Does imputation work for improvement of domestic hot water usage prediction

Steffen Moritz, Thomas Bartz-Beielstein, Olaf Mersmann, Martin Zaefferer, Jörg Stork

Fakultät für Informatik und Ingenieurwissenschaften, FH Köln
Steinmüllerallee 1, 51643 Gummersbach
Tel.: +49 2261 8196-0
Fax: +49 2261 8196-15
E-Mail: steffen.moritz10@gmail.com

1 Introduction

The "Internet of Things" - the connection of everyday objects to the internet - is claimed to be one of the most important future trends. More and more domestic devices like refrigerators, stoves, smoke detectors, television or even lamps are meanwhile available with integrated internet connectivity. However the pure ability to connect to the internet is only one part. Customers expect some extra value like smart functions from these devices. Providing these smart functions often goes along with building predictive models on the data recorded by these devices. Because of the huge amount of accruing data and occurring problems like missing data this can be quite a challenging task. Especially missing data is a quite common phenomenon, because multiple possible reasons can lead to data gaps.

In this paper we take up this issue and have a look on how different imputation methods to replace missing values influence the outcome of afterwards applied predictive models. For our experiments we used recorded and anonymized data from about 600 connected heating systems. We apply different imputation methods like EM, k-NN and regression based algorithms in order to fill in missing values and afterwards apply a model (e.g. SVM, Neural Net, Random Forest based models) that does a prediction for the customers domestic hot water usage for the upcoming week. In our experiments we want to show the interdependencies between imputation algorithm and prediction model and we want to clarify the question, if imputation in this case is a possible way to improve prediction accuracy.

The remainder of the paper is structured as follows. Section 2 provides an insight into the specific problem we want to solve. The section is followed

by a short summary of the research questions and goals in Section 3. Section 4 gives an overview on previous research and methods. In Section 5 follows a description of the performed experiments, while in Section 6 the results of the experiments are analyzed. A . The paper closes with short summary and outlook in Section 7.

2 Problem Description

2.1 Motivation

As already mentioned in the introduction, the "Internet of Things" enables several new applications from just giving simple status information up to intelligent functions. The recorded data of the networked everyday objects are hereby often the enabler for new applications based on predictive algorithms. But with the network character and the placement at normal households problems with the data recording come along. In comparison to installations in labs or other managed surroundings there are plenty of uncontrollable influences, which can lead to gaps in the data logging. These logging gaps can become a problem later on, because the predictive models used for intelligent functions work best with clean and complete datasets. Missing data in the input of predictive algorithms very likely leads to poorer results or can even lead to no results at all. Thats why finding out how to handle missing data can be useful.

In our specific practical case we are looking at in this paper, we are using anonymized data of about 600 connected heating systems. All the heating installations of our dataset are placed at real customer households. In contrast, the data storage itself is not at the customers home, it is at a server back-end. Which means in our case: the data gets recorded at the customers heating system, is then sent wireless to a connected device, which transfers the data via the customers router and the internet to the server back-end. This is done this way, because of several practical reasons like more computing power on the back-end and too small storage in the heating system itself. But this also leads to more vulnerabilities for the possible data losses. Possible causes for missing data can occur on the whole chain towards the server back-end. This can be issues with the sensors of the heating system itself, problems with the wireless reception, disturbances of the customer's internet connection, the customer switching the router off or even a failure with the server back-end. Partially these problems get addressed with technical solutions. But all in all we can see from the data

we got, that missing data is quite a common phenomenon. So also for this specific real-life problem it is very interesting to address the problem of how to handle missing values and data gaps.

2.2 Dataset

We used anonymized data from about 600 connected heating systems.The timespan of the recorded data goes from December 2013 till end of July 2014. But not for every heating system installation the complete timespan of data is available. Most of the systems start and end logging at different times. So we do have different timespans of recorded data for different heating systems.

The original data that arrives at the server back-end consists out of more than fifty different attributes. These are technical variables which give information about the internal status of the heating system. These are for example informations about temperatures and flags about current working modes of components of the heating system. Imagine the data looking like in table 1 just with many more attributes.

