Data Regression Framework for Time Series Data with Extreme Events

Yifan Zhang
Dept. of Computer Science and Engineering
University of Nevada, Reno
Reno, USA
yfzhang@nevada.unr.edu

Jiahao Li
Dept. of Computer Science
East Carolina University
Greenville, USA
lij17@students.ecu.edu

Ablan Carlo
Dept. of Computer Science
East Carolina University
Greenville, USA
ablanc17@students.ecu.edu

Alex K Manda
Dept. of Geological Sciences
East Carolina University
Greenville, USA
mandaa@ecu.edu

Scott Hamshaw
Dept. of Civil and Environmental Engineering
University of Vermont
Burlington, USA
scott.hamshaw@uvm.edu

Sergiu M. Dascalu
Dept. of Computer Science and Engineering
University of Nevada, Reno
Reno, USA
dascalus@cse.unr.edu

Frederick C. Harris, Jr.
Dept. of Computer Science and Engineering
University of Nevada, Reno
Reno, USA
fred.harris@cse.unr.edu

Rui Wu
Dept. of Computer Science
East Carolina University
Greenville, USA
wur18@ecu.edu

***Abstract*—Time series data are significant to scientific, social, economic, and other areas, such as the prediction of weather changes being instrumental for administrative decision-making. In recent years, deep learning methods have achieved great success in time series prediction when compared with classic machine learning methods. However, because time series data can dynamically change and the correlations between the target variable and other features can also vary, making predictions using time series data is often challenging. To further improve existing machine learning and deep learning models for time series prediction, we propose a framework to integrate machine learning models with anomaly detection algorithms. The extreme events are highlighted so the machine learning models can process them appropriately. We conducted extensive experiments on real-world datasets ranging in size from a few hundred to more than ten thousand records. The experimental results demonstrate that our proposed framework significantly improves machine learning model accuracy and mitigates the accuracy descending rate when the predicting horizon (i.e., the number of timestamps ahead) increases.**

***Index Terms*—Neural Networks, Recurrent Neural Network, Time Series Prediction, Time Series Analysis**

I. Introduction

Regression analysis of time series data is important as the research results are widely used in different domains to solve real-world problems. For example, Fischer et al. [1] applied long short-term memory networks (LSTM) for financial time series prediction and found out that LSTM is inherently suitable for this domain. Kratzert et al. [2] utilized LSTM in streamflow forecasting and conclude that physical constraints can be beneficial to the LSTM model. Wu et al. [3] leveraged genetic algorithm and linear regression models to accurately predict nitrate, which is very important for animals and plants. However, it is challenging to predict time-series data with machine learning models if the input data includes extreme events [4], [5]. Extreme events can be defined as unusual events, such as hurricanes, equipment failures, and stock market crashes. They are at the tail part of data distribution and less likely to happen. In this paper, we propose a data regression framework to help a machine learning model learn the differences between normal events and extreme events to achieve better performance.

Extreme events, such as hurricanes and earthquakes, are unusual but can be very important. Because the data distributions of normal events and extreme events can be different and change over time, the prediction of time series data (such as of stream flows) with extreme events can be challenging [6]. A single machine learning model may not be able to accurately predict differently distributed data. Because of this, the "sliding window" concept has been proposed and widely used to load the data inside of the window and make predictions [7]. However, the "sliding window" may not be enough based on our experimental results (see Baseline vs. Percentile vs. Mask in Table I, Table II, and Table III), especially when a model is used to forecast values for long periods of time ahead. Our proposed framework consists of two regression pipelines (*extreme event split* and *mask matrix*) to help machine learning models learn which are extreme events and which are normal

This project was supported by a grant from the North Carolina Sea Grant (Award number:2018-1045-08;R/20-RCE-2).

events within a sliding window. For the extreme event split pipeline, we propose to split time series data within a sliding window into extreme and normal events and implement two machine learning models to predict them, respectively. For the mask matrix pipeline, a mask matrix is created to mark extreme events and their weights. As shown later in the paper, the experimental results show that the proposed framework can enhance machine learning model prediction performance for time series datasets.

The contributions of this paper are:

- A framework is proposed to combine machine learning models with extreme event detection algorithms using extreme event split and mask matrix regression pipelines. The accuracy is improved based on our experimental results and the accuracy descending rate is smaller compared to machine learning models which are not using the proposed framework.
- We compare extreme event split and mask matrix pipelines and discuss the reasons that these two pipelines can enhance machine learning model predictions.
- The modular design of the proposed framework allows users to flexibly choose the appropriate sub-module content according to specific problems. For instance, the machine learning models and extreme event detection algorithms can be selected according to the characteristics of the data.

The rest of this paper is organized as follows: Section II describes the problem that we are trying to solve. Section III presents well-known and related work and discusses what can possibly be improved. Section IV introduces the proposed framework and illustrates how the two pipelines can be applied. Section V presents the experimental results, highlights the advantages of our proposed framework, and discusses which pipeline should be selected under different scenarios. Conclusions are presented in Section VI.

II. Problem Description

Let $X \in R^N$ denote a vector that includes N records of a single variable and x_i denote the single variable collected at timestamp i. Meanwhile, $x_{i-1}, x_{i-2}, \cdots, x_{i-l-1}, x_{i-l}$ denote variables collected at $1, 2, \cdots, l-1, l$ timestamps before timestamp i, considered as a total of l lags and used as extra features to predict x at timestamp $i+t$ denoted as $\hat{x}_{i+t}$. Univariate time series data prediction is defined with the following equation:

$$\hat{x}_{i+t} = f(x_i, x_{i-1}, x_{i-2}, \cdots, x_{i-l-1}, x_{i-l}) \tag{1}$$

where f is a regression model, $x_i, x_{i-1}, x_{i-2}, \cdots, x_{i-l-1}$, x_{i-l} are the features used for prediction, and $\hat{x}_{i+t}$ is the predicted value. In the above, t is the horizon, i.e., the number of steps ahead of i.

