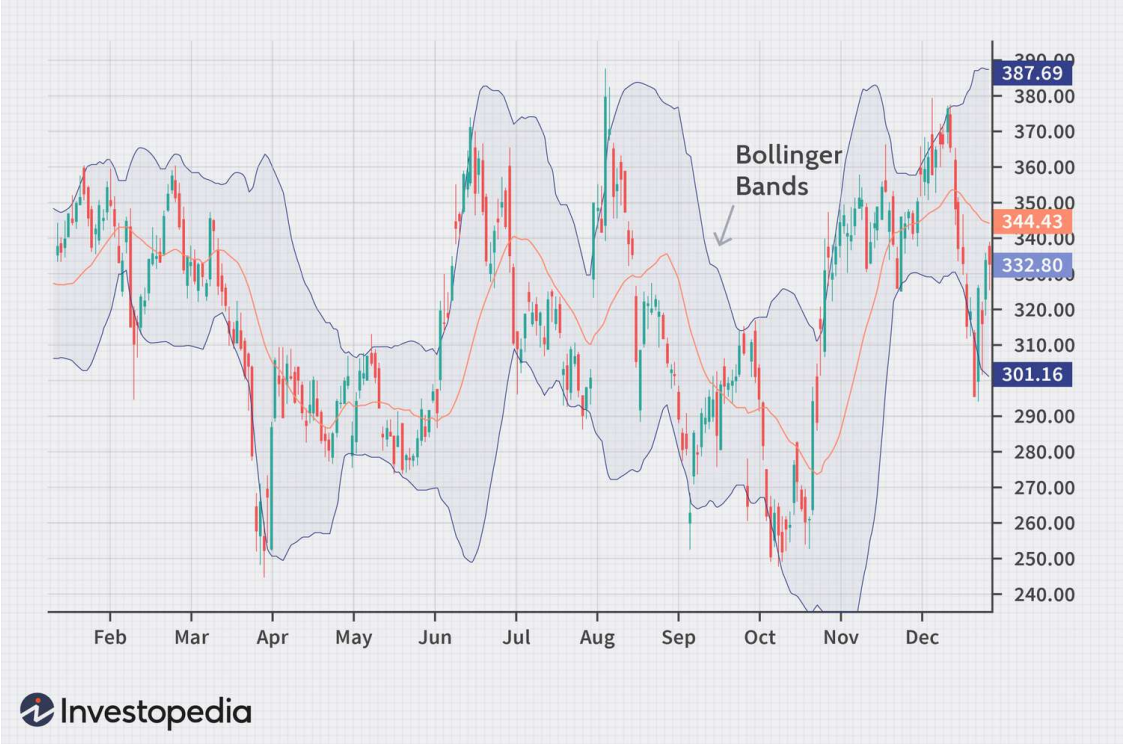


Figure 4.4: Bollinger Bands [38]

4.3 Forecasting methods (baseline)

So far we have explored some *technical indicators*. In this section, we investigate models that aim to predict the exact values of the time series in the future.
Within this thesis, these methods represent a baseline, as they are simpler models than NN for making predictions. Besides being less complicated, they require accurate setup, with very precise parameter setting, heavily influencing the results. Nevertheless, they represent a benchmark for evaluating how much the use of complex methods is worthwhile.

4.3.1 Autoregressive Models

Autoregressive models use data from previous periods as input for a linear regressor with the intent of predicting the value at the next time step. By linear regressor we mean a model that generates an output based on the linear combination of the input data. Because the regression occurs at the level of the same variable considered at different times, we use the term *autoregressive* (regression of the variable against itself). Therefore, an AR model can be expressed as [39]:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t, \tag{4.1}$$

where p is the order of the model and ε_t is white noise. These models start from the fundamental assumption that previous data are somehow useful in predicting future data. This relationship is called correlation, in this case *autocorrelation*, for the same reason as before. If the variables change in the same direction, the correlation is positive, if in the opposite direction, negative, otherwise they are uncorrelated. The greater the correlation with respect to a variable, the greater its weight in determining output. Since *autoregressive models* are too simplistic, some variations and extensions were invented. These include *autoregressive integrated moving average models (ARIMA)* and *Exponential Smoothing*, that were employed for the purpose of this thesis.

4.3.2 ARIMA

ARIMA stands for *autoregressive integrated moving average*. Analyzing the terms from which the name is composed [40]:

- Autoregressive (AR): as mentioned in the previous paragraphs, means a regression of the variable itself against its previous values.
- Integrated (I): refers to the differentiation operation, aimed at obtaining a stationary time series.

- Moving average (MA): represents the relationship between the present observation and a residual error term based on previous errors. To forecast future values, it takes the average of recent forecast errors into account.

The model is characterized by three parameters: p, d and q.

- p = order of the AR part
- d = degree of differencing
- q = order of the MA (it specifies the number of lagged forecast errors).

The overall model can be expressed as [39]:

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t \tag{4.2}$$

Parameters p, d and q are usually established through statistical tools such as autocorrelation plots, whereas c, θ and θ are estimated through the maximum *likelihood estimation (MLE)* method.

4.3.3 Exponential Smoothing

Exponential Smoothing is not strictly an autoregressive model in the sense that, as we mentioned earlier, this family of models predicts future values by assuming that they are a linear combination of past values. *Exponential Smoothing*, on the other hand, assigns exponentially decreasing weights to past values [41]. There is no linear relationship but emphasis is placed on the most recent observations and a *smoothing factor* is applied. There are several versions of this model such as *Single, Double and Triple Exponential Smoothing*. We briefly explore only the first case as it is functional to the type of data we used during our analysis.

Single Exponential Smoothing

Single (or Simple) Exponential Smoothing is indeed a suitable model for forecasting univariate data that do not exhibit a seasonal pattern. It is parameterized by a *smoothing factor* α, that controls the frequency at which the importance of older observations is damped. Typically, this parameter takes values between 0 and 1. Mathematically [41]:

$$\hat{y}_{T+1|T} = \alpha y_T + \alpha(1-\alpha) y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} + \cdots , \tag{4.3}$$

For each value of α, it results that the weights decrease exponentially going backward in time. Specifically, if α takes a value close to 0, more weight is given to observations far in the past than for "high" α values (close to 1).

Chapter 5

Contrastive Learning frameworks for Time Series

Lately, efforts are in the direction of creating representations of time series. The ultimate goal is to use these *embeddings* for a downstream task that may be, for example, classification or regression. Many other Deep Learning models exist in the literature such as DMLP, RNN, LSTM, CNN etc. but in the next sections we will only describe architectures based on Contrastive Learning. All the techniques that we will consider learn *embeddings* by mapping similar occurrences (positive pairs) to similar representations while pushing dissimilar observations (negative pairs) far apart. CL, in this context, has proven to be an extremely successful approach, due in part to the large availability of data belonging to different domains ranging from finance to medicine. Particularly, these self-supervised learning approaches have been demonstrated to perform better in time series forecasting tasks than traditional supervised models [42, 43, 44].
In addition, the following architectures can be divided into two categories: those designed for forecasting and those for time series segmentation. The difference is that the former require an additional classification or regression layer, the latter are designed for inferring latent states of time series in a non-supervised manner. In particular, the only architecture designed for time series segmentation is Time2State while TNC, CoST and TS2Vec are representation learning techniques for forecasting.

5.1 TNC

TNC [45] is a self-supervised framework capable of creating nonstationary time series representations. The acronym stands for Temporal Neighborhood Coding, in fact it defines neighborhoods in time that possess stationary properties. Being

a Contrastive Learning framework, it is ensured that representations within a neighborhood are similar to each other and dissimilar to those of other signals not belonging to it. The approach stems from the need to work with unlabeled datasets. For example, it is very common in the medical case that it is not possible, even for clinicians, to continuously attribute states, over long periods, to the health conditions of individuals. TNC leverages the inherent local smoothness of the signal generation process to generate representations of windows. These are denoted W_t, where t is the time instant with respect to which they are centered. Instead, N_t denotes the neighborhood relative to the window W_t t and is normally distributed $\mathcal{N}(t, \eta\delta)$ where δ is the window size. η governs the range of the neighborhood and it is dependent on the properties of the time series, in fact reasonably it is defined by a domain expert. Since the neighborhood consists of similar instances, the range should possess the property of stationarity, a check should be performed. It is performed by *Augmented Dickey-Fuller statistical test* to determine this region for each window. Samples outside the neighborhood are denoted by $\bar{N}_t$ and are likely to possess different characteristics, so negative samples are considered. However, there is a risk of creating false negatives [46]. For example, it may happen that a patient goes from a normal to a critical state and then returns to the former. With the approach just described, potentially, the points belonging to the two normal states can be considered negative samples only by virtue of their distance in time, committing an error. To mitigate this effect, Positive-Unlabeled (PU) learning is exploited, according to which a classifier is trained using data labeled by the positive class (P) and unlabeled data (U), a mixture of positive and negative samples [47, 48]. Among the different approaches there is one that gives less weight to samples belonging to U [49, 50]. Specifically, positive samples are given unit weight, while the others are treated as a combination of a positive and a negative sample, with weights w and 1-w, respectively.
Concretely, a loss is drawn that incentivizes the distinction, in terms of representations, of samples belonging to different neighborhoods.
TNC consists of two components:

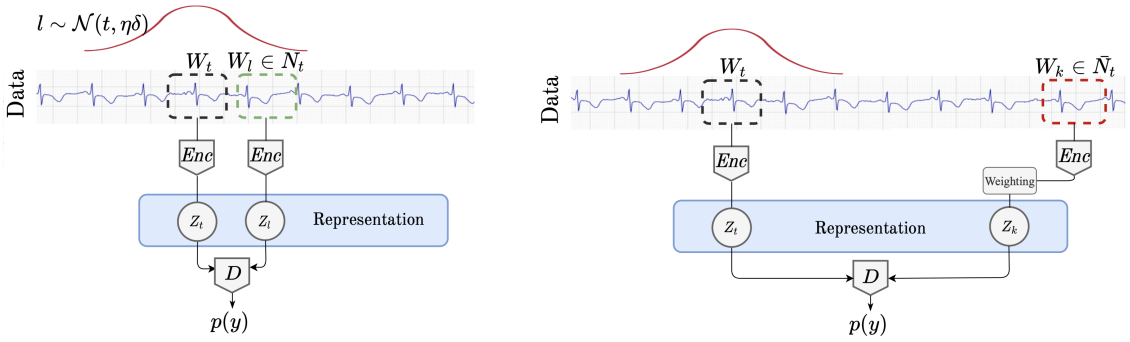
- $Enc(W_t)$: an encoder that maps each window to a Z_t representation within a latent space.

- $D(Z_t, Z)$: a discriminator that computes the probability of two samples to belong to the same neighborhood.

The loss is defined as follows:

$$\begin{aligned}\mathcal{L} = &- \mathbb{E}_{W_t \sim X}[\mathbb{E}_{W_l \sim N_t}[logD(Z_t, Z_l)]+ \\ &+ \mathbb{E}_{W_k \sim \bar{N}_t}[(1 - w - t) \times log(1 - D(Z_t, Z_k)) + w_t \times logD(Z_t, Z_k)]]\end{aligned} \tag{5.1}$$

Where Z_t, Z_l and Z_k are the output of the encoder. Both the encoder and the discriminator are trained, while in the inference phase only the former is used.



(a) Neighborhood samples

(b) Non-neighboring samples

Figure 5.1: TNC architecture [45]

An overview of the components is shown in Fig. 5.1
Specifically, for each W_t window, the neighborhood distribution is first defined. The encoder learns the distribution of sampled windows with respect to N_t and $\bar{N}_t$ in a latent space. The generated embeddings are then fed into the discriminator, which predicts the likelihood of them belonging to the same neighborhood.

5.2 CoST

CoST [51] is a time series representation framework that exploits CL methods to learn disentangled seasonal-trend embeddings. This requirement stems from the fact that data is frequently complex and outcome of the interplay of several sources, hence representations must be able to disentangle the various explanatory sources. Losses consider both time-and frequency-domain representations. The model was developed since the representations and prediction connections learned via the end-to-end training strategy do not effectively transfer or apply to situations where data is generated from a non-stationary environment, which is a prevalent scenario. The starting point is the property of time series for which they can be decomposed into trend, seasonal component and error [52, 53]. CoST can effectively learn trend representations by coping with the problem of choosing an appropriate lookback window. This is done by a mixture of autoregressive experts. It also succeeds in obtaining effective seasonal representations by exploiting a Fourier layer that allows interaction between different frequencies. Both components are trained with

a Contrastive Learning method.
As stated before, the key prior is the fact that a time series X is considered to be generated by an error variable E and X*. The latter is composed of T and S, variables representing trend and seasonality, which do not influence each other [54, 55]. Therefore, the mechanisms that model the two components can be detached. Furthermore, interventions on the error component do not affect the conditional distribution P(X*|T,S). Consequently, invariant embeddings of T and S are learned and augmentations on E are applied in a contrastive learning context. These interventions are the typical ones: scale, jitter, and drift.
As can be seen from Fig. 5.2, the architecture is composed of several modules. The first is a Backbone encoder that maps the input to a latent space. From these intermediate representations, trend and seasonal embeddings are extracted, using the respective modules. Specifically, the Seasonal feature disentengler extracts representations via a tractable Fourier layer with a loss in the frequency domain that includes an amplitude component and a phase component. In contrast, the Trend feature disentengler extracts representations with a mixture of autoregressive experts in the time domain. The overall loss is formulated as:

$$\mathcal{L} = \mathcal{L}_{time} + \frac{\alpha}{2}(\mathcal{L}_{amp} + \mathcal{L}_{phase}) \tag{5.2}$$

where α is a hyperparameter that balances the contribution of trend and seasonal factors. The final outcome is the result of the concatenation of the output of Trend and Seasonal Feature Disentanglers.

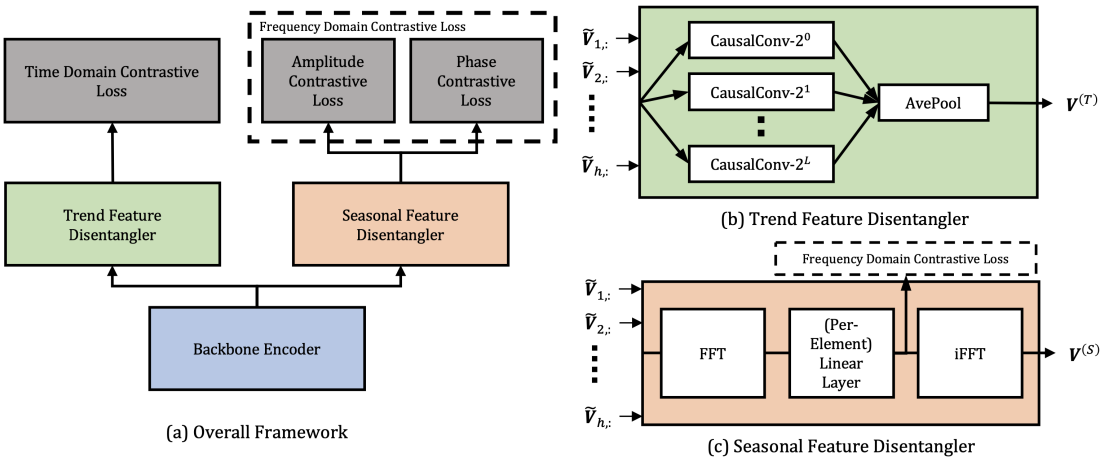


Figure 5.2: CoST architecture [51]

The Time Domain Contrastive Loss is defined as:

$$\mathcal{L}_{time} = \sum_{i=1}^{N} -log \frac{exp(q_i \cdot k_i/\tau)}{exp(q_i \cdot k_i/\tau) \sum_{j=1}^{K} exp(q_i \cdot k_i/\tau)} \tag{5.3}$$

where given a sample V^T, final trend representation, in order to generate q and k, to begin, we randomly choose a time step, denoted as t, for the loss function and use a one-layer MLP as the projection head. K and q represent the augmented versions of the relevant samples from the dynamic dictionary and momentum encoder, respectively. Specifically V^T is the result of L+1 regressive experts implemented as 1d causal convolution to which average pooling is applied.
Frequency Domain Contrastive Loss, as mentioned earlier is divided into two components:

$$\mathcal{L}_{amp} = \frac{1}{FN} \sum_{i=1}^{F} \sum_{j=1}^{N} -log \frac{exp(|F_{i,:}^{(j)}| \cdot |(F_{i,:}^{(j)})'|)}{exp(|F_{i,:}^{(j)}| \cdot |(F_{i,:}^{(j)})'|) + \sum_{k \neq j}^{N} exp(|F_{i,:}^{(j)}| \cdot |(F_{i,:}^{(k)})'|)} \tag{5.4}$$

$$\mathcal{L}_{phase} = \frac{1}{FN} \sum_{i=1}^{F} \sum_{j=1}^{N} -log \frac{exp(\phi(F_{i,:}^{(j)}) \cdot \phi((F_{i,:}^{(j)})'))}{exp(\phi(F_{i,:}^{(j)}) \cdot \phi((F_{i,:}^{(j)})')) + \sum_{k \neq j}^{N} exp(\phi(F_{i,:}^{(j)}) \cdot \phi((F_{i,:}^{(k)})'))} \tag{5.5}$$

where $F_{i,:}^{j}$ is the sample j within a mini-batch, and $(F_{i,:}^{j})'$ its version result of an augmentation. ϕ denotes the phase representation.
The loss is then evaluated in the frequency domain arrived at by a discrete Fourier transform followed by a learnable Fourier layer. However, the seasonal representation is obtained by returning the features to the time domain via an inverse discrete Fourier transform.

5.3 TS2Vec

TS2Vec [56] is a model that aims at overcoming three major limitations.
First, instance-level representations did not allow adequate granularities for tasks such as time series forecasting or anomaly detection.
Furthermore, many of the existing methods, including TNC, are unable to obtain features that are scale-invariant.
Third, most models for unsupervised time series representation have emulated approaches peculiar to computer vision and natural language processing, but they are inadequate for this context. For example, as mentioned in Fig. 3.1 the cropping operation is widely used to create image augmentations. From the perspective of time series, on the other hand, it would potentially mean losing too much relevant

information. To solve these challenges, TS2Vec is presented as a model capable of learning contextual embeddings for arbitrary segments at different semantic levels. To this end, a hierarchical contrastive method was introduced at both the instance and temporal levels to capture contextual information at different scales. Second, a contextual consistency is proposed for choosing positive pairs, which is more appropriate for time series than previous methods.
The overall architecture is described in Fig. 5.3, from which it can be seen that two overlapping segments are randomly extracted. The goal is to obtain a contextual representation of the common part.
The two segments are provided as input to the encoder f_θ , which is made up of three modules:

- Input projection layer: fc layer mapping the instances of the series at a time t into the corresponding in a high-dimensional latent space.
- Timestamp masking module: generates augmented context views by mapping the output of the previous module to random instants of time.
- Dilated CNN: used to obtain a contextual representation at each time instant [42].

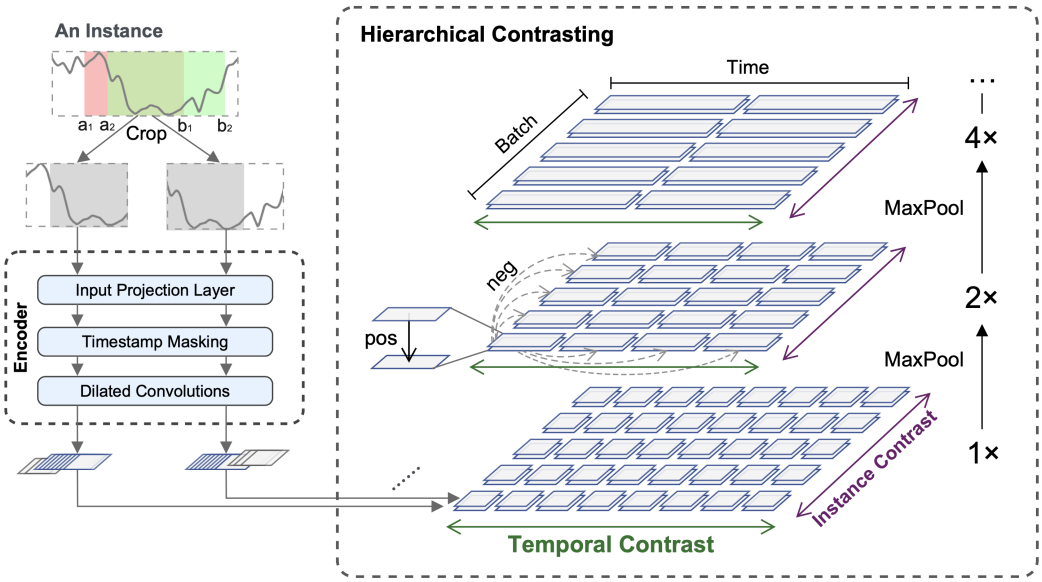


Figure 5.3: TS2Vec architecture [56]

A crucial step that differentiates TS2Vec from other models is the construction of positive pairs. There are other approaches in the literature such as *Subseries*

consistency [57], *Temporal consistency* [45], and *Transformation consistency* [58], but these are based on assumptions too strong to be generalized to the time series case. Instead, the authors of TS2Vec proposed a new method, *contextual consistency*, which considers representations in two augmented contexts, at the same time instant, as positive pairs. One context, in particular, is obtained applying timestamp masking and random cropping on the input series. In this way, not only magnitude is preserved, but also representations are more robust because timestamp reconstruction is forced with respect to different contexts.
In other words, two segments of the time series that share a portion are randomly sampled. Augmented versions of these two segments are created by masking and random cropping, but preserving the source timestamps. By doing so, positive samples can be created by pairing points relative to the same time which will obviously belong to the segment in common defined above. The method is represented in Fig. 5.4.

