ISSN 1995-0802, Lobachevskii Journal of Mathematics, 2023, Vol. 44, No. 8, pp. 3076–3090. © Pleiades Publishing, Ltd., 2023.

Exploring Hierarchical Forecasting of Data Popularity in High-Energy Physics Experiments

M. A. Grigorieva[1*], N. N. Popova[1], D. A. Vartanov[1***], and M. V. Shubin[1****]**

(Submitted by V. V. Voevodin)

[1] *Moscow State University, Moscow, 119991 Russia*

Received March 19, 2023; revised March 30, 2023; accepted April 10, 2023

Abstract—In high-energy physics, the current large-scale distributed computing environments are responsible for processing and analyzing vast amounts of data. Recently, CERN's total data storage reached an impressive 1 exabyte. This development has presented new challenges for data and workload management systems, which must ensure that resources are balanced and that data is evenly distributed across hundreds of computing centers in a storage-saving environment. To achieve this goal, the demand for data must take into account data popularity. This research explored two approaches to predicting data popularity: regression models (LSTM, Facebook Prophet) for predictive analysis of the popularity of groups of datasets, and classification models (LSTM, FCN, MLP, Logistic Regression, AdaBoost, CATBoost, XGBoost) for predicting the popularity of individual datasets.

DOI: 10.1134/S1995080223080206

Keywords and phrases: *data popularity, high-energy physics, distributed computing, machine learning.*

1. INTRODUCTION

As distributed computing systems continue to grow, understanding the popularity of data has become a critical aspect of analytical research. For the purposes of this analysis, data popularity refers to the number of times a particular dataset is accessed within a given time period. This metric is essential for managing workloads across distributed centers, particularly for determining data replication policies. In the field of high-energy physics (HEP), these policies are developed based on data format and the particular physics process being studied. The analysis of popularity is particularly important for physics analysis tasks, where the focus is on physics data most frequently utilized by users [1].

For example, the ATLAS experiment at the LHC utilizes the Rucio distributed data management system [2], which employs dynamic replication algorithms to optimize data processing during periods of high demand. These algorithms create temporary copies of frequently accessed datasets that have a limited lifespan (typically 1–2 weeks). By analyzing data popularity, these dynamic replication methods enable more efficient and effective data management. Such algorithms also complement existing data replication rules to ensure that the system performs optimally under various scenarios.

It is important to understand that the data processing pipelines in HEP can be categorized into two types: central production and user analysis.

- *Central production* refers to a series of data transformations that convert raw data into the derived format used for physics analysis. Given its reliance on batch processing of massive amounts of data, these pipelines are centrally planned and managed.

*E-mail: magsend@gmail.com
**E-mail: popova@cs.msu.ru
***E-mail: dima.vartan@gmail.com
****E-mail: mih.shub@gmail.com

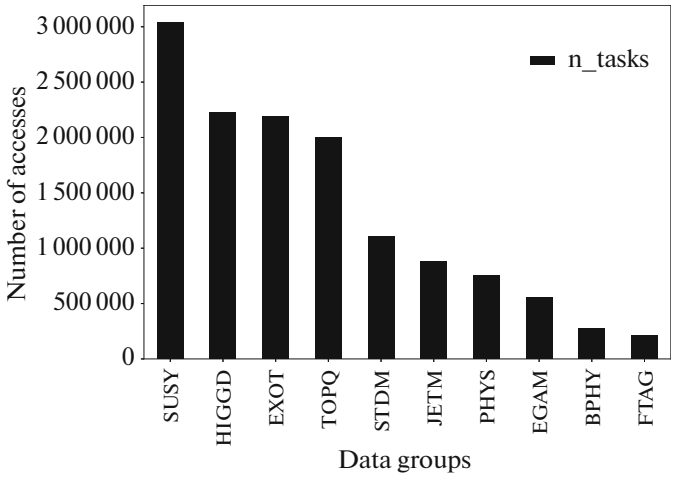

Fig. 1. The popularity of groups of datasets.

- On the other hand, *user analysis* is based on derived datasets and is conducted by researchers who may access the data in unpredictable patterns.

Thus, anticipating user analysis demands becomes crucial in a distributed environment. Accurately predicting workload increases can help prepare for spikes in demand by creating additional copies of expected high-demand datasets. Forecasts of dataset popularity can also aid in the development of deletion policies, as datasets that are not regularly accessed and aren't expected to be in the near future can be marked for removal.

In this research, we are focused on the development of a hierarchical data popularity forecast system consisting of two layers: the first layer implies the analysis and prediction of the popularity of classes of datasets, and the second layer—the prediction of the popularity of individual datasets. The ultimate target for popularity prediction is of course the dataset, as it is the smallest unit of data replication. However, layered forecasting structure was chosen based on the assumption that each class of datasets has various access patterns. Instead of analyzing all available datasets, it is easier to find out which groups of datasets will be most in demand in the near future first. And, then, search for the most popular datasets within each of these groups.

2. DATASETS POPULARITY STATISTICAL RESEARCH

We analyzed a fragment of data from one of the largest HEP experiments at the LHC. The data sample has about one million datasets[1)] that have been accessed at least once during the last 5 years.

Figure 1 illustrates the categories of datasets[2)] that are accessed most frequently. The number of user analysis tasks, or dataset accesses, is represented on the vertical axis and the data groups are shown on the horizontal axis.

Moving on to Fig. 2, it showcases the distribution of dataset access frequency over the past five years. Approximately 50% of all datasets were accessed less than five times per year, while 15% were accessed between six to ten times, and $\sim 27\%$ were accessed up to 100 times. Only 2% of all datasets were accessed more than 100 times, which we consider to be the most in-demand datasets. However, these highly accessed datasets only make up 30% of the total share over five years. It's important to note that when considering data popularity, the usage period and frequency of access should also be taken into account. This means that the more local (for a shorter period of time) share of popular data may actually be smaller.

And Fig. 3 shows the distribution of datasets by usage periods. About 18% of datasets were accessed only within a week during the last 5 years, and about 7%—within a month. Let's consider this data to be

[1)]Dataset—logical group of files.

[2)]Groups of datasets are represented by a description of the physical process to which the dataset belongs, i.e., SUSY—supersymmetry, HIGGD—Higgs boson, TOPQ—top quark, STDM—Standard Model, etc.

