VU VRIJE UNIVERSITEIT AMSTERDAM

VU Research Portal

A Framework for Imbalanced Time-Series Forecasting

Silvestrin, Luis P.; Pantiskas, Leonardos; Hoogendoorn, Mark

published in
Machine Learning, Optimization, and Data Science
2022

DOI (link to publisher)
10.1007/978-3-030-95467-3_19

document version
Publisher's PDF, also known as Version of record

document license
Article 25fa Dutch Copyright Act

Link to publication in VU Research Portal

citation for published version (APA)
Silvestrin, L. P., Pantiskas, L., & Hoogendoorn, M. (2022). A Framework for Imbalanced Time-Series Forecasting. In G. Nicosia, V. Ojha, E. La Malfa, G. La Malfa, G. Jansen, P. M. Pardalos, G. Giuffrida, & R. Umeton (Eds.), Machine Learning, Optimization, and Data Science: 7th International Conference, LOD 2021, Grasmere, UK, October 4–8, 2021, Revised Selected Papers, Part I (Vol. 1, pp. 250-264). (Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Vol. 13163 LNCS). Springer Science and Business Media Deutschland GmbH. https://doi.org/10.1007/978-3-030-95467-3_19

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

E-mail address:
vuresearchportal.ub@vu.nl

Download date: 28. Mar. 2024



A Framework for Imbalanced Time-Series Forecasting

Luis P. Silvestrin(✉), Leonardos Pantiskas, and Mark Hoogendoorn

Computer Science Department, Vrije Universiteit Amsterdam, Amsterdam, Netherlands
l.p.silvestrin@vu.nl

Abstract. Time-series forecasting plays an important role in many domains. Boosted by the advances in Deep Learning algorithms, it has for instance been used to predict wind power for eolic energy production, stock market fluctuations, or motor overheating. In some of these tasks, we are interested in predicting accurately some particular moments which often are underrepresented in the dataset, resulting in a problem known as *imbalanced regression*. In the literature, while recognized as a challenging problem, limited attention has been devoted on how to handle the problem in a practical setting. In this paper, we put forward a general approach to analyze time-series forecasting problems focusing on those underrepresented moments to reduce imbalances. Our approach has been developed based on a case study in a large industrial company, which we use to exemplify the approach.

Keywords: Imbalanced regression · Deep learning · Time-series forecasting · Multivariate time-series

1 Introduction

Due to the recent advances in artificial intelligence research, the task of time-series forecasting is being increasingly tackled with machine learning and deep learning techniques. There has been a large number of approaches suggested, ranging from relatively simple machine learning models [1] to a variety of deep learning models [18]. Those approaches have been utilized in a broad spectrum of forecasting tasks, such as wind power forecasting, stock market prediction and motor temperature prediction [18]. In the above examples of tasks, as well as in multiple other applied cases, some samples are more crucial from the point of view of the user and thus would require a more accurate prediction from a model compared to its average performance. At the same time, those data points may be scarce in the training data. Hence, if left unattended performance might be worse than average for that data, which is highly undesirable. This issue is characterized as imbalanced regression, and so far has been addressed with data pre-processing or ensemble model methods [3].

© Springer Nature Switzerland AG 2022
G. Nicosia et al. (Eds.): LOD 2021, LNCS 13163, pp. 250–264, 2022.
https://doi.org/10.1007/978-3-030-95467-3_19

Despite the existing methods to tackle imbalanced regression problems, it is still a non-trivial task for data scientists and machine learning practitioners to identify and solve them in real-life time-series forecasting contexts. In the effort of developing the best performing machine learning model and minimizing the error across all data points, some important artifacts in the data might be overlooked. Moreover, data sampling methods or ensemble model approaches [3,4] up until now focused on minimizing the prediction error in the underrepresented data samples and assume that the remaining data is negligible. That assumption is inaccurate, as in some applications, for example in stock market prediction, the cost of larger forecasting error in the more frequent cases could offset in the long run the potential benefit of a smaller error in a rare case. In order to tackle real-world applications, there is a need for a broader, balanced, flexible and iterative approach, honed through interaction with domain experts and integrating the latest research in predictive models.

In this paper, we propose such an approach that has been designed based on a case study in a large industrial company, targeted to forecast the temperature of a core component in a large production line. The approach involves three steps: first selecting a weight function which quantifies the sample importance; then applying one or more sampling methods to the data, and finally training and evaluating the model with and without sampling. In the last step, we also analyze the input importance learned by the model using SHAP [10] to gain insights into the effect of the imbalance. To exemplify our approach, we show how it is used for the aforementioned industrial task. We study the impact of choices in each of the steps, comparing different sampling techniques and deep learning models. In the end, we also combine the sampling with attention mechanisms [19] to extract insights into what is learned by a deep learning model. Furthermore, in order to verify that our conclusions hold across use cases and enable reproducibility, we apply the three main framework steps on an open industrial dataset dealing with the task of quality prediction of a mining process and present those experiment results as well.