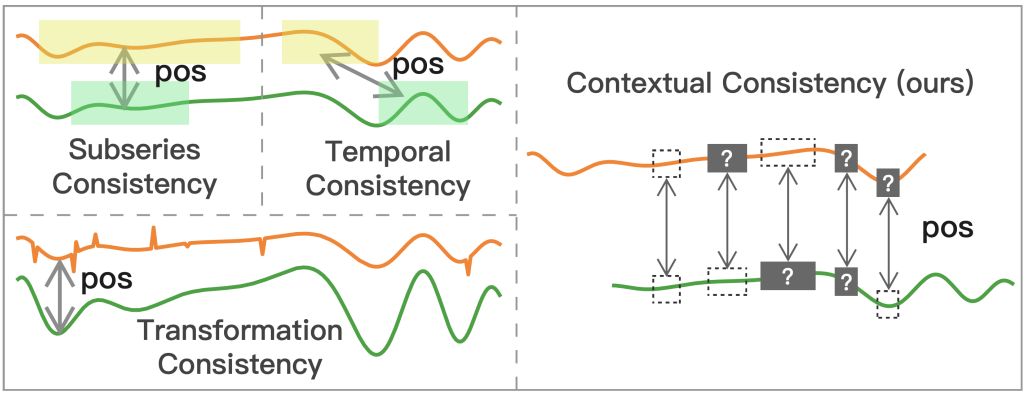


Figure 5.4: TS2Vec positive pair construction [56]

To obtain the contextual representations, the model uses two losses: the Temporal Contrastive Loss and the Instance-wise Contrastive Loss. Regarding the former, positive pairs (associated with the same timestamp) and negative pairs (pairing different instants) are considered to learn the over-time representations. The Temporal Contrastive Loss is presented as follows:

$$\mathcal{L}_{temp}^{(i,t)} = -log \frac{exp(r_{i,t} \cdot r'_{i,t})}{\sum_{t^i \in \Omega}(exp(r_{i,t} \cdot r'_{i,t}) + \mathbb{1}_{[t \neq t']} exp(r_{i,t} \cdot r'_{i,t}))} \tag{5.6}$$

where Ω is the set of overlapping timestamps, t the timestamp, i the index of the input timeseries, $r_{i,t}, r'_{i,t}$ are the two augmented representations, at a given timestamp.

Instead, Instance-wise Contrastive Loss is defined as:

$$\mathcal{L}_{inst}^{(i,t)} = -log\frac{exp(r_{i,t} \cdot r'_{i,t})}{\sum_{j=1}^{B}(exp(r_{i,t} \cdot r_{j,t}) + \mathbb{1}_{[i \neq j]}exp(r_{i,t} \cdot r_{j,t}))} \tag{5.7}$$

where B is the batch size. Negative samples are time series points associated with the same index but in different batches.
The overall loss is formulated as:

$$\mathcal{L}_{dual} = \frac{1}{NT}\sum_{i}\sum_{t}(\mathcal{L}_{temp}^{(i,t)} + \mathcal{L}_{inst}^{(i,t)}) \tag{5.8}$$

5.4 Time2State

Time2State [59] is an unsupervised model, i.e., capable of coping with the need to work with unlabeled data, avoiding a complex and time consuming operation. This is accomplished through a new self-supervised loss function that uses a sliding window to create embeddings from a time series. It is ductile as well, since by non-parametric Bayesian methods it can effectively estimate the number of states, which is not known a priori.
Time2State attempts to resolve some pitfalls of Triplet-Loss [20] and TNC [45], baseline in the context of time series segmentation. By the latter expression we mean the subdivision of a time series into portions that reflect a state of the objects. Some examples of these states may be running, walking, jumping etc. The problem with Triplet-Loss and TNC is that they only consider the pair-wise distance. In contrast, LSE-Loss (i.e., the proposed objective function), considers both intra-state and inter-state distance thus succeeding in creating a more effective representation.
Time2State consists of two phases: training and detection phase. During the former, an encoder capable of creating time series representations is trained. This is done by assuming that consecutive windows in time are characterized by the same state with high probability, as can be seen in Figure 5.5. In other words, if a window is associated with a state, then the N consecutive windows constitute a series of N states that are likely to be the same. These N states are named intra-state samples. The respective representations then should lie close together in the embedding space. In contrast, non-consecutive windows with high probability are not represented by the same state, so they should be separated in latent space. The latter are called inter-state samples.

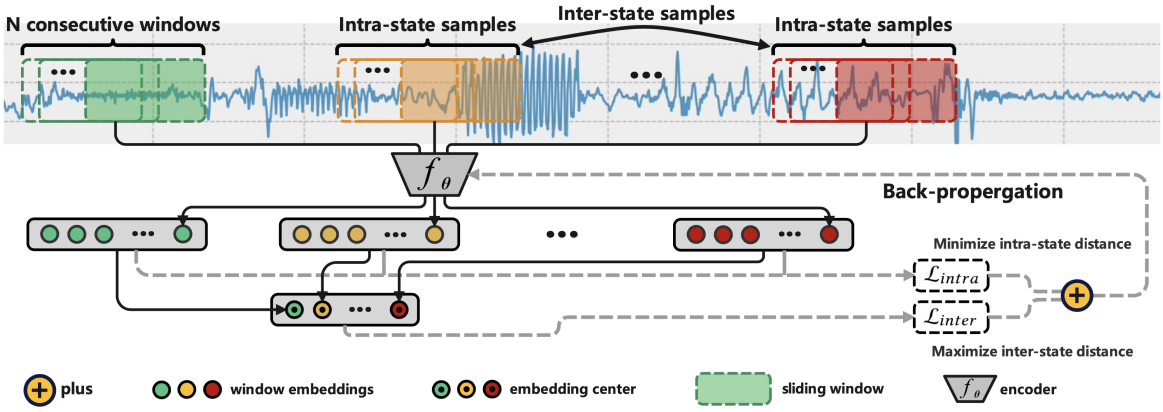


Figure 5.5: Time2State training procedure [59]

In the training phase, non-consecutive M windows are placed, obtaining the inter-state samples. From each of them, N consecutive windows are obtained (with a unit step), constituting the intra-state samples. Based on this sample distinction, the LSE-Loss was devised. This loss consists of two parts:

$$\mathcal{L}_{LSE} = \mathcal{L}_{intra} + \mathcal{L}_{inter} \tag{5.9}$$

where $\mathcal{L}_{intra}$ incentivizes the encoder to maximize the similarity of consecutive windows, $\mathcal{L}_{inter}$ on the other hand tries to minimize the similarity between intra-state samples by separating them in the embedding space.
More specifically, these two losses are defined as follows:

$$\mathcal{L}_{intra} = \alpha_1 \sum_{k=1}^{M} \sum_{i=1}^{N} \sum_{j=1, j<i}^{N} -\log(\sigma(f_\theta(o_i^k)^T f_\theta(o_j^k)))) \tag{5.10}$$

where $\alpha_1 = \frac{2}{M*M*(N-1)}$ averages the similarity, σ is a sigmoid function, f is the encoder, θ is a parameter. o_i^k is the i-th window in the k-th group. The product between the output of the encoders is the well known dot product used to quantify the similarity.

$$\mathcal{L}_{inter} = \alpha_2 \sum_{k=1}^{M} \sum_{l=1, l<k}^{M} -\log(\sigma(-c_k^T c_l)) \tag{5.11}$$

where $\alpha_1 = \frac{2}{M*(M-1)}$ is used to average the similarity as above and $c_k = \frac{1}{N}\sum_{i=1}^{N}(f_\theta(o_i^k))$, denoting the embedding center of the k-th group. The negative sign is used to minimize the similarity between inter-state samples.

Finally, in the detection phase, the whole time series with a step equal to s is analyzed by generating embeddings for a downstream task. In the specific case of the paper, clustering is applied to identify to assign labels.

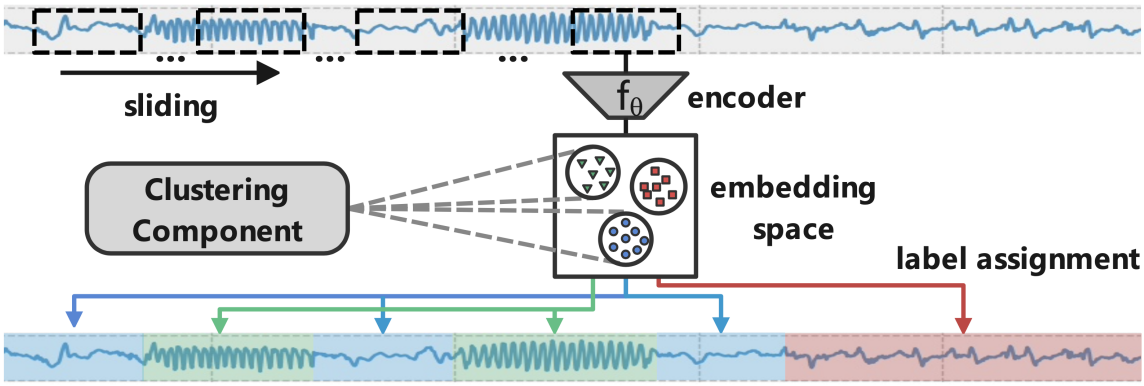


Figure 5.6: Time2State inference procedure [59]

5.5 Summary

MODEL	PAIR CONSTRUCTION CRITERION	AUGMENTATIONS	LOSS DOMAIN	LOSS COMPONENTS
TNC	Neighborhood-wise	-	Time	1
CoST	Augmentations	Scaling Shifting Jittering	Time and Frequency	3
TS2Vec	Augmentations	Timestamp masking Cropping	Time	2
Time2State	Neighborhood-wise	-	Time	2

Figure 5.7: Contrastive frameworks main characteristics

Figure 5.7 shows the main features that differentiate the four models.
In particular, the construction criterion of positive and negative pairs in the context of CL, the domain in which the loss is evaluated, the number of components of the total loss, and the types of data augmentations (if needed) are reported.
What is certainly most relevant is the criterion for creating pairs, which allows us to divide these models into two groups: those that consider the neighborhood and those that create augmented versions for creating positive pairs.

Chapter 6

Methodology

6.1 Task

The task we considered is *financial forecasting*. The objective is to predict the future movements of an asset. The problem can be divided into two categories: *price forecasting* and *movement prediction*. In the first case, the aim is to predict the exact price of an asset at a certain number of days in the future (regression). In the second scenario, on the other hand, the objective is classifying the future trend into categories such as increasing, decreasing, and, sometimes, neutral (classification). There is a further categorization of the problem that considers the type of asset under consideration [60]:

- *Stock price forecasting*: prediction of the price of a single stock.
- *Index prediction*: prediction of an index instead of a single stock.
- *Forex price prediction*: prediction of the exchange prices of currency pairs.
- *Commodity price prediction*: prediction of future prices of commodities such as raw materials, primary products such as oil, gold, natural gas, wheat, coffee etc.
- *Bond price forecasting*: prediction of changes in the prices of bonds, which are debt instruments issued by governments and other entities to raise capital. Such prices reflect are index of the state.
- *Cryptocurrency price forecasting*: prediction of cryptocurrency price.

In this thesis work, we opted for *index price forecasting*.
A stock index summarizes the value of a set of stocks and their movements over time. The actual value is determined as a weighted average of the prices of the

stocks that belong to it. There are three categories of indexes that differ in how the weight is assigned:

- Equally weighted indices: the weights are equally distributed with respect to all the stocks that make up the index (Dow Jones U.S. Select Equal Weight Index, S&P 500 Equal Weight Index, Russell 2000 Equal Weight Index etc.)

- Price-weighted indices: the weights of individual stocks are proportional to their prices over time (Dow Jones Industrial Average, Nikkei 225, Amex Gold BUGS Index etc.).

- Value-weighted indexes: the weight of each stock is proportional to its market capitalization (NASDAQ 100, S&P 500 Index, FTSE All-World Index etc.).

Many researchers have opted for index forecasting because of their lower volatility than individual stocks as a composition of the latter's performance. For this very reason, indexes are more indicative of the actual state of the market and momentum.

6.2 Approach

To summarize this work in a nutshell, the objective is to make a comparison in terms of performance between two stock price forecasting (regression) methods. On the one hand, models that exploit *contrastive representations*, and on the other hand, simpler models that rely on *technical indicators*. As we can observe in Fig. 6.1 there are two subsets. The blue one represents the direct method for which time series representations are created by contrastive methods and then provided as input to a regressor. The red one, on the other hand, represents our proposed method. Thus, we start from the embeddings generated by the contrastive model to provide them as input to a clustering method. The resulting clusters are used as labels for the initial time series (not in the form of log-returns but as Close, High, Low etc. prices), of which technical indicators are calculated. The newly generated dataset is the input for an ensemble model from which the most important features are extracted using a threshold. The "filtered" dataset is provided as input to a regressor identical to the one that received the contrastive embeddings. The result of the regressors, trained on the same task, is evaluated in terms of MSE and MAE, and a comparison can be made.

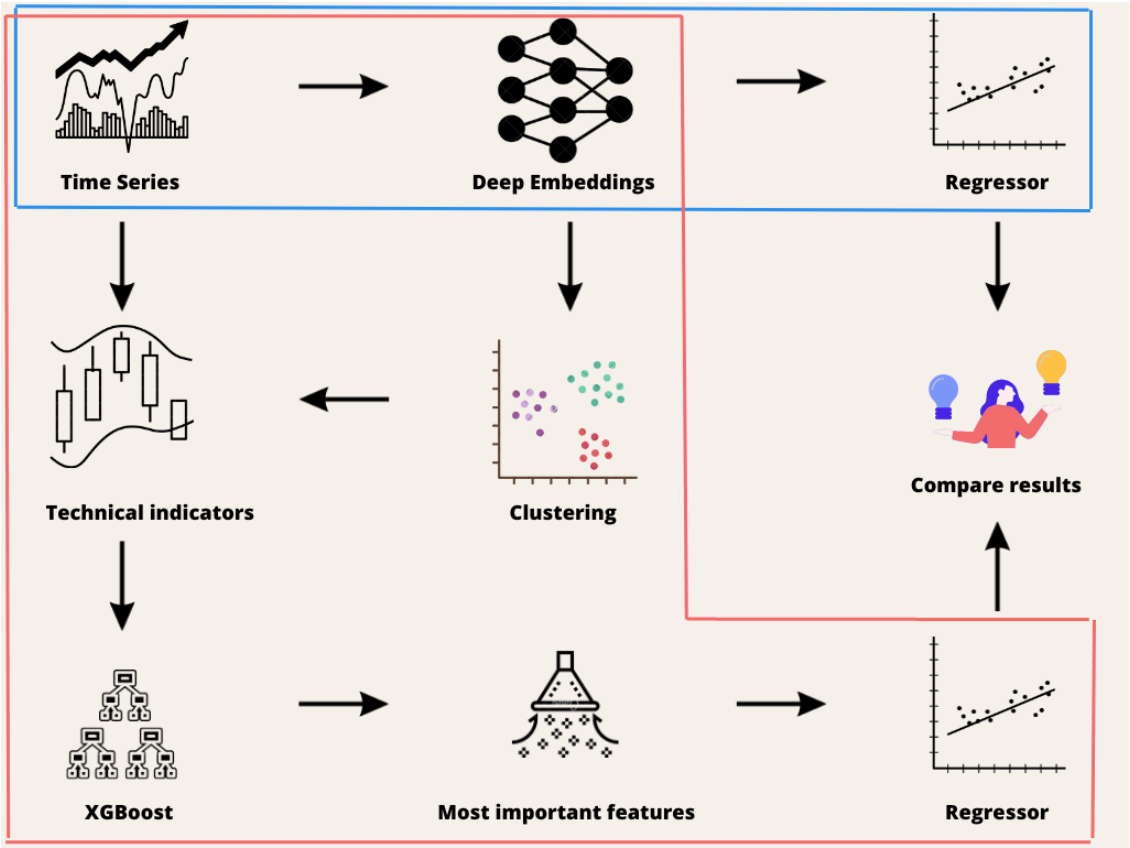


Figure 6.1: Overall methodology

The next paragraphs will describe in details the different steps, following the order observable in Figure 6.1.

6.2.1 Contrastive Representations

The first stage of the process consists in training the four models: TNC, CoST, TS2Vec, and Time2State. This phase is performed for 200 epochs for both univariate and multivariate datasets, separately. Recall that both possess log-returns as features, in the former case computed only with respect to NDX, in the latter, instead, with respect to all the stocks composing it. In both cases, the labels are the future values of the index with horizons of 3, 5 and 7 days. At the end of the training, the respective encoders are ready to generate a contrastive representation of a time series they receive as input.

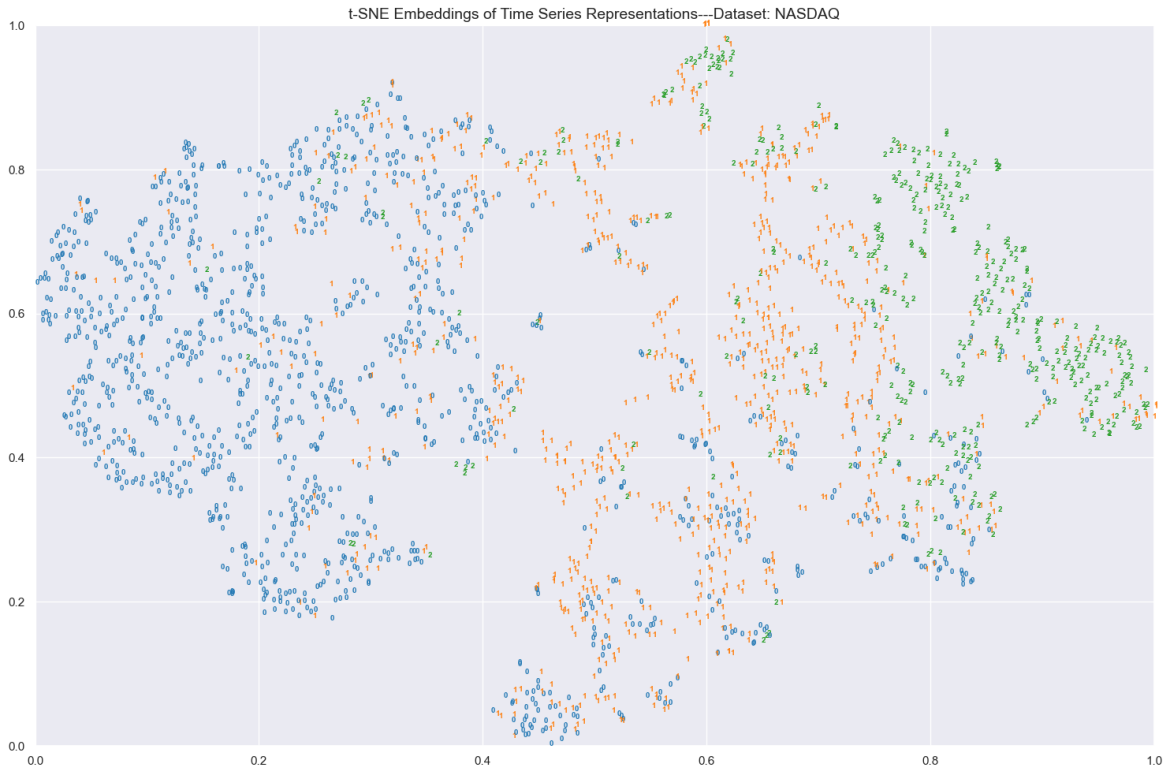


Figure 6.2: Time2State t-SNE embeddings in 2D, KMeans(k=3)

6.2.2 Clustering

The second step is functional to our attempt to make black-box models explainable. One of the major problems of Deep Learning is precisely the impossibility of understanding how a certain output is generated, which features are most relevant, and, in our case prominently, which feature of the time series is relevant. Consequently, the encoder output is provided as input to a clustering algorithm, in our case, KMeans. Since we could not determine a priori a reasonable number of clusters, we experimented with the following values: 2, 3, 5, 7, 10, 30. At this point, each temporal instant (day) corresponds to an embedding, and each of these representations corresponds to a cluster.

6.2.3 Dataset Labeling

The next step is to label the dataset characterized by the common features (Open, High, Low, Close, Adj Close, and Volume) and the technical indicators presented in section 4.2 with the extracted clusters. This new dataset is then given as input to *XGBoost*. This model (like *Decision Tree, Random Forest*, etc.) allows us to

understand which features are most important. By "important" we mean those that contribute most in improving the performance measure (e.g., the *Gini index*). The *feature importances* are then averaged across all of the the decision trees within the model. To actually filter out the most important ones, we set an arbitrary threshold. The top n features that cumulatively constitute no more than the threshold in terms of importance are selected. Since filtering is done by threshold, it is very likely that the features extracted will differ in number from case to case. As we can observe in Figure 6.3, there are cases where the importances are more concentrated and others where they are more evenly distributed.

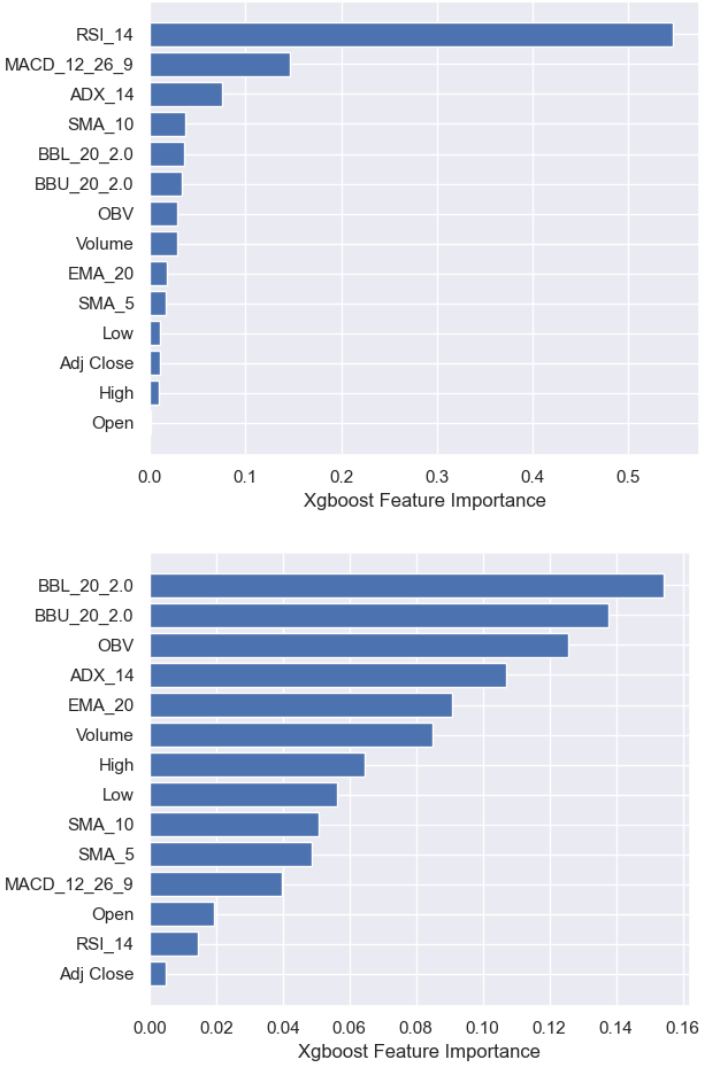


Figure 6.3: Example of Time2State and CoST embeddings feature importances

6.2.4 Forecasting

At this point one has, for each model, a pair of representations. On one hand, the contrastive representations, on the other, the features made by indicators, that best represent the criterion by which the deep embeddings were created. One can proceed to feed the same model the pairs. The model considered is Ridge but it is not actually relevant, what matters is the comparison in terms of performance between the two types of embeddings. If the gap is not large, then we were able to create a "light-weighted" version, i.e., a model that requires neither heavy training nor a complex architecture.