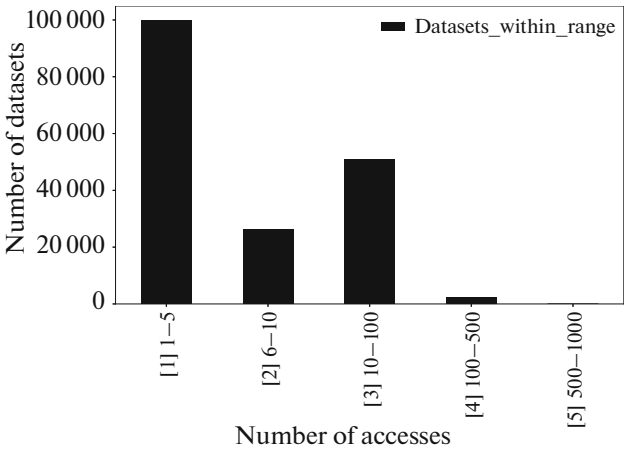

Fig. 2. Distribution of the number of datasets by the number of accesses during the last 5 years.

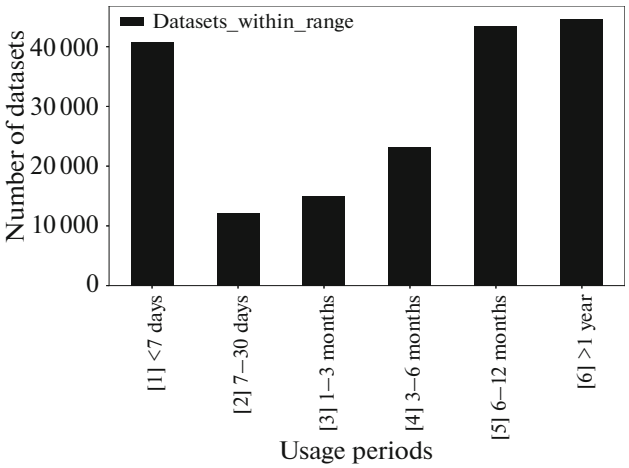

Fig. 3. Distribution of the number of datasets by usage periods over the last 5 years.

short-lived. Medium-lived datasets are those that have been accessed within 1–6 months, that account for almost 25% of all data. And long-lived data, which is accessed periodically for six months or more, is about 44%.

These statistical findings lead us to the idea of the hierarchical forecast system, and the analysis of the groups of data separately from each other.

3. EXISTING APPROACHES TO DATA POPULARITY PREDICTIONS IN HIGH ENERGY PHYSICS

There have already been several attempts, in some HEP experiments, to develop methods for data popularity forecasts [3].

In the ATLAS experiment an idea was to aggregate the accesses for each dataset per week and feed this to the artificial neural network (ANN) to get an estimate for the next week. The benchmarks showed that the neural network predicts the right trend in 77% of the cases [4].

The CMS experiment utilized various features, including dataset name and number of files, to analyze access patterns and predict dataset popularity using machine learning algorithms such as RandomForest, LinearSVC, SGDClassifier, VW, and xgboost. The model's accuracy was measured in terms of True Positive (TP) and True Negative (TN) rates, while the cost of false positives (FP) and false negatives (FN) was also considered. The SGDClassifier algorithm demonstrated the best performance on a certain group of data, with an error rate of less than 10% for TP/TN rates in most weeks [5].

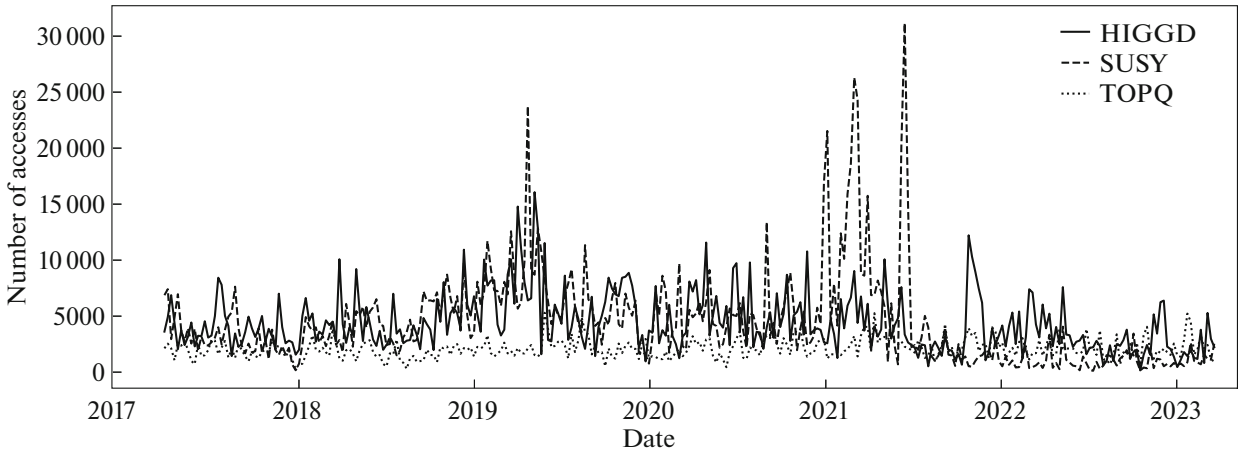

Fig. 4. The number of accesses for the different groups of datasets.

However, none of the predictive methods developed to make decisions about data replication, data placement, and other data management tools were eventually adopted. This shows that we have space to explore.

In this study we decided not to use complex data models consisting of a large number of dataset parameters. Instead, the simplest way to represent datasets was chosen—a time series consisting of weekly counts of requests for data.

4. DATASETS AS TIME-SERIES SEQUENCES

In this research we proceed with the assumption that a group of datasets or a specific dataset can be represented as a time series or sequence of data accesses over a certain period of time. For example, Table 1 shows a sequence of weeks for a group of HIGGD datasets from test data samples.

Time series for the groups of datasets, shown in Fig. 4, demonstrates the access patterns for the popular groups: HIGGD, SUSY, TOPQ.

In total, it appears that the data is accessed continuously throughout the time interval. Moreover, each group has its own access pattern and, consequently, should be analyzed separately.

Figure 5 represents the time series for the single dataset, and it is discrete, with a large number of random omissions.

In this study, we evaluate two predictive approaches:

1. Regression predictions for groups of datasets.

2. Classification for individual datasets.

In the first case, each dataset group was represented as a single time series—the aggregated sequence of the accesses to all datasets in a group used to predict the number of data accesses in the coming weeks. Since time series were involved, several regression models were available for forecasting.

