Towards Increasing Open Data Adoption Through Stream Data Integration and Imputation

Robert Kunicki[1,2] and Maciej Grzenda[1(✉)]

[1] Faculty of Mathematics and Information Science, Warsaw University of Technology, ul. Koszykowa 75, 00-662 Warszawa, Poland
{R.Kunicki,M.Grzenda}@mini.pw.edu.pl
[2] Digitalisation Department, The City of Warsaw, pl. Bankowy 2, 00-095 Warszawa, Poland

Abstract. Open data portals are used to make a growing number of government data resources public. However, difficulties in preprocessing and integrating multiple possibly incomplete open data resources hinder the potential of open data re-use for software development. While incomplete data sets can be imputed in an offline process, this is not the case for data streams expected to be published in an online manner.

In this work, we propose a novel data stream preprocessing method aimed at simplifying open data stream re-use through the unification of time resolution and imputation of incomplete instances. The method relies on stream mining methods to predict categorical values to be imputed. A separate online learning model is built for every incomplete feature. The method we propose allows the model to benefit from both inter-feature similarities and temporal dependencies present in data streams. We validate the proposed method with public transport data streams.

Keywords: Data preprocessing · Stream mining · Open data

1 Introduction

Smart city solutions are largely dependent on the availability of sensor data. To promote their development, numerous IoT platforms are deployed. Irrespective of the IoT platforms involved, it is the data collected by these platforms that is of particular value both for the cities and the providers of novel software. To promote civic engagement and foster the development of novel applications and services, open data portals have been started by numerous cities [10], which make urban data available to the general public. In the case of data sets, the data supposed to be made public could be prepared in an *off-line* process prior to their publication. Hence, some data sets such as those made public at the *Find Open Data* portal (http://data.gov.uk) contained in-filled data in cases where no observed data was available[1].

[1] Highways Agency network journey time and traffic flow data was an example of such a data set.

H. Fujita et al. (Eds.): IEA/AIE 2021, LNAI 12798, pp. 15–27, 2021.
https://doi.org/10.1007/978-3-030-79457-6_2

More recently, growing interest in online analytics (rather than periodic batch processing of the data), when coupled with the growing availability of sensors operated by the cities, has resulted in data streams shared through open data platforms. Many such data streams come from IoT sensors, whereas the data from IoT platforms are known to be noisy, and incomplete. There can be many reasons for such quality issues including the inability to transmit data, breakdowns of data collection platforms, and failures of metering devices. Low quality data requires even higher technical abilities to overcome its deficiencies. At the same time, insufficient technical abilities were observed to hinder the actual use of open data [8].

As many IoT data resources such as bus location data are of limited value if not published online, improving in an online manner the quality of the data, including the elimination of gaps resulting from missing and delayed sensor readings, turns out to be of the utmost importance for the actual software development. Not surprisingly, machine learning methods can be used for this purpose. In particular, rapid growth of stream mining methods [2,4], i.e. machine learning methods focused on data streams rather than data sets, has been observed in the recent period. However, in spite of the developments in classification and regression techniques for data streams [2–5], many issues still have to be addressed, such as the development of stream imputation techniques.

In this study, we focus on the provisioning of open data streams while increasing their quality in an on-line manner. The primary objective is to simplify the use of the data by software developers. This is done by proposing a novel stream-oriented data preprocessing method, which can be used to develop in an online manner instances integrating a number of the most recent preprocessed readings from IoT sensors with standardised time resolution and context data. Importantly, when gaps in raw time series of readings are observed, we propose a novel imputation method for data streams which relies on data stream classification methods and aims to substitute missing categorical values with estimated predicted values. As a consequence, the augmented data stream includes imputed instances with standardised time resolution.

Furthermore, to assess the performance of the data imputation method we propose, we ran a number of experiments with real data streams and stream mining techniques. The main motivation for these studies was to investigate how preprocessed data affects the accuracy of predictions of deviations from the planned arrival time of public transport vehicles. Since the true but missing values are not known in real scenarios, we focused on checking whether imputed data does not hinder prediction of delays, which is one of the key applications of the reference data streams we used in this study.