The multivariate time series data prediction problem is defined as predicting the value of the target variable at a certain timestamp or horizon forward from historical multivariate data. Let $X \in R^{M \times N}$ denote multivariate time series data of dimension $M \times N$, where the columns of X are sets of variables collected at certain timestamps and the rows of X are sets of variables collected at the same timestamp. For instance, x_{ij} denotes the value of variable j collected at timestamp i, X_i is a vector of N variables collected at timestamp i and can be defined as the vector of $\{x_{i1}, x_{i2}, x_{i3}, \cdots, x_{iN-1}, x_{iN}\}$. y_i denotes the predicted value of the target variable at timestamp i. Given sequences of historical multivariate data and target variable of length l, $\{X_i, X_{i-1}, \cdots, X_{i-l}\}$ and $\{y_i, y_{i-1}, ..., y_{i-l}\}$, the framework is aimed to forecast the value of target variable y at timestamp $i+t$, as the following equation shows:

$$\hat{y}_{i+t} = f((X_i, y_i), (X_{i-1}, y_{i-1}), ..., (X_{i-l}, y_{i-l})) \tag{2}$$

In this paper, our proposed framework aims to enhance the model f with the extreme event split pipeline and the mask matrix pipeline. These are described in detail in Section IV.

III. Related Work

In recent years, deep learning has achieved great success in the fields of computer vision and natural language processing. Inspired by this, deep learning was introduced to the time series prediction problem and showed excellent performance against conventional methods. Karim et al. [8] modified the Long Short Term Memory Fully Convolutional Network by augmenting the convolutional block with a squeeze-and-excitation block. Qiu et al. [9] utilized the ensemble method of deep learning belief networks (DBN) for time series forecasting and aggregated the outputs from various DBNs with a support vector regression model. Wu et al. [10] proposed a graph neural network based multivariate time series forecasting model, which treats the variables as nodes in the graph trained as a whole to capture the spatial and temporal dependencies within the multivariate time series. Siami et al. [11] introduced an LSTM model that outperformed various AutoRegressive Integrated Moving Average (ARIMA) variations on predicting time series data. It showed the great potential of artificial recurrent neural networks in the task of time series data prediction. Sagheer et al. [12] suggested that LSTM units and recent deep Recurrent Neural Networks (RNN) networks fall short when predicting highly nonlinear time series data during large horizons. Hence, they proposed a pre-trained LSTM based stacked autoencoder and conducted unsupervised learning to replace the traditional random weight initialization operation for RNN. Yamak et al. [13] investigated the performance of LSTM and Gated Recurrent Unit (GRU) and their experiment showed that the ARIMA model can outperform them occasionally on some datasets. The authors concluded that both LSTM and GRU have their vulnerabilities and limitations and need fine-tune hyperparameters in implementation to fully exploit their performance. Saini [14] demonstrated the effectiveness of GRU and LSTM compared to vanilla RNN and Support Vector Regression (SVR) in predicting short-term agriculture loads. It is worth mentioning that although GRU and LSTM had similar performances, the GRU model converged on the results quicker due to its simpler neural network architecture. Generally, deep learning methods can obtain the underlying patterns of the time series through data, without having to develop a model and manually designing features for specific problems, making them good for generalization.

Besides deep learning, traditional machine learning models, such as support vector machine [15], linear regression [16], and statistical methods, such as autoregressive models [17], are also commonly used for time series data prediction problems. To our best knowledge, there are very few publications about how to leverage extreme event information to enhance time-series data prediction.

However, what researchers often overlook is that extreme events can mislead a machine learning model because of different distributions. Therefore, extreme events should be either analyzed and processed or removed [18]. In our proposed framework, either pipeline requires that extreme events be detected and labeled. Many research papers have been published on anomaly event detection and can be applied in our proposed framework. For instance, Siffer et al. [19] utilized the Extreme Value Theory and proposed the SPOT method to detect anomalies in univariate time series. In this paper, SPOT and a simple percentile based split method was investigated. The SPOT method doesn't assume the distribution of the data and can automatically set thresholds in the sliding window. Meanwhile, the percentile method is a simple way to set the threshold directly based on the predefined percentile.

IV. Proposed Framework

In this section, we describe the framework for the multivariate time series prediction. As shown in Fig. 1, our proposed framework consists primarily of two data regression pipelines: extreme event split (upper path) and mask matrix (lower path).

A. Lags and Data Transformation

Using lags and transforming data are used in this paper to further improve the time-series machine learning model prediction accuracy.

1) Lags: For time series data, variables can have a high correlation with their historical record. Based on this assumption, we can leverage lagged selected variables at earlier timestamps as extra features, i.e., using X and the variables at earlier timestamps as model inputs. For example the temperature at timestamp t is always highly correlated with temperature at timestamp $t-1$, $t-2$, and $t-3$. We can use Y_{t-1}, Y_{t-2}, Y_{t-3} as extra features for temperature predictions.

2) Data Transformation: Different machine learning models can have different requirements about the dataset. For example, the input data follows normal distribution or the machine learning models are very sensitive to the scale of data. To fulfill these requirements, data transformation methods should be applied.

In this paper, we utilized `MinMaxScaler()` from the scikit-learn package [20] to scale the original variables to between 0 and 1. After obtaining the prediction results from the framework, the predictions are not on the same scale as the true values since we scaled the original data. Hence, we applied the inverse operation of `MinMaxScaler()` . Subsequently, the output of the back transformation operation is in the original scale. Therefore, we have the final prediction values.

B. Regression Pipelines: Extreme Event Split and Mask Matrix

The data distribution of a time series dataset can vary. This is commonly seen in climate data problems, such as underground water level, precipitation, and temperature forecasting. So, the forecasting strategy should be altered when the data distribution is changed. To handle the challenge of different data distributions, we propose two pipelines in this paper: *extreme event split* and *mask matrix*.

1) Extreme Event Split: Because data distributions of a time series dataset can change, a single machine learning model trained based on the whole time series dataset without data distribution variation information can be less accurate when compared with multiple machine learning models trained with different data segments. Accordingly, we propose to apply the "sliding window" concept and split data into normal and extreme events. The "sliding window" concept is commonly used for time series data predictions described as Algorithm 1.

Algorithm 1 Sliding window anomaly detection

```
**Input:** Target variable $y$, Target variable size $N$, Window size $w$, Anomaly detection method $f_s$.
**Sliding_window:**
$i = 1$
**while** $i \leq (N - w + 1)$ **do**
    $v_i = f_s(y_i, y_{i+1}, y_{i+2}, \cdots, y_{i+w-1})$
    $i = i + 1$
**end while**
**Output:** Label vector $v$. (The vector consists of 0 and 1 to identify anomalies)
```

Although data distribution may vary for the whole dataset. However, a data segment within the window can have a stable data distribution. The window can move forward with new data records added and old data records removed from the window, i.e., the data segment. The window size w can be decided by repeated patterns in the data and calculated using the autocorrelation formula [21].