Table 1. Example of access sequence for HIGGD dataset group

Week	01-02-23	01-09-23	01-16-23	01-23-23	01-30-23	02-06-23	02-13-23	02-20-23	02-27-23
Number of accesses	363	986	1698	1341	1870	1187	1083	3684	720

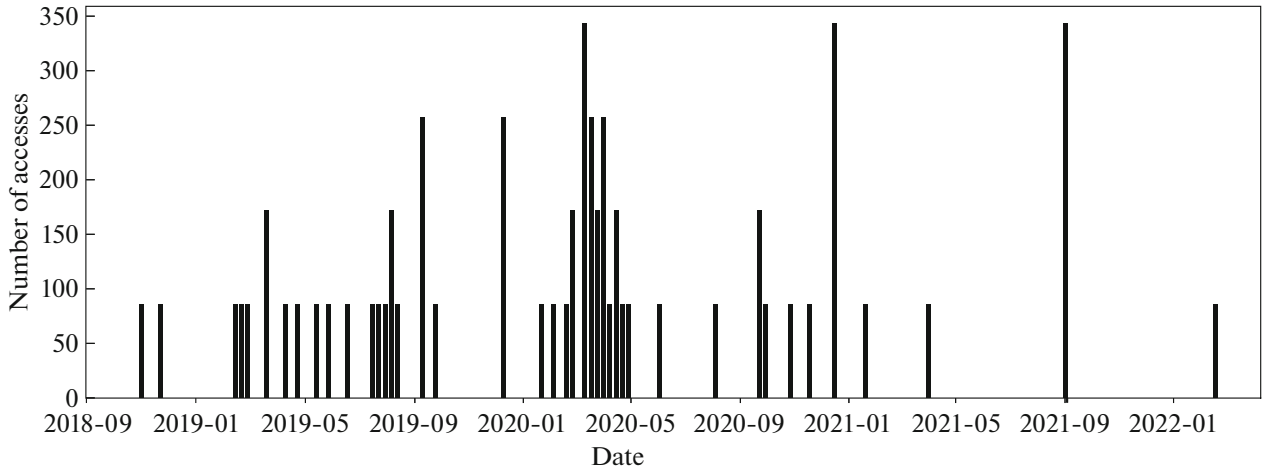

Fig. 5. Accesses of a specific dataset.

Let $D_i = \{n_{i,t}, n_{i,t+1}, n_{i,t+2}, \dots\}$ is access pattern of i_{th} dataset, where $n_{i,t}$ is the number of accesses to the dataset for t_{th} week. Then, the time series for the group of datasets can be described as

$$G = \sum_{i=1}^{G_{size}} D_i = \left\{ \sum_{i=1}^{G_{size}} n_{i,t}, \sum_{i=1}^{G_{size}} n_{i,t+1}, \dots \right\}.$$

In the second scenario, the data is presented as an array of time series for all datasets within a specific group. A single dataset can also be viewed as a sequence of access counts. However, to train prediction models, all available datasets (sequences) should be utilized to derive general correlations. The set of individual dataset time series can be represented as an $M \times N$ matrix, where M is the number of datasets and N is the number of weeks in the time series. Due to the sporadic use of individual datasets over extended time periods, this matrix has a sparse form. To map the data, we use a slice of all time series with a length of $(n-1)$ as the training set X, and Y represents the values from the n_{th} week converted from the number of accesses to datasets. If the number of accesses is greater than one, it is classified as "1," and if there are no accesses, it is classified as "0."

The total number of datasets is around 1 000 000, but we have excluded data that are rarely used (datasets that have less than 30 accesses during the whole period). The total number of weeks in each dataset is 300. Thus, the study uses approximately 10 000 time series of length 300, each time series being an integer vector representing the number of accesses to the data set per week i. The data may be represented as a matrix of size (M, N), where M=10 000, $N = 300$.

5. TIME-SERIES ANALYSIS: SEASONALITY, TRENDS

We should statistically explore time-series sequences and try to find some correlations, trends and seasonality. Figure 6 demonstrates time-series for three different groups of datasets, and their decomposition[3)]. All three groups are from the Monte-Carlo simulated derived datasets that are the most popular. They differ in physics process: Higgs, Top Quark and Supersymmetry.

The HIGGD and SUSY trends can be seen peaking around 2020 when they were the most popular. At the same time, the popularity of TOPQ increases over the years and reaches an asymptote to some extent. The yearly chart also shows some aspects. For example, the HIGGD datasets are least popular in August. The SUSY group shows lower usage in the second half of the year.

The presence of these trends and seasonality indicates that there are certain patterns in data access and they can be forecasted.

[3)]The decomposition was performed by Facebook Prophet [7].

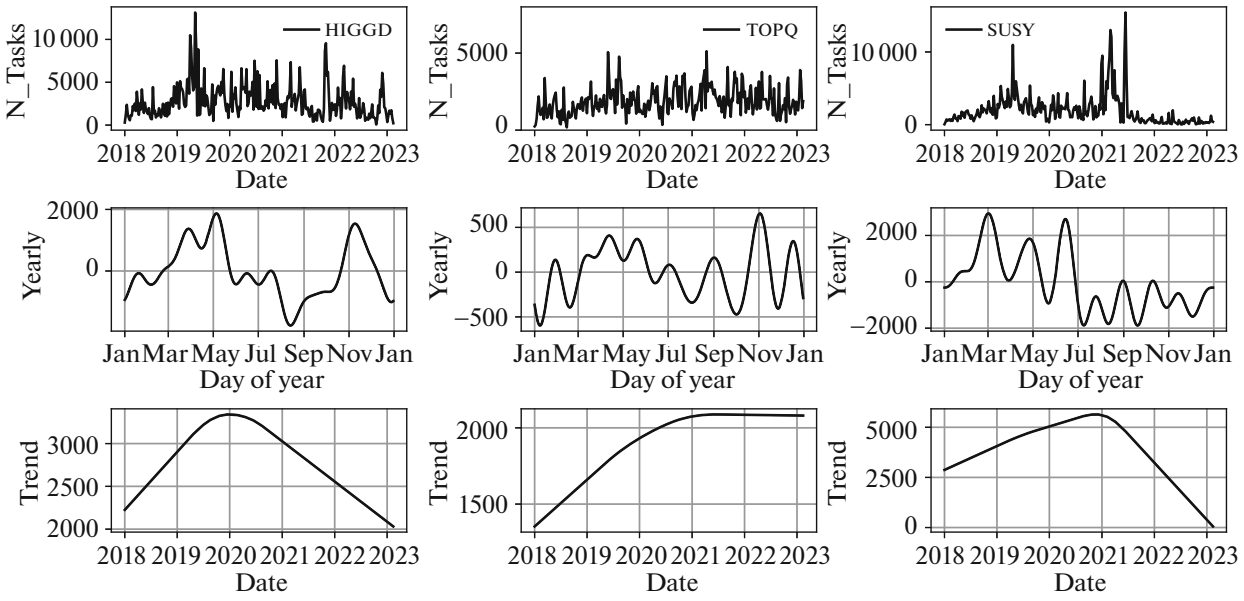

Fig. 6. Number of Tasks time-series and its decomposition for three different dataset groups. Data format: HIGGD (left), TOPQ (center), SUSY (right).

