Computational Economics (2024) 63:1477–1491
https://doi.org/10.1007/s10614-023-10370-x

Estimating the Likelihood of Financial Behaviours Using Nearest Neighbors

A case study on market sensitivities

Tiago Mendes-Neves[1,2] · Diogo Seca[1] · Ricardo Sousa[1] · Cláudia Ribeiro[3,4] · João Mendes-Moreira[1,2]

Accepted: 20 February 2023 / Published online: 25 March 2023

Abstract

As many automated algorithms find their way into the IT systems of the banking sector, having a way to validate and interpret the results from these algorithms can lead to a substantial reduction in the risks associated with automation. Usually, validating these pricing mechanisms requires human resources to manually analyze and validate large quantities of data. There is a lack of effective methods that analyze the time series and understand if what is currently happening is plausible based on previous data, without information about the variables used to calculate the price of the asset. This paper describes an implementation of a process that allows us to validate many data points automatically. We explore the K-Nearest Neighbors algorithm to find coincident patterns in financial time series, allowing us to detect anomalies, outliers, and data points that do not follow normal behavior. This system allows quicker detection of defective calculations that would otherwise result in the incorrect pricing of financial assets. Furthermore, our method does not require knowledge about the variables used to calculate the time series being analyzed. Our proposal uses pattern matching and can validate more than 58% of instances, substantially improving human risk analysts' efficiency. The proposal is completely transparent, allowing analysts to understand how the algorithm made its decision, increasing the trustworthiness of the method.

Keywords Automatic pricing validation · Nearest neighbors · Machine learning · Interpretable machine learning

✉ Tiago Mendes-Neves
tiago.m.neves@inesctec.pt

Extended author information available on the last page of the article

1 Introduction

Automation is coming into everything in an unprecedented way, and finance is no exception. Today the sector uses many models that price thousands of assets. However, these systems are not 100% reliable. In many scenarios, machine learning models or even mathematical formulas can fail. One example is that many models would assume that interest rates could not be negative, which we know is not true. Furthermore, since each model relies on multiple databases to provide information about variables, a single mistake in a database can ruin the calculation and produce an anomaly.

An anomaly is a data point that is outside of the expected distribution. The expected distribution is either a function of previous values for the same data point or a function of other variables with causal relationships. These anomalies can have a significant business impact. The banks that provide services based on automatic pricing systems require risk analysts to monitor the prices and market flows associated with ensuring that anomalies are corrected as quickly as possible. However, the increase in the number of automatic pricing systems quickly outpaces the rate analysts can verify the systems. In some cases, analysts can be responsible for validating up to 100 000 price points daily, approximately 4 data points per second. Manually verifying prices demands an ever-increasing amount of resources.

Therefore, there is a need for a decision support system that guides the risk analyst to the potential errors in the calculations or even an automated system that could potentially freeze the trades in case of anomaly detection. Today, analysts have access to statistical methods that help them filter potentially anomalous prices. However, the efficacy of these methods is very low. By increasing risk analysts' ability to address wrong calculations quickly, we can increase their effectiveness and the value they provide.

We propose a framework that validates upcoming data instances using pattern matching. Pattern matching is the process of finding similar instances in data to the one we are trying to validate. We can use these similar instances to calculate an expected range to consider a new value valid. Furthermore, our model is transparent regarding the decision provided, ensuring that analysts can inspect the algorithm's decision, increasing their trust in the models.

Pattern matching runs on the principle that if something has happened before, it can happen again. For this, it is essential that training data meets specific quality standards, like the absence, or a significantly reduced amount of anomalies, so that the model does not consider spurious instances as normal behavior. Pattern matching can handle data affected by the time variable (seasonality or decay). We used the Nearest Neighbors approach for pattern matching. This method finds similar patterns by checking which patterns have the lowest mathematical distance to our target.

In our case, we address the problem in an unsupervised setting since no labels are available about the validity of the values, i.e., if the value is an anomaly. Similar to many problems in finance, institutions' models are not publicly available, and disclosing them would lead to losses in competitive advantage. For this reason, working in a black-box setting, with no labels nor underlying models, is a fundamental problem to be addressed.

Our approach can validate at least 58% of the instances. This validation rate means that we increase the risk analysts' effectiveness by a similar amount. We filter out common behaviors, allowing the analyst to prioritize more urgent pricing anomalies. Note that, according to the confidence interval set, we can adjust how many instances are validated, always considering a trade-off between the number of instances validated and potential false positives.