Inside a window, we can further split the data into normal and extreme events. Normal events are the major part based on the data distribution inside the window (e.g., between the 20th percentile and 80th percentile of the data). The extreme events are usually very high or low values according to the data distribution (e.g., below 20th percentile or above 80th percentile). To split data into normal and extreme events, thresholds should be dynamically generated, and then classify the samples into normal and extreme categories. In this paper, the following two methods are experimented with and discussed:

- Percentile split is based on predefined percentiles hyperparameters (e.g., 95 percentile and 5 percentile) within

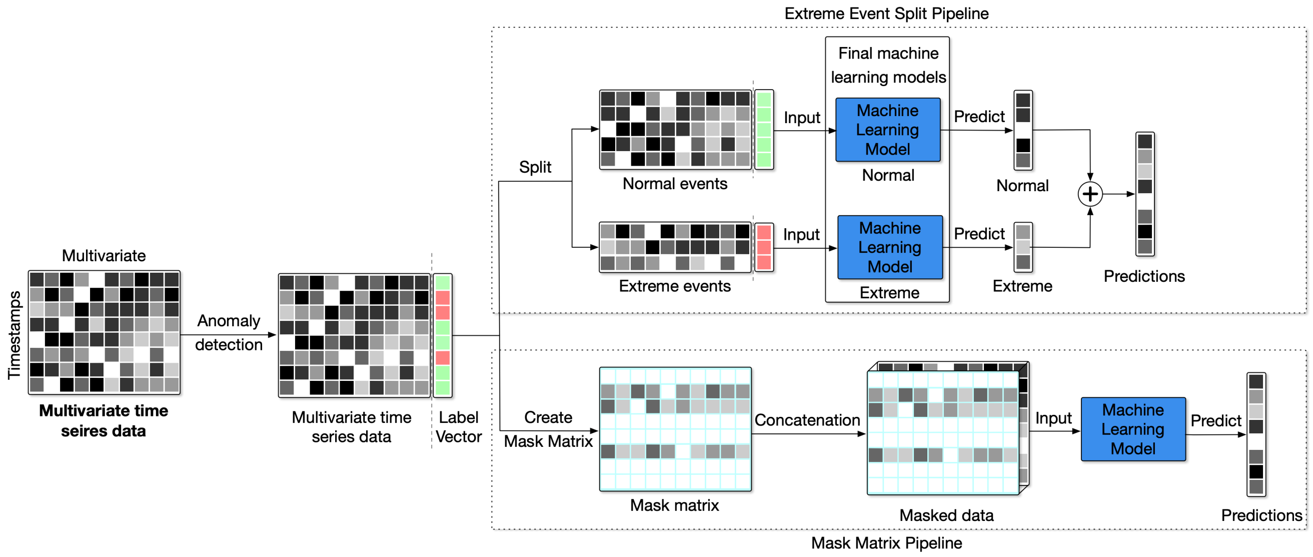


Fig. 1: Illustration of the proposed framework consists of two data regression pipelines: extreme event split and mask matrix. (1) A selected anomaly detection method first detects the extreme events in the data and creates a label vector from the target variable. Afterward, the data and label vector are used as input for both pipelines. (2) The extreme event split pipeline divides the data into two parts according to the label vector, then predicts normal and extreme events separately. Finally, the predictions of the two models are combined according to their order in the original data. (3) The mask matrix pipeline creates a mask matrix based on the input data and label vector and concatenates the mask matrix with the input data. Then machine learning model takes masked data as input to generate predictions.

the sliding window. It will calculate the threshold value of predefined percentiles within the sliding window as the thresholds. The values above the upper threshold or below the lower threshold will be labeled as extreme events. Concurrently, the values in between will be labeled as normal events. When the window slides over the timestamps of the dataset step by step, the thresholds will also be updated synchronously.

- SPOT [19] is an anomaly detection approach proposed in 2017 for univariate time series data based on extreme value theory. It can label extreme and normal events based on dynamically updated thresholds and does not make any assumptions on the distribution of data.

These two methods can be used to scan the target variable (i.e., Y) and help machine learning models distinguish extreme and normal events by splitting historical data into extreme event groups and normal event groups.

Both the percentile split and SPOT are threshold-based algorithms. Therefore, given an original multivariate time series data D, the anomaly detection algorithms, can be leveraged to split the original dataset as the following equation shows:

$$[D_{norm}, D_{ext}] = f_s(D) \tag{3}$$

Where f_s denotes the extreme event split method used to generate label vector, then using the label vector separate data into normal and extreme events. D_{norm} and D_{ext} are the normal event sub-dataset and the extreme event sub-dataset of dataset D. Each sub-dataset includes the target variable y and features X. To preprocess both sub-datasets, we perform data transformation (scaling with max and min of variables before split) and create lags for each sub-dataset as described in IV-A, defined as follows:

$$[X_{norm}, X_{ext}] = f_{preprocessing}(D_{norm}, D_{ext}) \tag{4}$$

Where $f_{preprocessing}$ denotes the preprocessing operation. It's worth mentioning that the X_{norm} and X_{ext} include lags created from both the feature variables and the target variable. Then X_{norm} and X_{ext} are then used as inputs to the models.

Two machine learning models, normal model f_{norm} for normal event predictions and extreme model and f_{ext} for extreme event predictions, are trained based on the two sub-datasets, respectively. The predictions can be denoted with the following equation:

$$\hat{Y} = f_{concat}(f_{norm}(X_{norm}), f_{ext}(X_{ext})) \tag{5}$$

where f_{norm} and f_{ext} are the normal model and the extreme model, respectively. After the predictions were generated by both models, we concatenate the normal predictions and extreme predictions by f_{concat} according to the order of events in the original data D. Therefore, $\hat{Y}$ will be the predictions including both normal and extreme events. Finally, we conduct the inverse operation of `MinMaxScaler()` to transform the predictions in the original scale.

