Prequential Model Selection for Time Series Forecasting based on Saliency Maps

1st Shivani Tomar
Trinity College Dublin
IBM Research, Ireland
Dublin, Ireland
tomars@tcd.ie/shivani.tomar1@ibm.com

2rd Seshu Tirupathi
IBM Research, Ireland
Dublin, Ireland
seshutir@ie.ibm.com

3nd Dhaval Vinodbhai Salwala
IBM Research, Ireland
Dublin, Ireland
dhaval.vinodbhai.salwala@ibm.com

4th Ivana Dusparic
Trinity College Dublin
Dublin, Ireland
ivana.dusparic@tcd.ie

5th Elizabeth Daly
IBM Research, Ireland
Dublin, Ireland
elizabeth.daly@ie.ibm.com

***Abstract*—Over the last few years, incremental machine learning for streaming data has gained significant attention due to the need to learn from a constantly evolving stream of data without the need to store it. The advent of big data has further fuelled research on developing systems that can cope with continuously changing data streams and tackle the challenges associated with historical data requirements. The problem of time series forecasting has been studied using varied approaches like neural networks, ensemble methods, decision trees and rules, support vector machines to name a few. However, neural network models have gained particular attention in dealing with changing data distribution due to their generalization abilities. In this paper, we propose a prequential framework named PS-PGSM which involves incrementally training the base models and online Regions of Competence (ROC) computation followed by selection of the best forecaster for the task of time series forecasting using saliency maps. We build upon the state-of-the-art approach named OS-PGSM (Online Model Selection using Performance Gradient based Saliency Maps) in which the model training and ROC computation is performed offline. Past research has demonstrated that a set of different models enables specialization for each model compared to a single forecasting model which is particularly useful when predicting for an evolving time series sequence. Our approach uses saliency maps for prequential calculation of ROC for each model to find the best forecaster based on the performance of each model. We evaluate the proposed approach against OS-PGSM, as well as against previous best performing model by first conducting preliminary experiments on 10 real-world time series datasets and then using 2 real-world big datasets to showcase its applicability to big data. Experimental results not only validate the effectiveness of our approach for big data but also demonstrate superior performance in terms of prediction accuracy and computational time efficiency while also handling concept drift.**

***Index Terms*—Incremental machine learning, saliency maps, time series, neural networks.**

I. Introduction

The notion of stationarity in streaming time series data is often violated in realistic settings where the underlying data distribution changes over time referred to as Concept Drift [11]. This is true for most real-world scenarios where the statistical properties of the incoming data tend to change due to factors like seasonality and changes in the outside environment. It is of vital importance to deal with this phenomenon for machine learning models to predict accurately on constantly evolving time series data. Previous research [7] [1] [4] in this field has shown that different forecasting models specialize on different parts of the input time series and also exhibit varying levels of performance over time. This has motivated the use of a combination of models specializing on different parts of the input sequence rather than a single model to predict over the entire incoming sequence of data. The next step is to either select the best model from this pool of models or combine the results of all the models using an aggregate function. When selecting the best model, meta-learning models [21] or metrics like ROC have been used to select the most suitable model for prediction at the next time-step. ROCs are parts of the input sequence where a candidate model performs better than the rest of the candidate models. Previous works [8] [22] demonstrate that ROCs are calculated on the validation dataset in an offline setting. We build upon the original work [22] which contributed the framework named OS-PGSM, a novel approach of online CNN selection based on the use of gradient-based saliency maps for computing ROCs offline for time series forecasting. The use of saliency maps is fairly new to the context of time series and this work provides a promising direction towards introducing explainability into the model selection process. Since we are dealing with big data in streaming setting, we use prequential approach [6] to evaluate the models as it is commonly used in such scenarios [11]. In this paper, we propose our approach named PS-PGSM (Prequential Selection using Performance Gradient based Saliency Maps) where we perform prequential training and ROC calculation to evaluate the model performances. We compute the ROCs online on the same training minibatch that is used to prequentially train

This work has been supported by the MORE project (grant agreement 957345), funded by the EU Horizon 2020 program.

the candidate models. This method is applicable to both small and big time series data, however it is of particular relevance to big data in streaming setting where historical data cannot be stored offline due to limited computational resources. We demonstrate the performance of our approach by predicting on 2 real-world time series big datasets and 10 small time series datasets as used in [22]. In order to further validate the performance in case of changing concepts we use synthetic data and show how our approach adapts to concept drift better than other approaches.

II. Related Work

Over the past few years, deep neural networks have been successfully applied to time series forecasting, in a variety of applications like renewable energy prediction [14], web traffic prediction [5], and weather forecasting [20]. This can be attributed to their better generalization capabilities compared to statistical approaches when the data involved contains high degree of non-linearity. With the immense growth of IoT and related monitoring and tracking applications, high-speed data is generated in streaming context which is high-dimensional in nature, containing complex non-linear relationships among features. Moreover, this data often shows changes in the underlying distribution as a result of factors like changes in usage patterns, customer profiles and seasonality. To handle this concept drift, incremental or online learning methods have been proposed that continuously update the model's knowledge as new data becomes available in a streaming fashion [17]. This eliminates the need to store the entire dataset and also provides most up-to-date models at any point in time.

Past research [24] [7] demonstrates that no single model performs consistently well over the entire input sequence and different forecasters vary in their predictive performance when it comes to forecasting for time series involving complex inherent structures. This motivates the use of techniques [8] involving either combination of different models in an ensemble framework or selection of the best forecaster based on a specific criteria [25]. Meta learning approaches [7] are widely used where the selection of the model is based on a pre-defined set of features. The idea is to define regions of competence for each of the models to find out which models performs better on which part of the input sequence and use this attribute to make predictions at test time. The pace at which big data is generated and its increasing complexity has brought the researchers to explore newer techniques of processing big data to make it applicable to machine learning algorithms. Meta learning is one of those preferred techniques used for model selection in big data analytics [18] as well. This approach has also been used for automation of ML models in the manufacturing industry [12] [13] to exploit industrial big data with minimum intervention of an ML expert.

Saliency maps have been widely used in the context of image classification. They help in visualising those parts of the image which are salient or important in predicting the target class [26]. This is achieved by creating class-specific heatmaps which highlight the important regions of the image to make the predictions explainable to a novice user. Only recently, saliency maps have been used for the task of time series prediction [2]. Both the temporal dimension as well as features of time series data are used in a 2-stage CNN architecture to explain which features and time intervals are responsible for making the predictions. Other explainable machine learning methods have been explored using big data from varied domains like fraud detection [19], web usage [15] to name a few.

In their recent work [22], Amal et. al. demonstrated further how saliency maps can be used to map the performance of candidate models over different parts of the time series facilitating an online model selection framework which selects the best forecaster to make prediction online. Though the work attempts to cope with the evolving nature of the time series by making model selection and prediction online, it trains the candidate models offline along with offline ROC computation which makes the whole process not extendable to streaming data.

