A Hybrid COMTE-LEFTIST Time-Series Explanation Method For a Time-series Classification Bitcoin Recommendation System

Lucas Rabelo de Araujo Morais
Universidade Federal de Pernambuco
Recife, PE, Brazil, 50670-901
lram2@cin.ufpe.br

Teresa Bernarda Ludermir
Universidade Federal de Pernambuco
Recife, PE, Brazil, 50670-901
tbl@cin.ufpe.br

Abstract

This paper introduces COMTE-LEFTIST, a hybrid explanation method for time-series classification, specifically applied to a Bitcoin recommendation system. The method combines COMTE, a counterfactual explanation framework, with LEFTIST, a feature-based local explainer, to enhance the interpretability of model predictions. COMTE-LEFTIST evaluates the impact of key shapelets from counterfactual examples on prediction scores, shedding light on how these shapelets influence trading decisions. We tested multiple models using one-minute Bitcoin closing price data across different time windows, with the MRSQM model achieving 70% accuracy for 30-minute sell/hold recommendations, outperforming deep learning models for shorter timeframes. This hybrid approach contributes to greater explainability in time-series classification, demonstrating how the integration of counterfactual and feature-based explanations can improve transparency in AI-driven financial strategies.

1 Introduction

The pioneering work of Nakamoto (2008) introduced Bitcoin as a decentralized payment system, eliminating the need for intermediaries such as financial institutions, governments, and banks. This system operates without relying on trust, instead utilizing blockchain technology, which ensures reliability while offering individuals greater autonomy and privacy in their transactions. As Hellani et al. (2018) noted, while Bitcoin cannot exist without blockchain, the reverse is not true; blockchain technology is independent of Bitcoin. After Bitcoin's inception and subsequent growth, a new market emerged around blockchain, giving rise to other cryptocurrencies like Dash and Ethereum. Caporale et al. (2017) identified predictable patterns within this market, suggesting that these patterns could serve as the foundation for trading strategies. This is where AI comes into play—by leveraging time-series patterns, AI-driven strategies can optimize trading, though they also need to provide transparency for the end user.

As pointed out by Faouzi (2024), standard machine learning classification algorithms are not always well-suited for time-series data due to the complexity of temporal patterns. Prior research, such as that of Kwon et al. (2019) and Fior et al. (2022), has explored classification strategies in cryptocurrency markets. Kwon et al. (2019) employed LSTM and Gradient Boosting to classify cryptocurrency price movements (e.g., price-up or price-down), finding Bitcoin particularly challenging to predict. Meanwhile, Fior et al. (2022) used XGBoost to estimate feature importance across various cryptocurrencies and developed CryptoMLE, an explainable AI tool that reveals how market technical features influence price trends.

Latinx in AI @ NeurIPS-24 (LXAI 2024).

Typically, models such as SHAP and LIME are employed to explain the outputs of black-box models, and these advancements have significantly contributed to the field of model interpretability. In this context, Guillemé et al. (2019) introduced LEFTIST, the first agnostic local explainer specifically designed for time-series classification. Further advancements in time-series model interpretability were made by Ates et al. (2021), who, to the best of our knowledge, developed COMTE—the first method to generate counterfactual explanations for machine learning frameworks in multivariate time series.

In light of this broader context, the goal of this paper is to present a hybrid explanation method, combining COMTE and LEFTIST. Our approach is designed to capture the impact of the most important shapelets from counterfactual explanations on the prediction score, thus enhancing interpretability in time-series classification.

2 Methodology

The machine learning frameworks used in this study include `sktime` (version 0.30.2), developed by Löning et al. (2020), and `tslearn` (version 0.6.3), developed by Tavenard et al. (2020), both of which were utilized for time-series classification (TSC). For tabular classification tasks, we employed `sklearn` (version 1.3.0), developed by Pedregosa et al. (2011). Deep learning (DL) models were implemented using `pytorch` (version 2.3.1). All explainability methods were integrated with `Python 3.10`. The Bitcoin closing price data, with one-minute granularity (in USD), spanning from 2011-09-01 to 2024-08-01, was sourced from the Bitstamp cryptocurrency exchange via the `ccxt` library (version 4.3.58).