2) Mask Matrix: We further propose the mask matrix pipeline. Instead of dividing the dataset into two sub-datasets and train two models. Mask matrix pipeline creates a mask matrix that can be used to label and quantify the extent of extreme events in the historical data. Prediction results can be more accurate with one machine learning model and the mask matrix. The original multivariate time series data with m variables collected in a total of n samples can be defined as a matrix, denoted as follows:

$$D_{n,m} = \begin{pmatrix} y_{1,1} & x_{1,2} & \cdots & x_{1,m-1} & x_{1,m} \\ y_{2,1} & x_{2,2} & \cdots & x_{2,m-1} & x_{2,m} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ y_{n,1} & x_{n,2} & \cdots & x_{n,m-1} & x_{n,m} \end{pmatrix}$$

where $y_{i,j}$ are values in target variables, $x_{i,j}$ are values in related feature variables. Then selected anomaly detection method f_s in Algorithm 1 and apply it to the original multivariate time series data $D_{n,m}$. For extreme events, by calculating the percentile of the value in the variable at the current timestamp, we can get the corresponding mask value in the mask matrix. The mask matrix is defined as follows:

$$M_{n,m} = \begin{pmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,m-1} & m_{1,m} \\ m_{2,1} & m_{2,2} & \cdots & m_{2,m-1} & m_{2,m} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ m_{n,1} & m_{n,2} & \cdots & m_{n,m-1} & m_{n,m} \end{pmatrix}$$

where $m_{i,j}$ are used as extreme weights. If an observed value is an extreme event, then $m_{i,j}$ should be the percentile value of the corresponding variable at timestamp i. Otherwise, $m_{i,j}$ will be assigned to zero. Then we preprocess data $D_{n,m}$, defined as follows:

$$X = f_{preprocessing}(D_{n,m}) \tag{6}$$

As for mask matrix $M_{n,m}$, We only need to create lags because their scales are already consistent, defined as follows:

$$M = f_{lag}(M_{n,m}) \tag{7}$$

where f_{lag} is the operation creates lags that described in IV-A. Noting that the data $X$and the mask matrix M are in the same shape, the element in the mask matrix M represents the mask value of the element at the same position in the data X matrix.

The objective is to predict target variables based on historical data and a mask matrix M defined as follows:

$$\hat{Y} = f(X, \alpha * M) \tag{8}$$

where preprocessed data X and mask matrix M are the inputs for the framework, $\hat{Y}$ denotes the predicted target variable, and α is the weight to control mask matrix impacts. The same, we conduct inverse operation of `MinMaxScaler()` to transform the predictions in the original scale.

V. Experiment Results

In this section, we conduct experiments on 10 datasets including extreme events collected from 3 sites with a sample size from several hundred to over ten thousand records. The results show that our proposed framework effectively alleviates the descending of accuracy with the increase of horizon.

A. Datasets and Implementation Details

Datasets and Metrics: To investigate the effectiveness of the proposed framework, we utilized two datasets collected from research sites located in North Carolina and one dataset collected in Vermont. The first dataset named EmerladIsle was collected from Carteret County. There are 375 data records are collected from 4 underground wells from 2017-2018 with a sampling frequency of one day. These wells were spread all throughout the Emerald Isle and Atlantic Beach in North Carolina. The second dataset named aquifers was collected from three surficial aquifers in Dublin, Littlefield, and Magnolia at Robeson and Bladen County also in North Carolina with a sampling frequency of one day, with a total of 4578 samples. The third dataset named Streamflow was collected from the Mad River and two of its tributaries [22], Shepard Brook and Mill Brook, located in the Lake Champlain basin in central Vermont. The Mad River watershed has a distinct seasonal variation. Summer months feature warm temperatures and frequent, fast-moving, convective, rainstorms that produce moderate rainfall. Discharge data is the target variable for prediction and was available from the USGS Geological Mad River gauging station (No. 04288000) for the Mad River, and by developing stage-discharge relationships for the Mill Brook and Shepard Brook sites. Rainfall and soil moisture data were collected from a network of tipping bucket rain gauges and a single meteorological station equipped with soil moisture sensors at multiple depths.

Three metrics were chosen to evaluate accuracy including, Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and coefficient of determination (R^2). For RMSE, MAE, lower values are better. For R^2, the value is between 0 and 1 and higher values are better.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2} \tag{9}$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i| \tag{10}$$

$$R^2 = \left\{ \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)(\hat{y}_i - \bar{\hat{Y}})}{\left(\sum_{i=1}^{N}(y_i - \bar{Y})^2\right)^{\frac{1}{2}} \left(\sum_{i=1}^{N}(\hat{y}_i - \bar{\hat{Y}})^2\right)^{\frac{1}{2}}} \right\}^2 \tag{11}$$

where $\hat{y}_i$ and y_i represent the predicted and observed values respectively; $\bar{Y}$ is the mean of the observed values and $\bar{\hat{Y}}$ is the mean of predicted values for the entire evaluation period.

Model implementations: To demonstrate the effectiveness of the proposed framework, we set GRU as the baseline model for comparison. The GRU model is composed of one GRU layer and one dense layer, the activation function in both layers is set to linear. The Mean Absolute Error was selected as the loss function along with the Adam optimizer, and the learning rate is set to 1×10^{-3}.

TABLE I: EmeraldIsle Site Daily Underground Water Level Forecasting Comparison

Dataset		**Well 13**			**Well 16**			**Well 19**			**Well 28**		
		Horizon			**Horizon**			**Horizon**			**Horizon**		
Method	**Metrics**	2	6	12	2	6	12	2	6	12	2	6	12
Baseline	R^2	0.678	0.193	-0.101	0.762	0.260	-0.058	0.688	0.211	-0.148	0.583	-0.02	-0.062
	RMSE	**0.002**	**0.006**	**0.010**	**0.002**	**0.006**	**0.009**	**0.003**	**0.007**	**0.011**	**0.004**	**0.010**	**0.012**
	MAE	0.028	0.053	0.074	**0.027**	0.055	0.075	0.030	0.063	0.083	**0.034**	0.069	0.082
SPOT	R^2	0.665	0.219	-0.098	0.690	0.230	-0.164	0.635	0.234	-0.092	0.438	-0.278	-0.325
	RMSE	0.049	0.078	0.095	0.045	0.076	0.095	0.054	0.083	0.103	0.069	0.106	0.115
	MAE	0.030	0.054	0.073	0.032	0.058	0.077	0.032	0.064	0.084	0.039	0.074	0.085
Percentile	R^2	**0.853**	**0.553**	0.079	**0.795**	**0.431**	-0.050	**0.724**	**0.463**	0.140	**0.683**	**0.439**	-0.318
	RMSE	0.034	0.055	0.088	0.036	0.060	0.091	0.048	0.066	0.087	0.054	0.069	0.116
	MAE	**0.024**	**0.040**	0.066	**0.027**	**0.045**	0.070	**0.034**	**0.049**	**0.064**	0.036	**0.049**	0.084
Mask	R^2	0.676	0.543	**0.240**	0.737	0.380	**0.042**	0.649	0.294	**0.168**	0.543	0.265	**0.005**
	RMSE	0.051	0.060	0.079	0.044	0.067	0.082	0.054	0.081	0.088	0.068	0.081	0.100
	MAE	0.033	0.046	**0.061**	0.032	0.053	**0.066**	0.035	0.064	0.068	0.041	0.063	**0.075**

[a]Sample of a Table footnote.