Our work introduces the prequential extension to their offline framework by using saliency maps in calculating ROCs in an online setting, making it possible to adapt model training to the time series data stream. It also helps combine the benefits of both model selection and explainability using saliency maps.

III. Proposed method

This section introduces our approach PS-PGSM showing the parts of OS-PGSM framework which have been converted to prequential computation. There are four stages in the framework, namely: candidate model training, ROC computation using saliency maps, model selection, and prediction. The original framework, as designed in [22], performs the first 2 stages, i.e. model training and ROC calculation, in an offline mode, assuming that the entire dataset is available in advance. Time series X_t is split into training and validation sets, where the training subset is used to train a pool of candidate models offline and the validation set is used to calculate ROCs for each of the candidate models, using a modified version of Grad-CAM. The candidate models are created using the basic structure of 1D convolutional layers while varying the filter and kernel sizes along with the number of convolutional layers. The addition of LSTM layer into the CNNs is done to create further variations resulting in the pool consisting of multiple candidate models (12 in original implementation). Our approach aims to train these candidates in prequential manner using minibatches of the time series dataset and uses the same training minibatch to calculate the ROCs for each model online. This makes the entire framework online and easily adaptive to the changing underlying distribution of the time series. Algorithm 1 outlines the steps involved in our approach.

As shown in Figure 1, which outlines our approach of prequential training and ROC computation, the time series denoted by X_t is sequentially divided into i number of minibatches, each of size j. Each minibatch, X^{mini} is first

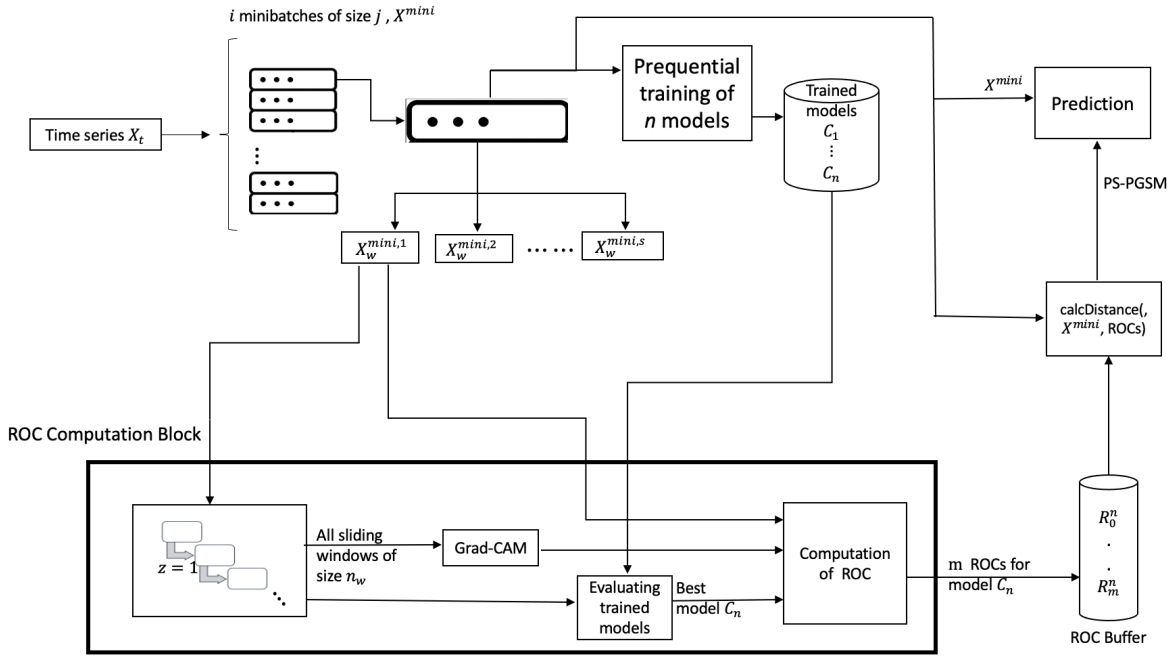

Fig. 1. PS-PGSM: Prequential Selection using Performance Gradient based Saliency Maps for Time Series Forecasting

Algorithm 1 Prequential training and ROC computation

Input: i minibatches and n candidate models
Output: $ROCs$ for each minibatch

1: Incrementally train n models using X^{mini}
2: Split X^{mini} into window size of w, $X_w^{mini,s}$
3: **for** each $X_w^{mini,s}$ in X^{mini} **do**
4: **for** each sliding window of size n_w in $X_w^{mini,s}$ **do**
5: SMAPE = evaluateTrainedModels(n, window n_w)
6: **end for**
7: $best\_model(C_n)$= argmin(SMAPE)
8: $ROCs(C_n)$ = computeROC(C_n, $X_w^{mini,s}$, CAM)
9: **end for**

predicted using PS-PGSM model. The candidate models are incrementally retrained on that minibatch, X^{mini} before making predictions on the next minibatch. We continue in this order for i iterations which is equal to the number of minibatches available. The next step involves calculation of ROC online for all the trained models in that iteration using the same minibatch as used for training. We can use the entire or a subset of the training minibatch for the purpose of ROC computation online. X^{mini} is further split into sub-sequences with a window size of w giving $X_w^{mini,1}$, $X_w^{mini,2}$ and so on to get different evaluations of the candidate models similar to [22]. These subsequences are used to calculate ROCs by performing sliding window operations of size n_w over each $X_w^{mini,s}$ with a step size of z equal to 1. The trained models are then evaluated on the sliding window of size n_w to find the best model C_n with the lowest SMAPE error. ROC for model C_n is then computed using saliency maps. It is important to note that each model can have multiple regions of competence $(R_0^n, R_1^n, .., R_m^n)$ owing to the fact that the same model can specialize on more than one region of the time series.

Saliency maps are generated using one of the most popular methods i.e., gradient-based Class Activation Maps (CAM) [23] which use feature maps (f_{maps}) from the last convolution layer as it is known to preserve detailed spatial information. Following [23], error ϵ_n^k associated with every k^{th} sub window n_w in the minibatch for each model n is used to evaluate the performance of that model over the k^{th} time interval. A weight measure w^ϵ is then calculated by taking the gradient of ϵ_n^k with respect to each feature map A and then averaging over all the activation units (U) in A.

$$w^\epsilon = \frac{1}{U} \sum_u \frac{\partial \epsilon_n^k}{\partial A_u} \tag{1}$$

Subsequently, applying the ReLU helps in removing all the negative contributions and L_n^k is used to find those areas in the k^{th} sub window of the minibatch that have contributed to the error ϵ_n^k.

$$L_n^k = ReLU \sum_{f_{maps}} w^\epsilon A \tag{2}$$

This aids in mapping the performance of a model to the sequence of the input time series. The best model is determined by comparing the obtained ROCs with the data points in the test set and selecting the model corresponding to the closest distance. However, if the distance between the

ROCs for all models and the input sequence is more than a pre-defined threshold which is essentially a very large number (1e8), then the best forecaster from the previous iteration is selected.