2 Related Work

The advancements in data availability from a plethora of sources, the increasing computational capacity and the progress in artificial intelligence research have led to the usage of machine and deep learning models across a multitude of applications of time-series forecasting. Previous work [18] surveys several use-cases including wind power forecasting, stock price prediction, and estimation of remaining useful life of motors. Despite this widespread use of models, the majority of works present specific architectures for specific datasets, while works focusing on integrated frameworks in the sense of structured approaches to a more generalized problem are more scarce.

Although there has been an extensive amount of work in handling imbalanced datasets in classification tasks [3], regression with imbalanced data in the area of machine and deep learning has not been largely covered. In [4], Branco et al. study the effect of three proposed sampling methods on the predictive

performance of machine learning models. In an applied example, in [17], the objective is that high water flows are predicted in a timely and accurate manner, and the problem is addressed with various sampling and ensemble methods, with an artificial neural network as a base. In the domain of extreme event prediction, a framework [5] was proposed based on a new loss function for recurrent neural networks and focusing on univariate time-series. Other works propose new neural network topologies to predict extreme events in weather data [9] and the client demand at Uber [8]. Salinas et al. [15] propose a new probabilistic autoregressive model for time-series forecasting which is trained by over-sampling the data according to its rarity. These works focus primarily on the model performance related to the extreme events while our framework focuses on finding the best performance trade-off across extreme and common events. In addition, our framework is model agnostic, which makes it possible for practitioners to adopt it without having to change the underlying machine learning method they have selected for a use case for other reasons, such as interpretability.

3 Methodology

It is common in time-series forecasting that certain data samples have more importance than others, but are also underrepresented in the dataset, resulting in an imbalanced regression problem. In this section, we present a new approach for identifying and tackling this discrepancy with respect to imbalances in the context of regression tasks. This approach uses a weight function that quantifies the importance of each sample which is combined with under-sampling methods to create a more balanced dataset. The new sampled dataset is then evaluated first visually, by making density plots of the data and then numerically, by using it for training and testing a predictive model.

3.1 Steps for Identifying and Treating Imbalanced Regression

We propose a set of general steps for approaching the imbalance on time-series forecasting problems, which has been defined based on experiences we have collected in applying machine learning in a large-scale industrial company. It consists of three steps illustrated in Fig. 1. The first step is to select or define a weight function w_i to quantify the sample importance, which allows us to identify and compare the different regions of interest in the data. The second step consists of selecting one or more sampling methods based on the weight function, applying them to the data, and comparing the resulting distribution of w_i against the original data using density plots. Finally, in the third step, a predictive model is trained and evaluated using both the sampled and the original versions of the dataset. A feedback loop can take the user from Step 3 back to Step 2 if the current combination of the selected sampling methods and models does not provide a satisfying performance after evaluation. If while applying the framework the user finds out that the weight function is not suitable, then a second feedback loop can take him back to Step 1 where it can be re-modeled. We provide more details about each step in the following sections.

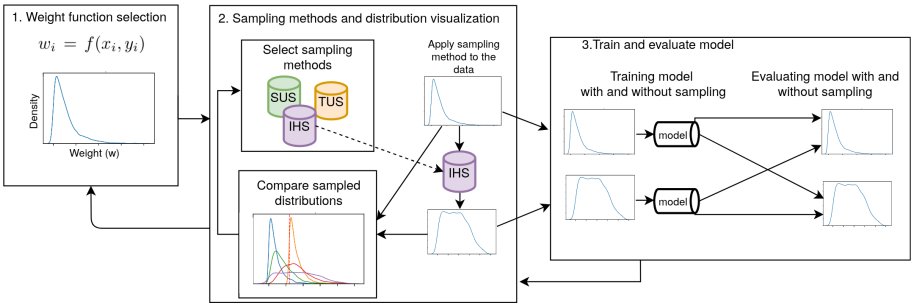


Fig. 1. Flowchart of the framework showing the three steps and how they interact with each other.

Step 1: Weight Function Definition. In the example of forecasting the temperature of a motor, there can be several days where the temperature is stable, with only small fluctuations, and only a few days where the temperature increases or decreases largely. Let us assume a use-case where the user is interested in building a model to predict accurately these rarer moments when the temperature changes more than usual. In such an example, the daily temperature variation can be computed as a function of the data, which we refer to in our framework as the weight function. In addition, we say that the data points mapped to a high variation by the weight function belong to a region of interest.