The remainder of this work is organised as follows. In Sect. 2, we provide an overview of stream data imputation in relation to stream mining methods, which is followed by Sect. 3, discussing the reference problem of incomplete location data streams. In Sect. 4, we propose a stream data preprocessing method, and discuss the results of applying it for a number of reference data streams in Sect. 5. Finally, conclusions and possible future works are outlined in Sect. 6.

2 Related Works

Streaming data is characterised by infinite size and speed of data growth. This makes the use of traditional methods of data set processing inappropriate. It is then necessary to use special techniques and algorithms that process data at the time of inflow while taking into account time and memory constraints. Hence, stream mining methods [2] have been developed. The most important requirements for the stream mining methods are [2]: to process every instance at most once, rely on constrained used of memory and processing power, and adapt to temporal changes.

There are three possible reasons for missing values in data [11]: missing completely at random (MCAR) - the occurrence of missing value in the instances does not depend on either the observed data or the missing value, missing at random (MAR) - the occurrence of the absence depends on the available data but not on the missing value, and not missing at random (NMAR) - the occurrence of the absence is related to the missing value. As gaps and delays in IoT-originated data acquisition are frequently due to device or transmission failures, they can be classified as being largely due to MCAR reasons. Handling of incomplete data can rely on one of three methods [7]: simple deletion, data imputation or special treatment. The first method involves removing all instances that contain missing entries, and the third involves dealing with deficiencies by choosing methods capable of processing incomplete data. The second category, i.e. data imputation, includes methods based on both statistical and machine learning approaches. Not surprisingly, it is the data imputation approach that is of particular value in the case of the provisioning of preprocessed open data accompanying the provisioning of raw open data.

Until now, in Massive Online Analysis (MOA) [2] i.e. the key stream mining platform, data imputation methods based on both statistical features of the attribute values observed so far in a stream (such as mean and median value) and the last known value of an attribute have been proposed. These methods include imputation with last known value (LKV) of an attribute, mean, maximum and minimum value for numerical attributes, and mode (MOD) for categorical data.

Another approach is to use machine learning methods to predict the actual value of an attribute and use it in place of the missing value [9]. Importantly, due to the characteristics of streaming data, there are special requirements for machine learning methods applicable for data streams [2], including periodic updates of a model, and its adaptation to nonstationary processes (concept drift) [4]. Hence, batch machine learning methods such as MLP networks can not be directly applied to impute data streams rather than data sets. As a consequence, works on stream mining are largely focused on the processing of the streams of complete instances [2–4]. In particular, in [5] i.e. a related study on prediction models for IoT public transport data streams, only complete instances were used for the prediction of the delay status of public transport vehicles.

3 Motivation

Open data portals are used to make public a growing number of data resources, including data sets and data streams. As of the time of writing this article, Greater London Authority (https://data.london.gov.uk) had published 1489 data sets, while San Francisco had published 1209 data sets (https://datasf.org). Importantly, open data is an enabler for the development of innovative software applications [6]. However, lack of validity and completeness of data, and lack of technical interoperability were already observed to be some of the most commonly identified barriers that hinder the generation of value from open data [8]. At the same time, open time series data were among these open data resources, which were most frequently accessed by software developers, as pointed out in the analysis of applications developed with open data [6]. Hence, increasing the quality and usability of open data streams, including IoT data streams is of particular importance. However, with this large number of data sets made public, data preprocessing methods should ideally be automated.

Let us discuss a sample use case of the imputation of public transport location data streams i.e. the location of vehicles. Similarly to other IoT sensor readings, the time between consecutive GPS sensor readings received by IoT platforms and made available as open data varies. This situation is illustrated in Fig. 1. It can be observed that the period $t(v_{i+1}) - t(v_i)$ between two consecutive sensor readings v_i and v_{i+1} acquired from the GPS sensor varies. In some cases $t(v_{i+1}) - t(v_i)$ exceeds 30 s. The second situation includes a lack of data due to the fact that the readings do not reach the server-side IoT platform in the assumed time. This is because the time between reading the data at a sensor and receiving these data in the system collecting data such as an open data portal is non-negligible.

