Unraveling Block Maxima Forecasting Models with Counterfactual Explanation

Yue Deng
Michigan State University
East Lansing, Michigan, USA
dengyue1@msu.edu

Asadullah Hill Galib
Michigan State University
East Lansing, Michigan, USA
galibasa@msu.edu

Pang-Ning Tan
Michigan State University
East Lansing, Michigan, USA
ptan@msu.edu

Lifeng Luo
Michigan State University
East Lansing, Michigan, USA
lluo@msu.edu

ABSTRACT

Disease surveillance, traffic management, and weather forecasting are some of the key applications that could benefit from block maxima forecasting of a time series as the extreme block maxima values often signify events of critical importance such as disease outbreaks, traffic gridlock, and severe weather conditions. As the use of deep neural network models for block maxima forecasting increases, so does the need for explainable AI methods that could unravel the inner workings of such black box models. To fill this need, this paper presents a novel counterfactual explanation framework for block maxima forecasting models. Unlike existing methods, our proposed framework, *DiffusionCF*, combines deep anomaly detection with a conditional diffusion model to identify unusual patterns in the time series that could help explain the forecasted extreme block maxima. Experimental results on several real-world datasets demonstrate the superiority of *DiffusionCF* over other baseline methods when evaluated according to various metrics, particularly their informativeness and closeness. Our data and codes are available at https://github.com/yue2023cs/DiffusionCF.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**.

KEYWORDS

Explainable AI, counterfactual explanation, time series forecasting

ACM Reference Format:
Yue Deng, Asadullah Hill Galib, Pang-Ning Tan, and Lifeng Luo. 2024. Unraveling Block Maxima Forecasting Models with Counterfactual Explanation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, Barcelona, Spain, 12 pages. https://doi.org/10.1145/3637528.3671923

1 INTRODUCTION

Block maxima forecasting is the task of predicting the maximum value of a time series for a future time window. Such a forecasting task has widespread applicability in many practical domains such as weather forecasting, disease monitoring, traffic management, and financial risk assessment. Block maxima forecasting models play a crucial role in these domains as the predicted block maxima can provide early warning to stakeholders about an impending severe event. Despite their growing importance, there is a noticeable gap in current research regarding the explainability of these models. Explainability is important for block maxima forecasting models as it enables stakeholders to comprehend and trust the model's predictions, fostering transparency and informed decision-making, particularly in critical scenarios. The growing field of explainable AI therefore plays a crucial role in this context, offering methodologies and tools that can enhance the explainability of these models.

Explainable AI involves two primary methodologies: feature attribution and counterfactual explanation methods [23]. Feature attribution methods, such as LIME [25], SHAP [21], Grad-CAM [26], CRP [1], and adversarial examples [10, 28], focus on elucidating the conditions behind a model's decision, shedding light on the influential features or input values. Differently, counterfactual explanation methods [13] seek to discover the smallest modification (*i.e.*, changes) to the input that leads to a completely opposite forecast, *a.k.a.*, *counterfactual target*, by the black box model. Counterfactual explanation methods are appealing as they offer a powerful means to explore alternative scenarios and assess the impact of different conditions on forecasting outcomes. For block maxima forecasting, the counterfactual instances enable us to identify historical patterns in the time series that may help explain the forecasted extreme block maxima so actions can be taken to prevent their future occurrence. In the example depicted in Figure 1, a prior incident of epidemic outbreak may likely explain the forecasted next wave by the black box model. The forecast is juxtaposed against a counterfactual scenario which assumes preventative intervention had been taken to mitigate the likelihood of the subsequent outbreak.

For time series, a good counterfactual instance must be (1) *informative*, *i.e.*, identifies the contrastive segment in the time series that explains the generated prediction by the black box model, (2) *closely mimics* the original time series, and (3) *realistic*, *i.e.*, drawn from the same distribution as the majority of the time series data. However, striking a balance among the three criteria can be tricky. For instance, neighborhood-based methods [8, 20, 33] consider the nearest training instance whose prediction matches the counterfactual target and utilize or partially modify them to form the

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-0490-1/24/08
https://doi.org/10.1145/3637528.3671923

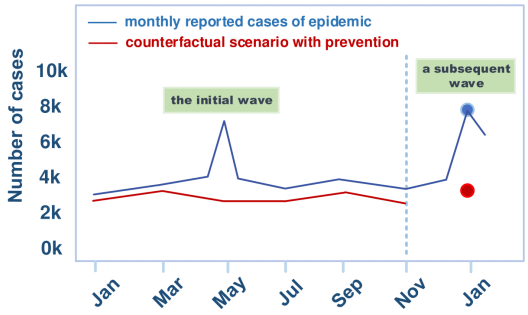

Figure 1: An illustration of counterfactual explanation for block maxima forecast of a disease outbreak. The blue dot denotes the forecasted epidemic outbreak while the red dot represents a counterfactual scenario devoid of any outbreak.

counterfactual instance. While the counterfactual instance found by these methods is quite realistic since it corresponds or is close to an actual training instance, it may not resemble the original time series. On the other hand, gradient-based methods [5, 30, 32] learn a counterfactual instance by perturbing the time series in such a way that maintains closeness to their original time series. Yet these methods may induce modification across the entire time series, making it difficult to pinpoint exactly the segment in the time series that helps explain the model forecast, thereby reducing its informativeness.

In this paper, we present a novel counterfactual explanation framework for block maxima forecasting models. Unlike counterfactual explanation for time series classification [3, 8, 18], choosing the right counterfactual target value for block maxima is non-trivial since the block maxima are continuous-valued, which means, there are infinitely many possible counterfactual targets to choose from. To address this challenge, we propose a principled way to create the counterfactual target by leveraging the generalized extreme value (GEV) distribution [7], which governs the distribution of block maxima values of a time series. Next, to ensure that the counterfactual instance is informative, we constrain the area for modification by identifying abnormal segments within the original time series, departing from the conventional practice of considering the entire time series for perturbation when constructing the counterfactual instance. Specifically, we apply anomaly detection to each segment of the time series and extract a subset of the segments with the highest anomaly scores as possible candidates for replacement. For each candidate, we employ a conditional diffusion model [29] to generate a new time series segment to replace the identified anomalous segment. This strategy of constructing a counterfactual instance by replacing only its anomalous segment helps create counterfactual instances that are informative, yet close to the original time series.

Our overall proposed framework, named *DiffusionCF*, encapsulates this comprehensive approach. The primary contributions of this work can be summarized as follows:

(1) We introduce the novel problem of counterfactual explanation for block maxima forecasting models in time series, where the counterfactual instances help identify anomalous patterns in the time series that lead to extreme values in the forecasted block maxima.
(2) We propose a method to create a counterfactual target for the block maxima by leveraging the generalized extreme value (GEV) distribution.
(3) We present *DiffusionCF*, a framework that balances the trade-off between generating counterfactual instances that are informative, yet realistic and close to the original time series.
(4) We perform extensive experiments comparing *DiffusionCF* against other baseline methods under different experimental settings. We demonstrate the versatility and effectiveness of *DiffusionCF* across different real-world domains.

2 RELATED WORK

Extreme value theory (EVT) [7] offers a well-grounded approach for modeling and forecasting extreme values in time series. The theory has recently been incorporated into various deep-learning formulations. For instance, Nishino et al. [24] proposed to predict the maximum value in a forecast window using GRU with the generalized extreme value (GEV) distribution. DeepExtrema [12] is another approach that uses deep learning to estimate parameters of the GEV distribution for block maxima forecasting. The GEV distribution has also been used to impute missing values in time series for block maxima forecasting task [11]. Despite these advances, the forecasts generated by the black box models can be hard to explain.

Explainable AI, as described by Molnar [23], encompasses two main methodologies: feature attribution methods, such as LIME [25], SHAP [21], and Grad-CAM [26], or counterfactual explanation methods. LIME [25], a representative feature attribution method, explains the predictions of complex machine learning models by approximating them locally with simpler models such as linear regression or decision trees. Differently, counterfactual explanation methods [13] focus on identifying the smallest modifications to the input features that would alter the model's decision, offering insights into how different inputs could lead to different outcomes. Such methods have been applied to various domains, including recommender systems [6, 31], computer vision [9, 16], and natural language processing [4]. For time series prediction, there are several ways to generate the counterfactual explanation. First, gradient-based methods [3, 18, 22, 30, 32] can be used to create counterfactual instances by minimizing the loss between the model prediction and the desired counterfactual target while maintaining the similarity between the perturbed and original instances. Attribution analysis methods [19] leverage domain knowledge to select the input features and quantify their impact on the counterfactual target. Our work differs from past research in several key aspects. First, we emphasize deriving counterfactual explanations for regression instead of classification tasks, which have been the focus of many previous studies. Second, we utilize generative AI to produce realistic counterfactual instances, which is a challenge for gradient-based methods. Third, unlike attribution analysis methods that require domain knowledge with a limited number of factors, our method can automatically identify the explanatory factors responsible for the prediction.