In general, the weight function can depend either only on the target variable (or a transformation of it), only on a subset of the input variables (e.g. to signify working points of interest), or on a combination of input and output variables, to express complex regions of interest. It can be written as $w_i = f(x_i, y_i)$, where x_i and y_i refer to the input (a vector for multivariate input) and the prediction target, respectively.

Equation 1 gives an example of such a function for the target variation in the context of time-series forecasting. The weight w_t models the variation of the forecast target y over the forecast horizon Δ at time step t.

$$w_t = |y(t + \Delta) - y(t)| \tag{1}$$

Step 2: Application of Sampling Method. At this step, a sampling method is applied to the dataset and its effects are analyzed. The sampling is based on the weight function previously selected and the relative proportions that the user wants to keep for the different regions of interest. We identify three scenarios for this step and we propose an under-sampling method for each one of them.

- **Threshold under-sampling:** when the user identifies which region of the weight function is not important for the forecast task and can be removed;
- **Stochastic under-sampling:** when some regions are more important than others, but none can be entirely discarded;

– **Inverse histogram under-sampling:** when all regions are equally important and the user wants to have a balanced distribution of data over all of them.

Threshold under-sampling (TUS) consists of removing all samples that lie below a given threshold of the weight function, and all the remaining samples have the same chance of being selected. This method is suitable for the cases where the user knows exactly what is the region of interest to be able to select the best threshold, and it assumes that the samples below the threshold aren't interesting to the prediction task. Equation 2 expresses the unnormalized probability of sampling data point t given its weight w_t.

$$\mathrm{TUS}(w_t) = \begin{cases} 1, & \text{if } w_t > \tau \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

$$\mathrm{SUS}(w_t) = w_t^f \tag{3}$$

$$\mathrm{IHS}(w_t) = h^{-1}(w_t) \tag{4}$$

Stochastic under-sampling (SUS) uses the weight w_t computed for each data point as the probability of sampling it. Different from the TUS, SUS allows every sample, even the ones with lower weight values, to be sampled to avoid the creation of a new imbalance against those samples. Equation 3 models the relative probability of sampling a window from the dataset at time t by using SUS. The factor f is used to increase (or decrease) the effect of the weights, thus emphasizing the more interesting moments which might be underrepresented in the data.

Inverse histogram under-sampling (IHS) is an automatic method to obtain a sample where data is approximately uniformly distributed across the selected weight function w_t. It consists of building a histogram of the values of w_t in the dataset and taking the inverse of the frequency of each value as the chance of sampling it. It ensures that each w_t will be under-sampled proportionally to its original frequency, so the most common values will have lower chance, while the rarer values will have higher chance. In Eq. 4 we can see a formalization of the method, where $h(w_t)$ represents the frequency of w_t in the data histogram.

A good approach to gain insight into the data and the result of the sampling method is to compare the density plot of the weight function with and without using the sampling. Such a plot can show the regions which are over or underrepresented in the data, and can also give insights about how to tune the parameters of the selected method or which method should be selected.

In some real-life cases, it might not be easy to infer directly which of the three scenarios fits the problem better. In those cases, a subset of these methods can be selected for the next step, where we provide a heuristic to select the final method.

Step 3: Predictive Model Training and Evaluation. Finally, at this step, we can assess how much a predictive model improves by using the selected sampling methods. For that, we train and evaluate the model with and without using the sampling method on a separate evaluation set, so we end up with different combinations of training and evaluation sets which we will use to contrast the obtained evaluation errors and determine if the model benefits from the sampling. For the cases where the goal is to have a model that performs well on the samples of higher weight without sacrificing the performance on the rest of the data, we propose a heuristic for selecting the final sampling method to train the predictive model based on the results of the different evaluation sets. It is defined as:

1. For each sampling method, sample a training set and train a model with it;
2. For each sampling method, sample a separate evaluation set and evaluate the trained model on it;
3. Make a list of highest error over all the evaluation sets of each trained model to get an upper bound on its RMSE error;
4. Select the model with the lowest error in the list.

Next to studying the impact of the sampling on the performance of the models, we also propose to study how the models themselves change by using SHAP [10], which is a model agnostic technique. SHAP gives the relative importance of each input feature to the output of the model which can be compared when the model is trained with and without the sampling.

