A Comparison of Data Cleaning Algorithms

Sam Dewar

Submitted in partial fulfilment of
the requirements of Edinburgh Napier University
for the Degree of
Computer Science BSC Hons

School of Computing

April 2023

Authorship Declaration

I, Sam Dewar, confirm that this dissertation and the work presented in it are my own achievement.

Where I have consulted the published work of others this is always clearly attributed;

Where I have quoted from the work of others the source is always given. With the exception of such quotations this dissertation is entirely my own work;

I have acknowledged all main sources of help;

If my research follows on from previous work or is part of a larger collaborative research project I have made clear exactly what was done by others and what I have contributed myself;

I have read and understand the penalties associated with Academic Misconduct.

I also confirm that I have obtained **informed consent** from all people I have involved in the work in this dissertation following the School's ethical guidelines

Signed: Sam Dewar

Date: 11/04/2023

Matriculation no: 40437441

General Data Protection Regulation Declaration

Under the General Data Protection Regulation (GDPR) (EU) 2016/679, the University cannot disclose your grade to an unauthorised person. However, other students benefit from studying dissertations that have their grades attached.

Please sign your name below *one* of the options below to state your preference.

The University may make this dissertation, with indicative grade, available to others.

The University may make this dissertation available to others, but the grade may not be disclosed.

The University may not make this dissertation available to others.

Sam Dewar

Abstract

Data cleaning is an important part of the data pre-processing process. Several kinds of anomaly such as missing values & outliers. Modern techniques allow data scientists to estimate the the true values of these datapoints to repair outlier & impute missing values. This project takes a look at data cleaning algorithms on univariate time series data. With the understanding that k nearest neighbours algorithms are ineffective on this kind of data, they are compared to other algorithms, some similar such as sliding window prediction & iterative impute with KNN regressor as an estimation method, some quite different such as autoregression. It is determined that k nearest neighbours are just as bad as they were thought to be & that autoregression massively outperforms them on this kind of data, while an iterative approach to KNN may still be effective.

Contents

1 INTRODUCTION 10

2 LITERATURE REVIEW 11

2.1 What Is data cleaning 11

2.2 Missing values (MV) 14

2.3 Noisy Data 18

2.4 Algorithm Effectiveness Metrics 23

2.5 Conclusion 26

3 EXPERIMENT 27

3.1 Caltrans Dataset 27

3.2 WUnderground Dataset 27

3.3 With missing values 28

3.4 Without missing values 29

3.5 Experiment Plan 30

3.6 Implementation 34

4 ANALYSIS & EVALUATION 40

4.1 Missing Value Imputation 40

4.2 Outlier Correction 55

5 CONCLUSION & FUTURE WORK 72

5.1 Summary of Findings 72

5.2 Self Appraisal 73

5.3 Further Work 74

REFERENCES 76

List of Tables

Table 1 Variables in AR Equations 19
Table 2 Areas of Comparison of Data Cleaning Algorithms 25
Table 3 Datasets in The Experiment 32
Table 4 SWP Variables 36
Table 5 Low Error Full Data Outputs Missing Value 42
Table 6 Low Error Half Data Outputs Missing Value 45
Table 7 High Error Full Data Outputs Missing Value 48
Table 8 High Error Half Data Outputs 51
Table 9 Missing Value Imputation Full 52
Table 10 Missing Value Imputation Accuracy Scaling 53
Table 11 Missing Value Imputation Time Scaling 54
Table 12 Low Error Full Data Outputs 58
Table 13 Low Error Half Data Outputs Outlier 62
Table 14 High Error Full Data Outputs Outlier 65
Table 15 High Error Half Data Outputs Outlier 69
Table 16 Outlier Repair Full Data 70
Table 17 Outlier Repair Time Scaling 71

List of Figures

Figure 1 EM Pseudocode 14
Figure 2 Multiple Imputation 15
Figure 3 KMI Pseudocode 16
Figure 4 Rough KNN Filter Pseudocode 18
Figure 5 Comparison of IMR, AR & ARX 20
Figure 6 IMR Pseudocode 21
Figure 7 KNN Outlier Detection & SWP on Furniture Sales 22
Figure 8 SWP on Weather Data 22
Figure 9 Dirty Data Injection Tool 33
Figure 10 GUI of The Tool 34
Figure 11 Python GUI 35
Figure 12 Python Timer Recording 35
Figure 13 SWP new Value Pseudocode 36
Figure 14 Python SWP 37
Figure 15 Python SWP Based Imputation 38
Figure 16 Python Autoregression 38
Figure 17 Python Autoregression With Exogenous Input 38
Figure 18 Python KNNI 39
Figure 19 Python Iterative Imputere With KNRegressor 39
Figure 20 Low Error Full Data SWP Missing Value 40
Figure 21 Low Error Full Data KNNI Missing Value 40
Figure 22 Low Error Full Data Iterative Missing Value 41
Figure 23 Low Error Full Data Proportional Errors Missing Value 42
Figure 24 Low Error Half Data SWP Missing Value 43
Figure 25 Low Error Half Data KNNI Missing Value 44
Figure 26 Low Error Half Data Iterative Missing Value 44
Figure 27 Low Error Half Data Proportional Errors Missing Value 45
Figure 28 High Error Full Data SWP Missing Value 46
Figure 29 High Error Full Data KNNI Missing Value 46
Figure 30 High Error Full Data Iterative Missing Value 47
Figure 31 High Error Full Data Proportional Errors Missing Value 48
Figure 32 High Error Half Data KNNI Missing Value 49
Figure 33 High Error Half Data SWP Missing Value 50
Figure 34 High Error Half Data Iterative Missing Value 50
Figure 35 High Error Half Data Proportional Errors Missing Value 51
Figure 36 Low Error Full Data SWP 55
Figure 37 Low Error Full Data Dirty 55
Figure 38 Low Error Full Data AR 56
Figure 39 Low Error Full Data ARX 56
Figure 40 Low Error Full Data KNN 57
Figure 41 Low Error Full Data Proportional Errors 57
Figure 42 Low Error Half Data AR Outlier 59
Figure 43 Low Error Half Data ARX Outlier 59
Figure 44 Low Error Half Data SWP Outlier 60
Figure 45 Low Error Half Data Truth Outlier 60
Figure 46 Low Error Half Data Dirty Outlier 61
Figure 47 Low Error Half Data Proportional Error Outlier 61
Figure 48 High Error Full Data AR Outlier 62

Figure 49 High Error Full Data ARX Outlier 63
Figure 50 High Error Full Data KNN Outlier 63
Figure 51 High Error Full Data SWP Outlier 64
Figure 52 High Error Full Data Dirty Outlier 64
Figure 53 High Error Full Data Proportional Errors Outlier 65
Figure 54 High Error Half Data ARX Outlier 66
Figure 55 High Error Half Data AR Outlier 66
Figure 56 High Error Half Data KNN Outlier 67
Figure 57 High Error Half Data SWP Outlier 67
Figure 58 High Error Half Data Dirty Outlier 68
Figure 59 High Error Half Data Proportional Errors Outlier 68

Acknowledgements

I would like to thank my supervisor Dr Taoxin Peng for his throughout the entire course of this project, his guidance was invaluable in helping me maintain some direction in my honours project & finish on time.

1 Introduction

Data cleaning is an important stage data pre-processing, through the use of various algorithms, it is possible to remove irrelevant & anomalous data from the dataset as well as imputing lost values. It is generally easier to work out these values with the use of additional clean data meaning that it can be difficult to accurately estimate the true values.

Efforts to ensure that data is complete, accurate & relevant involve a variety of algorithms, making use of nearest neighbour methods, regression, clustering & iterative approaches. Each of these methods are suited to different purposes, where some algorithms are unlikely to perform well on volatile data, others may perform better. Some algorithms are faster & some are slower.

In this project, both imputation & outlier correction algorithms were tested on univariate timeseries datasets… with the goal of finding a reasonable alternative to SWP on high volatility datasets such as the hourly weather from (Ranjan et al., 2019). There was also an attempt to see if it was possible to impute missing values to an extremely volatile dataset such as the California Transit hourly flow data without the use of external data.

This project outlines the features of the algorithms investigated and attempts to perform data cleaning on the various datasets. The time and accuracy are considered along with how these values change when the size of the dataset & rate of errors are changed.

For the outlier correction hourly temperature data was taken from WUnderground for Lincoln, Nebraska, USA. Additional outliers were added to this data as to gauge the effect of rate of outliers & the accuracy of the processed data. By halving this dataset, & timing each algorithm as it processed the data, it was possible to establish a trend of time in relation to file size. In the case of missing values, a similar process was undertaken, data from the California Transit Authority had 10% & 40% of datapoints removed & the accuracy questioned… in the case of this data, the increase in processing time according to rate of missing values was also considered. As with outlier repair, this dataset was also reduced in size by half to gain insight on time considerations as the data size scales upwards.

2 Literature Review

2.1 What Is data cleaning

Data cleaning… "Or data cleansing, includes operations that correct bad data, filter some incorrect data out of the data set and reduce the unnecessary detail of data."(García et al, 2014, p11). Data cleaning is a fairly broad concept, simply put, it is the process of turning dirty data into clean data. It is a very large topic with a huge amount of literature, spanning several different types of dirty data.

"The General Framework for Building Operational Data Preprocessing introduces the general data preprocessing framework for building operational data analysis. Data Cleaning Methods for Building Operational Data Analysis, Data Reduction, Data Scaling, Data Transformation, Data Partitioning describe representative techniques for typical data preprocessing tasks."(Cheng Fan et al., 2021). Data cleaning is typically viewed as the first stage of data pre-processing. It is arguably the most important stage as, without it, most datasets would be useless.

"Data quality is usually perceived as multidimensional concept that is characterized by different aspects, so called dimensions. Those dimensions can either refer to the extension of the data (i.e., data values), or to their intension (i.e., the schema)"(Hacid et al., 2018). Dirty data is one end of a spectrum of data quality. Data quality can be measured by the following dimensions: Accuracy, Completeness, Redundancy, Readability, Accessibility, Consistency, Usefulness, Trust (Batini & Scannapieco, 2016, p21-50) Some of these dimensions will be more relevant to the concept of data cleaning algorithms… for instance data with low redundancy may be more susceptible to data loss. It should also be noted that not all papers consider these exact set of dimensions, though similar themes are present, e.g. "Completeness, Accuracy, Validity, Consistency, Currency, Availability and Accessibility, Reliability and Credibility, and Usability and Interpretability"(R. Zhang et al., 2019).

Data cleaning is an essential stage of data pre-processing, without it, data analysis could become extremely untrustworthy. The quality of data can be measured with several dimensions, each representing an important aspect of the data with real-world consequences if diminished.

2.1.1 Accuracy

(Sidi et al., 2012) Collates several definitions for data accuracy:

- (Batini et al., 2009) & (Ballou & Pazer, 1985) Consider the primary factor in accuracy is comparison to real world value.
- (Wang & Strong, 1996) Describes accurate data as "correct, reliable and certified"
- (Batini et al., 2009) & (Thomas C. Redman, 1996) Define accuracy as how a datapoint is to the value regarded as true.
- (Danette McGilvray, 2008) Argues that the accuracy of data should be compared to "an authoritative source".

There are several definitions seen here, all with a similar theme… the most concise would be the second of these, originally cited from (Wang & Strong, 1996). Accuracy is a fairly simple concept to grasp, it has essentially the same meaning in and outside of data science. As it relates to the project of data cleaning; data with low accuracy needs to be cleaned. If data correction or imputation has been performed to a low standard then the resulting data can likely be considered low accuracy.

2.1.2 Completeness

As provided by (Sidi et al., 2012) the following are some helpful definitions for data completeness:

- (Batini et al., 2009) & (Wand & Wang, 1996) Considers completeness to be the extent to which the data can represent the entirety of the tangible entity it represents.
- "The extent to which data are of sufficient breadth, depth and scope for the task at hand"(Wang & Strong, 1996) Completeness requires enough "breadth, depth & scope"
- (Thomas C. Redman, 1996) & (Batini et al., 2009) Consider Completeness to be "The degree to which values are present in a data collection"
- (Batini et al., 2009) & (Jarke et al., 2002) state that completeness is amount of "real-world information" used in the data storage system.

A fairly straightforward concept, data can be considered complete if it is all included in the dataset. In cases where large proportions of data are missing, then missing value imputation may need to be employed… more on that later.

Based on the previous literature, (Cichy & Rass, 2019) defines a modern take on data quality dimensions, where the concepts of completeness & accuracy defined by (Wang & Strong, 1996) are still very much present.

2.1.3 Summary

Data quality can be measured by several different dimensions, many of which bring into question concepts like relevance and usefulness, these are not particularly important to this project. Data cleaning algorithms are much more focused on the accuracy and completeness of data, if data is inaccurate then it will need to be corrected and if data is incomplete then it will need to be imputed. The dimensions of data quality can also be used to evaluate the effectiveness of data cleaning algorithms, some examples of this may include:

- Removing anomalous data points can result in incomplete data
- An ineffective data imputation may result in inaccurate data
- Similarly, an ineffective noise outlier correcting algorithm will be result in inaccurate data

2.2 Missing values (MV)

The most straightforward method to deal with missing values is to ignore the examples that contain them. (García et al, 2014, p59) Generally, the first solution to occur in the event of missing data… just to remove the whole row in the database. This technique will not be covered in further detail due to the solution's simplicity which could be implemented with a single SQL query/excel filter/ etc. The solution is also not applicable to many datasets as there has to be a relatively small number of missing values for their absence to have a negligible effect… for this reason, we must turn to the imputation of these values.

2.2.1 Maximum Likelihood

An old concept with some of the literature dating back to the 1970s. The goal is to fit a distribution to the dataset which would help imply the correct area of the imputed data.

2.2.1.1 Expectation Maximization

Simply put, the expectation maximization algorithm follows these steps "impute, estimate and iterate until convergence"(Fazakis et al., 2020).

More specifically The goal of the EM algorithm is to find the best value of φ with an iteratively higher probability. (Dempster et al., 1977) The algorithm iteratively changes an unspecified variable to eventually find the best fit with the actual distribution.

```
1. START
2. WHILE true
3.         attempt to predict the true location of datapoint
4.         update distribution according to prediction
5.         IF distribution fits with initial THEN
6.                 BREAK
7. END
```

Figure 1 EM Pseudocode

2.2.1.2 Iterative Imputer

Similar to EM, Iterative Imputer is a modern python imputation method, that initially estimates a value according to initial estimate strategies of mean, median, etc. Then evaluates using another algorithm which can be specified in the declaration over a series of iterations, it has previously been found to return high accuracy data even in cases where 30% of the dataset has been removed(Altukhova, 2020). The tool can have several different modes applied to it & as such can be more easily tailored to an individual problem.(*Imputing Missing Values with Variants of IterativeImputer — Scikit-Learn 1.2.2 Documentation*, n.d.)

2.2.1.3 Multiple Imputation

Multiple Imputation is a method of imputing several value for each MV & selecting the best imputation(Fazakis et al., 2020). The separate datasets are imputed using various algorithms/ techniques and then recombined, paying respect to the most likely/ common values for imputed data.

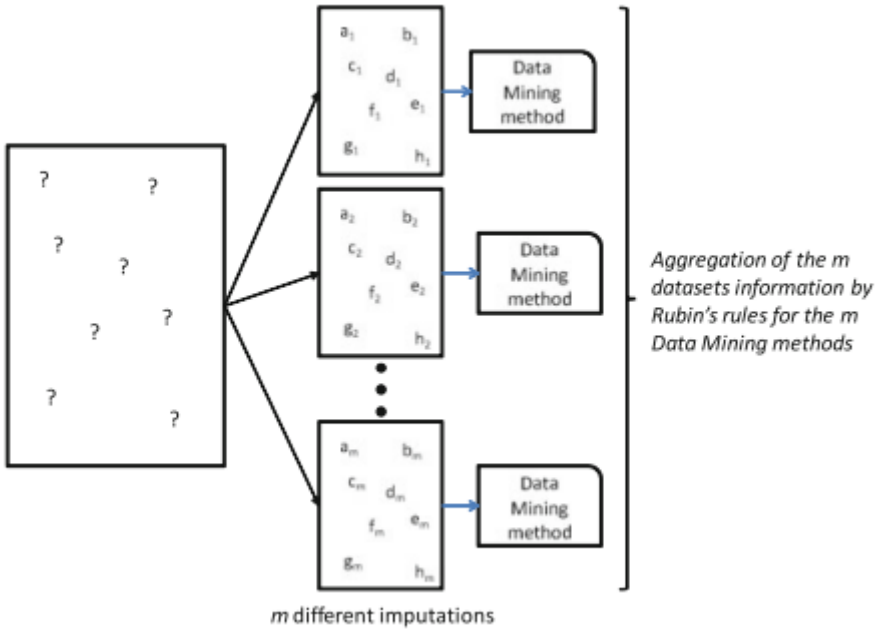


Figure 2 Multiple Imputation

This diagram from (García et al, 2014, p69) illustrates the concept of splitting the dataset into different partitions before going into their own imputation methods. It is clear that running several different imputation methods will be a more resource intensive option than just one imputation method but the result is a more accurate dataset.

2.2.2 Machine Learning based

2.2.2.1 Imputation with K-Nearest Neighbour (KNNI):

By specifying a distance from missing values, we can use Euclidean(most commonly) distance to identify nearby values to the missing datapoint, from this we can take the most common value and impute that. Values that are common but not in the majority (e.g. near values composed of 51% of one value and 49% of another) will be ignored.(Cenitta et al., 2021)

2.2.2.2 Weighted Imputation with K-Nearest Neighbour (WKNNI):

Similar to KNNI where values are taken within a specified distance but values closer to the centre are assigned a higher weight, a large proportion of the same value on the outskirts of the area may not be chosen if there is another much closer to the missing value.

2.2.2.3 K-Mean Clustering (KMI):

"The k-means algorithm is generally the most known and used clustering method."(Sinaga & Yang, 2020)

A multivariate imputation method considering the Euclidean distance between the missing value and the nearby datapoints to calculate a cluster(García et al., 2014). Below is some pseudocode outlining the procedure.

"1) Begin
    a) Randomly select k sample to be the initial medoids.
    b) Assign sample of training to the closet medoids using distance functions, most commonly used Euclidean distance.
    c) Now arbitrary choose 'O' a non-medoid object.
    d) Compute the total cost C, of swapping O_j with O random.
    e) If C < 0 the swap O_j with O random to form the new set of k-medoids.
2) Repeat until no changes."

Figure 3 KMI Pseudocode

2.2.2.4 Decision Trees

By using a training dataset, which does not contain missing values and creating a “flowchart like structure” to find correlations between datapoints and define rules which can then be used on the faulty dataset to guess and impute missing values.(Cenitta et al., 2021)

2.3 Noisy Data

"Real-world data is never perfect and often suffers from corruptions that may harm interpretations of the data, models built and decisions made."(García et al, 2014, p107) The concept of noise in data is similar to that of real-life noise. Data noise is just an extra layer of bad/corrupt/outlying data that has been mixed in with useful data with the overall effect of reducing the effectiveness of data analysis.

2.3.1 Noise filtering

2.3.1.1 Rough- KNN Noise Filter (RK- FILTER)

Another use of the K-Nearest Neighbour algorithm…

- Begin
  For j=1:s
  1. Calculate the k_2 nearest neighbors of modules $x_j \in B(X)$ from dataset $\{S_F \cup S_{NF}\} - x_j$ through KNN rule under rough set.
  2. If $x_j \in S_F$, then counts the number of Non-faulty modules among k_2 nearest neighbors, otherwisecount the number of faulty modules.
  3. If $x_j \in S_F$ and all the k_2 nearest neighbor of x_j are non-faulty modules then we can consider x_j as the noisy modules and vice versa.
- End For
- Delete all the noisy modules from S_F & S_{NF}
- Update new sets $S_F^{/}$ &$S_{NF}^{/}$

Figure 4 Rough KNN Filter Pseudocode

(Riaz et al.,2018)

2.3.1.2 Cross Validated Committees Filter (CVCF)

By Splitting a dataset into several subsets of equal size, then a decision tree algorithm (e.g. C4.5) is trained for each subset. The following decision trees are applied to each of the subsets with the exception of their respective training data. From this we can identify each mislabelled instance in the data, the decision trees then vote, where more trees showing a particular instance is incorrect improving the chances of it being filtered.(Liu et al., 2020)

2.3.1.3 Conclusion

Noise filtering is an easy solution to handle data anomalies. There are several algorithms similar to these which can be used to identify datapoints which should be removed, this can cause problems in some data, by blanket removing all incorrect data, we can find ourselves with datasets which no longer show any kind of trend and cannot be used for data mining... another possible solution would be to repair incorrect data.

2.3.2 Regression-Based Based Methods

"A straightforward idea is to directly interpret the predication values in anomaly detection, e.g., by AR or ARX, as repairs" (A. Zhang et al., 2017) Using regression allows us to fit a model while correcting instances of anomalous data. A more reasonable trend can be found, data points which are not close/ totally different from their corrected values are clearly incorrect…

2.3.2.1 Values

Symbol	Description
x	Data set
x_i	Datapoint (index i)
y	Repaired dataset(x)
$y^{(k)}$	y of iteration k
ϕ	Parameter of AR(p)/ARX(p) with order p
c	$\mu(1-\sum_{i=1}^{p}\phi_i)$
μ	*Mean of* ϵ_t
ϵ_t	*White noise (gaussian)*

Table 1 Variables in AR Equations

2.3.2.2 Autoregressive model

The autoregressive (AR) model uses previous known data to predict new values (x_t'), through doing this, anomalies can be detected in real time, making it a useful tool for monitoring software or in timeseries datasets. (A. Zhang et al., 2017)

$$x_t' = c + \sum_{i=1}^{p} \phi_i x_{t-i} + \epsilon_t$$

2.3.2.3 ARX(autoregressive with exogenous inputs) Model

A similar concept to the AR model, the difference is simply that ARX uses exogenous data (Akouemo & Povinelli, 2017) Where external data is used to supplement the calculation of the AR model.

$$y'_t = x_t + \sum_{i=1}^{p} \phi_i (y_{t-i} - x_{t-i}) + \epsilon_t$$

ARX improves on the accuracy from AR but in some cases it falls short…

2.3.2.4 Correcting data after autoregression

(A. Zhang et al., 2017) Defines two major kinds if data cleaning; smoothing based & constraint based. Smoothing-based cleaning is considered the option to "eliminate noisy data", an example of this can be seen above, looking at AR and ARX. Constraint based-cleaning is more effective at repairing the dirty data.

Smoothing has a tendency to modify an entire dataset when erroneous data is encountered while a constraint based approach would only alter the dirty data points (Shaoxu Song et al., 2015).

While it is possible to use a regression model to predict the actual value of a data point, when there is a particularly large anomaly in a data point, the regression model will only partially correct the datapoint, leaving a trend which does show that clearly there has been an anomaly but does not fully correct the trend.

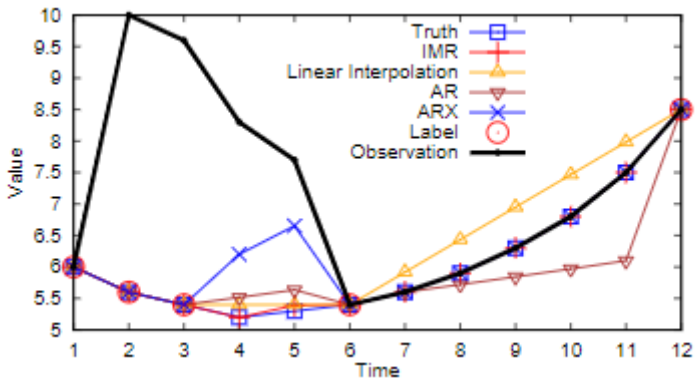


Figure 5 Comparison of IMR, AR & ARX

(A. Zhang et al., 2017)

As can be seen, when a few datapoints are incorrect, the whole trend is set off with each method, some methods like ARX under-corrected in the anomalous datapoints while other methods such as AR over-corrected on datapoints which weren't even wrong. The only method which returned the correct trend was IMR.

2.3.2.5 Iterative Minimum Repairing (IMR)

Input: time series x and partially labelled $y^{(0)}$
Output: $y^{(k)}$ with all the labelled $y_i^{(0)}$ unchanged and unlabelled $y_j^{(0)}$ repaired

1. For $k \leftarrow 0$ to *max-num-interations* do
2. $\phi^{(k)} \leftarrow$ Estimate($x, y^{(k)}$);
3. $\hat{y}^{(k)} \leftarrow$ Candidate($x, y^{(k)}, \phi^{(k)}$);
4. $y^{(k+1)} \leftarrow$ Evaluate($x, y^{(k)}, \hat{y}^{(k)}$);
5. If Converge($y^{(k)}, y^{(k+1)}$) then
6. break;
7. $k \leftarrow k + 1$
8. return $y^{(k)}$

Figure 6 IMR Pseudocode

By iteratively altering the data, it is possible to negate concerns of 'over change' that would be found by using AR & ARX(A. Zhang et al., 2017)

2.3.2.6 Summary of regression-based methods for cleaning noisy data

While simple regression models are easy to implement and generally not resource intensive and can reliably identify instances of dirty data, they are limited in their capacity for correcting this data. If data correction is the goal, then additional algorithms may be required, such as the IMR algorithm. In situations where resources such as time and computational power are limited then regression alone may be acceptable despite low accuracy results or limiting the system to noise removal rather than repairing.

2.3.3 Sliding Window Prediction

Sliding Window Prediction is an algorithm capable of estimating the value of a datapoint based on a window of neighbours weighted by the distance of the current datapoint to each neighbour. By calculating the prediction confidence interval (PCI), it can be determined if the value estimated is better than the original, if this is the case then the new value will be used, otherwise it will be discarded while the original is kept.(Yu et al., 2014)

(Ranjan et al., 2019) Compares SWP (sliding window prediction) and KNN based cleaning algorithms on several time series datasets, the results found that with an increasing volatility of data, the accuracy of the end product was diminished. This effect was much more prominent with SWP than KNN.

On a dataset with lower volatility, SWP performs well. In general, outliers are identified and are corrected to a value which is more in line with non-anomalous data.

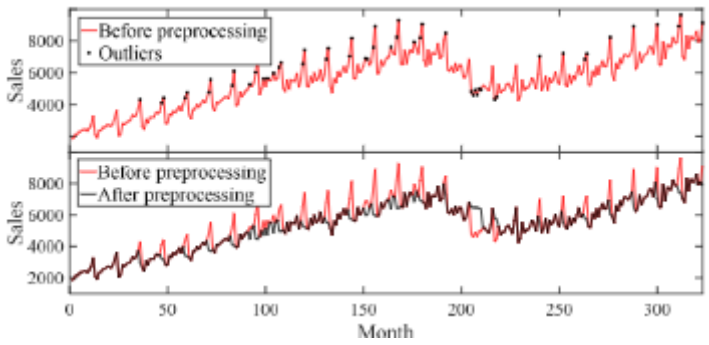


Figure 7 KNN Outlier Detection & SWP on Furniture Sales

As can be seen, with a greater degree of volatility in the dataset, the SWP algorithm does not fair well, eventually failing after several thousand datapoints. The resulting 'clean' dataset is now of a very poor standard and cannot be used.

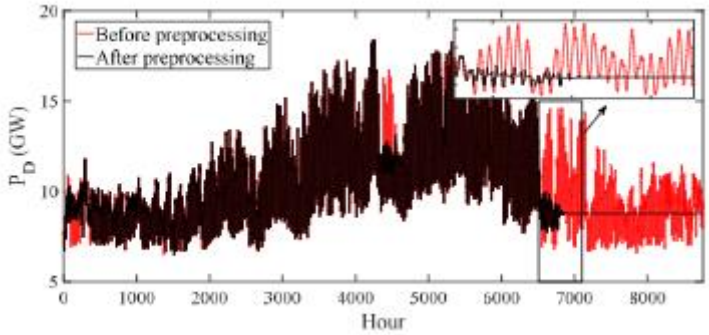


Figure 8 SWP on Weather Data

2.4 Algorithm Effectiveness Metrics

2.4.1 Similar Work

Before deciding on how to evaluate the fitness of a data cleaning algorithm, it would be wise to consider the methods used in literature which has also compared data cleaning algorithms…

(Ranjan et al., 2019) Compares the SWP and KNN methods for detecting and correcting anomalous data in several datasets. The datasets are of various sizes and levels of volatility… the results show that both methods struggle when higher levels of volatility are encountered but SWP completely failed when trying to impute correct values. The paper concludes that both methods are unsuitable for data cleaning on volatile datasets.

(Shengjie Liu et al., 2021) Discusses the practical applications of data cleaning with regard to neuroscience, it compares the accuracy of data cleaned with several automatic data cleaning methods(Laplacian, bipolar, ESR, CAR, GWR) along with the accuracy of the raw data… while this paper does not go into full detail about the data science side of this application, it still shows the importance of accuracy in data cleaning as well as providing evidence for the importance of data cleaning.

(Dasu & Loh, 2012) Discusses in detail the concept of measuring data cleaning algorithms and the potential dangers of data cleaning. By calculating a "glitch score" for each data point and finding the sum of all scores in a dataset, we can determine the accuracy dimension of a dataset both before and after cleaning. It also considers "Statistical Distortion", "If cleaning strategy C applied to data set D yields a cleaned dataset D_c ,then the statistical distortion of C on D is defined as: $S(C, D) = d(D, D_c)$ where $d(D, D_c)$ is a distance between the two empirical distributions."

Of all algorithm comparisons, the primary aspect used to determine the effectiveness seems to be accuracy. While scalability is also very important to accuracy as with larger datasets, a non-scalable algorithm may perform poorly, returning a low accuracy result. Time should also be considered a factor with consideration to scaling of computation time when applied to a larger dataset. To that end, here follows a basic outline for what the analysis of data cleaning algorithms might look like.

2.4.2 Time

It is no surprise that the question of which algorithm to use may take the time of computation into account.

(García et al, 2014, p182) Refers to Multiple imputation as a "greedy algorithm" and it is no surprise, the fact of the matter is that any algorithm will require compute time, and this is a central concept for computer science, even outside of data cleaning.

2.4.2.1 Scalability

While time itself is a fairly general concept throughout any processing task, so too is the idea that increasing the number/ size of inputs will increase compute time.

"in automated analysis of large quantities of data, the scalability of ML algorithms becomes one of the most important considerations."(Kurgan et al., 2006) and it is no surprise, one of the major reasons for cleaning data algorithmically is because the datasets are too large for a human to clean manually, so it is important that we consider how much longer it will take an algorithm to clean a dataset given its size.

2.4.3 Accuracy

2.4.3.1 RMSE

One measure of accuracy is the root mean square error (RMSE) seen in (Altukhova, 2020), a measure of the entire imputed dataset, where the average error rate can be determined, it is proportional to the magnitude of the data so in a dataset with an mean value 2000 can be expected to have a higher RMSE than one with a mean of 20.

RMSE can be used at a metric, calculated as follows:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} x_i^{observed} - x_i^{predicted}}{n}}$$

2.4.3.2 Scalability (continued)

This points to a potential metric; 'does the algorithm work on my dataset?'. It may also point back to the concept of scalability, given that this dataset was corrected up until about 6-7 thousand datapoints, we can hypothesize that the algorithm might work on a smaller dataset of similar content.

2.4.3.3 Summary

The following table outlines, how these metrics could be implemented

Accuracy	Graphical analysis, comparison of the dataset before and after cleaning (Dasu & Loh, 2012)'s Glitch index and statistical distortion should also be considered Root Mean Square Accuracy is a simple way to compare two post cleaning datasets.
Increasing size (accuracy)	With a variety of dataset sizes, it should be investigated whether an algorithm performs poorly on a large or small dataset.
Increasing errors (accuracy)	By adding anomalies to the dataset, it can should be determined whether a particular algorithm is able to return accurate data in the face of high error rates.
Time	Simply stopwatch/ timer from start of program to end, a fast algorithm will take a short amount of time
Increasing size (time)	Comparison of time for each several datasets, a highly scalable algorithm will have a low standard deviation
Increasing errors (time)	It should be determined whether an algorithm is capable of running in an acceptable timeframe

Table 2 Areas of Comparison of Data Cleaning Algorithms

2.5 Conclusion

Through this literature review, we have discussed the concept of data quality and how it relates to data cleaning, with data completeness relating to the missing value imputation and data accuracy relating to the correction of anomalous data. There are several techniques discussed for dealing with low quality data.

In the case of missing values, we have the option of finding a trend in the data which may point to the correct value with the use of maximum likelihood imputation, there is also the option of imputing the most common from neighbouring values through various forms of knni.

In terms of data correction, the methods are surprisingly similar, where dirty data can be identified through knn as well as a regression trend. Dirty data can then be corrected algorithmically (such as iterative minimum repairing), through a knn style imputation & by matching it will a regression trend.

Importantly, we must be able to compare our algorithms, simple questions such as ‘how fast does it run?’ and ‘does it produce accurate results?’ can go into much more depth when details of the dataset are changed, in an example, we saw that a larger dataset produced much less accurate results and even caused the algorithm to completely fail after about 6 thousand datapoints.

3 Experiment

As discussed in the literature review, a variety of datasets can be crucial to the comparison of data cleaning algorithms. To this end I have sourced the following:

3.1 Caltrans Dataset

This database is used to monitor traffic flow in California, USA. By counting the number of cars on each road per 5 minute intervals, trends can be formed to identify areas of both high and low traffic. The cars are counted using automated machinery, as the data gathered is for every road in the state of California, it is not that surprise that some of the instruments seem to be damaged/ not on.

The Caltrans database contains a huge number of missing values (60% total), and while the main body of the dataset imputes these values for users, they have included the raw data on the website. According to the (*PeMS User Guide*, 2020) several techniques, all of which being supervised techniques, using data from functional detectors to estimate the true values.

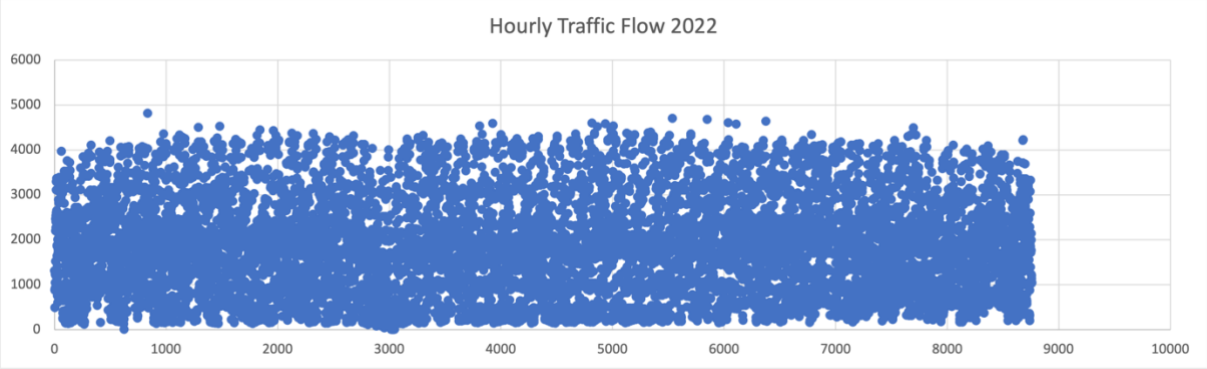


Figure 9 Full Caltrans Data

As can be seen, the Caltrans data is extremely volatile, with near 0 traffic during the night time but up to 5000 in some cases during the day.