Generative models [2, 17] have emerged as a key machine learning paradigm in recent years due to their capacity to synthesize

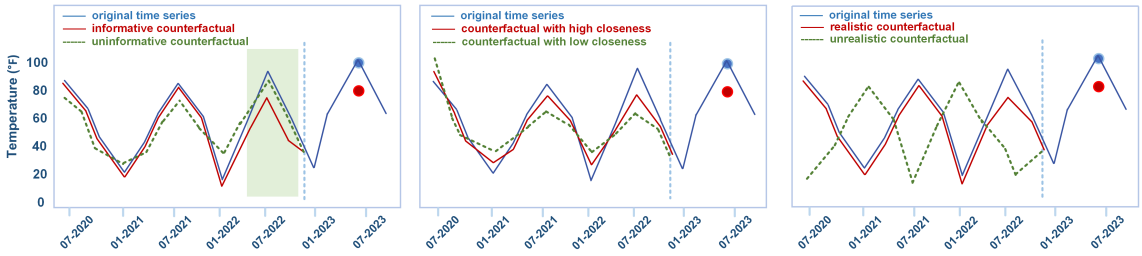

Figure 2: Comparison of counterfactual instances under the informativeness (left), closeness (middle), and realisticness (right) criteria. The blue dot in each figure denotes the forecasted extreme block maxima by a black box model for the period between January and September in 2023, while the red dot denotes its corresponding counterfactual target block maxima.

realistic samples replicating the underlying data distribution. Diffusion models [14, 27, 29] have recently emerged as a cutting-edge approach due to their ability to produce high-quality samples. These models have found success in diverse applications, including time series imputation [29] and counterfactual explanation for images [15]. However, their use for counterfactual explanations of extreme values in time series remains largely unexplored.

3 PRELIMINARIES

3.1 Problem formulation

Consider a time series $\mathcal{Z} = z_1 z_2 \dots z_T$, where T denotes the length of the time series. Assume the time series is partitioned into a set of distinct time windows, each denoted as $w_t = [t-\alpha, t+\beta]$, respectively, where t denotes the current time step. Each window encompasses a predictor time series, $X_t = z_{t-\alpha} z_{t-\alpha+1} \dots z_t$, where $\alpha+1$ is the length of the predictor window, and a forecast time series, $Y_t = z_{t+1} z_{t+2} \dots z_{t+\beta}$, where β is the length of the forecast window. The predictor window $[t-\alpha, t]$ contains historical data or other input variables employed by the forecasting model to generate its predictions, whereas the forecast window $(t, t+\beta]$ contains the future values to be predicted by the model. Let $y_t = \max_{\tau \in 1,\dots,\beta} z_{t+\tau}$ be the block maxima of Y_t, *i.e.*, the maximum value over the forecast window. Furthermore, we denote $w^k_{t-\tau} = [t-\tau+1, t-\tau+k]$ as a sub-interval within w_t such that $X_t(w^k_{t-\tau}) = z_{t-\tau+1} z_{t-\tau+2} \cdots z_{t-\tau+k}$ is the corresponding length-k time series segment of X_t, where $k \le \tau \le \alpha + 1$.

Let f be a black box model that generates a block maxima forecast, $\hat{y}_t$, for any given input X_t, *i.e.*, $\hat{y}_t = f(X_t)$. Our primary goal is to construct a *counterfactual predictor*, X'_t, such that $f(X'_t) \approx \hat{y}'_t$, where $\hat{y}'_t \neq \hat{y}_t$ is the desired *counterfactual target*. For example, if $\hat{y}_t$ corresponds to an extreme block maxima generated by the model f, then the counterfactual target $\hat{y}'_t$ would be a non-extreme block maxima value. In this paper, we employ the Generalized Extreme Value (GEV) Distribution to define whether a block maxima value is extreme or non-extreme. If the forecasted block maxima $\hat{y}_t$ is extreme, then a ***counterfactual instance*** $(X'_t, \hat{y}'_t)$ is generated. To do so, the counterfactual predictor, X'_t, for X_t should satisfy the following three desirable criteria:

(1) **Informativeness**: X'_t is (ρ, k)-informative if $\exists w^k_{t-\tau} \subset w_t, k \ll \alpha, \rho > 0 : \|X'_t(w^k_{t-\tau}) - X_t(w^k_{t-\tau})\| > \rho$ and $X_t \backslash X_t(w^k_{t-\tau}) \approx X'_t \backslash X'_t(w^k_{t-\tau})$, where $X_t \backslash X_t(w^k_{t-\tau})$ is the corresponding time series in X_t after excluding the segment $X_t(w^k_{t-\tau})$.
(2) **Closeness**: X'_t is ϵ-close to X_t if $\|X_t - X'_t\| < \epsilon$ for $\epsilon > 0$.
(3) **Realisticness**: X'_t is δ-realistic if $P(X'_t) \ge \delta$, where $P(\cdot)$ is the probability that X'_t is drawn from the same distribution as any randomly chosen segment from the time series $\mathcal{Z}$.

Figure 2 illustrates examples of counterfactual instances evaluated using the 3 criteria above when applied to a temperature block maxima forecasting model. In this hypothetical example, assume the forecast model predicts an extreme temperature value, say, 105°F (depicted as a blue dot), for the forecast window between January and September in 2023. Suppose a counterfactual instance with the counterfactual target of around 80°F (shown as a red dot) is to be constructed. The left panel shows a comparison between an informative and uninformative counterfactual instance. Even though both counterfactual instances yield the same counterfactual target, the informative counterfactual, shown as a red line, is nearly identical to the original time series, shown as a blue line, except for the larger deviation in the interval highlighted in green shadow. The counterfactual is informative as it pinpoints the segment within the predictor window whose anomalous values lead to the unusually extreme block maxima forecasted by the model. In contrast, the uninformative counterfactual, shown by the green line, exhibits deviations from the original time series throughout the entire time period, offering little information that could explain the forecasted extreme block maxima. The middle panel of Figure 2 shows the distinction between counterfactual instances with high (red line) and low (green line) closeness in terms of their proximity to the original time series. Finally, the right panel of Figure 2 illustrates the difference between realistic (red line) and unrealistic (green line) counterfactual instances. Specifically, the temperature profile of the unrealistic counterfactual is counter-intuitive as it exhibits higher temperatures in winter than in summer.

Unfortunately, balancing the trade-off among the criteria can be tricky. For example, the following theorem demonstrates the impossibility of satisfying both informative and closeness criteria when $\rho > \epsilon$.

Theorem 1. *Let X'_t be a (ρ, k)-informative counterfactual predictor of X_t. If $\rho > \epsilon$, then X'_t must not be ϵ-close.*

Proof. Since X'_t is (ρ, k)-informative, using the additive property of vector norm, we have:

$$\|X'_t - X_t\| = \|X'_t(w^k_{t-\tau}) - X_t(w^k_{t-\tau})\| + \|X'_t \backslash X'_t(w^k_{t-\tau}) - X_t \backslash X_t(w^k_{t-\tau})\|$$
$$\geq \rho + \|X'_t \backslash X'_t(w^k_{t-\tau}) - X_t \backslash X_t(w^k_{t-\tau})\|$$

Furthermore, given that $\rho > \epsilon$, this implies $\|X'_t - X_t\| > \epsilon$, which means X'_t must not be ϵ-close. □

Similarly, if X'_t is ϵ-close and $\rho > \epsilon$, then $\|X'_t(w^k_{t-\tau}) - X_t(w^k_{t-\tau})\| \leq \epsilon$, which means finding a (ρ, k)-informative counterfactual predictor would be impossible. An analogous impossibility theorem for realisticness is more challenging as it depends on the probability model of the time series. The difficulty of balancing the 3 criteria can be illustrated with an example. Let X'_t be the counterfactual predictor obtained via the nearest neighbor approach [8]. Since X'_t is an existing time series, $P(X'_t)$ must be large. However, it may not be close unless X'_t is in the ϵ-neighborhood of X_t. Even if it is ϵ-close, X'_t may not be (ρ, k)-informative if $\rho > \epsilon$, as shown above.

3.2 Generalized Extreme Value Distribution

Consider a time series $\mathcal{Z}$ of length T that is partitioned into m sequences, each of length n (*i.e.*, $nm = T$). For each sequence, let Y_n be its block maxim. The generalized extreme value (GEV) distribution [7] is often used to describe the probability distribution governing the block maxima values. The distribution is characterized by its shape (ξ), location (μ), and scalar (σ) parameters, with the following cumulative distribution function (CDF):

$$P(Y_n \leq y) = \exp\left\{-\left[1 + \xi\left(\frac{y-\mu}{\sigma}\right)\right]^{-\frac{1}{\xi}}\right\}, \tag{1}$$

subject to the constraint:

$$\forall y : 1 + \xi\frac{y-\mu}{\sigma} > 0 \tag{2}$$

The pth quantile of the distribution, y_p, can be calculated as follows:

$$y_p = \mu + \frac{\sigma}{\xi}\left[(-\log p)^{-\xi} - 1\right]. \tag{3}$$

Given the shape, location, and scale parameters of the GEV distribution[1], the preceding equation is used to determine the threshold for extreme block maxima and the counterfactual target for our proposed framework by setting the appropriate quantile values, p.

3.3 Deep Block Maxima Forecasting Model

Recent years have witnessed a growing number of research focusing on the use of extreme value theory to enhance the forecasting of extreme events in time series [12, 24, 34]. For example, Wilson et al. [34] introduced the DeepGPD framework with Generalized Pareto (GP) distribution to forecast excess values over some prespecified threshold while Galib et al. [12] and Nishino et al. [24] utilized the Generalized Extreme Value (GEV) distribution for block maxima forecasting problems. In this work, we choose DeepExtrema [12] as our black box model due to its superior performance in block maxima forecasting compared to other baselines. Nevertheless, our framework is model-agnostic, and thus, applicable to other block maxima forecasting models such as [24].