In addition, we take advantage of deep learning models with attention mechanisms [19] to gain extra insights into what is learned. As an example of an attention-based model, TACN [12] is a deep neural network model that provides the importance of the input time-series across time steps through an attention mechanism. The change of the patterns shown by the mechanism also provides insights into the sampling effects on the model.

4 Experimental Setup

To give a real-life example of our approach, in this section we present a case to evaluate it based on a motor temperature prediction dataset. We also explain the techniques used at each step of the experiments and why they were chosen.

4.1 Motor Temperature Dataset

The dataset used in this experiment is made of sensor measurements extracted from a steel processing conveyor belt. The prediction target is the temperature of a bridle motor, which should be forecasted 5 min in advance to allow the operators to take preventive actions before a possible overheat. The rest of the data consist of properties of the steel strip (i.e. width, thickness, and yield), the speed of the line, the tension applied by the bridles, the current temperature

measurements of the motor, among others. The sensors are sampled every 10 s, and there are in total about 2 million samples.

In this dataset, we identify the temperature variation as a special property regarding the prediction target. We analyze the dataset based on this property and follow the steps of our framework: selecting a weight function, then selecting the sampling methods, and visualizing the sampling result.

4.2 Instantiation of the Framework

Here we describe the choices made at each step of the framework for analyzing the imbalance of the temperature variation.

Step 1 - Temperature Variation Weight Function. The temperature variation is an important property to this forecasting task since the predictive model must predict accurately when the temperature will rise. Even if, on average, the model has a satisfying performance, it may still be inaccurate when predicting higher variation if the dataset is imbalanced. So for Step 1 of the framework, we select the temperature variation as the weight function, which is modeled by Eq. 1, using Δ as 30 time steps (5 min), which is the forecast horizon.

Step 2 - Sampling Method Choice. For Step 2, we experiment with three sampling methods: SUS with factor 1, SUS with factor 3, and IHS. Each one under-samples a different amount of low temperature variation data, creating a different balance, as shown by Fig. 2. SUS with factors 1 and 3 are chosen to compare the effect of the factor in the proportion of data samples with low and high variation. For the IHS method, we use the Freedman-Diaconis estimator [6] to compute the bin width of the histogram. 10.000 training data samples are extracted using each method.

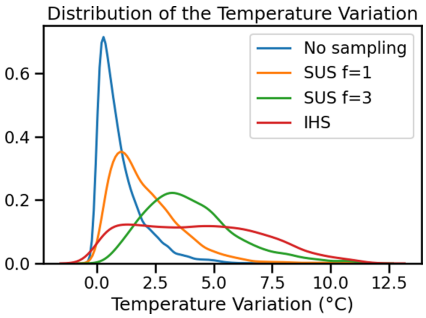


Fig. 2. Comparison of different sampling methods using the temperature variation as weight function.

Step 3 - Predictive Model Choices. In our experiments, we choose a multilayer perceptron [14] as a deep neural network baseline which has been used in time-series forecasting [1] and three deep neural networks specialized in temporal data. These specialized architectures are the long short-term memory (LSTM) [7], a popular recurrent neural network, the temporal convolutional network (TCN) [2], a sequence-to-sequence model which has shown promise when trained on a large amount of data [16] and the temporal attention convolutional network (TACN) [12].

The TACN is an architecture that combines a TCN with an attention mechanism [19] to achieve interpretable and accurate forecasting. The per-instance interpretability comes in the form of a vector, equal to the input window size, which shows the importance of each input step to the forecasting output. The higher the value of the vector at a specific step, the higher the contribution of the input value at that step to the final output. By scaling the vector to the 0–1 range, we can estimate the relative importance among the input steps. Although this vector is produced per instance, we can draw conclusions about the generic learned behavior of the model by collecting and analyzing the vectors from a large number of instances.

For data pre-processing, we extract a window of 5 min (or 30 time steps) for each sample, which is the input for the TCN, LSTM, and TACN models. For the MLP model, we extract basic features of each sensor such as the mean, standard deviation, minimal and maximal values for each window. We also keep the last time step as an additional feature and for later analysis of the temperature variation case. All the models are evaluated using the root-mean-square error metric (RMSE).

5 Results

In this section, we describe the results obtained after applying our framework starting from Step 2. Step 1 is already defined in Sect. 4. In the last subsection, we also present the results obtained from an additional experiment conducted using an open time-series forecasting dataset.