Date Time	Temp. 1	Temp. 2	DHW Req.	further
02/02/2014 14:00:01	60.1	50.3		...
02/02/2014 14:00:05	60.1	50.3		...
02/02/2014 14:00:11	60.0	50.3		...
02/02/2014 14:00:15			1	...
02/02/2014 14:00:17	59.9	50.3		...
02/02/2014 14:00:20			0	...

Table 1: Example extract of the data from one heating installation

As you can see, there is data being recorded nearly every second. The timestamps are not regularly spaced, which means, the time distances between the rows may differ. There are quite a lot of empty rows in the table, but this mustn't be misunderstood as missing values. Empty values in this specific case also just can mean there is no difference to the last broadcasted value. To reduce the amount of data we extract just the data we need to train and built our predictive model out of the complete data. For our experiments we will only need two of the more than fifty attributes - the timestamp and a attribute called DHW Request. The attribute DHW Request (domestic hot water request) indicates when a customer uses domestic hot water. Every time there is a domestic hot water usage, this flags

turns to 1 and afterwards again to 0. To get the dataset we need, we calculate for every heating system the amount of daily DHW Requests and save this together with the timestamp. What we get are about 600 datasets (one for each gateway) looking like table 2. The table can be read like this: 'There where 21 usages of domestic hot water on the second of February 2014 for the shown installation.'

Date Time	Sum DHW Requests
02/02/2014	21
02/03/2014	29
02/04/2014	22
02/05/2014	5
02/06/2014	1
02/07/2014	5
02/07/2014	10
...	...

Table 2: Data in the format used for our experiments

Because we want to compare similar timespans, we define 04/01/2014 and 07/30/2014 as start and end point of our datasets. This means we have to spare all gateways with shorter logging timespans. We also have to spare gateways with missing data in this period. We want to compare imputation methods later on in our experiments and will therefore delete data in a controlled way to measure the performance. In case of missing data being already present, before we apply our missing data mechanism, this could distort the process and results. After taking out the heating systems with data that does not fit these criteria, there are 170 systems left, we can look at.

Shown in a graphical way, our created timeseries looks like in figure 1. Looking at figure 1, we can see there are, depending of the day, up to forty hot water usages a day. But there are also days, where there has been no hot water usage at all.

3 Questions and Goals

What is interesting at our datasets, is, that we have a interesting real-life problem to look at. We have a real-life scenario, where missing data might be a real issue and good ways to handle them are really useful.

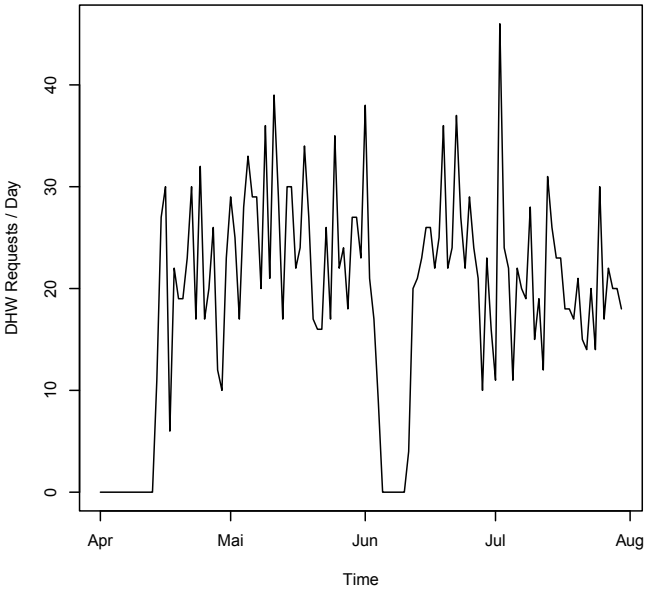


Figure 1: Timeseries of daily sums of DHW Requests for one heating system

What we want to achieve in this paper is, to look on the impact of missing data on predictive functions built on our specific data. To do this we use the preprocessed dataset as described in the 'Dataset' section of our paper. As an example for a predictive function we try to make a 14 day forecast of the amount of DHW requests. This predictive function is realized with different models like decision tree, multiple regression, support vector machines, exponential smoothing and a naive model.