[1]These parameters will be estimated by the block maxima forecast model.

DeepExtrema employs a combination of LSTM with fully connected layers to estimate the GEV parameters $\{\mu(X_t), \sigma(X_t), \xi(X_t)\}$ of an input time series X_t in a way that preserves the inequality constraints given in (2). It then utilizes a fully connected network (FCN) layer to generate the forecast of block maxima, $\hat{y}_t$, from the estimated GEV parameters. The framework is trained end-to-end to simultaneously learn both the GEV parameters and its block maxima forecast by minimizing the following loss function [12]:

$$L = \lambda_1 \hat{L}_{GEV} + (1 - \lambda_1)\sum_{i=1}^{N}(y_t^{(i)} - \hat{y}_t^{(i)})^2, \tag{4}$$

where $\hat{L}_{GEV}$ is a regularized negative log-likelihood of the GEV distribution while the second term is the least-square loss between the forecasted and actual block maxima of the training instances.

Note that the aim of our study is not to assess the predictive performance of DeepExtrema or other similar forecast models. Instead, it focuses on providing counterfactual explanations to elucidate the forecasts generated by the model, regardless of their accuracy.

3.4 Counterfactual Explanation

Existing studies on counterfactual explanation have mostly centered around binary classification problems [8, 30]. Specifically, let X_t denote the predictor time series and c denote the predicted class label of the block maxima in the forecast time series Y_t, according to a binary classifier f, *i.e.*, $f(X_t) = c$. The primary objective of counterfactual explanation is to find X'_t, a modified counterpart of X_t that will lead to the alternative class label c' by the model f, *i.e.*, $f(X'_t) = c' \neq c$. This objective is typically achieved by minimizing the following loss function [30]:

$$L(X_t, X'_t, c', \lambda) = \lambda\left(f(X'_t) - c'\right)^2 + d(X_t, X'_t), \tag{5}$$

where λ is a tuning parameter that balances the components of the loss function. The first term quantifies the difference between the model's prediction for the modified input X'_t and the counterfactual target class c'. The second term measures the dissimilarity between the input X_t and its counterfactual X'_t.

In this work, we will adapt the approach to generate a counterfactual explanation for extreme block maxima, a continuous value instead of a class label, forecasted by models such as DeepExtrema [12].

4 PROPOSED FRAMEWORK

To generate the counterfactual explanation for block maxima forecasting, our *DiffusionCF* framework performs the following steps. First, a continuous-valued counterfactual target $\hat{y}'_t$ associated with the forecasted block maxima $\hat{y}_t$ is constructed, as described in Section 4.1. Next, the corresponding counterfactual predictors X'_t for the target $\hat{y}'_t$ is learned using the approach described in Section 4.2. Finally, in Section 4.3, we demonstrate how to use X'_t to explain the extreme block maxima forecast generated by the model f.

4.1 Constructing Counterfactual Target $\hat{y}'_t$

Our goal is to generate a counterfactual target that satisfies the following two conditions. First, given an input X_t, if the forecasted block maxima $\hat{y}_t$ is an extreme value, then the counterfactual target must be non-extreme. This requires setting a threshold $\hat{y}'_U$ that

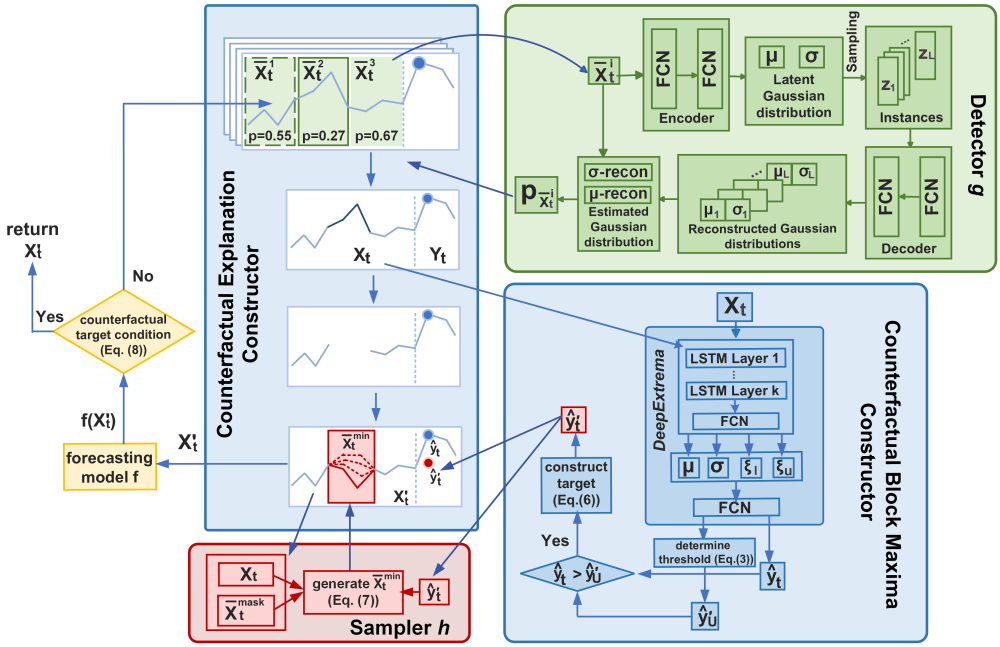

Figure 3: A schematic illustration of the proposed *DiffusionCF* framework.

determines whether the forecasted block maxima is extreme or non-extreme. Second, as the distribution of block maxima values is governed by the GEV distribution, the sampled counterfactual target should be drawn from the same distribution. Adhering to the GEV distribution enables us to construct the continuous-valued counterfactual target in a more principled fashion.

To achieve both conditions, we utilize the GEV parameters generated by the DeepExtrema model[2]. Specifically, given an input X_t, DeepExtrema will generate both the block maxima forecast $\hat{y}_t$ along with parameters of its associated GEV distribution, $\mu(X_t)$, $\xi(X_t)$, and $\sigma(X_t)$. Let $p \in (0, 1)$ be a hyperparameter corresponding to the desired quantile for defining an extreme block maxima. The extreme value threshold is computed by setting $\hat{y}'_U = z_p$ using the quantile formula for GEV distribution given by (3). If the predicted block maxima $\hat{y}_t \geq \hat{y}'_U$, then $\hat{y}_t$ is considered an extreme value.

If $\hat{y}_t$ is an extreme block maxima, then a counterfactual target will be constructed using the forecasted GEV distribution by setting a quantile $p' < p$ to ensure the counterfactual target is below the extreme threshold. Analogous to (3), the counterfactual target is computed as follows:

$$\hat{y}'_t = \mu(X_t) + \frac{\sigma(X_t)}{\xi(X_t)}\left[(-\log p')^{-\xi(X_t)} - 1\right], \tag{6}$$

where $\mu(X_t)$, $\sigma(X_t)$, and $\xi(X_t)$ are the learned parameters produced by the DeepExtrema model.

[2]For other black box models, we can fit a GEV distribution to all the block maxima values first to learn their GEV parameters and use them to generate the threshold $\hat{y}'_U$.

While our framework can be adapted to explain non-extreme block maxima forecasts, our current approach is specifically designed to generate counterfactual instances for extreme block maxima in this work. This is due to their significant practical implications and the valuable insights they offer for prevention and mitigation efforts.

4.2 Constructing Counterfactual Predictor X'_t

After identifying the counterfactual target $\hat{y}'_t$, the next step is to construct its corresponding counterfactual predictor, X'_t, such that $f(X'_t) \approx \hat{y}'_t$ while ensuring that X'_t is informative, realistic, and close to the original time series X_t. Unlike previous approaches (as examples in Section 5.2) that often resorted to searching or random (gradient-based) perturbations of the input data to produce the counterfactual predictors, our proposed *DiffusionCF* framework is designed to balance the tradeoff between informativeness, closeness, and realisticness of the counterfactual explanation. There are 3 main components in *DiffusionCF*, as shown in Figure 3.

4.2.1 Constructor for the Counterfactual Predictor. The key assumption guiding our approach is that when the black box model predicts an exceptionally extreme block maxima value, the time series segment(s) within X_t most likely contributing to the extreme forecast are those exhibiting a high level of anomaly. Our confidence in this assumption stems from the fact that the black box model derives its prediction solely from X_t. If the segments in X_t align with typical observations in the time series, it would be improbable for the forecast model to produce an extreme value output. While there may exist other scenarios that could lead to the forecasted extreme

block maxima, our assumption provides a highly plausible and computationally feasible way to pinpoint the explanatory factor behind the model's forecast.

Based on this assumption, our constructor for the counterfactual predictor needs to identify the most anomaly segment(s) in the time series and replace them with more typical patterns, while leaving the rest of the time series intact. This strategy enables our approach to create more informative yet realistic counterfactual instances. The counterfactual predictors are constructed as follows:

(1) **Extraction of Time Series Segments:** Given a time series X_t of length $\alpha+1$ for the predictor window $[t-\alpha, t]$, our algorithm first extracts all the time series segments $\left\{\overline{X}_t^i\right\}_{i=1}^{\alpha-d+2}$ within X_t by using a sliding window of fixed length, d.