We organize the paper as follows:

- Section 2 briefly describes the problems with options pricing models and describes the related work in the area of pattern mining and anomaly detection, primarily related to the financial sector.
- Section 3 describes the data set we use, along with the preprocessing steps required to reproduce this work.
- Section 4 talks about our method, describing our adaptation of the K-Nearest Neighbors algorithm for pattern matching step-by-step.
- In Sect. 5, we describe the experiments made and discuss the results.
- Finally, in Sect. 6, we make our conclusions and discuss possible future work in the field.

2 Related Work

Throughout history, many works have attempted to price assets. Regarding options, the most relevant work is from Black and Scholes (1973), proposing a theoretical valuation formula for pricing options. The motivation for finding the correct models to price assets is simple. If there are inefficiencies in the market, we can take advantage of them using our models.

However, the Black-Scholes model, and subsequent models based on it, assumes some ideal conditions. For example, it assumes that the underlying stock pays no dividend, but more than 3/4 of the companies in the SP500 do. With so many restrictions present in the formula, it is easy for anomalies and outliers to appear in data. While the pricing models consider many possible anomalies, like the skewness premium, in practice, these models require manual supervision to ensure their correct behavior.

To add even further complexity, machine learning models are increasing in share. Examples like agent-based approaches (Suzuki et al., 2009), or statistical models like GARCH (Christoffersen et al., 2013) are currently used to price assets. This added complexity increases the chance of anomalies happening substantially. Furthermore, these methods cannot explain the pricing procedure, increasing the difficulty of debugging such models and decreasing their reliability. Anomalies in data are very problematic for modeling with such machine learning techniques.

Financial data is especially susceptible to anomalies. In cases like electricity pricing (Janczura et al., 2013), managing anomalies is one of the most factors due to unexpected events impacting consumers and the producers' margins. Detecting financial fraud is another use case for anomaly detection with a high business impact. Many algorithms can address fraud detection, as described by Alsuwailem et al. (2022), ranging from Logistic Regression, decision tree-based algorithms like Random Forest and Neural Networks. The largest issue with these methods is that they do not provide a clear insight into how a decision is performed, which is crucial to ensure analysts' trust.

Data quality is important for reliable results (Cappiello et al., 2018). However, this falls out of the scope of this work, which works directly with the results produced by the model rather than verifying the underlying variables. Rigatos (2021) proposes a work with similar goals to ours, but focuses on agent-based systems.

Bishop (2006) defined pattern recognition as "concerned with the automatic discovery of regularities in data through the use of computer algorithms and with the use of these regularities to take actions such as classifying the data into different categories." We focused on a specific topic of pattern recognition, known as pattern matching.

The main applications of this technique are speech recognition (Rabiner, 1990), genome sequence analysis (Liew et al., 2005), and financial forecast. On financial applications, Wang and Chan (2007) used pattern matching to find bull/bear signals for optimal market entry/exit timing on the NASDAQ Composite Index and the Taiwan Weighted Index. Fu et al. (2007) compared several pattern matching techniques, both rule-based and template-based, in stock market time series. Wan et al. (2016) evaluated the effect of segmenting the data on the pattern matching, using financial time series from the Hong Kong stock market. Neural networks also represent a substantial amount of research in the field. For example, Douglas et al. (2014) attempts to discover whether certain events were predictable, namely the company failures in New Zealand during the 2006–2009 period. However, these methods can confirm whether events were predictable but do not provide underlying reasoning for the outcome.

The methods previously presented do not function as automated systems. The work developed in artificial intelligence usually focuses on building the models rather than a framework to validate outputs. This focus results in models that need constant supervision and are, therefore, inefficient from the automation point of view.

We can automate the monotonous tasks that represent a significant part of the workload, as is the goal of robotic process automation (Hofmann et al., 2020). Other tasks are currently hard to automate due to the absence of labels, which would need processes that would be costly, making implementation unadvised.

Hautamaki et al. (2004) proposed an extension of the K-Nearest Neighbors algorithm (KNN) for anomaly detection. Our work extends the proposal by (1) defining how to use KNN for anomaly detection in time series, and (2) providing a framework for anomaly detection that explains the reasoning behind the decision.