IV. Experiments

We run preliminary experiments to evaluate the performance of PS-PGSM first on 10 time series datasets from [22]. For these datasets, we split each into minibatches of size 110 and prequentially train the models for as many iterations as the number of minibatches created for each dataset which varies according to the length of the datasets. We execute 20 runs of the experiments for each dataset randomizing the seed to initialize the models. We use the same pool of 12 candidate CNN models used in [22] which are obtained by varying the filter sizes in $\{32,64,128\}$ and kernel sizes in $\{1,3\}$. In each iteration, all of the 12 candidate models are trained for 1 epoch with a fixed learning rate of 0.0001. Other parameters like the size of sliding window within the validation set is set to 5 which remains unchanged. However, the main difference in our approach is the fact that same minibatch is used for both training the models and calculating ROCs for each of them. It is important to note that for all datasets, the models are trained for 100 epochs in the offline version while only being trained for 1 epoch in our prequential approach as usually done for streaming data in real-world applications. We then use 2 time series big datasets to evaluate the suitability of our approach for big data. To this end, we split the instances into minibatch size of 100 and run the experiments for 8000 iterations keeping all the remaining model configurations same as mentioned above. We also use synthetic time series dataset which is composed of sequences of samples from 4 different concepts repeated in random order to analyse the performance of our approach in the presence of concept drift.

We compare the performance with the following baselines:

1) **Random** which selects a random model from the pool of 12 models further denoted in tables and plots as "Random model".
2) **Previous Best** which is the best model from the previous iteration denoted as "Prev Best" in the tables and plots and lastly,
3) **Offline** which is the model selected based on offline OS-PGSM approach denoted as "Offline" in the tables and plots.

We compare our approach PS-PGSM with the original offline framework OS-PGSM by training the models offline on data points making first 10 or 15 iterations of incremental training (it differs depending on the length of the dataset being used). These models are then used to predict on remaining data points batch-wise taking 110 points at a time which corresponds to minibatches from the last 7 iterations of prequential training.

A. Dataset description

A detailed description of all the datasets used in the experiments is as follows:

1) *Datasets for preliminary analysis:* To ensure consistency and our ability to directly compare the results, we used 10 of the exact same real-world datasets from the original work [22]. The 2 datasets for which we present a detailed analysis of the results are *Total rents* from Bike sharing dataset and *Temperature T4* from the Energy dataset which are described below. Additionally, we evaluate on 8 more datasets from [22] (Refer Supplementary material of [22]). We use *Total rents* and *Amount Registered* from bike dataset. We use a trimmed version containing first 2500 instances of *Temperature T4*, *Temperature T5*, and *Humidity RH1*, *Humidity RH2* values from the Energy dataset. *NASDAQ* and *RUSSELL* datasets are also used which include various features of these indices from 2010 to 2017 logged at per day frequency. Also, *AbnormalHeartbeat* dataset is used. All these datasets are derived from a variety of different domains and are highly relevant to the task of time series forecasting in real-world applications.
   - **Total rents**: This dataset is taken from the bike dataset. Bike dataset comes from [10] which collects the usage log of bike sharing system in Washington, D.C.,USA over a period of 2 years, i.e., 2011 and 2012. It captures the trip information including bike rental start date and start station, end date and end station, duration of journey, type of member: casual and registered and weather information like temperature. The data is recorded both on an hourly and daily basis. The hourly time series consists of 17379 records in total out of which only data corresponding from January 1 to March 1, 2011 is used in [22] to ensure computational efficiency.
   - **Temperature T4**: This dataset is extracted from the Energy dataset from the UCI Machine Learning repository. It contains data for energy used by appliances in a low energy building. The temperature values are logged at every 10 min from Jan. 11, 2016 17:00 to May 27, 2016 18:00.
2) *Big Data datasets:* To verify the preliminary results from smaller datasets hold for big data, we perform experiments using 2 real-world big datasets commonly used to benchmark in streaming context namely *Household Power Consumption* taken from the UCI repository [9] and *NYC-Bike* from [16]. Both datasets contain large number of instances of the order of 800,000 to mimic big data in a streaming context.
   - **Household Power Consumption**: This dataset contains measurements of electric power consumed in one household over a span of 47 months from December 2006 to November 2010. It consists of 9 attributes with a mix of electrical quantities like global active power, global reactive power and voltage along with values of 3 different sub meters. We use the global active power which measures the global minute-averaged active power (in kilowatt) of the household. The total

number of instances in this dataset are 2075259, out of which we use 800,000 values.

- **NYC-Bike**: This dataset contains bike sharing data from the New York City. It consists of trip data with the following fields: trip duration, trip start time and stop time, start station and end station longitudes and latitudes along with user type and gender. We use the data corresponding to trip duration from July 1, 2013 to July 31, 2013 with a total of 800,000 instances.

3) *Synthetic time series data with Concept Drift:* We generate a synthetic time series dataset using the python package TimeSynth[1]. In this dataset, we concatenate sequences of 500 samples for each of the 4 different concepts namely auto-regressive, harmonic, gaussian and Mackey-Glass signals. Each sequence represents a different concept where white gaussian noise has been injected along with the time series values. Concepts are repeated in random order to create a dataset of 7000 samples.

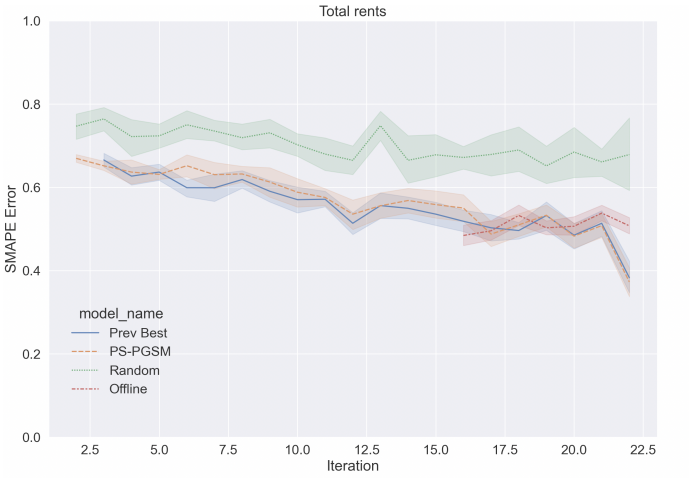

Fig. 2. Comparative results of SMAPE using PS-PGSM, Random, Previous Best and Offline for Total rents dataset

V. Results and Evaluation

In this section, we present the results of our prequential training and ROC computation approach on all the datasets described in the previous section. We first show the summary of the preliminary results for all 10 datasets in Table I followed by in depth findings presented using plots for *Total rents* and *Temperature T4* datasets. The results obtained on big data are documented in Table IV. Finally, the performance analysis is evaluated on the synthetic time series dataset with concept drift. SMAPE error is used to compare the performance of our approach with the original method and the baselines. Based on the preliminary results observed on the 10 small datasets, we only run the experiments for PS-PGSM and previous best model for big data. We skip the experiments with Random model selection as it clearly performs worse than all the other baselines on all the 10 datasets. We do not run the experiments for offline OS-PGSM using big data as it takes prohibitively more time to execute. A computational time study is given in the next section for more details.