3.2 WUnderground Dataset

The WUnderground database contains weather data for cities all over the world. Measurements are typically taken per hour. It was used in (Ranjan et al., 2019) as an example of a particularly large dataset for outlier detection. Though KNN and SWP are known to struggle with this particular dataset, it would be worth while to experiment with subsets as well as investigating the other algorithms on it.

The database takes data from registered users with their own measurement equipment but in some cases, this data appears to be of a low level of accuracy. There is also the option of manually altering data.

The database contains data on various aspects of the weather, such as temperature, humidity, wind, etc.

When algorithms required an exogenous variable, the humidity has been used as this is related data & provides a similar trend to that of the temperature data.

As per the literature review, there seems to be 2 major kinds of data cleaning algorithm, those based on the nearest neighbours for each datapoint and those based on a regression model to predict the location/ true location of a datapoint.

3.3 With missing values

3.3.1.1 KNNI

A fairly simple algorithm where data is simply taken from the closest data point, it would not be a surprise if this algorithm performs poorly on volatile data as well as data with a high level of missing values.

3.3.1.2 Iterative Imputer

The iterative imputer algorithm is provided by the sklearn python package, it works by initially imputing the missing values in a dataset with a simple average (generally mean but can be median & mode) & iteratively improving the estimate by applying some kind of estimator (in this case k-neighbours regressor).

Often compared to MICE, an R based data cleaning algorithm, iterative imputer was found to outperform other methods including MICE & KNN.(Altukhova, 2020)

3.3.1.3 SWP based Imputation

By altering the SWP algorithm outlined in the following section, it was possible to estimate the values of missing datapoints.

This algorithm was selected based on the observation of corrected values eventually flatlining on high error rate datasets, the hope was that by only applying the algorithm to a restricted few datapoints the values would be able to impute them without failing.

3.4 Without missing values

3.4.1.1 SWP

As seen in (Ranjan et al., 2019), the sliding window projection algorithm, can simply fail with complex data, it would be worth investigating this relationship… will it perform well on a small dataset with a large number of outliers? Will it perform well on a large dataset with very low levels of variance.

3.4.1.2 KNN

KNN is functionally similar to SWP, in that windows in SWP are each their own KNN instance, it would be worthwhile to consider the time difference of these two algorithms.

KNN is often used only for detection of erroneous data, as such this method alone would be significantly less desirable due to lack of correction. One avenue of research would be to apply KNNI to a dataset from which data has been removed according to KNN detection.

3.4.1.3 Autoregression

It does not seem that autoregression can fail in the same sense as SWP, while it can be thrown off & be forced to compensate, it would be worth investigating the levels to which is over/under compensates on different kinds of data as well as time comparisons with the other algorithms as this one is not especially similar to the others.

Another possible avenue of research would be the effect of introducing exogenous inputs to the algorithm.

AR is known to struggle with accuracy on highly erroneous dataset & while ARX is better, there is still room to improve(A. Zhang et al., 2017).

3.5 Experiment Plan

3.5.1 Metrics

As was discussed in the literature review, data cleaning algorithms can be evaluated on various metrics including time & accuracy along with how those metrics scale when the size of the dataset & rate of errors is increased.

RMSE will be the primary measure of accuracy, it is simple to see if one cleaning attempt is more accurate than another, it was also be pertinent to investigate proportional errors for each imputation, collecting the mean, median, maximum & standard deviation.

These values were calculated in an excel spreadsheet as follows…

RMSE:

‘=SQRT(SUMSQ(IMPUTED-ACTUAL)/COUNTA(ACTUAL))’

Where IMPUTED & ACTUAL are replaced with the column containing their respective data.

Proportional Error (individually):

=ABS(A1-B1)/B1

In the case of temperature data, this will require conversion to kelvin after cleaning so that a proportion can be found as it is impossible to divide by 0.

3.5.1.1 Imputation Algorithms

By taking a dataset with 100% known values removing some (the number removed will depend on the case).

The algorithms then attempted to impute the missing values back into the dataset. The time was recorded & accuracy was measured by the closeness of imputed datapoints to the original (proportional to the original value)..

3.5.1.2 Detection/Repair Algorithms

A timeseries dataset was used, this will have erroneous data introduced via a python script.

Each algorithm will attempt to clean datasets of various sizes, & rates of errors of errors.

3.5.2 Full Dataset Low Error Rate

A small number of errors were introduced to a dataset (10%), and each algorithm ran over the data once.

The time taken for each algorithm was recorded along with the output data.

This experiment considered the differences between algorithms on one dataset and as such empirical comparisons of 'how fast did it run' & and 'how well did it clean the dataset' could be gained.

3.5.3 Half Dataset Low Error Rate

A subset of the data was taken, and each algorithm ran over the data. This process was repeated 3 times while noting the time taken such that a mean time could be calculated.

This allows algorithms to be graded on their ability to scale with larger datasets, if one particular algorithm took 10× times longer to run on a large dataset then it might be concluded that the algorithm cannot scale effectively.

The ratio between file size & time should be 1:1 or better.

3.5.4 Full Dataset High Error Rate

A higher number of errors were introduced to this dataset (40%) and each algorithm cleaned the data. As was the case before, this was repeated 3 times to gain the average time.

From the observations gathered in this case, it can be ascertained whether the algorithms take longer to clean data of a higher level of dirtiness & how accurately they replace the incorrect values.

3.5.5 Half Dataset High Error Rate

The high error rate dataset was split in half, the algorithms were used to clean it & the average of 3 time values was gathered.

With these observations it is possible to deduce the combined effect of size & error rate on the time taken to process a dataset & the level of accuracy as well as finding how well highly erroneous data scales with size.

3.5.6 Description of Datasets

The new value used for detection/repair algorithms was calculated by adding a random percentage in the range -50% to +50% of the original value.

Detection/Repair	
Length	Number of injected errors (%)
8,680	10
8,680	40
4,340	10
4,340	40
Imputation	
Length	Number of MVs (%)
8,760	10
8,760	40
4380	10
4380	40

Table 3 Datasets in The Experiment

3.5.7 Dirty Data Injection

Due to the size of the datasets & the number of erroneous datapoints required for some experiment cases, it would not be humanly possible to alter enough datapoints without wasting unnecessary amounts of time or introducing bias. A simple python script will be required to both enter outlying data for detection/repair as well as to remove datapoints for imputation. The values will be removed/altered at random with a rate of 10% chance for the low error rate data & a chance of 40% for the high error rate data. The code is as follows:

```
import random
import numpy as np
import pandas as pd

filepath=input("Filepath:")
probability=np.double(input("Probability of injection:"))
raw_data = pd.read_csv(filepath)

mode=int(input("MV(1) or Outlier(2)"))
if(mode==1):
    data = raw_data['Column 10']
elif(mode==2):
    data = raw data['Column 3']


dirty_data=[]
for i in range(0,len(raw_data)):
    if(random.uniform(0,1)>probability):
        print(i,data[i])
        dirty_data.append(data[i])
        #break
    elif(mode==1):
        dirty_data.append(np.NaN)
        print("Removal", i)
    elif(mode==2):
        offset=random.uniform(-0.5,0.5)*data[i]
        new_data=data[i]+offset
        dirty_data.append(new_data)
        print("injection", i)

if(mode==1):
    raw_data['Column 10'] = dirty_data
elif(mode==2):
    raw_data['Column 3']=dirty_data


#file=open(input("Output:"))
raw_data.to_csv(input("Output file:"))
```

Figure 10 Dirty Data Injection Tool

3.6 Implementation

The experiment will require a simple python application to collect test data for the algorithms. The application will make use of a graphic user interface & will allow the user to load a dataset from the file system, process it & save the resulting data.

The application will also be required to record metrics at runtime, such as the time taken to complete an algorithm.

The experiments defined previously will not be hard coded into the application, they will be performed manually through the user interface.

The following python libraries can be utilized to implement the algorithms:

- pandas
- numpy
- statsmodels(Autoreg)
- sklearn(KNNImputer, KNeighbors,Regressor, TimeSeriesSplit)

3.6.1 GUI

Using tkinter for python the following GUI was implemented.

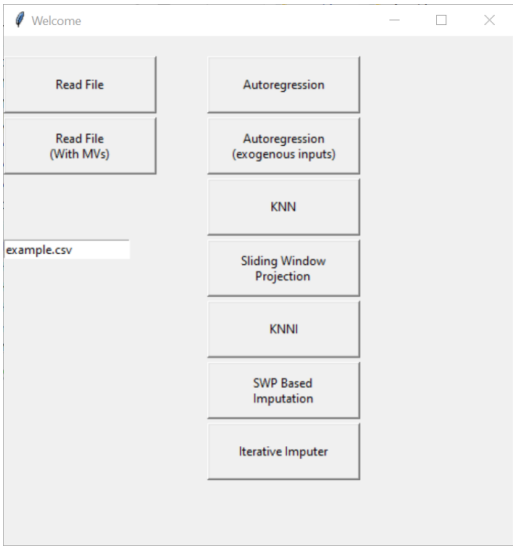


Figure 11 GUI of The Tool

The buttons on the right both read a file path specified by the input box bellow. The buttons on the right each clean & save the output data to a file. Where the lower button is capable of handling data with missing values and saving them as NaN.

```
def start_gui():
    win = Tk()  # creating the main window and storing the window object in 'win'
    win.title('Welcome')  # setting title of the window
    win.geometry('500x500')  # setting the size of the window

    filepath = Entry()
    filepath.pack
    filepath.focus_set()
    filepath.place(x=0, y=200)

#ALGORITHMS & FILE READING GO HERE

    def Example_Algo():
        raw_data=pd.read_csv(filepath.get())
        file=open('example_output.csv')
        #clean data
        file.write(raw_data)
        file.close()


    btn = Button(win, text="Example", width=20, height=3, command=Example_Algo)
    btn.place(x=0, y=20)

    win.mainloop()  # running the loop that works as a trigger

if __name__ == '__main__':
    start_gui()
```

Figure 12 Python GUI

A simplified version of the GUI capable of reading from a file and writing directly back to 'example_output.csv'.

3.6.2 Time Recording

Using the 'time' package for python a simple time recording could be made. This recording did not include reading from the file or writing the output to a file. Once the algorithm has finished cleaning the data, a message box with the value is displayed.

```
def Example_Algo():
    start=default_timer()
    #Algorithm Goes Here
    tkinter.messagebox.showinfo("Time",start-default_timer())
    #Write To File
```

Figure 13 Python Timer Recording

The value returned by this is partially dependant on uncontrollable factors on the device the tool is being run on, for this reason the same device was used each time and tests performed in one session without starting or stopping any new applications. Averages were taken.

3.6.3 SWP Implementation

Unfortunately, it does not seem that any existing python package can provide the entirety of the SWP algorithm outlined in (Ranjan et al., 2019). However, the paper does outline the process and as such, a python implementation has been developed.

The following equation defines the weight for a pair of datapoints where x_i is the value being analysed and x_{i-j} is a value within the window.

$$w_{i-j} = \frac{1}{|x_i - x_{i-j}|}$$

This can be quite easily converted to python

```
w=1/distance.euclidean((i,arr[i]),(i-j,arr[i-j]))
```

Where arr is an array of values contained in the dataset (x)

Building on this equation, the sum of all weights within a window are taken,

$$\sum_{j=1}^{2k'} w_{i-j}$$

This can be defined by a loop representing the window, initialized at the earliest datapoint in the window $(i - 2 * k)$ and terminating at i.

$$x'_i = \frac{\sum_{j=1}^{2k'} w_{i-j}\, x_{i-j}}{\sum_{j=1}^{2k'} w_{i-j}}$$

This for this, equation the values closest to the datapoint are given a higher weight

```
for j in range(i-(2*k),i):
    w=1/distance.euclidean((i,arr[i]),(i-j,arr[i-j]))
    sig_w=sig_w+w
    sig_w_x=sig_w_x+(w*arr[i-j])
    window.append(arr[i-j])
x=sig_w_x/sig_w
```

Figure 14 SWP new Value Pseudocode

sig_w	$\sum_{j=1}^{2k'} w_{i-j}$
sig_w_x	$\sum_{j=1}^{2k'} w_{i=j} x_{i-j}$

Table 4 SWP Variables

$$PCI = x_i' \pm \left(\frac{t_a}{2} 2k' - 1 \times s \sqrt{1 - \frac{1}{2k'}} \right)$$

Where t_a is Student's t distribution and s is the standard deviation of the window. PCI(predicted confidence interval) is used to ascertain the likelihood of data correctness. If PCI is larger than the old value then it will be ignored, otherwise… the new value is considered higher quality and so the calculated vale will be used.

```
for i in range(2*k+1, len(arr)):

    sig_w=0.0
    sig_w_x=0.0
    window=[]
    for j in range(i-(2*k),i):
        w=1/distance.euclidean((i,arr[i]),(i-j,arr[i-j]))
        sig_w=sig_w+w
        sig_w_x=sig_w_x+(w*arr[i-j])
        window.append(arr[i-j])
    x=sig_w_x/sig_w

    st_dev=statistics.stdev(window)

    PCI=x+(t.ppf(q=0.01,df=2*k-1)*st_dev*math.sqrt(abs(1-(1/2*k))))
    if(abs(arr[i])<abs(PCI)):
        arr[i]=x
```

Figure 15 Python SWP

3.6.4 SWP for Missing Value Imputation

This implementation is essentially the same as the original version however there is no longer any reason to calculate PCI as any value calculated is preferable to a missing value. For the sake of comparison, this version will also only target missing values rather than the entire array.

```
for i in range(2*k+1, len(arr)):
    try:
        temp=int(arr[i])+1
    except:
        sig_w=0.0
        sig_w_x=0.0
        window=[]
        for j in range(i-(2*k),i):
            #w=1/distance.euclidean((i,arr[i]),(i-j,arr[i-j]))
            #w=1/((i-arr[i])^2+(i-j-arr[i-j]))^2
            w=1/i-j
            sig_w=sig_w+w
            sig_w_x=sig_w_x+(w*arr[i-j])
            window.append(arr[i-j])
        x=sig_w_x/sig_w
        #st_dev=statistics.stdev(window)
        #PCI=x+(t.ppf(q=0.01,df=2*k-1)*st_dev*math.sqrt(abs(1-(1/2*k))))
        arr[i]=x
```

Figure 16 Python SWP Based Imputation

Due to the lack of y coordinate, Euclidean distance is no longer viable, so a simple subtraction of the x coordinate has been used.

Missing values are identified in this case using a try statement after difficulty identifying NaN values with a simple if statement.

3.6.5 AR & ARX Implementation

The autoregressive model used was found in the statsmodels package & the implementation was very similar.

```
ar = AutoReg(endog_var, 10, missing='drop').fit()
output=ar.predict(0,len(ar.data.orig_endog))
```

Figure 17 Python Autoregression

```
arx = AutoReg(endog_var, 10, exog=exog_var, missing='drop').fit()
output = arx.predict(0, len(arx.data.orig_endog))
```

Figure 18 Python Autoregression With Exogenous Input

The only difference between AR & ARX is the addition of another (exogenous) variable. In cases of missing value imputation, the MV is dropped, and the model generated will be generated to the original length.

3.6.6 KNN & KNNI Implementation

The timeseries is formatted into a state which can be accepted by the algorithm (this will be timed separately during the experiment as formatting is an important part of data pre-processing but not entirely relevant to algorithm choice)

```
knni = KNNImputer(weights='uniform')

indexes = []
knn_var = []
for i in range(0, len(endog_var)):
    indexes.append([i])
    knn_var.append([endog_var[i]])

output = knni.fit_transform(knn_var)
```

Figure 19 Python KNNI

3.6.7 Iterative Imputer Implementation

There exists a premade package for the iterative imputer provided by sklearn

```
imputer=IterativeImputer(initial_strategy="mean", max_iter=10, estimator=KNeighborsRegressor(
imputer.fit(arr)
output=imputer.transform(arr)
```

Figure 20 Python Iterative Imputere With KNRegressor

The default estimator is Bayesian ridge but a k-kneighbors regressor was used for this experiment due to similarity with other algorithms, the values of 10 iterations & mean initial strategy are default.

4 Analysis & Evaluation

4.1 Missing Value Imputation

4.1.1 Full Dataset Low Error Rate

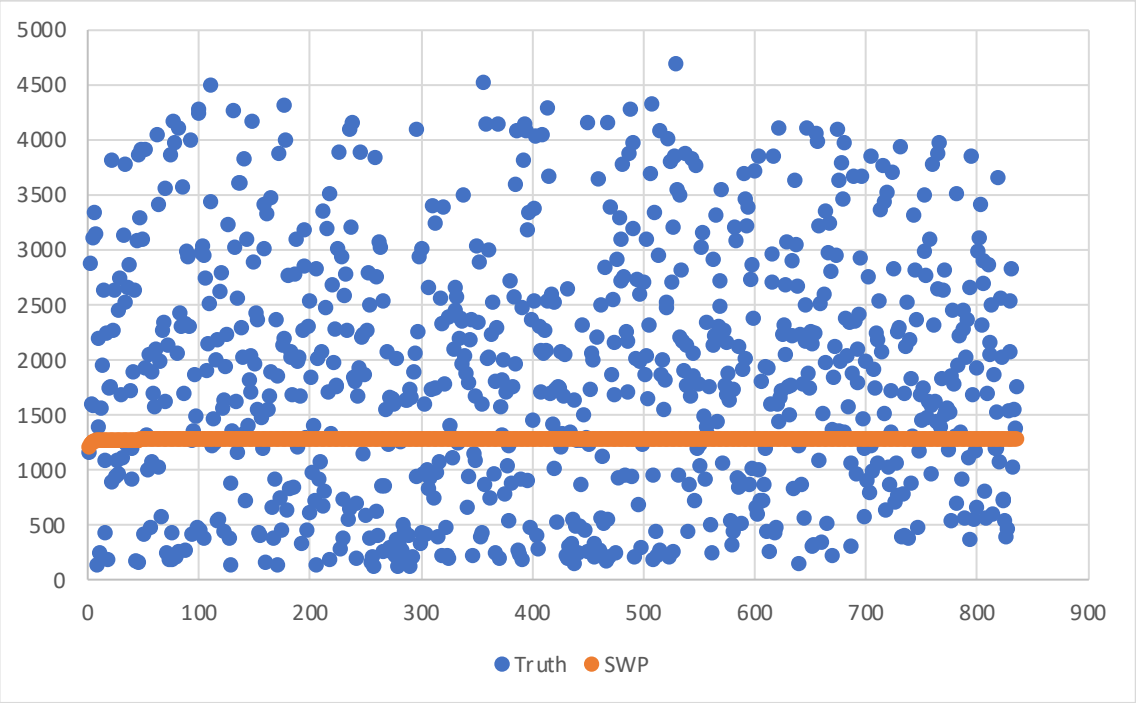


Figure 21 Low Error Full Data SWP Missing Value

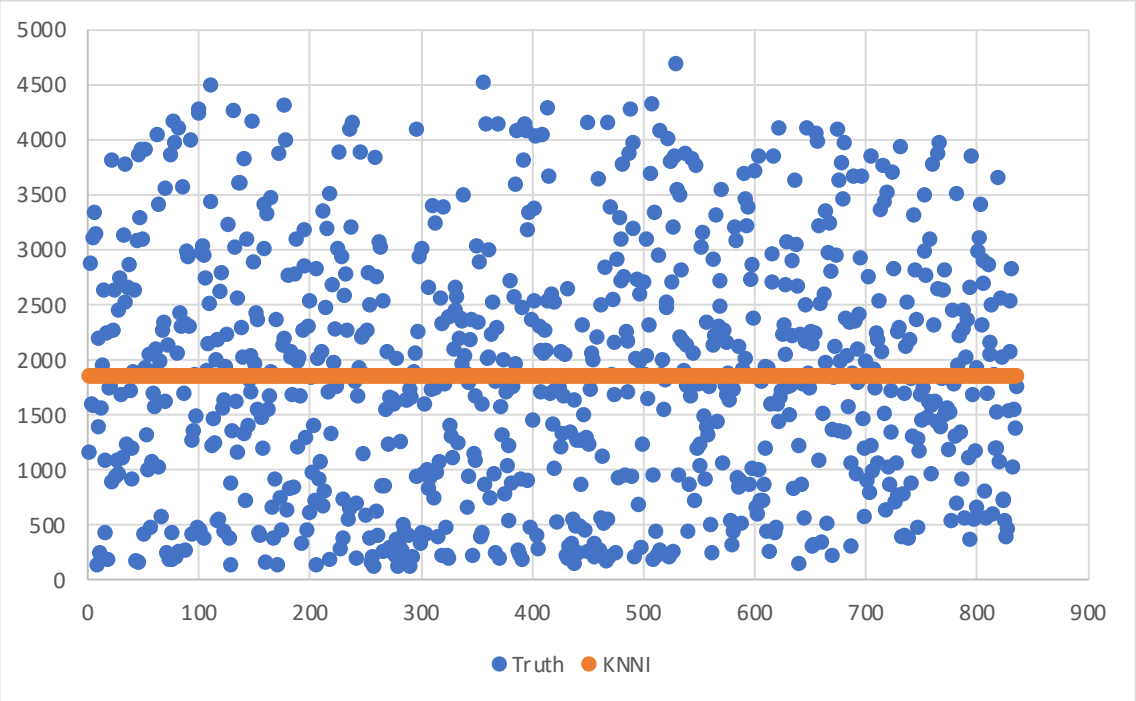


Figure 22 Low Error Full Data KNNI Missing Value

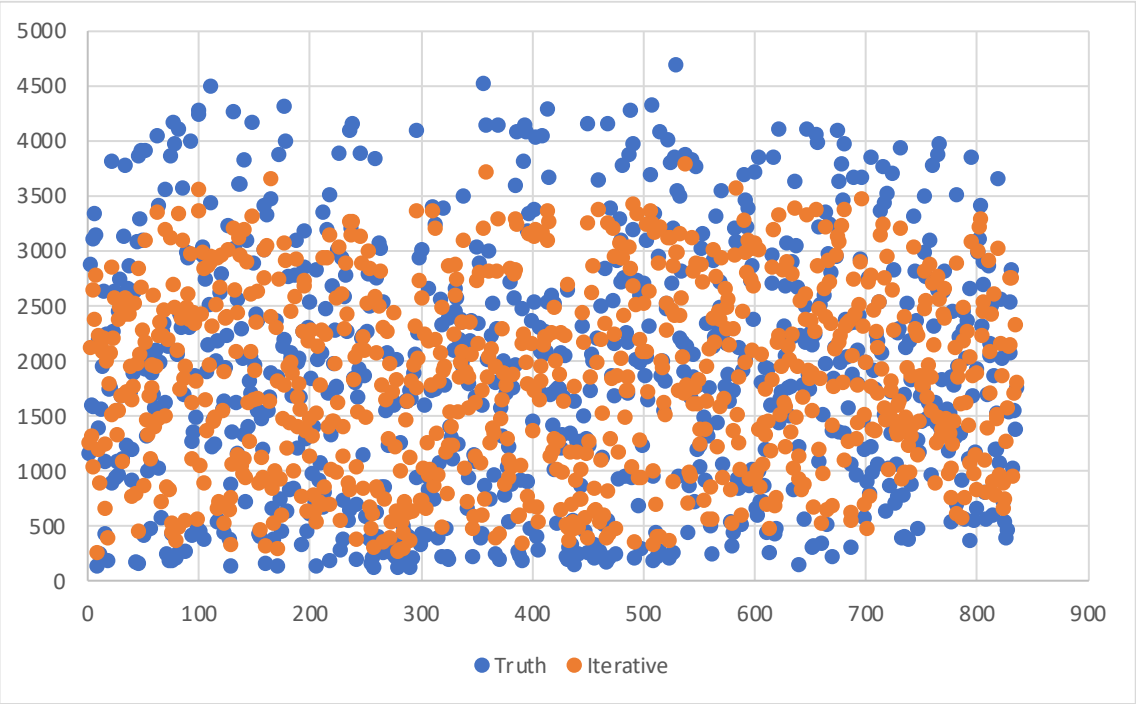


Figure 23 Low Error Full Data Iterative Missing Value

In practice, the results of SWP & KNNI are very similar, they both seem to have imputed the same value for each MV, while there is a very small variance in the case of SWP, it is clear such a volatile dataset was too great a challenge for this algorithm to impute even 10% of the values. The iterative imputer algorithm seems to have been much more successful with a variety of different imputations of different values. The iterative imputer does seem to have failed to achieve imputations on datapoints on the outer edges of the dataset.

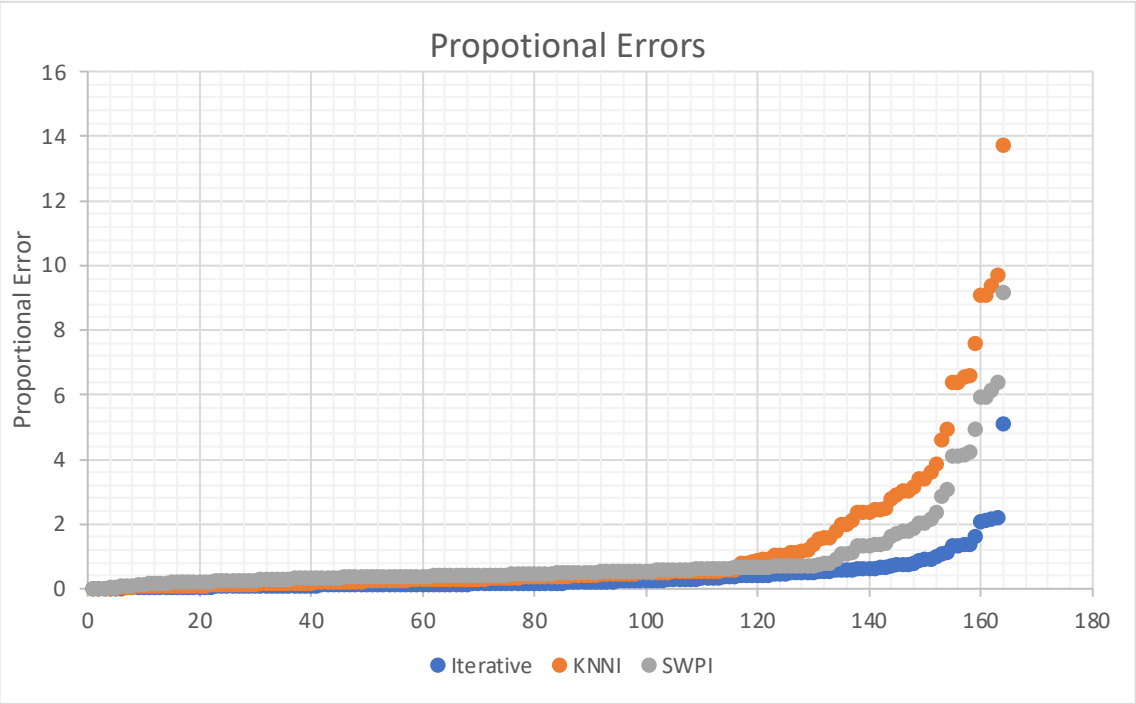


Figure 24 Low Error Full Data Proportional Errors Missing Value

	Iterative	KNNI	SWP
Median	0.188559	0.401931	0.493138
Highest Inaccuracy	5.080198	13.71914	9.146935
Mean	0.370839	1.260181	0.932888
Standard Deviation	0.555076	2.251246	1.405239
RMSE (Whole Dataset)	163.2530307	347.9121459	395.0611065
RMSE (Imputed values)	536.2076073	1153.002997	1294.525776
Time	0.030289933	0.304810967	0.140756167

Table 5 Low Error Full Data Outputs Missing Value

Unsurprisingly, iterative imputer greatly outperformed the others on RMSE where the mean value of the clean dataset is 1857.407 (this being the reason for such a large minimum value of RMSE). It can be determined that the average uncertainty of an imputation on this dataset is 1857.407 ± 28.9% (iterative imputer), 62.1% (KNNI) & 69.7% (SWP).

On this low error rate dataset, the iterative imputation algorithm performed best, generally keeping a low level of inaccuracy. There was one particularly high inaccuracy imputation 5.1, for which the other algorithms performed worse, with an inaccuracy of 8.2 & 5.3 for KNNI & SWP respectively so this is likely just a difficult imputation.

The worst-case imputations for KNNI & SWP were moth on the same datapoint where iterative imputation achieved an inaccuracy of only 1.1. It should be noted however that this particular imputation seems to be an outlier for both algorithms.

Iterative imputation was found to perform significantly better than the other algorithms in measures of time & accuracy.

4.1.2 Half Dataset with low error rate

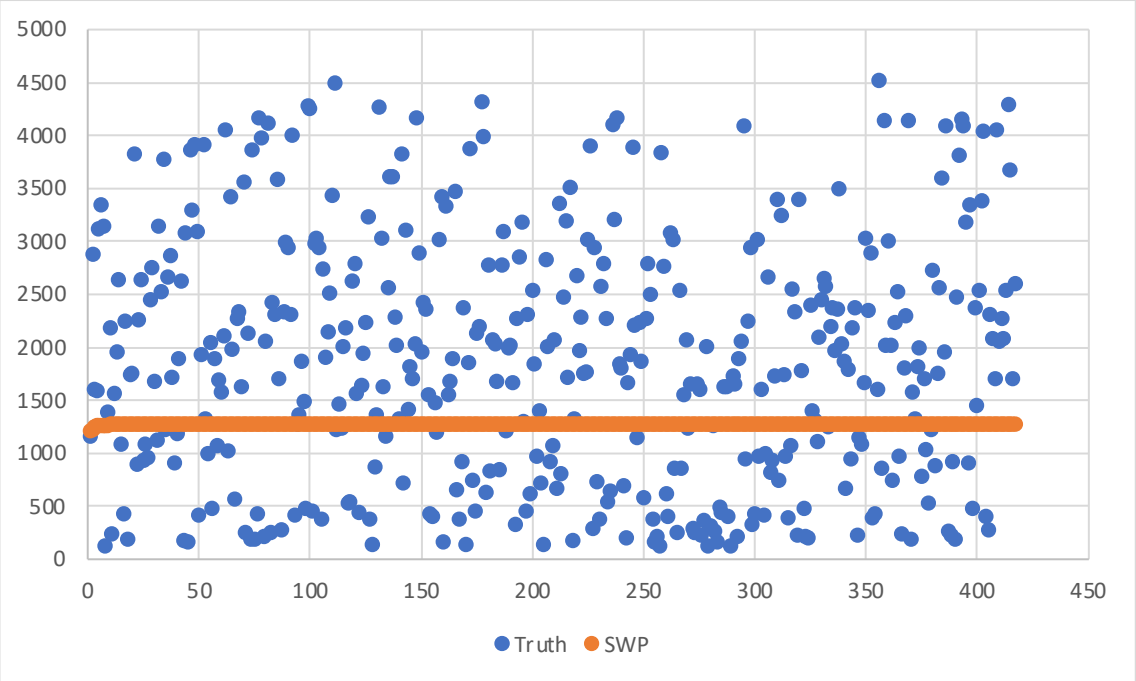


Figure 25 Low Error Half Data SWP Missing Value

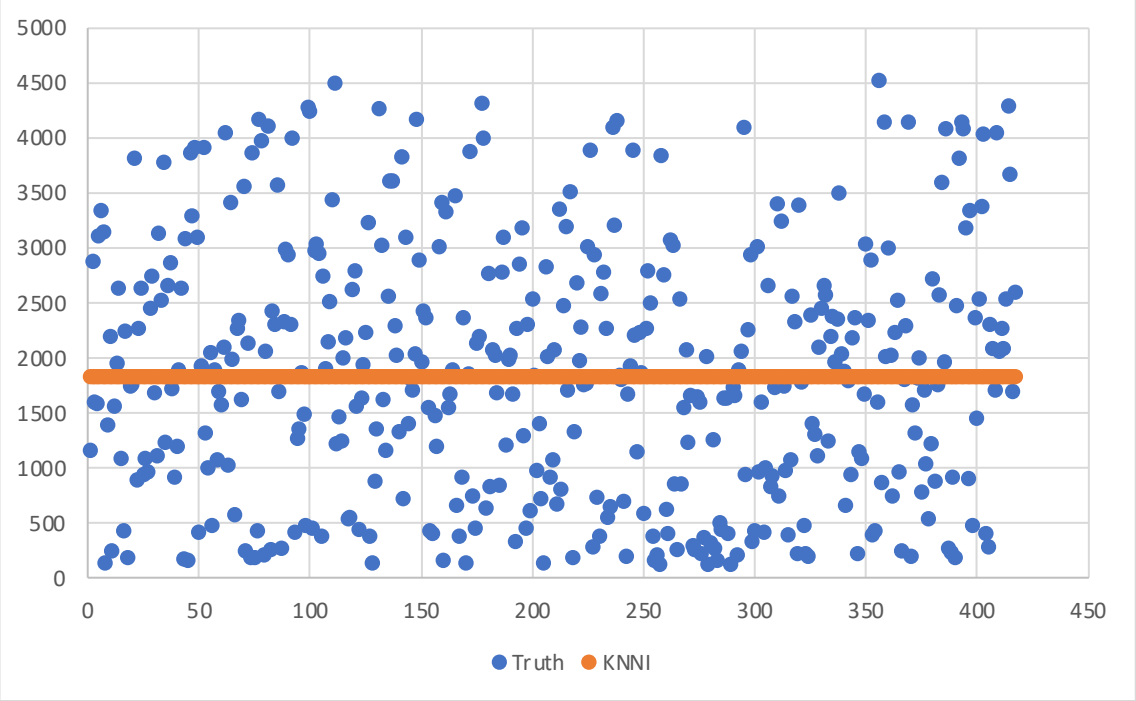


Figure 26 Low Error Half Data KNNI Missing Value

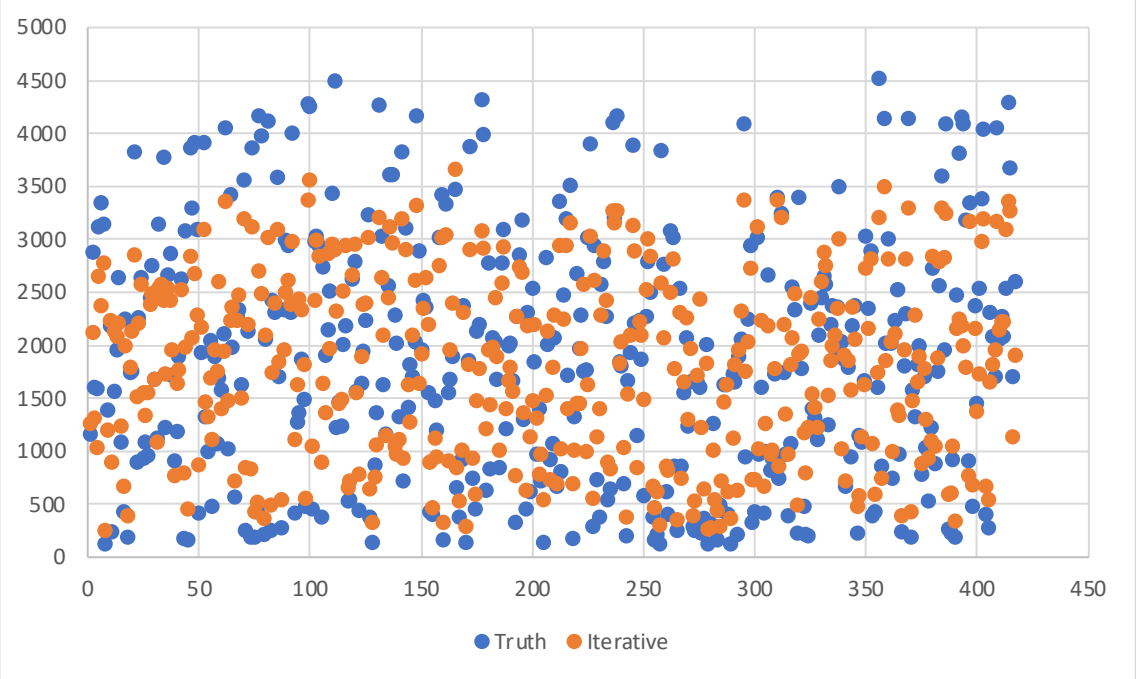


Figure 27 Low Error Half Data Iterative Missing Value

Unfortunately, it does not seem that a reduction in dataset size has allowed SWP & KNNI were able to correctly impute values on a smaller dataset.