3 Describing the Data

The data set contains Greek options data from Natixis (2020). Greeks are values that represent the sensitivity of the price of an option to a change in an underlying parameter. In our case, we have access to the Vega greek, which is the partial derivative of the option price $\delta V$with respect to the volatility of the underlying asset returns $\delta\sigma$. In discrete terms, it is the change in the option price due to a volatility increase of 1% (or 100 basis points). Equation 1 describes the calculation the Vega υsensitivity.

$$\upsilon = \frac{\delta V}{\delta\sigma} \tag{1}$$

The following points describe the scope of the dataset:

- From the start of August 2018 to the end of July 2019 with daily frequency.
- Swaptions (swap options) on the interest rate.
- Swaps against the LIBOR.
- Asian market.
- Short, medium, and long-term maturities.
- JPY representing 53% of the data points, AUD 35%, and USD 10%. Other currencies have residual values (<2%), such as SGD, EUR, and HKD.
- The data identifiers, such as the underlying instrument, are not disclosed for data anonymization purposes.
- This data set tracks only open positions.

Each combination of instrument and maturity provided represents a unique time series. Data is generated daily via automated models based on several parameters, such as time to maturity, strike price, and LIBOR rates. In this work, we avoid using information about the underlying to ensure that the method extends to various asset types. However, we can include underlying information in the model, and Sect. 5.2 presents the results of doing such.

To feed the data into our models, we transform the time series into segments of 5+1 values. The first five values will be the segment we will use to search for similar patterns in the data set. The remaining value is the data point to validate. This is detailed in Fig. 1. We explain why use segments of length five in the experiments in Sect. 5.1.

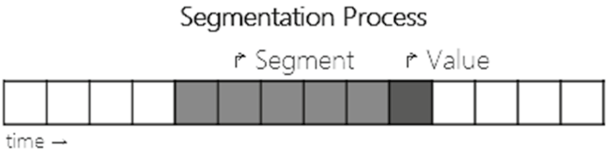


Fig. 1 Segmentation process of a time series. This process is repeated through all the time series, analogous to a sliding window process, as long as the segment can have length = 5. Note: *Value* does not belong to the *Segment*

These segments are then normalized based on the 5-point segment values. We use the maximum between the 1st and 3rd quartile to avoid interference from anomalies in the normalization process. This can be seen in Eqs. 2, 3, and 4, where $\mathbf{S}$ represents the segment vector, v the *Value* of the data point, Q the quartile function and x is the index of the data point. We omit v in the normalization coefficient Ω to avoid using future information.

$$\Omega = max(|Q_1(S_x)|, |Q_3(S_x)|) \tag{2}$$

$$\mathbf{S}_x^{norm} = \frac{\mathbf{S}_x}{\Omega} \tag{3}$$

$$v_x^{norm} = \frac{v_x}{\Omega} \tag{4}$$

With the steps identified in Fig. 1 and the normalization step, we have solved two problems. The first one is that when we normalize the data, we condense the segments into a smaller data point space. This dimensionality reduction happens due to reducing the influence of factors such as currency and substantially improves the performance of pattern matching. While transforming the data, we also stored all the required steps to reverse the normalization process. The second problem we have solved is that, due to segmentation, we can consider the resulting pattern stationary. Using stationary data is very important since allows models to better generalize their predictions to unseen data.

in the end, we have 3 409 507 segments, resulting from 36 151 time series. We present a more in-depth description of the final data set in Table 1.

It is important to split the data into train, test, and validation sets to ensure that our models generalize well to previously unseen data. We are opting to use 43% train, 10% validation, and 47% test. These numbers result from the temporal division of data. All data points until the 31st of January 2019 are in the train set; from there to the 28th of February are in the validation set, and the remaining data set goes to the test set. This procedure goes along with the good practices in data science.

Table 1 Brief description of the variables in the data set

Feature	Description
Date	Date when the data point was generated (YYYY/MM/DD)
SeriesID	Identifier of the Vega series
$Instance_x$	Value of the Vega series x days before *Date*. $x\, in\, [1, 2, ..., n = 5]$
Value	Value of the Vega series in *Date*. Used as the target variable

4 Methods

The K-Nearest Neighbors (kNN) is an algorithm that predicts a new data point using the K closest data points in the training set. The K parameter is user-defined. It uses a distance metric to find the closest instances, with Euclidean distance being the default distance function. To predict a value, it calculates the average of the K neighbors' responses, as we can observe in Fig. 2.

Since we have many anomalies in data, we are opting to calculate the prediction using the median instead of the mean. We describe the difference in performance in Sect. 5.1. For a more in-depth explanation of the KNN, see Kuhn and Johnson (2013).