Chapter 7

Experiments and Results

7.1 Implementation details

7.1.1 Dataset

We decided to use as input for all our models the NASDAQ 100, a representative index of Nasdaq performance made up of 100 listed equities issued by highly-capitalized non-financial companies. Its composition is modified weekly, and the stock weights are revised based on an algorithm that ensures a balance among the various economic sectors in which the 100 stocks are issued.

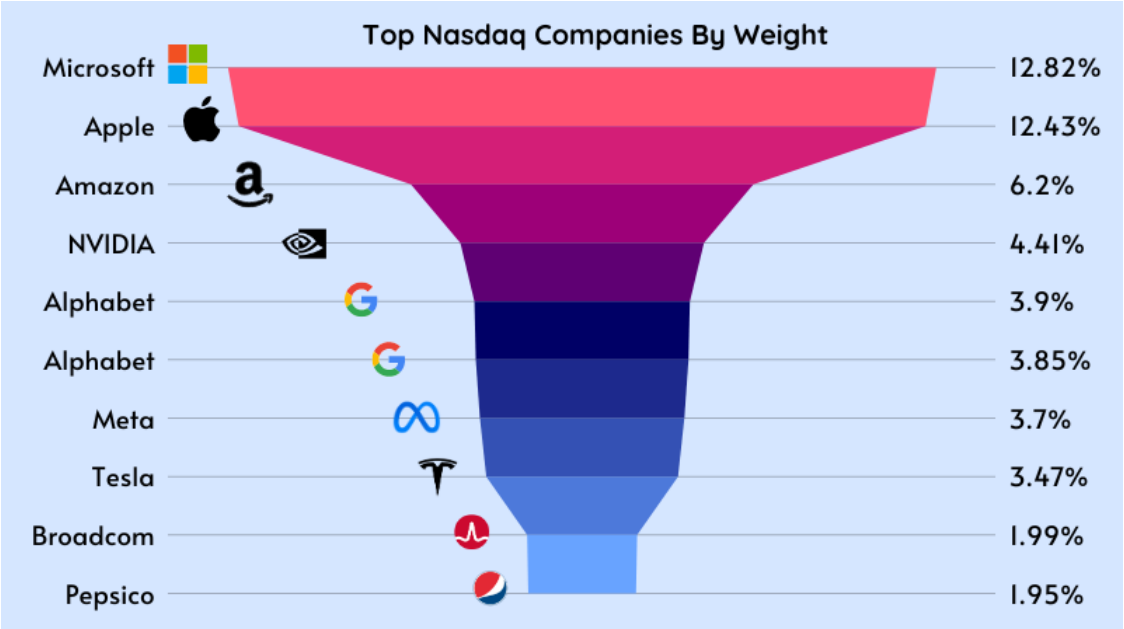


Figure 7.1: Top Nasdaq Companies By Weight, updated 2023 [61]

The data were sourced through Yahoo Finance and consists of the common features *Date, Open, High, Low, Close, Adjusted Close*, and *Volume*. From these raw features, we generated four additional datasets:

- Univariate: daily historical NDX data.
- Multivariate (stocks only): historical data of stocks belonging to the Nasdaq.
- Multivariate (stocks with index): historical data of stocks belonging to the Nasdaq, including the index itself.
- Multivariate (technical indicators): set of technical indicators calculated with respect to the index. These indicators are the same as those stated in the 4.2 section, namely SMA, MACD, RSI, OBV, ADX and BB.

In the first two cases, by "historical data" we refer not to the entire set of features listed above, but to the logarithmic returns calculated with respect to *Close price*. Precisely, in the multivariate case, these features are calculated for each stock. Logarithmic returns are defined as the logarithmic difference in prices at two different instants:

$$logRet_t = log\left(\frac{price_t}{price_{t-1}}\right)$$

Log-returns, through logarithm calculation, reveal the proportional change in stock value rather than the absolute change. There are many reasons why they are used, among the most significant [62]:

- *Additivity and Linearity*: they are additive, namely, with their sum we get the total log-return with respect to the period under consideration. The same is not true with simple returns.
- *Symmetry*: opposite values are equidistant from zero, making comparisons and analysis easier.
- *Time Aggregation*: by virtue of point 1, higher granularities can be obtained from a time perspective. In other words, one can easily switch from daily to weekly log-returns by a simple sum.
- *Interpretability*: log-returns can be thought of as growth rates that are continuously compounded. A log return of 0.03 (3%), for example, can be viewed as a 3% compounded growth rate during the provided time period. This interpretation is important for determining the relative change in an stock's value over time.

- *Normalization*: the values are normally distributed, and this in many statistical contexts is a great advantage because of the properties that characterize this type of distribution.

All data collected cover the period between 2010 and 2022 with daily frequency.

7.1.2 Training and Testing

Training

As mentioned earlier, the training phase consists of two moments: encoder training and regressor training. Regarding the former, the encoder receives as input the dataset containing the log-returns (univariate and multivariate) from 2010 to 2018. The training is carried out in all cases for 200 epochs. During the latter, one regressor is trained on the embeddings output from the encoder, another on the most important features extracted from XGBoost. In all cases Ridge receives an 80-20 split for train and validation tests. Figure 7.2 represents the training period from the perspective of both the index and the respective logarithmic returns. These will actually be given as input to the different contrastive models.

Figure 7.2: Univariate Training set, 2010-2018

Testing

The testing phase is carried out on 2019, 2020, 2021, and 2022. The cardinality of the test set varies slightly by year but is around 245 observations.
Equivalently to above, Figures 7.3, 7.4, 7.5 and 7.6, represent the time series of the index and respective log-returns in different years. As can be observed, all possible market conditions were chosen: bullish, bearish and mixed.

Figure 7.3: Univariate Test set 1, 2019

Figure 7.4: Univariate Test set 2, 2020

Figure 7.5: Univariate Test set 3, 2021

Figure 7.6: Univariate Test set 4, 2022

7.1.3 Metrics

The metrics used to quantify the regression results are *mean squared error* and *mean absolute error*. On the other hand, for gap assessment, *percentage difference* was used.

MSE

The *mean squared error* (MSE) represents the average squared difference between the predicted and true values. Mathematically it is defined as:

$$MSE = \frac{\sum(y_i - \hat{y}_i)^2}{n}$$

where y_i is the i-th observed value, $\hat{y}_i$ is the corresponding predicted value and n is the number of observations. Squaring increases the impact of larger errors. These calculations disproportionately penalize larger errors more than smaller errors.

MAE

The *mean squared error* (MSE) represents the average absolute difference between the predicted and true values. Mathematically it is defined as:

$$MAE = \frac{\sum|(y_i - \hat{y}_i)|}{n}$$

It is easily interpretable in fact it represents the average magnitude of errors in the original units of the target variable. Unlike the MSE it treats all errors equally with the risk of not capturing the effect of extreme errors.

Percentage Difference

This is not a metric for evaluating errors but a method for assessing the performance gap between the outputs of two different models. It is defined as:

$$perc\_diff = \frac{a - b}{\frac{a+b}{2}}$$

where a and b are the performances expressed both in terms of MSE or MAE.

7.1.4 Hardware and computation time

Processor: 2.6 GHz Intel Core i7 6 core
Memory: 16 GB 2400 MHz DDR4

GPU: Intel UHD Graphics 630 1536 MB

Table 7.1: Execution times

	Training time (min)	Encoding time (sec)
TNC (uni)	14	1
TNC (multi)	61	1
CoST (uni)	280	300
CoST (multi)	280	480
TS2Vec (uni)	18	2
TS2Vec (multi)	21	3
Time2State (uni)	6	19
Time2State (multi)	9	19

Table 7.1 shows the execution times for each phase associated with the individual models. Training time is expressed in minutes while encoding time is expressed in seconds.
As we can observe, there are clearly differences, in terms of training, between univariate and univariate models. In contrast, the time required for encoding remains almost unchanged. The only model that requires prohibitive time for both phases is CoST.
The light-weighted models creation module occurs following the execution of all contrastive models. In particular, one script is executed for each year and modality, taking 50 minutes. It includes all configurations of k clusters and thr threshold for filtering relevant features.

7.2 Univariate Forecasting

This first analysis concerns the univariate case. We now compare the performance of contrastive models with respect to technical indicators. These first analyses contain a limited number of comments to avoid repetitiveness. Detailed considerations and reasoning will be made in light of all the results, starting from section 7.5.
The results for the *technical indicators* represent the best scores from all clustering obtained with a different k (i.e. number of clusters) and all configurations for the threshold filtering most important features. The effect of these parameters will be investigated in the following paragraphs.
Gap statistics will refer to the MSE, as this metric is more sensitive to large errors.

In addition, a negative gap indicates that contrastive models perform better. Since the results represent errors, it is worth mentioning that in all graphical representations, the top performers are associated with the lowest height bars.

2019

As can be seen from Tables 7.2, 7.3, 7.4, 7.5, models that generate deep representations perform better, but often with a gap that is not too wide. In fact, in 83% of cases, the gap is less than 5%; in 91% of cases, it is less than 10%. Overall, we can observe that Time2state overperforms the other models with respect to the 3- and 5-day horizons, aligning instead in the 7-day case. The other three models are fairly aligned.

Table 7.2: Comparison between Time2State and Technical Indicators in 2019

	Time2State	TI	% gap
MSE 3	1.39E-04	1.57E-04	-12.27%
MSE 5	1.43E-04	1.57E-04	-9.35%
MSE 7	1.56E-04	1.57E-04	-0.48%
MAE 3	8.41E-03	8.88E-03	-5.44%
MAE 5	8.72E-03	8.89E-03	-1.93%
MAE 7	8.87E-03	8.90E-03	-0.37%

Table 7.3: Comparison between TS2Vec and Technical Indicators in 2019

	TS2Vec	TI	% gap
MSE 3	1.56E-04	1.58E-04	-1.35%
MSE 5	1.56E-04	1.57E-04	-0.70%
MSE 7	1.56E-04	1.58E-04	-1.46%
MAE 3	8.87E-03	8.96E-03	-1.02%
MAE 5	8.88E-03	8.92E-03	-0.41%
MAE 7	8.86E-03	8.99E-03	-1.54%

Table 7.4: Comparison between CoST and Technical Indicators in 2019

	CoST	TI	% gap
MSE 3	1.57E-04	1.57E-04	-0.25%
MSE 5	1.56E-04	1.58E-04	-1.19%
MSE 7	1.56E-04	1.58E-04	-1.27%
MAE 3	8.84E-03	8.92E-03	-0.91%
MAE 5	8.85E-03	8.98E-03	-1.43%
MAE 7	8.84E-03	9.01E-03	-1.84%

Table 7.5: Comparison between TNC and Technical Indicators in 2019

	TNC	TI	% gap
MSE 3	1.56E-04	1.58E-04	-1.48%
MSE 5	1.56E-04	1.58E-04	-1.28%
MSE 7	1.55E-04	1.58E-04	-1.73%
MAE 3	8.87E-03	8.95E-03	-0.86%
MAE 5	8.87E-03	8.97E-03	-1.12%
MAE 7	8.84E-03	8.97E-03	-1.47%

2020

2020 was a definitely pathological year for any sector. In particular, as we can clearly see in Figure 7.4, the trend has been noticeably irregular, with dips and rises dictated by the evolution of restrictions imposed to counter the spread of COVID-19. Consequently, as expected, the results reported in Tables 7.6, 7.7, 7.8, 7.9 are generally worse than in 2019. As in the previous case, deep representations perform better, but with a larger gap. In other words, in this mixed market condition, it is more convenient to use contrastive models. Specifically: in 50% of cases the gap is less than 10%, in the other 50% of cases it is above 10% but below 16%.
The same pattern observed in 2019 is evident: Time2State performances are equalized only with respect to the 7-day horizon.

Table 7.6: Comparison between Time2State and Technical Indicators in 2020

	Time2State	TI	% gap
MSE 3	5.12E-04	6.00E-04	-15.77%
MSE 5	5.04E-04	5.95E-04	-16.66%
MSE 7	5.36E-04	5.75E-04	-7.02%
MAE 3	1.51E-02	1.65E-02	-9.20%
MAE 5	1.53E-02	1.57E-02	-2.29%
MAE 7	1.53E-02	1.59E-02	-3.98%

Table 7.7: Comparison between TS2Vec and Technical Indicators in 2020

	TS2Vec	TI	% gap
MSE 3	5.34E-04	6.00E-04	-11.58%
MSE 5	5.34E-04	5.89E-04	-9.75%
MSE 7	5.34E-04	5.85E-04	-9.10%
MAE 3	1.52E-02	1.66E-02	-8.73%
MAE 5	1.53E-02	1.58E-02	-3.46%
MAE 7	1.52E-02	1.56E-02	-2.53%

Table 7.8: Comparison between CoST and Technical Indicators in 2020

	CoST	TI	% gap
MSE 3	5.26E-04	5.83E-04	-10.38%
MSE 5	5.35E-04	5.73E-04	-6.80%
MSE 7	5.29E-04	5.64E-04	-6.50%
MAE 3	1.51E-02	1.60E-02	-6.23%
MAE 5	1.52E-02	1.53E-02	-1.13%
MAE 7	1.51E-02	1.53E-02	-1.22%

Table 7.9: Comparison between TNC and Technical Indicators in 2020

	TNC	TI	% gap
MSE 3	5.27E-04	6.00E-04	-13.03%
MSE 5	5.26E-04	5.86E-04	-10.73%
MSE 7	5.26E-04	5.80E-04	-9.66%
MAE 3	1.51E-02	1.60E-02	-5.49%
MAE 5	1.51E-02	1.56E-02	-3.50%
MAE 7	1.51E-02	1.57E-02	-4.21%

2021

In terms of absolute performance, all models seem to perform much better than in 2020. As we can observe in Tables 7.10, 7.11, 7.12, 7.13, 2021 is rather difficult to interpret. Only in 33% of cases is the MSE gap below 5%, in 91% it does not exceed 9%, and in just 1 case the percentage difference is above 10%.
Once again the trend of Time2State is confirmed.
In addition, for the first time in some cases there is a convenience (not insignificant) of using technical indicators. In particular, both CoST and TNC are outperformed with a gap in MSE of at least 5%.

Table 7.10: Comparison between Time2State and Technical Indicators in 2021

	Time2State	TI	% gap
MSE 3	1.20E-04	1.27E-04	-6.00%
MSE 5	1.23E-04	1.25E-04	-1.61%
MSE 7	1.36E-04	1.27E-04	6.84%
MAE 3	8.29E-03	8.57E-03	-3.26%
MAE 5	8.40E-03	8.44E-03	-0.52%
MAE 7	8.79E-03	8.79E-03	0.02%

Table 7.11: Comparison between TS2Vec and Technical Indicators in 2021

	TS2Vec	TI	% gap
MSE 3	1.36E-04	1.38E-04	-1.28%
MSE 5	1.36E-04	1.38E-04	-1.01%
MSE 7	1.34E-04	1.30E-04	2.76%
MAE 3	8.75E-03	9.18E-03	-4.82%
MAE 5	8.79E-03	9.21E-03	-4.69%
MAE 7	8.68E-03	8.73E-03	-0.63%

Table 7.12: Comparison between CoST and Technical Indicators in 2021

	CoST	TI	% gap
MSE 3	1.37E-04	1.28E-04	6.64%
MSE 5	1.36E-04	1.25E-04	8.53%
MSE 7	1.36E-04	1.18E-04	14.04%
MAE 3	8.84E-03	8.55E-03	3.40%
MAE 5	8.82E-03	8.43E-03	4.48%
MAE 7	8.81E-03	8.22E-03	6.91%

Table 7.13: Comparison between TNC and Technical Indicators in 2021

	TNC	TI	% gap
MSE 3	1.36E-04	1.28E-04	5.70%
MSE 5	1.35E-04	1.24E-04	8.10%
MSE 7	1.35E-04	1.27E-04	6.28%
MAE 3	8.79E-03	8.71E-03	0.90%
MAE 5	8.74E-03	8.47E-03	3.15%
MAE 7	8.74E-03	8.65E-03	0.97%

2022

As we can see from Tables 7.14, 7.15, 7.16, 7.17, the results for 2022 are worse in terms of performance than for 2019 and 2021. In terms of MSE percentual gap, in

11 over 12 cases the gap is less than 5%, in 100% it is less than 10%.
As in all other cases, Time2State performs better than 3-day and 5-day. The percentage gaps, in general, are quite low and, except in the case of Time2State show a very slight convenience in using technical indicators.

Table 7.14: Comparison between Time2State and Technical Indicators in 2022

	Time2State	TI	% gap
MSE 3	3.91E-04	4.28E-04	-9.00%
MSE 5	4.13E-04	4.25E-04	-2.85%
MSE 7	4.34E-04	4.32E-04	0.47%
MAE 3	1.58E-02	1.64E-02	-3.60%
MAE 5	1.65E-02	1.64E-02	0.51%
MAE 7	1.67E-02	1.65E-02	1.25%

Table 7.15: Comparison between TS2Vec and Technical Indicators in 2022

	TS2Vec	TI	% gap
MSE 3	4.34E-04	4.29E-04	1.19%
MSE 5	4.32E-04	4.25E-04	1.81%
MSE 7	4.32E-04	4.31E-04	0.06%
MAE 3	1.68E-02	1.64E-02	2.06%
MAE 5	1.67E-02	1.64E-02	1.76%
MAE 7	1.67E-02	1.64E-02	1.55%

Table 7.16: Comparison between CoST and Technical Indicators in 2022

	CoST	TI	% gap
MSE 3	4.24E-04	4.25E-04	-0.19%
MSE 5	4.30E-04	4.24E-04	1.49%
MSE 7	4.30E-04	4.24E-04	1.28%
MAE 3	1.65E-02	1.63E-02	1.06%
MAE 5	1.67E-02	1.63E-02	2.11%
MAE 7	1.67E-02	1.63E-02	2.22%

Table 7.17: Comparison between TNC and Technical Indicators in 2022

	TNC	TI	% gap
MSE 3	4.28E-04	4.27E-04	0.23%
MSE 5	4.29E-04	4.27E-04	0.42%
MSE 7	4.31E-04	4.31E-04	0.10%
MAE 3	1.66E-02	1.64E-02	1.22%
MAE 5	1.66E-02	1.64E-02	1.29%
MAE 7	1.67E-02	1.65E-02	1.52%

In Figures 7.7, 7.8, 7.9, one can clearly see the trend of Time2State, regardless of the year, for which it dominates on the first two horizons and aligns on the 7-day. Notably, it is not the other models that reach Time2State, but it is the latter that is worse on the latter horizon.
It is also worth mentioning that CoST and TNC are often comparable and TS2Vec is usually the worst in terms of performance.