5.1 Step 2 - Comparison of the Sampling Methods

Figure 2 shows the variation distribution after applying the sampling methods. Without any sampling, the dataset has a strong bias towards samples with variation close to zero, meaning that the temperature is stable, or varies very slowly most of the time. SUS with factor 3 give more emphasis to samples with higher variation, while significantly reducing the number of samples with lower variation. The sampling using SUS with factor 1, on the other hand, is more conservative and preserves a considerable amount of samples with low variation. Finally, IHS gives the best balance across all values and is the one that gives the highest proportion of samples in the extreme of the temperature variation spectrum (above 6 degrees in Fig. 2).

5.2 Step 3 - Analysis of the Results

The results of the four models trained and tested with the selected sampling methods based on temperature variation can be seen in Table 1, with the lower error per evaluation set highlighted. The effect of the imbalance of the original data distribution is clearly shown in the "None" rows, where the models were trained without sampling. For those lines, the RMSE is much higher in the SUS 3 column, where there is a smaller number of samples of low variation, suggesting that the models are biased towards low variation samples if trained without sampling methods.

On the other hand, these results show that there is a trade-off between favoring samples with and without temperature variation. Models trained with a more aggressive kind of sampling, such as SUS with factor 3, have a much higher error when evaluated on the unsampled data than the models trained with SUS factor 1, for example. This can be explained by the density difference between samples with low variation (below 2.5 degrees in Fig. 2), the same samples that are more common in the "no sampling" dataset. With our approach, this trade-off which exhibits non-linear behavior can be estimated, taking into account the end-user preferences, and it can lead to a re-evaluation of the sampling method in Step 2. Also, together with these metrics, using insights about the model as described later in this subsection can indicate the sampling method that leads

Table 1. RMSE results for different sampling methods based on temperature variation. "None" means that the model was trained or evaluated without using a sampling method.

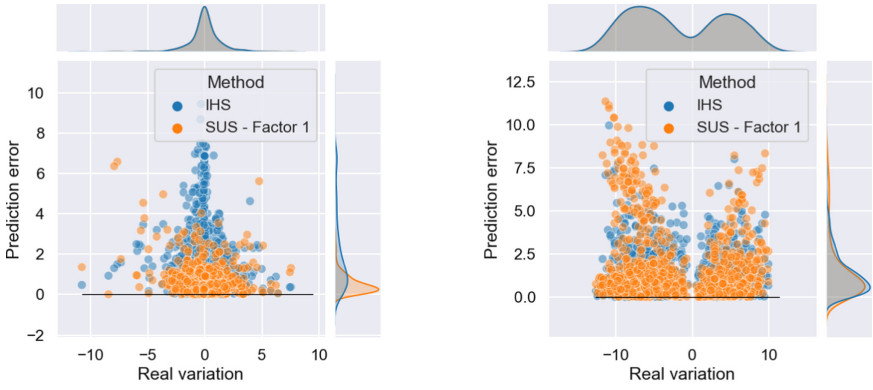
Model	Trained on	Evaluated on			
		None	SUS - 1	SUS - 3	IHS
MLP	None	1.401 ± 0.181	2.270 ± 0.177	3.611 ± 0.322	2.984 ± 0.241
	SUS - 1	1.704 ± 0.289	2.085 ± 0.239	2.886 ± 0.269	2.495 ± 0.234
	SUS - 3	4.150 ± 0.635	3.089 ± 0.390	2.430 ± 0.305	2.836 ± 0.32
	IHS	3.066 ± 0.401	2.728 ± 0.228	2.539 ± 0.248	2.663 ± 0.217
LSTM	None	1.032 ± 0.091	1.857 ± 0.112	3.275 ± 0.129	3.275 ± 0.13
	SUS - 1	1.283 ± 0.153	1.469 ± 0.123	2.769 ± 0.130	2.731 ± 0.094
	SUS - 3	3.595 ± 0.268	2.549 ± 0.128	**1.464 ± 0.24**	2.415 ± 0.095
	IHS	2.728 ± 0.174	2.131 ± 0.093	1.863 ± 0.106	2.27 ± 0.093
TCN	None	**0.871 ± 0.021**	1.684 ± 0.037	3.060 ± 0.068	3.142 ± 0.05
	SUS - 1	1.007 ± 0.079	**1.462 ± 0.041**	2.686 ± 0.066	2.703 ± 0.063
	SUS - 3	3.41 ± 0.213	2.4 ± 0.091	1.592 ± 0.124	2.283 ± 0.008
	IHS	2.579 ± 0.231	2.016 ± 0.091	1.845 ± 0.062	**2.145 ± 0.039**
TACN	None	1.171 ± 0.016	2.637 ± 0.051	5.167 ± 0.1	5.064 ± 0.101
	SUS - 1	1.334 ± 0.242	2.077 ± 0.355	3.878 ± 0.675	3.183 ± 0.694
	SUS - 3	3.802 ± 0.538	2.786 ± 0.428	2.36 ± 0.758	2.883 ± 0.547
	IHS	3.093 ± 0.918	2.424 ± 0.608	2.430 ± 0.748	2.752 ± 0.623

Table 2. The maximum error obtained by the TCN model over all the evaluation sets. Each line corresponds to a training set obtained from a different sampling method.