2.1 Pre-processing and Training Pipeline

The pre-processing and training pipeline is illustrated in Figure 1. In the first step (highlighted by the black box in the figure), data was collected from the Bitstamp cryptocurrency exchange using the `ccxt` library. In the second step, the full time-series was segmented into smaller time windows of 30 minutes, 1 hour, and 2 hours. Each window was labeled as either "1" or "0", where "1" represents a sell recommendation—indicating that if a certain amount of Bitcoin or satoshis (the smallest unit of Bitcoin) is bought during the specified time window, the average Bitcoin price is expected to rise in the next window. Conversely, "0" represents a hold recommendation, implying that the average Bitcoin price is expected to fall, and thus the market is not favorable for selling Bitcoin or satoshis at that time.

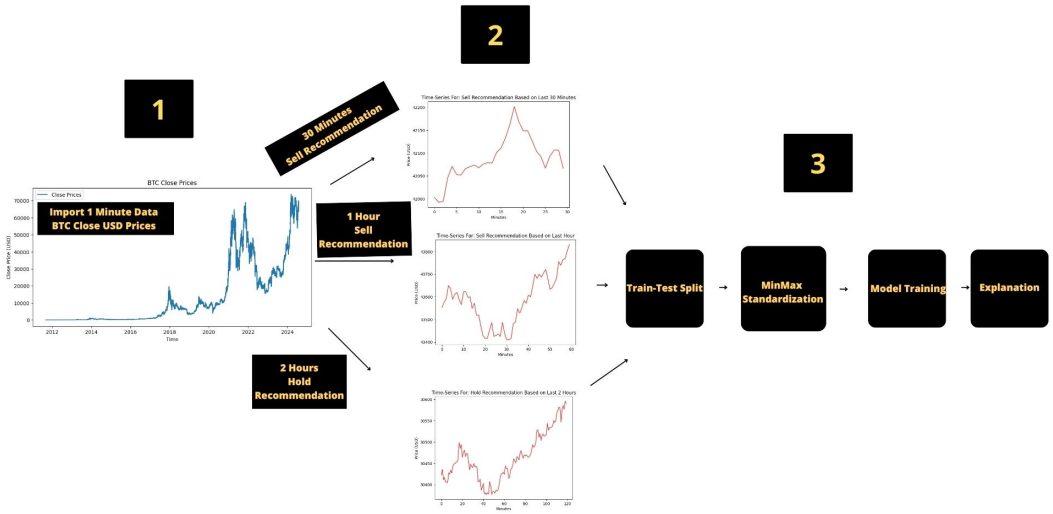

Figure 1: Data Preprocessing and Training Pipeline

In the third step, the training pipeline starts with a train-test split, where 80% of the data was allocated for training and 20% for testing. The split was stratified by label, ensuring that the proportions of classes 0 and 1 were maintained at 45%-55% in both the training and testing datasets across the 30-minute, 1-hour, and 2-hour windows. MinMax standardization was applied, fitted on the training data, and then used to transform the test data to avoid information leakage. Finally, machine learning (ML) and deep learning (DL) models were trained, and agnostic explanation methods were applied to enhance interpretability and support decision-making.

2.2 Classification Models

Three categories of models were selected for this study: ML Tabular models, ML time-series models (specifically designed for time-series classification), and DL models. Each model was trained on different time windows, but the same hyperparameters were maintained across the models to ensure a fair comparison of performance. Table 1 summarizes all models used for each time window. The simplest model employed was the dummy classifier, a time-series-specific model that always predicts the most frequent class label.

Table 1: Summarization of Models

Model	Parameters	Type / Library
MRSQM	`MrSQM(strat='R', random_state=42)`	ML Time-Series / sktime
Catch22	`Catch22Classifier(random_state=3)`	ML Time-Series / sktime
Dummy Classifier	`DummyClassifier(strategy='prior', random_state=3)`	ML Time-Series / sktime
Time-series SVM Classifier	`TimeSeriesSVC(kernel="sigmoid", gamma="auto", probability=True,random_state=42)`	ML Time-Series / tslearn
Composable Time-series Forest	`ComposableTimeSeriesForestClassifier(RocketClassifier(num_kernels=100), n_estimators=10,random_state=4)`	ML Time-Series / sktime
KNN	`KNeighborsClassifier(n_neighbors=5)`	ML Tabular / sklearn
SVM Classifier	`SVC(random_state = 42,probability=True, kernel = 'sigmoid',gamma="auto")`	ML Tabular / sklearn
XGBoost Classifier	`XGBClassifier(objective='binary:logistic',random_state = 42)`	ML Tabular / xgboost
Random Forest Classifier	`RandomForestClassifier(random_state = 5)`	ML Tabular / sklearn
CNN-GRU with Attention	Conv1D; MaxPooling1D; GRU Layer; Attention Layer; GlobalAveragePooling1D; BatchNormalization; Output Layer	Deep Learning / pytorch
CNN-LSTM	Conv1D Layer; AdaptiveMaxPooling1D; Flatten Layer; Fully Connected Layer; BatchNormalization; Output Layer	Deep Learning / pytorch
Fully Conected MLP	Flatten Layer; Fully Connected Layer; Output Layer	Deep Learning / pytorch
BiLSTM	Bidirectional LSTM; BatchNormalization; Output Layer	Deep Learning / pytorch