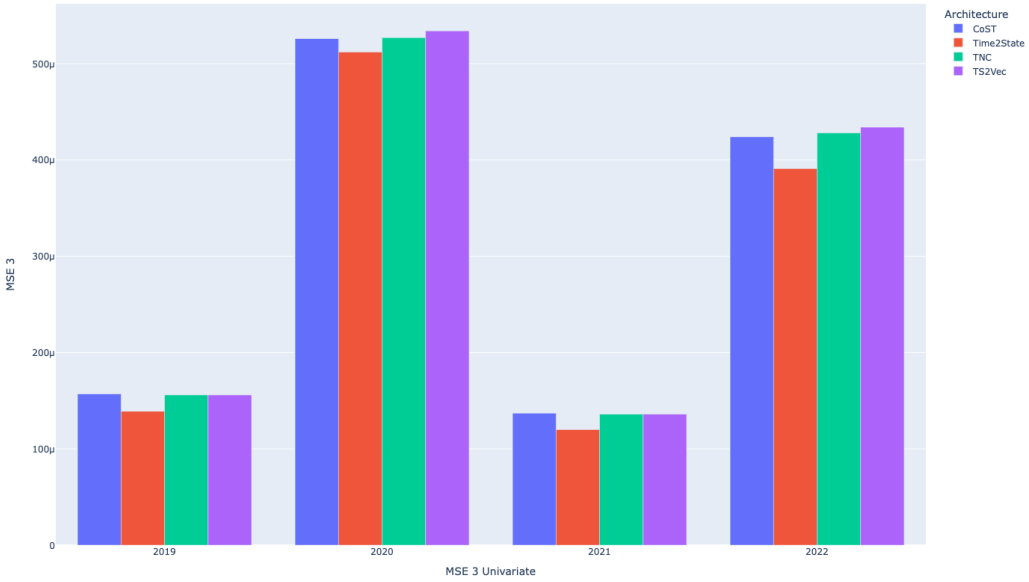


Figure 7.7: MSE 3 Univariate contrastive

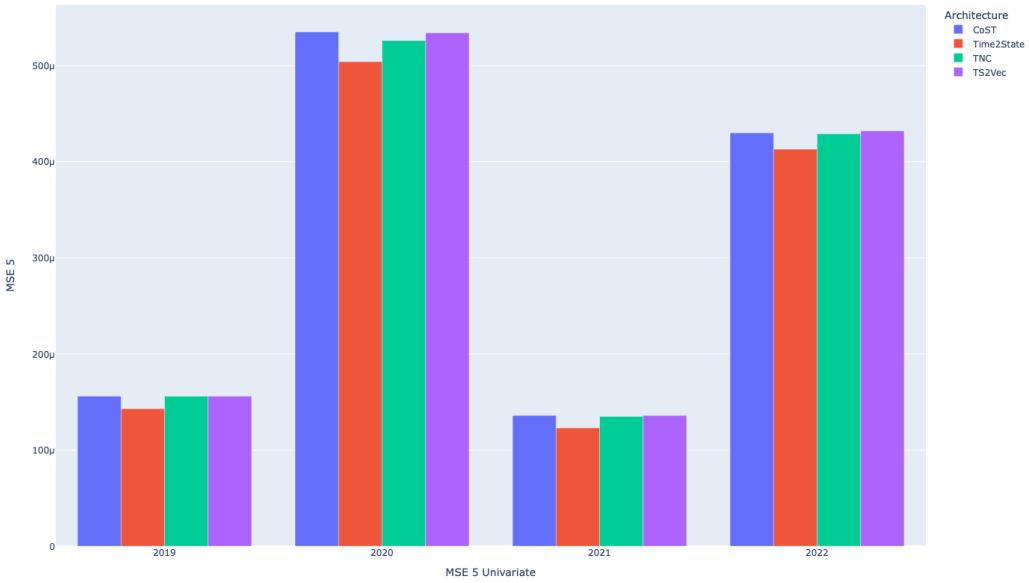


Figure 7.8: MSE 5 Univariate contrastive

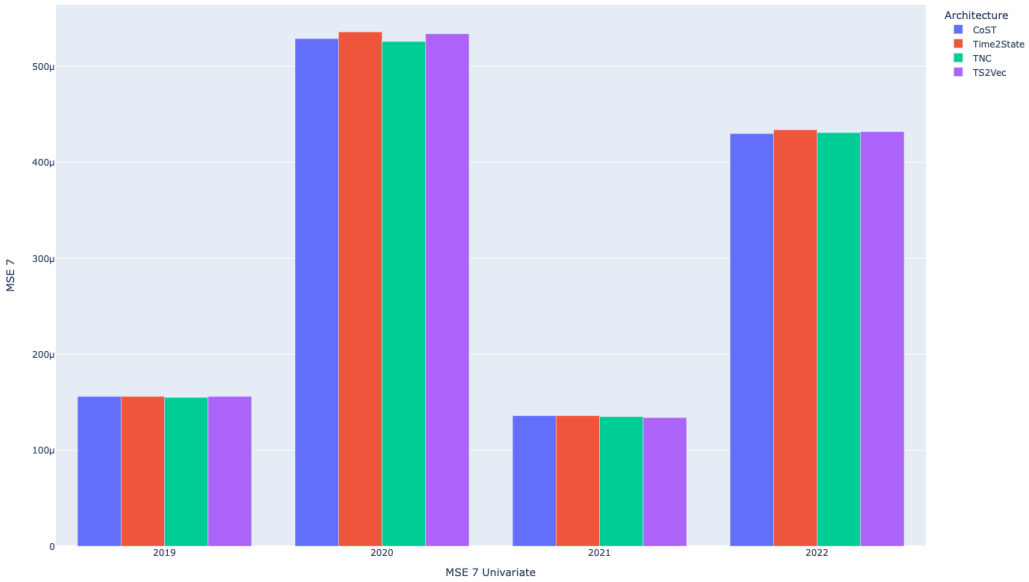


Figure 7.9: MSE 7 Univariate contrastive

Figures 7.10, 7.11, 7.12 depict the comparison of the best model associated

with the univariate case, namely Time2State, versus the respective light-weighted models.
Specifically, the boxplots represent all the results of the technical indicators associated with Time2State, relatively to the dataset under consideration. The stars, on the other hand, represent the results of Time2State itself. In this case we can clearly observe that the greatest advantage in using the contrastive method occurs in 2020, while in 2021 the technical indicators manage to prevail. Moreover, the smallest gap between the different approaches is observed in the prediction of the seventh day.

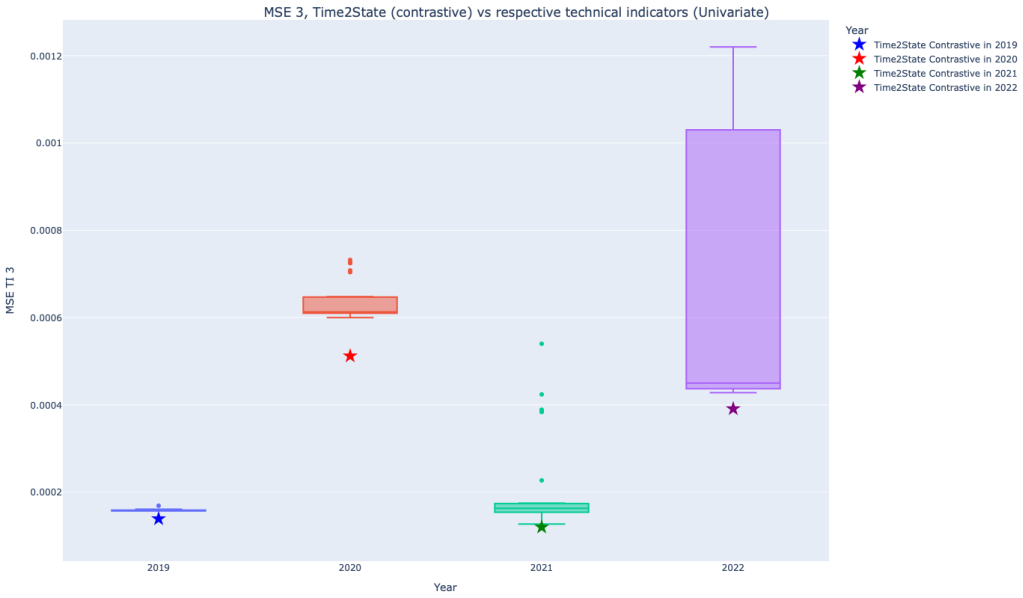


Figure 7.10: MSE 3 Univariate, Time2State vs technical indicators

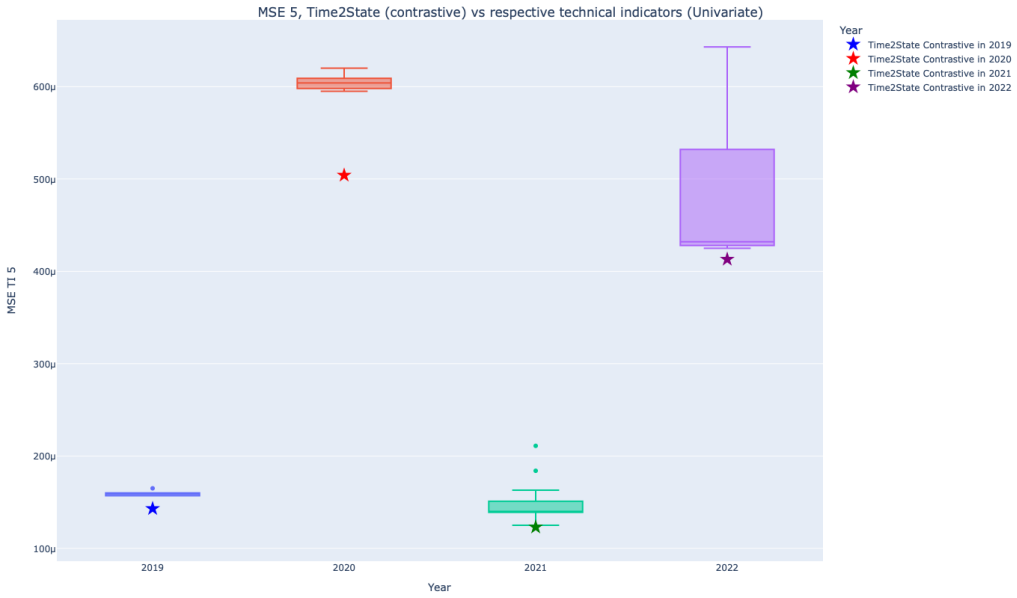


Figure 7.11: MSE 5 Univariate, Time2State vs technical indicators

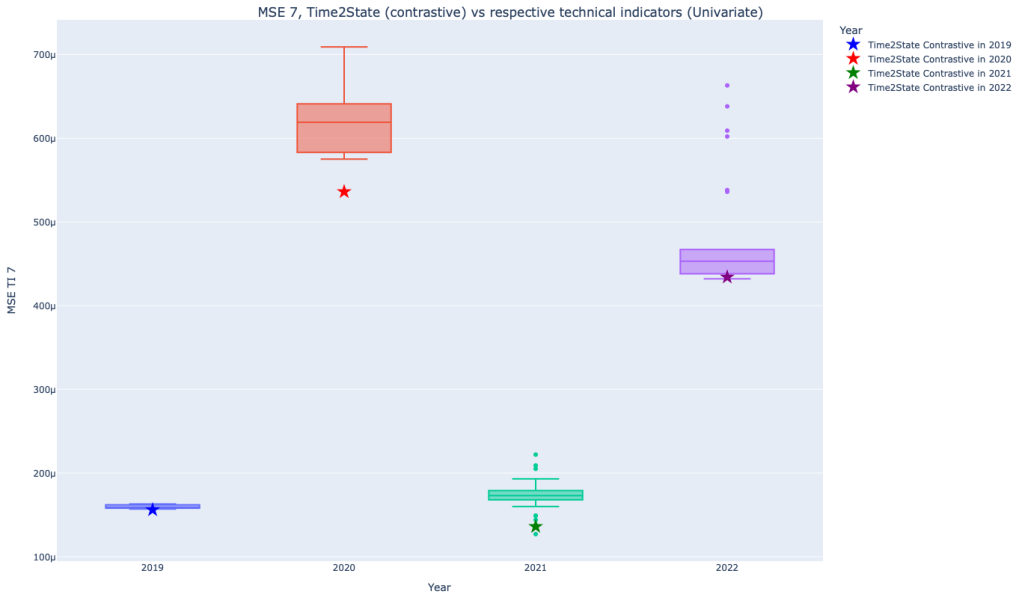


Figure 7.12: MSE 7 Univariate, Time2State vs technical indicators

7.3 Multivariate Forecasting

This section reports results for the multivariate case that comprises both a dataset with only stocks and a dataset that also includes the index.

7.3.1 Stocks only dataset

2019

As can be observed from Tables 7.18, 7.19, 7.20, 7.21, Time2State is the model that performs best and has the largest percentage gaps in absolute terms compared to the corresponding results obtained with the technical indicators. The other models are at about the same level. In all cases except one (CoST with respect to 3-day horizon), there is a slight convenience of using contrastive methods.

Table 7.18: Comparison between Time2State and Technical Indicators in 2019

	Time2State	TI	% gap
MSE 3	1.31E-04	1.57E-04	-17.97%
MSE 5	1.30E-04	1.58E-04	-19.37%
MSE 7	1.34E-04	1.58E-04	-16.35%
MAE 3	8.61E-03	8.91E-03	-3.43%
MAE 5	8.60E-03	8.96E-03	-4.03%
MAE 7	8.74E-03	8.98E-03	-2.66%

Table 7.19: Comparison between TS2Vec and Technical Indicators in 2019

	TS2Vec	TI	% gap
MSE 3	1.57E-04	1.57E-04	-0.51%
MSE 5	1.56E-04	1.58E-04	-1.53%
MSE 7	1.56E-04	1.58E-04	-1.37%
MAE 3	8.90E-03	8.96E-03	-0.58%
MAE 5	8.86E-03	9.01E-03	-1.69%
MAE 7	8.90E-03	9.04E-03	-1.52%

Table 7.20: Comparison between CoST and Technical Indicators in 2019

	CoST	TI	% gap
MSE 3	1.58E-04	1.57E-04	0.82%
MSE 5	1.54E-04	1.57E-04	-1.46%
MSE 7	1.56E-04	1.57E-04	-0.51%
MAE 3	8.91E-03	8.90E-03	0.03%
MAE 5	8.84E-03	8.90E-03	-0.72%
MAE 7	8.89E-03	8.91E-03	-0.23%

Table 7.21: Comparison between TNC and Technical Indicators in 2019

	TNC	TI	% gap
MSE 3	1.49E-04	1.57E-04	-5.20%
MSE 5	1.55E-04	1.57E-04	-1.13%
MSE 7	1.56E-04	1.57E-04	-0.29%
MAE 3	8.60E-03	8.88E-03	-3.15%
MAE 5	8.85E-03	8.92E-03	-0.80%
MAE 7	8.87E-03	8.90E-03	-0.30%

2020

In this case, Time2State's tendency to perform better on the short periods (3 and 5 days) and then align with the other contrastive models is again confirmed (Tables 7.22, 7.23, 7.24, 7.25). In fact, it is even outperformed by both CoST and TNC. In all cases there is a clear convenience in using contrastive models instead of technical indicators.

Table 7.22: Comparison between Time2State and Technical Indicators in 2020

	Time2State	TI	% gap
MSE 3	5.36E-04	6.03E-04	-11.87%
MSE 5	5.27E-04	5.99E-04	-12.83%
MSE 7	5.41E-04	5.88E-04	-8.36%
MAE 3	1.53E-02	1.63E-02	-6.20%
MAE 5	1.52E-02	1.60E-02	-5.14%
MAE 7	1.53E-02	1.59E-02	-3.85%

Table 7.23: Comparison between TS2Vec and Technical Indicators in 2020

	TS2Vec	TI	% gap
MSE 3	5.41E-04	6.03E-04	-10.78%
MSE 5	5.43E-04	5.99E-04	-9.80%
MSE 7	5.46E-04	5.87E-04	-7.27%
MAE 3	1.54E-02	1.63E-02	-5.63%
MAE 5	1.55E-02	1.60E-02	-3.09%
MAE 7	1.57E-02	1.59E-02	-1.41%

Table 7.24: Comparison between CoST and Technical Indicators in 2020

	CoST	TI	% gap
MSE 3	5.41E-04	5.90E-04	-8.65%
MSE 5	5.47E-04	5.88E-04	-7.34%
MSE 7	5.39E-04	5.76E-04	-6.74%
MAE 3	1.53E-02	1.59E-02	-3.79%
MAE 5	1.55E-02	1.61E-02	-4.08%
MAE 7	1.55E-02	1.60E-02	-3.15%

Table 7.25: Comparison between TNC and Technical Indicators in 2020

	TNC	TI	% gap
MSE 3	5.52E-04	5.98E-04	-7.98%
MSE 5	5.34E-04	5.98E-04	-11.40%
MSE 7	5.37E-04	5.86E-04	-8.82%
MAE 3	1.53E-02	1.61E-02	-4.91%
MAE 5	1.54E-02	1.61E-02	-4.83%
MAE 7	1.54E-02	1.61E-02	-4.05%

2021

Similar to what occurred in 2019, Tables 7.26, 7.27, 7.28, 7.29 show that Time2State prevails over all horizons. In addition, the respective model based on technical indicators in turn prevails over the 5- and 7-day, with a not insignificant gap. In the case of TNC and CoST, the technical indicators perform better in each case, with gaps between 2% and 7%.

Table 7.26: Comparison between Time2State and Technical Indicators in 2021

	Time2State	TI	% gap
MSE 3	1.31E-04	1.33E-04	-1.63%
MSE 5	1.30E-04	1.27E-04	2.16%
MSE 7	1.34E-04	1.23E-04	8.60%
MAE 3	8.61E-03	8.67E-03	-0.68%
MAE 5	8.60E-03	8.53E-03	-0.80%
MAE 7	8.74E-03	8.43E-03	3.64%

Table 7.27: Comparison between TS2Vec and Technical Indicators in 2021

	TS2Vec	TI	% gap
MSE 3	1.34E-04	1.42E-04	-5.60%
MSE 5	1.38E-04	1.47E-04	-6.21%
MSE 7	1.37E-04	1.29E-04	6.03%
MAE 3	8.74E-03	9.29E-03	-6.08%
MAE 5	8.82E-03	9.56E-03	-5.26%
MAE 7	8.81E-03	8.68E-03	1.46%

Table 7.28: Comparison between CoST and Technical Indicators in 2021

	CoST	TI	% gap
MSE 3	1.40E-04	1.36E-04	2.82%
MSE 5	1.38E-04	1.30E-04	6.00%
MSE 7	1.36E-04	1.26E-04	7.53%
MAE 3	8.95E-03	8.76E-03	2.14%
MAE 5	8.87E-03	8.72E-03	1.69%
MAE 7	8.74E-03	8.61E-03	1.47%

Table 7.29: Comparison between TNC and Technical Indicators in 2021

	TNC	TI	% gap
MSE 3	1.34E-04	1.32E-04	1.64%
MSE 5	1.37E-04	1.33E-04	3.15%
MSE 7	1.37E-04	1.29E-04	5.54%
MAE 3	8.76E-03	8.62E-03	1.52%
MAE 5	8.80E-03	8.88E-03	2.03%
MAE 7	8.79E-03	8.76E-03	0.37%

2022

Even in 2022, results from Tables 7.30, 7.31, 7.32, 7.33 prove that Time2State is the top performer only in the second horizon, while being outperformed by all

other models in the 7-day case and by TNC in the 3-day horizon. Gaps are very small with a slight convenience to using technical indicators.

Table 7.30: Comparison between Time2State and Technical Indicators in 2022

	Time2State	TI	% gap
MSE 3	4.14E-04	4.33E-04	-4.62%
MSE 5	4.12E-04	4.28E-04	-3.87%
MSE 7	4.36E-04	4.34E-04	0.27%
MAE 3	1.63E-02	1.66E-02	-1.49%
MAE 5	1.64E-02	1.65E-02	-0.14%
MAE 7	1.68E-02	1.66E-02	1.13%

Table 7.31: Comparison between TS2Vec and Technical Indicators in 2022

	TS2Vec	TI	% gap
MSE 3	4.35E-04	4.32E-04	0.76%
MSE 5	4.33E-04	4.30E-04	0.77%
MSE 7	4.32E-04	4.31E-04	0.12%
MAE 3	1.67E-02	1.65E-02	1.05%
MAE 5	1.67E-02	1.64E-02	1.62%
MAE 7	1.67E-02	1.65E-02	1.08%

Table 7.32: Comparison between CoST and Technical Indicators in 2022

	CoST	TI	% gap
MSE 3	4.27E-04	4.30E-04	-0.83%
MSE 5	4.34E-04	4.28E-04	1.29%
MSE 7	4.32E-04	4.30E-04	0.46%
MAE 3	1.65E-02	1.65E-02	0.25%
MAE 5	1.67E-02	1.64E-02	1.70%
MAE 7	1.67E-02	1.65E-02	1.35%

Table 7.33: Comparison between TNC and Technical Indicators in 2022

	TNC	TI	% gap
MSE 3	4.10E-04	4.33E-04	-5.47%
MSE 5	4.29E-04	4.32E-04	-0.69%
MSE 7	4.31E-04	4.31E-04	0.02%
MAE 3	1.62E-02	1.65E-02	-1.74%
MAE 5	1.66E-02	1.65E-02	0.85%
MAE 7	1.67E-02	1.65E-02	1.29%

As we can observe from Figures 7.13, 7.14, 7.15, even in this first multivariate case, the tendency of Time2State is to dominate in the prediction of the next 3 and 5 days. The only exception is 2019 where the model is also dominant in the longest horizon.