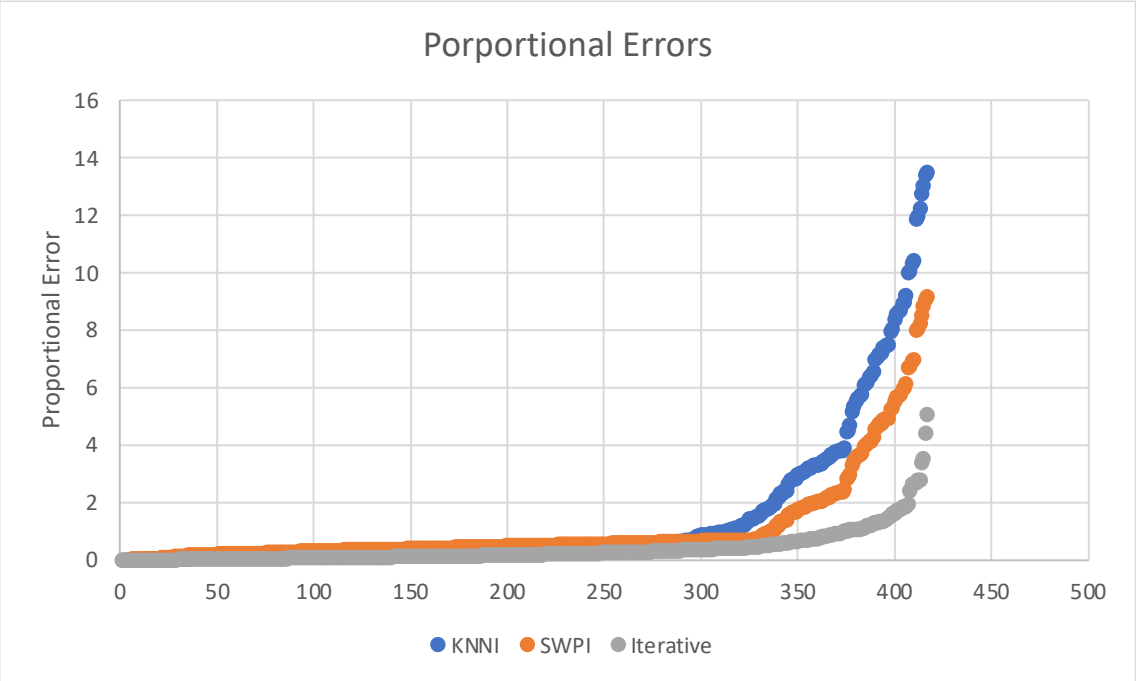


Figure 28 Low Error Half Data Proportional Errors Missing Value

	Iterative	KNNI	SWP
Median	0.19629331	0.41839596	0.50227844
Mean	0.39839243	1.44595621	1.07121083
Highest Inaccuracy	5.08019802	13.4939419	9.14693484
Standard Deviation	0.5993212	2.55400558	1.63706865
RMSE(Whole Dataset)	166.2168052	359.1921854	400.8757279
RMSE(Imputed Values)	538.6347591	1153.629341	1294.525776
Time	0.018973133	0.0859706	0.079884033

Table 6 Low Error Half Data Outputs Missing Value

The RMSE has not increased by a meaningful amount for any algorithm suggesting that none of these algorithms' accuracy can be affected by altering the size of the dataset.

Imputations using both KNNI & SWP seem to have remained fairly consistent with the previous dataset, this makes sense as both algorithms only use data from nearby datapoints… Iterative imputation seems to have continued outperforming the other algorithms.

4.1.3 Full Dataset High Error Rate

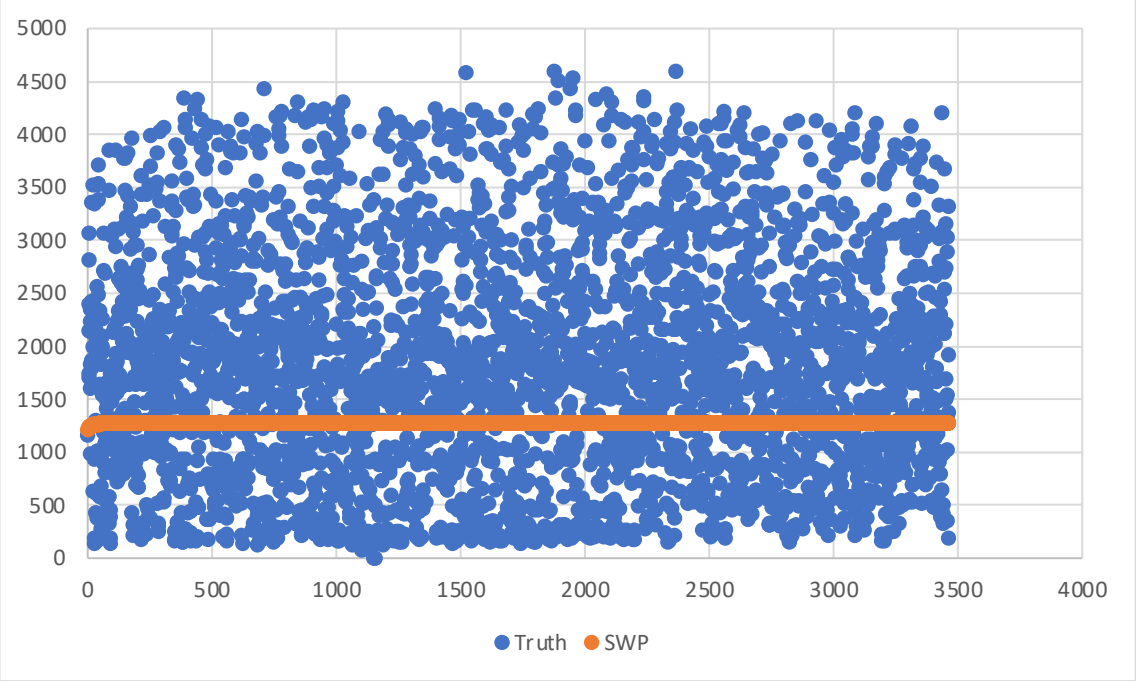


Figure 29 High Error Full Data SWP Missing Value

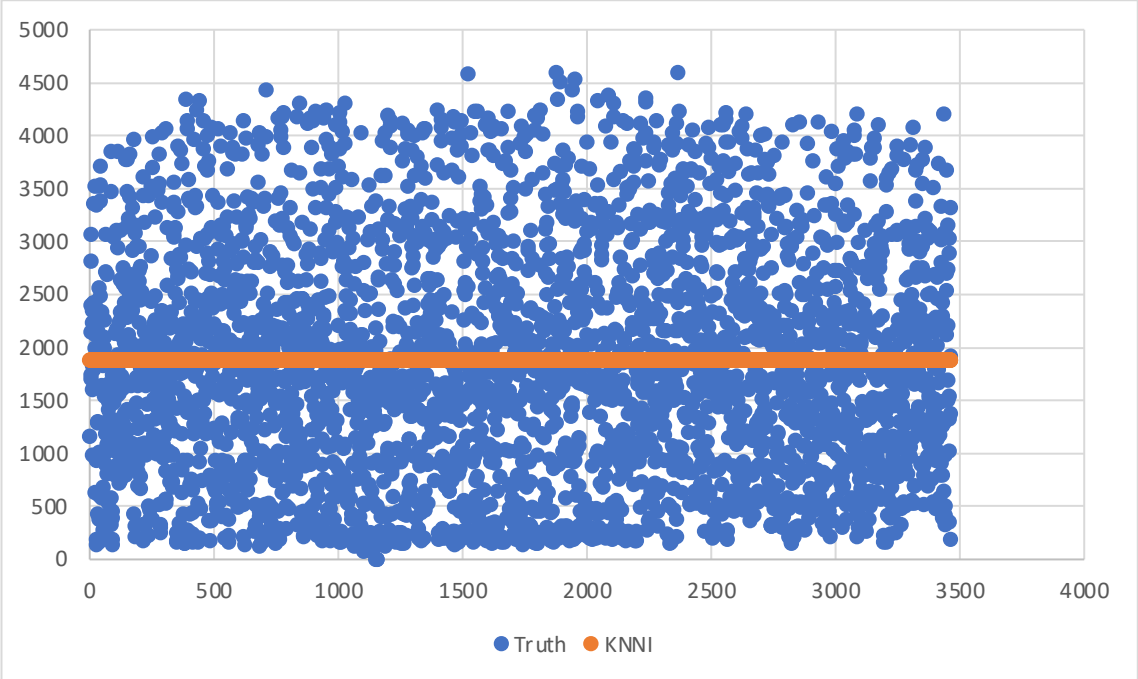


Figure 30 High Error Full Data KNNI Missing Value

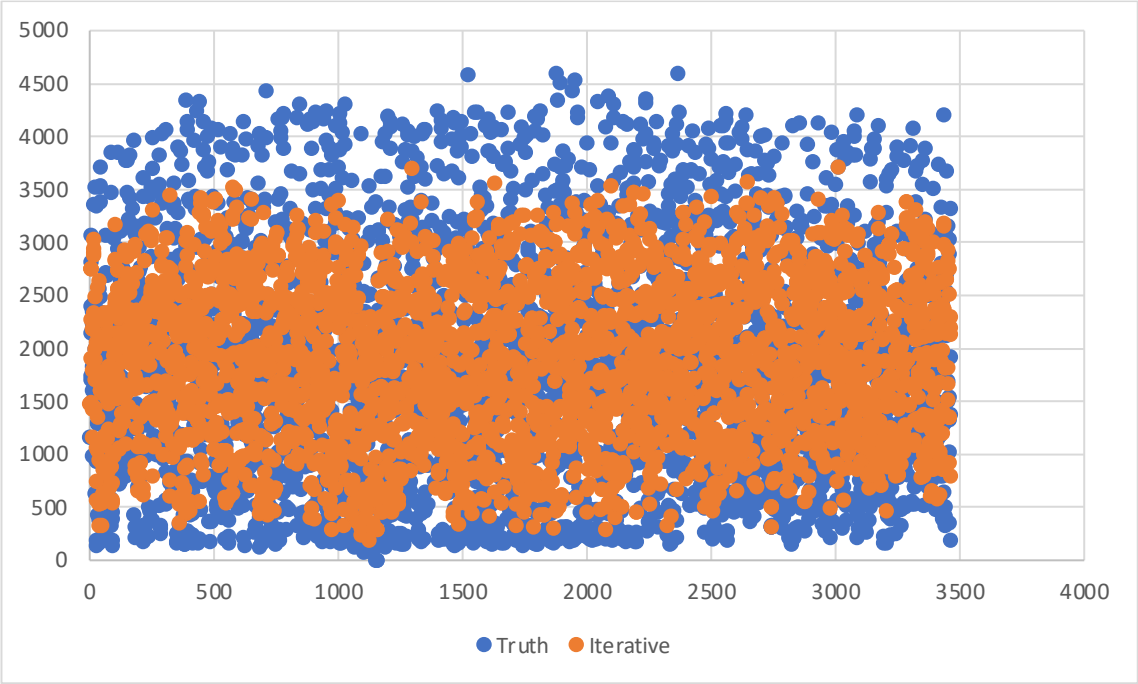


Figure 31 High Error Full Data Iterative Missing Value

In keeping with the previous dataset, SWP & KNNI seem to have imputed some fairly undeveloped values, while iterative imputer has benefitted from its ability to close in on a value over several iterations rather than simply calculating a value and forcing it into the dataset.

It should be noted that while SWP seems to have imputed a very low range of values, it seems to have chosen a more applicable value than KNNI. It seems the values of SWP lose variety as the data progresses, likely being influenced by its own imputations & ‘failing’.

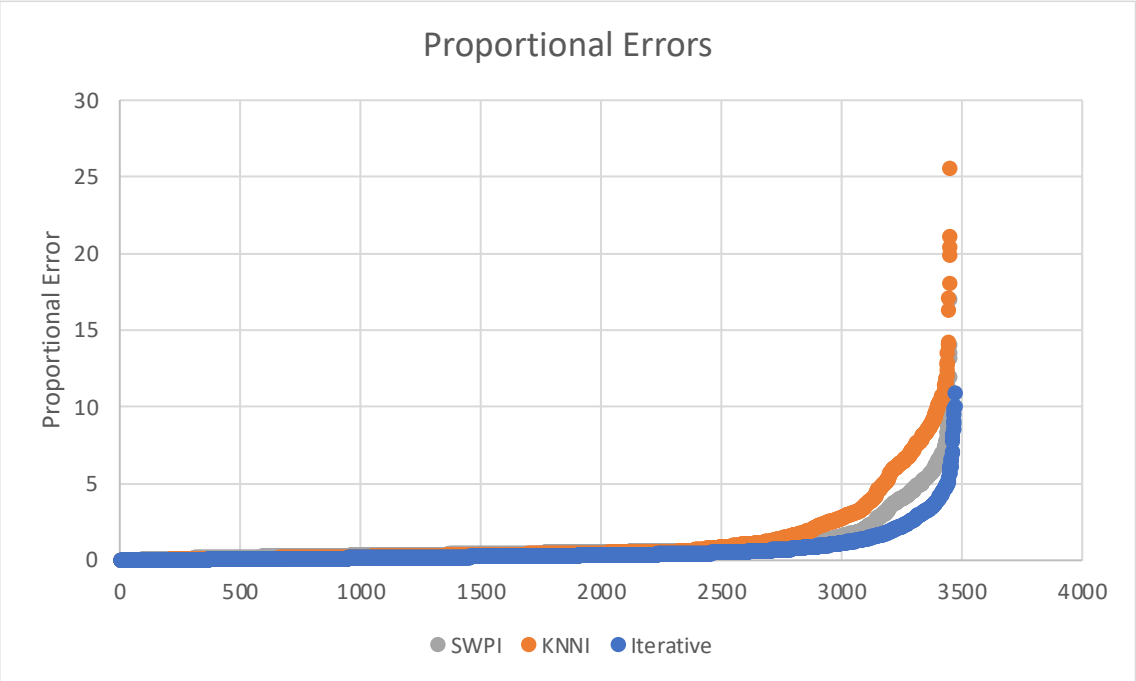


Figure 32 High Error Full Data Proportional Errors Missing Value

	Iterative	KNNI	SWP
Median	0.276411	0.401091	0.47299
Highest Inaccuracy	10.86875	25.5249	17.00702
Mean	0.607969	1.272054	0.911225
Standard Deviation	1.011215	2.336574	1.44611
RMSE(Whole Dataset)	446.5713191	684.0105731	762.9662638
RMSE(Imputed Values)	719.9413811	1096.373768	1212.165511
Time	0.039089267	1.2546521	1.305450767

Table 7 High Error Full Data Outputs Missing Value

The RMSE of imputed values of KNNI & SWP was found here to be lower than the low error rate dataset though not the RMSE of the whole dataset. This is likely down to the particular values removed rather than a trend of better imputations on high error rate data, this is supported by the relatively small decrease in RMSE. Iterative imputers RMSE was found to have increased along with the error rate with an increase of about 270% for the whole dataset & 130% on just the imputed data, this is below the ratio of 1:1 with error rate having increased by 400%.

The median inaccuracy of KNNI & SWP is comparable to that of the low error rate dataset (lower inaccuracy in fact). The median inaccuracy of the iterative imputer was found to have increased by an additional 47%.

The highest inaccuracy measure was found to have increased significantly for all algorithms, with KNNI & SWP found to be 186% of the low error data. The highest inaccuracy of iterative imputer has increased by over double the original data.

All algorithms increased in processing time from low error rate to high, in the case of iterative imputer, the increase was only 1.3× while the other two increased at a much higher rate, with an increase of 4.1× on KNNI (roughly inline with the increase in errors) and an increase 9.3× on SWP, far beyond the direct relationship that could be considered the bare minimum before calling an algorithm scalable with different levels of errors.

4.1.4 Half Dataset with high error rate

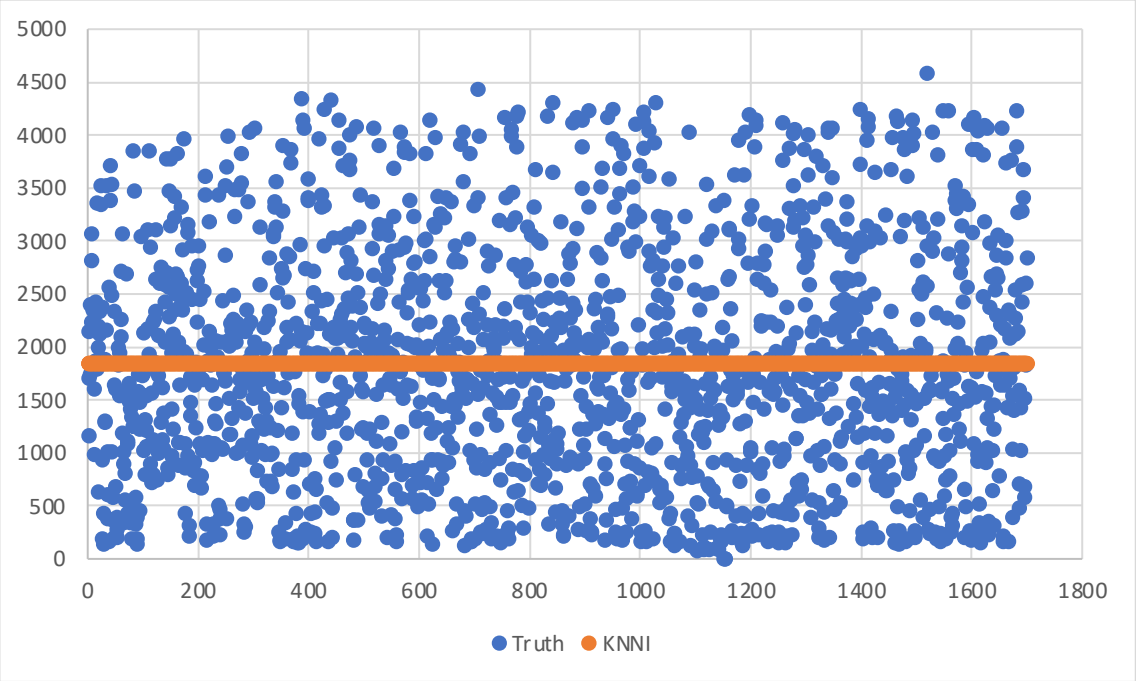


Figure 33 High Error Half Data KNNI Missing Value

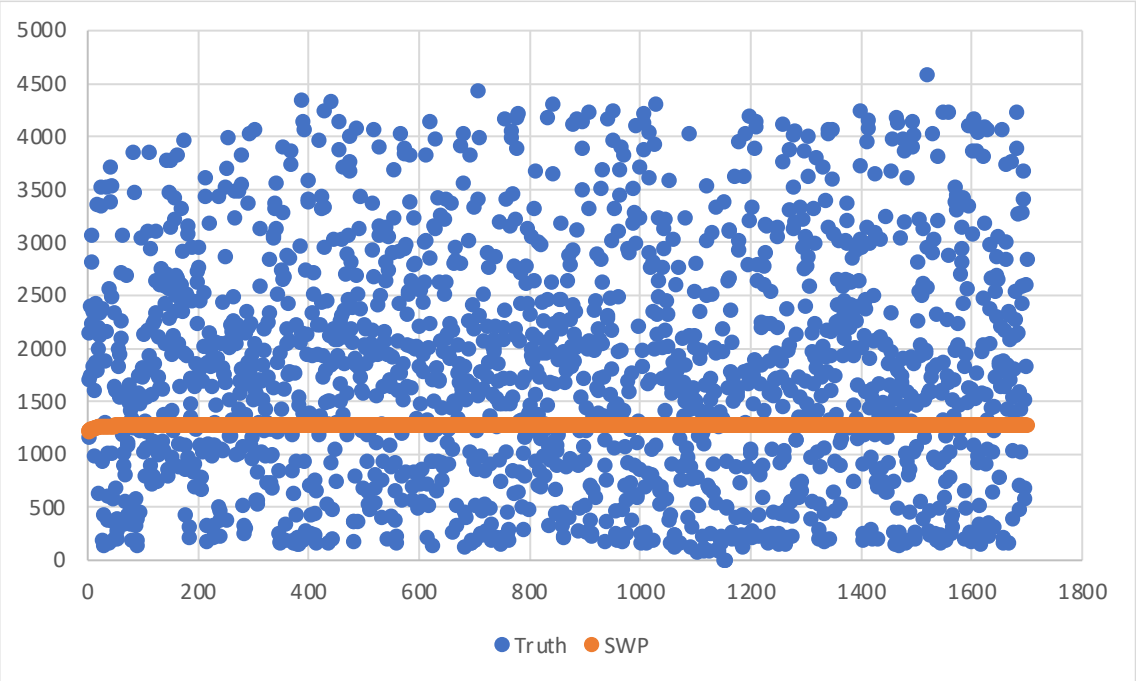


Figure 34 High Error Half Data SWP Missing Value

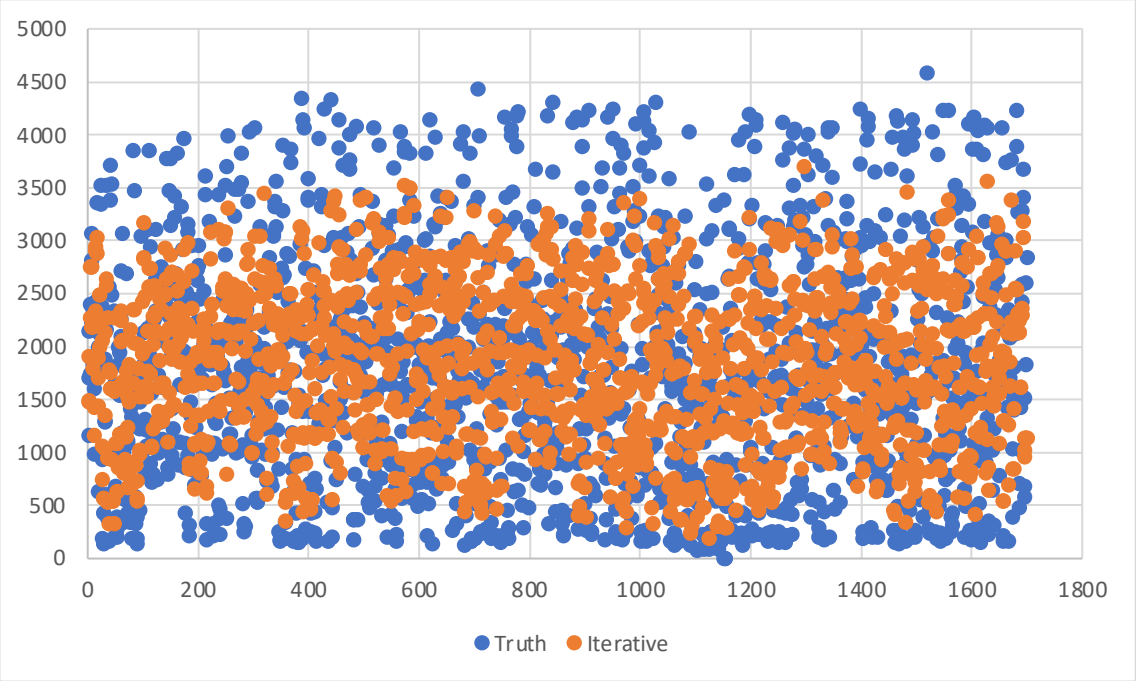


Figure 35 High Error Half Data Iterative Missing Value

This data seems to be inline with previous results, SWP & KNNI both returned values of low variety while iterative imputer has imputed values more inline with the original dataset but still with a lower level of volatility than the original data.

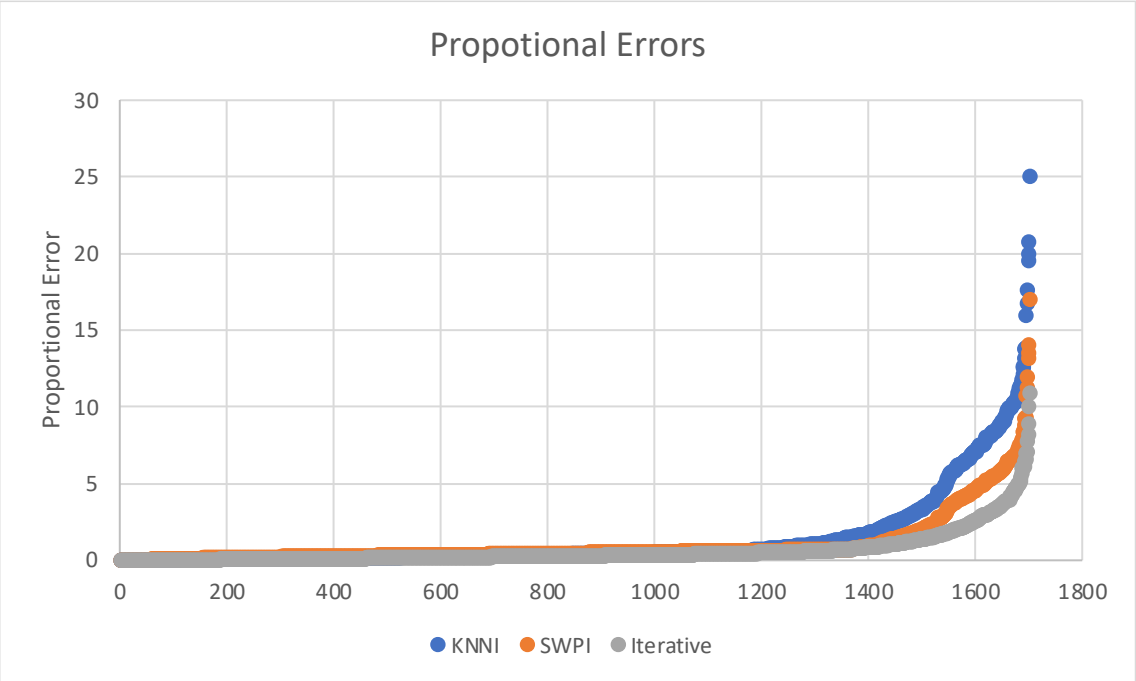


Figure 36 High Error Half Data Proportional Errors Missing Value

	Iterative	KNNI	SWPI
Median	0.28902252	0.40153687	0.47333385
Mean	0.64166186	1.38267604	1.00511316
Highest Inaccuracy	10.86875	25.0120626	17.0070151
Standard Deviation	1.0808127	2.63431887	1.68704956
RMSE(Whole Dataset)	450.9698677	683.9836799	757.7037827
RMSE(Imputed Values)	722.7245472	1090.663762	1211.92558
Time	0.026760633	0.3091653	0.2911209

Table 8 High Error Half Data Outputs

The results when regarding measures of inaccuracy from this experiment can be considered comparable to the experiment on the full dataset. It has been found that these algorithms' accuracy has not been affected by the size of the dataset on both low levels of MVs & high levels.

It took 3.6× longer to run KNNI & SWP on the full dataset which is greater than the proportional increase from doubling while iterative imputation increased by only 1.4× showing superior scaling capabilities with regard to time versus data quantity.

4.1.5 Summary

	Metric	KNNI	SWP	Iterative Imputation
Full Dataset With Low Error rate	Median Inaccuracy	0.401931	0.493138	0.188559
	Highest Inaccuracy	13.71914	9.146935	5.080198
	Mean Inaccuracy	1.260181	0.932888	0.370839
	Standard Deviation	2.251246	1.405239	0.555076
	RMSE (Full)	347.9121	395.0611	163.253
	RMSE (Imputed)	1153.003	1294.526	536.2076
	Time	0.304811	0.140756	0.030289933
Half Dataset With Low Error rate	Median Inaccuracy	0.41839596	0.50227844	0.19629331
	Highest Inaccuracy	1.44595621	1.07121083	0.39839243
	Mean Inaccuracy	13.4939419	9.14693484	5.08019802
	Standard Deviation	2.55400558	1.63706865	0.5993212
	RMSE (Full)	1153.629	1294.526	538.6348
	RMSE (Imputed)	359.1922	400.8757	166.2168
	Time	0.085971	0.079884	0.018973133
Full Dataset With High Error rate	Median Inaccuracy	0.401091	0.47299	0.276411
	Highest Inaccuracy	25.5249	17.00702	10.86875
	Mean Inaccuracy	1.272054	0.911225	0.607969
	Standard Deviation	2.336574	1.44611	1.011215
	RMSE (Full)	684.0106	762.9663	446.5713
	RMSE (Imputed)	1096.374	1212.166	719.9414
	Time	1.254652	1.305451	0.039089267
Half Dataset With High Error rate	Median Inaccuracy	0.40153687	0.47333385	0.28902252
	Highest Inaccuracy	1.38267604	1.00511316	0.64166186
	Mean Inaccuracy	25.0120626	17.0070151	10.86875
	RMSE (Full)	683.9837	757.7038	450.9699
	RMSE (Imputed)	1090.664	1211.926	722.7245
	Standard Deviation	2.63431887	1.68704956	1.0808127
	Time	0.309165	0.291121	0.026760633

Table 9 Missing Value Imputation Full

In terms of scaling the accuracy by error rate, KNNI was by far the worst affected as well as starting off with the poorest results. SWP did not start out as well as AR but their worst instances increased at the same rate, still the absolute value of SWP was worse than AR.

4.1.5.1 Effect of changing the error rate from 10% to 40% on full sized datasets

	Iterative	KNNI	SWP
Increase of Median proportional	1.465913	0.99791	0.959143
Increase of Mean proportional	1.639442	1.009422	0.976779
Increase of Max proportional	2.139434	1.860532	1.859314
RMSE(Imputed)	1.342654	0.950885	0.936378
RMSE(Whole)	2.735455	1.966044	1.931261

Table 10 Missing Value Imputation Accuracy Scaling

This experiment found links between the different algorithms' imputation accuracy & the rate of missing values within the dataset:

The Iterative imputation technique generally returned the best results on the individual tests but showed a worse trend when transitioning from a lower error rate to a higher error rate dataset. This technique was also found to perform at a reasonable speed regardless of error level.

Sliding Window Prediction returned generally mediocre accuracy but consistent. Unexpectedly, it imputed values which were (on average) slightly higher accuracy when presented with the higher error rate dataset than the low error rate. The improvements to accuracy were marginal & the worst case imputation was still worse on the high error rate data so it is NOT assumed that this algorithm generally works better on high error data. This technique was also found to perform at a comparatively poor speed on low error level data & extremely poorly when this level was increased.

The trends for k-Nearest Neighbours imputation were generally in line with those of sliding window prediction though with a much greater degree of inconsistency with a higher standard deviation & a greater disparity between mean & median. The worst-case value imputed by KNNI was also by far the worst in this experiment. This technique was found to perform poorly on the low error level data & the time did increase at an unacceptably high rate but not as high a rate as SWP.

4.1.5.2 Effect of doubling the size of a dataset

	Iterative	KNNI	SWP
Increase on Low Error Rate	1.596464474	3.545525641	1.762006258
Increase on High Error Rate	1.460700357	4.058191848	4.484222076

Table 11 Missing Value Imputation Time Scaling

Generally it was found that altering the size of the dataset had no bearing on the quality of the imputed data however there was a noticeable relationship between the size of a dataset & the time taken to impute the missing values.

It was found that on low error rate data that KNNI scaled very poorly when the size of the data increased on low error rates, with an increase of around 350% & it performed even worse on the high MV dataset with an increase of around 400%. It is clear that KNNI is very poor scaling algorithm.

SWP was found to have scale at a reasonable rate on low error data surpasses KNNI when used on data with a high rate of missing values.

Iterative Imputation performed well on both high error & low error rate data, while it did not perform as well on the higher error data, it was still bellow a ratio of 1:1 (time:filesize).