Among the time-series classification models, MRSQM (Multiple Representations Sequence Miner), introduced by Nguyen and Ifrim (2022), follows a three-step process. First, it performs symbolic transformation, converting numeric series into multiple symbolic representations using either Symbolic Aggregate Approximation (SAX) or Symbolic Fourier Approximation (SFA). Next, it selects relevant features from these symbolic representations, and finally, it applies a logistic regression learning algorithm. The variant used in this work, MRSQM-R, employs unsupervised feature selection.

Another time-series classification algorithm used was Catch22 (Canonical Time-series Characteristics classifier), developed by Lubba et al. (2019). This approach condenses a pool of 4791 time-series features from 93 classification problems into 22 high-performance features. In this study, the `sktime` library utilized a RandomForestClassifier as the default estimator for Catch22. Additionally, Support Vector Machine (SVM) classifiers were applied for time-series classification, as seen in prior studies by Zhang et al. (2010) and Cuturi (2011). The Composable Time-series Forest classifier, proposed by Deng et al. (2013), was also employed due to its ability to capture complex temporal patterns and effectively differentiate between time-series from different classes.

For traditional machine learning models, we used widely recognized algorithms available in the `scikit-learn` library. Regarding deep learning models, we employed architectures like Convolutional Neural Networks (CNNs), Long Short-Term Memory networks (LSTMs), Gated Recurrent Units (GRUs), and attention mechanisms. Additionally, a simpler fully connected Multi-Layer Perceptron (MLP) was tested. As highlighted by Du et al. (2018), LSTMs are effective at capturing long-term temporal dependencies (the same applies to GRUs), CNNs capture spatial sparsity and heterogeneity in the data, and attention mechanisms enhance the model's ability to extract more complex features from each layer. All Neural Network models were trained with 1000 epochs, using an early stop strategy based on the validation loss (if the loss did not improve for 50 epochs then the training had to stop). The Kruskal-wallis test, a non-parametric test created by Kruskal and Wallis (1952), was used to verify if there was any statistically significant difference between different classes of models, since we didn't have much experimental data and normality assumptions couldn't be fulfilled, a statistical significance level of 5% was chosen for the experiment.

2.3 Explanation Models

SHAP (Lundberg and Lee, 2017) and LIME (Ribeiro et al., 2016) are traditionally used as model-agnostic methods for explaining classification algorithms and quantifying the impact of features on predictions at both local and global scales. LIME estimates feature importance by minimizing the

following equation 1, where $g \in G$ and G is the class of linear models, $\Omega(g)$ is low enough, in order to attain interpretability and local fidelity, $\mathcal{L}$ is the loss function and $\Pi_x(z) = \exp\left(-\frac{D(x,z)^2}{\sigma^2}\right)$.

$$\xi(x) = *argmin_{g \in G} \left(\mathcal{L}(f, g, \pi_x) + \Omega(g)\right) \tag{1}$$

Whereas Shapley values of each feature are estimated by equation 2, where F is the set of all features, $S \subseteq F$, x_S are the input feature values in the set S. The terms $f_{S\cup\{i\}}$ and f_S correspond to models trained with and without feature i respectively.

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \left(f_{S\cup\{i\}}(x_{S\cup\{i\}}) - f_S(x_S)\right) \tag{2}$$

However, both LIME and SHAP face challenges when dealing with time-series data. To address this, the LEFTIST approach leverages shapelets—discriminative time-series subsequences that are representative of class membership—as interpretable features. Detailed information about LEFTIST can be found in Guillemé et al. (2019). In LEFTIST, time-series are segmented into interpretable components denoted as $S(t)$, with m_t^j representing the j^{th} neighbor of $m_t = (1, ..., 1)$. The explanation model is defined through the function h_t as shown below:

$$h_t(m_j^t) = \bigoplus_{i=1}^{m} \begin{cases} S_i^t & if m_t^{j,i} = 1 \\ transform_t(S_i^t) & if m_t^{j,i} = 0 \end{cases} \tag{3}$$