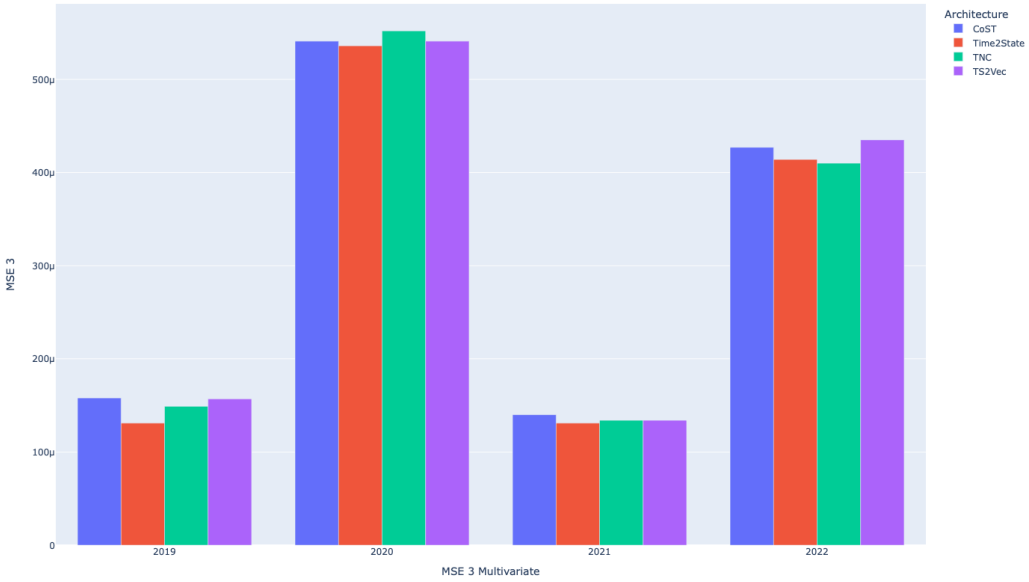


Figure 7.13: MSE 3 Multivariate contrastive

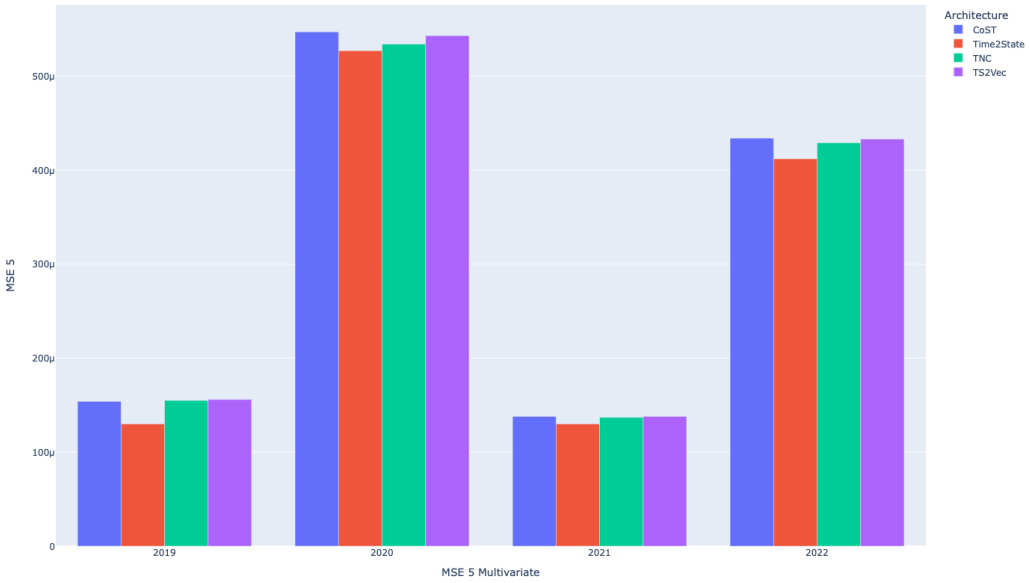


Figure 7.14: MSE 5 Multivariate contrastive

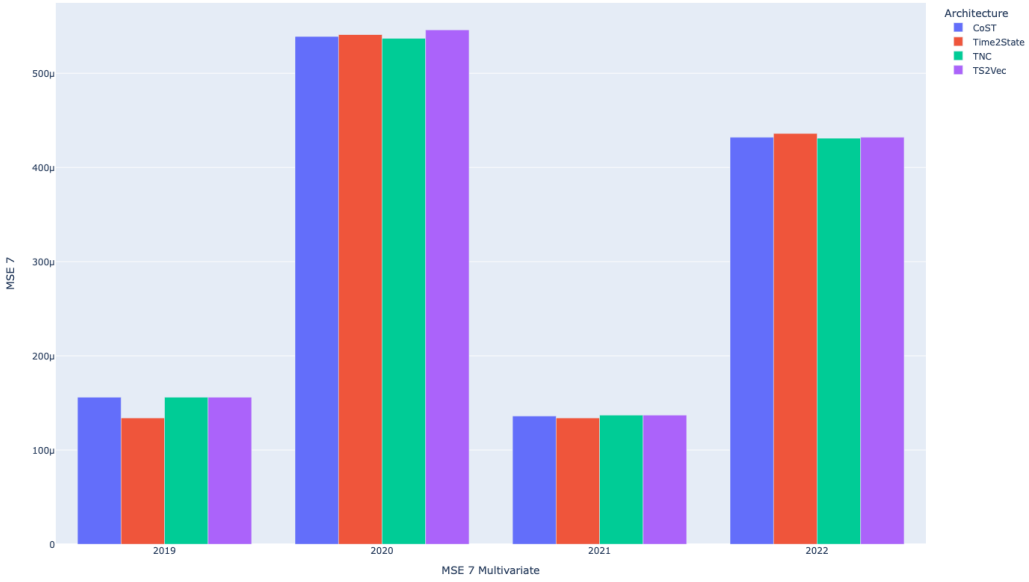


Figure 7.15: MSE 7 Multivariate contrastive

In Figures 7.16, 7.17, 7.18, Time2State is analyzed individually since it is the

overall top performer. Once again we observe how 2020 is visibly better handled by the contrastive model, while in 2021, especially on long horizons, technical indicators prevail.
The most noticeable difference with respect to the univariate case is the lower variance relative to technical indicators in 2022, at all three horizons.

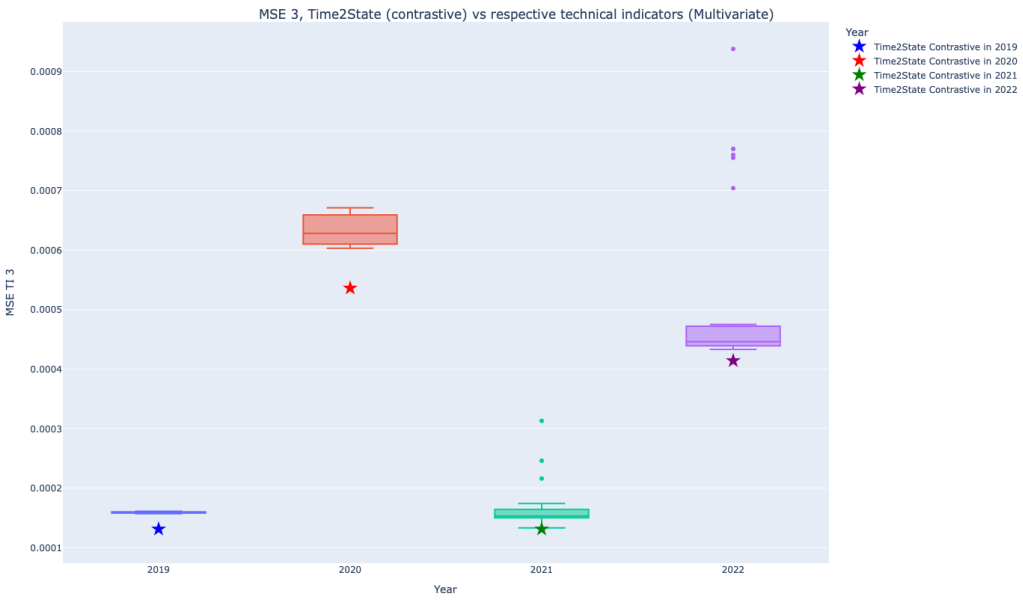


Figure 7.16: MSE 3 Multivariate, Time2State vs technical indicators

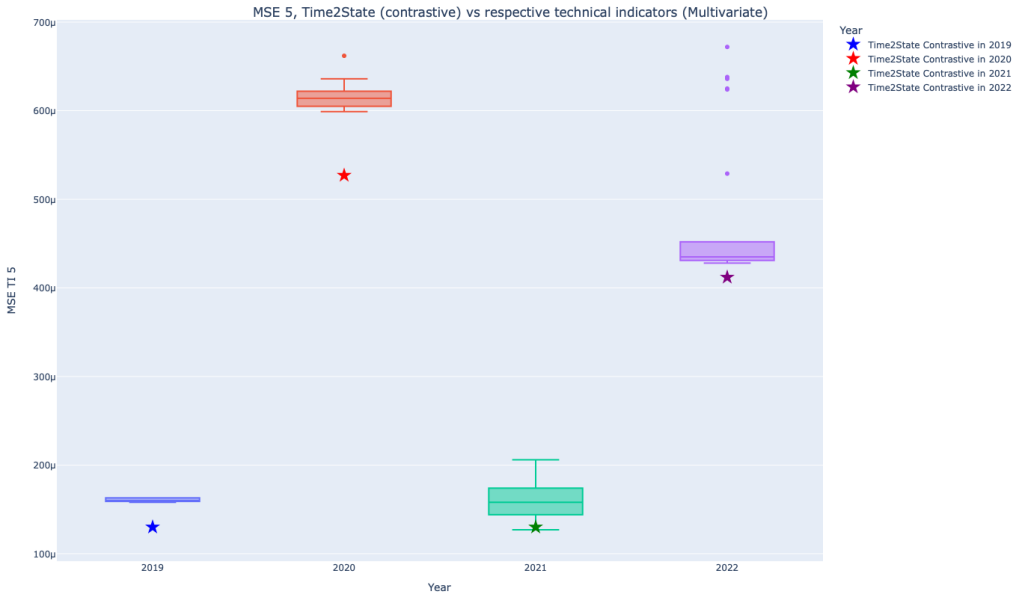


Figure 7.17: MSE 5 Multivariate, Time2State vs technical indicators

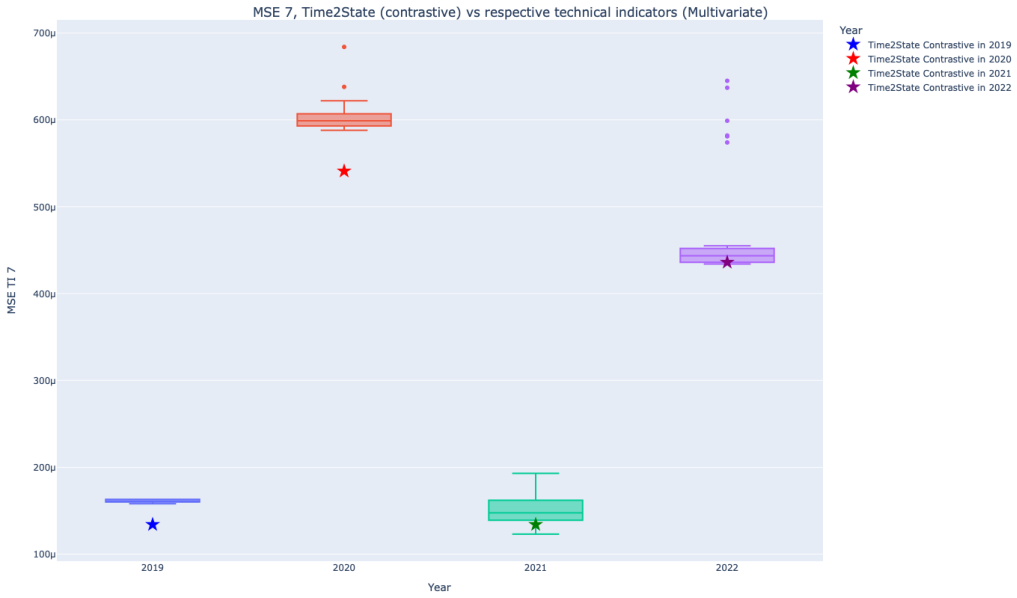


Figure 7.18: MSE 7 Multivariate, Time2State vs technical indicators

7.3.2 Stocks with index dataset

2019

Looking at the results in Tables 7.34, 7.35, 7.36, 7.37 we can observe that it is always convenient to use contrastive methods even if, often, with not too large gaps. In this case the top performer on the 3 days is TNC, on the 5 is Time2State while on the 7 all models are aligned.

Table 7.34: Comparison between Time2State and Technical Indicators in 2019

	Time2State	TI	% gap
MSE 3	1.40E-04	1.57E-04	-11.43%
MSE 5	1.42E-04	1.58E-04	-10.21%
MSE 7	1.56E-04	1.58E-04	-1.35%
MAE 3	8.45E-03	8.91E-03	-5.21%
MAE 5	8.64E-03	8.96E-03	-3.64%
MAE 7	8.87E-03	8.98E-03	-1.23%

Table 7.35: Comparison between TS2Vec and Technical Indicators in 2019

	TS2Vec	TI	% gap
MSE 3	1.56E-04	1.57E-04	-1.00%
MSE 5	1.56E-04	1.58E-04	-1.68%
MSE 7	1.56E-04	1.58E-04	-1.34%
MAE 3	8.86E-03	8.96E-03	-1.13%
MAE 5	8.85E-03	9.02E-03	-1.81%
MAE 7	8.89E-03	9.04E-03	-1.60%

Table 7.36: Comparison between CoST and Technical Indicators in 2019

	CoST	TI	% gap
MSE 3	1.57E-04	1.57E-04	-0.06%
MSE 5	1.56E-04	1.57E-04	-0.69%
MSE 7	1.56E-04	1.57E-04	-0.51%
MAE 3	8.87E-03	8.90E-03	-0.38%
MAE 5	8.84E-03	8.90E-03	-0.70%
MAE 7	8.89E-03	8.91E-03	-0.27%

Table 7.37: Comparison between TNC and Technical Indicators in 2019

	TNC	TI	% gap
MSE 3	1.35E-04	1.57E-04	-5.20%
MSE 5	1.56E-04	1.57E-04	-1.13%
MSE 7	1.56E-04	1.57E-04	-0.29%
MAE 3	8.25E-03	8.88E-03	-3.15%
MAE 5	8.85E-03	8.92E-03	-0.80%
MAE 7	8.87E-03	8.90E-03	-0.30%

2020

As we can observe from Tables 7.38, 7.39, 7.40, 7.41, in 2020 it is definitely preferable to rely on contrastive methods, since gaps always exceed 7%. Again TNC is confirmed as the top performer on 3 days, while on 5 and 7 days Time2State and CoST prevail respectively.

Table 7.38: Comparison between Time2State and Technical Indicators in 2020

	Time2State	TI	% gap
MSE 3	5.25E-04	6.06E-04	-14.33%
MSE 5	5.13E-04	6.01E-04	-15.79%
MSE 7	5.44E-04	5.87E-04	-7.76%
MAE 3	1.52E-02	1.64E-02	-7.70%
MAE 5	1.51E-02	1.59E-02	-5.39%
MAE 7	1.54E-02	1.58E-02	-3.02%

Table 7.39: Comparison between TS2Vec and Technical Indicators in 2020

	TS2Vec	TI	% gap
MSE 3	5.41E-04	6.05E-04	-11.29%
MSE 5	5.43E-04	6.02E-04	-10.29%
MSE 7	5.45E-04	5.91E-04	-8.06%
MAE 3	1.54E-02	1.62E-02	-4.80%
MAE 5	1.55E-02	1.61E-02	-4.12%
MAE 7	1.56E-02	1.60E-02	-2.27%

Table 7.40: Comparison between CoST and Technical Indicators in 2020

	CoST	TI	% gap
MSE 3	5.32E-04	5.91E-04	-10.57%
MSE 5	5.46E-04	5.89E-04	-7.57%
MSE 7	5.35E-04	5.78E-04	-7.67%
MAE 3	1.52E-02	1.59E-02	-4.64%
MAE 5	1.55E-02	1.61E-02	-3.83%
MAE 7	1.54E-02	1.59E-02	-3.16%

Table 7.41: Comparison between TNC and Technical Indicators in 2020

	TNC	TI	% gap
MSE 3	5.19E-04	5.99E-04	-14.43%
MSE 5	5.32E-04	5.95E-04	-11.15%
MSE 7	5.36E-04	5.84E-04	-8.62%
MAE 3	1.48E-02	1.61E-02	-8.06%
MAE 5	1.53E-02	1.60E-02	-4.31%
MAE 7	1.54E-02	1.57E-02	-1.59%

2021

Regarding 2021, TNC remains the best in predicting the third day, Time2State in predicting the fifth and seventh. In general, a preference in the use of technical indicators is observed, particularly pronounced in the case of the 7-day. All these results can be observed in Tables 7.42, 7.43, 7.44, 7.45

Table 7.42: Comparison between Time2State and Technical Indicators in 2021

	Time2State	TI	% gap
MSE 3	1.39E-04	1.32E-04	4.64%
MSE 5	1.35E-04	1.27E-04	6.32%
MSE 7	1.34E-04	1.22E-04	9.16%
MAE 3	9.00E-03	8.71E-03	3.20%
MAE 5	8.86E-03	8.62E-03	2.76%
MAE 7	8.76E-03	8.49E-03	3.13%

Table 7.43: Comparison between TS2Vec and Technical Indicators in 2021

	TS2Vec	TI	% gap
MSE 3	1.34E-04	1.33E-04	0.97%
MSE 5	1.38E-04	1.37E-04	0.39%
MSE 7	1.38E-04	1.29E-04	6.46%
MAE 3	8.73E-03	8.71E-03	0.14%
MAE 5	8.81E-03	9.14E-03	-3.72%
MAE 7	8.79E-03	8.82E-03	-0.29%

Table 7.44: Comparison between CoST and Technical Indicators in 2021

	CoST	TI	% gap
MSE 3	1.40E-04	1.36E-04	2.85%
MSE 5	1.37E-04	1.30E-04	4.82%
MSE 7	1.37E-04	1.26E-04	8.57%
MAE 3	8.91E-03	8.74E-03	1.90%
MAE 5	8.81E-03	8.62E-03	2.22%
MAE 7	8.81E-03	8.49E-03	3.91%

Table 7.45: Comparison between TNC and Technical Indicators in 2021

	TNC	TI	% gap
MSE 3	1.23E-04	1.33E-04	0.97%
MSE 5	1.39E-04	1.37E-04	0.39%
MSE 7	1.37E-04	1.29E-04	6.46%
MAE 3	8.36E-03	8.71E-03	0.14%
MAE 5	8.86E-03	9.14E-03	-3.72%
MAE 7	8.81E-03	8.82E-03	-0.29%

2022

Looking at Tables 7.46, 7.47, 7.48, 7.49, we can observe that, in 2022, the top performers are TNC on the 3-day and Time2State on the 5- and 7-day horizons.

Gaps tend to be small (except for TNC over the 3-day), sometimes in favor of the technical indicators.

Table 7.46: Comparison between Time2State and Technical Indicators in 2022

	Time2State	TI	% gap
MSE 3	4.17E-04	4.32E-04	-3.51%
MSE 5	4.27E-04	4.27E-04	-0.05%
MSE 7	4.35E-04	4.34E-04	0.26%
MAE 3	1.64E-02	1.66E-02	-0.89%
MAE 5	1.67E-02	1.64E-02	1.68%
MAE 7	1.67E-02	1.65E-02	1.41%

Table 7.47: Comparison between TS2Vec and Technical Indicators in 2022

	TS2Vec	TI	% gap
MSE 3	4.36E-04	4.30E-04	1.37%
MSE 5	4.34E-04	4.31E-04	0.77%
MSE 7	4.34E-04	4.37E-04	-0.86%
MAE 3	1.68E-02	1.66E-02	1.22%
MAE 5	1.67E-02	1.65E-02	1.28%
MAE 7	1.67E-02	1.66E-02	0.78%

Table 7.48: Comparison between CoST and Technical Indicators in 2022

	CoST	TI	% gap
MSE 3	4.28E-04	4.30E-04	-0.40%
MSE 5	4.31E-04	4.28E-04	0.74%
MSE 7	4.30E-04	4.29E-04	0.44%
MAE 3	1.66E-02	1.65E-02	0.53%
MAE 5	1.66E-02	1.64E-02	1.31%
MAE 7	1.66E-02	1.65E-02	0.84%

Table 7.49: Comparison between TNC and Technical Indicators in 2022

	TNC	TI	% gap
MSE 3	3.63E-04	4.33E-04	-17.71%
MSE 5	4.30E-04	4.32E-04	-0.44%
MSE 7	4.32E-04	4.34E-04	-0.43%
MAE 3	1.51E-02	1.65E-02	-8.79%
MAE 5	1.66E-02	1.65E-02	0.80%
MAE 7	1.67E-02	1.65E-02	0.90%

In the case of the latter modality (Figures 7.19, 7.20, 7.21), TNC turns out to be the best performer by far on the 3 days, Time2State on the 5, while on the 7 there is an alignment that we have already noticed with previous datasets.

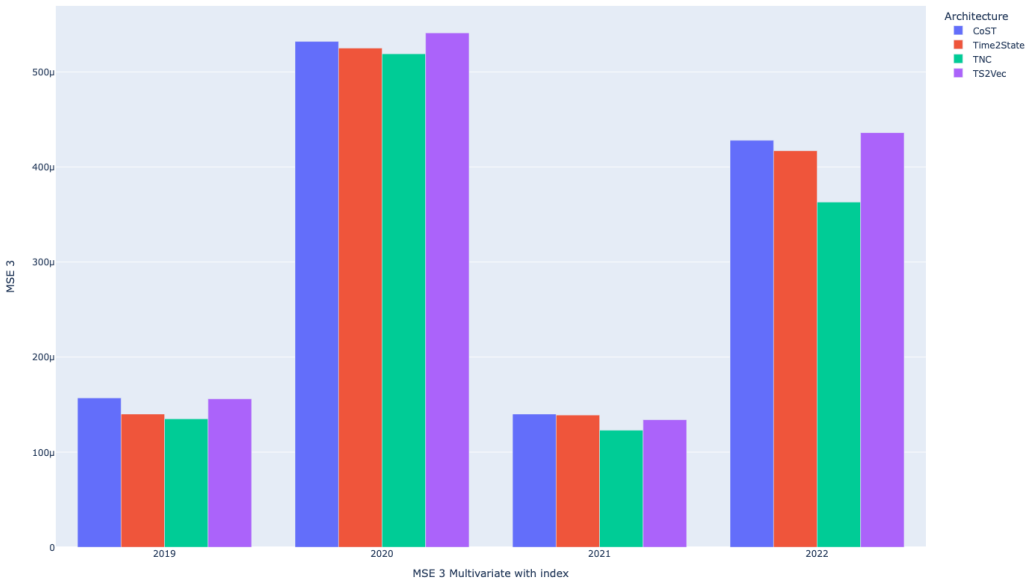


Figure 7.19: MSE 3 Multivariate (with index) contrastive

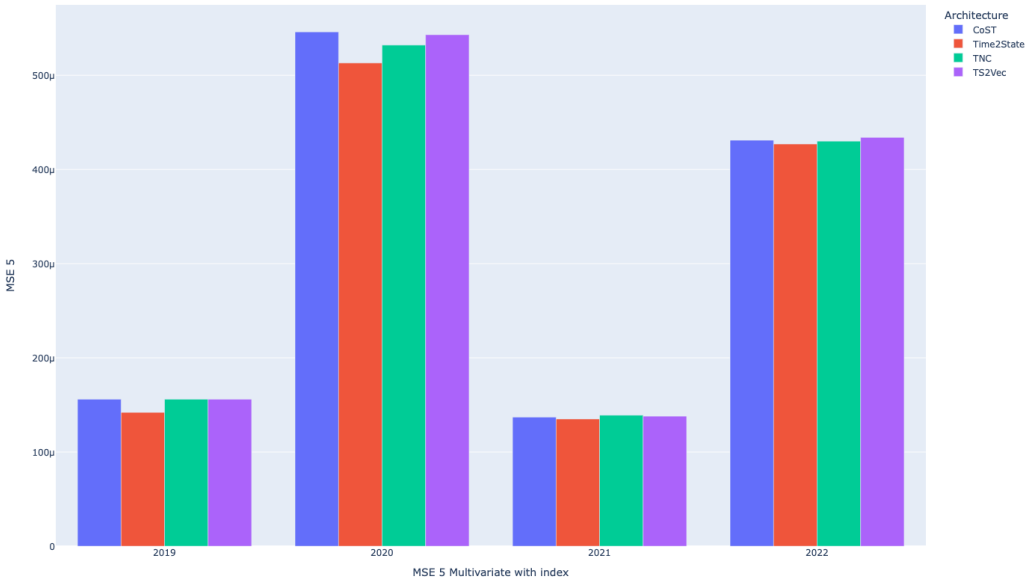


Figure 7.20: MSE 5 Multivariate (with index) contrastive

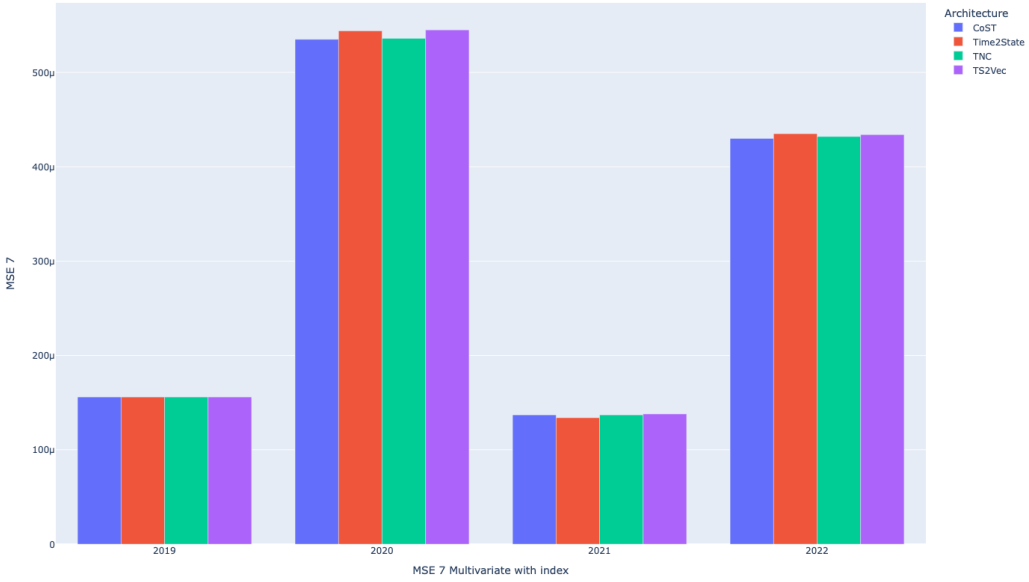


Figure 7.21: MSE 7 Multivariate (with index) contrastive

Once again we analyze through Figures 7.22, 7.23, 7.24 the behavior of Time2State

Again the considerations made earlier apply, such as that in 2020 there is more convenience in using the contrastive method and that in 2021 the best results are obtained with the technical indicators. Moreover, it is evident how as the horizon increases the gap, and thus the convenience, decreases.