[1]Available at https://github.com/TimeSynth/TimeSynth

A. Preliminary results for 10 datasets

Table I shows a summary of the preliminary results which includes the mean and standard deviation of the SMAPE obtained for training and testing phase for all 10 datasets across the 4 approaches. Only for the purpose of performance comparison of the prequential and offline approach, we show the average results split into the training and testing phase which implies the offline version was trained during the training phase while PS-PGSM was trained incrementally across all iterations making up the training and testing phase. A detailed analysis of *Total rents* and *Temperature T4* datasets is presented below, which shows that PS-PGSM provides better results compared to offline OS-PGSM with a drop of 3.45% and 10.86% in the mean error for both datasets respectively. The same observations can be derived from the average SMAPE values for the remaining 8 datasets as shown in Table I. We observe a significant improvement of 63.26% in the error rate when using PS-PGSM for *Humidity RH1* dataset. However, for datasets like *Amounts registered*, the improvement in the mean error is negligible at 1.23%. We also observe that the previous best model performs better than PS-PGSM in 3 out of the total 10 datasets which needs further investigation. Overall, we obtain roughly 40-45% accuracy improvement in the mean SMAPE using our approach, PS-PGSM over offline one across all datasets.

1) *Analysis of Total rents dataset:* In Figure 2, we show the performance of PS-PGSM and the baselines, the solid line for each shows the mean of SMAPE error across 20 runs with the shaded area depicting the 95% confidence interval. The randomly selected forecaster performs worst with the highest SMAPE across iterations which is expected. To produce a fair comparison with the original framework, we first train it for 15 iterations and use the selected model from offline OS-PGSM to predict on the same minibatches as used in the prequential approach from iterations 16 to 22. As a consequence, the red dotted line depicting the SMAPE for offline OS-PGSM starts at the 16th iteration only. Figure 3 displays box plots to show how the average error varies across the different approaches both during training and testing phase. It is clearly demonstrated that the model selected using our prequential approach at each iteration produces lower SMAPE error after 20 iterations compared to the model selected through offline OS-PGSM which was trained on instances up until 15th iteration. Mean SMAPE for PS-PGSM is 0.4920 while its equal to 0.5096 for offline OS-PGSM which is a percentage drop of 3.45% only. However, this improvement in the behaviour can be attributed to the fact that the models trained using our prequential approach are sufficiently trained in the first 15 iterations to provide better results towards the end. We also notice an interesting observation from Figure 2,

TABLE I
PRELIMINARY RESULTS FOR 10 DATASETS

Dataset	SMAPE	Model Type							
		Random		Prev Best		PS-PGSM		Offline	
		Training	Testing	Training	Testing	Training	Testing	Training	Testing
Total Rents	Mean	0.7164	0.6737	**0.5871**	**0.4901**	0.6070	0.4920	-	0.5096
	Std dev	0.0911	0.1276	**0.0703**	**0.0839**	0.0754	0.0875	-	0.0526
Temperature T4	Mean	0.7106	0.6819	0.4302	0.4953	**0.3681**	**0.3691**	-	0.4141
	Std dev	0.2231	0.2390	0.2031	0.2477	**0.1750**	**0.1799**	-	0.1603
AbnormalHeartbeat	Mean	0.7031	0.7175	**0.6279**	**0.6333**	0.6334	0.6385	-	0.7042
	Std dev	0.0762	0.0751	**0.0692**	**0.0818**	0.0567	0.0821	-	0.0487
Humidity RH1	Mean	0.7034	0.6387	0.4702	0.2991	**0.3698**	**0.2238**	-	0.5233
	Std dev	0.2191	0.2827	0.2566	0.1964	**0.1821**	**0.1297**	-	0.2190
Humidity RH2	Mean	0.6868	0.6792	0.4311	0.3089	**0.3949**	**0.2356**	-	0.6414
	Std dev	0.2253	0.2750	0.2340	0.2030	**0.2133**	**0.1388**	-	0.2648
Temperature T5	Mean	0.6982	0.6731	0.4583	0.4849	**0.3841**	**0.3466**	-	0.4939
	Std dev	0.2326	0.2543	0.2572	0.2466	**0.1968**	**0.2054**	-	0.2298
NASDAQ	Mean	0.7219	0.6366	0.4347	0.2476	**0.3284**	**0.1849**	-	0.4956
	Std dev	0.2805	0.3144	0.2356	0.1822	**0.1505**	**0.1440**	-	0.1713
RUSSELL	Mean	0.7110	0.6772	0.4111	0.2834	**0.3111**	**0.2087**	-	0.3650
	Std dev	0.2819	0.2804	0.2377	0.1880	**0.1481**	**0.1531**	-	0.1368
Amounts registered	Mean	0.7215	0.6775	**0.5874**	0.4991	0.6096	**0.4954**	-	0.5016
	Std dev	0.0861	0.1354	**0.0583**	0.0855	0.0703	**0.0932**	-	0.0393
Temperature	Mean	0.6018	0.4727	0.1619	0.1129	**0.1605**	**0.0978**	-	0.1637
	Std dev	0.3237	0.3584	0.1189	0.0658	**0.1173**	**0.0616**	-	0.2011