6. EXISTING APPROACHES FOR TIME-SERIES FORECASTS

Table 2 provides a comprehensive list of algorithms that can be utilized for the development of the time series forecast models.

7. POPULARITY OF DATASET GROUPS OF HIGH ENERGY PHYSICS EXPERIMENTS

Predicting the popularity of a group of datasets is crucial for efficient search and analysis. We used data accesses time series for the past five years and employed two models, namely LSTM and Facebook Prophet, to address this regression problem.

7.1. LSTM Forecast Model for the Groups of Datasets

The neural network model based on LSTM can be applied to different variations of the prediction task:

- *One-step prediction*—prediction one new point (a value for the future week in our case) based on several past points (n_{lags} points);
- *Multi-step prediction*—sequential applying one-step model with addition of a new predicted point to the array of historical values at each step;
- *Multi-output prediction*—using a model that outputs several future points simultaneously based on the several points in the past.

7.1.1. One-step prediction. LSTM layer consists of blocks with sigmoid and tanh activation. Those functions give values in $[-1, 1]$ segment. Meanwhile the values in the input time series are positive and may reach tens of thousands (they stands for the number of accesses to the datasets). For better forecasting the next transformation of the data was used: $\mathrm{Diff}(\log(x+1))$, where the Diff transformation returns a time series with difference between current value and the previous value for every time moment.

The dimension of the LSTM hidden state (n_{units}) is a hyperparameter of the model that has been chosen to be equal to 24 as a trade-off between model complexity and prediction time. The detailed research on choosing this hyperparameter is one of the future directions for our work. Therefore, the

Table 2. Existing algorithms used for time series prediction

Class of models	Implementation	Description
Statistical	ARIMA/SARIMA [6]	ARIMA (Autoregressive Integrated Moving Average) is successor of ARMA (Autoregressive Moving Average) model. It combines Autoregression and Moving average models together, and applies them for differences of input time-series values. Seasonal ARIMA (SARIMA) is an ARIMA model where lag parameters are multiples of period (the length of the season)
	Facebook Prophet [7, 8]	Facebook prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It takes the whole dataframe with time-series as input for training, and provides forecasting for many time moments in the future
Classical machine learning methods	Logistic Regression [9]	Logistic regression is a statistical analysis method to predict a binary outcome, such as yes or no, based on prior observations of the event. It uses a linear combination of one or more independent variables
	Random Forest [10]	Random forests or random decision forests is an ensemble learning method for classification and regression that operates by constructing a multitude of decision trees at training time
	SVM (Support Vector Machine) [11]	SVM is a group methods which objective is to find a hyperplane in an N-dimensional space (N is the number of features) that distinctly classifies the data points
Neural Network models	LSTM (Long Short-Term Memory) [12]	A recurrent neural network (RNN) that takes a sequence of input values and returns a single (possibly vector) value. It has a hidden state. Such a simple network also called LSTM Cell. It includes four layers to learn with sigmoid and tanh activation. In basic configuration LSTM can be used as a one-step forecaster that takes several historical values and gives one value for next timestamp. More complex networks based on LSTM are widely used [13, 14] for prediction of different types of time series
	GRU (Gated Recurrent Unit) [15]	Is an RNN, similar to LSTM. It contains only three layers to learn (two with sigmoid activation, and one with tanh activation). GRU has fewer parameters than LSTM, and takes less number of calculations for training. This type of network was also applied [16] to time series forecasting tasks
	TFT (Temporal Fusion Transformer, Google) [17]	An attention-based DNN engineering for multi-horizon forecasting which combines high-performance multi-horizon estimating with interpretable insights into transient elements. TFT utilizes recurrent layers for neighborhood handling and interpretable self-attention layers for long-term dependencies. TFT uses specialized components to select relevant functions and a series of gate layers to suppress unnecessary components to achieve high performance in a variety of scenarios
Boosting models	XGBoost [18], AdaBoost [19], CATBoost [20]	Boosting is an ensemble learning method that combines a set of weak learners into a strong learner to minimize training errors. In boosting, a random sample of data is selected, fitted with a model and, then, trained sequentially—that is, each model tries to compensate for the weaknesses of its predecessor. With each iteration, the weak rules from each individual classifier are combined to form one, strong prediction rule
Hybrid models	Neural Prophet [21, 22]	A successor to Facebook Prophet based on PyTorch. It has a similar interface, and a lot of hyperparameters. Hyperparameters include: n_lags (number of historical data points to input to the model), forecast horizon, model training parameters, seasonality parameters, etc.
	DeepAR (Amazon) [23]	A time series model that combines both combines both deep-learning and autoregressive characteristics. DeepAR learns from chronicled of all time series within the information set together. The model learns regular behavior and conditions on given covariates over time series, strategy does not assume Gaussian noise, but can utilize a wide range of probability functions, the model is able to supply provide forecasts for sequences with little or no history at all and it makes probabilistic forecasts in form of Monte Carlo samples

LSTM outputs n_{units} values, but the model should give only one value for future prediction. Thus, a dense layer of $(n_{units} \times 1)$ was used in this case.

Figure 7 presents prediction example for one-step forecasting by LSTM-based model. The time

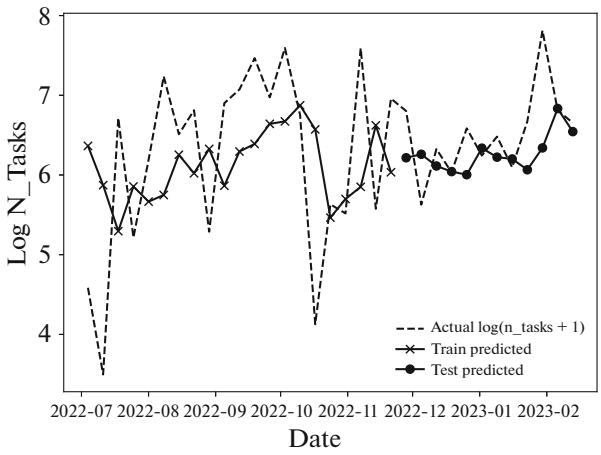

Fig. 7. One-step prediction by LSTM-based model for time-series of SUSY format. Every next point are predicted based on actual previous data.