4.2 Outlier Correction

4.2.1 Full Dataset With low error rate

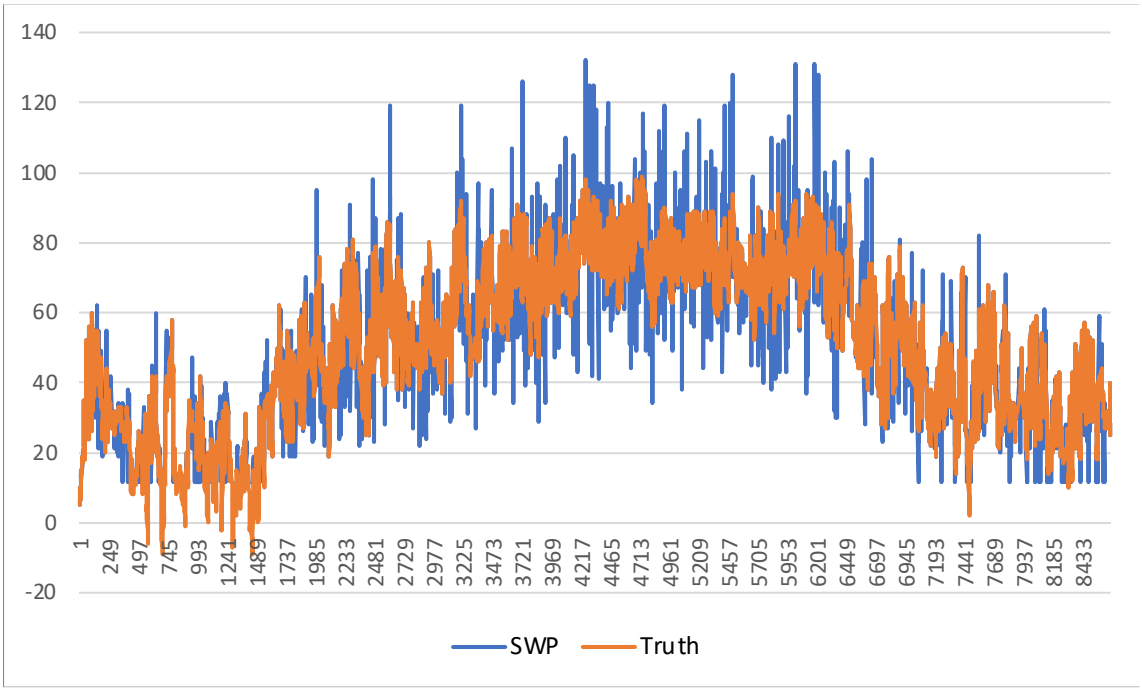


Figure 37 Low Error Full Data SWP

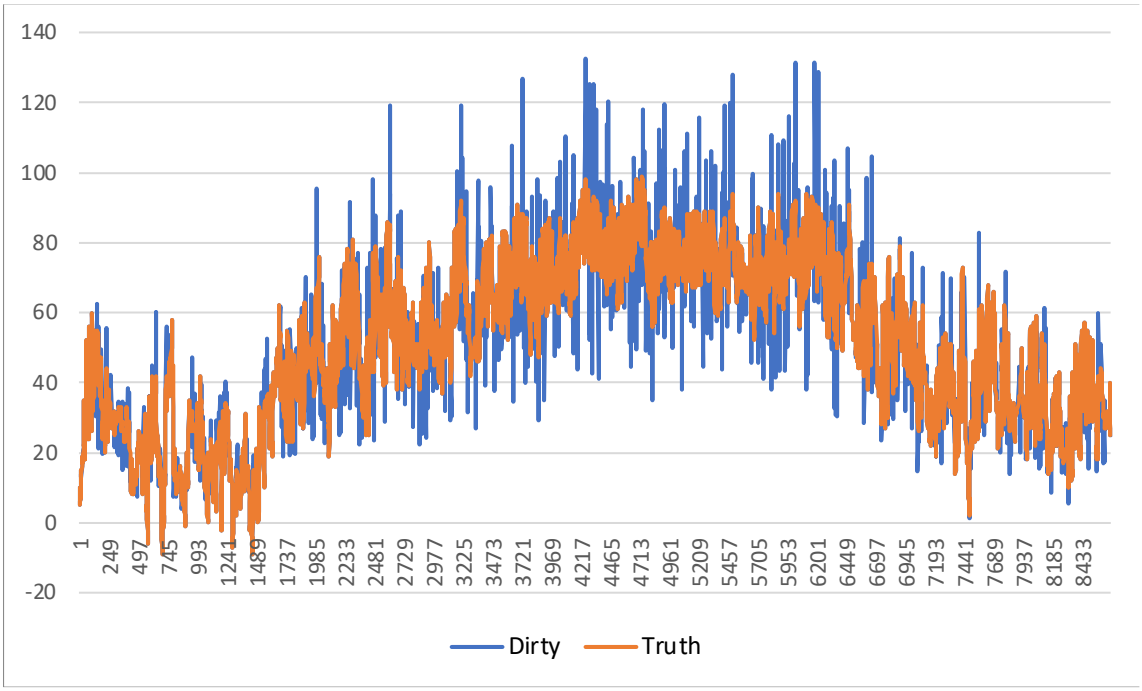


Figure 38 Low Error Full Data Dirty

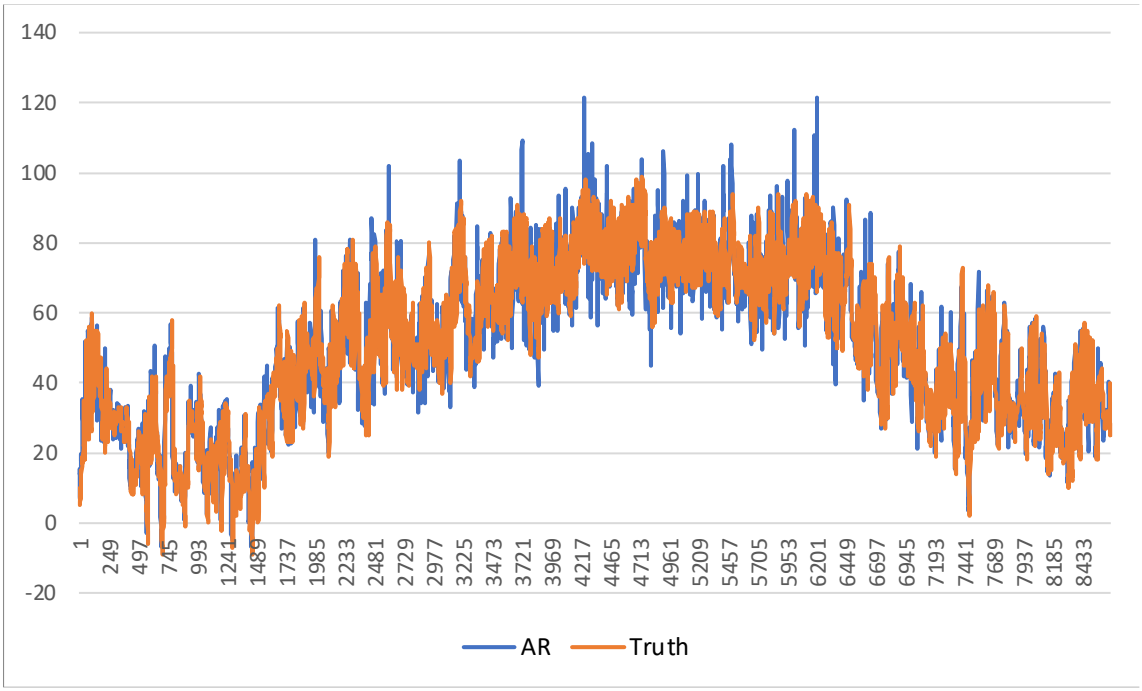


Figure 39 Low Error Full Data AR

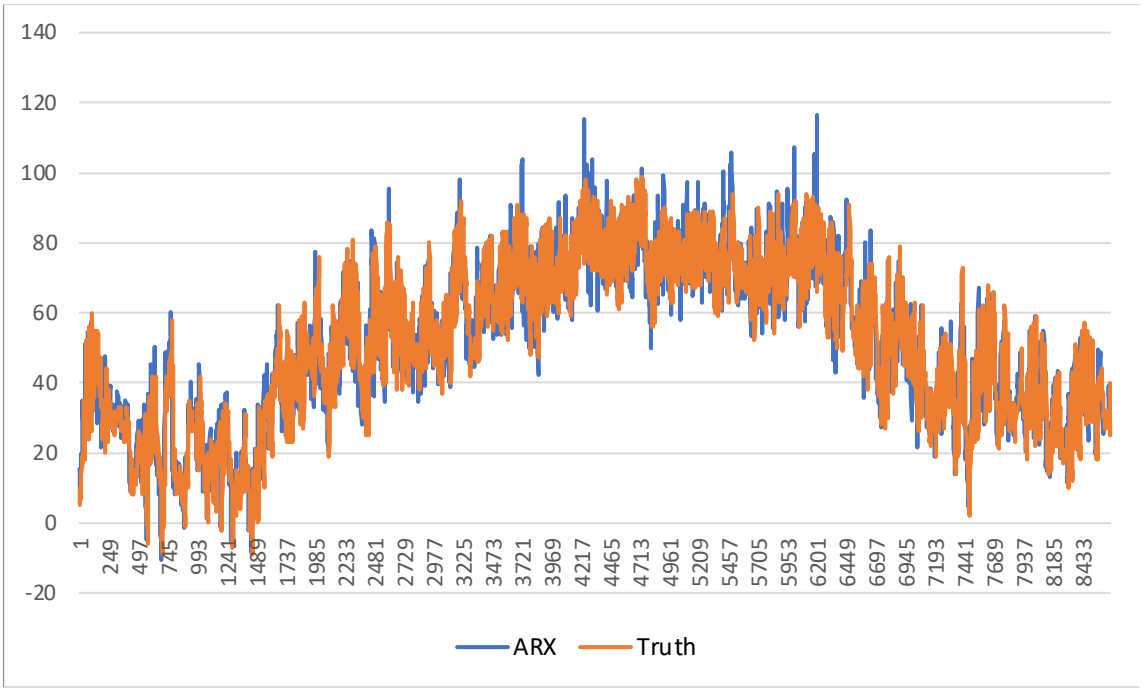


Figure 40 Low Error Full Data ARX

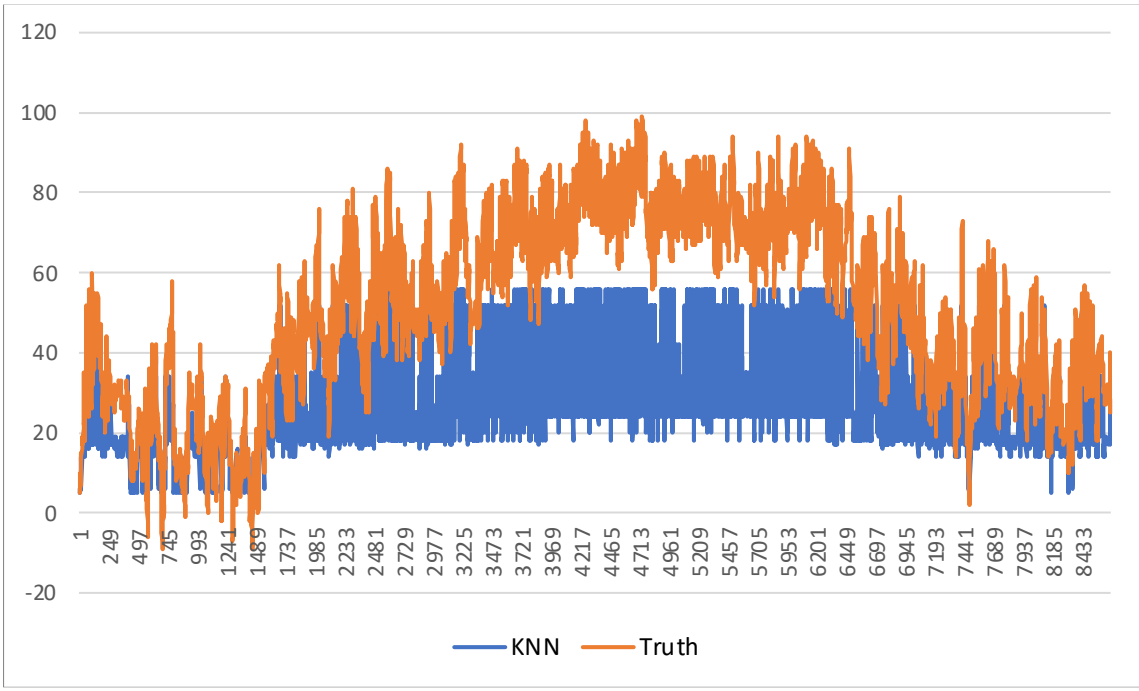


Figure 41 Low Error Full Data KNN

As can be seen the results for SWP are not especially different from the dirty data likely due to its ability to reject a calculated value in the case of low probability. AR & ARX both show cleaner data than the dirty data & ARX seems to have slightly more accurate data. KNN seems to have corrected with data closer to an average than following the trend expressed in the original dataset.

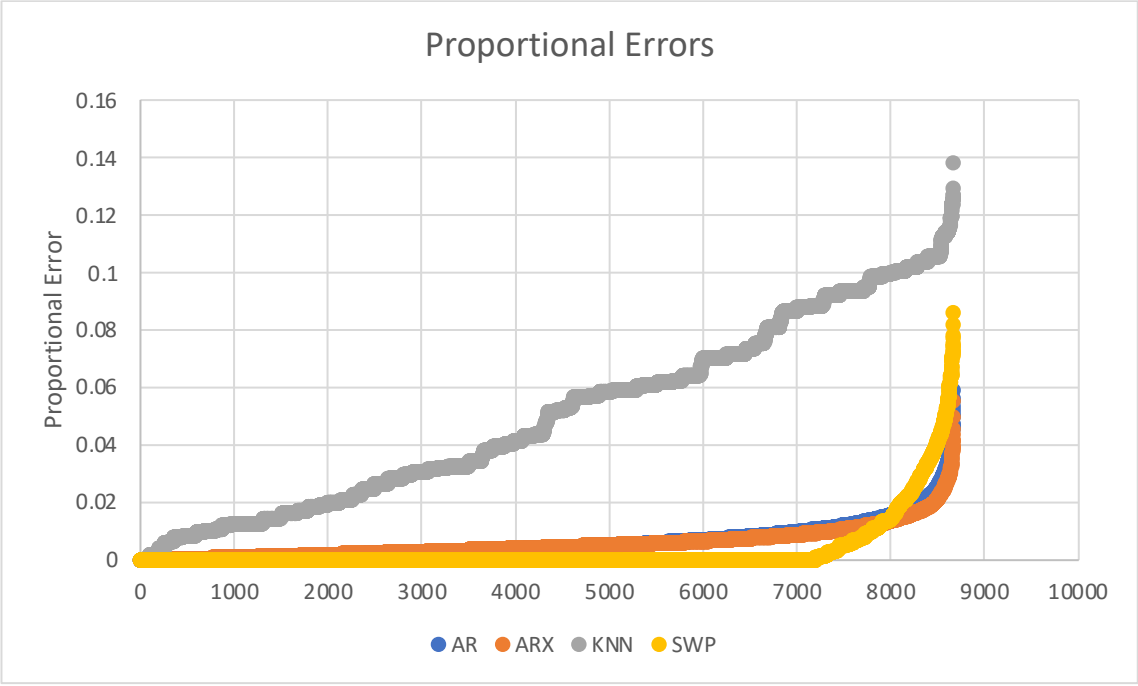


Figure 42 Low Error Full Data Proportional Errors

	AR	ARX	KNN	SWP
Max	0.059191	0.055479	0.138323	0.086227
Median	0.004294	0.00449	0.049051	0
Mean	0.006203	0.00578	0.050289	0.003187
Standard Deviation	0.006363	0.00531	0.031716	0.009785
RMSE	4.349757	3.912273	28.34284	5.223187
Mean Time	0.018515	0.016746	0.210156	4.012226

Table 12 Low Error Full Data Outputs

The RMSE for the dirty data was found to be 4.609634, as such it can be determined that SWP & KNN were unsuccessful in their attempts to clean the dataset as they made it dirtier.

As was expected, the majority of cleansed values from SWP were no different from the true value, due to its ability to assess the likelihood of a newly calculated value, this seems to be its only redeeming factor

The results from AR & ARX are quite similar with ARX returning values marginally better.

KNN performed terribly on this data, likely due to this dataset already being quite volatile & KNN only being able to take the exact values from neighbouring datapoints. This poor performance points to the likelihood that KNN is not an appropriate data cleaning method for time series data.

AR took 0.0185153 seconds, ARX took 0.016745933 seconds, KNN took 0.210156333 & SWP took 4.012226267. SWP performs a lot of calculations which turn out to be unnecessary after discarding them for having a poor PCI.

4.2.2 Half Dataset With Low Error Rate

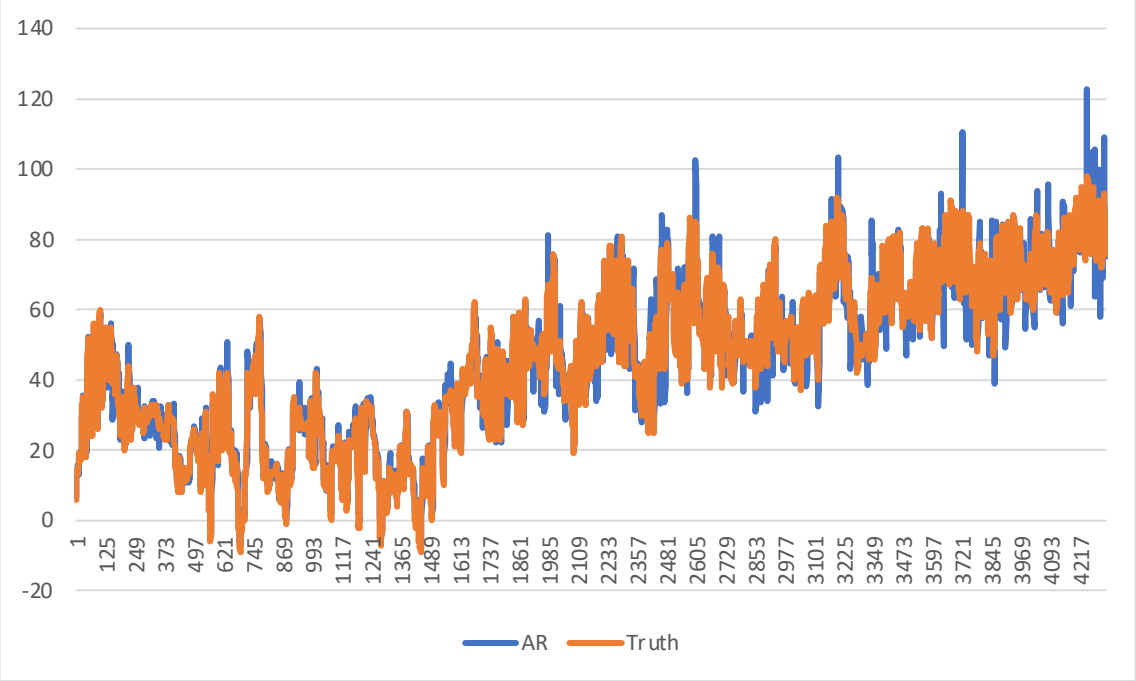


Figure 43 Low Error Half Data AR Outlier

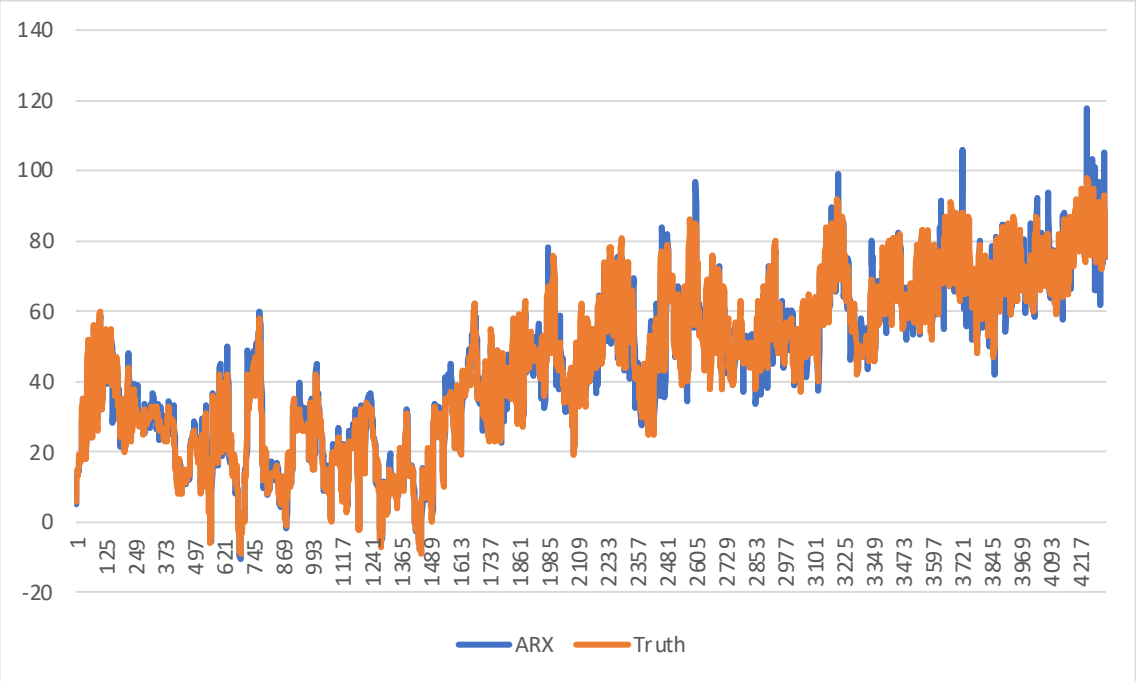


Figure 44 Low Error Half Data ARX Outlier

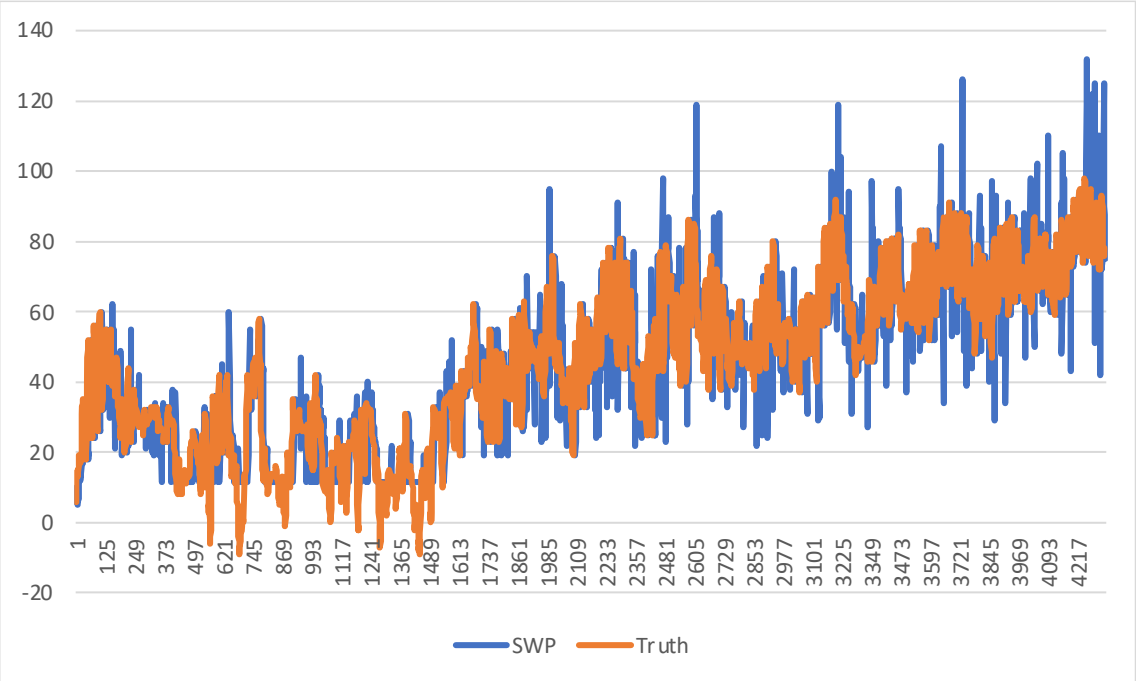


Figure 45 Low Error Half Data SWP Outlier

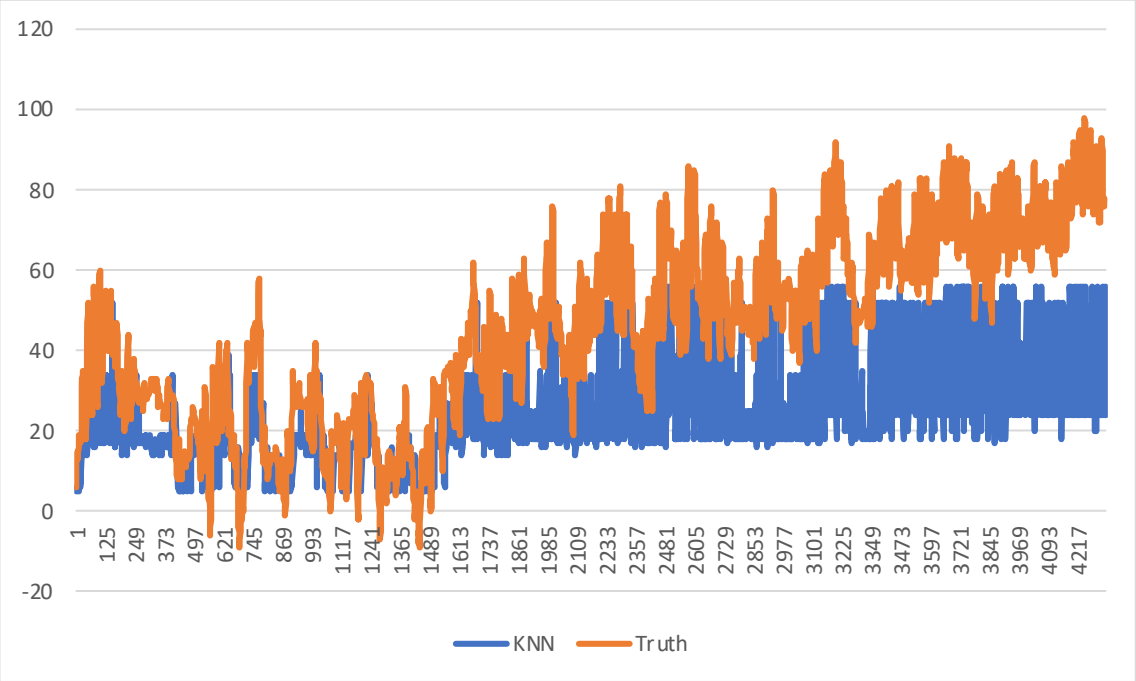


Figure 46 Low Error Half Data Truth Outlier

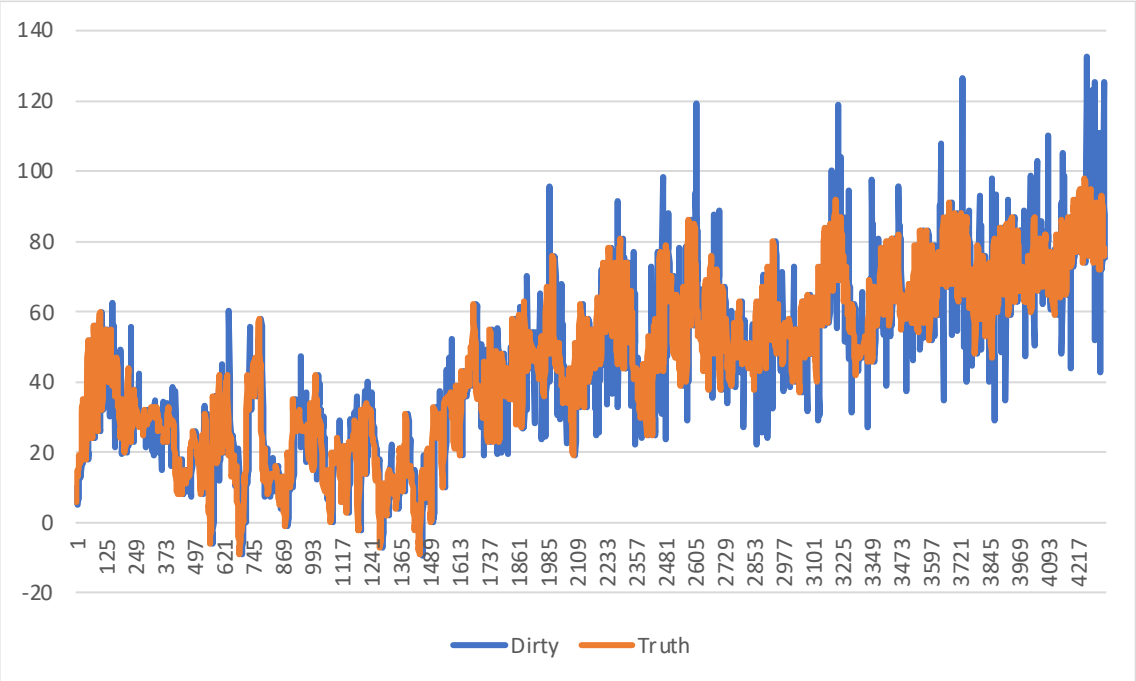


Figure 47 Low Error Half Data Dirty Outlier

These trends are generally inline with the full dataset… it should be noted that a large amount of the more volatile anomalies were present in the mid points of the dataset

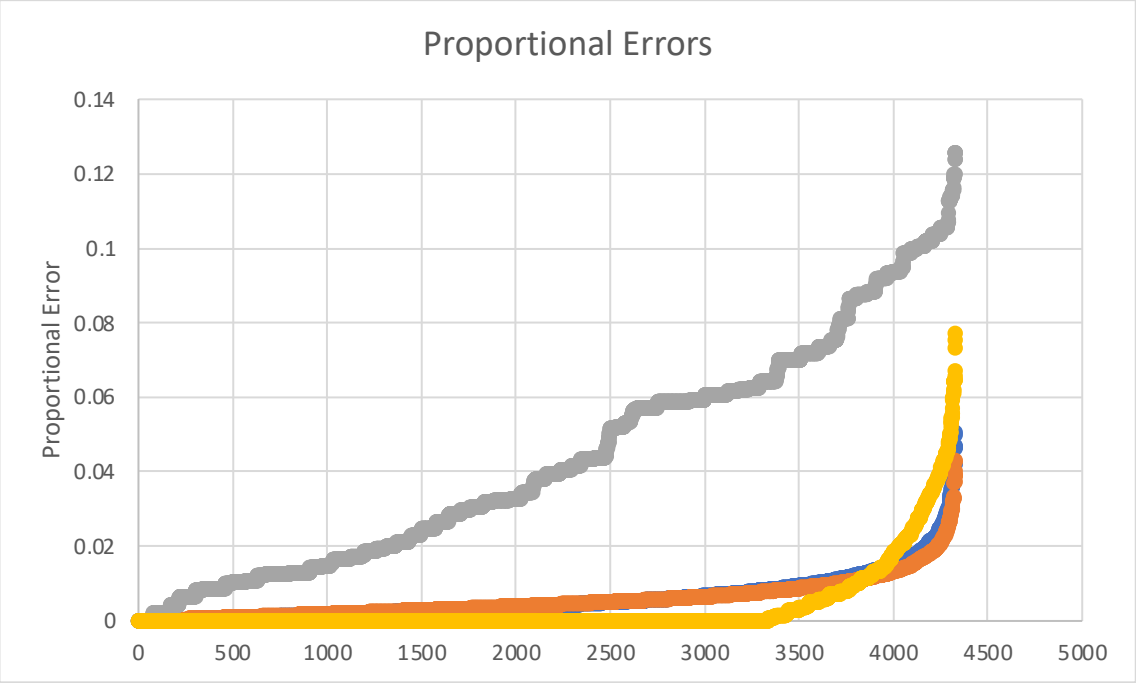


Figure 48 Low Error Half Data Proportional Error Outlier

	AR	ARX	KNN	SWP
Max	0.050431	0.042936	0.125759	0.077111
Median	0.003949	0.004226	0.03963	0
Mean	0.005855	0.005563	0.043466	0.003633
Standard Deviation	0.006107	0.00516	0.029921	0.009629
RMSE	4.354195	3.905678	27.75466	5.209444
Time	0.011954	0.020929	0.113882	1.801031

Table 13 Low Error Half Data Outputs Outlier

The RMSE of the dirty data was 4.57844 meaning that KNN & SWP were unable to improve the quality of this dataset.

The time taken for AR, KNN & SWP was found to be slightly higher but not by a long way (<1.1×) on the larger dataset while ARX had increased by 1.5×.

As the time taken for all of these algorithms to process a dataset was less than 2× the time taken for a dataset 2× the size, it can be concluded that all algorithms have scaled well with regard to time on a low error rate dataset.