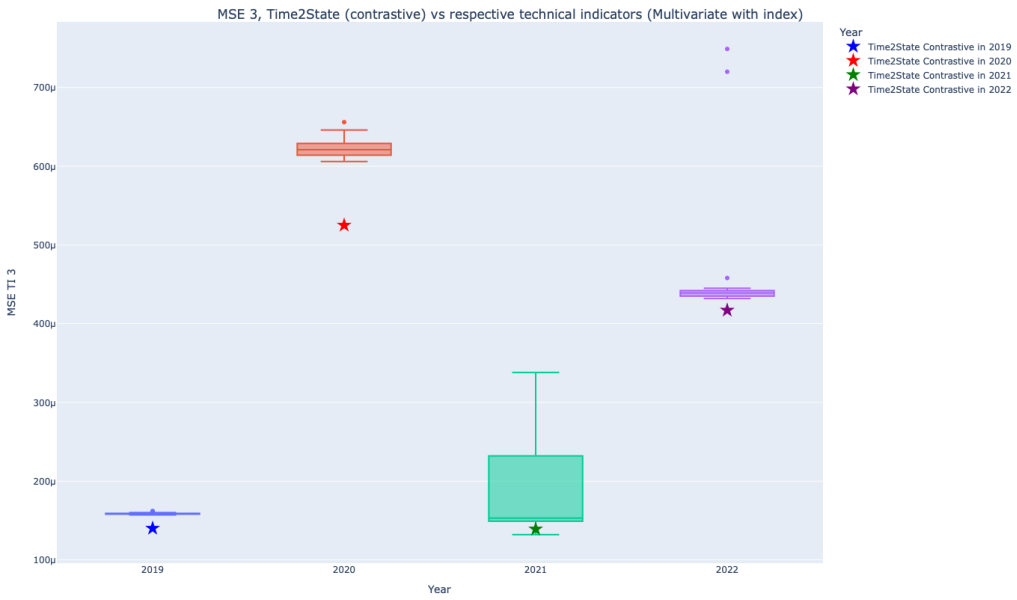


Figure 7.22: MSE 3 Multivariate with index, Time2State vs technical indicators

Figure 7.23: MSE 5 Multivariate with index, Time2State vs technical indicators

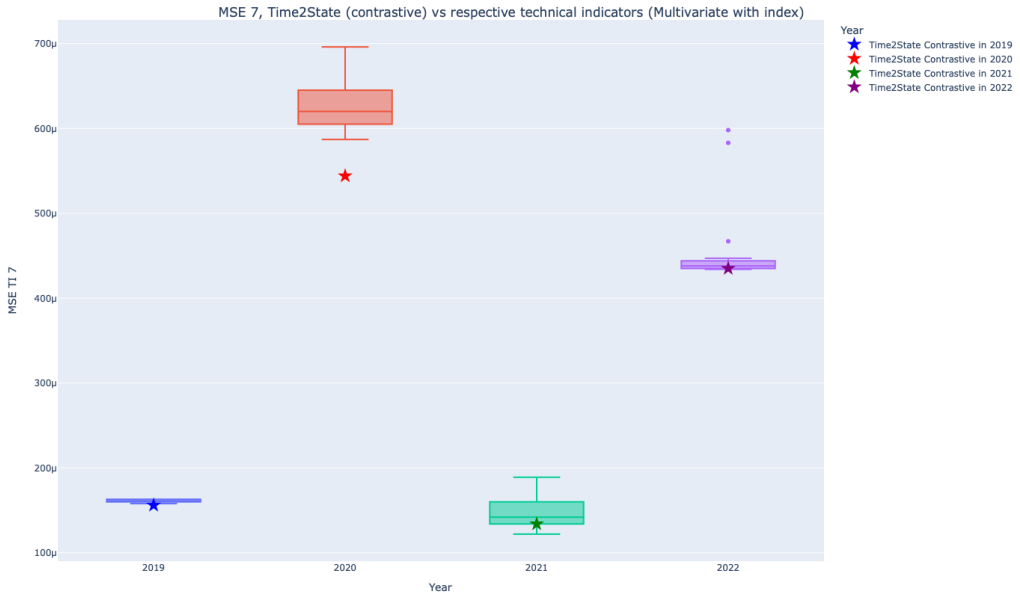


Figure 7.24: MSE 7 Multivariate with index, Time2State vs technical indicators

7.4 Differences between datasets

The most relevant differences for individual architectures will be reported below, again distinguishing the results by year. The analysis is done by distinguishing by model to better understand the peculiarities, associated with the individual case, resulting from the choice of a particular set of features.

In most cases Time2State gets better results with the univariate dataset, for each horizon. The only exception is 2019 where the multivariate dataset (without index) overperforms (Figures 7.25, 7.26, 7.27).

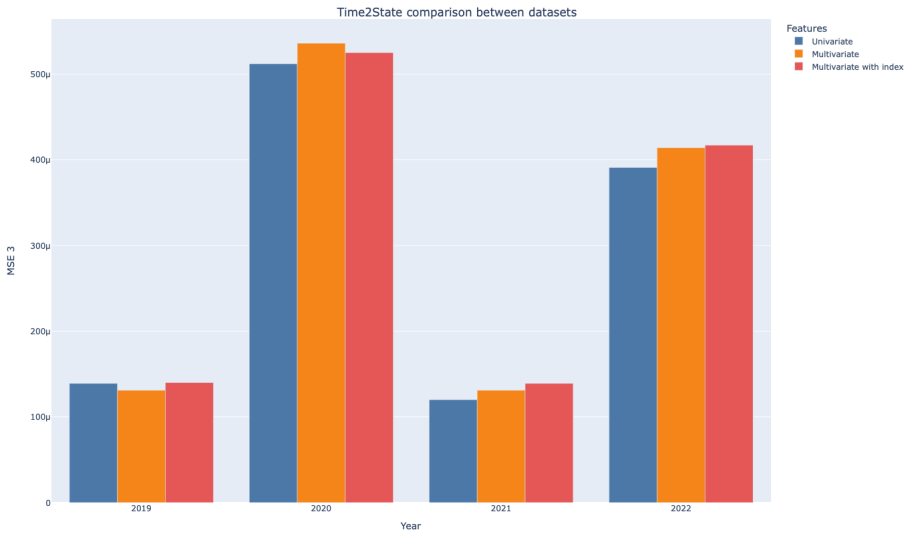


Figure 7.25: MSE 3 Time2State, comparison per dataset

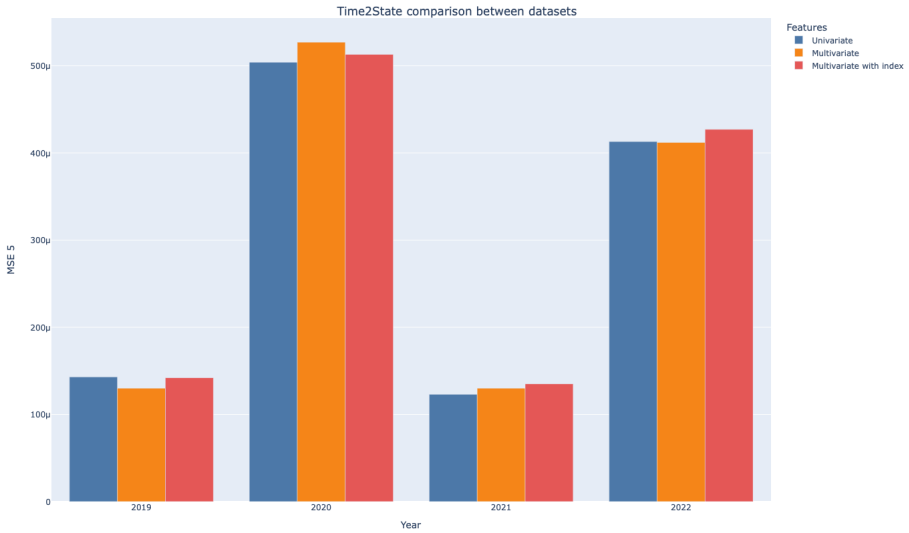


Figure 7.26: MSE 5 Time2State, comparison per dataset

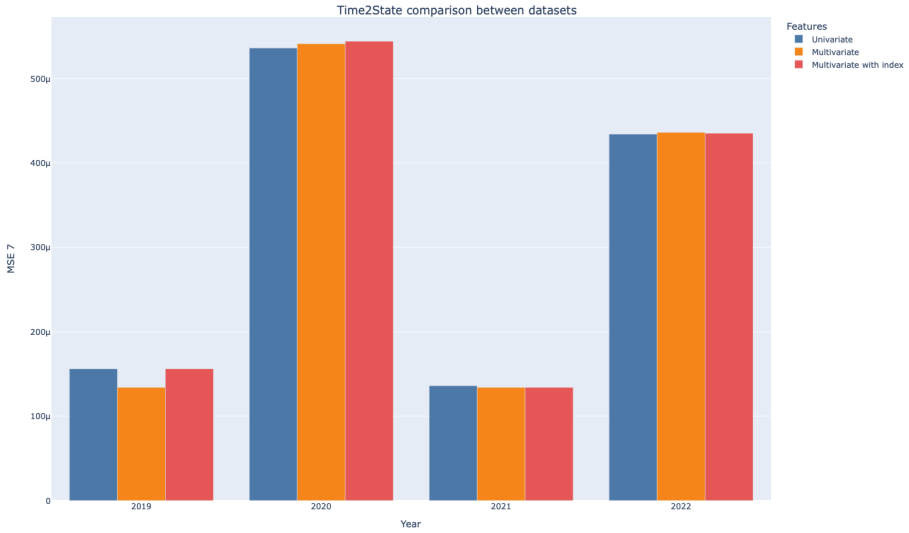


Figure 7.27: MSE 7 Time2State, comparison per dataset

As for TNC, in Figures 7.28, 7.29, 7.30 we observe a significant improvement in the results by adding the index, limiting to the 3-day horizon. Instead, the univariate dataset is often associated with the worst results.

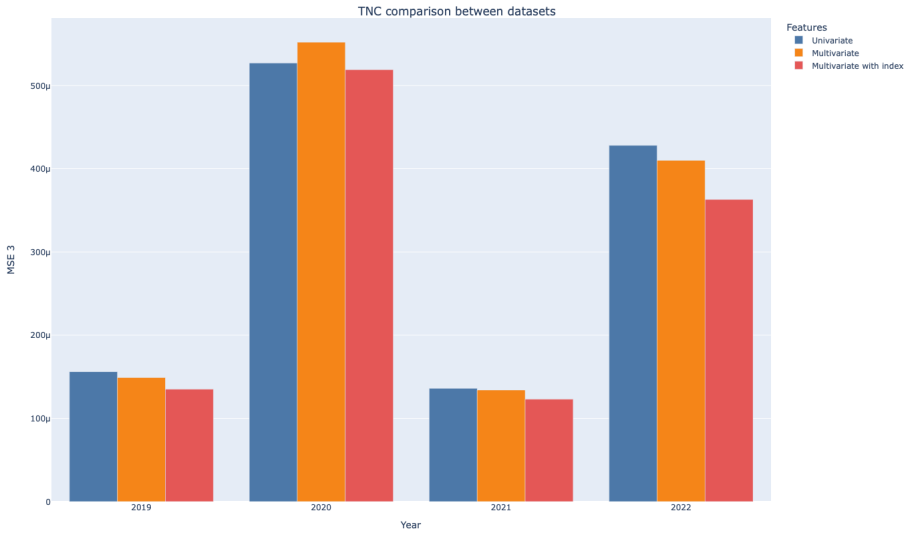


Figure 7.28: MSE 3 TNC, comparison per dataset

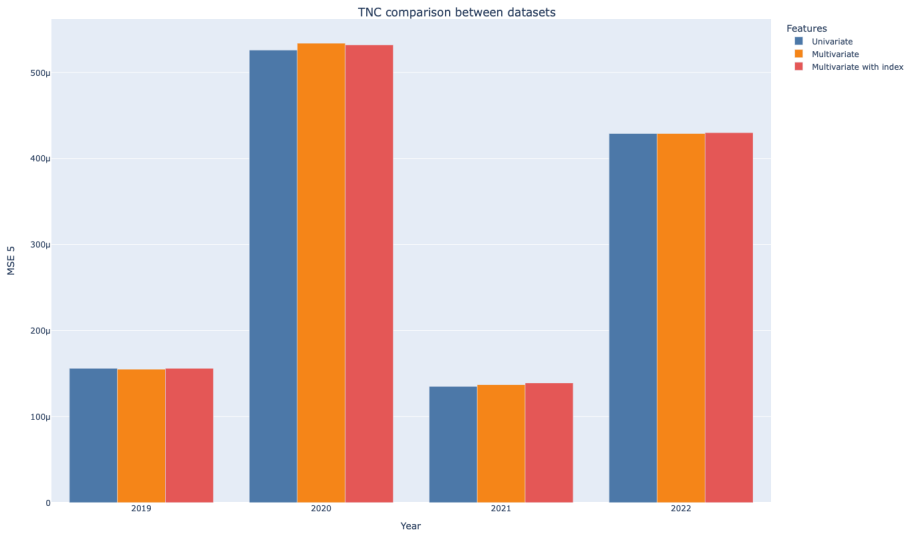


Figure 7.29: MSE 5 TNC, comparison per dataset

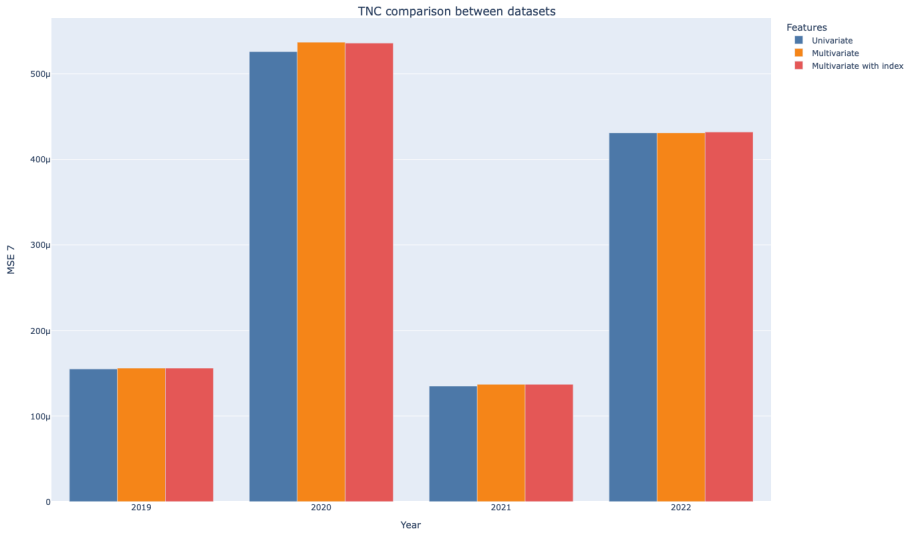


Figure 7.30: MSE 7 TNC, comparison per dataset

In the case of both Time2State and TNC, the differences tend to flatten out with the 7-day horizon.

7.5 Best performers and gaps

In light of the previous results, we can draw some general conclusions. As we have already verified, Time2State turns out to be the best contrastive model in most contexts. The primacy of this method, in particular, extends to the first two time horizons, namely the 3- and 5-day horizons. With respect to the 7-day, however, the models tend to align, in the sense that Time2State worsens its performance. Moreover, as the horizon increases, the gaps with respect to technical indicators decrease, highlighting how the convenience is limited to the short term. In addition, we could observe that in some contexts technical indicators are more convenient than their contrastive counterparts, justifying the usefulness of light-weighted models.
However, it is also useful to make distinctions by year to gain further insights.
In 2019 Time2State was the best performer (multivariate) outperforming its competitors on all horizons (Table 7.50). In general, it always pays to use contrastive models at the expense of technical indicators, especially in the case of Time2State.

Referring to 2020, the gaps in general are wider (i.e. lower negative values), as we can observe in Figures 7.31, 7.32, 7.33, denoting a greater ability of contrastive models to predict complicated situations such as the case of the crash due

to the COVID-19 pandemic outbreak. Again, Time2State returns the best results on the next 3 and 5 days, outperformed, however, by TNC on the 7 (univariate dataset in all cases, Table 7.50).

As for 2021, we observe for the first time the victory of technical indicators over their contrastive counterparts Table 7.51. Moreover, the convenience grows as the horizon increases; in fact, the 7-day horizon has rather large gaps that fall below 5% in only one case (Figures 7.31, 7.32, 7.33). However, the best performer on 3 and 5 days is confirmed to be Time2State (univariate) while on 7 days technical indicators (CoST univariate) prevail.

Finally, in 2022, rather small gaps are observed in Figures 7.31, 7.32, 7.33. One cannot identify a clear pattern, but certainly to predict the next 7 days it often pays, slightly, to use technical indicators. In general, however, the best performers are TNC (multivariate with index), Time2State (multivariate), and technical indicators (multivariate CoST) for 3-, 5-, and 7-day horizons, respectively (Table 7.51).

In addition, it is worth noting a pattern that occurs in 7 out of 12 cases: Time2State turns out to be the best contrastive performer with respect to 3- and 5-day, aligning with or even being outperformed on the 7-day. In all cases where this does not hold, TNC performs better, but only on the 3-day.

Table 7.50: Best absolute contrastive performers for different years

		Model	Dataset	Error
2019	MSE 3	Time2State	Multivariate	1.31E-04
	MSE 5	Time2State	Multivariate	1.30E-04
	MSE 7	Time2State	Multivariate	1.34E-04
2020	MSE 3	Time2State	Univariate	5.12E-04
	MSE 5	Time2State	Univariate	5.04E-04
	MSE 7	TNC	Univariate	5.26E-04
2021	MSE 3	Time2State	Univariate	1.20E-04
	MSE 5	Time2State	Univariate	1.23E-04
	MSE 7	TS2Vec	Univariate	1.34E-04
2022	MSE 3	TNC	Multivariate (with index)	3.63E-04
	MSE 5	Time2State	Multivariate	4.12E-04
	MSE 7	CoST	Univariate	4.30E-04

As we can observe in Table 7.51 among the best overall performers there are two light-weighted models, in both cases associated with CoST, related to the 7-day horizon.

Table 7.51: Best overall performers for different years

		Model	Dataset	Error
2019	MSE 3	Time2State	Multivariate	1.31E-04
	MSE 5	Time2State	Multivariate	1.30E-04
	MSE 7	Time2State	Multivariate	1.34E-04
2020	MSE 3	Time2State	Univariate	5.12E-04
	MSE 5	Time2State	Univariate	5.04E-04
	MSE 7	TNC	Univariate	5.26E-04
2021	MSE 3	Time2State	Univariate	1.20E-04
	MSE 5	Time2State	Univariate	1.23E-04
	MSE 7	TI (CoST)	Univariate	1.18E-04
2022	MSE 3	TNC	Multivariate (with index)	3.63E-04
	MSE 5	Time2State	Multivariate	4.12E-04
	MSE 7	TI (CoST)	Univariate	4.24E-04

Figures 7.31, 7.32, 7.33 represent the convenience of using contrastive methods with respect to technical indicators. In general, as the horizon increases, two patterns are observed.
First, the variance decreases, an aspect we already noted earlier. In other words, models, regardless of dataset, tend to have more pronounced differences for short time horizons and align with respect to 7 days.
Second, as the horizon increases, the convenience of contrastive models decreases. Visually, the boxes tend to approach positive values.
Moreover, making a distinction by year, it is evident that regardless of the time horizon, 2020 (mixed market) is the year in which contrastive methods perform better than the corresponding technical indicators. By contrast, in 2021 (bullish market) light-weighted models outperform deep models, sometimes by non-negligible margins (between 5% and 10%).
All of this, overall, can be interpreted as the ability of contrastive models to predict complex situations but to be outperformed when the trend is easier to predict.