Our first step then is to compare the results of the different models on complete datasets. Our second step will be to take out data from the datasets in a controlled way to look how this influences the performance of the models. In the third step we will also take out data, but afterwards impute them and again look how this influences the results.

The research questions we want to answer are:

1. What is the correlation between the amount of missing data and the performance of the predictive models

2. What general problems occur with missing data
3. Can imputation help to increase the performance of the predictive models
4. Are there best fits for prediction algorithm and imputation algorithm combinations

As performance metric for the comparison of our results we will use the standard metrics RMSE and MAE.

MAE The *mean absolute error* (MAE) between the predicted time series $\hat{y}$ and the respective true DHW Request time series y., i.e.,

$$\mathrm{MAE}(\hat{y}, y) := \frac{\sum_{t=1}^{n} |\hat{y}_t - y_t|}{n}$$

RMSE The *root mean square error* (RMSE) between the predicted time series $\hat{y}$ and the respective true DHW Request (test) time series y, i.e.,

$$\mathrm{RMSE}(\hat{y}, y) := \sqrt{\frac{\sum_{t=1}^{n} (\hat{y}_t - y_t)^2}{n}}$$

4 Previous Research and Methods

4.1 Imputation Methods

When missing data occurs, it may hinder the training of suitable prediction models. In some cases, it may be feasible to simply stop predicting new data once data-samples are missing. But often, especially in technical processes, a more continuously working system is needed.

Friese et al. [1] give an overview of several imputation methods. One important distinction, is whether multi-variate or uni-variate data is predicted. In the multi-variate case, missing data from one signal may be repaired by employing data from other, correlated signals. In the uni-variate case, methods can only rely on information and structure contained in the observed signal.

For uni-variate cases, the most simple method to replace missing data is Last Observation Carried Forward (LOCF). Here, a missing value is replaced by the last observed value. While being conceptually simple, this

has the advantage of being easy to calculate. As such, it will be used in this study. The herein described work will also use methods from the `mice` R-package[1] [2] that collects several methods for multivariate (as well as uni-variate) imputation.

We use:

mean Imputation of unconditional mean values.

norm.predict Imputation with linear regression models.

rf Random Forest is a tool for regression and classification that is based on ensembles of decision trees. It has been developed by Breiman [3]

Besides, any conceivable regression method can basically be employed for imputation. Research has also been applied to whether the imputation of several values for a single missing data-point may be a feasible [4].

4.2 Forecasting Methods

Various data driven methods can be used for prediction of time series data.

Most of the prediction methods used in this paper are taken from the `rminer` R-package. Only the ETS predictor is used from the `forecast` R-package. In the following, a short description of the employed methods is presented.

naive The naive prediction is simply the average of the observations. It is a simple and easy to implement predictor.

dt Decision Trees (DT) determine the outcome of the prediction (leaves of the tree) by several decisions made based on the presented data. The root of the tree (or first node) is the first decision to make, e.g., whether a value is larger or smaller than a certain threshold. Based on the given data, a branch leads to the next node (a new decision) or to a leave (predicted value). Their very simple structure makes decision trees easy to train. Furthermore, users may easily understand rules presented as decision trees.

[1] R is a programming language for statistical computing, see `http://www.r-project.org/` for further information and downloadable software.

svm Support Vector Machines (SVM) have originally been developed for classification by Cortes and Vapnik [5]. By mapping to a higher-dimensional feature space, SVMs can predict nonlinear data with linear hyperplanes. They have been extended to regression [6], which enables their application in this study.

mr In Multiple Regression (MR), linear effects of vector-valued predictor variables are learned.

ets While the above methods are all intended for regression problems in general, Exponential Smoothing State Space Models are more specifically developed towards predicting time-series data. The herein used methods are based on Work by Hyndman et al. [7].

5 Experiments

After we cleaned and preprocessed the initial data, as described in the section 'Dataset', we have 170 files with data to perform our experiments on. Each one of these files contains the daily domestic hot water requests from 04/01/2014 to 07/30/2014 for one specific heating system installation. In our experiments we want to compare different algorithms performing a 14 day forecast on this data. Further on we want to evaluate how missing values in the training data affect our forecast. Afterwards we check, if imputation is able to improve the results.