4.2.3 Full Dataset with High error rate

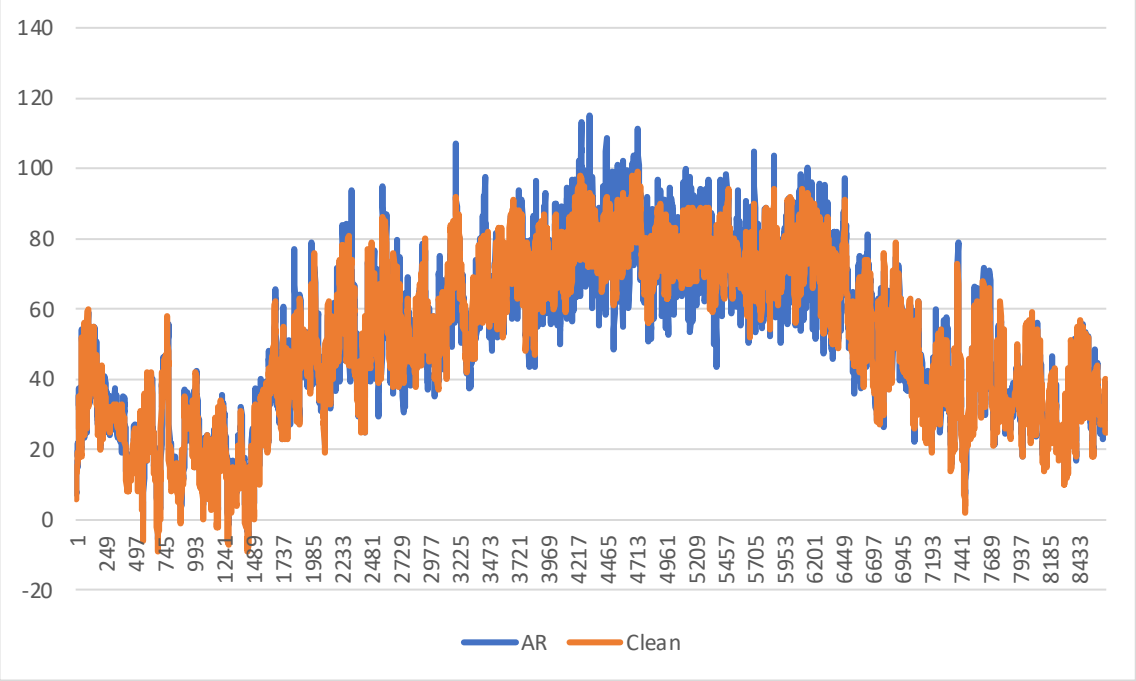


Figure 49 High Error Full Data AR Outlier

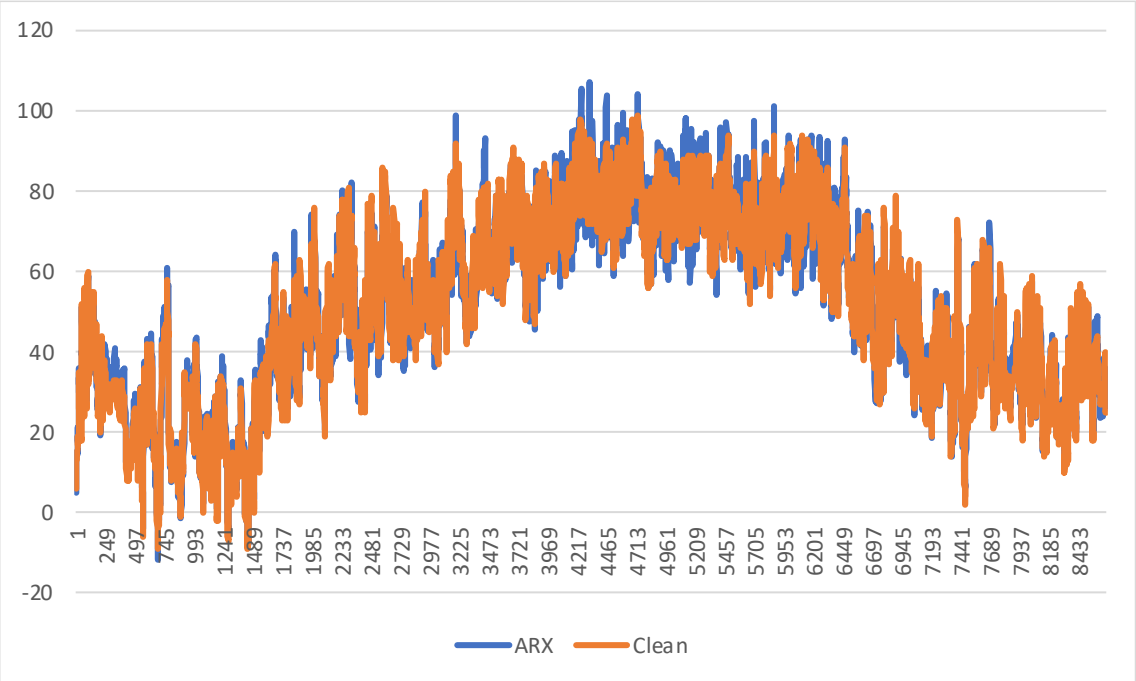


Figure 50 High Error Full Data ARX Outlier

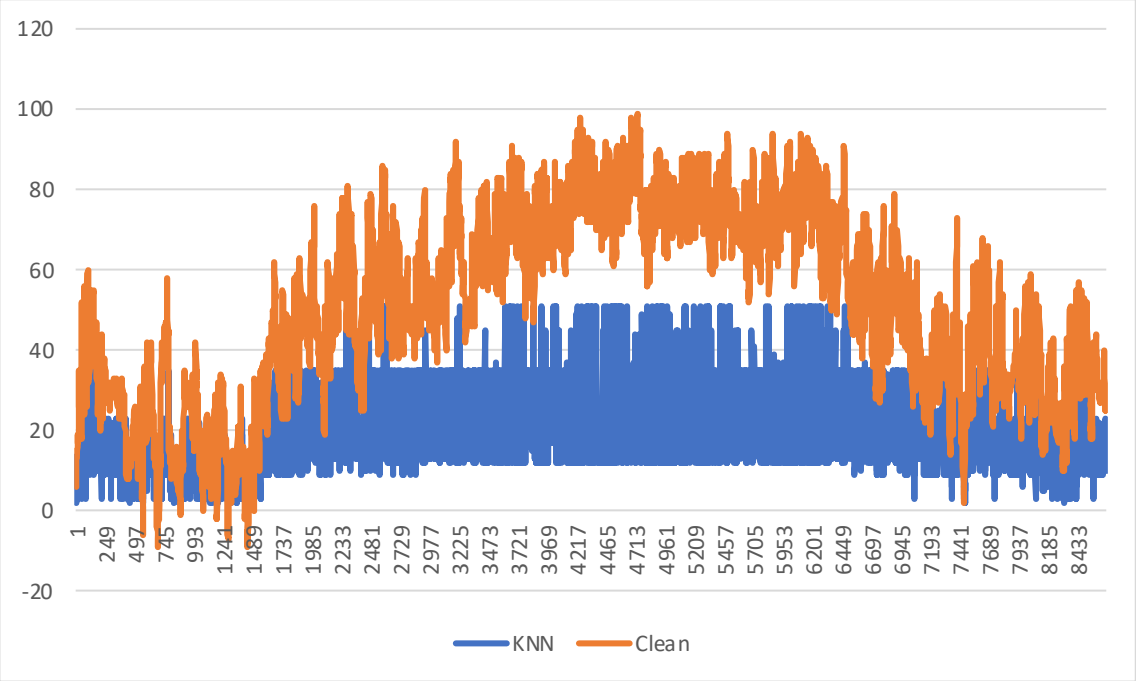


Figure 51 High Error Full Data KNN Outlier

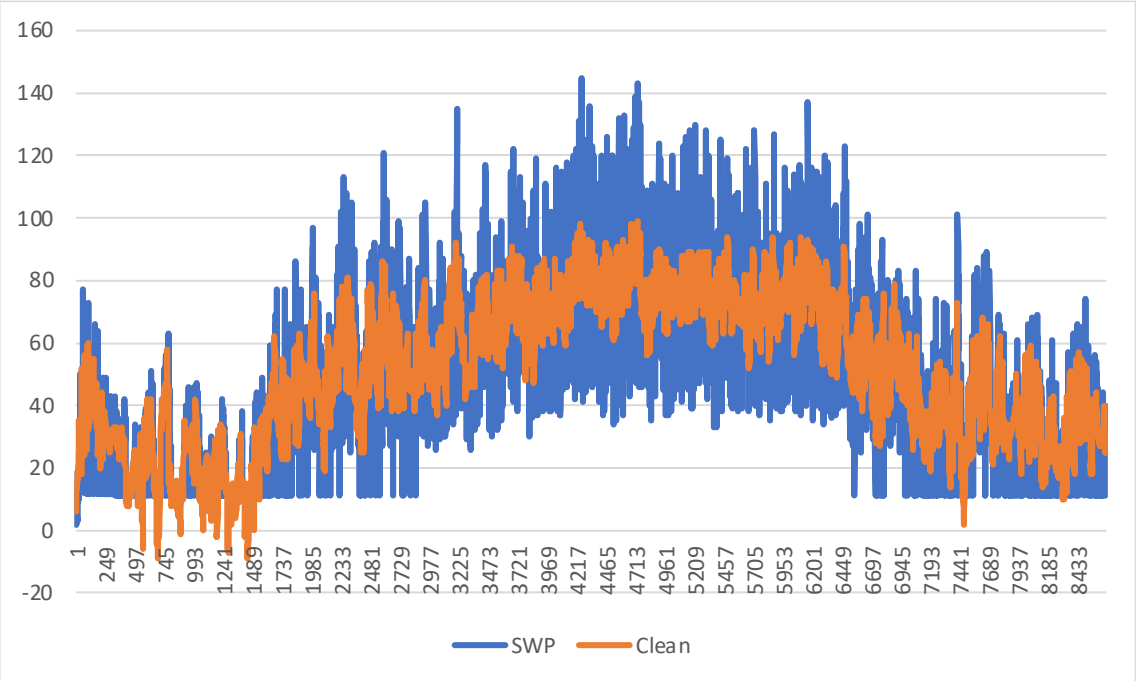


Figure 52 High Error Full Data SWP Outlier

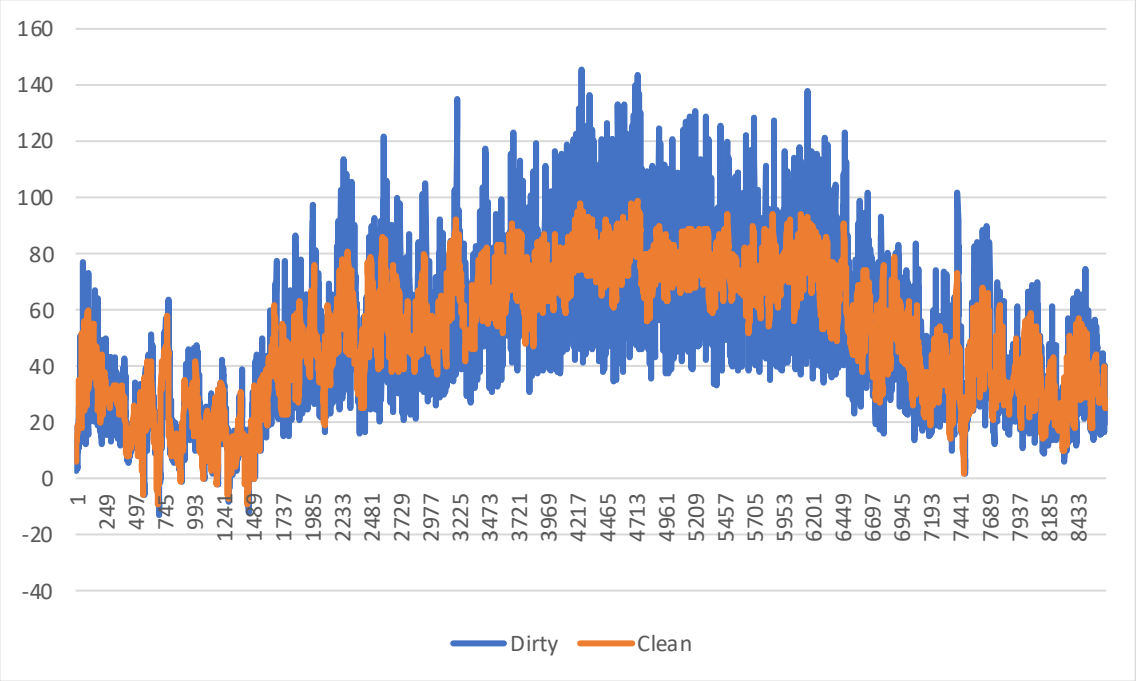


Figure 53 High Error Full Data Dirty Outlier

AR & ARX handle individual datapoints generally fairly well, there doesn't seem to be many/ any major outliers remaining, though it does seem that a lot of the values are still slightly off.

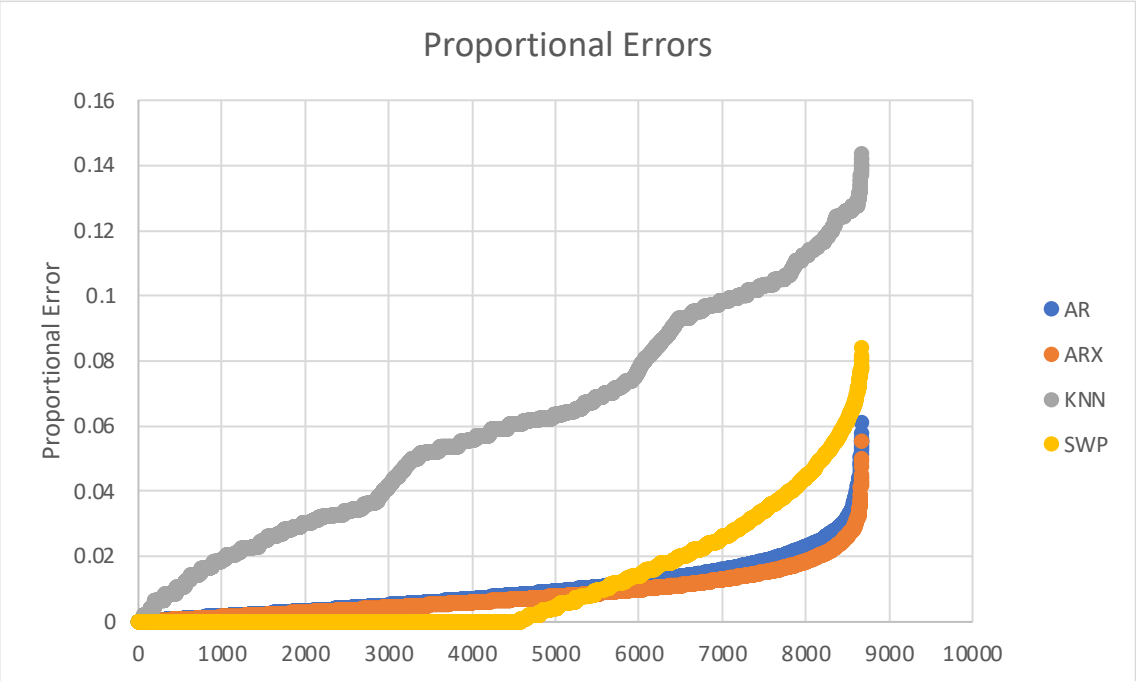


Figure 54 High Error Full Data Proportional Errors Outlier

	AR	ARX	KNN	SWP
Max	0.060958	0.055479	0.143462	0.084279
Median	0.00775	0.006468	0.058972	0
Mean	0.009759	0.008035	0.060553	0.0118
Standard Deviation	0.008203	0.006609	0.034055	0.017574
RMSE	6.282781	5.21699	33.27731	10.33632
Mean Time	0.018711	0.024819	0.212992	4.284844

Table 14 High Error Full Data Outputs Outlier

The dirty data's RMSE 9.419497389, once again SWP & KNN did not adequately correct this data while AR & ARX returned data closer to the true data than the dirty data.

The most obvious difference between this data and that of the low error rate data is the lower proportion of datapoints ignored by SWP, showing this to be an algorithm heavily affected by increasing the error rate. The results for KNN don't seem to be much worse though they started off quite poorly, this is likely due to the algorithm viewing all data in the set as anomalous & thus, introducing actual anomalies has no effect.

There is a slightly higher disparity between AR & ARX, with ARX taking the lead, by keeping the exogenous data unchanged, some aspects of the endogenous data can be retrieved.

The time taken by KNN was largely unchanged from low error rate to high error rate, again this is likely due to the algorithm treating most datapoints as anomalies. A similar effect can be found with SWP which only increased by 1.07×. The greatest time increase was ARX by 1.5×.

4.2.4 Half Dataset with high error rate

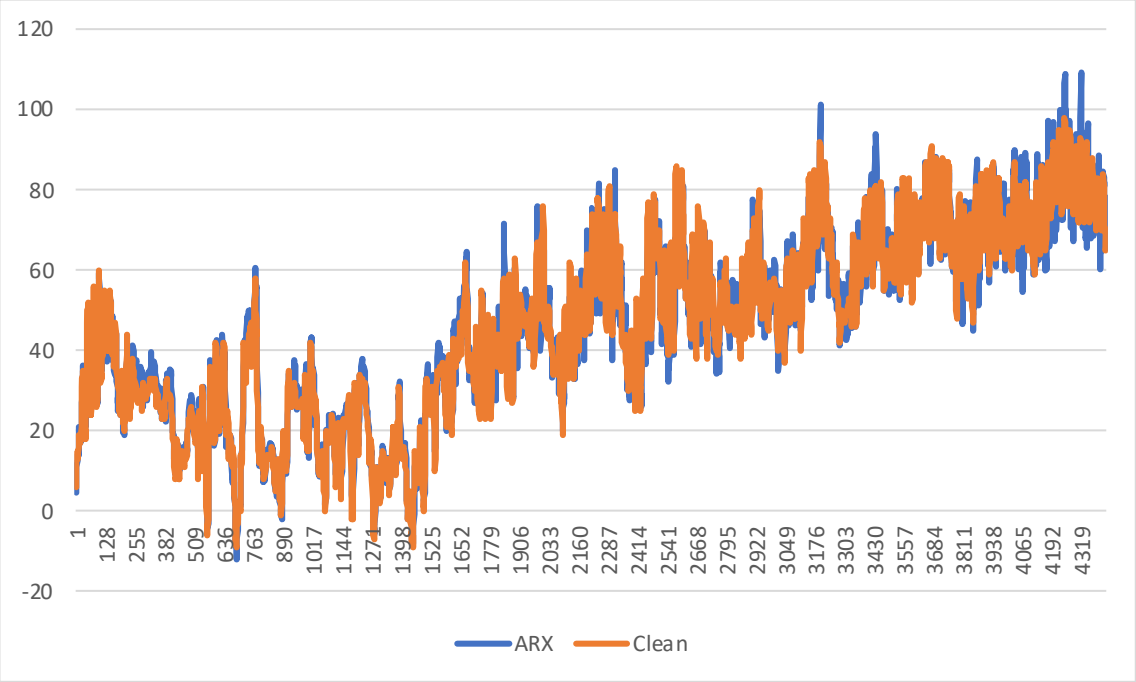


Figure 55 High Error Half Data ARX Outlier

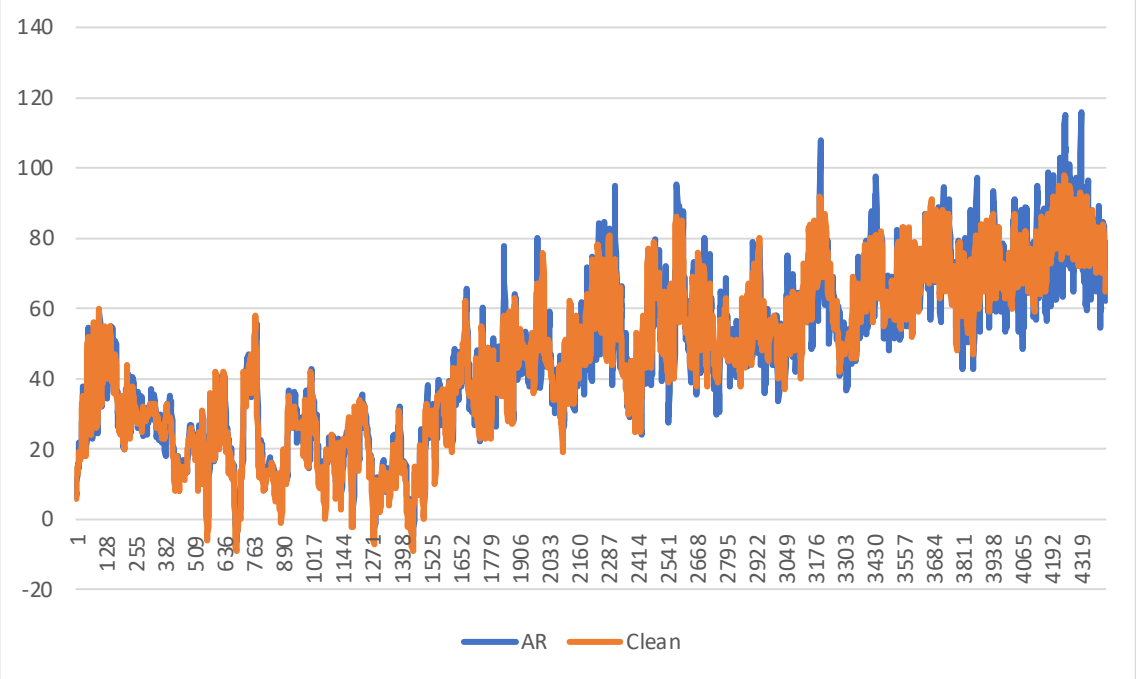


Figure 56 High Error Half Data AR Outlier

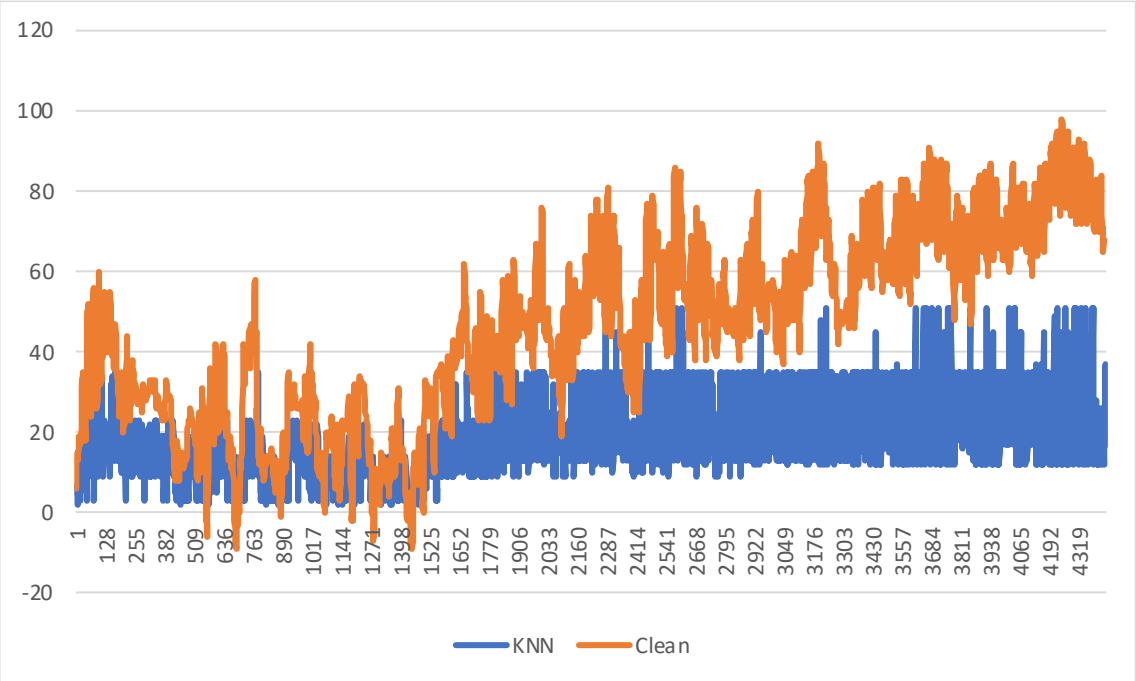


Figure 57 High Error Half Data KNN Outlier

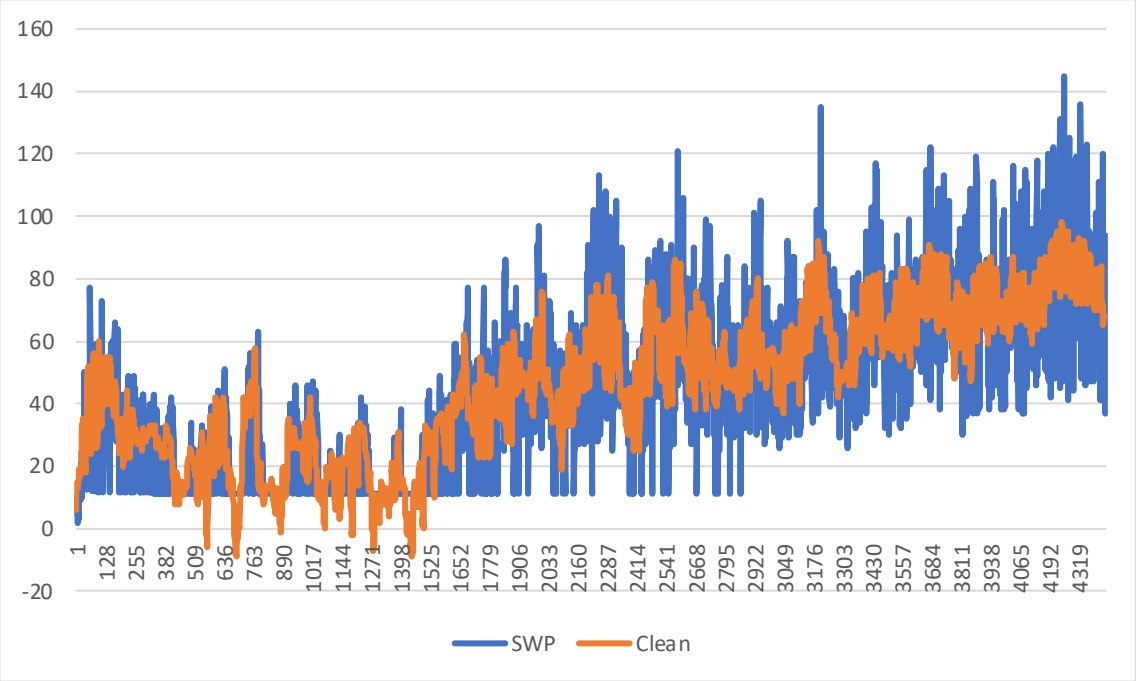


Figure 58 High Error Half Data SWP Outlier

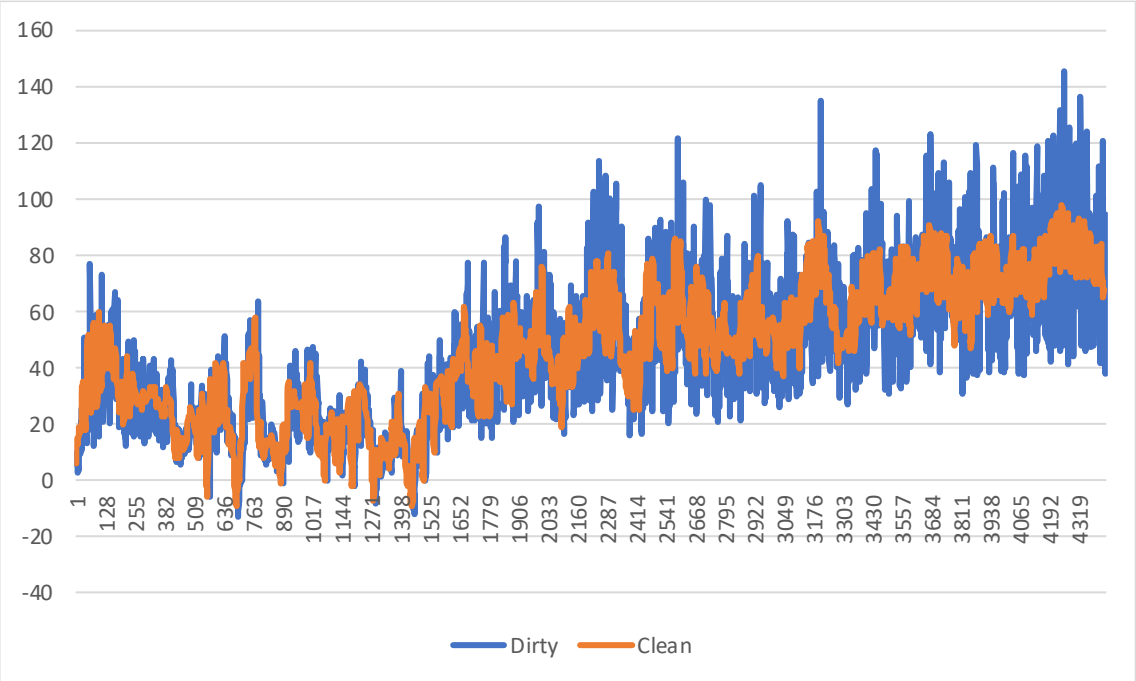


Figure 59 High Error Half Data Dirty Outlier

Once again, it can be seen that changing the size of the dataset does not necessarily alter the imputations made by KNN & SWP. AR & ARX on the other hand seem to have performed quite well on this data.

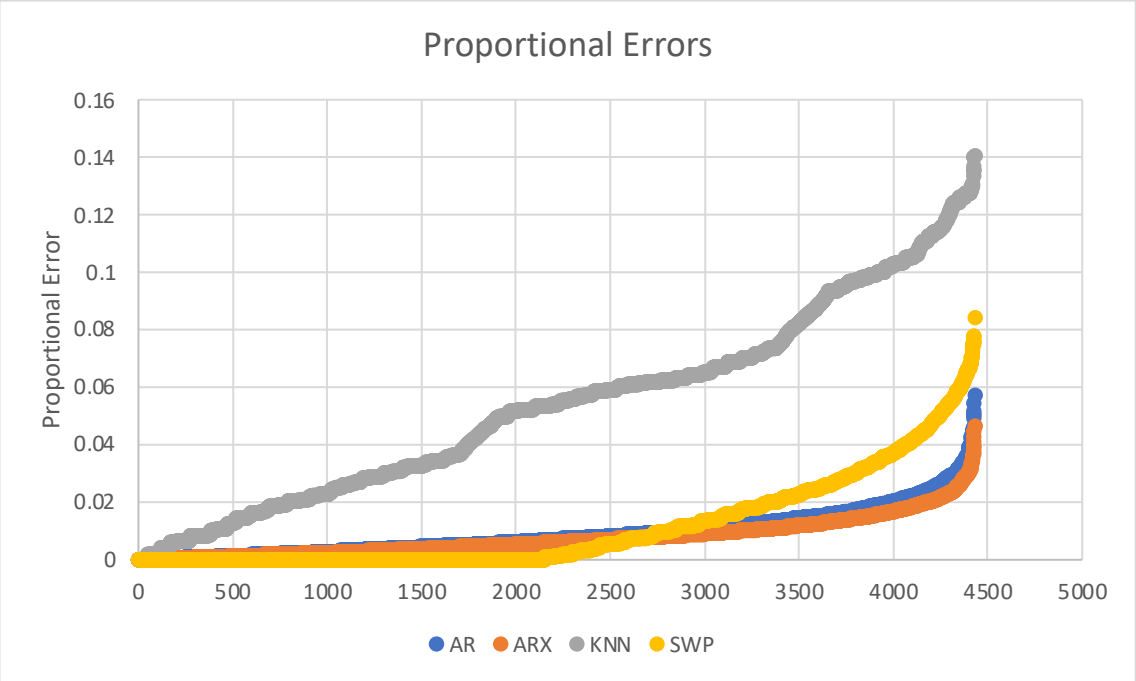


Figure 60 High Error Half Data Proportional Errors Outlier

	AR	ARX	KNN	SWP
Max	0.057345	0.046503	0.140624	0.084279
Median	0.0071	0.006082	0.05382	0.001264
Mean	0.009195	0.007696	0.053904	0.011479
Standard Deviation	0.00798	0.006503	0.033067	0.016429
RMSE	6.299919	5.226742	33.27731	10.33632
Time	0.021026	0.010232	0.113337	1.98722

Table 15 High Error Half Data Outputs Outlier

The RMSE of the dirty dataset was 9.419497 showing the same trend as before, SWP failed by a small amount, KNN failed significantly, ARX did slightly better than AR but both succeeded in cleaning that data.

Reducing the size of this data did not return clean data of a particularly different quality than the full dataset.

The time taken was generally was similar for both SWP & KNN as was seen before in the low error rate data. In the case of AR the time had significantly increased by 1.6× from small to large while ARX actually decreased in time with a larger dataset.

4.2.5 Summary

	Metric	AR	ARX	KNN	SWP
Full Dataset With Low Error rate	Median Inaccuracy	0.004294	0.00449	0.049051	0
	Highest Inaccuracy	0.059191	0.055479	0.138323	0.086227
	Mean Inaccuracy	0.006203	0.00578	0.050289	0.003187
	Standard Deviation	0.006363	0.00531	0.031716	0.009785
	RMSE	4.349757	3.912273	28.34284	5.223187
	Time	0.0185153	0.016745933	0.210156333	4.012226267
Half Dataset With Low Error rate	Median Inaccuracy	0.003949	0.004226	0.03963	0
	Highest Inaccuracy	0.050431	0.042936	0.125759	0.077111
	Mean Inaccuracy	0.005855	0.005563	0.043466	0.003633
	Standard Deviation	0.006107	0.00516	0.029921	0.009629
	RMSE	4.354195	3.905678	27.75466	5.209444
	Time	0.011954	0.020929	0.113882	1.801031
Full Dataset With High Error rate	Median Inaccuracy	0.00775	0.006468	0.058972	0
	Highest Inaccuracy	0.060958	0.055479	0.143462	0.084279
	Mean Inaccuracy	0.009759	0.008035	0.060553	0.0118
	Standard Deviation	0.008203	0.006609	0.034055	0.017574
	RMSE	6.282781	5.21699	33.27731	10.33632
	Time	0.018711	0.024819	0.212992	4.284844
Half Dataset With High Error rate	Median Inaccuracy	0.0071	0.006082	0.05382	0.001264
	Highest Inaccuracy	0.057345	0.046503	0.140624	0.084279
	Mean Inaccuracy	0.009195	0.007696	0.053904	0.011479
	Standard Deviation	0.00798	0.006503	0.033067	0.016429
	RMSE	6.299919	5.226742	33.27731	10.33632
	Time	0.021026	0.010232	0.113337	1.98722

Table 16 Outlier Repair Full Data

The comparison from Small Dataset to Large is as follows

	Error Rate	AR	ARX	KNN	SWP
Time	Low Error Rate	1.548823	0.800118	1.845379	2.227738
	High Error Rate	0.889904	2.425775	1.879273	2.1562
Median Inaccuracy	Low Error Rate	1.087348	1.06233	1.237737	N/A
	High Error Rate	1.091457	1.063373	1.095728	N/A
Mean Inaccuracy	Low Error Rate	0.943996	0.962497	0.864312	1.139855
	High Error Rate	1.061326	1.043983	1.123347	1.028007

Table 17 Outlier Repair Time Scaling

This table gives an insight to how effective each algorithm is at scaling to a larger dataset.

Seen here SWP will take longer for larger dataset (direct relationship) as it will increase at the same rate as the size of the data but maintains a consistent level of accuracy.

The AR & ARX models are heavily dependant on the error rate where ARX scales above a direct relationship for time at a high error rate but less than direct at a low error rate. The two algorithms seem to perform differently here, with no apparent reason for this combined with a relatively low disparity, it can be concluded that it is simply an anomaly. They both scale well in measures of accuracy remaining at ≤ 1.1.

While KNN scales quite well in measures fairly well in both time and accuracy measures. The time scaling is below a direct relationship & the accuracy is inline with that of the other algorithms though given it started so poorly, the effect could be explained by considering that there was no further down for it to go.

5 Conclusion & Future Work

5.1 Summary of Findings

All algorithms attempted on missing data imputation for the California transit dataset were ineffective, while iterative imputer was able to make some reasonable imputations, it should be concluded that the systems already in place by the transit authority are adequate.

SWP was unable to effectively clean both outlying data & missing values even on low error rate missing values the calculated values were generally very poor quality & lacked diversity, similar to the findings of the literature on high volatility datasets. It should be concluded that this algorithm only works effectively on very low volatility data.

KNNI was even more ineffective than SWP but with an iterative approach, using a KNN based measure of probability, it was possible to impute values much more effectively through the use of Iterative imputer.

KNN for outlier repair was found to be marginally successful in the early (low volatility) parts of the datasets but was prone to very poor results as volatility increased.

AR & ARX both algorithms performed well even with increases to the rates of anomalies, ARX performed marginally better as could be expected with the inclusion of an exogenous timeseries.

It should be noted that some of these algorithms performed poorly because they are not intended for the purposes used while other are.

5.1.1 Missing Value Imputation Literature Comparison

Due to the high levels of volatility within the timeseries the use of unsupervised univariate imputation algorithms was unsuccessful. The most effective algorithm overall was iterativeimputer which was able to scale effectively with regard to computational resources/time but was still unable to achieve an uncertainty of less than 28%.

An iterative approach was found to negate the risks of over correction discussed (A. Zhang et al., 2017) but to the extent that iterative imputer was found to be unable to reach a the full value it was attempting to estimate… this is not a reflection of the proposed IMR technique but simply an example of the principals discussed.

Once again, SWP was found to fail, it seems that the cause for this is in part due to its tendency to estimate values based on previous calculations, where the first imputed values in this dataset

were slightly different to the following imputations. Perhaps future incarnations of this algorithm should not output new data to the set it is currently processing.

5.1.2 Outlier Correction Literature Comparison

It was expected AR, would have a low strain on computer resources but would be easily thrown off by increasing the number of errors, with this effect lessened by using ARX.(A. Zhang et al., 2017). There were concerns over the use of autoregression on seasonal data, the requirement for data to have a clear trend was avoided through the use of lags. Both AR & ARX showed themselves to be excellent at removing outliers from this timeseries data in a short time without being significantly affected by changes in the rate of errors or the size of the dataset.

SWP was expected to 'fail' when presented with a highly volatile timeseries (Ranjan et al., 2019). This did not occur on either the high error or low error rate datasets. This could be down to the specifics of the implementation as the q-value of the t-test was not apparent in the paper. It is likely that by using a different q-value that the PCI was less likely to be high enough to justify the new value. As such, the SWP algorithm was unable to significantly alter dirty data, this was not necessarily a problem as the values calculated by SWP were often worse quality than the dirty data so by refusing to add them to the dataset, SWP avoided worsening it. It was not yet known at the start of this project how fast the SWP algorithm would run, it seems that due to the algorithm calculating a new value for each datapoint regardless of if the value is accepted combined with a total window size of 28 datapoints (recommended by literature) that the algorithm is heavily affected by the dataset's size.

KNN was expected to perform poorly on this data as it was being used to clean a univariate timeseries, something KNN is not intended to do, it performed significantly worse than expected & was found to be totally incapable of handling this dataset. A KNN based method had previously been used by (Ranjan et al., 2019) for simple outlier detection which it was capable of doing to an acceptable standard but seemingly, it was unable to correct outliers. The algorithm took a long time to run.

5.2 Self Appraisal

This project has successfully achieved its goals… the goal of this project was to implement, run & compare data cleaning algorithms & techniques. Factors relevant to the topic of data pre-processing such as time & accuracy in the face of different data sizes & rates of error were considered & associations discovered for each algorithm.

Throughout the course of this project, the technical aspects of these algorithms have been studied to attempt to identify possible areas of weakness & strength. Scientific papers were used to inform decisions such as the choice of datasets & the appropriate parameters of the algorithms. The expected findings from literature were used for comparison after the tests had been completed.

Not all algorithms used in this project could be easily migrated between outlier correction & missing data imputation as such comparisons could not be made on the basis of effectiveness on missing data imputation & outlier correction, though other algorithms were investigated on these types of data.

The goal of writing a tool for data cleaning was a success, with a graphic user interface & buttons allowing the user to read data from a file before cleaning & then writing to a file.