For this study, we utilized the `TSinterpret` library, version 0.4.5, developed by Höllig et al. (2023). The `LEFTIST` function within this library was applied, with the uniform transformation function chosen for this case. LIME values were used to estimate feature importance within the transformed space. Additionally, `TSinterpret` provides counterfactual explanations through the COMTE function. As described by Ates et al. (2021), COMTE employs a greedy search algorithm that selects candidates from the training set (referred to as distractors) for counterfactual generation. These distractors ensure that the counterfactuals are derived from actual data samples, preventing the use of synthetic data, which might lead to non-meaningful explanations. COMTE identifies the distractor that minimizes the following Equation 4.

$$L(f, c, A, x') = \left((\tau - f_c(x'))^+\right)^2 + \lambda \left(\|A\|_1 - \delta\right)^+ \tag{4}$$

In this equation, c, represents the class of interest, τ is the target probability, $f_c(x)$ is the probability for class c, f represents the classification model. A is a binary diagonal matrix, λ is a tuning parameter, x' is the optimum counterfactual explanation and δ is the threshold below which reducing the number of variables does not improve explanations.

2.4 COMTE-LEFTIST

In the proposed method, the LEFTIST and COMTE approaches are combined to emphasize the impact of the most important shapelets from the counterfactual explanation on class predictions. To generate these hybrid explanations, a mask is constructed using the positive LIME or SHAP values derived from the LEFTIST approach. This mask filters out only the shapelets that contribute positively to the class prediction. The complete process is illustrated in Figure 2.

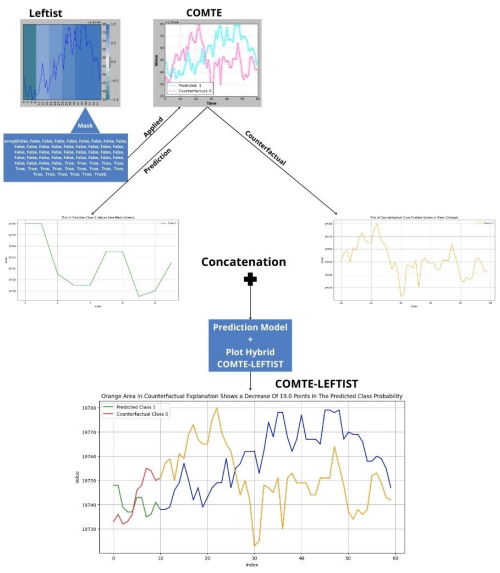

Figure 2: COMTE-LEFTIST Pipeline

The process begins by generating a mask of positive LEFTIST values, estimated either by LIME or SHAP. In this mask, positive values are marked as True, while negative or zero values are marked as False. This mask is then applied to the COMTE values, where the positive values are extracted to form part of the counterfactual explanation, while the non-positive values retain the shapelets from the predicted time-series. The predicted and counterfactual values under the mask are then concatenated, after which they are fed into the prediction model to compute the effect of the counterfactual explanation on the prediction score, as represented by Equation 5.

$$h(f, x, x') = f(x) - f\left(\bigoplus_{i=1}^{m} \begin{cases} x_i & if\ mask^i = False \\ x'_i & if\ mask^i = True \end{cases}\right) \tag{5}$$

In Equation 5, $mask = (True, \ldots, False)$, denotes the mask of LEFTIST values, where $i = 1, \ldots, m$ refers to the position of the time-series values. The variable x represents the values of the predicted time-series, while x' corresponds to the counterfactual values. The operator $\bigoplus$ represents concatenation, and f is the prediction function, which may or may not include standardized values depending on the training configuration of the model.

The hybrid explanation is then visualized by plotting the most important shapelets of both the counterfactual and predicted series, each highlighted in different colors. This visualization shows the impact of the highlighted counterfactual shapelets on the final prediction score, offering a clearer understanding of how these counterfactual components influence the model's decision-making process.

3 Results

The models were evaluated based on their accuracy, with the complete set of metrics presented in Tables 2 and 3. Across all time windows, the best-performing model was MRSQM, surpassing even deep learning (DL) models. For 30-minute time-series, MRSQM achieved 70% accuracy in providing sell and hold recommendations. For 60-minute windows, the accuracy was 69%, and for 120-minute windows, it was 68%. The performance of deep learning models was comparatively better for the 120-minute time window, exceeding that of the machine learning (ML) tabular models. However, for the other time windows (30 minutes and 60 minutes), ML models slightly outperformed DL models. Overall, the time-series-specific models consistently outperformed the tabular-based ones in all cases,

SVM based models, both in ML tabular and time-series classes, were worst than the baseline model, all DL models outperformed the baseline model.