Figure 3 describes all the steps to calculate the two outputs from the KNN algorithm: (1) the expected value $M(\mathbf{y})$, calculated by the median of the neighbors' values $\mathbf{y}$, and (2) the expected variance $\sigma(\mathbf{y})$, calculated as the standard deviation of the neighbors' values $\mathbf{y}$.

We want to compute the confidence level to assess whether an instance is valid or not. For that, we leverage the value of neighbors found by the KNN. Considering that the prediction follows a normal distribution, we use the KNN expected value (median of the neighbors) as the mean, and the standard deviation of the K closest instances, and we can obtain the confidence interval. We describe this procedure in Eq. 5, where x is the user-defined multiplier for managing the interval width. We will use one standard deviation ($x = 1$) in the experiments, except when referred otherwise. Figure 4 visually presents the process of validating data points. When data points fall within the confidence interval, they are validate. Otherwise, they are flagged as potential anomalies.

$$\text{Confidence Interval} = \text{M}\ (y)\ \pm \text{x} * \sigma(y) \tag{5}$$

Since we do not have labels indicating an instance is an anomaly, we cannot show accuracy or confusion matrix metrics. To evaluate the model we have fitted, we will calculate the median absolute error (MAE), comparing the median value used on

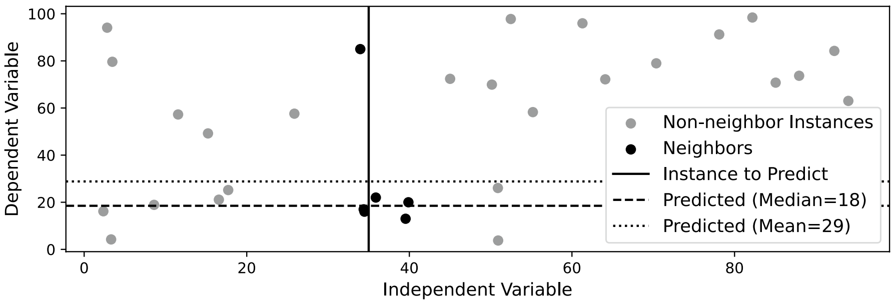
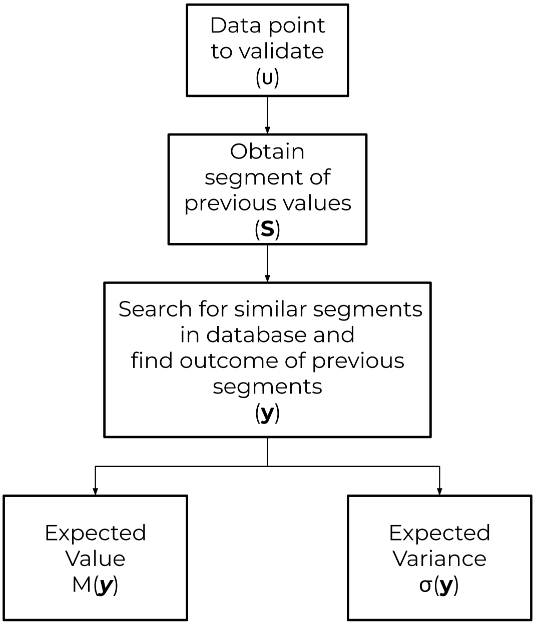


Fig. 2 A synthetic data example of a prediction using the KNN algorithm. To predict the expected value, the KNN searches for the closest instances in the independent variable and then provides a prediction by grouping the dependent variable's values of the neighbors. Note that due to the anomaly in the dataset, the prediction using the median is substantially different from the mean

Fig. 3 The steps required to calculate the confidence interval


Data point to validate (u)
Obtain segment of previous values (**S**)
Search for similar segments in database and find outcome of previous segments (**y**)
Expected Value M(***y***)
Expected Variance σ(**y**)

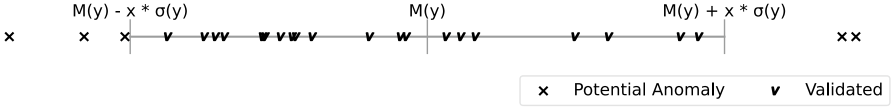


Fig. 4 Validated data points are contained within the confidence interval, as calculated by Eq. 5

the confidence interval with the actual Vega value. To assess the model's real-world effectiveness, we will show the percentage of valid instances for several confidence levels.

5 Experiments

5.1 Segment Length

To find the segment length, we iterate over several lengths. Our top priority is to choose an optimal value that minimizes median absolute error and, if possible, minimizes time spent by the algorithm. We present the results of this experiment in Fig. 5.