Due to the lack of an API key, data from WUnderground had to be manually downloaded from each page for each day. This took a large amount of time that could have been spent on other aspects of the project. In future, a more appropriate course of action would be to contact the author(s) of the paper to inquire if they still have the dataset & would be willing to share it.

Some of the initial algorithms considered for investigation were later discarded and replaced with other more appropriate ones. Algorithms such as WKNNI which were known to have similar results to KNNI, though it was not considered that the results would be exactly the same on timeseries data. Another instance of this was the expectation maximisation algorithm, for while a python package could not be found so it was replaced with another algorithm due to time considerations.

5.3 Further Work

One important aspect missed by this report would be the relative lack of diversity of datasets, by introducing a variety including the distinctions between seasonal data /non seasonal & univariate/multivariate.

Given the success of AR & ARX on outlier correction, it would be interesting to see how well techniques that make use of these algorithms can impute missing data,.(Chen et al., 2019) Introduces & compares some methods with each other, further work would include comparisons with a greater range of techniques such as iterative imputer.

As stated by (A. Zhang et al., 2017) AR & ARX should both be outperformed in terms of accuracy by iterative minimum repairing, it was considered out of the scope of this project to implement this algorithm in python but the results of a comparison would have been interesting.

References

Akouemo, H. N., & Povinelli, R. J. (2017). Data Improving in Time Series Using ARX and ANN Models. *IEEE Transactions on Power Systems*, *32*(5), 3352–3359. https://doi.org/10.1109/TPWRS.2017.2656939

Altukhova, O. (2020). Choice of method imputation missing values for obstetrics clinical data. *Procedia Computer Science*, *176*, 976–984. https://doi.org/10.1016/J.PROCS.2020.09.093

Ballou, D. P., & Pazer, H. L. (1985). Modeling Data and Process Quality in Multi-Input, Multi-Output Information Systems. *Http://Dx.Doi.Org/10.1287/Mnsc.31.2.150*, *31*(2), 150–162. https://doi.org/10.1287/MNSC.31.2.150

Batini, C., Di Milano, P., & Maurino, A. (2009). Methodologies for Data Quality Assessment and Improvement CINZIA CAPPIELLO CHIARA FRANCALANCI. *ACM Comput. Surv*, *41*(3), 16. https://doi.org/10.1145/1541880.1541883

Batini, C., & Scannapieco, M. (2016). *Data-Centric Systems and Applications Data and Information Quality*. https://doi.org/10.1007/978-3-319-24106-7

Cenitta, D., Arjunan, R. V., & K V, P. (2021). Missing Data Imputation using Machine Learning Algorithm for Supervised Learning. *2021 International Conference on Computer Communication and Informatics (ICCCI)*, 1–5. https://doi.org/10.1109/ICCCI50826.2021.9402558

Chen, X., Wang, H., Wei, Y., Li, J., & Gao, H. (2019). *Autoregressive-Model-Based Methods for Online Time Series Prediction with Missing Values: an Experimental Evaluation*.

Cheng Fan, Meiling Chen, Xinghua Wang, Jiayuan Wang, & Bufu Huang. (2021). A Review on Data Preprocessing Techniques Toward Efficient and Reliable Knowledge Discovery From Building Operational Data [Article]. *Frontiers in Energy Research*, *9*. https://doi.org/10.3389/fenrg.2021.652801

Cichy, C., & Rass, S. (2019). An Overview of Data Quality Frameworks. *IEEE Access*, *7*, 24634–24648. https://doi.org/10.1109/ACCESS.2019.2899751

Danette McGilvray. (2008). *Executing Data Quality Projects*.

Dasu, T., & Loh, J. M. (2012). Statistical Distortion: Consequences of Data Cleaning. *Proceedings of the VLDB Endowment*, *5*(11), 1674–1683. https://doi.org/10.48550/arxiv.1208.1932

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Source: Journal of the Royal Statistical Society. Series B (Methodological)*, *39*(1), 1–38. https://www.jstor.org/stable/2984875

Fazakis, N., Kostopoulos, G., Kotsiantis, S., & Mporas, I. (2020). Iterative Robust Semi-Supervised Missing Data Imputation. *IEEE Access*, *8*, 90555–90569. https://doi.org/10.1109/ACCESS.2020.2994033

García, S., Luengo, J., & Herrera, F. (2014). *Data preprocessing in data mining* (J. Luengo & F. Herrera, Eds.) [Book]. Springer. https://doi.org/10.1007/978-3-319-10247-4

Hacid, H., Sheng, Q. Z., Yoshida, T., Sarkheyli, A., & Zhou, R. (2018). *Data Quality and Trust in Big Data*. https://doi.org/10.1007/978-3-030-19143-6

Imputing missing values with variants of IterativeImputer — scikit-learn 1.2.2 documentation. (n.d.). Retrieved April 6, 2023, from https://scikit-learn.org/stable/auto_examples/impute/plot_iterative_imputer_variants_comparison.html#sphx-glr-auto-examples-impute-plot-iterative-imputer-variants-comparison-py

Jarke, M., Lenzerini, M., Vassiliou, Y., & Vassiliadis, P. (2002). *Fundamentals of Data Warehouses*. Springer Science & Business Media.

Kurgan, L. A., Cios, K. J., & Dick, S. (2006). Highly scalable and robust rule learner: Performance evaluation and comparison. *IEEE Transactions on Systems, Man, and*

Cybernetics, Part B: Cybernetics, *36*(1), 32–53. https://doi.org/10.1109/TSMCB.2005.852983

PeMS User Guide. (2020). https://pems.dot.ca.gov/Papers/PeMS_Intro_User_Guide_v6.pdf

Ranjan, K. G., Prusty, R., & Jena, D. (2019). *Comparison of Two Data Cleaning Methods as Applied to Volatile Time-Series; Comparison of Two Data Cleaning Methods as Applied to Volatile Time-Series*. https://doi.org/10.1109/PETPES47060.2019.9004012

Riaz, S., Arshad, A., & Jiao, A. L. (2018). *Rough Noise-Filtered Easy Ensemble for Software Fault Prediction*. https://doi.org/10.1109/ACCESS.2018.2865383

Shaoxu Song, Aoqian Zhang, Jianmin Wang, & Philip S. Yu. (2015). *SCREEN: Stream Data Cleaning under Speed Constraints*. https://sxsong.github.io/doc/15sigmod-screen.pdf

Shengjie Liu, Guangye Li Shize, Jiang Xiaolong Wu, Jie Hu, Dingguo Zhang, & Liang Chen. (2021). *Investigating Data Cleaning Methods to Improve Performance of Brain–Computer Interfaces Based on Stereo-Electroencephalography*. https://europepmc.org/article/pmc/pmc8528199

Sidi, F., Hassany, P., Panahy, S., Affendey, L. S., Jabar, M. A., Ibrahim, H., & Mustapha, A. (2012). Data quality: A survey of data quality dimensions. In *2012 International Conference on Information Retrieval & Knowledge Management*. https://doi.org/10.1109/InfRKM.2012.6204995

Sinaga, K. P., & Yang, M.-S. (2020). Unsupervised K-Means Clustering Algorithm. *IEEE Access*, *8*, 80716–80727. https://doi.org/10.1109/ACCESS.2020.2988796

Thomas C. Redman. (1996). *Data Quality for the Information Age*.

Wand, Y., & Wang, R. Y. (1996). Anchoring data quality dimensions in ontological foundations. *Communications of the ACM*, *39*(11), 86–95. https://doi.org/10.1145/240455.240479

Wang, R. Y., & Strong, D. M. (1996). Beyond Accuracy: What Data Quality Means to Data Consumers. *Https://Doi.Org/10.1080/07421222.1996.11518099*, *12*(4), 5–33. https://doi.org/10.1080/07421222.1996.11518099

Yu, Y., Zhu, Y., Li, S., & Wan, D. (2014). Time Series Outlier Detection Based on Sliding Window Prediction. *Mathematical Problems in Engineering*, *2014*, 1–14. https://doi.org/10.1155/2014/879736

Zhang, A., Song, S., Wang, J., & Yu, P. S. (2017). *Time Series Data Cleaning: From Anomaly Detection to Anomaly Repairing*. http://ise.thss.tsinghua.edu.cn/sxsong/doc/anomaly.pdf

Zhang, R., Indulska, M., & Sadiq, S. (2019). Discovering Data Quality Problems The Case of Repurposed Data. *Business & Information Systems Engineering*, *61*. https://doi.org/10.1007/s12599-019-00608-0

Appendix 1 Initial Project Document

Sam Dewar 40437441

Initial Project Overview

SOC10101 Honours Project (40 Credits)

Title of Project:

A comparison of methods and techniques for data cleaning within data pre-processing

Overview of Project Content and Milestones The Main Deliverable(s):

- Script for comparing data cleaning algorithms.
- Final report containing:
    - analysis of the script's outputs

o lit review
o methodology (including dataset and description of the script) o identification and description of algorithms

The Target Audience for the Deliverable(s):

- Data scientists
- Data analysists
- Researchers
- Anyone using machine learning

The Work to be Undertaken:

- Research:
    - Data cleaning algorithms
    - Dirty Data
    - Evaluation metrics for data cleaning o Find data to run the algorithms on
- Lit review (based on research)
- Write a methodology
- Write a python script to compare algorithms
- Perform analysis on the results of the algorithm

Additional Information / Knowledge Required:

- Key metrics for measurement of algorithm performance
- Different kinds of dirty data (including the source of dirty data and its relationship with

each algorithm)

- What dataset to use (size, datatype, etc)
- What algorithms to compare
- Details of the algorithms (how they work, pros & cons)

Sam Dewar 40437441

Information Sources that Provide a Context for the Project:

Data Pre-processing in data mining (book from 2014), A Review on Data Pre-processing Techniques Toward Efficient and Reliable Knowledge Discovery From Building Operational Data (Article from 2021), Machine Learning and Big Data Analytics Paradigms: Analysis, Applications and Challenges (Book from 2021)

The Importance of the Project:

Important to machine leaning where corrupt data may need to be corrected... potentially in ‘big data’

The Key Challenge(s) to be Overcome:

- Researching algorithms
- Categorising dirty data types and causes (then finding dataset)
- establishing evaluation metrics
- making a script for comparing the algorithms
- statistical analysis of the results

Appendix 2 Second Formal Review Output

SOC10101 Honours Project (40 Credits) Week 9 Report

Student Name: Samuel Dewar **Supervisor**: Taoxin Peng **Second Marker**: Alistair Lawson **Date of Meeting**: 11/11/22

Can the student provide evidence of attending supervision meetings by means of project diary sheets or other equivalent mechanism? **yes no***

If not, please comment on any reasons presented

Please comment on the progress made so far

Project started a bit lte but is on track

Is the progress satisfactory? **yes no***
Can the student articulate their aims and objectives? **yes no***
If yes then please comment on them, otherwise write down your suggestions.

Yes, but they need more clearly stated in the introduction, and also the conclusion of the lit review should let the reader know what has been learned for the literature that will be applied to the project, and aim and objectives clarified in light of the literature of required

* Please circle one answer; if **no** is circled then this **must** be amplified in the space provided

Does the student have a plan of work? **yes no***
If yes then please comment on that plan otherwise write down your suggestions.

Plan of work could be more understandable

Does the student know how they are going to evaluate their work? **yes no*** If yes then please comment otherwise write down your suggestions.

Any other recommendations as to the future direction of the project

The literature needs some more up to date papers, and more than one paper per technique

Signatures: Supervisor Taoxin Peng Second Marker Alistair Lawson

Student Sam Dewar

The student should submit a copy of this form to Moodle immediately after the review meeting; A copy should also appear as an appendix in the final dissertation.

* Please circle one answer; if **no** is circled then this **must** be amplified in the space provided

Appendix 3 Diary Sheets

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Samuel Dewar **Supervisor: Dr Taoxin Peng**

1. **Date: 30/09/2022** **Last diary date: First**

Objectives:

1. Define aims and objectives of the project
2. Create a G-Chart (project plan)
3. Complete IPO
4. Start literature review
    a. Search references, especially research papers.
    b. Background study
5. Look at the module's website, pay attention to the following information:
    a. How to do a literature review
    b. How to search information, papers etc
    c. What is a methodology
    d. Find the dissertation template
    e. Find a couple of sample dissertations
6. Reference list

Progress:

Completed IPO document
Defined objective (comparison of data cleaning algorithms)
Found some relevant papers (details in IPO document)
Made a zube.io chart for the next month (vague outline for beyond)
Read over some of the materials on Moodle
Found a couple of previous dissertations to look through

Read over dissertation from previous years on the same topic as mine
Adjusted IPO document to more clearly display my plan for the honours project

Supervisor's Comments:

A slow start. However, it is good that the aims and objectives are defined. It is good to have a plan for next month, which should be refined. A plan for the whole project is useful, even though the outline is vague.

It is suggested to use the dissertation template for writing up.

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Samuel Dewar **Supervisor: Dr Taoxin Peng**

2. **Date: 14/10/2022** **Last diary date: 30/09/22**

Objectives:

7. Modify G-Chart (project plan) with more detailed tasks and milestones
8. literature review - Continue
    a. techniques/methods/algorithms
    b. existing similar work
    c. Find some datasets
9. Experiments – How are you going to do them?
    a. Use existing tools, like Weka Or
    b. Create an app by Python, which has most of algorithms in its libraries.
10. Reference list

Progress:

- Found scikit-learn python library, seems like it may contain a lot of the tools I may need
- Updated zube.io plan with “epics”(project milestones) and accompanying tasks detqailing the writing of my lit review, the creation of my comparison script as well as the analysis & methodology for the final report
- Continued work on lit review: wrote mainly on data imputation methods

Did not attend meeting due to family concerns

- Continued work on essay, rough draft complete
- Found Caltrans dataset (in total 60% is missing which will be useful for imputation algorithms, given this, the data accuracy may be low)

Supervisor’s Comments:

A reasonable progress. At this stage, it is suggested that you focus on algorithms and define which sets of algorithms you are going to compare, and how to do the comparison.

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Samuel Dewar **Supervisor: Dr Taoxin Peng**

3. **Date: 28/10/2022** **Last diary date: 14/10/22**

Objectives:

11. Define the structure of the dissertation, and add a brief introduction to each of the chapters.
12. literature review - Continue
    a. techniques/methods/algorithms
    b. existing similar work
    c. Find some datasets – a couple of more
    d. Metrics for comparing
13. Experiments – Methodology
    a. Create an app by Python, which has most of algorithms in its libraries.
14. Reference list

Progress:

1. Added headings and some text to each section
    a. Also reformatted document, introduction no longer in the lit review, table of contents, etc
2. Lit review
    a. defined algorithms to compare
    b. Added section on other work that compares data cleaning algorithms
    c. Found PVOutput data (found some others but I am still working on getting access)
    d. Defined metrics for comparison
3. Metrics for comparison now defined,
    a. Basic plan of how to compare (use python to plot datasets before and after cleaing, time cleaning and apply to various datasets)
4. Reference list included in document

Supervisor's Comments:

Very good progress, found some relevant datasets. A few of them are fine for testing.

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Samuel Dewar **Supervisor: Dr Taoxin Peng**

4. **Date: 09/12/2022** **Last diary date: 28/10/22**

Objectives:

15. Complete literature review
    a. Define algorithms for comparing
16. Datasets collection – 2 or 3 will be fine for the project
17. Experiment design – Methodology
    a. Evaluation metrics
    b. Experiment cases
18. Software design
    a. Requirements; both functional and non-functional
    b. Architecture of the application
    c. Models
    d. UML diagrams might be needed.
19. Implementation
    a. Create an app by Python, which has most of algorithms in its libraries.
    b. Test some function, such as input, output
20. Reference list

Progress:

1. literature review has been largely completed
   a. Algorithms defined
2. Datasets have been found, (caltans & wunderground)
3.
   a.metrics defined, for time, accuracy with different sizes & error rate
   b. cases defined for varied sizes of dataset & level of error

5.
   a.simple gui written & relevant python modules found

Supervisor's Comments:

A very good progress made during the holiday. Since your main task is comparing algorithms, it might be good to have a simple design in implementation. However, the experiment design is important, which should focus on

- Datasets collected
- Features of each algorithm
- Differences among algorithms
- Differences between datasets

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Samuel Dewar **Supervisor: Dr Taoxin Peng**

5. **Date: 26/01/2023** **Last diary date: 09/12/22**

Objectives:

21. Experiment design – Methodology
    a. Method of the experiment
    b. Evaluation metrics
    c. Experiment cases
        i. Focus on datasets and algorithms
        ii. Consider features of each algorithms and differences
    d. Architecture of the experiment

22. Software design (might be merged with implementation)
    a. Requirements; both functional and non-functional
    b. UML diagrams might be needed.
23. Implementation
    a. Create an app by Python, which has most of algorithms in its libraries.
    b. Test some function, such as input, output
24. Reference list

Progress:

1.
A. experiment plan altered according to features of the algorithms in order to get a better idea of the accuracy metric (vaguely: introduce dirty data, clean, compare with original)
B. accuracy metric now takes into account the ability of an algorithm to clean the dataset back to the point before dirty data was introduced
C. section written on chosen algorithms along with potential findings/ justification for cases Found the user manual for California transit data which contains their methods for imputation
Table of dataset sizes, number of errors & magnitude of errors created

2.
a. will require: all 6 algorithms specified, graphic user interface, reading from & writing to files, random dirty data injection script, potentially also the ability to specify automated tasks for experiments requiring repetition

3.
Implemented & checked 3/6 of the algorithms on a small dataset
Expecting the rest to be done by next week

Supervisor's Comments:

Very impressive progress. It is very good to have all datasets and algorithms prepared. There might be too many test cases. You can divide them into two parts: Must-Do and Optional.

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Samuel Dewar **Supervisor: Dr Taoxin Peng**

6. **Date: 16/02/2023** **Last diary date: 26/01/22**

Objectives:

25. Experiment design
    a. Must-Do: 4 datasets for each type of errors (outlier, missing values): Size (Small, Large); error level (light. Heavy). 8 datasets in total
    b. Optional: Add 2 datasets for each type of errors (outlier, missing values): Size (medium); error level (medium). 4 datasets in total
26. Implementation
27. Testing and evaluation
28. Writing up

Progress:

1. updated table of datasets to specify optional & non optional, will prepare the datasets over the weekend
2. Finished implementing algorithms, tool should be working, any issues will be dealt with during experiments

Supervisor's Comments:

Another solid progress made. It is great to have all algorithms implemented at this stage. So you can start experiments and analysing/evaluation, paying more attention to differences among the datasets.

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Samuel Dewar **Supervisor: Dr Taoxin Peng**

7. **Date: 09/03/2023** **Last diary date: 16/02/22**

Objectives:

29. Experiment
30. Analysing and evaluation
31. Writing up

Progress:

1.dataseets mostly prepared, python script written to inject dirty data, will run this after meeting & can start experiments
2.will begin today
3.added section outlining the datasets for the experiment & description of the python experiment

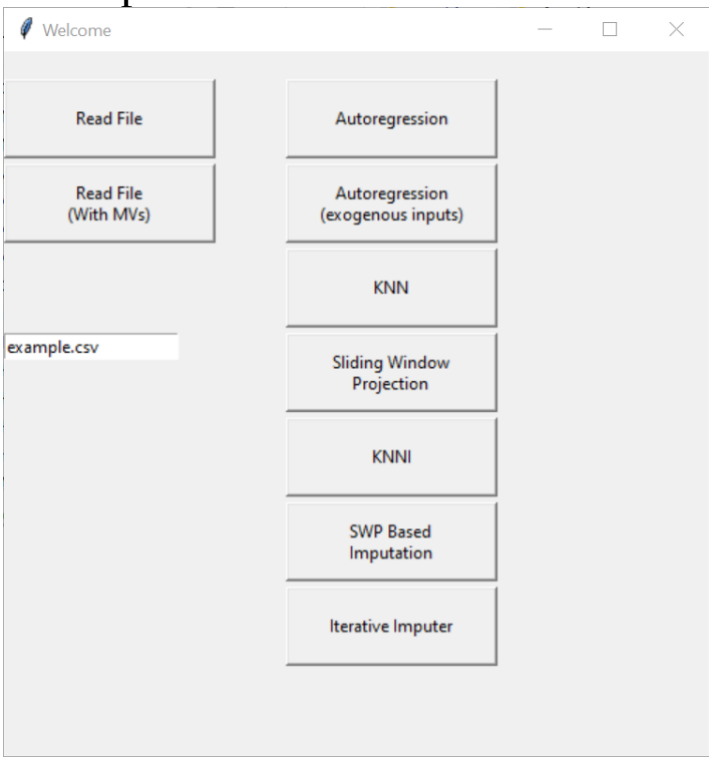
Supervisor's Comments:

Excellent! The way of preparing the 8 datasets for experiments needs to be described in detail with the purpose. Also go back to the findings in literature review, make sure to compare your results against the findings (existing similar work) regarding different types of datasets and algorithms.

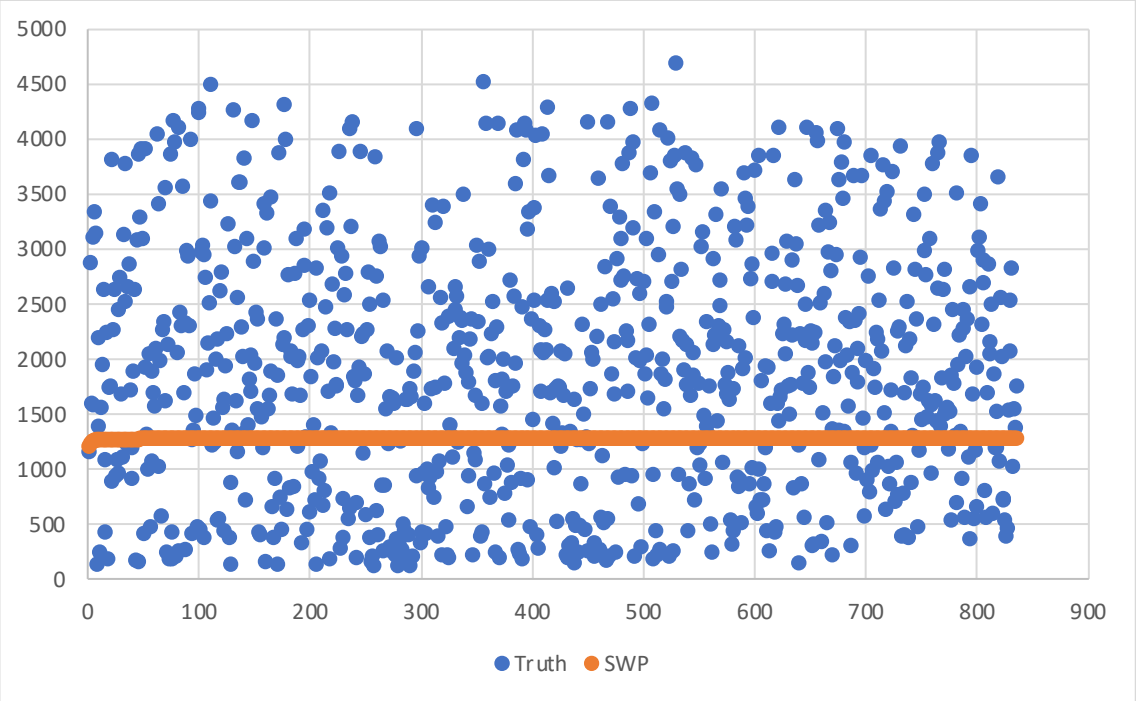
Appendix 4 Outline Of Datasets

Detection/Repair	
Length	Number of injected errors (%)
8,680	10
8,680	40
4,340	10
4,340	40
Imputation	
Length	Number of MVs (%)
8,760	10
8,760	40
4380	10
4380	40

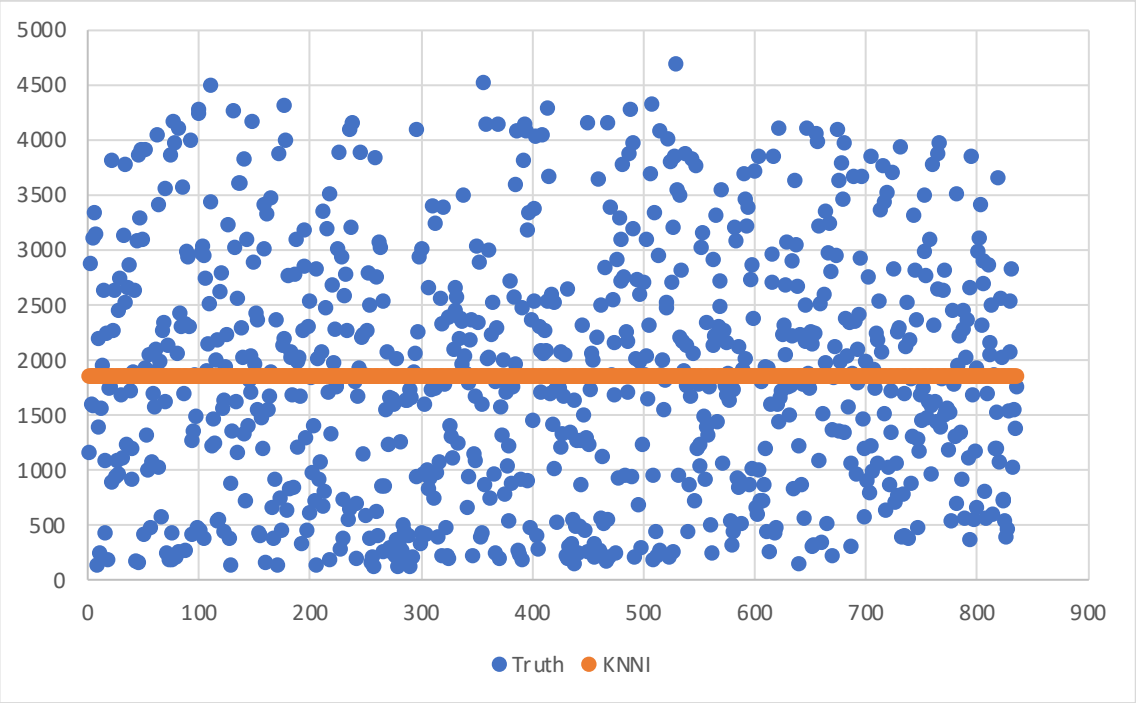
Appendix 5 Graphic User Interface



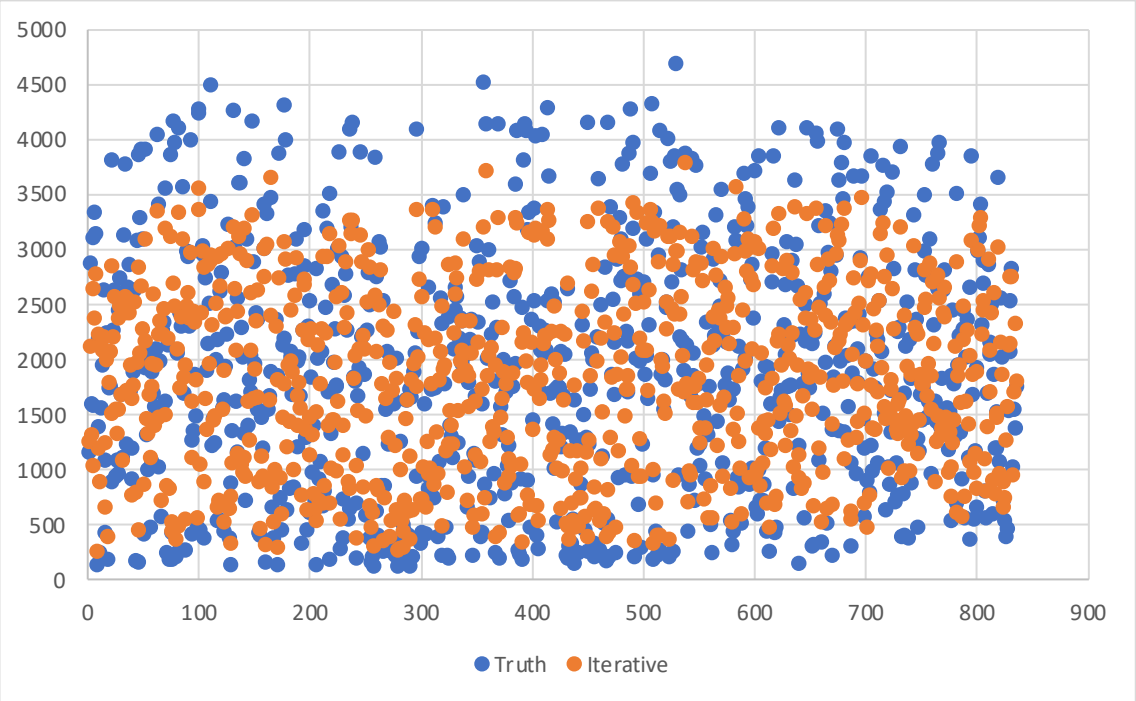
Appendix 6 SWP Output on Low Rate Missing Values



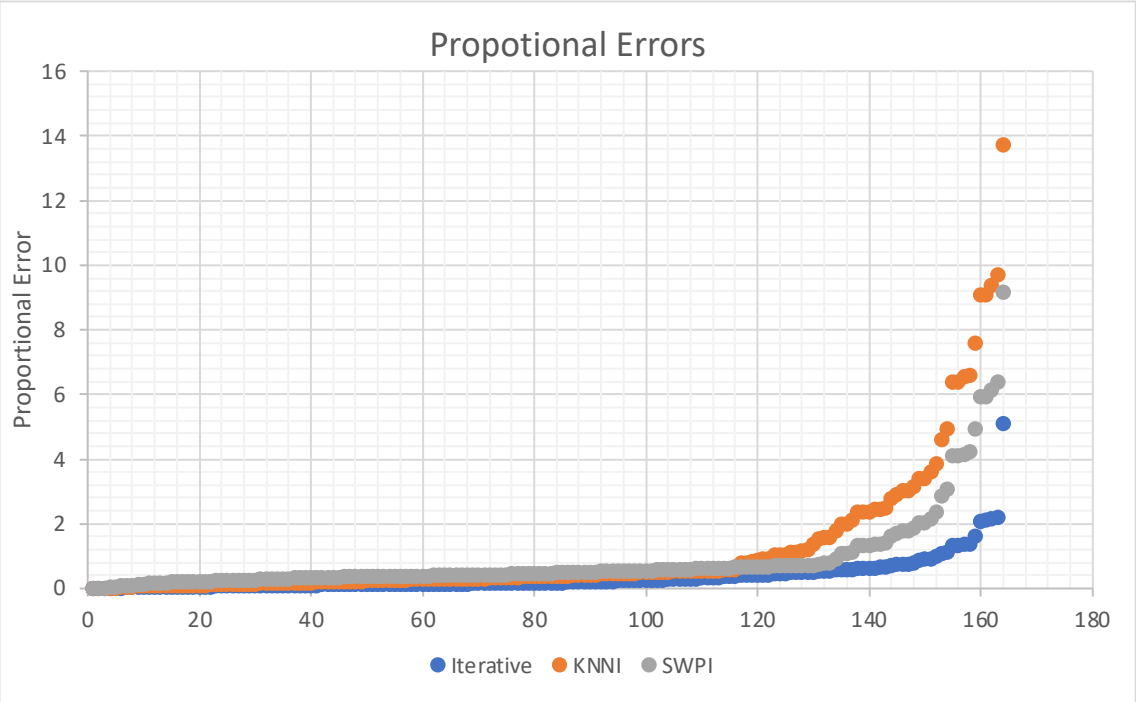
Appendix 7 KNNI Output on Low Rate Missing Values



Appendix 8 Iterative Imputer Output On low Rate Missing Values



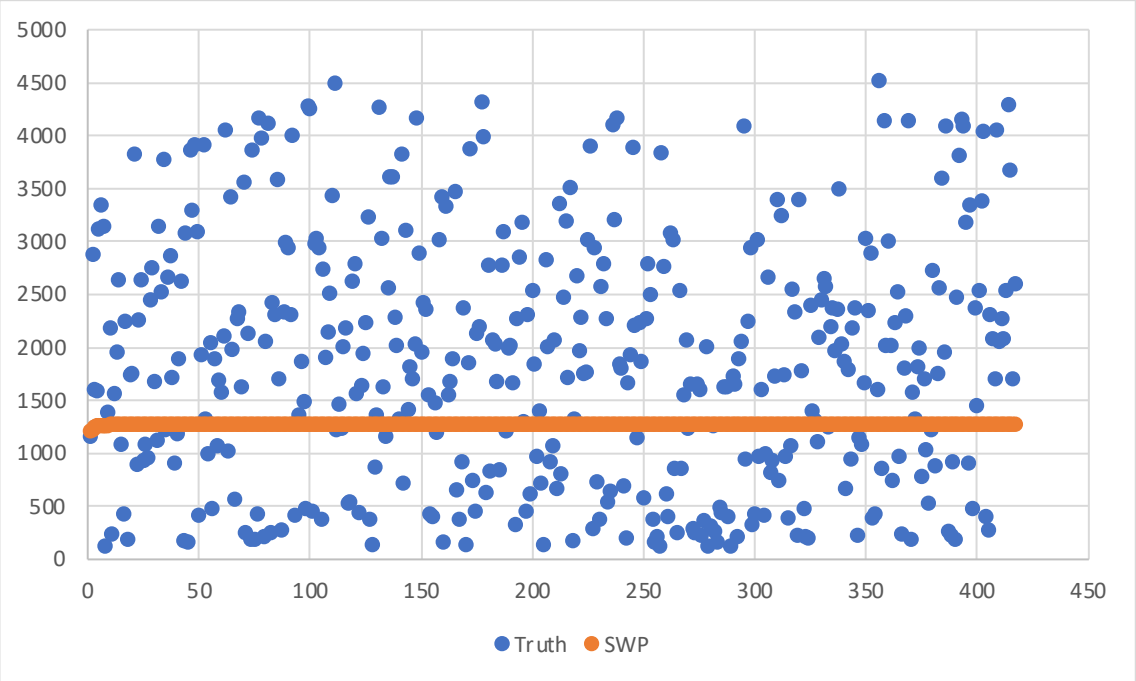
Appendix 9 Proportional Errors of Low Rate Missing Values