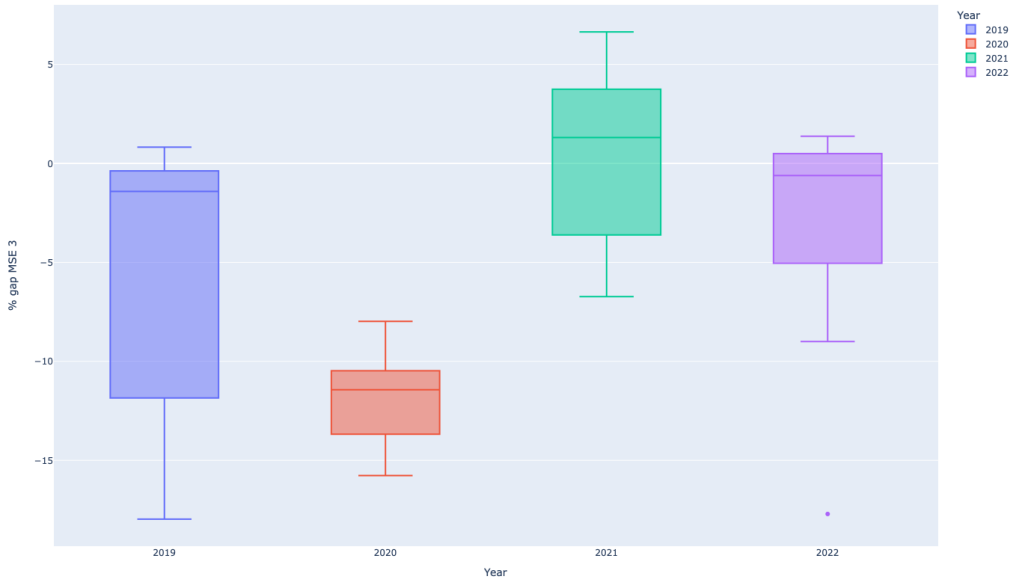


Figure 7.31: MSE 3 % gap between contrastive models and technical indicators

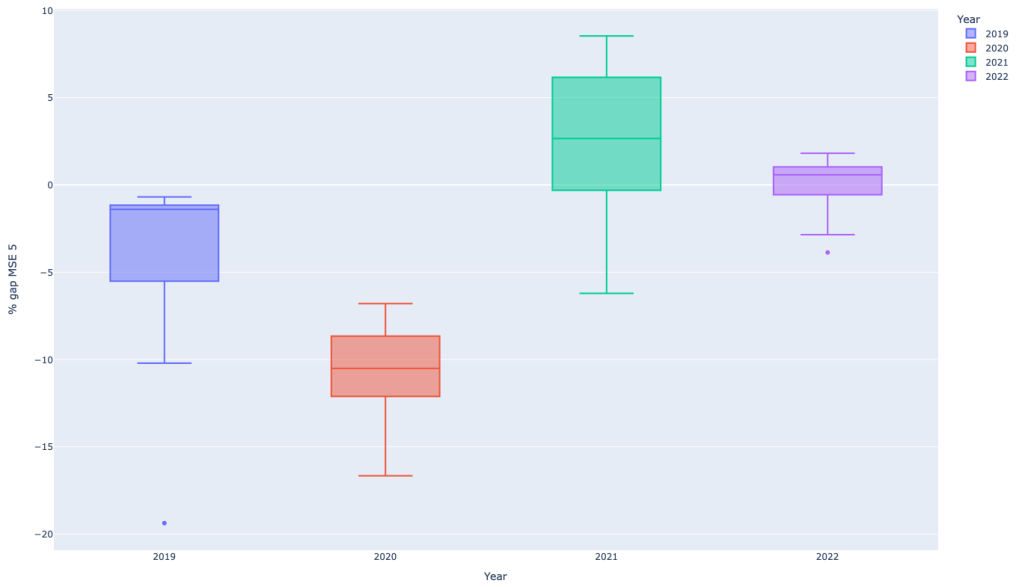


Figure 7.32: MSE 5 % gap between contrastive models and technical indicators

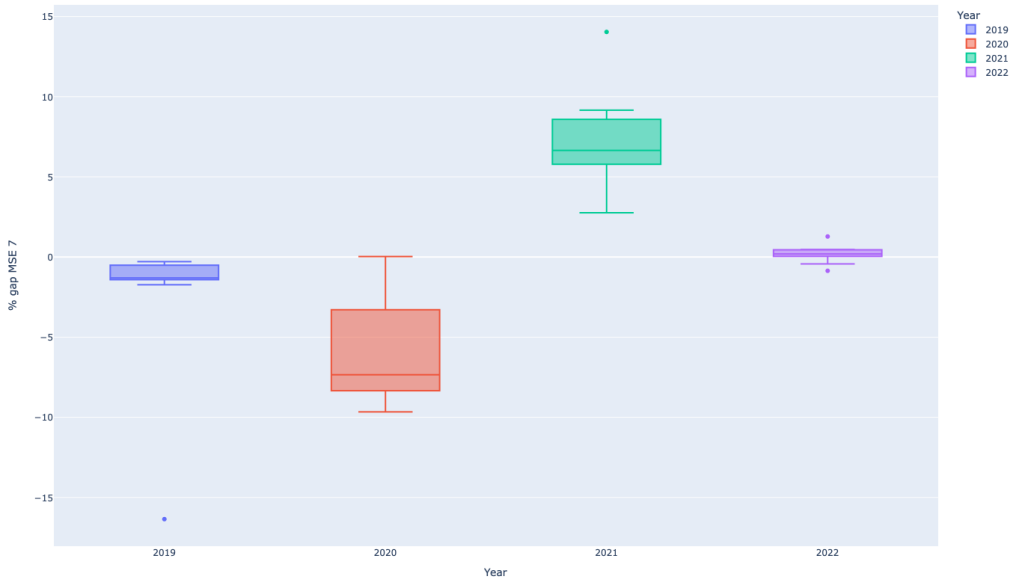


Figure 7.33: MSE 7 % gap between contrastive models and technical indicators

7.6 Effect of threshold

As we mentioned earlier, we make use of a threshold to discriminate the most important features and select them for light-weighted models. In this section we investigate the effect of this parameter, again in terms of MSE.
Recall that the value of the parameter does not fall below 0.5 because it represents a value below which no feature would be selected in certain cases.

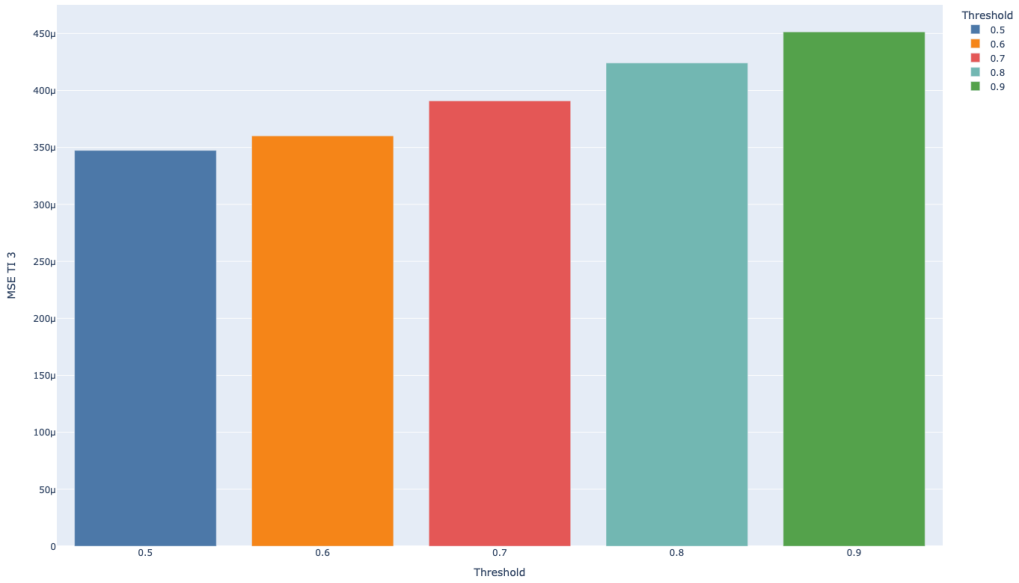


Figure 7.34: MSE 3, effect of Threshold

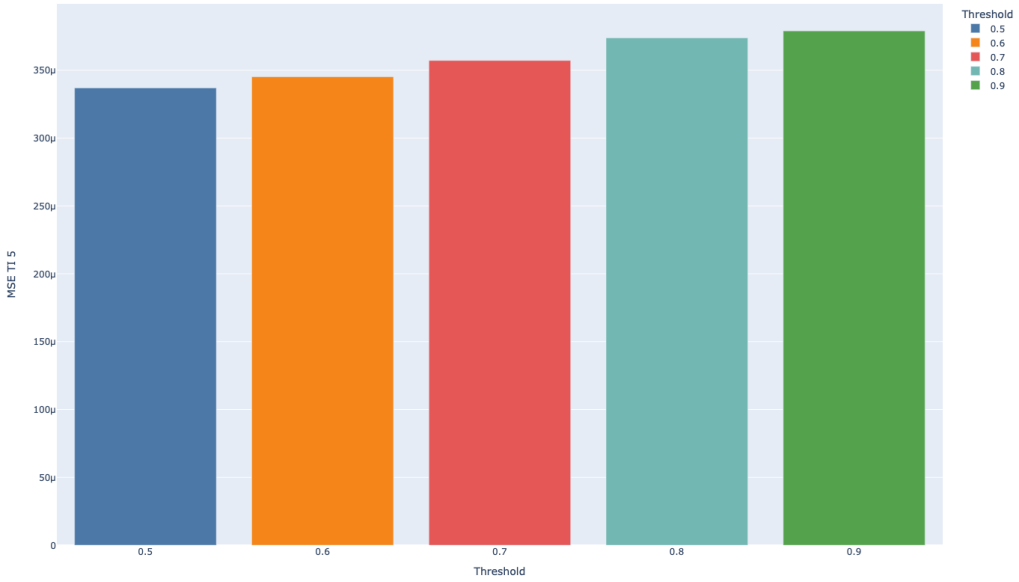


Figure 7.35: MSE 5, effect of Threshold

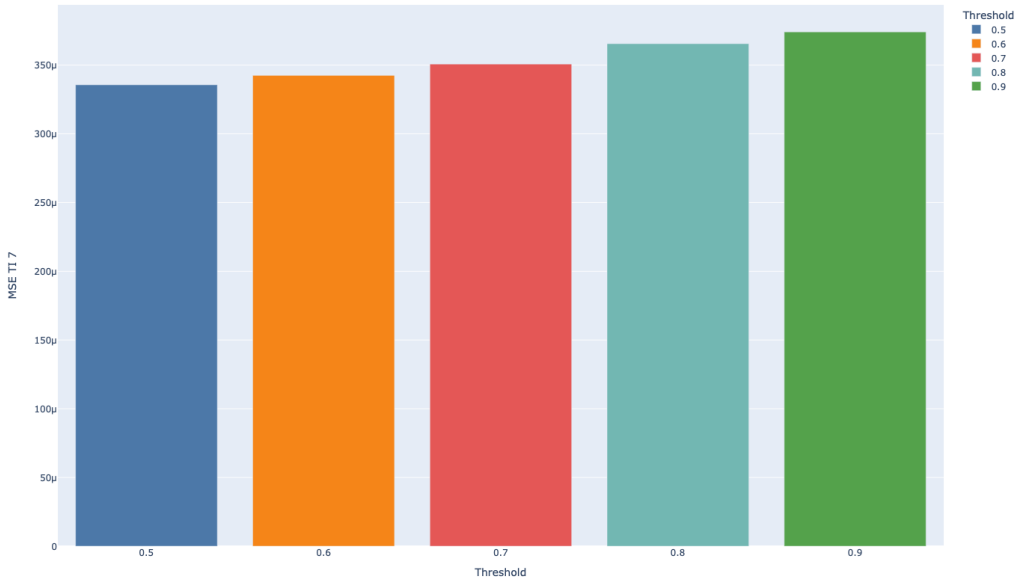


Figure 7.36: MSE 7, effect of Threshold

In Figures 7.34, 7.35, 7.36 we can observe that the error grows as the threshold increases in all three horizons. This may mean that keeping only the most relevant features avoids overfitting resulting in a greater ability to generalize.

7.7 Performance comparison vs Autoregressive models

In this section we analyze the difference in performance from a baseline, the autoregressive models mentioned earlier such as ARIMA and Exponential Smoothing.

Table 7.52: Autoregressive vs contrastive models in 2019

	Arima	Exp Sm	Best contrastive	% gap Arima	% gap Exp Sm
MSE 3	1.64E-04	1.68E-04	1.31E-04	-22.27%	-24.94%
MSE 5	1.65E-04	1.69E-04	1.30E-04	-23.97%	-26.01%
MSE 7	1.65E-04	1.69E-04	1.34E-04	-20.65%	-23.05%
MAE 3	9.01E-03	9.26E-03	8.61E-03	-4.54%	-7.29%
MAE 5	9.05E-03	9.27E-03	8.60E-03	-5.11%	-7.55%
MAE 7	9.07E-03	9.29E-03	8.74E-03	-3.71%	-6.12%

Table 7.53: Autoregressive vs contrastive models in 2020

	Arima	Exp Sm	Best contrastive	% gap Arima	% gap Exp Sm
MSE 3	9.06E-04	5.56E-04	5.12E-04	-55.60%	-8.16%
MSE 5	9.09E-04	5.60E-04	5.04E-04	-57.31%	-10.44%
MSE 7	9.14E-04	5.62E-04	5.26E-04	-53.93%	-6.57%
MAE 3	1.93E-02	1.54E-02	1.51E-02	-24.61%	-2.26%
MAE 5	1.93E-02	1.55E-02	1.51E-02	-24.25%	-2.77%
MAE 7	1.93E-02	1.55E-02	1.51E-02	-24.44%	-2.63%

Table 7.54: Autoregressive vs contrastive models in 2021

	Arima	Exp Sm	Best contrastive	% gap Arima	% gap Exp Sm
MSE 3	1.44E-04	1.42E-04	1.20E-04	-18.18%	-16.58%
MSE 5	1.41E-04	1.43E-04	1.23E-04	-13.93%	-14.98%
MSE 7	1.40E-04	1.42E-04	1.34E-04	-4.24%	-5.67%
MAE 3	8.80E-03	8.93E-03	8.29E-03	-5.94%	-7.47%
MAE 5	8.85E-03	8.97E-03	8.40E-03	-5.22%	-6.55%
MAE 7	8.85E-03	8.85E-03	8.68E-03	-1.94%	-1.99%

Table 7.55: Autoregressive vs contrastive models in 2022

	Arima	Exp Sm	Best contrastive	% gap Arima	% gap Exp Sm
MSE 3	4.51E-04	4.65E-04	3.63E-04	-21.54%	-24.64%
MSE 5	4.53E-04	4.68E-04	4.12E-04	-9.57%	-12.67%
MSE 7	4.64E-04	4.72E-04	4.30E-04	-7.59%	-9.23%
MAE 3	1.67E-02	1.75E-02	1.51E-02	-9.95%	-14.32%
MAE 5	1.66E-02	1.75E-02	1.65E-02	-0.70%	-5.92%
MAE 7	1.69E-02	1.76E-02	1.67E-02	-1.24%	-5.52%

We therefore observe from Tables 7.52, 7.53, 7.54, 7.55 that there is always a large gap from these traditional models, especially in unpredictable situations such as the 2020 crash.
It is worth noting that the gap is large especially in the short run, while in predicting the seventh day the difference is smaller, but still significant.
In addition, cases in which the gap in terms of MAE is low, but high in terms of MSE, should be interpreted as the inability of autoregressive models to avoid gross errors, which in a trading context are certainly not negligible.

Chapter 8

Conclusion and future works

With this thesis work, we explored financial time series forecasting from different perspectives. On one hand, we exploited tools that have been used for decades in the trading world, namely technical indicators. On the other, we have leveraged some of the latest models in the field of Deep Learning. In particular, the latter architectures are united by the use of Contrastive Learning, a paradigm that allowed us not to rely on labels to create the representations of the aforementioned time series.
As extensively described in previous chapters, these two approaches are not unrelated. In fact, the technical indicators represent an attempt to explain the aspects that concurred in the creation of the contrastive representations.
The purpose of the thesis is to evaluate the difference of both approaches with respect to the common task of forecasting future values of the series, with respect to three different time horizons. The performance evaluation was carried out through the use of two common metrics, the MSE and the MAE, which, in pairs, allowed us useful conclusions. Performance gap assessment, on the other hand, was done through the percentage difference.
In order to provide a complete overview, tests were performed against all possible market conditions (bullish, bearish, and mixed).
The results, obtained from the experiments conducted on three modalities of the same dataset, are manifold. Below we summarize them by points:

- The convenience of using contrastive models decreases as the time horizon increases.
- The market conditions that most benefit from using contrastive models are mixed markets (2020), peculiar conditions that simple models cannot handle effectively.
- The longer the time horizon, the smaller the differences between different

contrastive models.

- The best performer on most occasions is Time2State. In particular, it often prevails over 3- and 5-day horizons and then aligns with or is outperformed by other models.
- The best performers in general are associated with univariate and multivariate datasets without index.
- Top performers include light-weighted models
- The model that really benefits from the addition of the index in the multivariate case is TNC, which is, limited to the context of that dataset, the best performer on the 3-day horizon.
- Autoregressive models are always exceeded with considerable gaps.
- The error grows as the threshold for filtering out the most important features increases.

There are several ways to extend this work.
First, one could test other regression models instead of Ridge or use other clustering models as an alternative to K-Means. These configurations have not been explored because of the obvious current complexity of the pipeline, but they could certainly reveal additional patterns.
Second, one could test performance against other tasks, such as classification, in the case of trying to predict trend (increasing, decreasing, stable). We did not pursue this path since this would have introduced the problem of labeling, a complicated task that requires indispensable domain knowledge to obtain meaningful results.
Alternatively, one could change the selection of contrastive models employed in the comparison. Being the direction in which the research is currently heading, it is perfectly likely that these the models, within a short time, could be greatly improved. As mentioned earlier, the upgrades would consist of adopting more reasonable criteria for the creation of the pairs, positive and negative, which currently still generate many false negatives.
In any case, we can be satisfied that we have obtained significant results that have reflected most of the initial expectations.

Appendix A

Other material

The differences between modalities related to TS2Vec and CoST are reported next in Figures A.1, A.2, A.3, A.4, A.5 and A.6. These results are presented in the appendix because they do not show interesting patterns in fact they report negligible differences. The only thing worth noting is that these disparities are slightly accentuated during 2020, where the univariate dataset seems to prevail slightly. All these considerations are valid regardless of time horizon and model.