The length 5 minimizes the error. Since the time spent increase is very low at small lengths, it has no impact on the choice of segment length. However, the time required increases exponentially with the length of the segment. Also, note that using the median in the KNN algorithm decreases the error and shifts the best-performing segment length to a lower value.

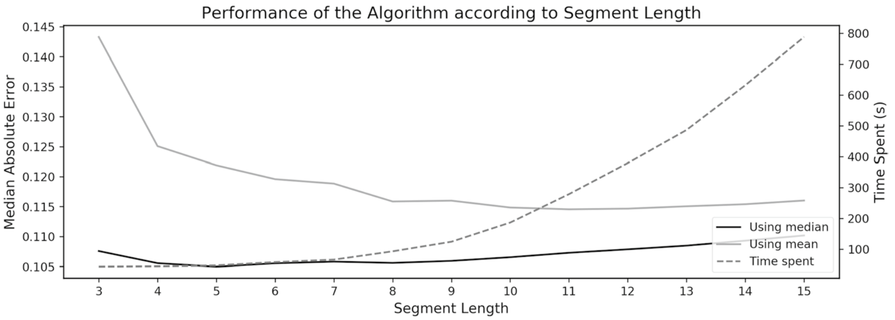


Fig. 5 Variation in the error and time spent by the algorithm as the segment length increases. Note that using the median yields better results over the mean

5.2 Other Data Points

More features are available in the data set other than the value, namely strike price and distance to maturity. We made several tests, summarized in Table 2, to discover the impact of several features.

As seen in Table 2, none of the features managed to improve the result of the segment-only approach. Segment-only is the best approach since it has the best performance and low complexity (fewer features). If any of the features positively impacted the model, we would find the ideal weight to give to each feature since, by default, all features of the K-Nearest Neighbors algorithm have a weight equal to 1.

5.3 Number of Neighbors

Defining the correct number of neighbors (K) is a crucial step. We opted to find the ideal value iteratively due to the dependence on the length and dimensionality of the data set. We built an experiment where we tested all the possible numbers of neighbors from 3 to 9. We present the results in Fig. 6.

From the results in Fig. 6, we conclude that the model error decreases as the K value increases, and the opposite occurs with the time spent. Moreover, the confidence interval calculated increases due to a higher sample size. On the other hand, the time required for the algorithm's training increases as K increases. One factor we could not

Table 2 Results of the tests for several feature sets in terms of Median Absolute Error (MAE)

Feature set	MAE
Median of last 5 days (Baseline)	0.1189
Segment only	**0.1116**
Segment + Distance to maturity	0.1131
Segment + Notional + Current notional	0.1144
Segment + Strike price	**0.1116**
Segment + All features	0.1152

Bold values indicate the best performers

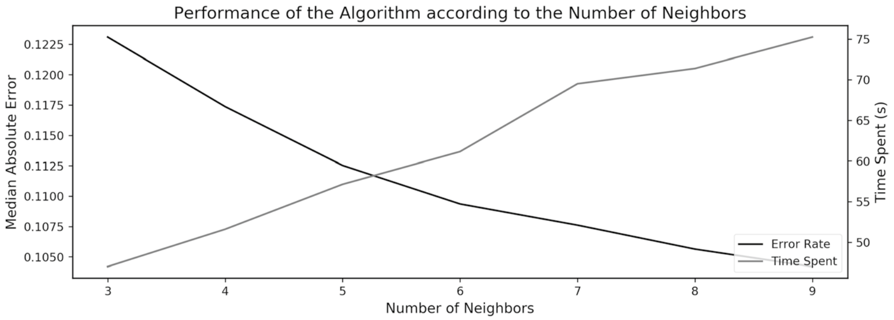


Fig. 6 Variation in the error and time spent by the algorithm as the number of neighbors K increases

quantify is that when the value of K increases, the method loses interpretability for the end user. Since interpretability is very important in our use case, we aim to use a value of K that provides good performance in terms of error and confidence interval, while keeping a low value of K for interpretability purposes and fast inference.

Knowing all the effects K has on the model, we found that $K = 5$ is the value that finds the best tradeoff balance for our particular case.

5.4 Test Set Results

Having set our K value, we can evaluate the algorithm's performance in the test data set. The first result is the estimation error distribution for the K-Nearest Neighbors in Fig. 7. Note that the estimation error is centered around zero, as expected, and the distribution has a low error variance. These results signal that the model fits the data well and has no generalization problem.