(2) **Detection of Anomalous Segments:** For each extracted segment, $\overline{X}_t^i$, the algorithm computes the probability $p_{\overline{X}_t^i}$ that conforms to the underlying distribution of the time series data. The probability is estimated using a detector function g (to be described in Section 4.2.2), where $p_{\overline{X}_t^i} = g(\overline{X}_t^i)$.

(3) **Counterfactual Generation:** Starting from the most anomalous segment, $\overline{X}_t^{mask} = \arg\min_{s\in\{\overline{X}_t^i\}} p_s$, the algorithm would remove this segment from X_t and replaces it with a more "typical" segment. This is achieved by using a sampler h (see Section 4.2.3) to generate a set of m candidate replacements, denoted as $C_t^m = h(X_t, \overline{X}_t^{mask})$, where each candidate $\overline{X}_t^s \in C_t^m$ has a higher probability to be drawn from the time series than $\overline{X}_t^{mask}$, *i.e.*, $p_{\overline{X}_t^s} > p_{\overline{X}_t^{mask}}$. The best candidate $\overline{X}_t^{\min}$ is then selected based on its distance to the counterfactual target $\hat{y}'_t$:

$$\overline{X}_t^{\min} = \arg\min_{\overline{X}_t^s \in C_t^m} \left\| f\left((X_t \setminus \overline{X}_t^{mask}) \oplus \overline{X}_t^s\right) - \hat{y}'_t \right\|_1 \tag{7}$$

Here $(X_t \setminus \overline{X}_t^{mask}) \oplus \overline{X}_t^s$ denotes the resulting times series after replacing the anomalous segment $\overline{X}_t^{mask}$ with the candidate $\overline{X}_t^s$. The counterfactual predictor X'_t is obtained by replacing the anomalous segment with the best candidate, *i.e.*, $X'_t = (X_t \setminus \overline{X}_t^{mask}) \oplus \overline{X}_t^{\min}$. As the forecast for the counterfactual predictor X'_t generated from the anomalous segment, $f(X'_t)$ may not always be close to the counterfactual target, $\hat{y}'_t$, this step is repeated using the next K most anomalous segment(s) until the following conditions are met:

Counterfactual Target Condition:

$$f(X'_t) < \hat{y}'_U \text{ and } f(X'_t) \in [\hat{y}'_t - \epsilon, \hat{y}'_t + \epsilon] \tag{8}$$

For efficiency reasons, this process of removal and replacement of the anomalous segment is repeated for at most K times. If no viable counterfactual predictor X'_t is found, the algorithm will return the best X'_t it has discovered.

4.2.2 Detector g. Our *DiffusionCF* framework uses a variational auto-encoder (VAE) for anomaly detection [2] as its detector g. The VAE for anomaly detection is trained to learn the underlying distribution of all time series segments $\{\overline{X}_t^i\}_{i=1}^{\alpha-d+2}$ of length d extracted from the predictor time series X_t of the training data. It comprises two main components—an encoder V_{encoder} and a decoder V_{decoder}. The encoder V_{encoder} takes each $\overline{X}_t^i$ as input and maps it to a latent Gaussian distribution with mean vector μ and isotropic covariance $\sigma^2\mathbf{I}$. The Gaussian distribution is used to draw L samples from the h-dimensional latent space, denoted as $\{I_i \in R^h | i = 1, 2, ..., L\}$. The decoder V_{decoder} would attempt to reconstruct the original time series segment, $\overline{X}_t^i$, from each sampled latent instance. The VAE for anomaly detection is trained to minimize the average reconstruction error of the time series segments.

During the detection step, for each input $\overline{X}_t^i$, the VAE would compute the parameters of its latent distribution, which are used to calculate the probability $p_{\overline{X}_t^i}$. The probability determines whether $\overline{X}_t^i$ is anomalous. The higher the probability, the more likely $\overline{X}_t^i$ belongs to the same distribution as the majority of the time series segments, so the less anomalous it is. Details of the *Detector* module are shown in Figure 3.

4.2.3 Sampler h. Our framework uses a conditional diffusion model as its sampler h to construct a candidate replacement, $\overline{X}_t^s$, for an anomalous segment, $\overline{X}_t^{mask}$. Specifically, it will first mask the anomalous segment from the input time series X_t and provide the masked input to CSDI [29], a conditional diffusion model that is adept at imputing missing segments of a time series (see ***Appendix*** A.1 for details). We chose CSDI because it can leverage the unmasked portion of the time series (*i.e.*, $X_t \setminus \overline{X}_t^{mask}$) to create a new candidate $\overline{X}_t^s$ for imputing the masked segment. This allows *DiffusionCF* to produce imputed segments that are consistent with the rest of the time series, leading to more realistic counterfactual predictors. Nevertheless, our framework is flexible and can incorporate other samplers such as DDPM [14] and VAE [17].

4.3 Using Counterfactual Predictor to Explain Block Maxima Forecast

The counterfactual predictor, X'_t, generated by *DiffusionCF* can be used to elucidate the specific segment within the input time series X_t that largely contributes to the forecasted extreme block maxima. Let $\Delta X = X_t - X'_t = (\Delta z_{t-\alpha}, \Delta z_{t-\alpha+1}, ..., \Delta z_t)$ be a vector of absolute difference between the counterfactual and original predictor, *i.e.*, $\Delta z_i = |X_{t,i} - X'_{t,i}|$. The time steps within the predictor window, $[t-\alpha, t]$ can be sorted in decreasing magnitude of their Δz_i. If $|\Delta z_i|$ exceeds some threshold, then the segment can be considered a notable contributor to the extreme block maxima forecast by the black box model f.

5 PERFORMANCE EVALUATION

5.1 Datasets

We use the following datasets for our experiments: (1) **Global Surface Summary of the Day (GSOD)**, a dataset that contains daily observations of precipitation and temperature from 79 weather stations in the Mobile/Pensacola area in the southwestern United States. The dataset spans a time period from August 1, 1929, to

Table 1: Summary of datasets, where $|X_t|$ and $|Y_t|$ denote the length of the predictor and forecast windows, respectively.

Dataset	#Training samples	#Validation samples	# Testing samples	$\lvert X_t\rvert$	$\lvert Y_t\rvert$
GSOD	14798	548	511	24	6
S&P 500	7022	251	251	25	5
Dodgers sensor	846	248	241	40	8

November 22, 2023. (2) **S&P 500**, a dataset comprises of daily closing prices of the S&P-500 index from January 13, 1994 to January 12, 2024. (3) **Dodgers loop sensor**, a dataset containing traffic volume data from April 10, 2005, to October 2, 2005, at the Glendale ramp of the 101 North freeway in Los Angeles, near the Dodgers stadium. Summary statistics of the datasets are given in Table 1 while their pre-processing steps are described in ***Appendix A.3***.

5.2 Baseline Methods

We compare the performance of *DiffusionCF* against the following baseline methods:

- **BaseNN**, a baseline used in [32] to identify the nearest-unlike neighbor X'_t from the training set, whose true block maxima value aligns with the counterfactual target.
- ω**-CF** [30] learns X'_t by minimizing the loss between $f(X'_t)$ and $\hat{y}'_t$, as well as the distance between X_t and X'_t.
- **Native guide (NG-CF)** [8] constructs X'_t by identifying a nearest-unlike neighbor to X_t and modifying it to produce a model forecast close to the counterfactual target.
- **ForecastCF** [32] employs a mask objective function to learn X'_t, aiming at minimizing the loss between $f(X'_t)$ and $\hat{y}'_t$.
- **SPARCE** [18] generates a counterfactual explanation for time series by using a generative adversarial network (GAN).

As some baseline methods were developed for classification tasks, they were adapted to generate counterfactual instances for block maxima forecasts. Details are given in ***Appendix A.4***.

5.3 Evaluation Metrics

As noted in Section 3.1, a good counterfactual predictor should be realistic, informative, and close to the original time series. Let $X_t = z_{t-\alpha}z_{t-\alpha+1}...z_t$ be the original predictors and $X'_t = z'_{t-\alpha}z'_{t-\alpha+1}...z'_t$ be the counterfactual predictors. We employ the following metrics to assess the performance of the various methods:

- **Informativeness**. As noted in Section 3.1, an informative counterfactual should modify only a small segment of X_t, keeping the rest of the predictor time series intact. We use a combination of *sparsity* and *consecutiveness* metrics to determine whether a counterfactual predictor X'_t is informative.

$$\text{Sparsity}(X'_t) = \frac{1}{\alpha+1}\,\text{count}\{i : z_{t-i} \neq z'_{t-i}, i = 0, \cdots, \alpha\} \quad (9)$$

A lower value of sparsity means fewer modifications to X_t. Next, we construct the following sequence of binary values:

$$\Delta(z_{t-i}, z'_{t-i}) = \begin{cases} 1, \text{ if } |z_{t-i} - z'_{t-i}| > \rho, \\ 0, \text{ otherwise,} \end{cases} \quad (10)$$

where $\rho > 0$ is a threshold. Let $L_{\max}$ represent the maximum length of consecutive 1's in the binary sequence. The consecutiveness metric is defined as

$$\text{Consecutiveness}(X'_t) = \frac{L_{\max}}{\sum_{i=1}^{\alpha}\Delta(z_{t-i}, z'_{t-i})}. \quad (11)$$

A higher consecutiveness implies the notable difference between X_t and X'_t is mostly concentrated in a local segment of the time series. Thus, an informative counterfactual should have low sparsity but high consecutiveness values.