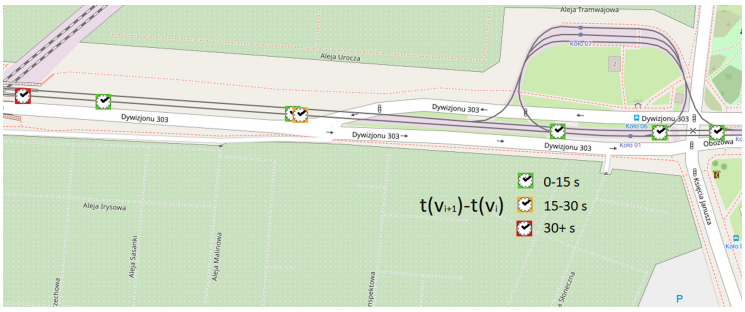

Fig. 1. Different times of recording the GPS location. Based on location data stream for Warsaw trams from 5$^{\text{th}}$ Jan. 2020.

This illustrates the fact that making raw IoT data streams available for online processing results in major data preprocessing burden for third-party software developers, who frequently have to try to skip excessive readings at some periods and perform the imputation of gaps at other periods. The key need to be

answered is to provide sensor readings for standardised time resolution e.g. one minute time resolution, possibly combined with context data and with imputed values, when needed.

4 Data Integration and Imputation Method

To overcome the problem of incomplete IoT data streams affected by varied time resolution, we propose integrating raw sensor readings into instances containing a number of recent consecutive sensor readings with unified time resolution and context data, including imputed feature values in case the raw ones were not available within the expected time resolution. The Unified Time-Resolution Integration and Imputation of Data Streams (UTRIIDS) method proposed in this work consists of two stages: the unification of time resolution and integration stage, and the data imputation stage.

Input: $I(i)$ - raw stream of sensor readings, Δ_{R} - acceptance time, N - the number of previous instances that we aim to integrate with the current reading while taking into account Δ_{R}, δ - time resolution of output instance, $\Delta_{\mathrm{P}} \geq 0$ - prediction horizon
Output: D - merged data stream composed of $\mathcal{S}_i$ instances
begin
  `/* for every received sensor reading */`
  for $i = 1, \ldots$ **do**
    $\mathcal{S}_i.\mathbf{x} = I(i)$;
    $\mathcal{S}_i.\mathbf{x} = \mathcal{S}_i.\mathbf{x}.Append\_Additional\_Features(I(i))$;
    `/* Find` N `readings closest to` $t-\delta, t-2\delta, \ldots, t-N\delta$ `*/`
    for $k = 1, \ldots, N$ **do**
      if *Exists* $I(j) :\mid t(I(j)) - k\delta \mid \leq \frac{\Delta_{\mathrm{R}}}{2}$ **then**
        $\tilde{I}_k = I(m) :\mid t(I(m)) - k\delta \mid = min\big(\{\mid t(I(j)) - k\delta \mid, j < i\}\big)$;
        $\tilde{I}_k.Append\_Additional\_Features(\tilde{I}_k)$;
      else
        $\tilde{I}_k =$ NULL;
      $\mathcal{S}_i.\mathbf{x} = \mathcal{S}_i.\mathbf{x}.append(\tilde{I}_k)$;
    end
    `/* Add` $\mathcal{S}_i$ `to output stream */`
    $D.Submit(\mathcal{S}_i)$;
    `/* Add now known true label values for` Δ_{P} `prediction horizon for previously processed instances */`
    $D.Submit(\mathcal{S}_j.\mathbf{y}) : j < i, t(\mathcal{S}_j) < t(\mathcal{S}_i) - \Delta_{\mathrm{P}} + \frac{\Delta_{\mathrm{R}}}{2}$;
  end
end

Algorithm 1: UTRIIDS method: the unification of time resolution and data integration stage.

Unlike standard methods of data stream imputation, which rely on simple statistics of attribute value such as the aforementioned methods available in the MOA environment, the method we propose relies on machine learning methods to impute missing values. Importantly, taking into account the unbounded number of instances of a data stream, we propose the use of stream mining methods for data imputation. The applicable methods include the streaming version of Naive Bayes, Hoeffding Trees (HT) [3], Hoeffding Adaptive Tree (HAT) [1] and the recently proposed adaptation of the random forest method to stream processing needs i.e. Adaptive Random Forest (ARF) [4].