In Table 3, we summarize the tests and results obtained for several confidence intervals.

based on the results of Table 3, we conclude that, in the strictest approach of a low confidence interval, the reduction of instances needing manual validation

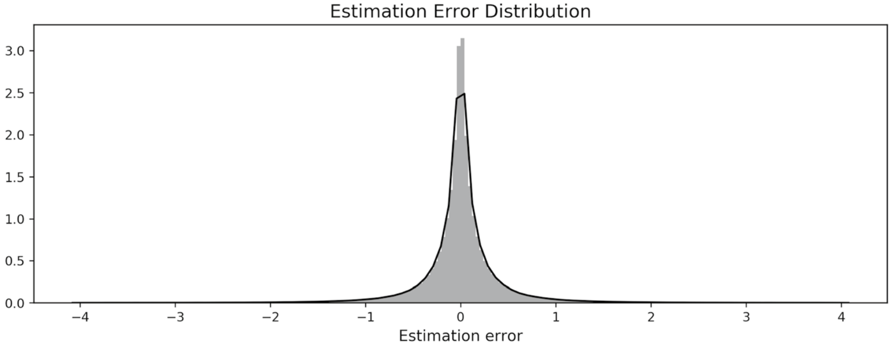


Fig. 7 Distribution of the error of the estimation in the test set

Table 3 Results of the tests for several confidence intervals

Confidence Interval	Percentage of validated instances
$[M-\sigma; M+\sigma]$	0.58
$[M-2\sigma; M+2\sigma]$	0.79
$[M-3\sigma; M+3\sigma]$	0.87
$[M-4\sigma; M+4\sigma]$	0.91

would decrease by 42%. Since risk analysts usually run on tight schedules, this improvement should increase productivity by a substantial margin.

We can obtain a higher percentage of validated instances depending on the confidence level. However, due to the lack of labeled data, we cannot assess how different confidence intervals impact the anomalies that can become false negatives. In Fig. 8, we show several examples of time series. Note that the clear anomalies are never validated.

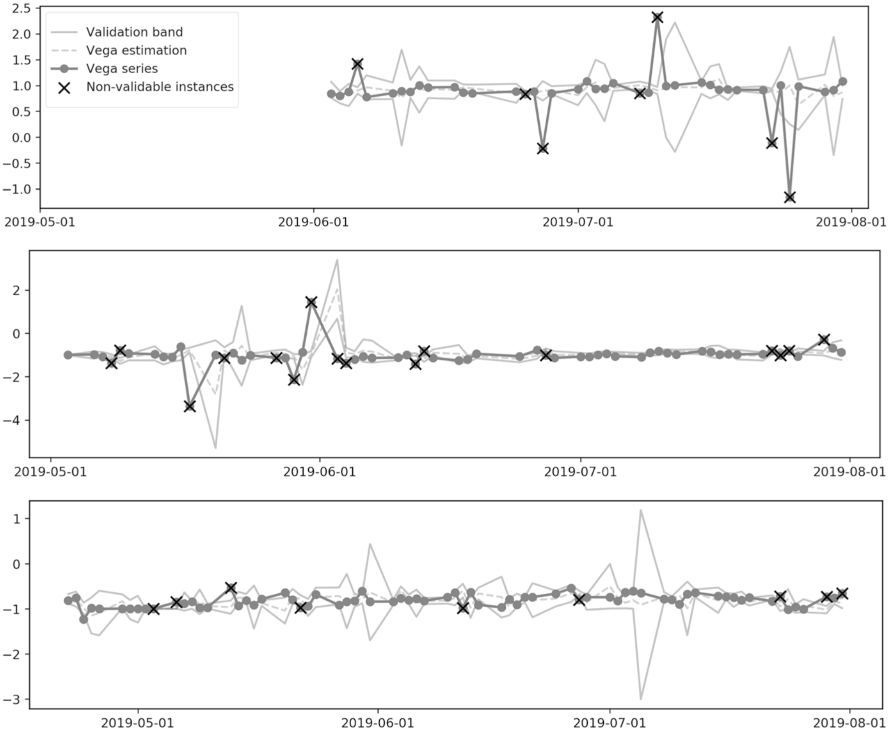


Fig. 8 Examples of the pattern matching output in three different time series. Note that anomalies were not removed from the data, leading to the confidence interval being miscalculated whenever an anomaly occurs. However, if anomalies are removed, this problem disappears, validating the assumption that data needs to be as clean as possible not to damage the technique's performance. This visualization uses a 2σ confidence interval

5.5 Risk Analyst Interface

This method aims to assist the risk analyst when looking for miscalculated values in the newly generated data. In Figs. 9 and 10, we show examples of the interface that is available for the risk analysts. Note that, although the results we present are not performed in real-time, the method's performance would be the same in a real-time scenario.