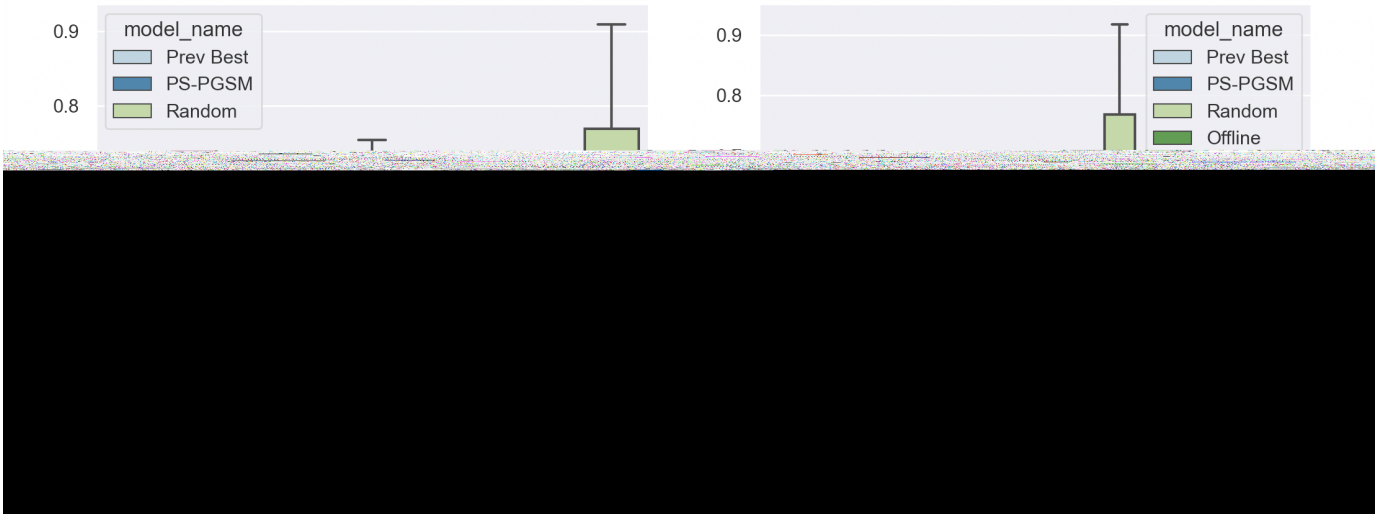

Fig. 3. Box plots showing the average error for Total rents dataset during training (first 15 iterations) and testing (last 7 iterations)

TABLE II
FREQUENCY OF THE MODELS SELECTED BY EACH OF THE APPROACHES IN THE LAST FIVE ITERATIONS FOR TOTAL RENTS DATASET

Dataset	Approach	Model					
		C_0	C_1	C_2	C_3	C_4	C_{10}
Total Rents	Random	1	1	1	1	-	1
	Prev Best	-	-	4	-	1	-
	Offline	-	-	-	-	5	-
	PS-PGSM	-	-	3	-	2	-

which shows that previous best model performs at par with our approach owing to the fact that the current dataset being used shows a uniform trend with very minor variability.

Figure 4 captures the ROCs computed for different models online in the last five iterations (iterations 18 - 22) of a single run for *Total rents* dataset using PS-PGSM. The x and y axis correspond to timesteps and target value y respectively. Table II shows the frequency of the models which were selected by all the approaches during those iterations. We can see from Table II, model C_2 was selected 3 times by PS-PGSM, in Fig 4, those are iterations 18, 20 and 22. Likewise, C_4 gets selected twice, once in iteration 19 and then again in iteration 21. In iteration 18, C_2 gets selected over C_5 as the ROCs of model C_2 are closest to the input sequence in the time series compared to ROCs corresponding to model C_5. It is important to note here that some models do not

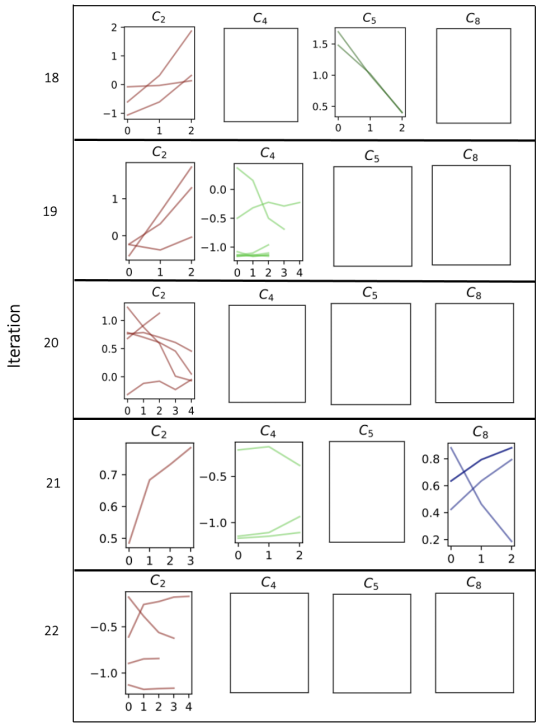

Fig. 4. Plot showing the ROCs for different models in the last 5 iterations for Total rents dataset.

have an ROC after applying the saliency maps due to two reasons: first, a particular model is never selected in the evaluation phase before ROC computation or second, the value obtained from saliency maps is too small to create a pattern and therefore it gets filtered out.

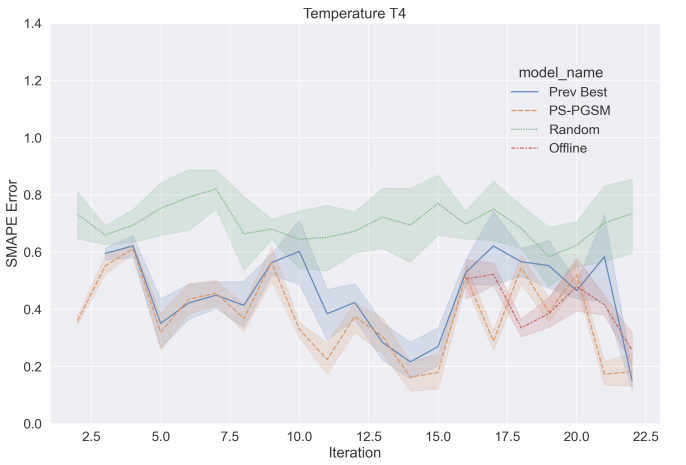

Fig. 5. Comparative results of SMAPE using PS-PGSM, Random, Previous Best and Offline for Temperature T4 dataset.

Therefore, we display only the models with ROCs in the plots and leave all the remaining models. However, in the offline OS-PGSM approach, model C_4 is selected once at the time of prediction at 16th iteration and is then used to predict for all the remaining instances. We also notice that previous best selects C_2 and C_4 which results in similar performance as above to our approach for this particular dataset. As expected, Random always selects a different model at each iteration. At this point, our prequential approach proves beneficial over offline one because ROC computation happens at each iteration to find the model with ROC closest to incoming data pattern resulting in lower error rate and adaptive to the changing trends in the incoming data.