TABLE II: Aquifer Site Daily Underground Water Level Forecasting Comparison

Dataset		**Dublin**				**Littlefield**				**Magonia**			
		Horizon				**Horizon**				**Horizon**			
Method	**Metrics**	2	6	12	20	2	6	12	20	2	6	12	20
Baseline	R^2	0.913	0.802	0.672	0.543	0.820	0.612	0.452	0.302	0.946	0.800	0.655	0.502
	RMSE	**0.095**	**0.222**	**0.378**	**0.538**	**0.121**	**0.263**	**0.372**	**0.481**	**0.126**	**0.453**	0.814	1.251
	MAE	0.158	0.313	0.452	0.569	0.174	0.319	0.441	0.532	0.205	0.461	0.674	0.874
SPOT	R^2	0.906	0.758	0.618	0.502	0.806	0.577	0.413	0.255	0.938	0.811	0.614	0.456
	RMSE	0.335	0.533	0.670	0.784	0.349	0.523	0.624	0.707	0.379	0.673	0.989	1.187
	MAE	0.201	0.350	0.469	0.589	0.199	0.333	0.456	0.546	0.237	0.465	0.724	0.893
Percentile	R^2	0.910	0.809	0.692	0.580	0.813	0.627	0.462	0.330	0.942	**0.863**	**0.777**	**0.651**
	RMSE	0.328	0.464	0.585	0.700	0.340	0.476	0.587	0.665	0.376	0.573	**0.716**	**0.898**
	MAE	0.245	0.458	**0.397**	**0.506**	0.176	0.291	0.392	0.463	0.213	0.401	**0.531**	**0.633**
Mask	R^2	**0.927**	**0.836**	**0.735**	**0.616**	**0.863**	**0.707**	**0.583**	**0.457**	**0.955**	0.855	0.742	0.635
	RMSE	0.283	0.429	0.558	0.684	0.290	0.439	0.528	0.609	0.322	0.579	0.780	0.931
	MAE	**0.157**	**0.280**	0.420	0.520	**0.158**	**0.289**	**0.360**	**0.433**	**0.173**	**0.395**	0.568	0.688

In order to test the effectiveness of the proposed framework, we compared the results of the framework working with SPOT and Percentile extreme event labeling algorithms. For SPOT, we utilized the Github code from Alban Siffer [23]. The percentile split algorithm was implemented by calculating a threshold percentile of the targeted variable in the sliding window. The size of the sliding window and the threshold percentile are required as the framework input. The window size can be decided based on data patterns and can be estimated using the auto-correlation formula [21]. Then extreme events can be labeled according to whether they are within the threshold range.

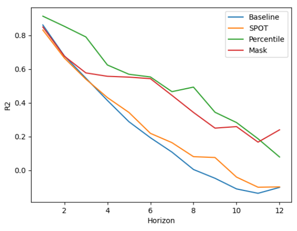
For the extreme event split pipeline, we implemented the normal model and extreme model to accommodate the different distributions of the normal and extreme events. In order to maintain consistency, we utilize the same GRU model with baseline as the Normal model. For extreme events, to ensure the fairness of comparison, we utilize the Gradient Boosting for regression(GBR) machine learning model. By splitting the data into normal and extreme and predicting separately, the following results show that even if we choose GBR as the extreme model, the prediction accuracy is still improved when compared with using GRU alone.

For the mask matrix pipeline, we used an extreme event labeling algorithm (i.e., SPOT or Percentile) to label each timestamp and created the mask matrix in a column by column manner. The mask matrix tells a machine learning model how far away this variable at a certain timestamp is from the threshold percentile. We utilize the same GRU model with baseline. The results of baseline and the other methods are presented next.

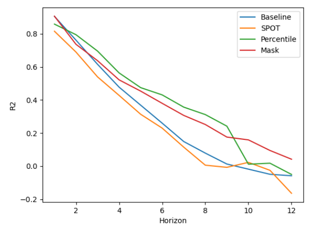
B. *Experiment Results and Analysis*

The experiment results are summarized in Table I, Table II, and Table III. We tested our framework with 10 datasets in total and evaluated the performance using R^2, $RMSE$, and MAE metrics. We selected horizons of 2, 6, 12, and 20 to investigate how the proposed framework performs on different horizons. The results show that our framework can effectively improve the baseline model on the time series data prediction. We further tested our framework from horizon 1 to horizon 20 and drew box plots of average descending rate in Fig. 2, Fig. 3, and Fig. 4. The accuracy, i.e. R^2, is decreasing with the increase of horizon. However, the results show that our proposed framework has an overall smaller descending rate, which means the accuracy gap between baseline and our framework enlarges larger as the horizon increases.