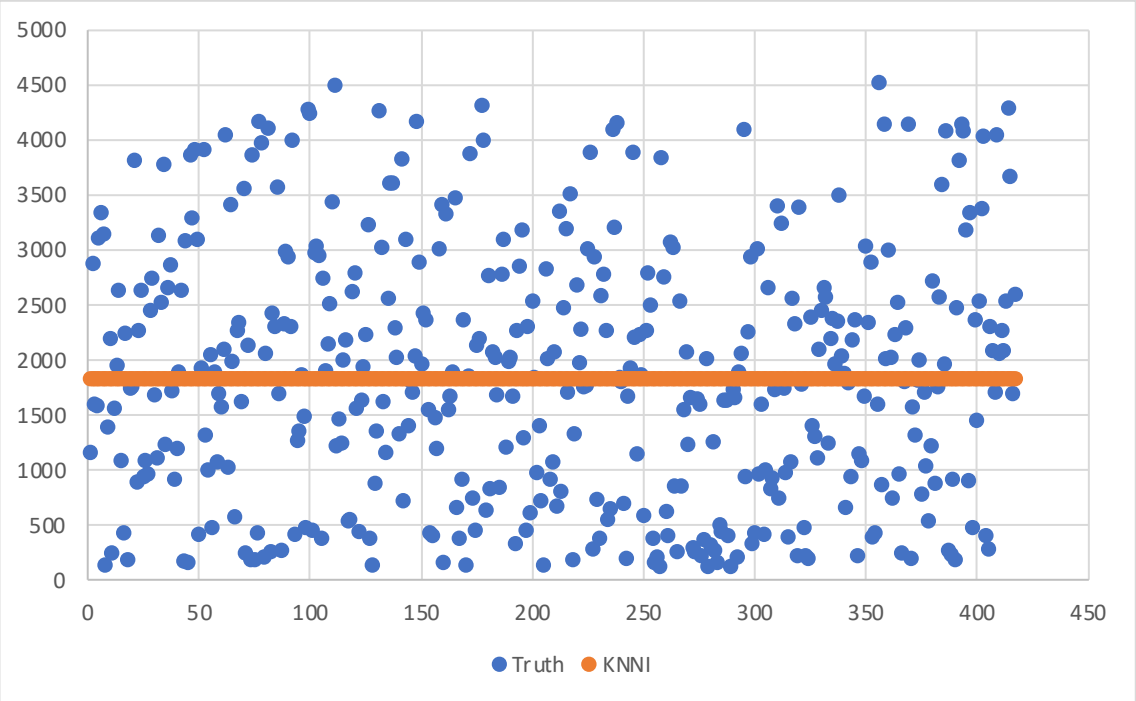
Appendix 10 Table of outputs of Low Rate Missing Value Imputation

	Iterative	KNNI	SWP
Median	0.188559	0.401931	0.493138
Highest Inaccuracy	5.080198	13.71914	9.146935
Mean	0.370839	1.260181	0.932888
Standard Deviation	0.555076	2.251246	1.405239
RMSE (Whole Dataset)	163.2530307	347.9121459	395.0611065
RMSE (Imputed values)	536.2076073	1153.002997	1294.525776
Time	0.030289933	0.304810967	0.140756167

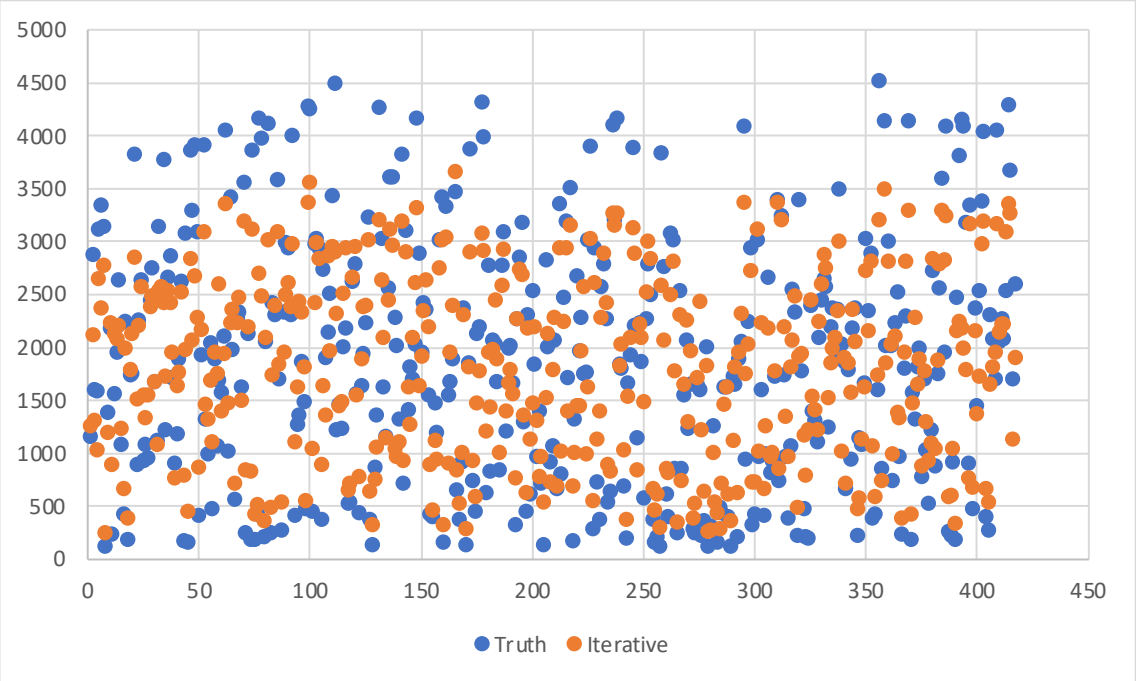
Appendix 11 SWP Output on Low Rate Missing Values Half



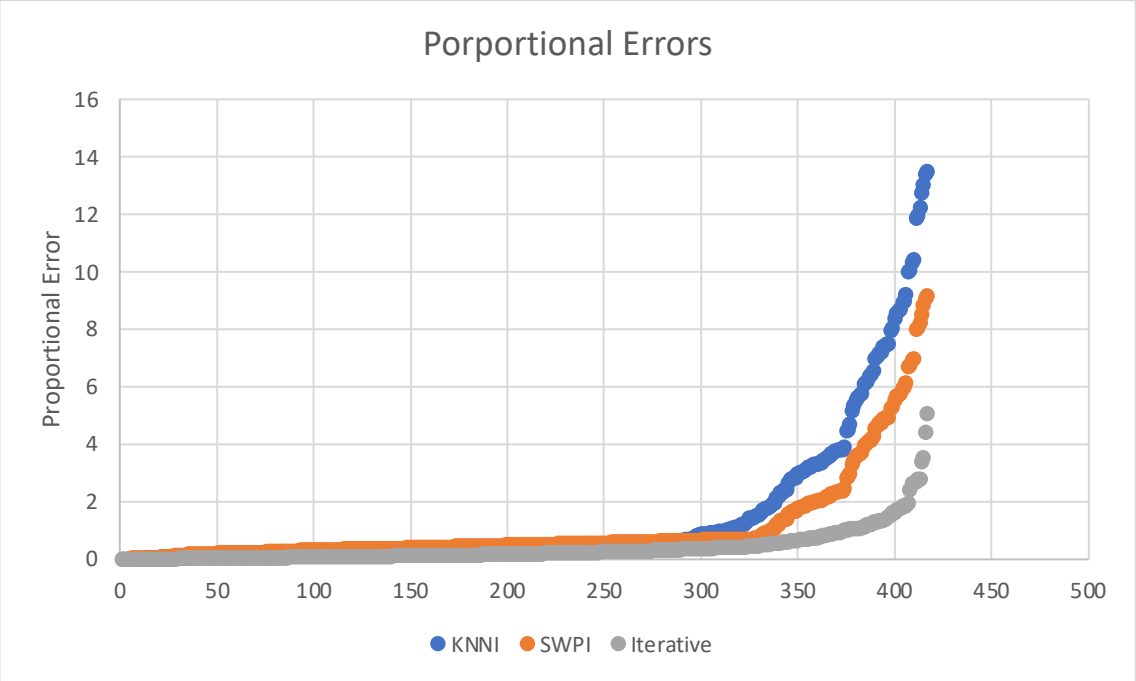
Appendix 12 KNNI Output on Low Rate Missing Values Half



Appendix 12 Iterative Imputer Output on Low Rate Missing Values Half



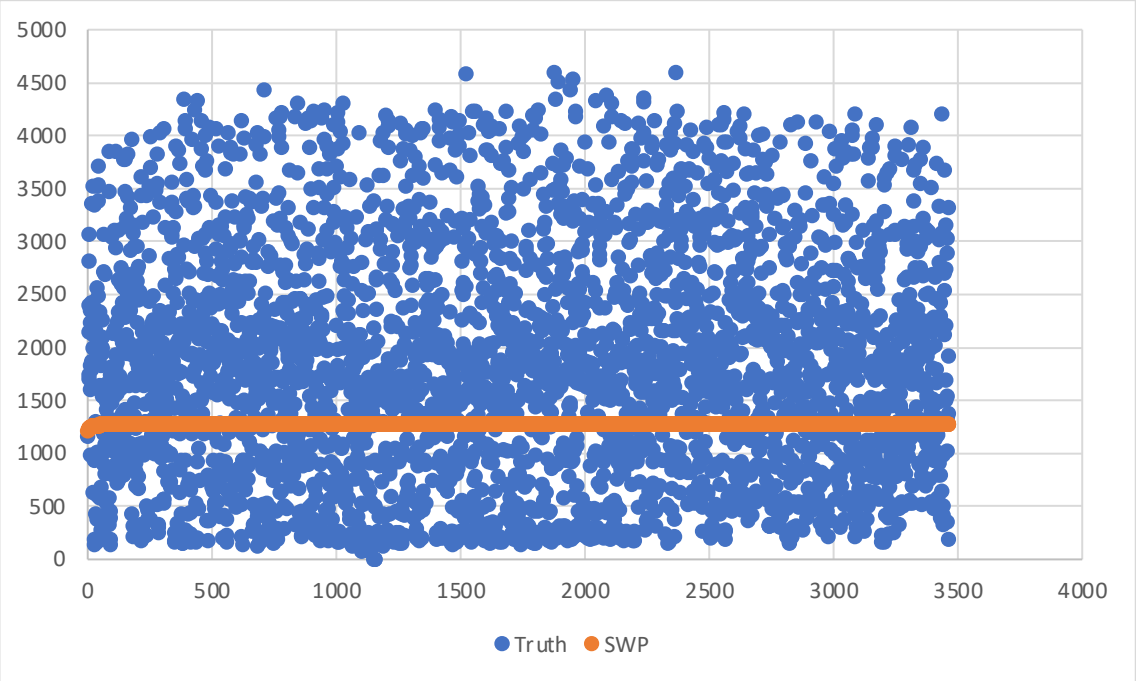
Appendix 13 Proportional Errors on Low Rate Missing Values Half



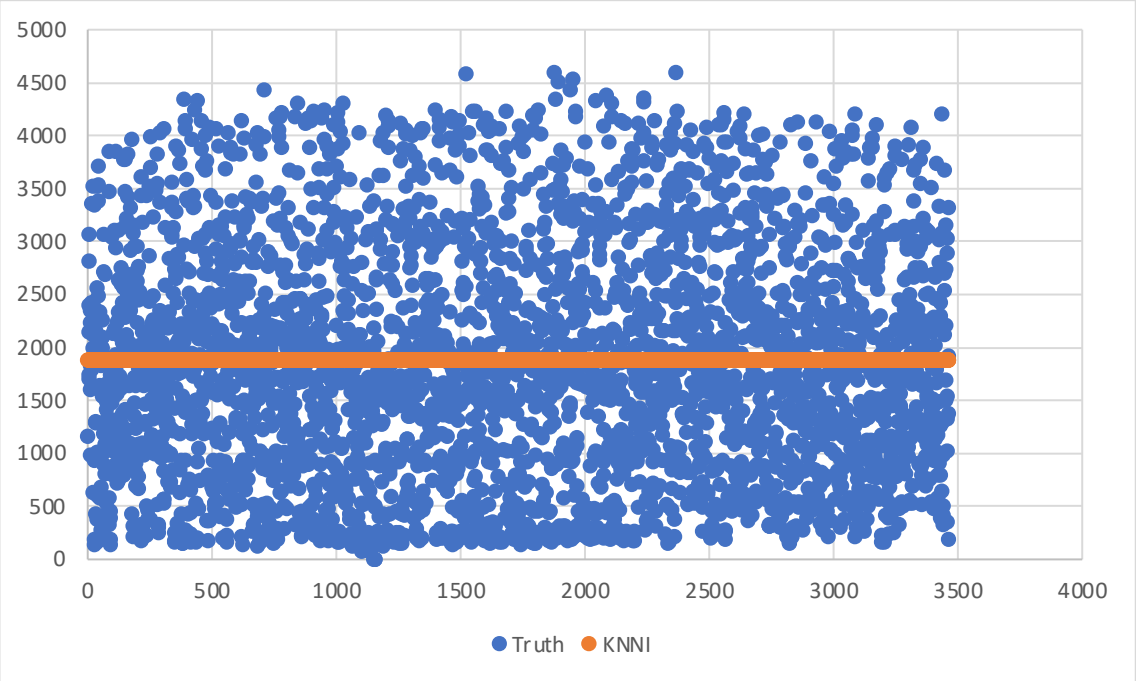
Appendix 14 Table Outputs on Low Rate Missing Values Half

	Iterative	KNNI	SWP
Median	0.19629331	0.41839596	0.50227844
Mean	0.39839243	1.44595621	1.07121083
Highest Inaccuracy	5.08019802	13.4939419	9.14693484
Standard Deviation	0.5993212	2.55400558	1.63706865
RMSE(Whole Dataset)	166.2168052	359.1921854	400.8757279
RMSE(Imputed Values)	538.6347591	1153.629341	1294.525776
Time	0.018973133	0.0859706	0.079884033

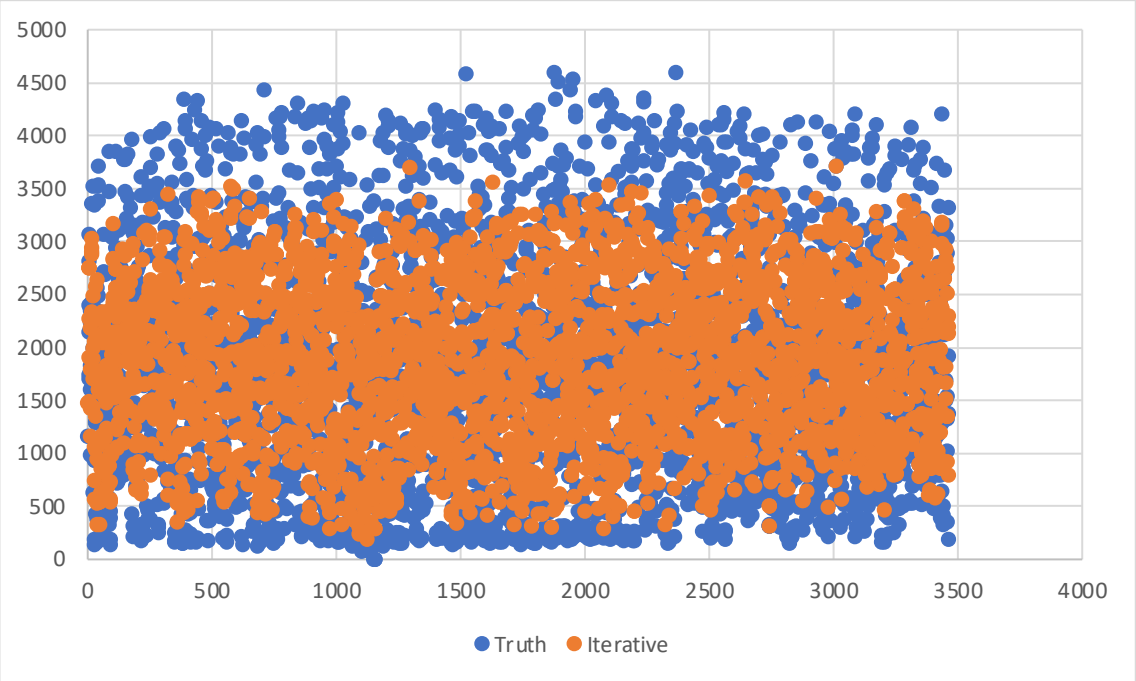
Appendix 15 SWP Output on High Rate Missing Values



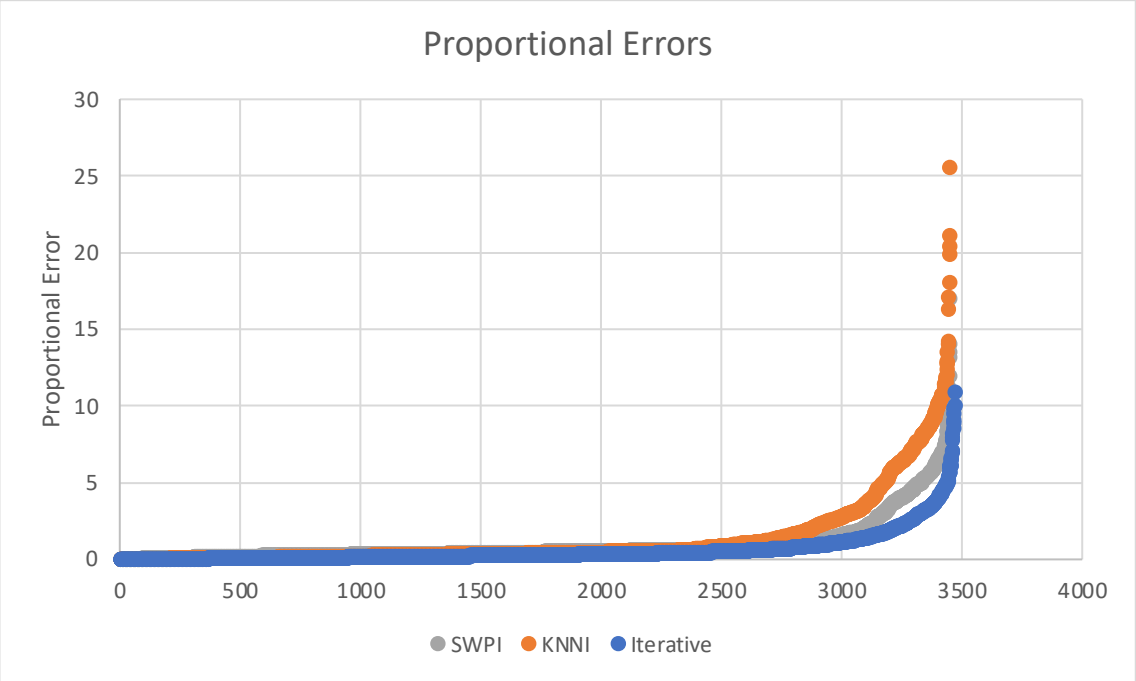
Appendix 16 KNNI Output on High Rate Missing Values



Appendix 17 Iterative Imputer Output On High Rate Missing Values



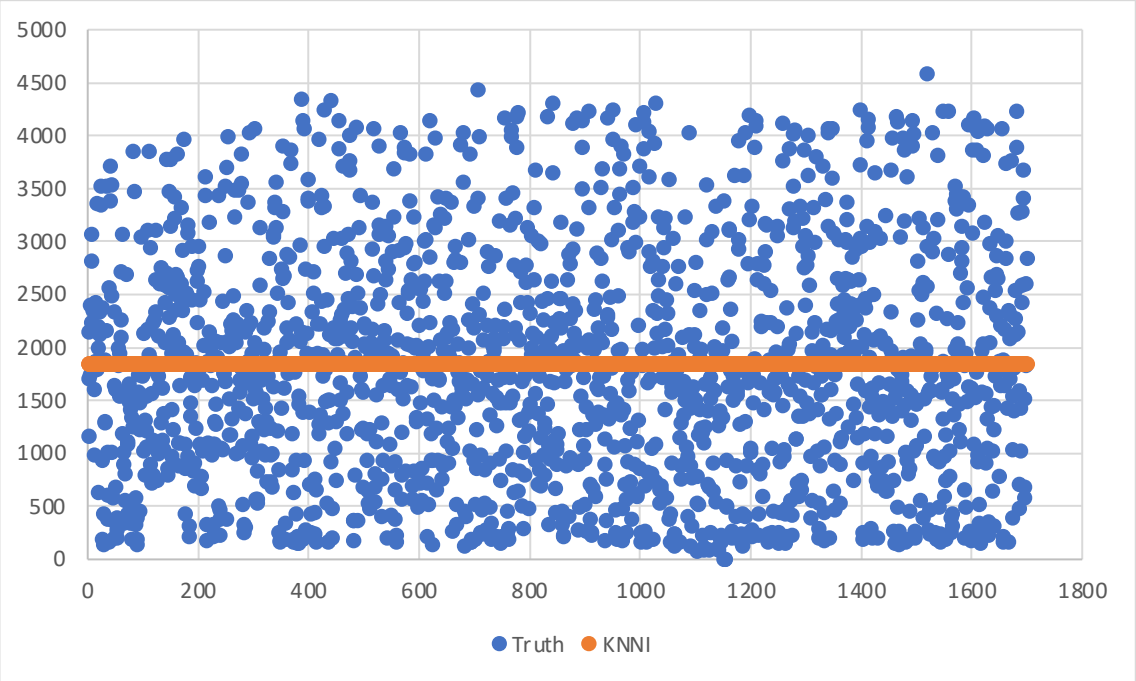
Appendix 18 Proportional Errors on High Rate Missing Values



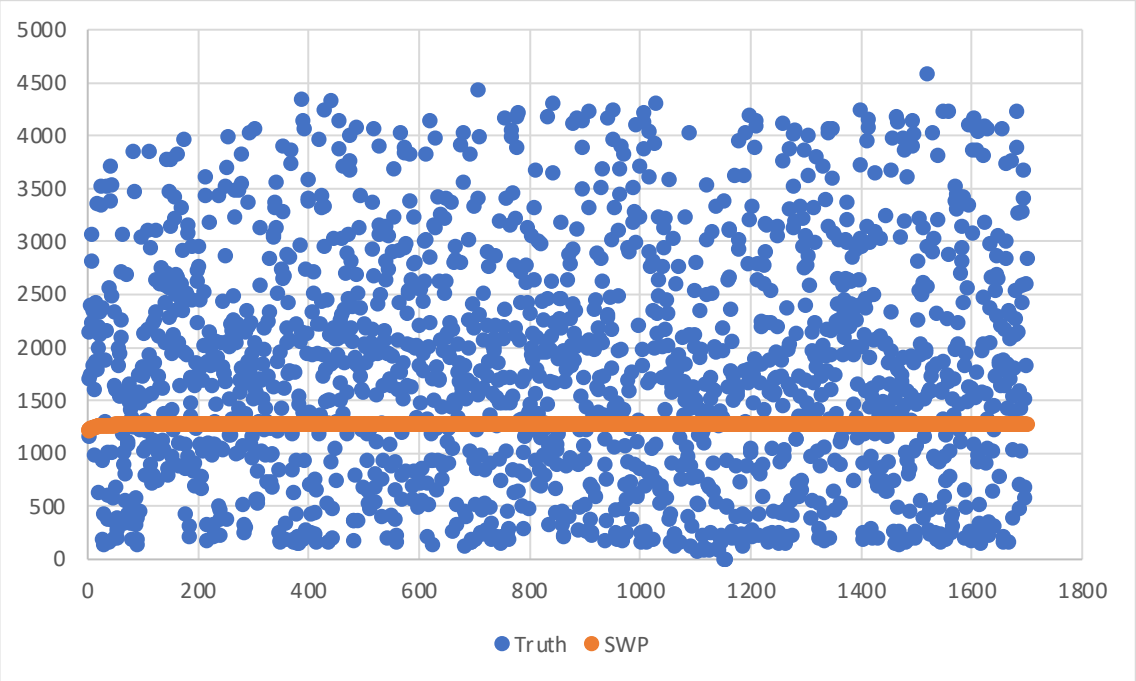
Appendix 19 Table of outputs of High Rate Missing Value Imputation

	Iterative	KNNI	SWP
Median	0.276411	0.401091	0.47299
Highest Inaccuracy	10.86875	25.5249	17.00702
Mean	0.607969	1.272054	0.911225
Standard Deviation	1.011215	2.336574	1.44611
RMSE(Whole Dataset)	446.5713191	684.0105731	762.9662638
RMSE(Imputed Values)	719.9413811	1096.373768	1212.165511
Time	0.039089267	1.2546521	1.305450767

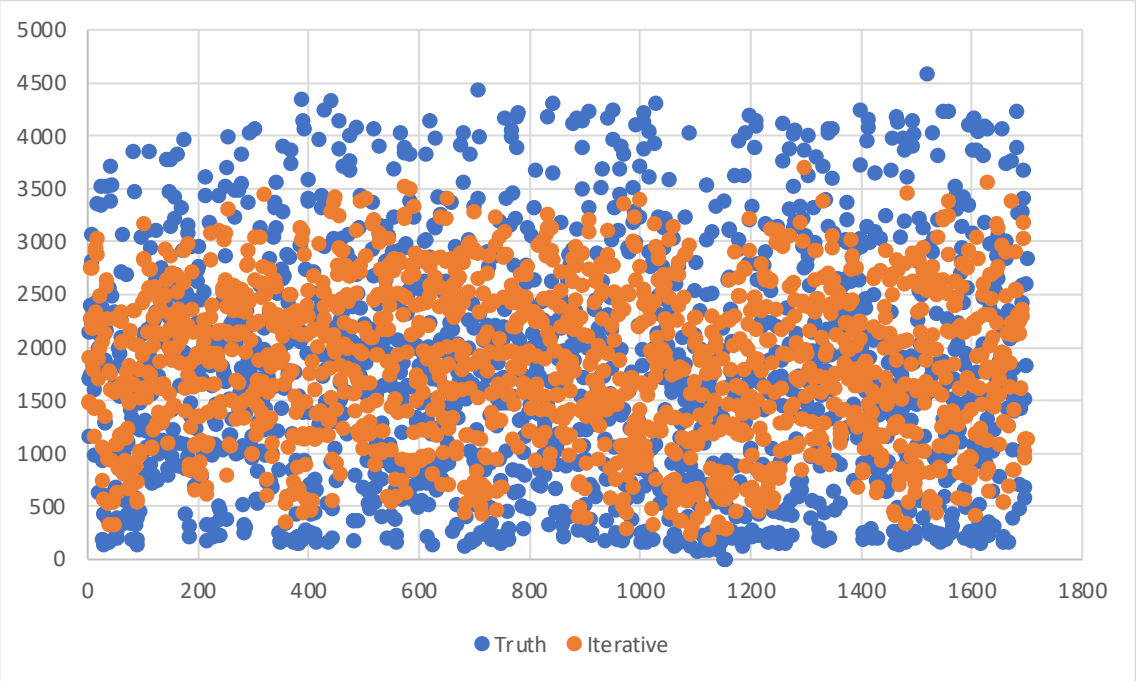
Appendix 20 KNNI Output of High Rate Missing Values Half



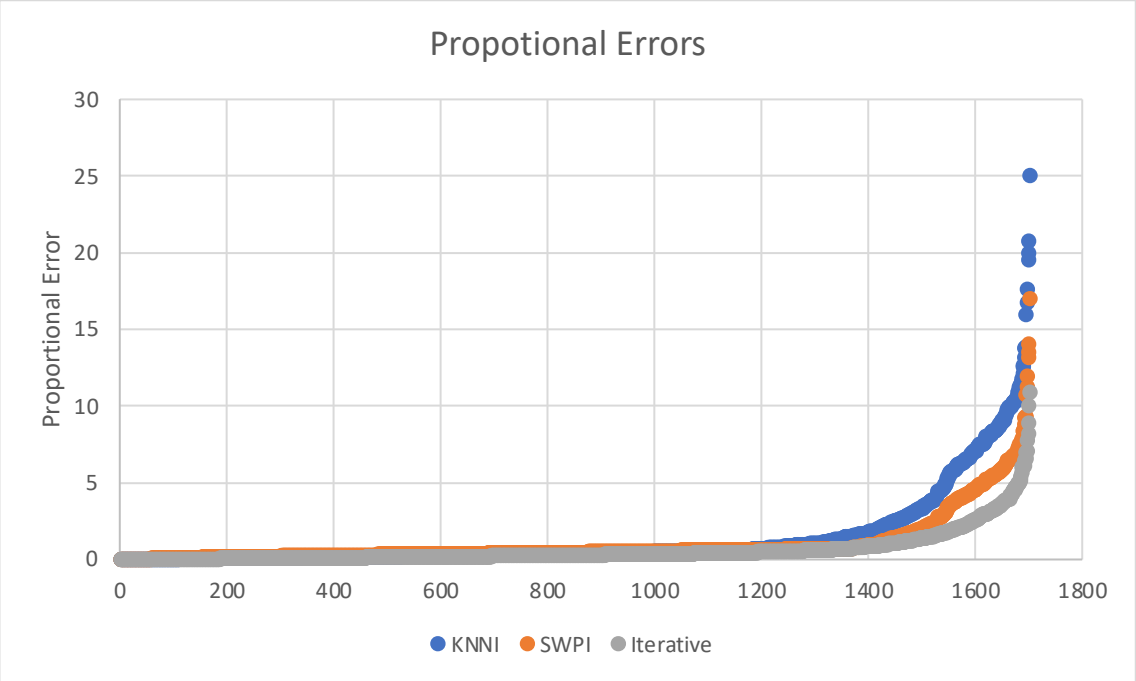
Appendix 21 SWP Output of High Rate Missing Values Half



Appendix 22 Iterative Imputer Output of High Rate Missing Values Half



Appendix 23 Proportional Errors of High Rate Missing Values Half



Appendix 24 Table of Outputs of High Rate Missing Value Imputation Half

	Iterative	KNNI	SWPI
Median	0.28902252	0.40153687	0.47333385
Mean	0.64166186	1.38267604	1.00511316
Highest Inaccuracy	10.86875	25.0120626	17.0070151
Standard Deviation	1.0808127	2.63431887	1.68704956
RMSE(Whole Dataset)	450.9698677	683.9836799	757.7037827
RMSE(Imputed Values)	722.7245472	1090.663762	1211.92558
Time	0.026760633	0.3091653	0.2911209

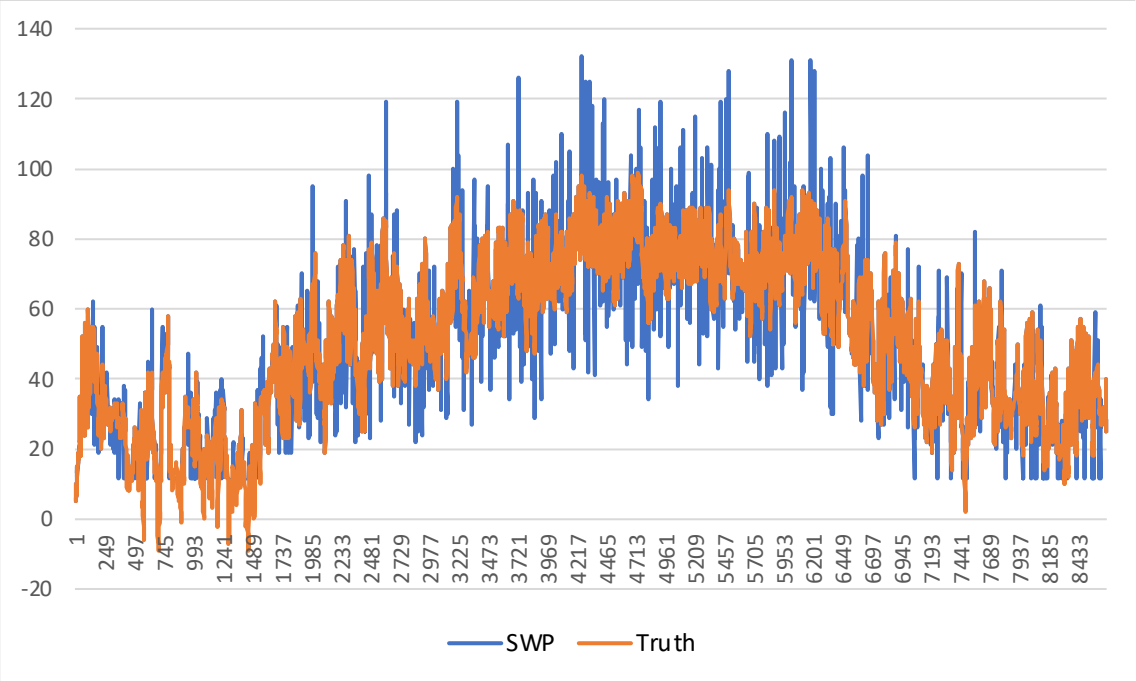
Appendix 25 Table of Outputs of Missing Value Imputation

	Metric	KNNI	SWP	Iterative Imputation
Full Dataset With Low Error rate	Median Inaccuracy	0.401931	0.493138	0.188559
	Highest Inaccuracy	13.71914	9.146935	5.080198
	Mean Inaccuracy	1.260181	0.932888	0.370839
	Standard Deviation	2.251246	1.405239	0.555076
	RMSE (Full)	347.9121	395.0611	163.253
	RMSE (Imputed)	1153.003	1294.526	536.2076
	Time	0.304811	0.140756	0.030289933
Half Dataset With Low Error rate	Median Inaccuracy	0.41839596	0.50227844	0.19629331
	Highest Inaccuracy	1.44595621	1.07121083	0.39839243
	Mean Inaccuracy	13.4939419	9.14693484	5.08019802
	Standard Deviation	2.55400558	1.63706865	0.5993212
	RMSE (Full)	1153.629	1294.526	538.6348
	RMSE (Imputed)	359.1922	400.8757	166.2168
	Time	0.085971	0.079884	0.018973133
Full Dataset With High Error rate	Median Inaccuracy	0.401091	0.47299	0.276411
	Highest Inaccuracy	25.5249	17.00702	10.86875
	Mean Inaccuracy	1.272054	0.911225	0.607969
	Standard Deviation	2.336574	1.44611	1.011215
	RMSE (Full)	684.0106	762.9663	446.5713
	RMSE (Imputed)	1096.374	1212.166	719.9414
	Time	1.254652	1.305451	0.039089267
Half Dataset With High Error rate	Median Inaccuracy	0.40153687	0.47333385	0.28902252
	Highest Inaccuracy	1.38267604	1.00511316	0.64166186
	Mean Inaccuracy	25.0120626	17.0070151	10.86875
	RMSE (Full)	683.9837	757.7038	450.9699
	RMSE (Imputed)	1090.664	1211.926	722.7245
	Standard Deviation	2.63431887	1.68704956	1.0808127
	Time	0.309165	0.291121	0.026760633

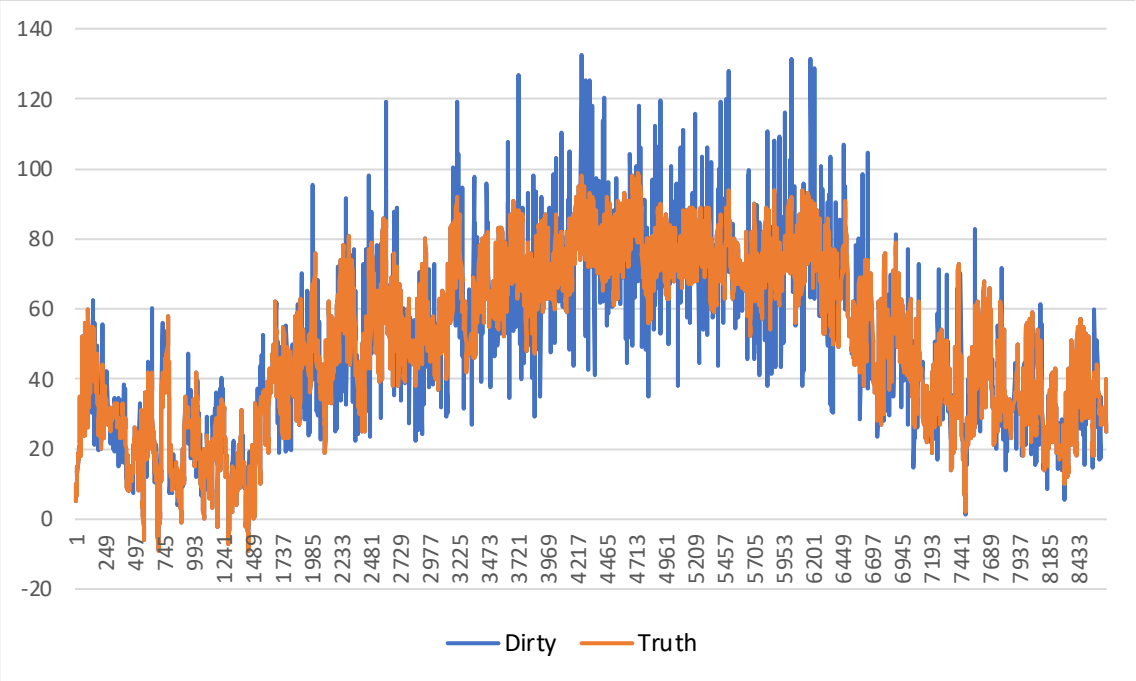
Appendix 26 Dataset Scaling of Missing Value Imputation