Table 2: Prediction Accuracy for Time Series Specific and ML Tabular Models

Time Window	Time Series Specific					Machine Learning Tabular			
	Dummy Classifier	Catch22	MRSQM	SVC	TS Forest	Random Forest	KNN	SVC	XGB
30 minutes	0.55	0.64	**0.70**	0.48	0.66	0.68	0.65	0.53	0.57
1 Hour	0.54	0.63	**0.69**	0.48	0.66	0.68	0.65	0.52	0.58
2 Hours	0.54	0.64	**0.68**	0.51	0.66	0.60	0.58	0.53	0.57

Table 3: Prediction Accuracy for Deep Learning Models

Time Window	Deep Learning			
	Attention CNN-GRU	CNN-LSTM	MLP	BiLSTM
30 minutes	0.64	0.64	0.56	0.63
1 Hour	0.64	0.63	0.56	0.61
2 Hours	0.64	0.63	0.58	0.59

However, we cannot state that time-series specific ML models are statistically better than DL or ML tabular models, since given the Kruskal-Wallis test we don't have enough evidence at the 5% level of significance to reject the hypothesis that all groups of models are equal (P-value = 0.8386), that's why, if the focus is mainly on accuracy of the models, it's important to test between different model classes before selecting one. Regarding precision of both classes, with results in Tables 4, 5 and 6, MRSQM (model with best accuracy) achieved the highest precision in class 1 for all time windows, meaning that it's the most precise model in giving Sell recommendations.

Table 4: Prediction Precision for Time Series Specific Models (Class 0) (Class 1)

Time Window	Time Series Specific				
	Dummy Classifier	Catch22	MRSQM	SVC	TS Forest
30 minutes	(0.00)(0.55)	(0.61)(0.68)	**(0.65)(0.74)**	(0.43)(0.52)	(0.61)(0.70)
1 Hour	(0.00)(0.54)	(0.61)(0.66)	**(0.66)(0.73)**	(0.44)(0.52)	(0.62)(0.71)
2 Hours	(0.00)(0.54)	(0.62)(0.66)	**(0.65)(0.71)**	(0.47)(0.54)	(0.62)(0.70)

Table 5: Prediction Precision for ML Tabular Models (Class 0) (Class 1)

Time Window	Machine Learning Tabular			
	Random Forest	KNN	SVC	XGB
30 minutes	(0.65)(0.71)	(0.62)(0.67)	(0.48)(0.57)	(0.59)(0.57)
1 Hour	(0.66)(0.70)	(0.62)(0.66)	(0.48)(0.56)	(0.59)(0.58)
2 Hours	(0.66)(0.68)	(0.62)(0.64)	(0.50)(0.53)	(0.62)(0.61)

Table 6: Prediction Precision for Deep Learning Models (Class 0) (Class 1)

Time Window	Deep Learning			
	Attention CNN-GRU	CNN-LSTM	MLP	BiLSTM
30 minutes	(0.70)(0.62)	(0.71)(0.62)	(0.64)(0.55)	(0.65)(0.63)
1 Hour	(0.73)(0.61)	(0.72)(0.61)	(0.83)(0.55)	(0.75)(0.59)
2 Hours	(0.75)(0.61)	(0.71)(0.61)	(0.76)(0.56)	(0.62)(0.58)

Figure 3 presents a set of random time-series for a 1-hour time window, where the MRSQM model, identified as the top-performing model across all time frames, was used to generate hybrid explanations by combining the COMTE and LEFTIST methods. To ensure reproducibility, a random seed of 2 was used (employing the `random` and `numpy` libraries). In these figures, the orange highlighted line

in the counterfactual series marks the most significant time window in the counterfactual explanation, based on LEFTIST values from the predicted series. The impact on prediction probability is indicated in each figure's title. The blue highlighted line represents the most important segment of the predicted series influencing the class prediction.