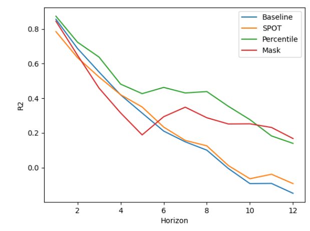
Baseline. Table I shows the results of four datasets collected from Emerald Isle. The four datasets include daily data col-



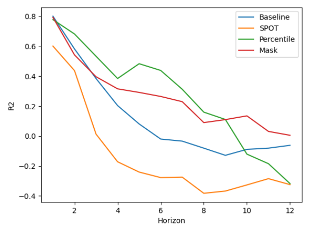
(a) R^2 Well 13



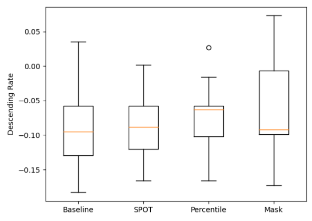
(b) R^2 Well 16



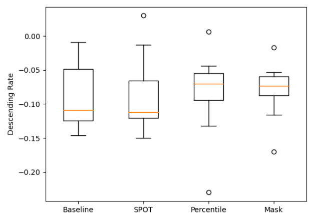
(c) R^2 Well 19



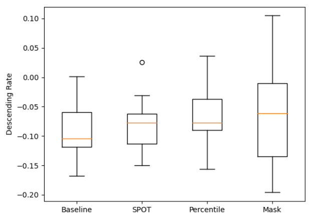
(d) R^2 Well 28



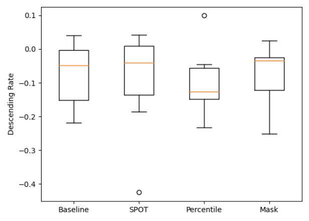
(e) Descending Rate Well 13



(f) Descending Rate Well 16

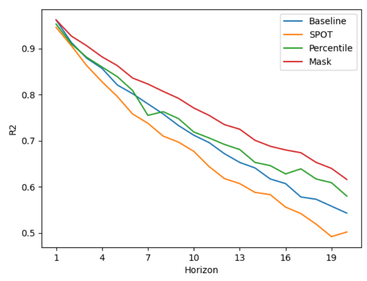


(g) Descending Rate Well 19

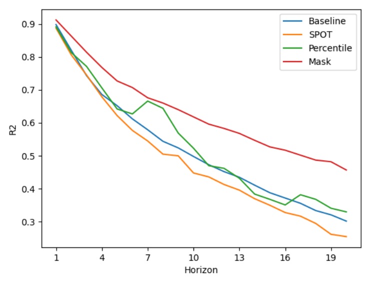


(h) Descending Rate Well 28

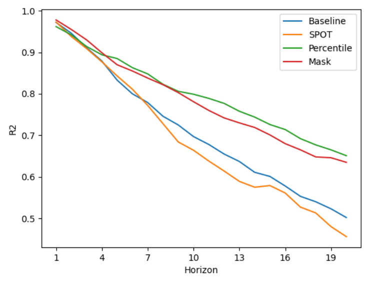
Fig. 2: EmeraldIsle Site



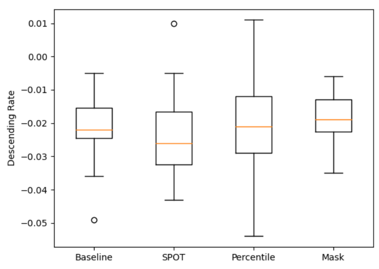
(a) R^2 Dublin



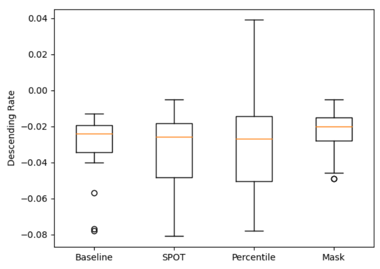
(b) R^2 LittleField



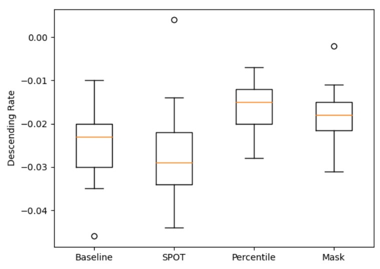
(c) R^2 Magonia



(d) Descending Rate Dublin



(e) Descending Rate LittleField

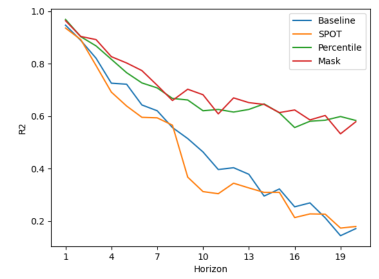


(f) Descending Rate Magonia

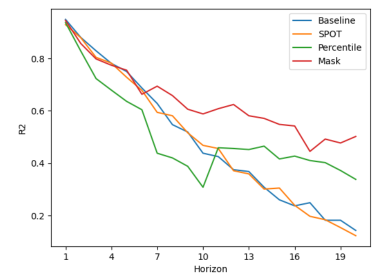
Fig. 3: Aquifers Site

TABLE III: Streamflow Site Hourly Streamflow Forecasting Comparison

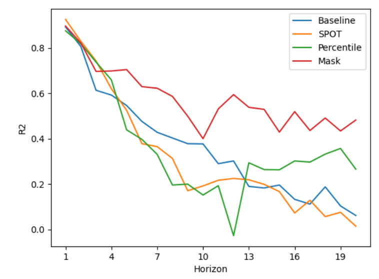
Dataset		Mill				Mad				Shepard			
		Horizon				Horizon				Horizon			
Method	**Metrics**	2	6	12	20	2	6	12	20	2	6	12	20
Baseline	R^2	0.889	0.643	0.404	0.172	**0.880**	**0.687**	0.375	0.143	0.806	0.476	0.302	0.062
	RMSE	**0.357**	0.977	1.451	1.998	8.718	31.597	61.532	79.882	**0.117**	**0.329**	**0.440**	0.575
	MAE	0.205	0.379	0.517	0.607	1.235	1.973	2.803	3.321	0.090	0.196	0.230	0.286
SPOT	R^2	0.892	0.596	0.345	0.180	0.879	0.678	0.371	0.123	0.829	0.378	0.225	0.015
	RMSE	0.467	0.844	1.177	1.292	**2.714**	**4.693**	6.655	7.753	0.288	0.521	0.609	0.671
	MAE	**0.189**	0.330	0.545	0.608	**1.085**	1.987	3.001	3.670	**0.072**	0.188	0.247	0.313
Percentile	R^2	0.903	0.727	0.616	**0.584**	0.828	0.604	0.456	0.338	**0.863**	0.338	0.380	0.203
	RMSE	0.461	**0.760**	0.912	0.942	3.179	4.902	5.981	6.413	0.268	0.544	0.548	0.576
	MAE	0.202	0.356	0.452	0.458	1.421	**1.941**	2.502	**2.601**	0.092	0.203	0.213	0.226
Mask	R^2	**0.904**	**0.774**	**0.670**	0.579	0.857	0.663	**0.624**	**0.502**	0.836	**0.636**	**0.566**	**0.487**
	RMSE	0.457	0.774	**0.842**	**0.939**	3.043	4.838	**5.108**	**5.970**	0.285	0.434	0.489	**0.526**
	MAE	0.248	**0.294**	**0.366**	**0.451**	1.236	2.410	**1.974**	2.681	0.097	**0.149**	**0.194**	**0.222**