Trained on	Max. error	Measured on
No sampling	3.142 ± 0.05	IHS
SUS - 1	2.703 ± 0.063	IHS
SUS - 3	3.41 ± 0.213	No sampling
IHS	2.579 ± 0.231	No sampling

to the most encompassing, generalizable patterns learned by the models, thus creating a balance for the performance across data samples.

Since the results show that the TCN achieves a relatively lower error in all the evaluation sets, we select it as the best model and follow the heuristic described in Sect. 3.1. Table 2 shows the maximum error obtained by it over all the evaluation samples. The two lowest RMSE values reported in that table are from the TCN model trained with SUS factor 1 and IHS, and the evaluation sets where they have the highest error are *No sampling* and IHS. By comparing the performance of the TCN trained with both methods on the evaluation set without sampling, we can clearly see that the model trained with SUS with factor 1 has a lower spreading of the error (Fig. 3a). On the other hand, the same comparison on the evaluation set with higher variation (Fig. 3b) shows that both models have similar error spreading, and the small advantage of using IHS, in this case, does not compensate for the increase in error in the low variation samples. Therefore, we can conclude that SUS with factor 1 is the sampling method with the best performance across samples with low and high temperature variation.



(a) (b)

Fig. 3. (a) Prediction error for TCN model trained using IHS and SUS with factor 1 on "no sampling" evaluation set and (b) on SUS with factor 3 evaluation set.

Imbalance Effects in DL Models. In the third step of the framework, we also verify how the imbalances affect the performance and learned patterns of the predictive models, when they are trained on data with sampling versus unsampled data. To do that, we focus on the temperature variation property of the dataset, and we measure the SHAP values of the MLP model, as well as the attention importance values of the TACN.

To assess how the sampling methods influence the MLP model, we extract its SHAP values using the SUS 3 evaluation set. We compare both the MLP trained with SUS 3 and without sampling. Figure 4 shows that both models rely mostly on the last temperature measurement to make the forecast. This could be explained by the fact that the last temperature is relatively close to the predicted temperature, even when there is high variation. One hypothesis for such fact is that the MLP does not handle the time dependency of the inputs and, thus has a disadvantage in comparison to other temporal models such as the TCN or the LSTM.

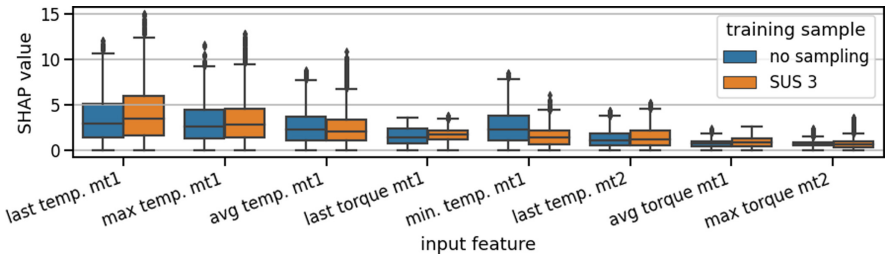


Fig. 4. Comparison of the absolute SHAP values estimated for the MLP model trained with SUS factor 3 and without sampling. The values are computed based on inputs from the evaluation set sampled using SUS factor 3. The input features are computed over a window of 5 min. *mt1* and *mt2* correspond to the first and second motors of the bridle set.

To gain insights about the differences in the behavior of the trained TACN models using the interpretability mechanism, we run inference on the SUS with factor 3 evaluation set for the models trained on (a) unsampled data and (b) on the SUS factor 3 train set, and we study the resulting attention pattern variation.