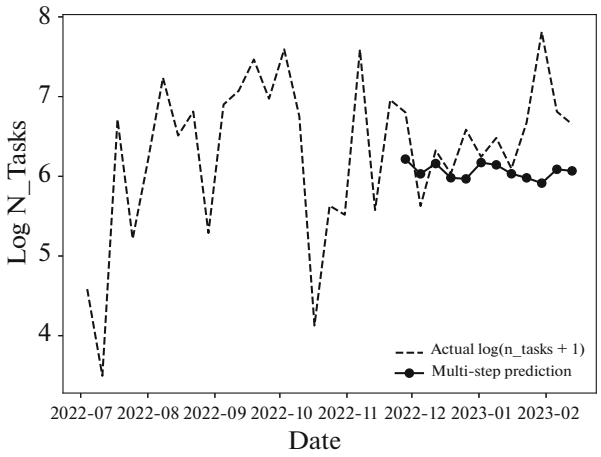

Fig. 8. Multi-step prediction for 12 weeks by LSTM-based model for time-series of SUSY format. Every next point are predicted based on previously predicted value.

series has been split into train and test parts. For better clarity data is shown since July 2022. The model was evaluated on the data, transformed by log, while the accuracy on the original data becomes substantially smaller, that can be caused by rapidly oscillating time series with peaks, which can not be properly predicted by the model.

7.1.2. Multi-step prediction. The above learned model can be applied several times so that the predicted value at each step is added to input for the next steps. Figure 8 presents an example of multi-step prediction. Here, the prediction was made for 12 weeks in the future starting from the time moment of November 28, 2022.

7.1.3. Multi-output prediction. The multi-output LSTM model may have the following form: LSTM cell, that returns whole sequence, 1D max-pooling layer along the time axis, dense layer with $n_{forecasts}$ output neurons (where $n_{forecasts}$ is a number of future weeks to forecast).

Figure 9 demonstrates an example of multi-output prediction for 12 weeks forward. The data transformation before passing it to the model is the same as for one-step model.

7.2. Facebook Prophet Forecasting for the Groups of Datasets

Unlike the previous model, Facebook Prophet provides prediction for arbitrary time interval in the future. But it is important to have enough historical data to train the model. Prophet reveals trends and

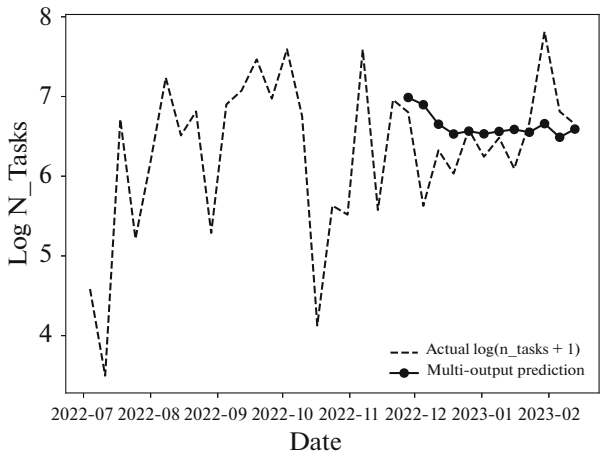

Fig. 9. Multi-output prediction by LSTM-based model for time-series of SUSY format. Prediction for 12 weeks in the future.

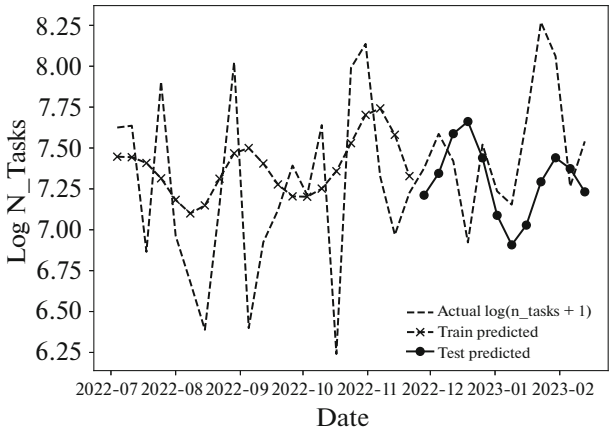

Fig. 10. Prediction by Facebook Prophet for time-series of TOPQ format. Prediction for 12 weeks in the future.

seasonality (and possible considers events such as holidays). It has several parameters: type of trend growth, seasonalities (daily, weekly, yearly), changepoints, etc.

The data was normalized by $\log(x+1)$ transformation for smoothing of the large peaks. The yearly seasonality parameter was enabled, as our time series tend to have calendar specialties. The other parameters were left default. An example of prediction is shown at the Fig. 10. The data was split into train (to learn Prophet) and test (12 last weeks to predict) parts.

7.3. Cross Validation Results of LSTM, Prophet and Naive Forecasts

Since one of the quality indicators of a time series predictive model is high prediction accuracy compared to naive prediction (based on shifted values), we have added it to the comparison Table 3 with the results of cross validation, which was implemented by selecting cutoff points in the history, and for each of them fitting the model using data only up to that cutoff point.

Table 3 shows that the models we studied were unable to significantly surpass naive predictions on logarithmic data. In some cases, for TOPQ and HIGGD groups, the accuracy of Naive forecast is almost equal to LSTM and Prohpet.

The reason the models don't predict the popularity of different groups with the same accuracy is explained by the specific patterns of how users work with the data. From Fig. 1, it can be seen that SUSY data format is the most popular, but when predicting, we obtain an error of 9–15%, which is significantly higher than for HIGGD or TOPQ. This may be explained by the specific access patterns to this group.

Table 3. Symmetric mean absolute percentage error) metric) for LSTM and Facebook Prophet models in comparison with the Naive forecast for three different groups of datasets represented with logarithmic transformation

Model	Type of Prediction	No. weeks	HIGGD, %	TOPQ, %	SUSY, %
LSTM	One-step	1	7.21	4.90	11.68
	Multi-step	4	7.70	4.97	13.81
		12	7.85	5.05	15.03
	Multi-output	4	8.64	6.20	13.05
		12	8.83	6.67	12.78
Prophet	Multi-output	4	8.16	7.23	9.36
		12	7.67	6.83	12.15
Naive	Multi-output	4	9.85	6.77	13.31
		12	11.21	8.03	13.69

In general, if we compare models without taking into account the Naive predictions, then it can be seen that the multi-step LSTM forecasting model on TOPQ group with a horizon of 4 and 12 weeks provides the highest accuracy (with an error of about 5%). Meanwhile, the error on HIGGD is almost 8%, and on SUSY data it is about 14% and 15% for 4 and 12 weeks correspondingly.

The multi-output LSTM model yields comparable results for the three datasets, and there is no variation in forecasting accuracy as the prediction horizon extends from 4 to 12 weeks.