The complete procedure of our experiments looks like described in Algorithm 1. For the implementation we used the R programming language. For building forecasting models we relied on rminer and forecast R-packages. The used imputation algorithms were mainly taken out of the mice R-package.

Cluster	Size
Cluster 1	68
Cluster 2	27
Cluster 3	16
Cluster 4	1
Cluster 5	45

Cluster	Size
Cluster 6	3
Cluster 7	5
Cluster 8	3
Cluster 9	1
Cluster 10	1

Table 3: Size of clusters

The first step is a clustering of the 170 input files. The reason for this is, in our experiments we realized quite early, that there is a quite huge difference in behavior and results between the different heating system installations. Single installations had a MAE about ten times higher than others. In an overall evaluation with an averaged MAE these heating systems had a too huge impact compared to others. To fix this issue we are building clusters with similar heating systems. To do this we create a distance matrix, which is based on the euclidean distance between the data from the individual heating systems. From the distance matrix we create 10 clusters using the hclust() R-function. Afterwards heating systems with similar variations in daily domestic hot water usage are grouped together. The resulting clusters can be seen in table 3. As can be seen there are four clusters with more than ten heating installations and six clusters with just one to five installations. Our experiments we performed due to time issues just on cluster number 5.

Our experiment process mainly consists out of five loops, processing all the different combinations of results we want to get. The first loop iterates over the different gateway files, in our tested case this are the 45 files of cluster five. Loop two iterates over the different imputation methods we are testing. The third loop is for variation of the missing data rate. Furthermore we have a loop for choosing different random seeds. This one is to avoid random effects coming from the exponential distribution we use to create the missing data . The last loop iterates over the different models we use to make our predictions. In between the loops the necessary methods for our experiments get called. These are the functions for creating the train/test set, the function for creating the missing values, the function to do the imputation and the functions to train the models and to to the predictions. In the following subsection 'Predictions based on complete datasets' we will take a closer look how in general we make the predictions. In the subsection 'Predictions based on incomplete datasets without imputation' we explain our missing data mechanism. And in the final subsection 'Predictions based on incomplete datasets with imputation' we illustrate what it looks like when we do predictions on imputed datasets.

5.1 Predictions based on complete datasets

To be able to evaluate our predictions later on, we split our data in a training and a testing set. Because we are dealing with time-series, we do not create the test sets by random holdouts. Therefore what we do is, we cut the last 14 days of the time-series and define this as the test data. The rest of the

Algorithm 1: Program Structure

input : List of InputFiles $input$
input : List of Imputation Algorithms $sel.algoImp$
input : List of lamda values for exponential distribution $sel.rate$
input : List of random seeds $sel.seed$
input : List of Prediction Models $sel.model$
output: Result Vector $results$

```
files ← cluster(input)
for i.file in files do
    for i.algoImp in sel.algoImp do
        for i.rate in sel.rate do
            for i.seed in sel.seed do
                trainData ← createTrainData(i.file)
                testData ← createTestData(i.file)
                trainData ← missval(trainData, i.rate, i.seed)
                trainData ← impute(trainData, i.algoImp)
                for i.models in sel.model do
                    model ← fitModel(trainData, i.model)
                    prediction ← predict(model, testData)
                    results ← evaluate(prediction)
                end
            end
        end
    end
end
```

data becomes the training data. During our experiments we were trying about 9 different algorithms:

1. naive
2. naivebayes - naive bayes
3. lr - logistic regression (multinom R-package)
4. lda - linear discriminant analysis (MASS R-package)
5. dt - decision tree (rpart R-package)
6. mr - multiple regression (nnet R-package)
7. bruto - additive spline mode (mda R-package)

8. mars - multivariate adaptive regression splines (mda R-package)
9. knn – k-nearest neighbor (kknn R-package)
10. svm – support vector machine (ksvm R-package)
11. ets - exponential smoothing state space model (forecast R-package)

Most of this algorithms we called using the rminer R-package, which as meta package simplifies the usage of different algorithms. Actually later on in our 'Analysis' section we only used results of five different algorithms. This is because at some point we focused on naive, dt, svm, ets and mr, because these had the best results. The naive algorithm is a good indicator, if a algorithm is a fail for the task. Naive just outputs the mean value as a result and is very good in terms of computing time.