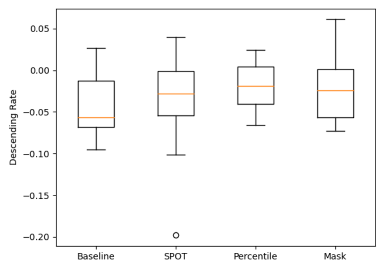
(a) R^2 Mill



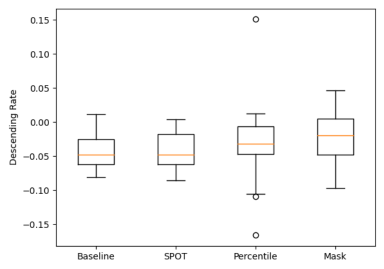
(b) R^2 Mad



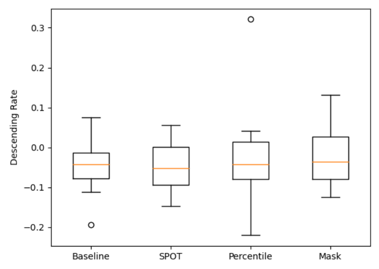
(c) R^2 Shepard



(d) Descending Rate Mill



(e) Descending Rate Mad



(f) Descending Rate Shepard

Fig. 4: Streamflow Site

lected from 2007 to 2008. Baseline achieved an average R^2 of 0.678 for horizon 2. However, the accuracy of the baseline drops rapidly as the horizon increases. The average R^2 of horizon 6 is 0.161, 76.3% lower than horizon 2. When the horizon was 12, the baseline model achieves an R^2 lower than zero on all datasets in Table I.

In Table II, Baseline achieved very promising prediction accuracy with an average R^2 of 0.893 when the horizon was 2. This number decreased to 0.738, 0.593, and 0.449, when the horizon was 6, 12, 20, they are respectively 17.4%, 33.6%, and 49.7% lower than when the horizon was 2. The performance on horizons 12 and 20 are still very promising, mainly because this site has 4,578 samples, which enabled the models to be better trained.

For the Streamflow site, there are more than ten thousand samples. However, the R^2 Baseline drops rapidly. In Table III, the Baseline has an average R^2 of 0.858 on horizon 2, and drops to 0.602, 0.360, and 0.126 for horizon 6, 12, and 20. Considering that this site was collected hourly when we want to predict a day into the future (horizon 24), the accuracy of the Baseline will be terrible.

Extreme Event Split. After analyzing the experimental results, both the extreme event split and mask matrix pipelines show promise for improving machine learning model predictions. Although the Mask Matrix framework achieved better performance in datasets with a larger sample size, the Extreme Event Split framework still performed with the best accuracy in some circumstances. It is worth mentioning that, the improvement of accuracy varied based upon the extreme event split method selected.

Based on the experimental results in Table I, Table II, and Table III, it can be seen that the Extreme Event Pipeline with SPOT split algorithm leads to comparable accuracy with baseline, in some cases even worse. From Figure 3 we notice that SPOT failed to distinguish the data properly, which leads to lower R^2 in all datasets. The descending rate displayed in Fig. 4 boxplot shows that SPOT even has the highest descending rate in Dublin and Magonia.

One possible explanation is, SPOT was developed for detecting the peak of time series data. This method naturally only detects the most extreme cases, which results in the total number of samples in extreme events being too small to meet the needs of training machine learning models. In this experiment, the extreme event model used is GBR (In Fig. 1 this is the upper Blue Box inside the Orange Box). Simply separating the extreme cases will not necessarily improve the effectiveness of the framework, and choosing the inappropriate split method will even limit the performance of the framework.

For the Percentile extreme event split framework, it achieved a significant accuracy improvement on R^2. In Table I, for horizon 2, R^2 was improved by 0.175, 0.031, 0.036, and 0.1, respectively, and 0.086 on average. When the horizon increases to 6, the average increment of R^2 was 0.31. As for horizon 12, Percentile achieved 0.079 and 0.140 on Well 13 and Well 19, however, less than zero in Well 16 and Well 28. It is worth mentioning that the percentile framework achieved the best performance in most testing cases for horizons 2 and 6 on four datasets in the EmeraldIsle site according to the R^2 and MAE metrics. By analyze Figure 2, the R^2 of Percentile framework drops faster when horizons are larger than 12 on the Well 16 and Well 28 datasets. We can conclude that for relatively small datasets, the percentile framework is a better choice for small horizons.

In Table II, Percentile failed to outperform Baseline on horizon 2. However, Percentile has an average increase of 0.028, 0.051, and 0.07 on the three datasets when the horizon is 6, 12, and 20 on the R^2 respectively. It is worth noting that for the Magonia dataset, Percentile achieved the best performance in all three metrics for horizons 12 and 20.

As for the Streamflow site, the percentile framework outperformed Baseline on the Mill dataset. For Mad and Shepard datasets, the percentile framework shows better R^2 than Baseline when the horizon is greater than 14.

Mask Matrix. In Table I, when the horizons are 2 and 6, the Mask achieved comparable results in R^2 and MAE and was outperformed by the Percentile framework. However, when the horizon is 12, the Mask achieved the best R^2 performance on all four datasets and the best MAE value on three out of four datasets. In addition, Mask is the only one that achieved greater than zero R^2 on all four data sets when the horizon is 12.

In Table II, Mask achieved the best performance in R^2 on all horizons for Dublin and Littlefield. For Magonia, Mask achieved the highest R^2 on horizon 2, outperforming baseline by 0.055, 0.087, and 0.133 on horizons 6, 12, and 20. From Figure 3 we can see that the Mask outperformed Baseline in all 3 datasets on all horizons, and has a lower descending rate than Baseline. Meanwhile, Mask outperformed Percentile on all horizons in Dublin and Littlefield.