- **Closeness**. The *proximity* metric below is used to determine the extent to which X_t is close to X'_t:

$$\text{Proximity}(X_t, X'_t) = \frac{1}{\alpha+1}\sum_{i=0}^{\alpha}|z_{t-i} - z'_{t-i}|. \quad (12)$$

The lower the proximity, the closer the counterfactual predictor is to the original predictor.

- **Realisticness**. The *negative likelihood* function of X'_t is used to determine whether a counterfactual instance is realistic, $\text{NLL}(X'_t) = -\log p_{X'_t}$. Here, $p_{X'_t}$ represents the probability assigned to X'_t by detector g. The lower the NLL, the more realistic the counterfactual predictor.

Finally, we use the *precision* metric to ascertain how well X'_t will ensure that $f(X'_t)$ is classified as a non-extreme block maxima.

5.4 Experimental Results

5.4.1 Performance Comparison. Experiments results are summarized in Table 2 based on the setup discussed in ***Appendix A.5***. Figure 4 provides an illustrated example of the counterfactual instances found by the different methods. In general, *DiffusionCF* generates the most informative counterfactual instances in all 4 datasets, achieving the best sparsity and consecutiveness scores. It also appears among the top 2 approaches with the best proximity and precision in at least 3 of the 4 datasets. These results suggest that *DiffusionCF* generally demonstrate superior performance in explaining the extreme block maxima forecast generated by the black box model compared to other baselines. Though its *NLL* score is slightly worse than BaseNN and NG-CF, this is not surprising as the latter two approaches create their counterfactuals by sampling from the training instances.

Specifically, (1) For BaseNN, its superior performance in terms of *NLL* is attributable to its strategy of using training instances as X'_t. However, since BaseNN does not optimize for closeness between X_t and X'_t, it performs poorly in terms of *proximity* and *sparsity* metrics, as illustrated in Figure 4(left). BaseNN also has lower *precision* because the forecasted counterfactual block maxima $f(X'_t)$ may not be consistent with the desired counterfactual target since the nearest-unlike neighbor is chosen based on the true block maxima value instead of its forecasted block maxima. (2) For ω-CF and ForecastCF, their precision is perfect in all 4 datasets, which is not surprising as they both employ optimization-based approaches to ensure $f(X'_t)$ is close to the counterfactual target, $\hat{y}'_t$. Nevertheless, ω-CF has a better *proximity* score compared to ForecastCF. This is because the loss function used by ω-CF includes the distance between X_t and X'_t to encourage smaller modifications to the input. In contrast, ForecastCF does not consider such a factor, allowing it to make larger modifications. Hence, its *proximity* score is higher.

Table 2: Evaluation of performance on 4 real-world datasets conducted for scenarios where the forecast $\hat{y}_t$ is extreme, and the counterfactual target $\hat{y}'_t$ is non-extreme. Within the comparative results, red entries indicate the top-performing result, while blue entries signify the second-best performance for each metric.

Method	GSOD-precipitation					GSOD-temperature				
	Spars.↓	*Consecu.*↑	*Proxi.*↓	*NLL*↓	*Prec.*↑	*Spars.*↓	*Consecu.*↑	*Proxi.*↓	*NLL*↓	*Prec.*↑
BaseNN	1.00 (±0.00)	0.86 (±0.18)	1.12 (±0.19)	1.18 (±0.15)	0.27	1.00 (±0.00)	0.86 (±0.17)	1.13 (±0.21)	**1.08** (±0.36)	0.48
ω-CF	1.00 (±0.00)	0.91 (±0.13)	**0.17** (±0.07)	1.07 (±0.18)	**1.00**	1.00 (±0.00)	0.65 (±0.18)	**0.18** (±0.08)	1.12 (±0.07)	**1.00**
NG-CF	**0.83** (±0.26)	0.67 (±0.26)	0.57 (±0.24)	**0.99** (±0.17)	0.38	**0.94** (±0.17)	0.65 (±0.25)	0.56 (±0.22)	**1.09** (±0.08)	0.56
ForecastCF	1.00 (±0.00)	**0.92** (±0.18)	0.45 (±0.17)	1.13 (±0.37)	**1.00**	1.00 (±0.00)	**0.90** (±0.15)	0.35 (±0.13)	1.21 (±0.09)	**1.00**
SPARCE	1.00 (±0.00)	0.72 (±0.22)	1.10 (±0.21)	1.13 (±0.17)	0.68	1.00 (±0.00)	0.74 (±0.21)	1.10 (±0.22)	1.18 (±0.07)	**0.87**
DiffusionCF	**0.25** (±0.00)	**0.93** (±0.13)	**0.16** (±0.07)	**1.06** (±0.22)	**0.98**	**0.12** (±0.00)	**0.96** (±0.13)	**0.08** (±0.04)	1.20 (±0.08)	**1.00**

Method	S&P 500					Dodgers loop sensor				
	Spars.↓	*Consecu.*↑	*Proxi.*↓	*NLL*↓	*Prec.*↑	*Spars.*↓	*Consecu.*↑	*Proxi.*↓	*NLL*↓	*Prec.*↑
BaseNN	1.00 (±0.00)	**0.88** (±0.17)	1.43 (±0.40)	**0.84** (±0.49)	0.98	1.00 (±0.01)	0.75 (±0.22)	1.12 (±0.39)	**0.36** (±0.17)	**0.66**
ω-CF	1.00 (±0.00)	0.46 (±0.17)	**0.59** (±0.11)	1.13 (±0.26)	**1.00**	1.00 (±0.00)	**0.76** (±0.25)	**0.02** (±0.01)	0.45 (±0.11)	**1.00**
NG-CF	1.00 (±0.00)	0.82 (±0.21)	0.92 (±0.23)	**0.96** (±0.29)	**0.99**	0.99 (±0.03)	0.50 (±0.17)	0.52 (±0.18)	0.56 (±0.24)	0.61
ForecastCF	1.00 (±0.00)	0.85 (±0.19)	0.81 (±0.16)	1.10 (±0.23)	**1.00**	1.00 (±0.00)	0.56 (±0.24)	0.09 (±0.04)	0.44 (±0.12)	**1.00**
SPARCE	**0.98** (±0.00)	0.75 (±0.22)	1.43 (±0.38)	1.71 (±0.20)	**1.00**	**0.97** (±0.00)	0.66 (±0.21)	1.14 (±0.35)	0.46 (±0.12)	**1.00**
DiffusionCF	**0.40** (±0.00)	**0.96** (±0.10)	**0.51** (±0.16)	1.68 (±0.19)	0.97	**0.12** (±0.00)	**0.81** (±0.18)	**0.05** (±0.03)	**0.43** (±0.13)	**1.00**

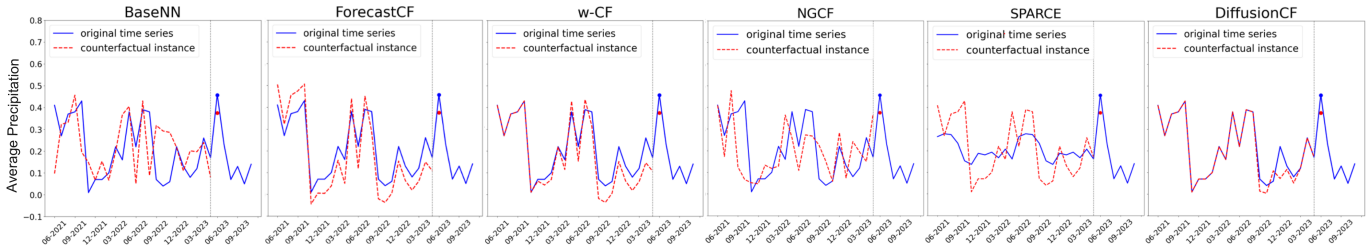

Figure 4: A comparative study of X'_t, generated by *BaseNN*, *ForecastCF*, *ω-CF*, *NGCF*, *SPARCE*, and *DiffusionCF*, when applied to precipitation forecasting between 2021-06 and 2023-09 for a weather station in Pensacola, Florida. The blue dot represents the forecasted block maxima, $\hat{y}_t$, while the red dot represents the counterfactual target, $\hat{y}'_t$.

This difference in *proximity* between ω-CF and ForecastCF can be clearly seen in the middle two plots of Figure 4. Furthermore, in terms of informativeness, both approaches perform poorly in terms of their *sparsity* metric, though ForecastCF demonstrates a relatively higher *consecutiveness* score compared to ω-CF. (3) For SPARCE, which utilizes GAN for optimization, it outperforms ω-CF and ForecastCF in terms of *sparsity* on the S&P 500 and Dodgers loop sensor datasets. However, its sparsity is similar to other baselines and worse than NG-CF on the GSOD datasets. SPARCE also struggles in terms of informativeness, closeness, and realisticness metrics. This underscores the difficulty of using GAN to optimize multiple objectives simultaneously. ForecastCF, ω-CF, and SPARCE all fall short compared to BaseNN in terms of *NLL*, suggesting the counterfactual predictors produced by these optimization-based methods are less realistic. (4) For NG-CF, which is a combination of neighbor searching and perturbation-based methods, can balance the trade-off in terms of *proximity*, *precision*, and *NLL* but falls short in terms of producing informative explanation due to the generally low *consecutiveness* scores. Finally, (5) for *DiffusionCF*, it excels in terms of *sparsity*, *consecutiveness*, *proximity*, and *precision*. This advantage is crucial as it renders the counterfactual explanation more informative, as shown in Figure 4(right).