For evaluation of our results, we calculate for every heating installation the MAE and the RMSE for the 14 day predictions.

5.2 Predictions based on incomplete datasets without imputation

To do predictions on incomplete datasets, we have to take out a certain amount of data. We do achieve this by applying a missing data function on the training data. The test dataset of course remains the same. The important part here is the implementation of the missing data function.

To simulate the days to the next failure of logging and therefore loss of data, we use the exponential distribution. Using the exponential distribution we can also vary the failure rate given as λ. If there is a failure, we do not replace the values with *NA*, we completely delete this data. Replacing with *NA* would lead to errors in many of the prediction algorithms. To avoid outliers because of a lucky outcome of the exponential distribution for one run, we perform the same run with different random seeds. In our experiments we use the following failure rates: 0 (means we do not apply the missing data function at all), 0.4, 0.8, 1.2, 1.6. The rates we use are quite high, as can been seen in table 4. For example a rate of 1.6 means there are only 16 rows of the former 107 rows of the training set left.

After we minimized the test dataset we go on as we did with the normal data. We train our models and do the predictions. The only difference is the reduced training set, with which we train the models.

Rate	0	0.4	0.8	1.2	1.6
Rows	107	70	48	29	16

Table 4: Connection between rate and remaining rows

5.3 Predictions based on incomplete datasets with imputation

When doing imputation, we are using the function for creating missing data described in the section above and afterwards perform imputations on the reduced dataset. To be able to do this we don't completely remove the missing values, instead we replace them with *NA*. We are using the following algorithms for imputation.

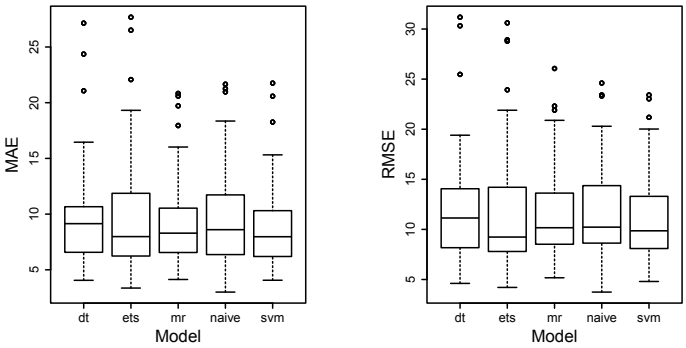
1. norm.predict - Linear regression
2. rf - Random forest imputations
3. mean - Unconditional mean imputation
4. locf - Last observation carried forward

The further steps stay the same as always, we take the reduced and afterwards by imputation filled up training data set to learn our models. Then we predict the following 14 days and calculate the MAE and RMSE.

6 Analysis

We ran our experiments as mentioned in the 'Experiments' section for 45 different heating system installations. Five different prediction algorithms and five different imputation methods were used. We also chose to run with three different random seeds and to take five different missing data rates. This means there were 16875 (45*5*5*3*5) models built and predictions made. Luckily the amount of the training data isn't too high, so that it was possible to perform this with a standard computer within one day.

The first surprise comes along as we look on the data for predictions without missing data in figure 2. None of the algorithms was able to perform significantly better than the 'naive' method. Also the performance of 'ets' who we thought to be especially good for time-series is not better. What also surprises is, that decision tree performs also quite ok compared to the



(a) MAE (b) RSME

Figure 2: MAE and RMSE for different models without missing data

other algorithms. We didn't expect decision tree to gain good results on a univariate time-series. What can't be seen here, we already had taken out some algorithms before our final run, to speed up processing. These were lda, knn, mars, bruto and bayes. Lda, bruto and bayes we took mostly out, because of their bad results that were significantly poorer than the naive algorithm. The rest of these algorithms performed approximately on the same level as the naive and the other algorithms here and were taken out just because of performance reasons. What also can be seen in this graphic, the variations in results for one algorithm are quite huge. There are huge MAE and RMSE outliers in the boxplots. Origin of these variations are not the different random seeds, as could be expected. The reason are the differences between the single gateways.