For all three groups of datasets, Facebook Prophet showed almost equal accuracy on 4 and 12 weeks forecast horizon in comparison with LSTM-based multi-output model. The most accurate result for the SUSY group were obtained using the Prophet 4 weeks ahead forecast.

The fact that the model performs well in predicting logarithmic data and performs significantly worse when the data is transformed to a real-world scale determines the models limitations: they can be utilized to predict the increase or decrease in the number of requests for a given group of datasets, but it cannot be used to predict a specific number of those requests.

8. INDIVIDUAL DATASETS POPULARITY FORECAST: CLASSIFICATION APPROACH

The popularity of individual datasets is the second layer of the hierarchical prediction system. And it is based on weekly access patterns of each dataset. To develop a predictive model, it is essential to identify the access pattern of a dataset. This can be done by analyzing a sequence of numbers that represents the total number of data accesses for each week over a specific period (such as a year). By collecting such time series data for each dataset, we can combine them and leverage classification techniques to train our model.

Given the abundance of short- or medium-term data over the next 5 years, combining all datasets would result in a sparse matrix with numerous zeros. To address this, we need to reduce the training interval and normalize the data.

Statistical analysis indicates that a significant portion of data has a long lifespan, so it is advisable to use extended time periods when creating predictive models to capture more sequences. It is important to note that analyzing data access patterns over the past 5 years may yield distinct outcomes, as depicted in Fig. 11. Consequently, in addition to examining each dataset's pattern, it is crucial to determine the ideal training interval for forecasting.

When forecasting the popularity of individual datasets, it is challenging to use regression methods due to the large number of time series involved. Predicting on a vast set of data samples can be time-consuming, and statistical methods may not consider the potential dependencies between different time series, leading to unstable predictions. While these methods can account for specific features such as dates, it is up to the analyst to define them correctly, potentially resulting in the omission of crucial details. Additionally, it is crucial to use extended time periods when creating predictive models as a

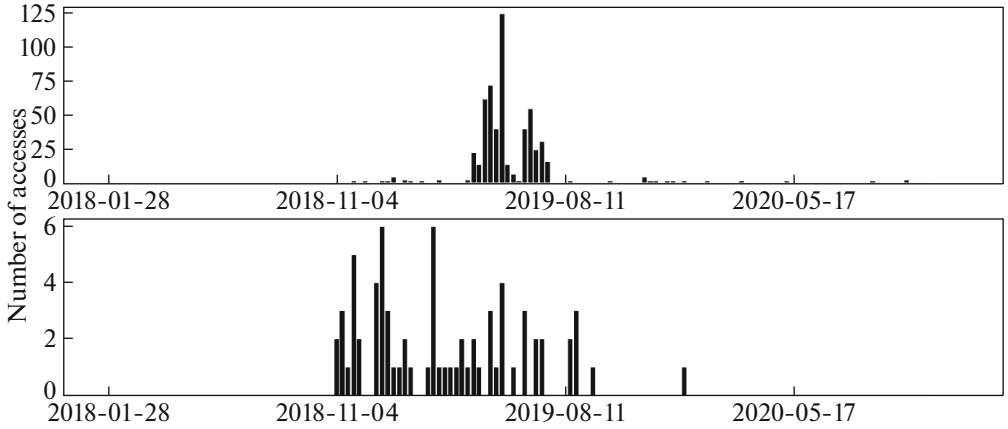

Fig. 11. Access patterns for two random datasets.

significant portion of data has a long lifespan. It is essential to determine the optimal training interval for forecasting. Through experimentation, we have determined that a 30-week interval is an optimal time period that can be used as a basis for predictions. Thus, the main possible way to solve the classification problem for a large set of time series is machine learning, which is able to overcome the problems described above.

Further, we will assume that the sample for a single dataset consists of triplets $[A_{before}, A_{week_id}, A_{week_id+1}]$, where

- A_{before}—a vector of 30 values. The values are the number of accesses for each week from $(week\_id - 30)$ to $(week\_id - 1)$.
- A_{week_id}—the number of accesses for a specific week $(week\_id)$.
- A_{week_id+1}—the number of accesses for a specific week $(week\_id + 1)$.

Accuracy is used as a general metric on three different validation scenarios:

1. *Next week prediction on train*. The general dataset is all given to train subsample, trained on $[A_{before}, A_{week\_id}]^{train}$ and trying to predict $A^{train}_{week\_id+1}$. This is so far the most important metric, as in the most cases alogrithm is expected to predict accesses for known long-living datasets, that can be used to train model as well.
2. *Next week prediction on test*. The general dataset is divided into train and test subsamples. The proposed methods are trained on $[A_{before}, A_{week\_id}]^{train}$ and predict $A^{test}_{week\_id+1}$. The high score results of this metric will show that the number of accesses to different datasets correlates strongly enough to predict the future weeks of unknown time series. This feature may be effective for the classifying the new datasets, but could not be used for training.

Note that there is no need to predict several weeks ahead at once. The model can predict one week and be further trained again on its prediction.

8.1. Results

There are many different methods for classifying time series. In this study, we conduct experiments on several popular models.

Predictions were made for weeks from A_{140} to A_{297} with a step of 3. Based on the results obtained for the models, the median value of f1-score metrics for classes "0" and "1", as well as accuracy metrics, were calculated. The results are presented in the Table 4.

Table 4. Cross-validation results on set of metrics for different approaches on *Next week prediction on TRAIN/TEST* sample

Model	f1-score Class "0"	f1-score Class "1"	Accuracy
LSTM	**0.71 / 0.67**	**0.73 / 0.75**	**0.73 / 0.71**
CATBoost	0.71 / 0.65	0.72 / 0.71	0.71 / 0.71
XGBoost	0.68/ 0.65	0.71 / 0.71	0.7 / 0.69
MLP	0.66 / 0.62	0.7 / 0.65	0.69 / 0.64
FCN	0.63 / 0.59	0.66 / 0.67	0.64 / 0.63
Logistic regression	0.58 / 0.6	0.64 / 0.67	0.63 / 0.64
AdaBoost	0.6 / 0.58	0.68 / 0.66	0.64 / 0.63

According to the results, LSTM model performed better than the others in classifying access to datasets, although it did not significantly outperform its closest competitor, CATBoost. All metrics are close to 0.7, which means that the models are finding some data patterns, but they are still insufficient. This is likely due to the sparsity of the data and partially unpredictable behavior: most individual datasets do not have a trend in the number of accesses. This is because the total number of requests to individual datasets is not large. Moreover, some of them are not popular and not used in subsequent years.