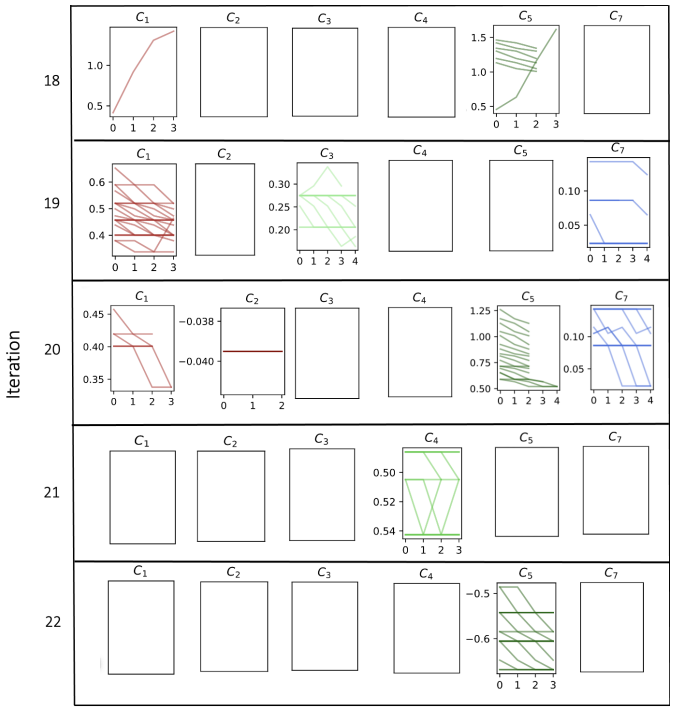

Fig. 6. Plot showing the ROCs for different models in the last 5 iterations from top to bottom for Temperature T4 dataset.

2) *Analysis of Temperature T4 dataset:* Figure 5 shows the comparative results for Temperature T4 dataset for the 4 approaches. It clearly shows that the model selected using PS-PGSM performs better than both the baselines: *Random* and *Prev Best*. Figure 7 uses box plots to further zoom in to the average error and shows that our PS-PGSM approach indeed gets better than the offline OS-PGSM. The mean SMAPE for PS-PGSM is 0.3691 compared to offline OS-PGSM which is 0.4141. This clears shows that the average error drops by 10.86% using PS-PGSM.

We further analyse the model selection for a single run on Temperature T4 dataset and plot the ROCs computed in the last 5 iterations using PS-PGSM as shown in Figure 6. Likewise, Table III shows the frequency of the models selected by the different approaches during those iterations. The offline OS-PGSM version chooses C_5 to predict for all the instances comprising the testing phase i.e., 16-22 iterations. However, we observe that PS-PGSM selects C_1, C_3, C_7, C_9 and C_5 respectively for the last five (18-22) iterations shown in Figure 6. The same is indicated in Table III. These selections for each

TABLE III
FREQUENCY OF THE MODELS SELECTED BY EACH OF THE APPROACHES IN THE LAST FIVE ITERATIONS FOR TEMPERATURE T4 DATASET

Dataset	Approach	Model									
		C_0	C_1	C_2	C_3	C_5	C_6	C_7	C_8	C_9	C_{11}
Temperature T4	Random	-	-	-	-	1	1	1	1	-	1
	Prev Best	1	1	3	-	-	-	-	-	-	-
	Offline	-	-	-	-	5	-	-	-	-	-
	PS-PGSM	-	1	-	1	1	-	1	-	1	-

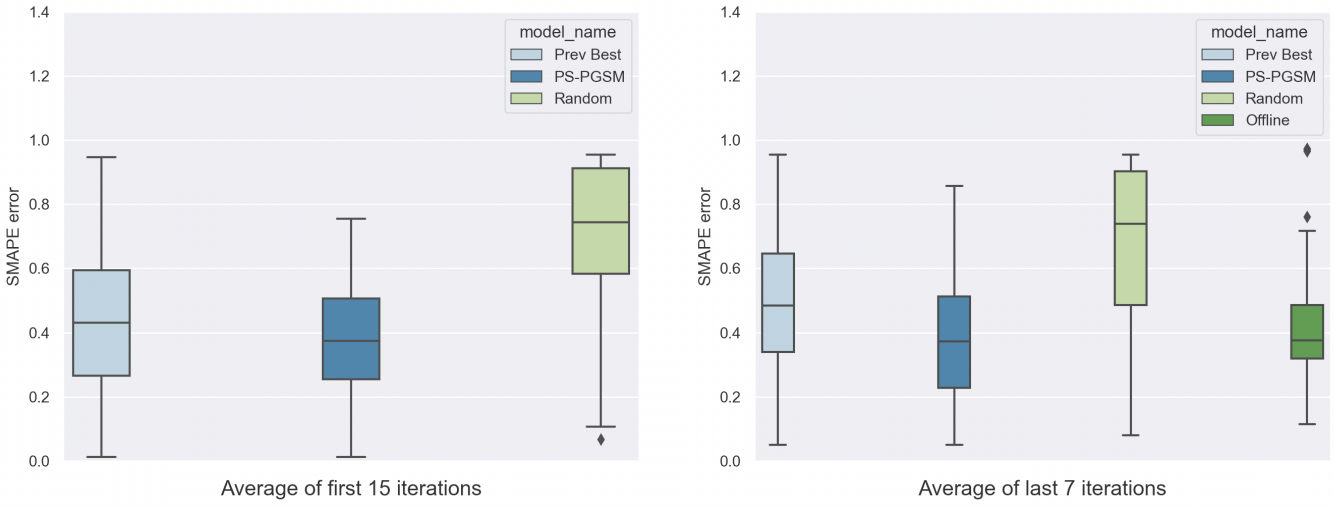

Fig. 7. Box plots showing the average error for Temperature T4 dataset during training(First 15 iterations) and testing(last 7 iterations)

iteration depend on the model having ROC closest to the minibatch for that iteration. As stated above, each model can have multiple ROCs but only a single model is selected for each iteration. For example, in iteration 20, C_7 gets selected over C_1, C_2 and C_5 as the ROCs of C_7 are closest to the input sequence being predicted for. Random baseline always selects a different model which are not necessarily the same models selected by other approaches. We observe that offline OS-PGSM selects C_5 across all 5 iterations whereas our approach selects a different model at each iteration which clearly results in lower average SMAPE and percentage drop of 10.8% over offline OS-PGSM as mentioned in the above paragraph.