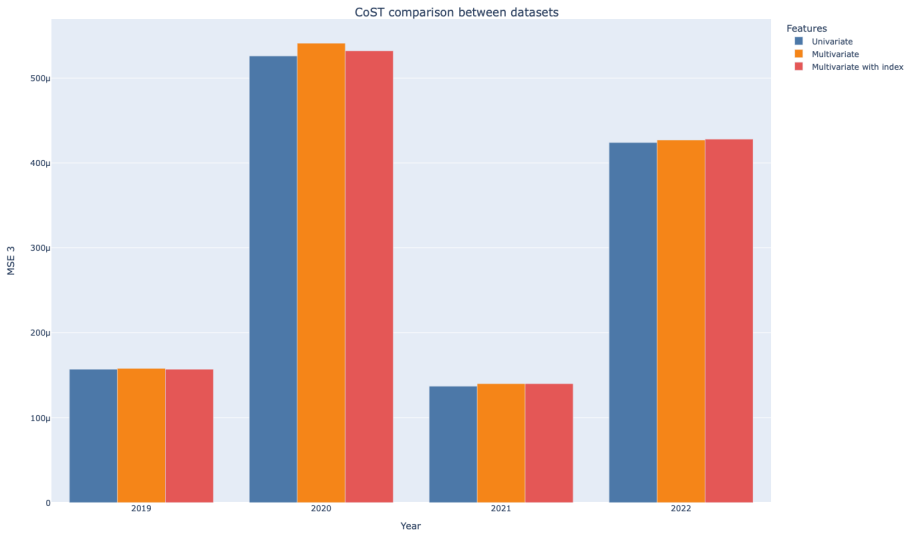


Figure A.1: MSE 3 CoST, comparison per dataset

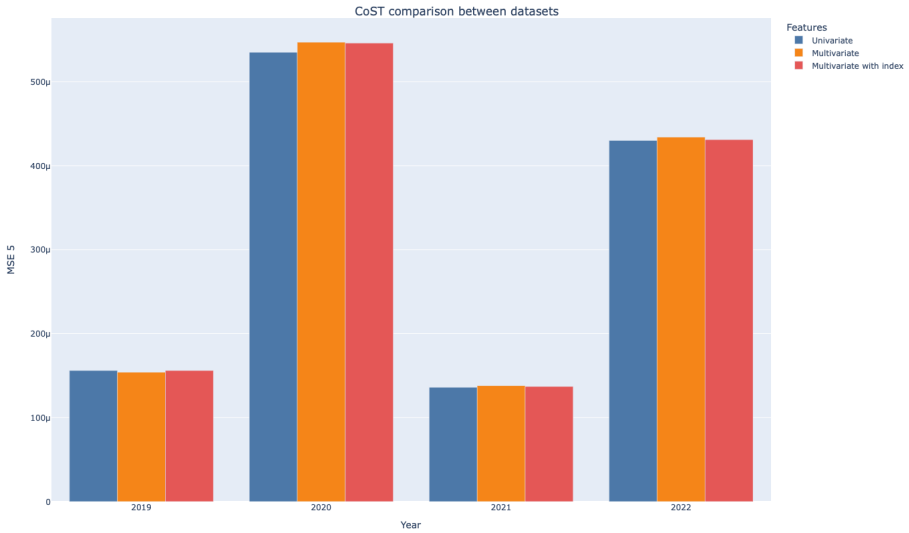


Figure A.2: MSE 5 CoST, comparison per dataset

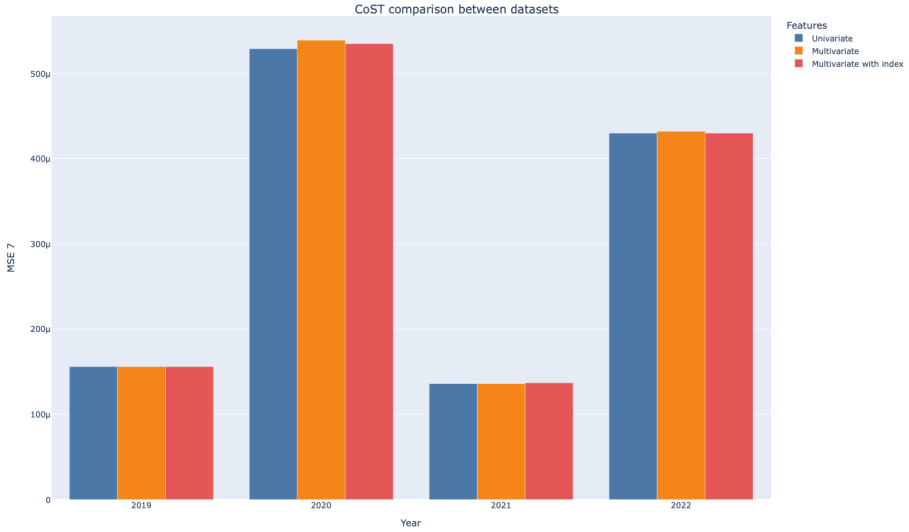


Figure A.3: MSE 7 CoST, comparison per dataset

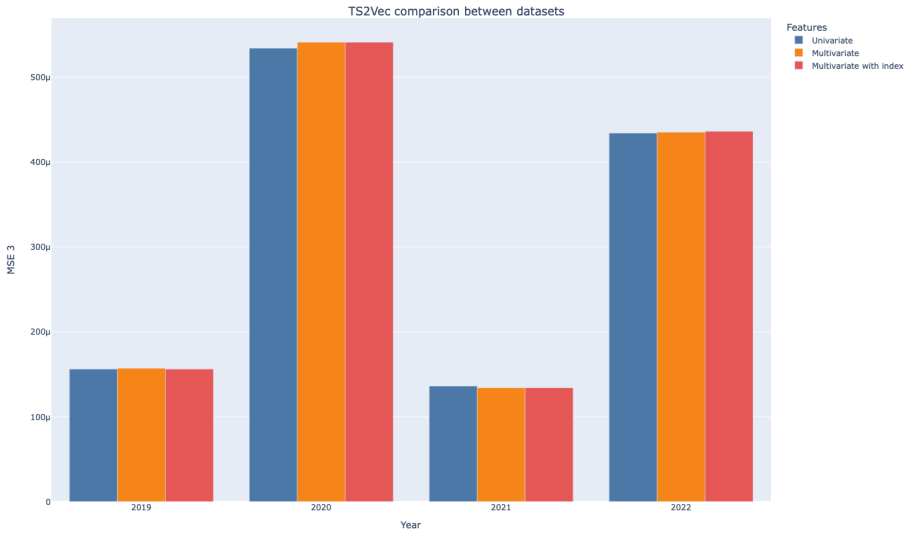


Figure A.4: MSE 3 TS2Vec, comparison per dataset

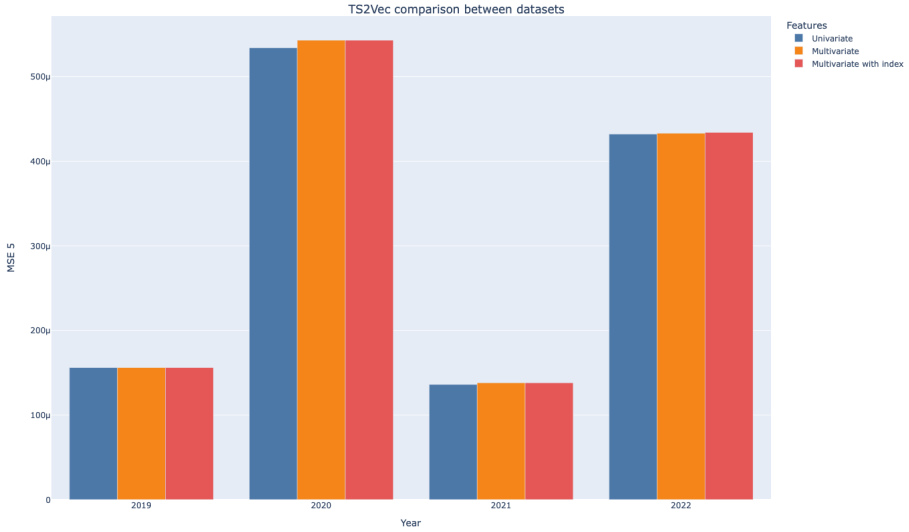


Figure A.5: MSE 5 TS2Vec, comparison per dataset

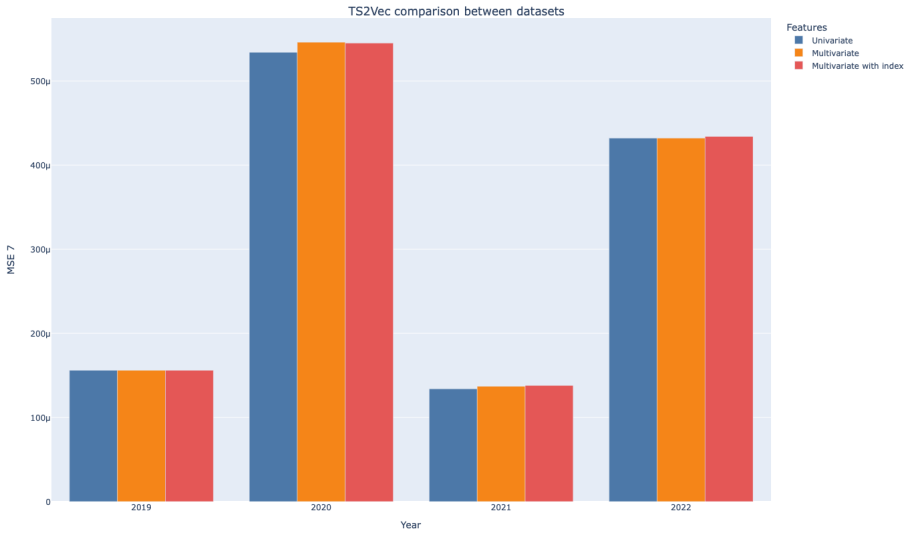


Figure A.6: MSE 7 TS2Vec, comparison per dataset

Bibliography

[1] Lex Fridman. *Deep Learning Basics: Introduction and Overview*. Jan. 2019 (cit. on p. 4).

[2] IBM. *What is machine learning?* URL: https://www.ibm.com/topics/machine-learning (cit. on p. 5).

[3] Berkeley. *What Is Machine Learning (ML)?* URL: https://ischoolonline.berkeley.edu/blog/what-is-machine-learning/ (cit. on p. 5).

[4] Qiong Liu and Ying Wu. «Supervised Learning». In: (Jan. 2012). DOI: 10.1007/978-1-4419-1428-6_451 (cit. on p. 5).

[5] Salim Dridi. *Supervised Learning - A Systematic Literature Review*. Apr. 2022. DOI: 10.31219/osf.io/tysr4. URL: osf.io/tysr4 (cit. on p. 5).

[6] Salim Dridi. *Unsupervised Learning - A Systematic Literature Review*. Apr. 2022. DOI: 10.31219/osf.io/kpqr6. URL: osf.io/kpqr6 (cit. on p. 6).

[7] Lior Rokach and Oded Maimon. «Clustering Methods». In: *Data Mining and Knowledge Discovery Handbook*. Ed. by Oded Maimon and Lior Rokach. Boston, MA: Springer US, 2005, pp. 321–352. ISBN: 978-0-387-25465-4. DOI: 10.1007/0-387-25465-X_15. URL: https://doi.org/10.1007/0-387-25465-X_15 (cit. on p. 6).

[8] Krzysztof J. Cios, Roman W. Swiniarski, Witold Pedrycz, and Lukasz A. Kurgan. «Unsupervised Learning: Association Rules». In: *Data Mining: A Knowledge Discovery Approach*. Boston, MA: Springer US, 2007, pp. 289–306. ISBN: 978-0-387-36795-8. DOI: 10.1007/978-0-387-36795-8_10. URL: https://doi.org/10.1007/978-0-387-36795-8_10 (cit. on p. 6).

[9] Akanksha Toshniwal, Kavi Mahesh, and R. Jayashree. «Overview of Anomaly Detection techniques in Machine Learning». In: *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*. 2020, pp. 808–815. DOI: 10.1109/I-SMAC49090.2020.9243329 (cit. on p. 6).

[10] Pierre Baldi. «Autoencoders, Unsupervised Learning, and Deep Architectures». In: *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*. Ed. by Isabelle Guyon, Gideon Dror, Vincent Lemaire, Graham Taylor, and Daniel Silver. Vol. 27. Proceedings of Machine Learning Research. Bellevue, Washington, USA: PMLR, Feb. 2012, pp. 37–49. URL: https://proceedings.mlr.press/v27/baldi12a.html (cit. on p. 6).

[11] Xiangli Yang, Zixing Song, Irwin King, and Zenglin Xu. *A Survey on Deep Semi-supervised Learning*. 2021. arXiv: 2103.00550 [cs.LG] (cit. on p. 6).

[12] Yann LeCun & Ishan Misra. *Self-supervised learning: The dark matter of intelligence*. URL: https://ai.facebook.com/blog/self-supervised-learning-the-dark-matter-of-intelligence/ (cit. on p. 7).

[13] Linus Ericsson, Henry Gouk, Chen Change Loy, and Timothy M. Hospedales. «Self-Supervised Representation Learning: Introduction, advances, and challenges». In: *IEEE Signal Processing Magazine* 39.3 (May 2022), pp. 42–62. DOI: 10.1109/msp.2021.3134634. URL: https://doi.org/10.1109%5C%2Fmsp.2021.3134634 (cit. on p. 7).

[14] Jeffrey Dastin. *Amazon scraps secret AI recruiting tool that showed bias against women*. URL: https://www.reuters.com/article/us-amazon-com-jobs-automation-insight-idUSKCN1MK08G (cit. on p. 7).

[15] Francesco La Trofa. *Explainable AI: cos'è, quali sono i principi e gli esempi*. URL: https://tech4future.info/explainable-ai-cose-principi-esempi/ (cit. on p. 8).

[16] Ouren X. Kuiper, Martin van den Berg, Joost van der Burgt, and Stefan Leijnen. «Exploring Explainable AI in the Financial Sector: Perspectives of Banks and Supervisory Authorities». In: *CoRR* abs/2111.02244 (2021). arXiv: 2111.02244. URL: https://arxiv.org/abs/2111.02244 (cit. on p. 8).

[17] Phuc H. Le-Khac, Graham Healy, and Alan F. Smeaton. «Contrastive Representation Learning: A Framework and Review». In: *IEEE Access* 8 (2020), pp. 193907–193934. DOI: 10.1109/access.2020.3031549. URL: https://doi.org/10.1109%2Faccess.2020.3031549 (cit. on p. 9).

[18] Rohit Kundu. May 22, 2022. URL: https://www.v7labs.com/blog/contrastive-learning-guide#:~:text=Contrastive%5C%20Learning%5C%20is%5C%20a%5C%20technique,a%5C%20data%5C%20class%5C%20from%5C%20another (cit. on pp. 10, 12).

[19] S. Chopra, R. Hadsell, and Y. LeCun. «Learning a similarity metric discriminatively, with application to face verification». In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. 2005, 539–546 vol. 1. DOI: 10.1109/CVPR.2005.202 (cit. on p. 10).

[20] Florian Schroff, Dmitry Kalenichenko, and James Philbin. «FaceNet: A Unified Embedding for Face Recognition and Clustering». In: *CoRR* abs/1503.03832 (2015). arXiv: 1503.03832. URL: http://arxiv.org/abs/1503.03832 (cit. on pp. 10, 32).

[21] Wallstreetmojo Team. *What Is Time Series Analysis?* URL: https://www.wallstreetmojo.com/time-series-analysis/#h-what-is-time-series-analysis (cit. on p. 13).

[22] Clay Grewcoe. *The Ultimate Guide to Time-Series Analysis (With Examples and Applications)*. URL: https://www.timescale.com/blog/what-is-time-series-analysis-with-examples-and-applications/ (cit. on p. 13).

[23] Omer Berat Sezer, M. Ugur Gudelek, and Ahmet Murat Özbayoglu. «Financial Time Series Forecasting with Deep Learning : A Systematic Literature Review: 2005-2019». In: *CoRR* abs/1911.13288 (2019). arXiv: 1911.13288. URL: http://arxiv.org/abs/1911.13288 (cit. on p. 14).

[24] ADAM HAYES. *Volatility: Meaning In Finance and How it Works with Stocks*. URL: https://www.investopedia.com/terms/v/volatility.asp#:~:text=Volatility%5C%20is%5C%20a%5C%20statistical%5C%20measure%5C%20of%5C%20the%5C%20dispersion%5C%20of%5C%20data,prices%5C%2C%5C%20on%5C%20an%5C%20annualized%5C%20basis. (cit. on p. 14).

[25] *Fundamental Analysis: A Complete Guide*. URL: https://www.winvesta.in/blog/fundamental-analysis-a-complete-guide/#:~:text=Fundamental%20analysis%20is%20a%20method,stock%20does%20not%20change%20overnight. (cit. on p. 15).

[26] Wallstreetmojo Team. *What is Trade Signal?* URL: https://www.wallstreetmojo.com/trade-signal/. (cit. on p. 15).

[27] ADAM HAYES. *Simple Moving Average (SMA): What It Is and the Formula*. URL: https://www.investopedia.com/terms/s/sma.asp (cit. on p. 16).

[28] Sabrina Jiang. 2020. URL: https://www.investopedia.com/terms/c/crossover.asp (cit. on p. 17).

[29] J.B. Maverick. *How Is the Exponential Moving Average (EMA) Formula Calculated?* URL: https://www.investopedia.com/ask/answers/122314/what-exponential-moving-average-ema-formula-and-how-ema-calculated.asp#:~:text=The%5C%20exponential%5C%20moving%5C%20average%5C%20(EMA)%5C%20is%5C%20a%5C%20technical%5C%20chart%5C%20indicator,importance%5C%20to%5C%20recent%5C%20price%5C%20data. (cit. on p. 17).

[30] BRIAN DOLAN. *MACD Indicator Explained, with Formula, Examples, and Limitations*. URL: https://www.investopedia.com/terms/m/macd.asp (cit. on p. 18).

[31] Brian Dolan. *MACD Indicator Explained, with Formula, Examples, and Limitations*. URL: https://www.investopedia.com/terms/m/macd.asp (cit. on p. 18).

[32] Sabrina Jiang. 2022. URL: https://www.investopedia.com/terms/m/macd.asp (cit. on p. 19).

[33] JASON FERNANDO. *Relative Strength Index (RSI) Indicator Explained With Formula*. URL: https://www.investopedia.com/terms/r/rsi.asp (cit. on p. 19).

[34] ADAM HAYES. *On-Balance Volume (OBV): Definition, Formula, and Uses as Indicator*. URL: https://www.investopedia.com/terms/o/onbalancevolume.asp (cit. on p. 20).

[35] Sabrina Jiang. 2020. URL: https://www.investopedia.com/articles/active-trading/091514/essential-strategies-trading-volume.asp (cit. on p. 21).

[36] CANDY SCHAAP. *ADX: The Trend Strength Indicator*. URL: https://www.investopedia.com/articles/trading/07/adx-trend-indicator.asp#:~:text=The%5C%20average%5C%20directional%5C%20index%5C%20(ADX,as%5C%20a%5C%20trend%5C%20strength%5C%20indicator. (cit. on p. 21).

[37] ADAM HAYES. *Bollinger Bands®: What They Are, and What They Tell Investors*. URL: https://www.investopedia.com/terms/b/bollingerbands.asp (cit. on p. 22).

[38] Sabrina Jiang. 2021. URL: https://www.investopedia.com/terms/b/bollingerbands.asp (cit. on p. 22).

[39] Athanasopoulos G. Hyndman R.J. *Forecasting: principles and practice*. OTexts, 2018. Chap. 8 (cit. on pp. 23, 24).

[40] ADAM HAYES. *Autoregressive Integrated Moving Average (ARIMA) Prediction Model*. URL: https://www.investopedia.com/terms/a/autoregressive-integrated-moving-average-arima.asp (cit. on p. 23).

[41] Athanasopoulos G. Hyndman R.J. *Forecasting: principles and practice*. OTexts, 2018. Chap. 7 (cit. on p. 24).

[42] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. *An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling*. 2018. arXiv: 1803.01271 [cs.LG] (cit. on pp. 25, 30).

[43] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. «Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting». In: *CoRR* abs/2012.07436 (2020). arXiv: 2012.07436. URL: https://arxiv.org/abs/2012.07436 (cit. on p. 25).

[44] David Salinas, Valentin Flunkert, and Jan Gasthaus. *DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks*. 2019. arXiv: 1704.04110 [cs.AI] (cit. on p. 25).

[45] Sana Tonekaboni, Danny Eytan, and Anna Goldenberg. «Unsupervised Representation Learning for Time Series with Temporal Neighborhood Coding». In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=8qDwejCuCN (cit. on pp. 25, 27, 31, 32).

[46] Ching-Yao Chuang, Joshua Robinson, Lin Yen-Chen, Antonio Torralba, and Stefanie Jegelka. *Debiased Contrastive Learning*. 2020. arXiv: 2007.00224 [cs.LG] (cit. on p. 26).

[47] Marthinus Christoffel du Plessis, Gang Niu, and Masashi Sugiyama. «Class-prior Estimation for Learning from Positive and Unlabeled Data». In: *CoRR* abs/1611.01586 (2016). arXiv: 1611.01586. URL: http://arxiv.org/abs/1611.01586 (cit. on p. 26).

[48] Marthinus C du Plessis, Gang Niu, and Masashi Sugiyama. «Analysis of Learning from Positive and Unlabeled Data». In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger. Vol. 27. Curran Associates, Inc., 2014. URL: https://proceedings.neurips.cc/paper_files/paper/2014/file/35051070e572e47d2c26c241ab88307f-Paper.pdf (cit. on p. 26).

[49] Wee Sun Lee and B. Liu. «Learning with Positive and Unlabeled Examples Using Weighted Logistic Regression». In: *International Conference on Machine Learning*. 2003 (cit. on p. 26).

[50] Charles Elkan and Keith Noto. «Learning Classifiers from Only Positive and Unlabeled Data». In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '08. Las Vegas, Nevada, USA: Association for Computing Machinery, 2008, pp. 213–220. ISBN: 9781605581934. DOI: 10.1145/1401890.1401920. URL: https://doi.org/10.1145/1401890.1401920 (cit. on p. 26).

[51] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. *CoST: Contrastive Learning of Disentangled Seasonal-Trend Representations for Time Series Forecasting*. 2022. arXiv: 2202.01575 [cs.LG] (cit. on pp. 27, 28).

[52] Steven L. Scott and Hal R. Varian. «4. Bayesian Variable Selection for Nowcasting Economic Time Series». In: *Economic Analysis of the Digital Economy*. Ed. by Avi Goldfarb, Shane M. Greenstein, and Catherine E. Tucker. Chicago: University of Chicago Press, 2015, pp. 119–136. ISBN: 9780226206981. DOI: doi:10.7208/9780226206981-007. URL: https://doi.org/10.7208/9780226206981-007 (cit. on p. 27).

[53] S. Rao Jammalamadaka, Jinwen Qiu, and Ning Ning. *Multivariate Bayesian Structural Time Series Model*. 2018. arXiv: 1801.03222 [stat.ML] (cit. on p. 27).

[54] Dominik Janzing Jonas Peters and Bernhard Scho lkopf. *Elements of causal inference: foundations and learning algorithms*. The MIT press, 2017 (cit. on p. 28).

[55] Giambattista Parascandolo, Niki Kilbertus, Mateo Rojas-Carulla, and Bernhard Schölkopf. *Learning Independent Causal Mechanisms*. 2018. arXiv: 1712.00961 [cs.LG] (cit. on p. 28).

[56] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. *TS2Vec: Towards Universal Representation of Time Series*. 2022. arXiv: 2106.10466 [cs.LG] (cit. on pp. 29–31).

[57] Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. *Unsupervised Scalable Representation Learning for Multivariate Time Series*. 2020. arXiv: 1901.10738 [cs.LG] (cit. on p. 31).

[58] Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, Xiaoli Li, and Cuntai Guan. *Time-Series Representation Learning via Temporal and Contextual Contrasting*. 2021. arXiv: 2106.14112 [cs.LG] (cit. on p. 31).

[59] Chengyu Wang, Kui Wu, Tongqing Zhou, and Zhiping Cai. «Time2State: An Unsupervised Framework for Inferring the Latent States in Time Series Data». In: *Proc. ACM Manag. Data* 1.1 (May 2023). DOI: 10.1145/3588697. URL: https://doi.org/10.1145/3588697 (cit. on pp. 32–34).

[60] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. *Financial Time Series Forecasting with Deep Learning : A Systematic Literature Review: 2005-2019*. 2019. arXiv: 1911.13288 [cs.LG] (cit. on p. 36).

[61] 2023. URL: https://finasko.com/nasdaq-100-companies/ (cit. on p. 42).

[62] Gregory Gundersen. *Returns and Log Returns*. URL: https://gregorygundersen.com/blog/2022/02/06/log-returns/ (cit. on p. 43).