Boxplot is a statistical graph that shows a set of statistical features of a sample: the median, the first and third quartiles, the maximum and minimum values, as well as outliers. Figures 12 and 13 show boxplots for the f1-score metric on classes 0 and 1. There are almost no outliers in the classification, but the metric values vary greatly from week to week. Thus, among the predicted weeks, there will be both very well-predicted weeks and unsuccessful ones, reaching a metric value of 0.5.

8.2. Model Architecture and Hyperparameters

Neural networks. Neural networks were trained for 100 epochs using the Adam optimizer and a learning rate of 0.001. The number of model parameters:

- LSTM(6,532 parameters), which consists of two LSTM cells and Linear layer for classification;
- MLP(2,342 parameters) with several linear layers;
- FCN(7,002 parameters) containing 3 convolutional layers and linear one at the end.

Bigger architectures were tested as well but did not show any good results and were strongly affected by overtraining.

Boosting models and logistic regression. All non-neural network machine learning methods were trained with default parameters or based on grid-search results [25].

8.3. Ensemble Appoach

Ensemble architectures are often used to improve the stability and accuracy of machine learning models [26]. However, in our case, it appears that combining different methods will not yield significant improvements. Despite initial assumptions that ensembles would be beneficial, analysis of the f1-score metric values for different models on various weeks reveals that most models behave similarly according to Fig. 14. Therefore, it can be concluded that combining them into an ensemble model is unlikely to provide any substantial improvement as they all behave almost identically from week to week.

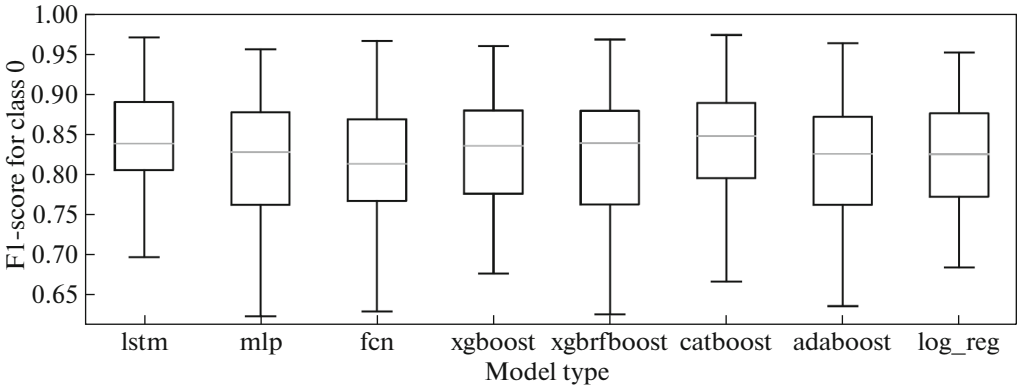

Fig. 12. Boxplots for *Next week prediction on test sample*. f1-score on Class "0".

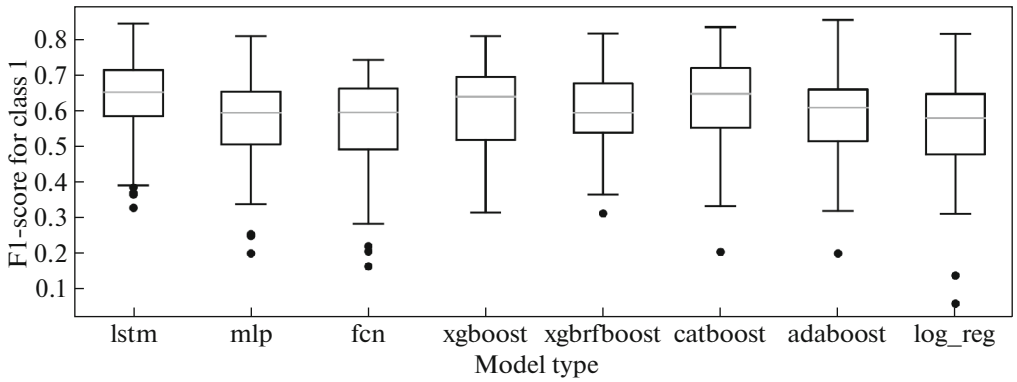

Fig. 13. Boxplots for *Next week prediction on test sample*. f1-score on Class "1".

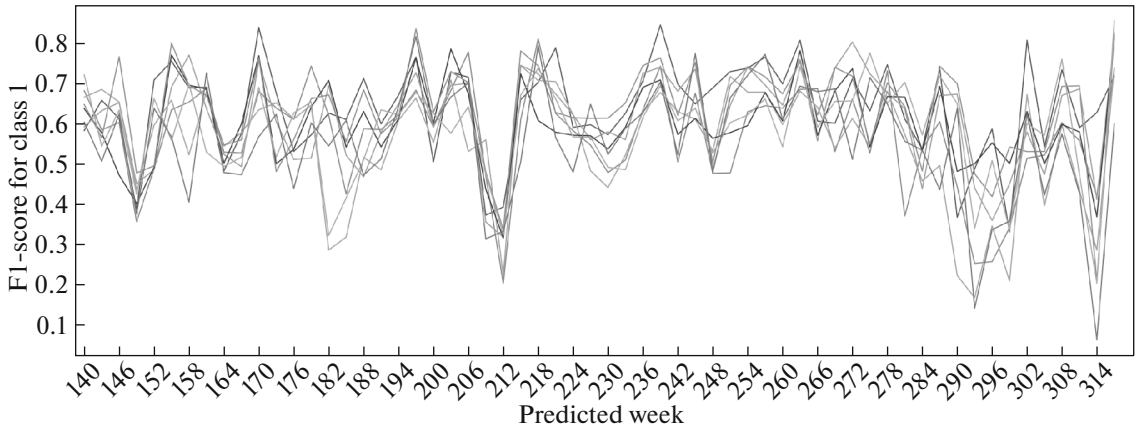

Fig. 14. F1-score on class 1 for different models by week.

9. CONCLUSION

Proposed hierarchical method for popularity forecasting involves two levels—one for groups of data and the other for individual datasets. Popularity of data groups is estimated using statistical models or regressions, while a classification approach is used for individual datasets, categorizing them as either popular or not popular for the coming time interval.

By applying a log-transformation to the data, we were able to generate popularity forecasts for data groups up to 12 weeks in advance with a certain degree of accuracy. However, these models could not significantly exceed the naive predictions. Moreover, they struggles to predict peak values resulting from

specific user behaviors such as benchmarking or validation. Ongoing research is focused on identifying and filtering out datasets affected by biased user access patterns. All three evaluated models (LSTM, Prophet and Naive) are generally interchangeable, but the multi-step LSTM showed the best results for the HIGGD and TOPQ groups, while surprisingly, the Naive multi-output model showed the best results for SUSY. These models might be used to anticipate increases or decreases in the number of accesses for a given group, but not to predict the exact number of accesses.