Interesting now, how the MAE values change if we increase the missing data rate in the training data. This can be seen in figure 3. To avoid confusion, the starting values of the algorithms, that are shown here are not shown in the boxplots of figure 2. In figure 3 the MAE values are the average MAE values for the specific prediction algorithm. The thick line in figure 2 is compared to this the median and not the average. First thing we see here in terms of average MAE, is that svm is the best overall algorithm. The next thing, that is really surprising is, except for ets none of the algorithms perform really bad for high missing data rates. The naive method is very constant over all missing data rates. This is actually not very that surprising, because it takes the average over all data as prediction. The other algorithms results alternate depending on the rate, getting even slightly better sometimes. This effect probably has to do with outliers

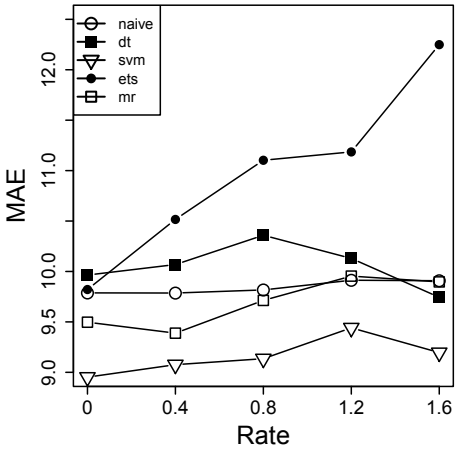


Figure 3: Algorithm performance for different missing data rates

being removed by the missing data function. Considering that a rate of 1.6 means, that just around 16 of 207 values of the training data remain, this the results are anyway quite unexpected.

Knowing, that even a huge missing data rate didn't have much impact on the results, it is already clear, that imputation won't bring a big gain. The question that remains for imputation is now rather: 'Will imputation lead to poorer results?'. This question is still interesting, because sometimes it is the case, that algorithms require a certain maximum of values to run without throwing an error. This was also the reason, we did not take rates higher than 1.6, because this then led to errors. So sometimes it can be useful to impute, just to have enough data. Furthermore it is interesting, weather imputation can bring ets on the level of the other algorithms. Figure 4 and figure 5 show results for different prediction/imputation combinations. The rate thereby stays constant - in figure 4 it is 0.4 and in figure 5 it is 1.2.

The results of the imputations are quite interesting. We have to differentiate between imputation at rate 0.4 and rate 1.2. For rate 0.4 as seen in figure 4 no imputation algorithm leads to significant poorer results than doing no imputation. The interesting thing is now, mean, rf and norm.predict

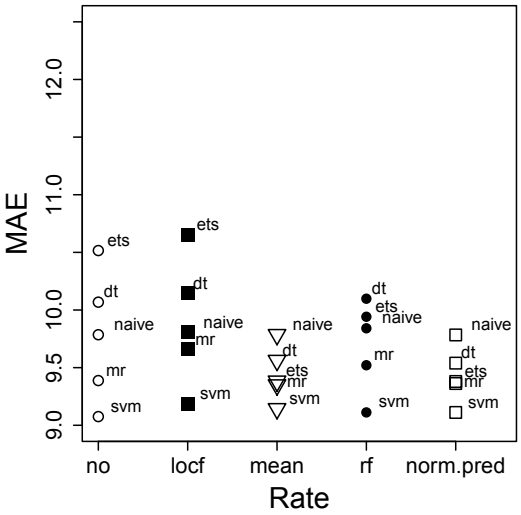


Figure 4: Prediction results for different imputation methods with rate 0.4

influence to prediction in that way, that also ets now reaches the same MAE results as the other algorithms. Last observation carried forward seems to be the worst imputation option. So overall there might be no big improvement, but mean, rf and norm.predict are solid options to use instead of no imputation.

Looking at the imputation with the higher missing rate of 1.2 in figure 5 it looks quite similar. Here it's is even clearer that locf is the worst option. Mean and norm.predict create solid results, being at least as good as the results with no imputation. Even ets has good results for these algorithms. But actually that is quite logical. We know from the naive prediction algorithm that forecasting the mean gives good results. We also do know, that at rate 1.2 already over 50 percent of data is missing and gets imputed. If now all this data gets imputed by mean, also ets will predict something very close to mean. So to sum up the imputation results: imputation does not lead to a new overall best result, but taking the right imputation method, it also does not worsen the results. On the contrary it helps improving results for algorithms that performed poorly before.