The first stage of the UTRIIDS method is defined in Algorithm 1. The objective of this stage is to develop from raw sensor readings $I(i)$, the instances $\mathcal{S}_i$ of the output stream D. Every instance $\mathcal{S}_i$ includes the current sensor reading and N previous readings separated by approximately δ seconds. One of the key settings of UTRIIDS is acceptance time Δ_{R}. If no sensor reading acquired from a sensor in the period $[t-k\delta-\frac{\Delta_{\mathrm{R}}}{2}, t-k\delta+\frac{\Delta_{\mathrm{R}}}{2}], k = 1, \ldots, N$ exists in $I(i)$, readings for both the k-th preceding time step and dependent data such as vehicle delay status of a vehicle for the k-th step will be considered missing. Otherwise, sensor readings from the time closest to $t - k\delta$ will be placed in the $\mathcal{S}_i$ instance. It is important to note that the dependent data such as delay status calculated based on location readings can be even more important in practical cases. Moreover, Δ_{R} controls the precision of time resolution; a low value results in an increased proportion of missing entries.

Furthermore, context data for a stream of interest, such as the tram line the vehicle is serving now, which can be appended to instance data based on schedule data for location data streams, can be added to every instance $\mathcal{S}_i$ developed in Algorithm 1. Finally, the algorithm can be used to develop instances, which can be used next to predict future values for the stream of interest i.e. predict the $\mathcal{S}(j).\mathbf{y}$ labels before they are available. As an example, the location of a vehicle or delay status of the vehicle in approx. Δ_{P} seconds can be predicted. Hence, every time a new reading is received from a sensor, the true label(s) to previous instance(s) can be added to be used to verify the quality of such predicted labels.

A data stream produced in Algorithm 1 is typically incomplete. This is because of possibly major time differences between consecutive sensor readings, i.e. raw $I(i)$ readings periodically not matching the time resolution and acceptance levels, defined by δ and Δ_{R}, respectively. Hence, the second stage of the UTRIIDS method, which we defined in Algorithm 2, can be used to impute incomplete instances and add predicted labels $\tilde{y}_i$. For every input feature j of $\mathcal{S}_i$ instances, which may be incomplete, a prediction model is developed with a stream mining method. Based on the remaining features of an instance and the data of U previous instances and U previous instances for which the value of this attribute was known, the attribute model M_j is built. Hence, the attribute model can learn both the relation between attributes of the current instance and possibly preceding instances and the attribute of interest to possibly exploit temporal dependencies present in the data.

Input: $\mathcal{S}_1, \mathcal{S}_2, \ldots$ - a data stream, E - ML method used to train models performing data imputation, M - ML method used to train models used to predict future values, U - the number of previous instances used to predict attribute values to be imputed
Data: $L_j, j = 1, \ldots, J$ - sliding windows of U most recent instances with known value of attribute j, $P = [\mathcal{S}_{i-1}, \mathcal{S}_{i-2}, ..., \mathcal{S}_{i-U}]$ - sliding window of U previous instances, I_m - merged instance, A_i - imputed instance
begin
  /* For every instance in stream */
  for $i = 1, \ldots$ **do**
    $A_i = \mathcal{S}_i.\mathbf{x}$;
    /* For every attribute of the instance */
    for $j = 1, \ldots$ **do**
      $I_m = [\mathcal{S}_i.\mathbf{x}, P_1, \ldots, P_U, L_j[1], \ldots, L_j[U]]$;
      if $A_i[j]$ *!= NULL* **then**
        /* If the attribute is not empty update list with the instance */
        $L_j.Insert(\mathcal{S}_i)$;
      if $A_i[j] = NULL$ **then**
        /* Perform imputation */
        $A_i[j] = M_j.Predict(I_m)$;
      else
        /* Update model using merged instance as an input and known value of j-th attribute as a true label */
        $M_j = E.Update(M_j, I_m, A_i[j])$
    end
    $P.Insert(A_i)$; $\tilde{y}_i = \mathcal{M}.predict(A_i)$; $Publish([A_i, \tilde{y}_i])$;
    $\mathcal{M} = M.Update(\mathcal{M}, A_i, \mathcal{S}_i.y)$;
  end
end

Algorithm 2: UTRIIDS method: data imputation stage

When the value of an attribute j is known, the model M_j is updated in an incremental manner to learn the dependencies between other attributes of the current instance and previously observed attributes and the value of attribute j. Otherwise, the $M_j()$ model is used to predict the value to be imputed and used instead of the missing attribute value.