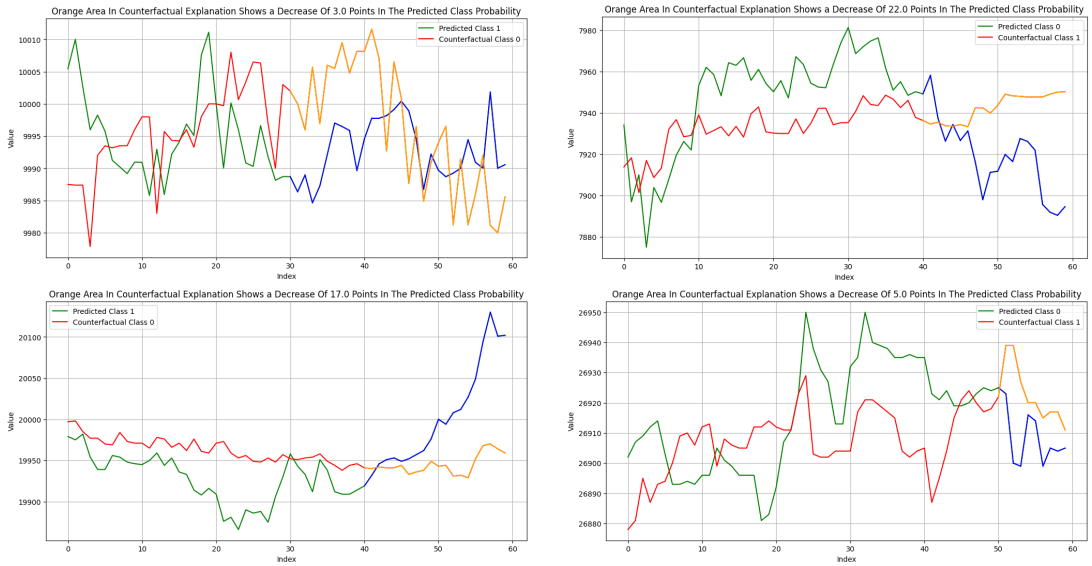

Figure 3: COMTE-LEFTIST Hybrid Model

For the explanations where the series were classified as Class 1 (Sell Recommendation), shown in Figure 3, the first series (top-left) demonstrates a stable pattern. Considering the highlighted counterfactual explanation, the model would not give a sell recommendation if the series exhibited more chaotic behavior, followed by a drop in Bitcoin prices during that period. In the second series of Class 1 (bottom-left), there is a notable rise in Bitcoin price within the highlighted area. Based on the counterfactual explanation, the model would not issue a sell recommendation if the Bitcoin price had been slowly declining, followed by only a minor increase in the last minutes of the time window.

For the Hold recommendation (Class 0) predictions, displayed on the right side of Figure 3, the first explanation (top-right) indicates that a hold recommendation would not have been considered if there had been a price drop followed by a subsequent rise and stabilization in the final minutes. In the bottom-right series, the sell recommendation would only be triggered if, following the highlighted counterfactual explanation, an increase in Bitcoin price was followed by a noticeable devaluation in the last minutes of the time window.

4 Discussion

Time-series specific models have outperformed ML tabular models, which is also reported by Faouzi (2024) stating that standard ML classification algorithms are not always well-suited for time series data. Regarding DL models, although these methods did not outperform the time-series specific models used in this study, Fawaz et al. (2019) conducted a review on DL methods for TSC and found that end-to-end deep learning architectures can achieve state-of-the-art performance. However, a fair comparison is necessary, as training time can be a limiting factor when working with DL.

The model that achieved the best performance was the MRSQM for 30 minutes, achieving 70% accuracy, which means that out of 10 attempts, the model correctly predicts 7 of the Hold and Sell recommendations. Considering the methodological differences, given that our F1-score was also 70%, the performance of the MRSQM is comparable to that found in the work of Kwon et al. (2019), who compared Gradient Boosting (GB) and LSTM for TSC of various cryptocurrencies. In their best performance predicting Bitcoin price trends, the authors, using an F1-score, reported approximately 69% with LSTM, which surpassed the GB algorithm. Ranjan et al. (2023) also used a machine

learning approach to classify price increases and decreases of Bitcoin, with daily and 5-minute granularity data. They achieved the best performance with logistic regression (accuracy of 64.8%) for daily price predictions, whereas for minute granularity their best performance was 59.4%. Therefore, given methodological differences, the MRSQM is as good as or better than the models presented by these authors.