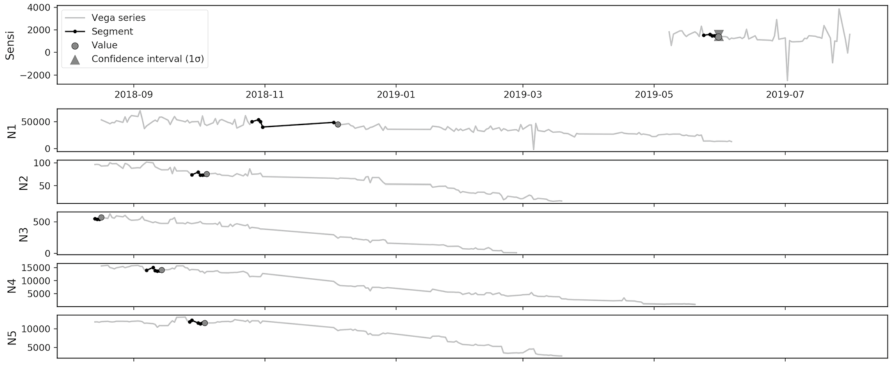


Fig. 9 An example of the trader interface, where we have an instance that is considered valid since the *Value* is in the *Confidence interval*. The time series signaled as N_x correspond to the neighbors. Since the algorithm found neighbors that behave similarly to the current data point we are trying to validate, the instance is also considered valid

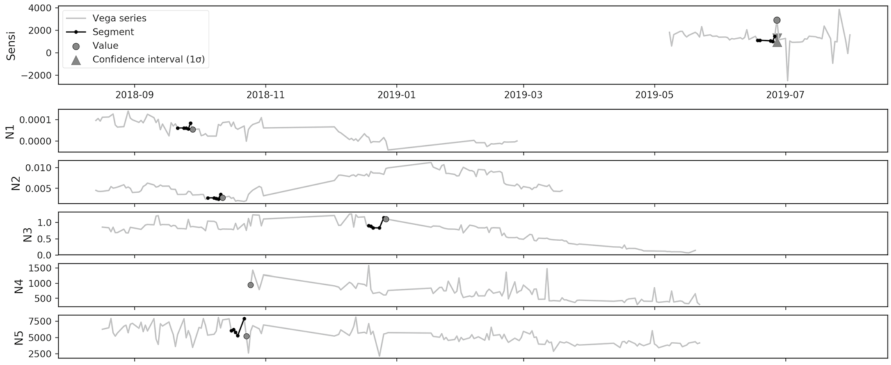


Fig. 10 An example where we have an anomaly. The *value* falls outside of the *Confidence interval*, and, therefore, is not validated. According to the neighbors found in the database, the expected behavior was substantially different from the actual behavior. Therefore, the instance is signaled as requiring manual validation by the risk analysts

6 Conclusions

We created a system that aims to help risk analysts find anomalies. We can also apply this system to several other types of financial data, especially pricing models. Fraud detection and decision support systems are also suitable for this technique. We believe the model can automate many procedures in the right circumstances, substantially reducing the analyst workload. The results show that we can automate the validation of many instances even with a low confidence interval (i.e., using predictions with high certainty). Another potential use case is using the models to create priorities over which instances the analyst should analyze first, using different confidence intervals to define instance priority.

The presented technique is highly transparent, and relevant in the current paradigm of interpretable machine learning.

The absence of labeled data is a limiting factor, making validating the results a tough challenge. We also found it challenging to obtain labeled financial data for anomaly detection. This data is essential to allow us to assess the performance of techniques to understand how the proposed approach can be further improved.

Author Contributions The first author, TMN contributed to the conceptualization, data preparation, programming, and testing of the algorithms, plus writing the article; DS and RS contributed to the data preparation and testing phases; Professors CR and JMM helped conceptualize the algorithms and the reviewing process of the article.

Funding Open access funding provided by FCT|FCCN (b-on). This work is financed by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within Project UIDB/50014/2020. The authors also thank Natixis for providing the datasets for this work.

Data Availability and Materials The datasets are not available due to contractual obligations with our data supplier.