Let us note that each of the possibly incomplete attributes necessitates a separate model. However, all M_j models are stream mining models i.e. models with limited memory and computational cost. Their number is no larger than the constant $J = dim(\mathcal{S}.\mathbf{x})$ i.e. the number of all attributes in an instance. Moreover, the question of how to evaluate the quality of imputed instances arises, taking into account that the missing attribute values are not known. We propose that, since every instance $\mathcal{S}_i$ can include a label $\mathcal{S}_i.y$, the quality of imputed $\mathcal{S}_i$ instances can be assessed by evaluating their impact on the performance of the $\mathcal{M}$ models predicting the $\mathcal{S}_i.y$ labels, while using imputed instances as an input. This performance can be compared to the performance of the models predicting the $\mathcal{S}_i.y$ labels while using incomplete data as input data.

5 Results

To analyse the UTRIIDS method, reference data streams were selected. We focused on tram location data from the City of Warsaw to verify the method. These data streams are available via the Open Data portal of the City of Warsaw and were used *inter alia* in [5,6]. Raw data include vehicle identification such as tram line and brigade number, location and GPS timestamp. Once integrated with schedule data, they can be extended with additional features such as vehicle delay. All the data streams described here were processed with the UTRIIDS method. For each reading I, $N = 5$ preceding readings were sought as defined in Algorithm 1, assuming $\delta = 60\,\mathrm{s}$ resolution. In this way, merged instances $\mathcal{S}_i$ spanning the period of the last 5 mins., and including location data and dependent data from $N + 1 = 6$ time steps, were developed. In addition, tram delay status in $\Delta_\mathrm{P} = 60\,\mathrm{s}$ was denoted as $\mathcal{S}_i.y$. Five vehicle statuses were adopted: `accelerated` (delay of $[-\infty, -60]\,\mathrm{s}$), `on time` - delay of $(-60, 60)\,\mathrm{s}$, `small delay`, `delay`, and `major delay` for delays in the range of $[60, 120)$, $[120, 240)$, and $[240, \infty)$, respectively. The acceptance time Δ_R was set to different values, as described in Table 1, which describes all data streams developed with Algorithm 1. The streams collected in different time periods were used. The tram line (L) and brigade numbers (B) for every stream are given in the L/B column. Different Δ_R settings were applied to analyse the impact of the acceptance level on the proportion of missing attribute values in $\mathcal{S}$ instances. The decreasing Δ_R value affects the number of stream instances. This is due to the necessity to have a true label $\mathcal{S}.y$ for a $[t(I(i)) + \Delta_\mathrm{P} - \frac{\Delta_\mathrm{R}}{2}, t(I(i)) + \Delta_\mathrm{P} + \frac{\Delta_\mathrm{R}}{2}]$ period. Let us note that this is only because of developing ground truth labels that will be used to evaluate the merits of imputed values by analysing the accuracy of predicted labels $\tilde{y}_i$. Otherwise, the number of instances for varied Δ_R, but the same line, brigade and period would be the same. Furthermore, the proportion of I readings that arrived to server platform collecting data from GPS sensors after Δ_t is reported. It follows from Table 1 that for every data stream, over 90% of readings arrived after more than 5 s, which confirms the limited timeliness of the data. Moreover, the small Δ_R values show the tradeoff between unifying time resolution and completeness of $\mathcal{S}_i$ instances.

Table 1. Data streams used for the evaluation of UTRIIDS method.