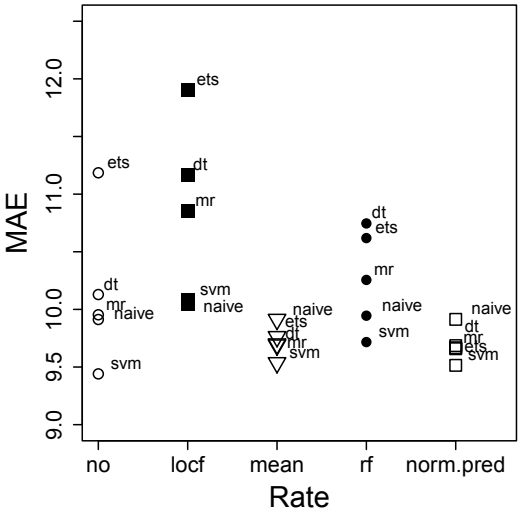


Figure 5: Prediction results for different imputation methods with rate 1.2

7 Summary and Outlook

One major insight we got out of our experiments is, that for this problem it is really hard to surpass the mean calculation of the naive algorithm. Another finding was, that all tested algorithms were extremely robust dealing with high rates of missing data. But we do not credit this to the capabilities of the algorithms - it has rather to do with the specific problem we are working on. With svm being slightly better than the naive algorithm, we figured out a favorite algorithm for this problem. But because of the proximity of the results the naive algorithm would also be a good solution. Our experiments with imputation showed, that most of the imputation methods led to similar results as doing no imputation. This can in some cases be of advantage, because sometimes there is a need of a certain amount of data to be present. Imputation also helped the prediction algorithms that performed poor before to improve their results.

As an outlook it would be interesting, if these results remain stable, even if we change something in our experimental settings. For example we plan to enhance the data set by adding additional attributes like 'day of week'. Another point we would like to check, is if our missing data function reflects reality. A possible way to check this would be to take data with real missing values and look if imputation on this data improves prediction results. Of course it wouldn't be possible to see what the results without missing data would be, but it could be seen weather the results are better with or without imputation. The clustering in front of the processing is also still an open point. In our Analysis we have seen, that there is still a huge variation in the results for different heating systems. Throwing the results of very different heating systems together and building an averaged MAE is not optimal. It would be good to find a method for clustering that better sticks similar heating systems together.

References

[1] Friese, M.; Stork, J.; Guerra, R. R.; Bartz-Beielstein, T.; Thaker, S.; Flasch, O.; Zaefferer, M.: UniFIeD Univariate Frequency-based Imputation for Time Series Data. Techn. Ber., Bibliothek der Fachhochschule Koeln Bibliothek, Betzdorfer Str. 2, 50679 Koeln. URL http://opus.bsz-bw.de/fhk/volltexte/2013/49. 2013.

[2] van Buuren, S.; Groothuis-Oudshoorn, K.: mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software* 45 (2011) 3, S. 1–67. URL http://www.jstatsoft.org/v45/i03.

[3] Breiman, L.: Random Forests. *Machine Learning* 45 (2001) 1, S. 5–32.

[4] Yuan, Y. C.: Multiple imputation for missing data: Concepts and new development (Version 9.0). *SAS Institute Inc, Rockville, MD* (2010).

[5] Cortes, C.; Vapnik, V.: Support-vector networks. *Machine Learning* 20 (1995) 3, S. 273–297. URL http://dx.doi.org/10.1007/BF00994018.

[6] Drucker, H.; Burges, C. J.; Kaufman, L.; Smola, A.; Vapnik, V.: Support vector regression machines. *Advances in neural information processing systems* 9 (1997), S. 155–161.

[7] Hyndman, R. J.; Akram, M.; Archibald, B. C.: The admissible parameter space for exponential smoothing models. *Annals of the Institute of Statistical Mathematics* 60 (2008) 2, S. 407–426.