To quantify this variation, we enumerate for both models the unique learned attention values for each input time step across all test samples, rounded to the second decimal, and present the results on Fig. 5. For the model trained on unsampled data, the unique values for each position are at most 3, while for the SUS model they are between 30 and 50. The above observations lead us to the following conclusions: The model trained on the unsampled data has learned a high reliance on the last value and a limited number of patterns, which serves well in minimizing the error for the majority of the samples but results in low performance on the large variation samples. In contrast, the model trained on

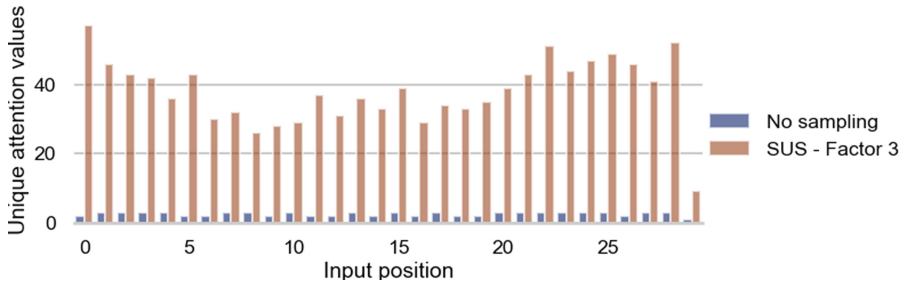


Fig. 5. Unique attention values per input step from sample TACN models trained on data with (a) no sampling and (b) SUS with factor 3

the SUS data is forced to learn a larger variety of patterns to accommodate for this target variation.

5.3 Additional Results from Mining Process Dataset

To confirm our conclusions and enable reproducibility of our methods, we apply our suggested framework on an open dataset from another real-world industrial application [11], which is the quality prediction of the output of a mining process. Specifically, the goal is to predict the impurity of the final ore product, which is measured as the percentage of silica contained in it. It is reasonable to assume that the process engineers are less interested in the majority of the cases where this percentage is low enough for the quality to be acceptable and more interested in the more rare cases, where the percentage is higher and may cross a threshold that requires the final product to be discarded. Following our framework, the steps are as follows:

Step 1 - Weight Function. Given the formulation of this task, it is clear that the prediction output is the most important property and is thus selected as the weight function.

Step 2 - Sampling Method Choice. Following the example of the motor temperature use case, we experiment with three sampling methods: SUS with factor 1, SUS with factor 3, and IHS.

Step 3 - Predictive Model Choices. Given the time series nature of this problem and its input variables, we elect to experiment with the LSTM and TCN models.

The results from our experiments with this task can be seen in Table 3 and the maximum RMSE across all evaluation sets for each model and sampling method combination in Table 4. We observe that using an appropriate sampling method in combination with the weight function reduces the error for the rarer and most important cases (high silica percentage). Moreover, the heuristic of the third framework step helps select the most appropriate sampling method for each forecasting model (SUS with factor 1 for the LSTM and IHS for the TCN). The code used to obtain these results is accessible at [13].

Table 3. RMSE results for different sampling methods based on silica percentage. "None" means that the model was trained or evaluated without using a sampling method.

	Trained on	Evaluated on			
		None	SUS - 1	SUS - 3	IHS
LSTM	None	0.620 ± 0.016	0.528 ± 0.048	0.675 ± 0.052	0.547 ± 0.048
	SUS - 1	0.594 ± 0.031	0.634 ± 0.016	0.600 ± 0.039	0.563 ± 0.036
	SUS - 3	0.731 ± 0.028	0.621 ± 0.022	0.633 ± 0.012	0.609 ± 0.031
	IHS	0.638 ± 0.032	0.587 ± 0.031	0.571 ± 0.045	0.643 ± 0.018
TCN	None	0.570 ± 0.009	0.533 ± 0.023	0.705 ± 0.031	0.578 ± 0.027
	SUS - 1	0.603 ± 0.015	0.586 ± 0.010	0.644 ± 0.026	0.606 ± 0.023
	SUS - 3	0.690 ± 0.022	0.580 ± 0.015	0.592 ± 0.009	0.561 ± 0.020
	IHS	0.617 ± 0.018	0.538 ± 0.019	0.566 ± 0.021	0.599 ± 0.013

Table 4. Maximum RMSE error across all evaluation sets for each model and sampling method.

	Trained on	Max RMSE
LSTM	None	0.675 ± 0.052
	SUS - 1	**0.634 ± 0.016**
	SUS - 3	0.731 ± 0.028
	IHS	0.643 ± 0.018
TCN	None	0.705 ± 0.031
	SUS - 1	0.644 ± 0.026
	SUS - 3	0.690 ± 0.022
	IHS	**0.617 ± 0.018**

6 Conclusion

We presented a framework to analyze imbalanced time-series forecasting problems and to train and evaluate ML models taking into account important properties. To our knowledge, this is the first framework that provides clear steps to help practitioners to select and compare different sampling methods and predictive models for such problems. It is put into practice to forecast the temperature of a motor in a steel processing conveyor belt, based on data extracted from a real-world industrial process and validated in cooperation with domain experts. The problem analysis is made through the lens of the temperature variation property. We study the dataset using three different sampling methods and train four different DL models to evaluate and compare the effectiveness of each combination of sampling and model. We also show the imbalance of the temperature variation and how it changes the models' predictions when they are trained with different proportions of samples with high temperature variation. Finally, we use

SHAP values and the TACN model's attention mechanism to show the effect of low temperature variation in the dataset on the forecast models, inducing them to rely mostly on the last observed temperature. For reproducibility purposes and verification of our conclusions, we also apply the framework to an open industrial dataset regarding the quality prediction of the output of a mining process, experimenting with three sampling methods and two DL models, and we present the results from this use case as well.