TABLE IV
RESULTS FOR BIG DATA

Big Dataset	**SMAPE**	**Model Type**	
		Prev Best	**PS-PGSM**
House Power Consumption	Mean	0.1537	0.1293
	Std dev	0.1257	0.1050
NYC Bike	Mean	0.7143	0.7155
	Std dev	0.1063	0.1061

B. Analysis of experiments on Big Data

Table IV shows the mean and standard deviation of SMAPE obtained after running the experiments on the 2 big datasets

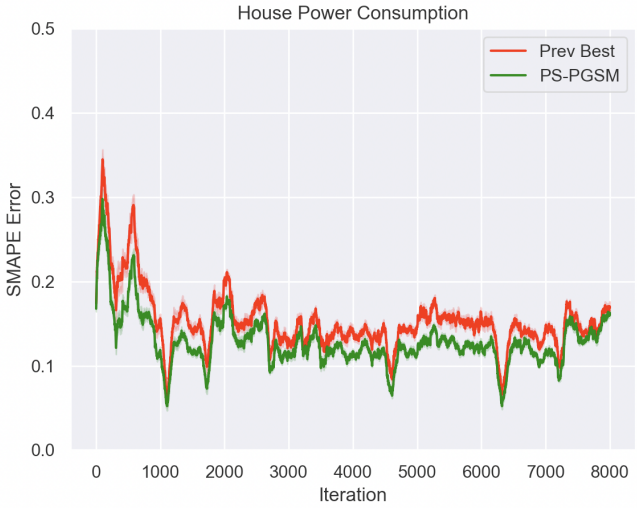

Fig. 8. Moving Average plot for Household Power Consumption dataset.

mentioned in the previous section. The results on big data confirm the results obtained by running on preliminary datasets. Our approach, PS-PGSM consistently outperforms in case of big data as well with lower average SMAPE when compared to the predictions from previous best forecaster. Figure 8 shows the moving average SMAPE over 8000 iterations for both PS-PGSM and previous best forecaster for House Power

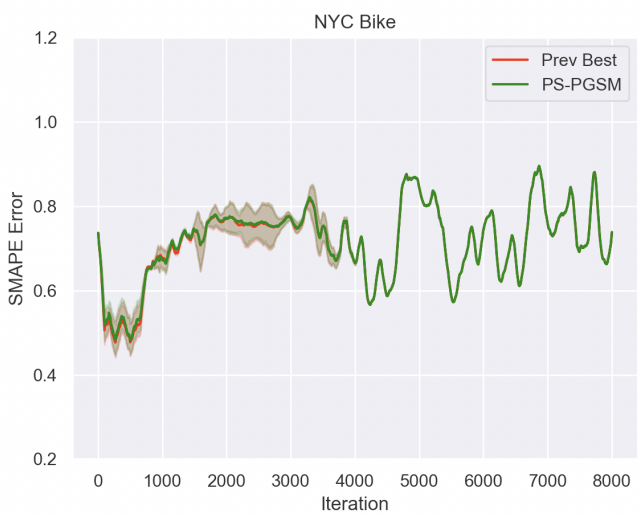

Fig. 9. Moving Average plot for NYC Bike dataset.

Consumption dataset. It can be clearly seen that PS-PGSM performs consistently better than previous best forecaster on House Power Consumption dataset. For the NYC bike data, we see that previous best forecaster performs just as good as our approach as shown in Figure 9. However, as will see in the next section, our approach is still better as it adapts well to concept drift compared with offline OS-PGSM and previous best forecaster.

TABLE V
COMPUTATIONAL TIME STUDY

Model Type	Computational Time (min)	
	120 Iterations	8000 Iterations
Prev Best	2.02	134.66
PS-PGSM	4.47	298.65
Offline	17.05	1136.66

C. Analysis of experiments on Synthetic Time Series

We perform experiments using synthetic time series modelled with concept drift to account for the dynamic nature of temporal data streams in real-world where concept changes and recurrence of specific concepts is most likely to occur. We observe that our approach is capable of handling concept drift better than the original framework OS-PGSM which employs a drift detection algorithm. This can be attributed to the incremental training approach that we have adopted. Figure 10 shows the SMAPE obtained for the 3 approaches on a synthetic time series dataset composed of 4 different concepts as used in [3]. The concepts change at every 5 iterations/500 data points as indicated with the vertical lines on the graph. The 4 concepts shown in the graph repeat in random order after 20th iteration until the 70th iteration. This means each concept is repeating 2-3 times in the dataset containing a total of 7000 data points. Similar to the preliminary experiments we divide 70 iterations into training and testing to compare with offline OS-PGSM. Therefore, the blue line in the plot starts only from 33rd iteration as the testing phase comprises iterations from 33-70. The graph shows evident peaks in the average SMAPE each time a new concept is introduced in the data stream. PS-PGSM clearly outperforms offline OS-PGSM which has a drift detection mechanism in place. The average SMAPE for offline is 0.5531 compared to 0.3969 for PS-PGSM which is a drop of 28.24% in the average error rate. This shows our approach is better at handling concept drift on streaming data without the need of an explicit drift detection algorithm which makes it less computationally expensive to implement in streaming context. PS-PGSM also shows a percentage drop of 13.07 when compared with previous best during the testing phase. This clearly demonstrates that PS-PGSM is robust to concept changes providing better performance compared with all of the baselines.

D. Computational Time Analysis for Big Data

Table V shows the time taken to run the experiments on big data by the various approaches. All the experiments are performed in Python 3.7 environment running on MacBook Pro with 2.4 GHz processor and 32 GB RAM. We have not performed the experiments using offline OS-PGSM for 8000 iterations so the time shown in the table is just an estimated projection of the actual time it would have taken to run it for 8000 iterations. This can be mainly attributed to the process of drift detection which enriches the offline ROCs calculated in the original work. Also, the training in OS-PGSM takes 100 epochs as compared to 1 epoch in our case. However, this makes the execution process prohibitively slower. The table shows that our approach is much faster compared to the offline OS-PGSM taking almost one-third of the time.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed an online approach for model training and ROC computation using saliency maps. The results discussed in the previous section provide evidence that prequential approach provides better results in the context of streaming data. It is observed that in a real-world application generating data continuously, we often do not have the freedom and resources required to store and process historical data. Our prequential approach is particularly suited for such scenarios where it shows promising results compared to the original framework which comprises offline training and ROC calculation. In the future work, we will focus on improving the current algorithm by computing ROCs only when required as opposed to calculating them at each iteration. Another direction to explore will be to define a strategy/threshold to select the most appropriate forecaster among the models with closest ROC and the previous best model.

REFERENCES

[1] Marco Aiolfi and Allan Timmermann. Persistence in forecasting performance and conditional combination strategies. *Journal of Econometrics*, 135(1):31–53, 2006.

[2] Roy Assaf and Anika Schumann. Explainable deep neural networks for multivariate time series predictions. In *IJCAI*, pages 6488–6490, 2019.