To forecast individual dataset popularity using classification, the model was trained on 30 weeks of data, mapped by the values of the 31st week into two classes: Class "0" (datasets were not accessed) and Class "1" (datasets were accessed). When trained on all available datasets until a certain week, the LSTM model can predict each dataset's class for the next week on trained sample with roughly 71% and 73% accuracy for Class "0" and Class "1," respectively.

Sparse data is always difficult to predict and dataset accesses prediction is not an exception. The conducted experiments showed that the data have certain features and models can identify them. However, it is worth further developing the issue of using machine learning methods, since the retrieved result of dry metrics is not yet ideal.

The classification model can be applied using the method of weekly retraining of existing datasets, with subsequent prediction for the next week. On output, we will receive datasets and their corresponding classes. And, in any case, there will be a certain error present for both classes "0" and "1," However, as the final goal of the popularity forecast is assisting in making a decision regarding increasing or decreasing of the number of temporal copies of datasets in accordance with their anticipated popularity, then the error can be made in both directions: that is, create more replicas than necessary in the case of False Positive predictions for Class "1," or remove more replicas than necessary for False Negative predictions for Class "0." Therefore, one error cancels out the other if the total volumes of newly created and removed copies of datasets are equal. While this technique may have sound theoretical underpinnings, it can be used in practical applications, but only for high-precision models.

Both models, regression for dataset groups and classification for individual datasets, have certain limitations, which must be considered when developing forecasting applications. The training data should be scrutinized and purified, and alternative methods can be applied to refine these models further. This study carried out with the simplest representation of data and basic predictive data models, and this approach can inspire further advancements in this area.

REFERENCES

1. M. Grigorieva et al., "High energy physics data popularity: ATLAS datasets popularity case study," in *Proceedings of the 2020 Ivannikov Memorial Workshop IVMEM, Orel, Russia* (2020), pp. 22–28. https://doi.org/10.1109/IVMEM51402.2020.00010
2. M. Barisits, T. Beermann, F. Berghaus, et al., "Rucio: Scientific data management," Comput. Software Big Sci. **3**, 11 (2019). https://doi.org/10.1007/s41781-019-0026-3
3. T. Beermann, O. Chuchuk, A. di Girolamo, M. Grigorieva, A. Klimentov, M. Lassnig, M. Schulz, A. Sciaba, and E. Tretyakov, "Methods of data popularity evaluation in the ATLAS experiment at the LHC," EPJ Web Conf. **251**, 02013 (2021). https://doi.org/10.1051/epjconf/202125102013
4. T. Beermann, P. Maettig, G. Stewart, M. Lassnig, V. Garonne, M. Barisits, R. Vigne, C. Serfon, L. Goossens, A. Nairz, and A. Molfetas (on behalf of the ATLAS Collab.), "Popularity prediction tool for ATLAS distributed data management," J. Phys.: Conf. Ser. **513**, 042004 (2014). https://doi.org/10.1088/1742-6596/513/4/042004
5. V. Kuznetsov, T. Li, L. Giommi, D. Bonacorsi, and T. Wildish, "Predicting dataset popularity for the CMS experiment," J. Phys.: Conf. Ser. **762**, 012048 (2016). https://doi.org/10.1088/1742-6596/762/1/012048
6. G. Box and G. Jenkins, *Time Series Analysis: Forecasting and Control* (Holden-Day, San Francisco, 1970).
7. Facebook Prophet Home Page. https://facebook.github.io/prophet/. Accessed 2023.
8. S. Taylor and B. Letham, "Forecasting at scale," Am. Statist. **72**, 37–45 (2018).
9. D. G. Kleinbaum et al., *Logistic Regression* (Springer, New York, 2002).
10. L. Breiman, "Random forests," Mach. Learn. **45**, 5–32 (2001).
11. M. A. Hearst et al., "Support vector machines," IEEE Intell. Syst. Appl. **13** (4), 18–28 (1998).
12. B. Lindemann, T. Müller, H. Vietz, N. Jazdi, and M. Weyrich, "A survey on long short-term memory networks for time series prediction," Proc. CIRP **99**, 650–655 (2021). https://doi.org/10.1016/j.procir.2021.03.088

13. F. Karim, S. Majumdar, H. Darabi, and S. Harford, "Multivariate LSTM-FCNs for time series classification," arXiv: 1801.04503 (2018). https://doi.org/10.48550/arXiv.1801.04503
14. F. Karim, S. Majumdar, and H. Darabi, "Insights into LSTM fully convolutional networks for time series classification," IEEE Access **7**, 67718–67725 (2019). https://doi.org/10.1109/ACCESS.2019.2916828
15. D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," arXiv: 1409.0473 (2014). https://doi.org/10.48550/arXiv.1409.0473
16. M. Rahman, Md. Hossain, T. Junaid, Md. Forhad, and M. Hossen, "Predicting prices of stock market using Gated Recurrent Units (GRUs) neural networks," Int. J. Comput. Sci. Network Secur. **19**, 213–222 (2019).
17. B. Lim et al., "Temporal fusion transformers for interpretable multi-horizon time series forecasting," Int. J. Forecast. **37**, 1748–1764 (2021).
18. T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016), pp. 785–794.
19. R. E. Schapire, "Explaining adaboost," in *Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik* (Springer, 2013), pp. 37–52.
20. L. Prokhorenkova and G. Gusev, "CatBoost: Unbiased boosting with categorical features," in *Proceedings of the International Conference on Adv. on Neural Information Processing Systems NIPS'18* (2018), Vol. 31, p. 6639.
21. Neural Prophet. https://neuralprophet.com/. Accessed 2023.
22. O. Triebe, H. Hewamalage, P. Pilyugina, N. Laptev, C. Bergmeir, and R. Rajagopal, "NeuralProphet: Explainable forecasting at scale," arXiv: 2111.15397 (2021).
23. D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "DeepAR: Probabilistic forecasting with autoregressive recurrent networks," Int. J. Forecast. **36**, 1181–1191 (2020).
24. Z. Chen and Y. Yang, "Assessing forecast accuracy measure," (2004).
25. CATBoost Grid Search. https://catboost.ai/en/docs/concepts/python-reference_catboost_grid_search.
26. H. Ismail Fawaz et al., "Deep neural network ensembles for time series classification," in *Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN) Budapest, Hungary* (2019), pp. 1–6.