A novel and promising approach that merges time-series classification and image classification to predict Bitcoin prices (increase, decrease, and stability) was proposed by Yamak et al. (2024). The authors state that this approach enhances the unique areas of a sequence and establishes temporal correlations, leading to improved accuracy. Indeed, their model, Wide-TSNet, achieved 94% accuracy. However, explainability was not explored in their work, and explaining the transition of a time-series to an image via Markov Transition Fields might complicate understandings for those outside the AI field. Given this, approaches using solely TSC methods (for classifying Bitcoin prices, EEG, equipment failure, motion recognition, etc.) still need to benefit from explainability methods.

Previous works with Bitcoin and cryptocurrencies have utilized explainability in various tasks. Fior et al. (2022), Babaei et al. (2022), and Gupta et al. (2023) have respectively used SHAP to explain features associated with price directions (upward and downward trends) through classification models, for cryptocurrency portfolio management, and to explain via regression models features associated with downward and upward trends for Ethereum and Solana. These approaches illuminate what is happening underneath the models and aid decision-making. However, none dealt with univariate time-series, relying on other features to capture temporal behavior. By using LEFTIST or COMTE, recent advances in the field of explainable AI in TSC for Bitcoin and cryptocurrency recommendation systems based on classification models, which may or may not focus on intraday trading (like the experiments we conducted), systems can be more transparent and visually appealing to end users, ranging from domain experts to people with basic knowledge of the crypto market, since neither SHAP nor LIME provides time-series as explanations.

5 Conclusion

This work introduced a hybrid approach named COMTE-LEFTIST, combining two well-established agnostic explanation models for time-series classification (TSC). Despite certain limitations encountered during the experiments, the contribution of hybrid explanations is crucial for advancing the field. As noted by Rojat et al. (2021), significant progress is still needed in explainability for time-series data. The hybridization of counterfactual and feature-based methods presented here represents a small yet important step towards more sophisticated and optimized explainability techniques in TSC. By employing TSC-specific machine learning algorithms such as MRSQM, we achieved 70% accuracy in generating Sell and Hold recommendations for Bitcoin, furthermore there was no statistically significant difference between different classes of models, so we highlight that it's important to experiment between them to find the one with best accuracy, since it may depend on the problem and complexity of the time-series. Emerging methods like Wide-TSNet, which integrate time-series and image classification, have shown promising results and warrant further exploration. In future research, surrogate models could be trained on these advanced models to enhance explainability using COMTE-LEFTIST and other time-series XAI (Explainable AI) techniques. The experiments, models, and Python notebooks supporting this work are maintained in a GitHub repository. For access to the source code or further inquiries, readers are encouraged to contact the authors directly.

5.1 Limitations

Limitations in our work must be addressed. One key issue with COMTE-LEFTIST is its tendency to exhibit highly stochastic behavior, largely due to the random nature of the LEFTIST component. This randomness can affect the calculation of the counterfactual explanation's impact on the predicted class probability, leading to higher variability in the results. In some cases, instead of reducing the predicted class probability, the explanation may show an increasing or no effect at all. To mitigate this, it is important to use different random seeds when generating explanations and to change the seed if the counterfactual explanation unexpectedly increases the predicted class probability. However, the underlying causes of this randomness were not fully explored in this paper and warrant further investigation in future studies.

References

Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Retrieved from `https://www.bitcoin.org`.

Hellani, H., Samhat, A. E., Chamoun, M., El Ghor, H., & Serhrouchni, A. (2018). On BlockChain Technology: Overview of Bitcoin and Future Insights. In *2018 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET)* (pp. 1-8). doi: 10.1109/IMCET.2018.8603029.

Caporale, G. M., Gil-Alaña, L. A., & Plastun, A. (2017). Persistence in the Cryptocurrency Market. *CESifo Working Paper, No. 6811*, Center for Economic Studies and ifo Institute (CESifo), Munich.

Faouzi, J. (2024). Time Series Classification: A Review of Algorithms and Implementations. In *Time Series Analysis - Recent Advances, New Perspectives and Applications*. IntechOpen. Available from: `http://dx.doi.org/10.5772/intechopen.1004810`.

Kwon, D., Kim, J., Heo, J., Kim, C., & Han, Y. (2019). Time Series Classification of Cryptocurrency Price Trend Based on a Recurrent LSTM Neural Network. *Journal of Information Processing Systems*, 15(3), 694-706. DOI: 10.3745/JIPS.03.0120.

Fior, J., Cagliero, L., & Garza, P. (2022). Leveraging Explainable AI to Support Cryptocurrency Investors. *Future Internet*, 14, 251. https://doi.org/10.3390/fi14090251

Guillemé, M., Masson, V., Rozé, L., & Termier, A. (2019). Agnostic Local Explanation for Time Series Classification. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, Portland, OR, USA, pp. 432-439. doi: 10.1109/ICTAI.2019.00067.