Table 3: Ablation study results on GSOD-precipitation data.

	Spars.↓	*Consecu.*↑	*Proxi.*↓	*NLL*↓	*Prec.*↑
CSDI+3	0.12 (±0.00)	0.97 (±0.13)	0.06 (±0.06)	1.14 (±0.21)	0.87
CSDI+6	0.25 (±0.00)	0.93 (±0.13)	0.16 (±0.07)	1.06 (±0.22)	0.98
CSDI+9	0.38 (±0.00)	0.90 (±0.17)	0.29 (±0.13)	1.11 (±0.28)	1.00
CSDI+12	0.50 (±0.00)	0.91 (±0.16)	0.36 (±0.10)	1.08 (±0.24)	1.00
VAE+3	0.12 (±0.00)	0.98 (±0.15)	0.02 (±0.01)	1.13 (±0.19)	0.36
VAE+6	0.25 (±0.00)	0.90 (±0.17)	0.07 (±0.03)	1.08 (±0.21)	0.42
VAE+9	0.38 (±0.00)	0.84 (±0.17)	0.13 (±0.04)	1.04 (±0.21)	0.56
VAE+12	0.50 (±0.00)	0.87 (±0.17)	0.25 (±0.05)	1.02 (±0.16)	0.49

5.4.2 Ablation study of DiffusionCF. We conduct an ablation study for *DiffusionCF* by varying the window size d for segment extraction, as described in Section 4.2.2 and employing VAE as our alternative sampling technique. A detailed breakdown of the results is given in Table 3. Our key conclusions are as follows:

- Increasing the window size d from 3 to 12 generally degrades the performance of *DiffusionCF* in terms of *sparsity*, *proximity*, and *consecutiveness*, while enhancing its *realisticness* and *precision*. This is because a larger d increases the number of time steps available for adding perturbation, allowing greater deviation from the original time series, thus reducing sparsity and increasing their dissimilarity. Larger window size also reduces the percentage of consecutive time steps that were perturbed, as shown in Table 3, with the exception of $d = 12$. Nevertheless, it also gives more flexibility for *DiffusionCF* to construct counterfactual instances that are close to the desired target, thus enhancing its precision. Interestingly, the NLL values for VAE exhibit a decreasing trend with larger d, warranting future investigation.
- In terms of *informativeness* and *precision*, *DiffusionCF* based on CSDI outperforms the one based on VAE. However, the latter is superior in *proximity* and *realisticness*. As shown in Table 3, the CSDI-based method slightly surpasses the VAE-based method in *consecutiveness*. The CSDI-based method's advantage in *precision* is evident as the diffusion model used for imputation generates samples that, while deviating more from the original time series compared to the VAE-based method, still remain realistic, thereby contributing to more accurate counterfactual targets. However, this increased deviation results in worse performance in *proximity* and *NLL* compared to the VAE-based method.

5.4.3 Case study of DiffusionCF. Our case study on the *Dodgers loop sensor* dataset focuses on the specific example when there are two consecutive game days at the Dodgers stadium, on May 31 and June 1, 2005. Figure 5 depicts the average 3-hourly traffic flow at a ramp near the stadium from Friday, May 27 to Wednesday, June 1,

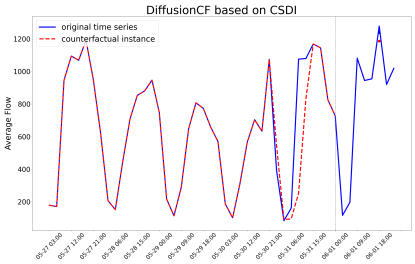

Figure 5: Observed average traffic flow (May 27-Jun 01) vs. counterfactual instance found by CSDI-based *DiffusionCF*.

2005. In this example, the model utilizes traffic flow data from the preceding five days to forecast the maximum traffic flow for June 1st. The resulting block maxima value, approximately 1300, is deemed extreme, surpassing 80% of the average 3-hourly traffic flows within the dataset. This extreme value is attributed to a baseball game held at the stadium on that day. The figure also depicts a counterfactual target, with the block maxima value set around 1150, representing the typical traffic flow if the baseball match had not occurred. Based on the counterfactual target, a counterfactual instance is generated by the CSDI-based *DiffusionCF* model.

The counterfactual instance generated by *DiffusionCF* adjusts the traffic pattern for Tuesday, May 31st, depicting a decrease in traffic flow for that day, indicative of the absence of a baseball game at the stadium. The absence of a game on Tuesday suggests the likelihood of no game the following day. In essence, the increase in traffic flow on Tuesday attributed to the baseball game could elucidate the extreme block maxima forecasted for Wednesday. The result shown in Figure 5 thus underscores *DiffusionCF*'s capability in identifying anomalies (*i.e.*, pattern of elevated traffic flow on game days) and modify them towards a normative state (*i.e.*, pattern of reduced traffic flow) as its counterfactual instance.

6 CONCLUSIONS AND FUTURE WORK

This paper introduces the novel problem of counterfactual explanation for block maxima forecasting models in time series. We propose a methodology for creating counterfactual block maxima and introduce the *DiffusionCF* framework to balance the trade-off between generating counterfactual explanations that are informative, close to the original time series, and realistic. Experimental results show that *DiffusionCF* generates better counterfactual instances compared to other baselines. Nevertheless, the current framework has two potential limitations. First, it considers only univariate time series. For future endeavors, we plan to extend our methodology to the multivariate case. Second, *DiffusionCF* is biased towards constructing its counterfactual predictor by modifying only one of the anomalous segments in the time series. We plan to investigate approaches that could modify multiple segments instead.

7 ACKNOWLEDGMENT

This research is supported by the U.S. National Science Foundation under grant IIS-2006633. Any use of trade, firm, or product names is for descriptive purposes only and does not imply endorsement by the U.S. Government.

REFERENCES

[1] Reduan Achtibat, Maximilian Dreyer, Ilona Eisenbraun, Sebastian Bosse, Thomas Wiegand, Wojciech Samek, and Sebastian Lapuschkin. 2023. From attribution maps to human-understandable explanations through concept relevance propagation. *Nature Machine Intelligence* 5, 9 (2023), 1006–1019.

[2] Jinwon An and Sungzoon Cho. 2015. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE* 2, 1 (2015), 1–18.

[3] Emre Ates, Burak Aksar, Vitus J Leung, and Ayse K Coskun. 2021. Counterfactual explanations for multivariate time series. In *2021 International Conference on Applied Artificial Intelligence (ICAPAI)*. IEEE, 1–8.

[4] Lorenzo Betti, Carlo Abrate, Francesco Bonchi, and Andreas Kaltenbrunner. 2023. Relevance-based infilling for natural language counterfactuals. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 88–98.

[5] Dieter Brughmans, Pieter Leyman, and David Martens. 2023. Nice: an algorithm for nearest instance counterfactual explanations. *Data Mining and Knowledge Discovery* (2023), 1–39.

[6] Ziheng Chen, Fabrizio Silvestri, Jia Wang, Yongfeng Zhang, and Gabriele Tolomei. 2023. The dark side of explanations: poisoning recommender systems with counterfactual examples. In *Proceedings of the 46th International ACM SIGIR conference on Research and Development in Information Retrieval*. 2426–2430.

[7] Stuart Coles, Joanna Bawa, Lesley Trenner, and Pat Dorazio. 2001. *An introduction to statistical modeling of extreme values*. Vol. 208. Springer.

[8] Eoin Delaney, Derek Greene, and Mark T Keane. 2021. Instance-based counterfactual explanations for time series classification. In *International Conference on Case-Based Reasoning*. Springer, 32–47.

[9] Bhat Dittakavi, Bharathi Callepalli, Aleti Vardhan, Sai Vikas Desai, and Vineeth N Balasubramanian. 2024. CARE: counterfactual-based algorithmic recourse for explainable pose correction. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 4902–4911.

[10] Gil Fidel, Ron Bitton, and Asaf Shabtai. 2020. When explainability meets adversarial learning: detecting adversarial examples using shap signatures. In *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[11] Asadullah Hill Galib, Andrew McDonald, Pang-Ning Tan, and Lifeng Luo. 2023. Self-recover: forecasting block maxima in time series from predictors with disparate temporal coverage using self-supervised learning. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI 2023)*.

[12] Asadullah Hill Galib, Andrew McDonald, Tyler Wilson, Lifeng Luo, and Pang-Ning Tan. 2022. DeepExtrema: a deep learning approach for forecasting block maxima in time series data. *In Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI* (2022).

[13] Riccardo Guidotti. 2022. Counterfactual explanations and how to find them: literature review and benchmarking. *Data Mining and Knowledge Discovery* (2022), 1–55.

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems* 33 (2020), 6840–6851.

[15] Guillaume Jeanneret, Loïc Simon, and Frédéric Jurie. 2022. Diffusion models for counterfactual explanations. In *Proceedings of the Asian Conference on Computer Vision*. 858–876.

[16] Guillaume Jeanneret, Loïc Simon, and Frédéric Jurie. 2023. Adversarial counterfactual visual explanations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16425–16435.

[17] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *ArXiv Preprint ArXiv:1312.6114* (2013).