As future work, our framework could be put into practice to analyze new time-series prediction tasks, combining with more sampling techniques. The use of the input x in the weight function can be further explored to make the framework suitable for an even wider range of tasks. In addition, our results point out a possible relationship between the prediction error and the distribution of the training data which might be worth investigating.

Acknowledgements. This work has been conducted as part of the Just in Time Maintenance project funded by the European Fund for Regional Development. We also thank Tata Steel Europe for providing the data and technical expertise required for our experiments.

References

1. Ahmed, N.K., Atiya, A.F., Gayar, N.E., El-Shishiny, H.: An empirical comparison of machine learning models for time series forecasting. Econometric Rev. **29**(5–6), 594–621 (2010)
2. Bai, S., Kolter, J.Z., Koltun, V.: An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv:1803.01271 [cs] (2018)
3. Branco, P., Torgo, L., Ribeiro, R.P.: A survey of predictive modeling on imbalanced domains. ACM Comput. Surveys **49**(2), 1–50 (2016)
4. Branco, P., Torgo, L., Ribeiro, R.P.: Pre-processing approaches for imbalanced distributions in regression. Neurocomputing **343**, 76–99 (2019). https://doi.org/10.1016/j.neucom.2018.11.100
5. Ding, D., Zhang, M., Pan, X., Yang, M., He, X.: Modeling extreme events in time series prediction. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1114–1122. KDD 2019. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3292500.3330896
6. Freedman, D., Diaconis, P.: On the histogram as a density estimator: L 2 theory. Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete **57**(4), 453–476 (1981)
7. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
8. Laptev, N., Yosinski, J., Li, L., Smyl, S.: Time-series extreme event forecasting with neural networks at Uber. In: 2017 International Conference on Machine Learning Time Series Workshop (2017)
9. Liu, Y., et al.: Application of deep convolutional neural networks for detecting extreme weather in climate datasets. CoRR abs/1605.01156 (2016). http://arxiv.org/abs/1605.01156

10. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 30, pp. 4765–4774. Curran Associates, Inc. (2017)
11. Magalhães, E.: Quality prediction in a mining process, https://www.kaggle.com/edumagalhaes/quality-prediction-in-a-mining-process. Accessed 19 July 2021
12. Pantiskas, L., Verstoep, K., Bal, H.: Interpretable multivariate time series forecasting with temporal attention convolutional neural networks. In: 2020 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1687–1694 (2020). https://doi.org/10.1109/SSCI47803.2020.9308570
13. Pantiskas, L., Silvestrin, L.P.: Open code of the experiment using the mining process dataset (2021). https://gitlab.com/lpsilvestrin/imbalanced-time-series-forecast. Accessed 20 July 2021
14. Rosenblatt, F.: The perceptron: a probabilistic model for information storage and organization in the brain. Psychol. Rev. **65**(6), 386 (1958)
15. Salinas, D., Flunkert, V., Gasthaus, J., Januschowski, T.: DeepAR: probabilistic forecasting with autoregressive recurrent networks. Int. J. Forecasting **36**(3), 1181–1191 (2020). https://doi.org/10.1016/j.ijforecast.2019.07.001. https://www.sciencedirect.com/science/article/pii/S0169207019301888
16. Silvestrin, L.P., Hoogendoorn, M., Koole, G.: A comparative study of state-of-the-art machine learning algorithms for predictive maintenance. In: 2019 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 760–767. IEEE (2019)
17. Snieder, E., Abogadil, K., Khan, U.T.: Resampling and ensemble techniques for improving ANN-based high streamflow forecast accuracy. Hydrol. Earth Syst. Sci. Discuss. **2020**, 1–35 (2020)
18. Torres, J.F., Hadjout, D., Sebaa, A., Martínez-Álvarez, F., Troncoso, A.: Deep learning for time series forecasting: a survey. Big Data **9**(1), 3–21 (2021). https://doi.org/10.1089/big.2020.0159
19. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems Systems (NIPS), pp. 5998–6008 (2017)