Code Availability Our data supplier has rights over the code produced in our project. Therefore we cannot publish the code. However, the paper describes the implementation in its entirety.

Declarations

Conflict of interest There are no conflicts of interest from any of the authors.

Consent for publication All authors consent for publication.

References

Alsuwailem, A. A. S., Salem, E., & Saudagar, A. K. J. (2022). Performance of different machine learning algorithms in detecting financial fraud. *Computational Economics*. https://doi.org/10.1007/s10614-022-10314-x

Bishop, C. M. (2006). *Pattern recognition and machine learning (Information science and statistics)*. Springer-Verlag.

Black, F. (1973). The pricing of options and corporate liabilities. *Journal of Political Economy, 81*(3), 637–654.

Cappiello, C., Cerletti, C., Fratto, C., & Pernici, B. (2018). Validating data quality actions in scoring processes. *Journal of Data and Information Quality, 9*(2), 1–27. https://doi.org/10.1145/3141248

Christoffersen, P., Heston, S., & Jacobs, K. (2013). Capturing option anomalies with a variance-dependent pricing kernel. *The Review of Financial Studies, 26*(8), 1963–2006.

Douglas, E., Lont, D., & Scott, T. (2014). Finance company failure in New Zealand during 2006–2009: Predictable failures? *Journal of Contemporary Accounting and Economics, 10*(3), 277–295. https://doi.org/10.1016/j.jcae.2014.10.002

Fu, T. C., Chung, F. L., Luk, R., & Ng, C. M. (2007). Stock time series pattern matching: Template-based vs. rule-based approaches. *Engineeirng Applications of Artificial Intelligence, 20*(3), 347–364. https://doi.org/10.1016/j.engappai.2006.07.003

Hautamaki, V., Karkkainen, I., & Franti, P. (2004). Outlier detection using k-nearest neighbour graph. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004., IEEE, Cambridge, UK* pp 430–433 Vol.3, https://doi.org/10.1109/ICPR.2004.1334558,http://ieeexplore.ieee.org/document/1334558/

Hofmann, P., Samp, C., & Urbach, N. (2020). Robotic process automation. *Electronic Markets, 30*(1), 99–106. https://doi.org/10.1007/s12525-019-00365-8

Janczura, J., Trück, S., Weron, R., & Wolff, R. C. (2013). Identifying spikes and seasonal components in electricity spot price data: A guide to robust modeling. *Energy Economics, 38*, 96–110.

Kuhn, M., & Johnson, K. (2013). *Applied predictive modeling*. Springer.

Liew, A. W. C., Yan, H., & Yang, M. (2005). Pattern recognition techniques for the emerging field of bioinformatics: A review. *Pattern Recognition, 38*(11), 2055–2073. https://doi.org/10.1016/j.patcog.2005.02.019

Natixis (2020). https://www.natixis.com/.

Rabiner, L. R. (1990). *Speech recognition based on pattern recognition approaches* (pp. 355–368). Springer-Verlag.

Rigatos, G. (2021). Statistical validation of multi-agent financial models using the H-infinity Kalman filter. *Computational Economics, 58*(3), 777–798. https://doi.org/10.1007/s10614-020-10048-8

Suzuki, K., Shimokawa, T., & Misawa, T. (2009). Agent-based approach to option pricing anomalies. *IEEE Transactions on Evolutionary Computation, 13*(5), 959–972. https://doi.org/10.1109/TEVC.2008.2011745

Wan, Y., Gong, X., & Si, Y. W. (2016). Effect of segmentation on financial time series pattern matching. *Applied Soft Computing, 38*, 346–359. https://doi.org/10.1016/j.asoc.2015.10.012

Wang, J. L., & Chan, S. H. (2007). Stock market trading rule discovery using pattern recognition and technical analysis. *Expert Systems with Applications, 33*(2), 304–315. https://doi.org/10.1016/j.eswa.2006.05.002

Authors and Affiliations

Tiago Mendes-Neves[1,2] · **Diogo Seca**[1] · **Ricardo Sousa**[1] · **Cláudia Ribeiro**[3,4] · **João Mendes-Moreira**[1,2]

[1] LIAAD-INESC TEC - Laboratory of Artificial Intelligence and Decision Support, Porto, Portugal

[2] FEUP - Faculdade de Engenharia da Universidade do Porto, Porto, Portugal

[3] cefUP - Centro de Economias e Finanças da UP, Porto, Portugal

[4] FEP - Faculdade de Economia da Universidade do Porto, Porto, Portugal