[18] Jana Lang, Martin A Giese, Winfried Ilg, and Sebastian Otte. 2023. Generating sparse counterfactual explanations for multivariate time series. In *International Conference on Artificial Neural Networks*. Springer, 180–193.

[19] Nicholas J Leach, Antje Weisheimer, Myles R Allen, and Tim Palmer. 2021. Forecast-based attribution of a winter heatwave within the limit of predictability. *Proceedings of the National Academy of Sciences* 118, 49 (2021), e2112087118.

[20] Peiyu Li, Soukaïna Filali Boubrahimi, and Shah Muhammad Hamdi. 2022. Motif-guided time series counterfactual explanations. In *International Conference on Pattern Recognition*. Springer, 203–215.

[21] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems* 30 (2017).

[22] Han Meng, Christian Wagner, and Isaac Triguero. 2023. Explaining time series classifiers through meaningful perturbation and optimisation. *Information Sciences* (2023), 119334.

[23] Christoph Molnar. 2020. *Interpretable machine learning*. Lulu.com.

[24] Kaneharu Nishino, Ken Ueno, and Ryusei Shingaki. 2022. Deep learning-based block maxima distribution predictor for extreme value prediction. In *8th SIGKDD International Workshop on Mining and Learning from Time Series – Deep Forecasting: Models, Interpretability, and Applications*.

[25] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 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*. 1135–1144.

[26] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*. 618–626.

[27] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. 2020. Score-based generative modeling through stochastic differential equations. *ArXiv Preprint ArXiv:2011.13456* (2020).

[28] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. 2019. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation* 23, 5 (2019), 828–841.

[29] Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. 2021. CSDI: conditional score-based diffusion models for probabilistic time series imputation. *Advances in Neural Information Processing Systems* 34 (2021), 24804–24816.

[30] Sandra Wachter, Brent Mittelstadt, and Chris Russell. 2017. Counterfactual explanations without opening the black box: automated decisions and the GDPR. *Harv. JL & Tech.* 31 (2017), 841.

[31] Xiangmeng Wang, Qian Li, Dianer Yu, Qing Li, and Guandong Xu. 2024. Counterfactual explanation for fairness in recommendation. *ACM Transactions on Information Systems* 42, 4 (2024), 1–30.

[32] Zhendong Wang, Ioanna Miliou, Isak Samsten, and Panagiotis Papapetrou. 2023. Counterfactual explanations for time series forecasting. *ArXiv Preprint ArXiv:2310.08137* (2023).

[33] James Wexler, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda Viégas, and Jimbo Wilson. 2019. The what-if tool: interactive probing of machine learning models. *IEEE Transactions on Visualization and Computer Graphics* 26, 1 (2019), 56–65.

[34] Tyler Wilson, Pang-Ning Tan, and Lifeng Luo. 2022. DeepGPD: a deep learning approach for modeling geospatio-temporal extreme events. *Proceedings of the AAAI Conference on Artificial Intelligence* 36, 4 (Jun. 2022), 4245–4253.

A APPENDICES

A.1 Details of CSDI model

CSDI [29] is a conditional score-based diffusion model for imputing missing values in time series by leveraging the available observed values. Specifically, CSDI is designed to estimate the true conditional data distribution $q(x^{ta}|x^{co})$ via the model distribution $p_\theta(x^{ta}|x^{co})$, where x^{ta} denotes the missing values to be imputed and x^{co} denotes the observed values.

Diffusion models such as DDPM [14] typically follow a two-step training process. First, during the *forward process*, the model starts from an initial input x_0^{ta} and iteratively perturbs the time series values to $x_1^{ta}, x_2^{ta}, \cdots$ by adding random noise until it converges to x_T^{ta}, where x_T^{ta} is of known, simple distribution. Next, during the *reverse process*, a neural network is trained to convert x_T^{ta} back to the original values x_0^{ta}. CSDI modifies the reverse process of DDPM with a conditional model defined as follows:

$$p_\theta(x_{0:T}^{ta}|x_0^{co}) = p(x_T^{ta}) \prod_{t=1}^{T} p_\theta(x_{t-1}^{ta}|x_t^{ta}, x_0^{co}), \quad x_T^{ta} \sim \mathcal{N}(0, \mathbf{I}),$$

where $p_\theta(x_{t-1}^{ta}|x_t^{ta}, x_0^{co}) = \mathcal{N}\big(x_{t-1}^{ta}; \mu_\theta(x_t^{ta}, t|x_0^{co}), \sigma_\theta(x_t^{ta}, t|x_0^{co})\mathbf{I}\big)$. Here, $\mu_\theta(x_t^{ta}, t|x_0^{co}) = \frac{1}{\alpha_t}\big(x_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\epsilon_\theta(x_t^{ta}, t|x_0^{co})\big)$, where β_t is a small positive constant representing the noise level, $\alpha_t = \prod_{i=1}^{t} \hat{\alpha}_i$, $\hat{\alpha}_t = 1 - \beta_t$, and $\epsilon_\theta : (\mathcal{X}^{ta} \times \mathbb{R}|\mathcal{X}^{co}) \to \mathcal{X}^{ta}$ is a conditional denoising function. Furthermore, $\sigma_\theta(x_t^{ta}, t|x_0^{co}) = \tilde{\beta}_t^{1/2}$, where

$$\tilde{\beta}_t = \begin{cases} \frac{1-\alpha_{t-1}}{1-\alpha_t}\beta_t & \text{if } t > 1, \\ \beta_1 & \text{if } t = 1. \end{cases} \tag{13}$$

Given x_0^{co} and x_0^{ta}, noisy samples for diffusion step t is given by: $x_t^{ta} = \sqrt{\alpha_t}x_0^{ta} + (1 - \alpha_t)\epsilon$, where ϵ is the added noise. During the reverse process, ϵ_θ is estimated by minimizing the loss function

$$\min_\theta \mathcal{L}(\theta) := \min_\theta E_{x_0 \sim q(x_0), \epsilon \sim \mathcal{N}(0,\mathbf{I}), t} \|(\epsilon - \epsilon_\theta(x_t^{ta}, t|x_0^{co}))\|_2^2, \tag{14}$$

where $q(x_0)$ is the data distribution of x_0. During training, the choice of imputation target is important, either by random strategy, historical strategy, mix strategy, or test pattern strategy. See [29] for details. Once trained, x_0^{ta} can be sampled from $p_\theta(x_{t-1}^{ta}|x_t^{ta}, x_0^{co})$.

A.2 The DiffusionCF algorithm

The pseudocode of the *DiffusionCF* algorithm is summarized in Algorithm 1. In this context, $X_t \setminus \overline{X}_t^{mask}$ denotes the remaining part of X_t after excluding $\overline{X}_t^{mask}$. The expression $(X_t \setminus \overline{X}_t^{mask}) \oplus \overline{X}_t^{s}$ represents the time series obtained after replacing $\overline{X}_t^{mask}$ in X_t with $\overline{X}_t^{s}$.

Algorithm 1 DiffusionCF

Input: Time series $Z = X_t \cup Y_t$, quantiles p and p', sliding window length d, sampling epochs L, searching epochs K
Output: Counterfactual instances $(X'_t, \hat{y}'_t)$

train, validation, test ← dataset.
f ← forecast(train, validation).
g ← detector(train, validation).
h ← sampler(train, validation).
probs = []; C_t^m = [].
for each $Z = X_t \cup Y_t$ in test **do**
  $\hat{y}_t, \hat{y}'_U$ = f(X_t).
  if $\hat{y}_t > \hat{y}'_U$ **then**
    $\hat{y}'_t \leftarrow \mu(X_t) + \frac{\sigma(X_t)}{\xi(X_t)}\left[(-\log p')^{-\xi(X_t)} - 1\right]$
  end if
  for each of $\{\overline{X}_t^i\}_{i=1}^{\alpha-d+2}$ within sliding windows of length d on X_t **do**
    probs.append(g($\overline{X}_t^i$)).
  end for
  $\overline{X}_t^{mask} \leftarrow \arg\min_{\overline{X}_t^i}$ probs.
  remove $\overline{X}_t^{mask}$ from X_t.
  while searching epoch<K and conditions $f(X'_t) < \hat{y}'_U$ and $f(X'_t) \in [\hat{y}'_t - \epsilon, \hat{y}'_t + \epsilon]$ does not meet **do**
    while sampling epoch< L **do**
      C_t^m.append(h($X_t, \overline{X}_t^{mask}$)).
    end while
    $\overline{X}_t^{\min} = \arg\min_{\overline{X}_t^s \in C_t^m} \|f((X_t \setminus \overline{X}_t^{mask}) \oplus \overline{X}_t^s) - \hat{y}'_t\|_1$.
    $X'_t \leftarrow (X_t \setminus \overline{X}_t^{mask}) \oplus \overline{X}_t^{\min}$.
  end while
end for

A.3 Data Preprocessing

Global Surface Summary of the Day (GSOD)[3]. Prior to analysis, the GSOD dataset undergoes a series of pre-processing steps. Initially, the daily weather data are converted into monthly aggregates by averaging each month's daily recordings, excluding any missing or invalid entries (*e.g.*, missing or invalid daily precipitation (PRCP) was recorded as 99.99 while that of daily temperature (TEMP) was recorded as 999.99). To mitigate seasonal influences, standardization is applied on a monthly basis. The datasets are then divided temporally into training, validation, and test sets: data before January 1, 2022, forms the training set; data between January 1, 2022, and January 1, 2023, forms the validation set; data after January 1, 2023, forms the test set. Further, sliding windows of 30 time steps (months) are applied to the series. Within each sliding window, the first 24 time steps (months) are used as predictors and the last 6 time steps (months) are used as forecasts.