Data stream	Δ_R	L/B	Data period	Records	Missing data [%]	Late data for Δ_t of 5,10,15,20,30 s [%]
W6_A_20	6	23/1	1st Jan.–30th Jun.2020	112,722	14.40	94, 49, 5, 1, 1
W6_B_20	6	24/2	2nd Feb.–1st May 2020	50,407	14.68	91, 36, 5, 1, 1
W10_B_20	10	24/2	2th Feb.–1st May 2020	83,454	10.02	91, 37, 5, 1, 1
W6_C_20	6	26/1	1st Mar.–1st Jun.2020	83,780	14.01	91, 36, 5, 1, 1
W10_C_20	10	26/1	1st Mar.–1st Jun.2020	155,932	8.82	91, 37, 5, 1, 1
W16_C_20	16	26/1	1st Mar.–1st Jun.2020	191,927	4.44	93, 43, 7, 2, 1

The question arises of how to evaluate the quality of data imputed by stream mining methods. As we focused on the imputation of categorical features such as tram delay status, we performed a number of experiments with standard imputation techniques i.e. mode and LKV of an attribute. The results of the imputation performed using these methods were compared with the results of applying stream mining methods to provide imputed values, as defined in Algorithm 2. In the experiments performed with Algorithm 2 and the data described in Table 1, ARF, Naive Bayes, kNN, HT and HAT methods were used as the M method, while ARF and Naive Bayes were used as the E method and $U = 1$ was used in Algorithm 2. In this way, probabilistic, instance-based, tree-based and ensemble models were included. Furthermore, the simple strategies of re-using the last known label as a prediction for the next instance i.e. the NoChange (NC) method and Majority Class (MC), i.e. the use of the most frequent label as a predicted label, were used as the M method. Notably, the two latter techniques produce labels $\tilde{y}_i$, which do not depend on the $\mathcal{S}_i.\mathbf{x}$ features i.e. the $\tilde{y}_i$, labels, which are the same irrespective of what missing features values were replaced with. Hence, the results for these two techniques are reported in Table 2. MOA was extended in this study to provide the implementation of Algorithm 2. All results were obtained with standard implementation of the M methods available in MOA.

Table 2. The accuracy of predicted labels for two simple M methods.

M Method	W6_A_20	W6_B_20	W10_B_20	W6_C_20	W10_C_20	W16_C_20
NC	89.88	88.80	92.48	89.98	94.01	94.73
MC	68.66	64.01	63.75	67.96	67.88	67.71

It follows from Table 2 that the majority of the y labels ranging between 63.75% and 68.66% can be predicted by using the most frequent class as a prediction. An even higher proportion of labels can be predicted with NoChange i.e. by re-using the previous label in a stream, which illustrates the temporal dependencies frequently observed in IoT data streams. The accuracy of the predictions made with imputed data by the remaining M methods is presented in Table 3. As far as imputation methods are concerned, NOT denotes no imputation, whereas LKV and MOD denote the use of the last known value, and the mode of the attribute for the imputation, respectively. The UTRIIDS used with ARF and Naive Bayes, denoted by U_ARF and U_NB, was used to predict the missing attribute values to be imputed. Let us note that when a method such as NoChange is used as the E method, we get the same results for U_NCH as when using LKV for the imputation, which we show in Table 3.

It follows from Table 3 that for highly correlated consecutive labels, the simple NoChange strategy can yield superior accuracy of M predictions, irrespective of whether data imputation is performed or not. This shows that even largely

Table 3. Accuracy of predicted labels i.e. vehicle delay statuses for M and E methods.