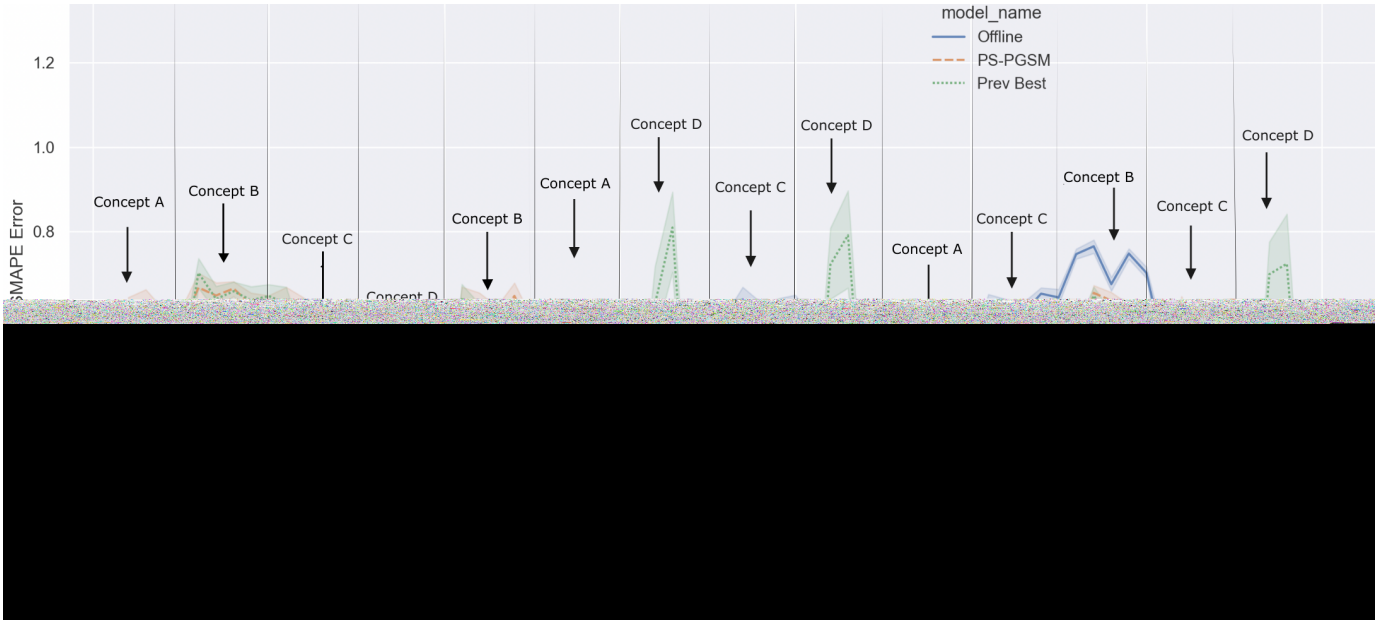

Fig. 10. Comparative results of SMAPE using PS-PGSM, Previous Best and Offline for Synthetic Time Series with Concept Drift.

[3] Dihia Boulegane, Albert Bifet, and Giyyarpuram Madhusudan. Arbitrated dynamic ensemble with abstaining for time-series forecasting on data streams. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 1040–1045. IEEE, 2019.

[4] Gavin Brown and Ludmila I. Kuncheva. "Good" and "Bad" Diversity in Majority Vote Ensembles. pages 124–133. Springer, 2010.

[5] Roberto Casado-Vara, Angel Martin del Rey, Daniel Pérez-Palau, Luis de-la Fuente-Valentín, and Juan M Corchado. Web traffic time series forecasting using LSTM neural networks with distributed asynchronous training. *Mathematics*, 9(4):421, 2021.

[6] Vitor Cerqueira, Luis Torgo, and Igor Mozetič. Evaluating time series forecasting models: An empirical study on performance estimation methods. *Machine Learning*, 109(11):1997–2028, 2020.

[7] Vítor Cerqueira, Luís Torgo, Fábio Pinto, and Carlos Soares. Arbitrated ensemble for time series forecasting. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 478–494. Springer, 2017.

[8] Rafael MO Cruz, George DC Cavalcanti, and Tsang Ing Ren. A method for dynamic ensemble selection based on a filter and an adaptive distance to improve the quality of the regions of competence. In *The 2011 International Joint Conference on Neural Networks*, pages 1126–1133. IEEE, 2011.

[9] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[10] Hadi Fanaee-T and Joao Gama. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, 2(2):113–127, 2014.

[11] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37, 2014.

[12] Moncef Garouani, Adeel Ahmad, Mourad Bouneffa, Mohamed Hamlich, Gregory Bourguin, and Arnaud Lewandowski. Using meta-learning for automated algorithms selection and configuration: an experimental framework for industrial big data. *Journal of Big Data*, 9(1):1–20, 2022.

[13] Moncef Garouani, Adeel Ahmad, Mourad Bouneffa, Arnaud Lewandowski, Grégory Bourguin, and Mohamed Hamlich. Towards the automation of industrial data science: A meta-learning based approach. In *ICEIS (1)*, pages 709–716, 2021.

[14] Zeineb Hammami, Moamar Sayed-Mouchaweh, Wiem Mouelhi, and Lamjed Ben Said. Neural networks for online learning of non-stationary data streams: a review and application for smart grids flexibility improvement. *Artificial Intelligence Review*, 53(8):6111–6154, 2020.

[15] Carson K Leung, Fan Jiang, and Yibin Zhang. Explainable machine learning and mining of influential patterns from sparse web. In *2020 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, pages 829–836. IEEE, 2020.

[16] Mingzhe Liu, Bowen Du, and Leilei Sun. Co-prediction of multimodal transportation demands with self-learned spatial dependence. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 824–833. IEEE, 2021.

[17] Viktor Losing, Barbara Hammer, and Heiko Wersing. Incremental online learning: A review and comparison of state of the art algorithms. *Neurocomputing*, 275:1261–1274, 2018.

[18] Mustafa V. Nural, Hao Peng, and John A. Miller. Using meta-learning for model type selection in predictive big data analytics. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 2027–2036, 2017.

[19] Ismini Psychoula, Andreas Gutmann, Pradip Mainali, Sharon H. Lee, Paul Dunphy, and Fabien Petitcolas. Explainable machine learning for fraud detection. *Computer*, 54(10):49–59, 2021.

[20] Pablo Romeu, Francisco Zamora-Martínez, Paloma Botella-Rocamora, and Juan Pardo. Time-series forecasting of indoor temperature using pre-trained deep neural networks. In *International conference on artificial neural networks*, pages 451–458. Springer, 2013.

[21] André Luis Debiaso Rossi, André Carlos Ponce de Leon Ferreira, Carlos Soares, Bruno Feres De Souza, et al. MetaStream: A meta-learning based method for periodic algorithm selection in time-changing data. *Neurocomputing*, 127:52–64, 2014.

[22] Amal Saadallah, Matthias Jakobs, and Katharina Morik. Explainable online deep neural network selection using adaptive saliency maps for time series forecasting. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 404–420. Springer, 2021.

[23] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[24] David H Wolpert. The supervised learning no-free-lunch theorems. *Soft computing and industry*, pages 25–42, 2002.

[25] Kevin Woods, W. Philip Kegelmeyer, and Kevin Bowyer. Combination of multiple classifiers using local accuracy estimates. *IEEE transactions on pattern analysis and machine intelligence*, 19(4):405–410, 1997.

[26] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.