Ates, E., Aksar, B., Leung, V. J., & Coskun, A. K. (2021). Counterfactual Explanations for Multivariate Time Series. In *2021 International Conference on Applied Artificial Intelligence (ICAPAI)*, Halden, Norway, pp. 1-8. doi: 10.1109/ICAPAI49758.2021.9462056.

Markus Löning, Anthony Bagnall, Sajaysurya Ganesh, Viktor Kazakov, Jason Lines, Franz Király (2019): “sktime: A Unified Interface for Machine Learning with Time Series".

Tavenard, R., Faouzi, J., Vandewiele, G., Divo, F., Androz, G., Holtz, C., Payne, M., Yurchak, R., Rußwurm, M., Kolar, K., & Woods, E. (2020). Tslearn, A Machine Learning Toolkit for Time Series Data. *Journal of Machine Learning Research*, 21(118), 1-6. Retrieved from `http://jmlr.org/papers/v21/20-091.html`.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830.

Nguyen, T. L., & Ifrim, G. (2022). MrSQM: Fast Time Series Classification with Symbolic Representations. *arXiv preprint* arXiv:2109.01036. Retrieved from `https://arxiv.org/abs/2109.01036`.

Lubba, C. H., Sethi, S., Knaute, P., Schultz, S. R., Fulcher, B. D., & Jones, N. S. (2019). catch22: Canonical time-series characteristics. *Data Mining and Knowledge Discovery*, 33(6), 1821-1852. https://doi.org/10.1007/s10618-019-00647-x.

Cuturi, M. (2011). Fast Global Alignment Kernels. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*.

Zhang, D., Zuo, W., Zhang, D., & Zhang, H. (2010). Time Series Classification Using Support Vector Machine with Gaussian Elastic Metric Kernel. In *Proceedings of the 2010 20th International Conference on Pattern Recognition (ICPR)* (pp. 29-32). Istanbul, Turkey. https://doi.org/10.1109/ICPR.2010.16.

Deng, H., Runger, G., Tuv, E., & Vladimir, M. (2013). A time series forest for classification and feature extraction. *Information Sciences*, 239, 142-153.

Du, Q., Gu, W., Zhang, L., & Huang, S.-L. (2018). Attention-based LSTM-CNNs for Time-series Classification. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems (SenSys '18)*. https://doi.org/10.1145/3274783.3275208.

Kruskal, W. H., Wallis, W. A. (1952). Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47, 583–621.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/2939672.2939778.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS '17)* (pp. 4768-4777). Red Hook, NY, USA: Curran Associates Inc.

Höllig, J., Kulbach, C., & Thoma, S. (2023). TSInterpret: A Python Package for the Interpretability of Time Series Classification. *Journal of Open Source Software*, 8(85), 5220. https://doi.org/10.21105/joss.05220.

Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., & Muller, P.-A. (2019). Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33, 917–963. https://doi.org/10.1007/s10618-019-00619-1.

Ranjan, S., Kayal, P., & Saraf, M. (2023). Bitcoin price prediction: A machine learning sample dimension approach. *Computational Economics*, 61, 1617–1636. https://doi.org/10.1007/s10614-022-10262-6.

Yamak, P. T., Li, Y., Zhang, T., & Gadosey, P. K. (2024). Wide-TSNet: A novel hybrid approach for Bitcoin price movement classification. *Applied Sciences*, 14(9), 3797. https://doi.org/10.3390/app14093797.

Babaei, G., Giudici, P., & Raffinetti, E. (2022). Explainable artificial intelligence for crypto asset allocation. *Finance Research Letters*, 47(B), 102941. `https://doi.org/10.1016/j.frl.2022.102941`.

Gupta, A., et al. (2023). Cryptocurrency prediction and analysis between supervised and unsupervised learning with XAI. *2023 IEEE International Conference on Blockchain and Distributed Systems Security (ICBDS)*, New Raipur, India, pp. 1-7. `https://doi.org/10.1109/ICBDS58040.2023.10346583`.

Rojat, T., Puget, R., Filliat, D., Del Ser, J., Gelin, R., & Díaz-Rodríguez, N. (2021). Explainable Artificial Intelligence (XAI) on time series data: A survey. arXiv preprint. `https://arxiv.org/abs/2104.00950`.