Data stream	Main method M	Imputation method E					
		LKV	MOD	U_ARF	U_NB	U_NCH	NOT
W6_A_20	Naive Bayes	54.52	53.85	53.70	**55.15**	54.52	54.03
	ARF	87.83	87.83	87.84	87.84	87.83	87.73
	kNN	77.74	74.26	76.43	**78.34**	77.74	75.87
	Hoeffding Tree	87.70	87.78	87.78	87.78	87.70	87.60
	HAT	86.77	87.38	**87.38**	86.53	86.77	86.68
W6_B_20	Naive Bayes	44.46	43.24	43.71	**45.02**	44.46	44.16
	ARF	86.51	86.50	86.51	86.49	86.51	86.40
	kNN	75.59	71.16	73.64	**75.87**	75.59	72.77
	Hoeffding Tree	86.46	86.41	86.42	86.41	86.46	85.90
	HAT	86.28	85.87	86.25	86.30	86.28	85.89
W10_B_20	Naive Bayes	**44.26**	43.26	43.73	43.86	**44.26**	43.91
	ARF	86.12	86.14	86.19	86.11	86.12	85.97
	kNN	**77.50**	74.66	74.66	76.99	**77.50**	75.16
	Hoeffding Tree	86.10	86.14	86.13	86.13	86.10	85.90
	HAT	83.92	83.47	83.87	**85.05**	83.92	84.97
W6_C_20	Naive Bayes	36.95	34.75	**37.26**	35.55	36.95	35.43
	ARF	86.12	**86.15**	86.14	86.14	86.12	85.94
	kNN	**75.61**	73.02	75.35	75.31	75.61	73.18
	Hoeffding Tree	86.04	86.04	86.03	86.03	86.04	85.81
	HAT	**86.03**	85.84	85.84	85.84	86.03	85.95
W10_C_20	Naive Bayes	**35.00**	33.29	34.58	34.16	**35.00**	33.93
	ARF	**86.46**	86.37	86.40	86.35	**86.46**	86.31
	kNN	**77.98**	77.32	76.37	77.54	**77.98**	76.48
	Hoeffding Tree	85.96	85.96	85.95	85.96	85.96	85.94
	HAT	84.92	84.33	84.90	85.22	84.92	85.42
W16_C_20	Naive Bayes	**34.55**	33.65	34.35	33.70	**34.55**	34.20
	ARF	87.00	86.97	86.97	87.00	87.00	86.98
	kNN	**80.11**	79.45	79.84	79.91	**80.11**	79.54
	Hoeffding Tree	86.28	86.30	86.30	86.27	86.28	86.16
	HAT	**85.61**	85.52	85.44	85.20	**85.61**	84.74

incomplete instances could be accepted from the prediction performance point of view. However, this would not answer the need for imputed open data.

Still, even under highly correlated consecutive labels, the use of imputation can yield a slight increase in the accuracy of prediction. What follows from Fig. 2 is that even though the overall accuracy of predicted labels attained with

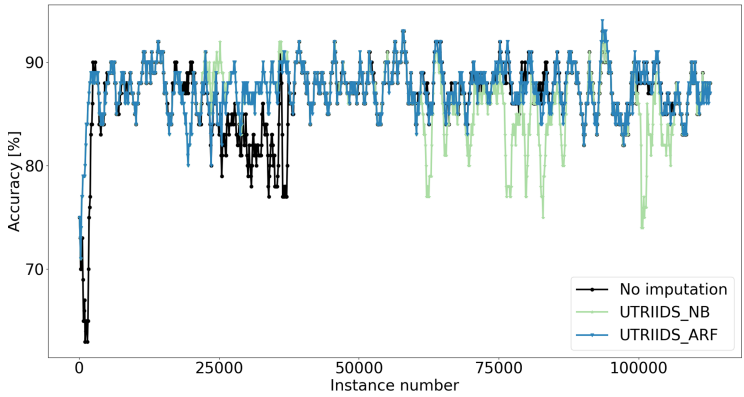

Fig. 2. Prediction accuracy for W6_A_20 data. Main method M: HAT.

Table 4. The average κ coefficient of predicted labels for different M and E methods

Main prediction method M	Imputation method E					
	LKV	MOD	U_ARF	U_NB	U_NCH	NOT
Naive Bayes	24.44	23.39	23.75	23.93	24.44	23.85
ARF	74.20	74.18	74.23	74.18	74.20	73.95
NoChange	82.91	82.91	82.91	82.91	82.91	82.91
Majority Class	0.02	0.02	0.02	0.02	0.02	0.02
kNN	45.00	34.29	41.34	47.17	45.00	39.41
Hoeffding Tree	71.60	71.60	71.60	71.61	71.60	71.22
HAT	69.00	68.97	69.17	69.33	69.00	69.22

imputed data may be similar, lack of imputation may result in major periodic degradation of accuracy, which can be avoided when UTRIIDS_ARF is used. In addition to the increase in the accuracy of prediction, a significant benefit itself is the imputed data stream, which makes software development with the imputed data stream much easier.