S&P 500[4]. Its pre-processing involves two primary steps. First, to eliminate long-term trends, the *differencing* method is applied. This technique computes the difference between consecutive data points, effectively detrending the time series. Following *differencing*, the entire time series is standardized. Data with the end time step before January 13, 2023, forms the training set; data between January 13, 2023, and January 13, 2024, forms the validation set; data after January 13, 2024, forms the test set. Sliding windows of 30 time steps (days) are applied to the series. Within each sliding window, the first 25 time steps (days) are used as predictors and the last 5 time steps (days) are used as forecasts.

Dodgers loop sensor[5]. The pre-processing of this dataset involves several steps. First, approximately 5.76% of the dataset contains missing values, which are addressed using the Forward Fill technique. Here, each missing value is replaced with the most recent observed data point. Second, traffic data are aggregated in 3-hour intervals, with each interval represented as a single time step, marked by the final timestamp of each 3-hour period. Third, the entire time series is standardized, normalizing the data to ensure uniformity in scale. Data before August 1, 2005 forms the training set; data between August 1, 2005, and September 1, 2005, forms the validation set; data after September 1, 2005, forms the test set. Additionally, sliding windows of 48 time steps (equivalent to 144 hours or 6 days) are applied to the series. Within each sliding window, the first 40 time steps (120 hours or 5 days) are used as predictors and the last 8 time steps (24 hours or 1 day) are used as forecasts.

A.4 Details for Baseline Algorithms

A.4.1 BaseNN. BaseNN, which is used as a baseline in the study by Wang et al. [32], identifies the closest instance in the training set that aligns with the desired target outcome. Specifically, BaseNN selects from the training set the predictor time series X_t whose subsequent block maxima y_t is closest to the desired target $\hat{y}'_t$. Once identified, it is used as the counterfactual predictor X'_t.

A.4.2 ω-CF. Although it was originally developed for classification tasks, ω-CF [30] can be adapted to a regression scenario. The counterfactual predictor X'_t can be learned through an optimization process defined as

$$X'_t = \arg\min_{X_t^*} \max_{\lambda} \lambda \left| f(X_t^*) - \hat{y}'_t \right| + d(X_t, X_t^*),$$

where λ is a tuning parameter that balances the two components of the objective function and $d(X_t, X_t^*)$ is the Manhattan distance. The learning of *ω-CF* is based on the *Adam* optimizer.

A.4.3 Native guide (NG-CF). Native guide (NG-CF) [8] encompasses a two-step process. Initially, it identifies the nearest unlikely neighbor of original instance X_t. Then, by leveraging the Dynamic Time Warping (DTW) algorithm, NG-CF modifies the identified

[3] https://www.ncei.noaa.gov/access/search/data-search/global-summary-of-the-day?pageNum=1

[4] https://finance.yahoo.com/quote/%5EGSPC/history?period1=1673481600&period2=1705017600&interval=1d&filter=history&frequency=1d&includeAdjustedClose=true

[5] https://archive.ics.uci.edu/dataset/157/dodgers+loop+sensor

neighbor of X_t to construct X'_t, which is engineered to yield a prediction $f(X'_t)$ closely aligned with the desired outcome $\hat{y}'_t$.

A.4.4 ForecastCF. ForecastCF [32], which was initially designed to explain trend forecasting in time series, can be adapted for explaining block maxima prediction by replacing the time series in the forecast window of its original formulation with block maxima value. Specifically, in the real world, assume that the forecasted block maxima $\hat{y}_t$ corresponding to a X_t is extreme. In the counterfactual world, if $\hat{y}_t$ is expected to be non-extreme instead, denoted by $\hat{y}'_t$, how should the X_t be changed to X'_t? Given a forecasting model f, take $\alpha = \hat{y}'_t - \epsilon$ and $\beta = \hat{y}'_t + \epsilon$ as the lower and upper boundaries where ϵ is the tolerance, *ForecastCF* [32] learns X'_t *s.t.* $f(X'_t) \approx \hat{y}'_t : R^{|X'_t|} \rightarrow R^1$ by minimizing the loss function

$$L = \begin{cases} 0, \text{ if } \|f(X'_t) - \hat{y}'_t\| \leq \epsilon, \\ \|f(X'_t) - \alpha\| + \|\beta - f(X'_t)\|, \text{ otherwise,} \end{cases} \tag{15}$$

The learning of *ForecastCF* is based on the *Adam* optimizer.

A.4.5 SPARCE. SPARCE [18] is a GAN-based method used to generate sparse counterfactual explanations for time series, employing both a discriminator and a generator. The discriminator attempts to distinguish between a real instance X_t from its counterfactual instance X'_t while the generator attempts to generate counterfactual instances that could fool the discriminator into misclassifying them as real instances. In our implementation, the generator initially receives a real instance X_t linked to an extreme forecasted block maxima $\hat{y}_t$ and then generates a counterfactual instance X'_t that yields the counterfactual target $\hat{y}'_t$, which is non-extreme. This setup enables the generator to learn how to modify (or perturb) its initial input X_t to construct X'_t, which yields a non-extreme block maxima $\hat{y}'_t$, achieving the desired target. The generator-discriminator architecture is jointly trained to minimize the difference between $\hat{y}'_t$ and $f(X'_t)$, which is also constrained to guarantee the sparsity of X'_t.

A.5 Experimental Setup

For a fair comparison, all the experiments were conducted using the same trained DeepExtrema model on each dataset to generate block maxima forecasts. For the predictor and forecast windows set for the DeepExtrema model, their lengths were chosen as the ones that yielded the best performance or were used in previous works. The RMSE of the trained DeepExtrema model evaluated on the test set varies between 0.40 and 0.75 on the given datasets. For each block maxima forecast, a counterfactual target is then constructed to be utilized by both *DiffusionCF* and all the baselines. To define the counterfactual target, we first identify a counterfactual threshold $\hat{y}'_U$ to determine whether the forecasted block maxima is extreme. Towards this end, we set the quantile p of the GEV distribution in such a way that around 10% (8.8% − 11.1%) of the forecasted block maxima will be considered as extreme values. Based on this threshold, a counterfactual target $\hat{y}'_t$ is subsequently constructed using the quantile $p' = p - 0.2$. It is worth noting, however, that our framework operates independently of the DeepExtrema model's performance.

On the *GSOD* precipitation dataset, the DeepExtrema model is configured with the following hyperparameters: *batch size* is set to 128, *learning rate* to 0.005, *dimension of hidden layer* to 32, *number of hidden layers* to 2, λ_1 to 0.8, and λ_2 to 0.5. For ForecastCF and ω−CF, when $\hat{y}_t$ is extreme and $\hat{y}'_t$ is not, the *maximum iterations* for learning is set to 300. For ForecastCF, *tolerance* is set to 0.05. For SPARCE, we use the default parameters. Considering the balance between time and accuracy for *DiffusionCF*, the *sampling window length* is searched in $[3, 6, 9, 12]$. For each of them, the searching epochs K (as shown in line 16 in Algorithm 1) is set to 5. In each of the search windows, the *sampling epochs* L (as shown in line 17 in Algorithm 1) is set to 300. This involves determining the least likely sub-time series $\overline{X}_t$ of *sampling window length* in X_t, removing it, and then sampling a new $\overline{X}'_t$ of the same length from the learned latent distribution for imputation. The parameter balancing the two components of the loss function in VAE is set to 10^{-8}

On the *GSOD* temperature dataset, the DeepExtrema model is configured with the following hyperparameters: *batch size* is set to 128, *learning rate* to 0.001, *dimension of hidden layer* to 10, *number of hidden layers* to 2, λ_1 to 0.1, and λ_2 to 0.5. The parameter balancing the two components of the loss function in the VAE is set to 10^{-9}, and for anomaly detection within the VAE, it is set to 1.0. Other parameters are identical to those used for the GSOD precipitation data.

On the *S&P 500* dataset, the DeepExtrema model is configured with the following hyperparameters: *batch size* is set to 128, *learning rate* to 0.0001, *dimension of hidden layer* to 8, *number of hidden layers* to 2, λ_1 to 0.2, and λ_2 to 0.5. The *sampling window length* is searched in $[5, 10, 15]$. Other parameters are identical to those used for the GSOD temperature data.

On the *Dodgers loop sensor* traffic forecasting dataset, the DeepExtrema model is configured with the following hyperparameters: *batch size* is set to 128, *learning rate* to 0.0005, *dimension of hidden layer* to 16, *number of hidden layers* to 2, λ_1 to 0.1, and λ_2 to 1.5. The *sampling window length* is searched in $[5, 10, 15, 20]$. Additionally, the parameter balancing the two components of the loss function in the VAE for anomaly detection is set to 1.0 when its sampling size is 40. For smaller sampling sizes, specifically 5, 10, 15, or 20, the balancing parameter is adjusted to 5.0. This variation in settings is aimed at optimizing the model's performance for different data granularity and anomaly detection contexts. Other parameters are identical to those used for the GSOD temperature data.