In Table III Mask shows great performance on all 3 datasets. For the Mill site, Mask achieved the best R^2 on all horizons. It also has a lower RMSE and MAE when the horizon is larger than 6. On Mad and Shepard, Mask also achieved good results when the horizon is larger than 6 in R^2, RMSE, and MAE. From the results, we can conclude that our proposed mask framework has a lower descending rate in performance, which makes our method more suitable for large horizons. It is worth mentioning that the Streamflow site is collected at a sampling frequency of 1 hour. Therefore, Mask has more significance in practical applications on this data set.

Extreme Event Split VS Mask Matrix. Both extreme event split and mask matrix pipelines are promising for improving results when compared to the baseline model according to our experimental results.

However, if most of the time series data are normal events and there are very few extreme events the proposed framework may not outperform the original machine learning model. This is because the extreme event split pipeline will not have enough data for extreme event machine learning model training. The overall performance will drop. On the other hand, the mask matrix pipeline will have a mask matrix with mainly 0s. We have tested machine learning models with all 0s in the mask matrix and the accuracy will be very close to the baseline model.

Which pipeline should be selected is another question. Based on the experimental results in Table II and Table III, the mask matrix pipeline requires can outperform the extreme event split pipeline if abundant time series data are collected. However, if limited data is available, the extreme event split pipeline is better. This is because the mask matrix serves as hints for machine learning models to learn the difference between normal and extreme events from the historical data. The learning requires a lot of historical data. The extreme event split pipeline is a more straightforward method. A single machine learning is not competent to learn the differences between normal and extreme events. Therefore, this is handled by an extreme event labeling algorithm and two machine learning models.

VI. Conclusions

Time series data regression problems can be challenging if the data has extreme events (e.g., greater than 90 percentile or below 10 percentile). In this paper, we propose a time series data regression framework to tackle this problem. The framework includes two pipelines: extreme event split and mask matrix to help machine learning models learn the differences between normal events and extreme events. The experimental results show that both the pipelines are promising to improve a baseline machine learning model and decrease the accuracy descending rate when the horizon grows.

Acknowledgment

The data collected from Streamflow site was supported by the National Science Foundation under VT EPSCoR grant OIA-1556770 and the VT EPSCoR grant EPS-IIA 1330446.

References

[1] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research*, vol. 270, no. 2, pp. 654–669, 2018.

[2] F. Kratzert, D. Klotz, M. Herrnegger, A. K. Sampson, S. Hochreiter, and G. S. Nearing, "Toward improved predictions in ungauged basins: Exploiting the power of machine learning," *Water Resources Research*, vol. 55, no. 12, pp. 11 344–11 354, 2019.

[3] R. Wu, J. T. Painumkal, J. M. Volk, S. Liu, S. J. Louis, S. Tyler, S. M. Dascalu, and F. C. Harris, "Parameter estimation of nonlinear nitrate prediction model using genetic algorithm," in *2017 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2017, pp. 1893–1899.

[4] R. Eskandarpour and A. Khodaei, "Machine learning based power grid outage prediction in response to extreme events," *IEEE Transactions on Power Systems*, vol. 32, no. 4, pp. 3315–3316, 2016.

[5] M. L. Huang and C. Nguyen, "High quantile regression for extreme events," *Journal of statistical distributions and applications*, vol. 4, no. 1, pp. 1–20, 2017.

[6] L. I. Kuncheva, "Classifier ensembles for detecting concept change in streaming data: Overview and perspectives," in *2nd Workshop SUEMA*, vol. 2008, 2008, pp. 5–10.

[7] T. G. Dietterich, "Machine learning for sequential data: A review," in *Joint IAPR international workshops on statistical techniques in pattern recognition (SPR) and structural and syntactic pattern recognition (SSPR)*. Springer, 2002, pp. 15–30.

[8] F. Karim, S. Majumdar, H. Darabi, and S. Harford, "Multivariate lstm-fcns for time series classification," *Neural Networks*, vol. 116, pp. 237–245, 2019.

[9] X. Qiu, L. Zhang, Y. Ren, P. N. Suganthan, and G. Amaratunga, "Ensemble deep learning for regression and time series forecasting," in *2014 IEEE symposium on computational intelligence in ensemble learning (CIEL)*. IEEE, 2014, pp. 1–6.

[10] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the dots: Multivariate time series forecasting with graph neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 753–763.

[11] S. Siami-Namini, N. Tavakoli, and A. S. Namin, "A comparison of arima and lstm in forecasting time series," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2018, pp. 1394–1401.

[12] A. Sagheer and M. Kotb, "Unsupervised pre-training of a deep lstm-based stacked autoencoder for multivariate time series forecasting problems," *Scientific reports*, vol. 9, no. 1, pp. 1–16, 2019.

[13] P. T. Yamak, L. Yujian, and P. K. Gadosey, "A comparison between arima, lstm, and gru for time series forecasting," in *Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence*, 2019, pp. 49–55.

[14] U. Saini, R. Kumar, V. Jain, and M. Krishnajith, "Univariant time series forecasting of agriculture load by using lstm and gru rnns," in *2020 IEEE Students Conference on Engineering & Systems (SCES)*. IEEE, 2020, pp. 1–6.

[15] S. Rüping, "Svm kernels for time series analysis," Technical report, Tech. Rep., 2001.

[16] B. Kedem and K. Fokianos, *Regression models for time series analysis*. John Wiley & Sons, 2005, vol. 488.

[17] C. M. Hurvich and C.-L. Tsai, "Regression and time series model selection in small samples," *Biometrika*, vol. 76, no. 2, pp. 297–307, 1989.

[18] A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano, "A review on outlier/anomaly detection in time series data," *arXiv preprint arXiv:2002.04236*, 2020.

[19] A. Siffer, P.-A. Fouque, A. Termier, and C. Largouet, "Anomaly detection in streams with extreme value theory," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 1067–1075.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[21] P. Legendre, "Spatial autocorrelation: trouble or new paradigm?" *Ecology*, vol. 74, no. 6, pp. 1659–1673, 1993.

[22] S. D. Hamshaw, M. M. Dewoolkar, A. W. Schroth, B. C. Wemple, and D. M. Rizzo, "A new machine-learning approach for classifying hysteresis in suspended-sediment discharge relationships using high-frequency monitoring data," *Water Resources Research*, vol. 54, no. 6, pp. 4040–4058, 2018.

[23] Amossys-team, "Amossys-team/spot: Spot algorithm implementation (with variants)." [Online]. Available: https://github.com/Amossys-team/SPOT