The question remains of how to evaluate the quality of the imputed data stream in the presence of highly correlated consecutive true labels making NC the best prediction method. In order to answer this question, we propose using the values of Cohen's κ coefficient [2] provided in Table 4 for the scenarios already described in Table 3. Importantly, κ shows the scale of improvement that a model $\mathcal{M}$ provides over a random prediction method. It follows from Table 4 that UTRIIDS compared to NOT yields the same or higher performance of prediction models built with different methods. The largest performance gains are observed when U_NB imputes data used with kNN. This can be explained by the fact that tree-based models can overcome incomplete data in one attribute by using the data from other attributes. At the same time, kNN explicitly aims to

use the data of all attributes, which makes it particularly vulnerable to incomplete data. Hence, the quality of the imputed data stream can be assessed with the κ measure calculated for distance-based classifiers.

6 Conclusions

Modern IoT platforms provide growing volumes of sensor data, which enable novel software services. However, this raises the question of the quality of the data shared in the open data model. In this study, we addressed one of the main features of IoT data streams, i.e. their limited quality. The method we propose provides integrated instances with unified time resolution, making it possible to obtain a number of the most recent sensor readings, features based on them, and the value of a predicted feature. An important part of the method is the imputation of missing attribute values. Experiments with a number of data streams developed under different quality settings show that the method not only provides imputed instances, but also increases the quality of the data compared to no imputation. Importantly, as IoT data streams frequently exhibit temporal dependencies, it is the κ measure rather than accuracy that can be used to evaluate the quality of imputed data by analysing its impact on the performance of prediction models. Moreover, in the future, the development of stream mining methods combining data imputation with exploiting temporal dependencies in the data is planned.

Acknowledgements. The project was funded by the POB Research Centre for Artificial Intelligence and Robotics of Warsaw University of Technology within the Excellence Initiative Program - Research University (ID-UB).

References

1. Bifet, A., Gavaldà, R.: Adaptive learning from evolving data streams. In: Adams, N.M., Robardet, C., Siebes, A., Boulicaut, J.-F. (eds.) IDA 2009. LNCS, vol. 5772, pp. 249–260. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03915-7_22
2. Bifet, A., Gavaldà, R., Holmes, G., Pfahringer, B.: Machine Learning for Data Streams with Practical Examples in MOA. MIT Press, Cambridge (2018)
3. Domingos, P., Hulten, G.: Mining high-speed data streams. In: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2000, pp. 71–80. ACM, New York (2000)
4. Gomes, H.M., et al.: Adaptive random forests for evolving data stream classification. Mach. Learn. **106**(9), 1469–1495 (2017)
5. Grzenda, M., Kwasiborska, K., Zaremba, T.: Hybrid short term prediction to address limited timeliness of public transport data streams. Neurocomputing **391**, 305–317 (2020). https://doi.org/10.1016/j.neucom.2019.08.100
6. Grzenda, M., Legierski, J.: Towards increased understanding of open data use for software development. Inf. Syst. Front. **23**(2), 495–513 (2019). https://doi.org/10.1007/s10796-019-09954-6

7. Grzymala-Busse, J.W., Grzymala-Busse, W.J.: Handling missing attribute values. In: Maimon, O., Rokach, L. (eds.) Data Mining and Knowledge Discovery Handbook, pp. 37–57. Springer, Boston (2005). https://doi.org/10.1007/978-0-387-09823-4_3
8. Jetzek, T., Avital, M., Bjorn-Andersen, N.: Data-driven innovation through open government data. J. Theor. Appl. Electron. Commer. Res. **9**(2), 100–120 (2014). https://doi.org/10.4067/S0718-18762014000200008
9. Miao, X., Gao, Y., Guo, S., Liu, W.: Incomplete data management: a survey. Front. Comput. Sci. **12**(1), 4–25 (2018). https://doi.org/10.1007/s11704-016-6195-x
10. Thorsby, J., Stowers, G.N., Wolslegel, K., Tumbuan, E.: Understanding the content and features of open data portals in American cities. Gov. Inf. Q. **34**(1), 53–61 (2017)
11. Yu, Q., Miche, Y., Eirola, E., van Heeswijk, M., Séverin, E., Lendasse, A.: Regularized extreme learning machine for regression with missing data. Neurocomputing **102**(C), 45–51 (2013)