	Iterative	KNNI	SWP
Increase of Median proportional	1.465913	0.99791	0.959143
Increase of Mean proportional	1.639442	1.009422	0.976779
Increase of Max proportional	2.139434	1.860532	1.859314
RMSE(Imputed)	1.342654	0.950885	0.936378
RMSE(Whole)	2.735455	1.966044	1.931261
	Iterative	KNNI	SWP
Increase on Low Error Rate	1.596464474	3.545525641	1.762006258
Increase on High Error Rate	1.460700357	4.058191848	4.484222076

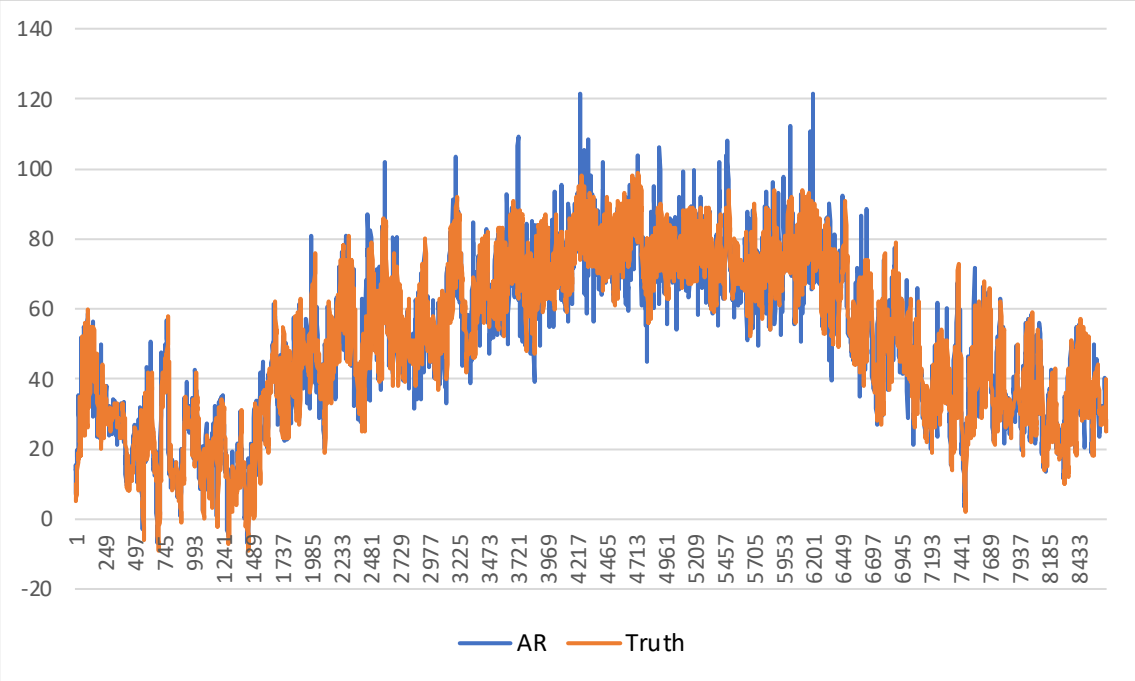
Appendix 27 SWP Output on Low Rate Outliers



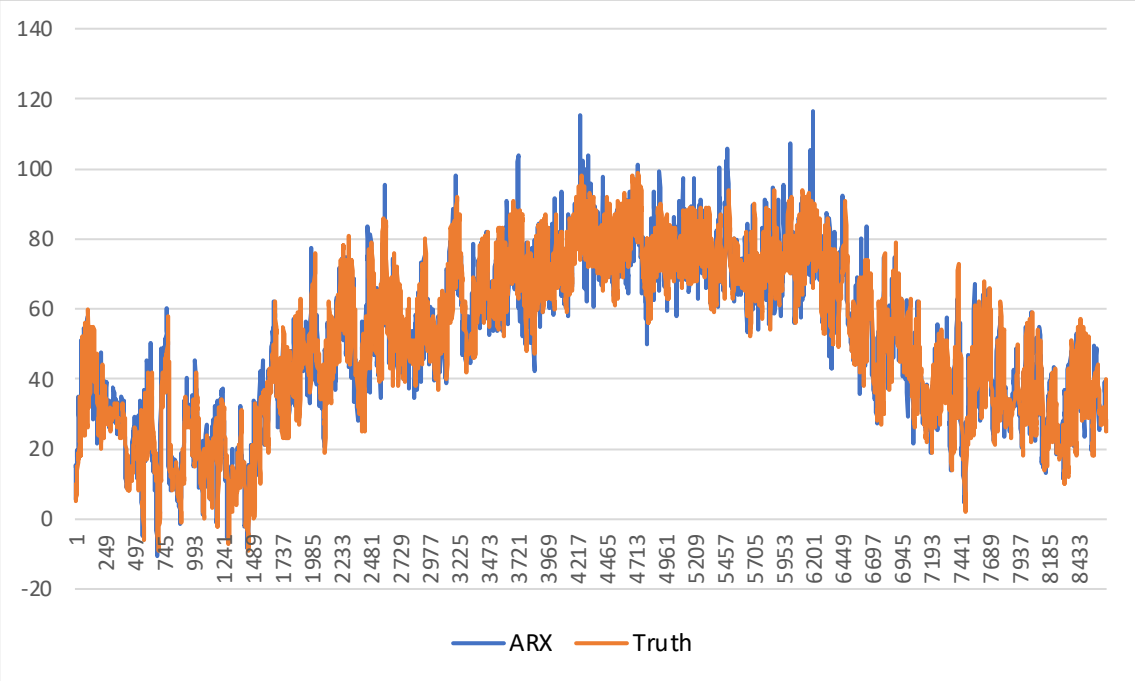
Appendix 28 Low Rate Dirty Data Outliers



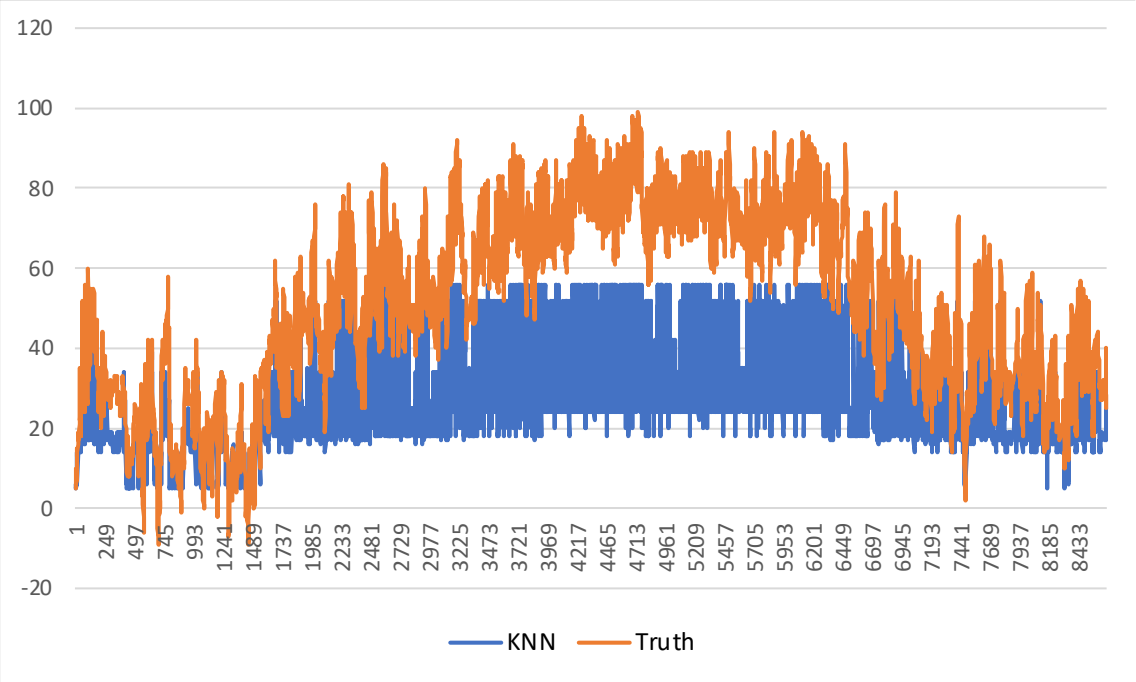
Appendix 29 AR Output on Low Rate Outliers



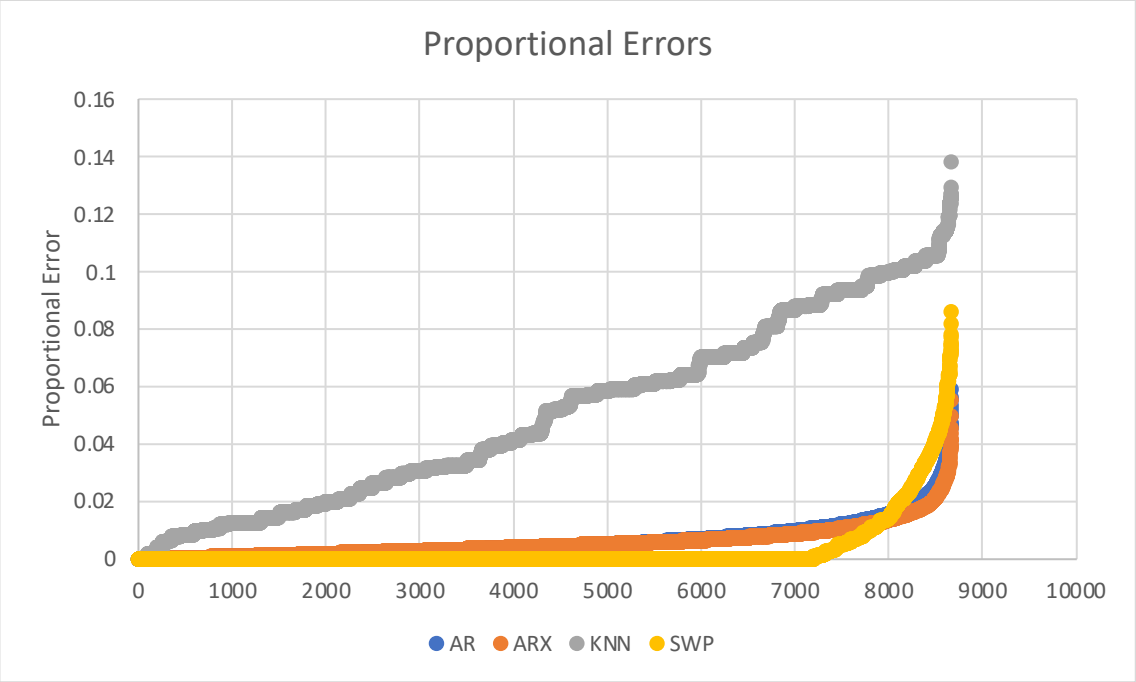
Appendix 30 ARX Output on Low Rate Outliers



Appendix 31 KNN Output on Low Rate Outliers



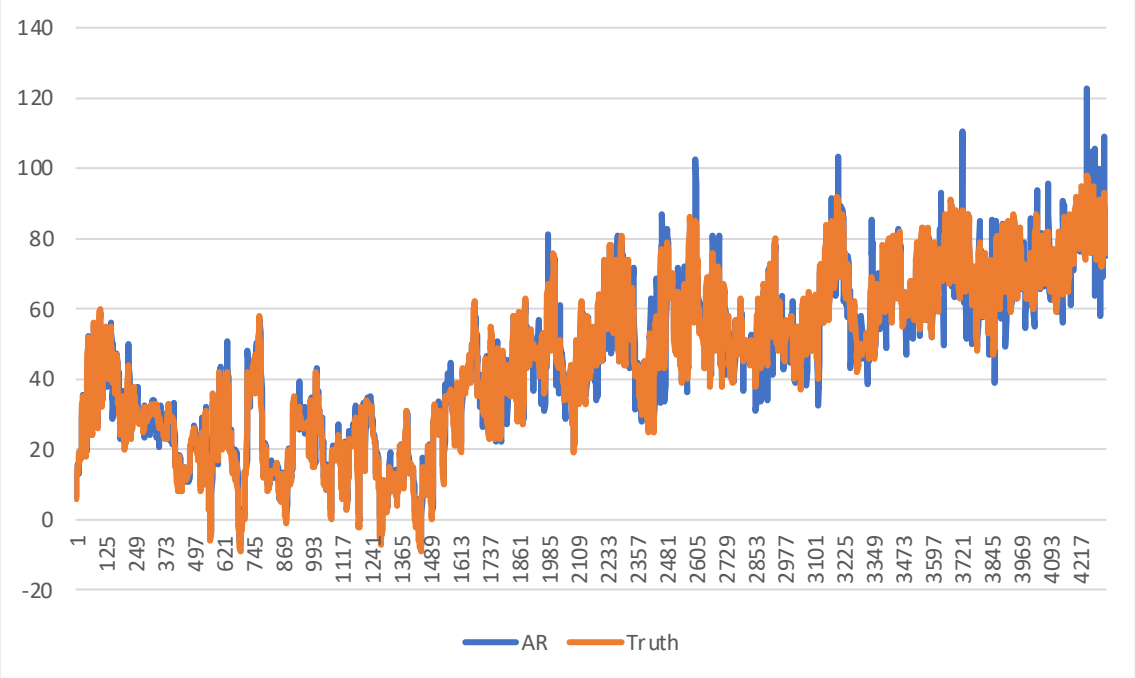
Appendix 32 Proportional Errors of Low Rate Outliers



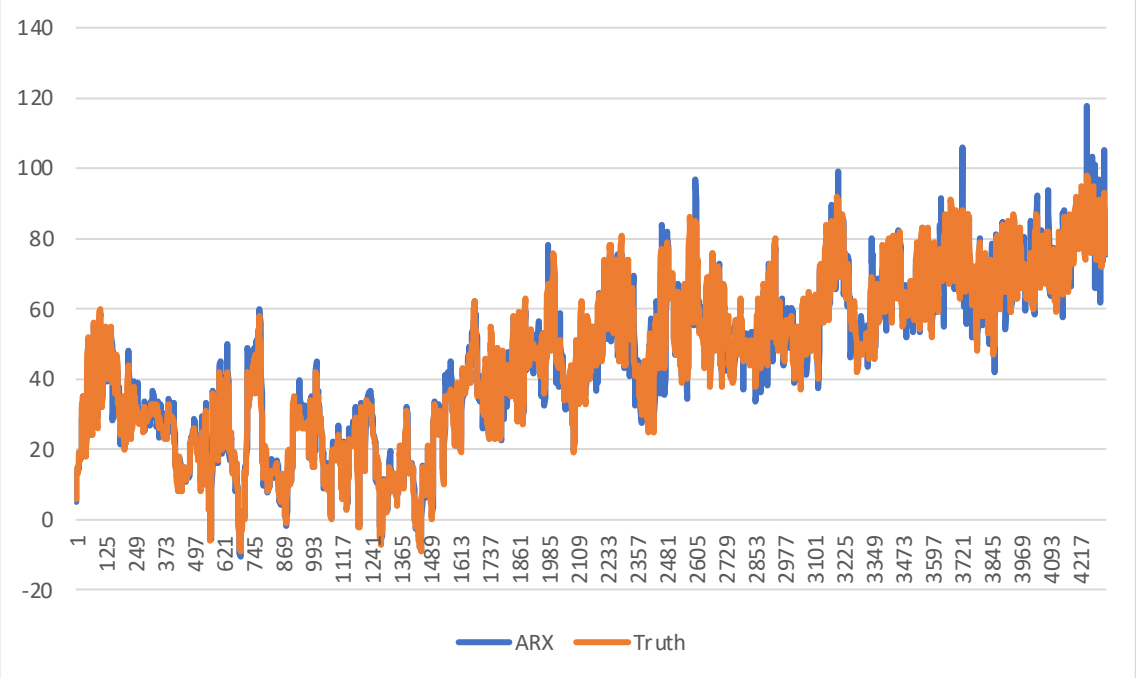
Appendix 33 Outputs of Low Rate Outlier Repair

	AR	ARX	KNN	SWP
Max	0.059191	0.055479	0.138323	0.086227
Median	0.004294	0.00449	0.049051	0
Mean	0.006203	0.00578	0.050289	0.003187
Standard Deviation	0.006363	0.00531	0.031716	0.009785
RMSE	4.349757	3.912273	28.34284	5.223187
Mean Time	0.018515	0.016746	0.210156	4.012226

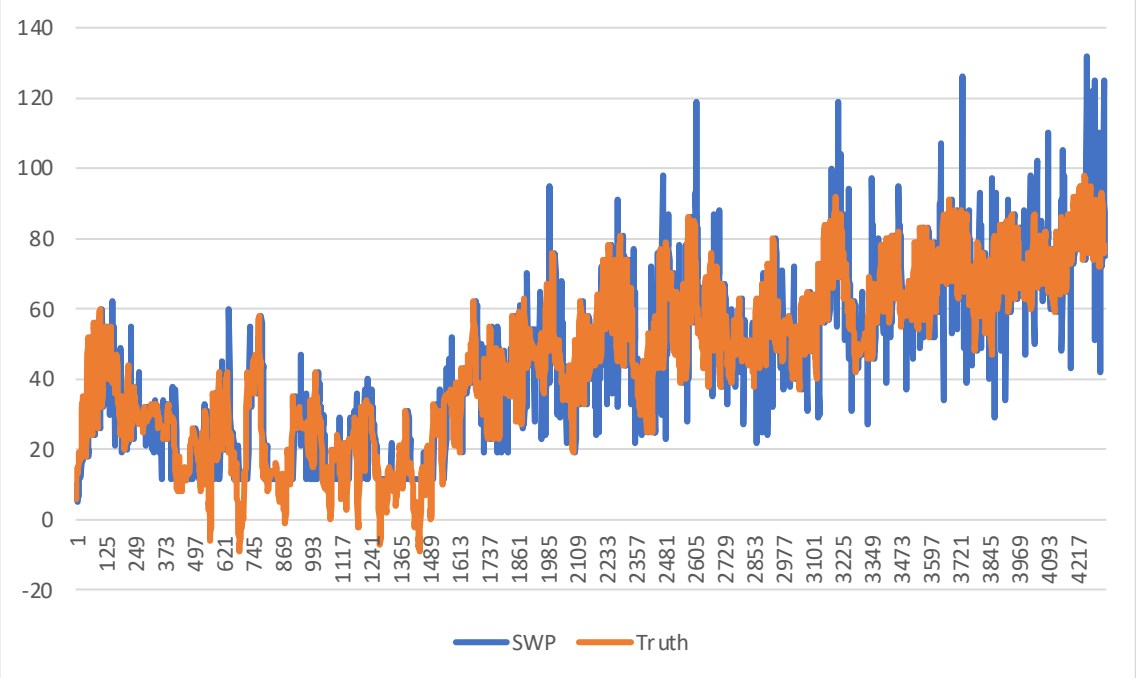
Appendix 34 AR Outputs of Low Rate Outlier Repair Half



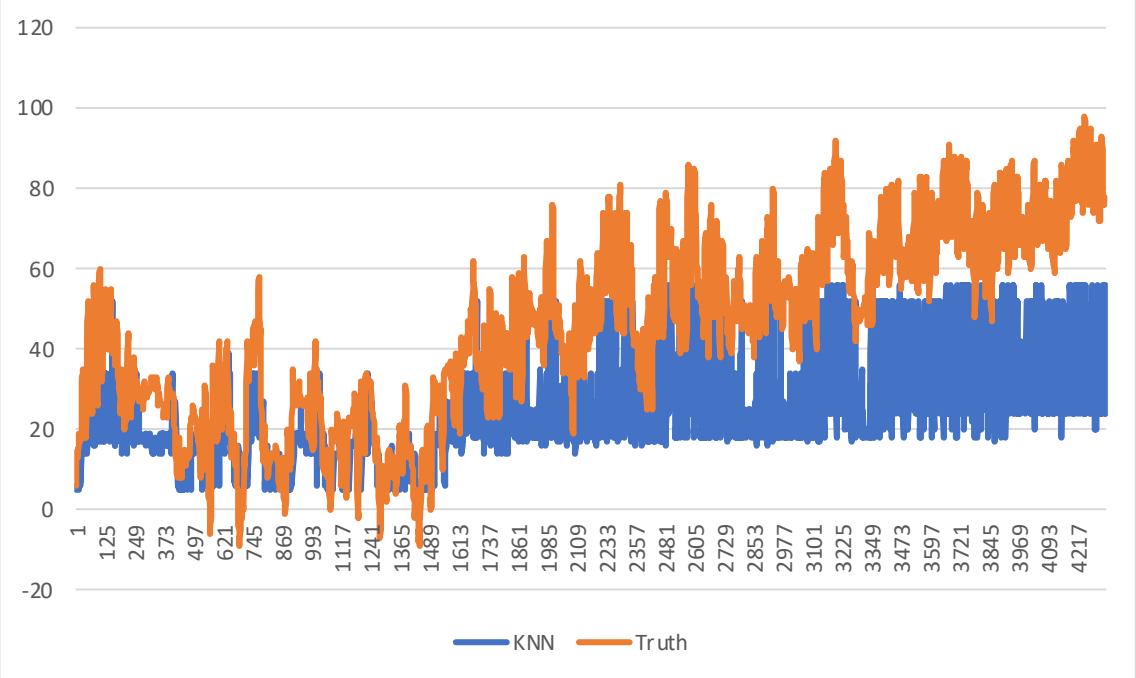
Appendix 35 ARX Outputs of Low Rate Outlier Repair Half



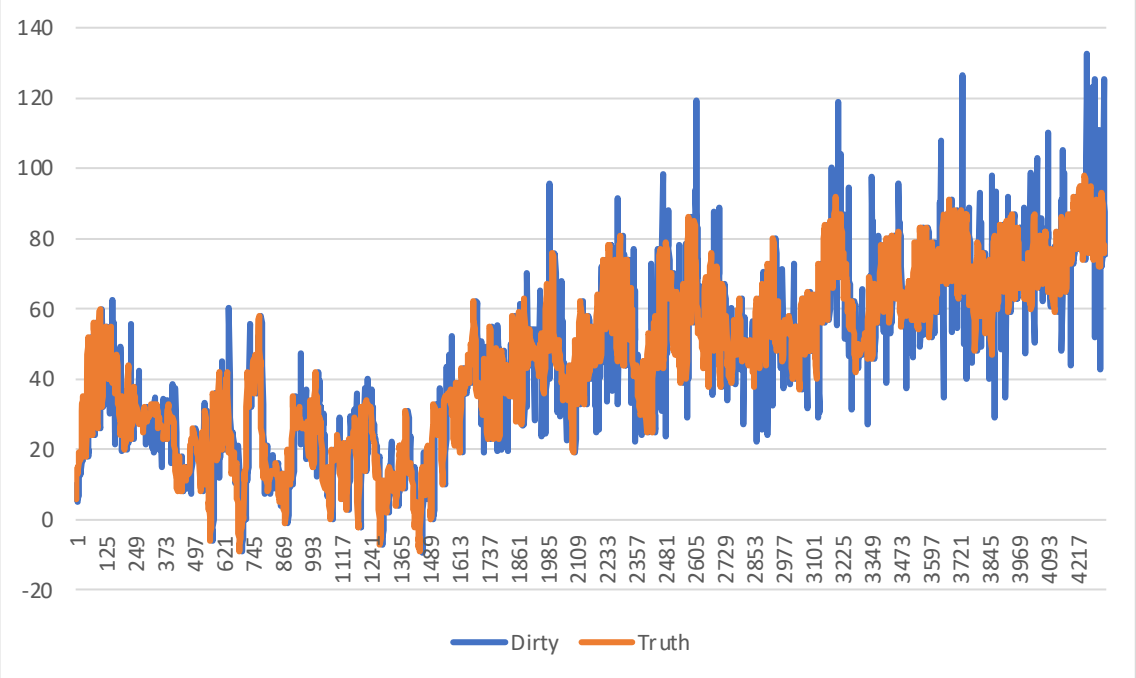
Appendix 37 SWP Outputs of Low Rate Outlier Repair Half



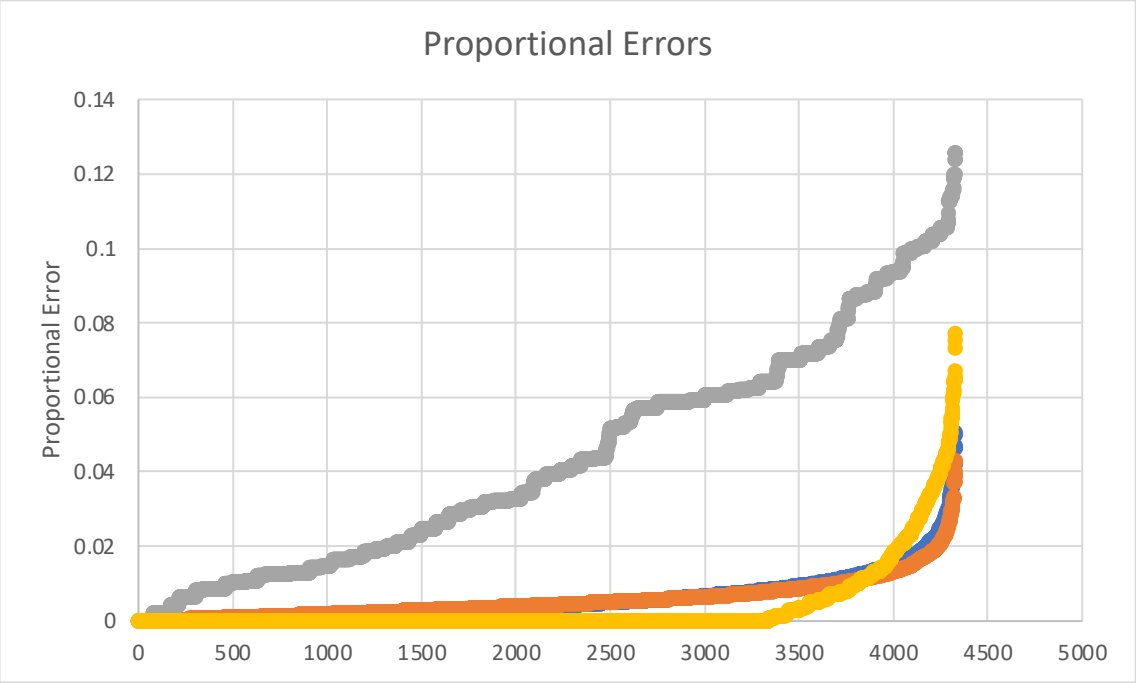
Appendix 38 KNN Outputs of Low Rate Outlier Repair Half



Appendix 39 AR Outputs of Low Rate Outlier Repair Half



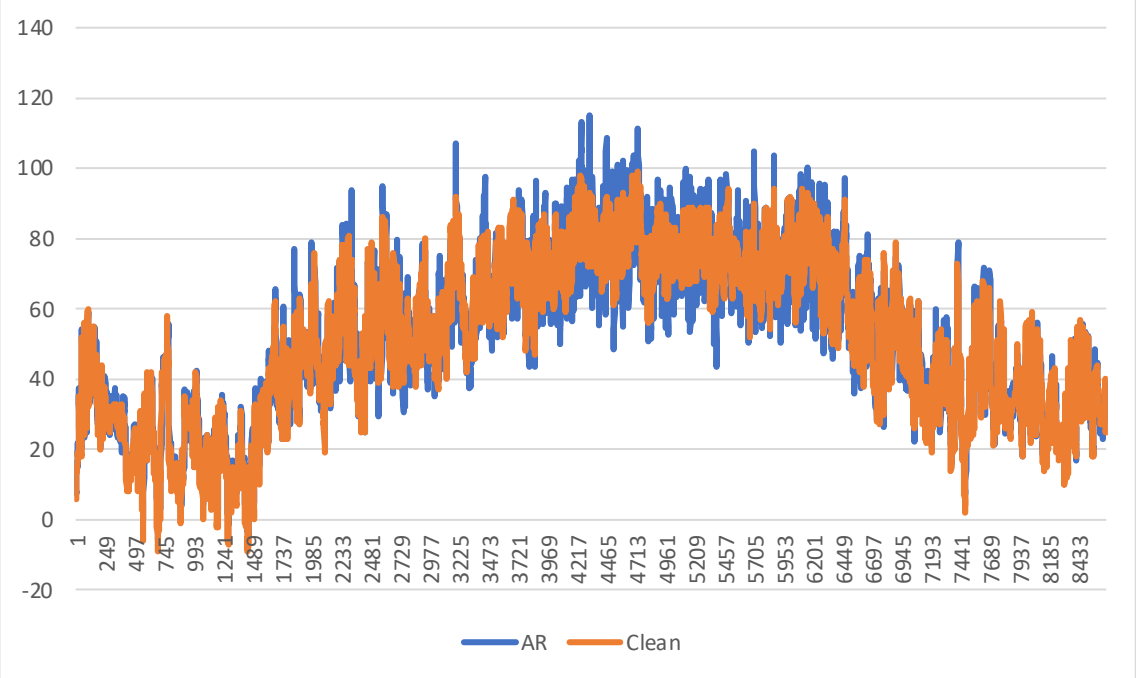
Appendix 40 Proportional Errors Of Low Rate Outlier Repair Half



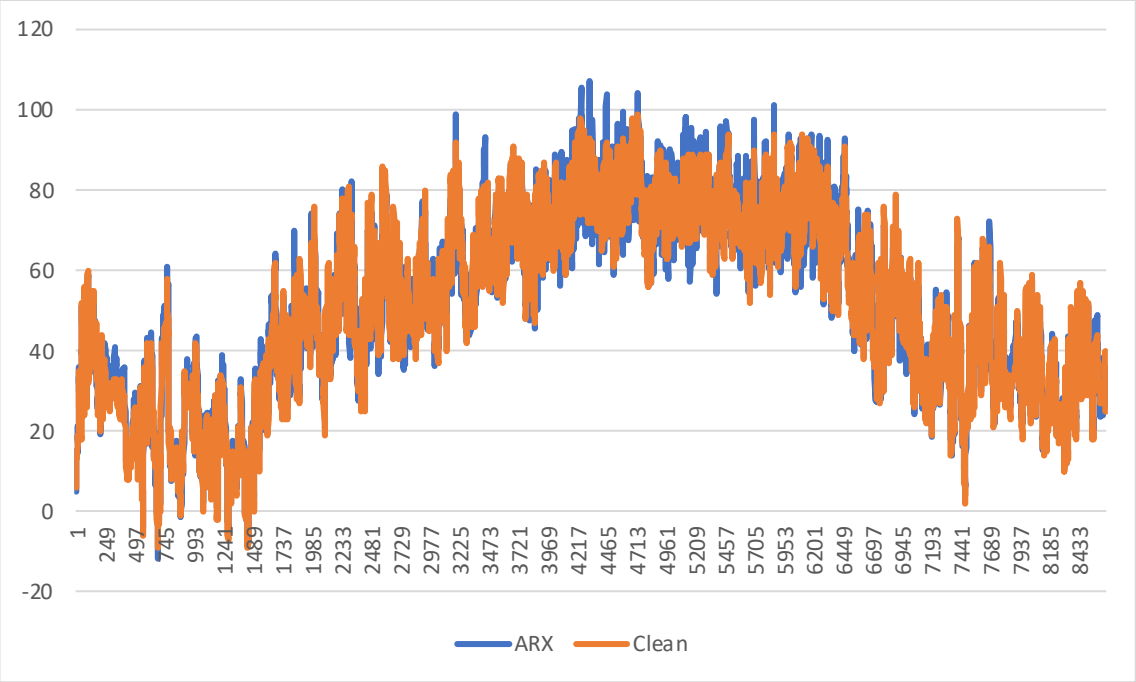
Appendix 41 Table Of Outputs of Low Rate Outlier Repair Half

	AR	ARX	KNN	SWP
Max	0.050431	0.042936	0.125759	0.077111
Median	0.003949	0.004226	0.03963	0
Mean	0.005855	0.005563	0.043466	0.003633
Standard Deviation	0.006107	0.00516	0.029921	0.009629
RMSE	4.354195	3.905678	27.75466	5.209444
Time	0.011954	0.020929	0.113882	1.801031

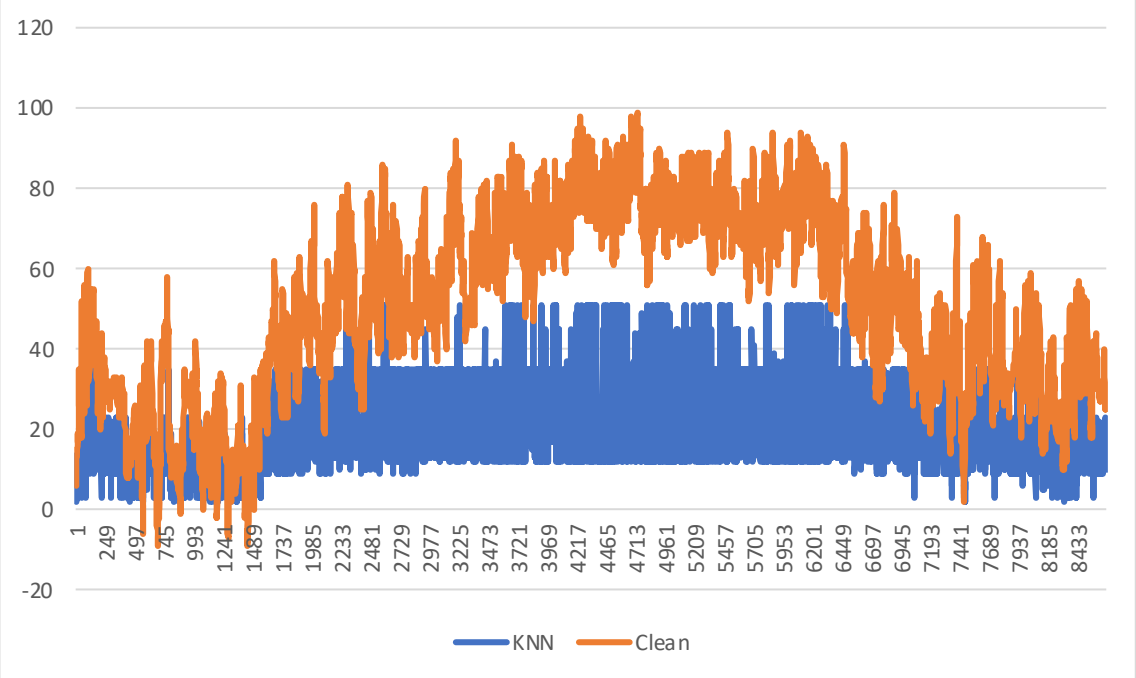
Appendix 42 AR Output Of High Rate Outlier Repair



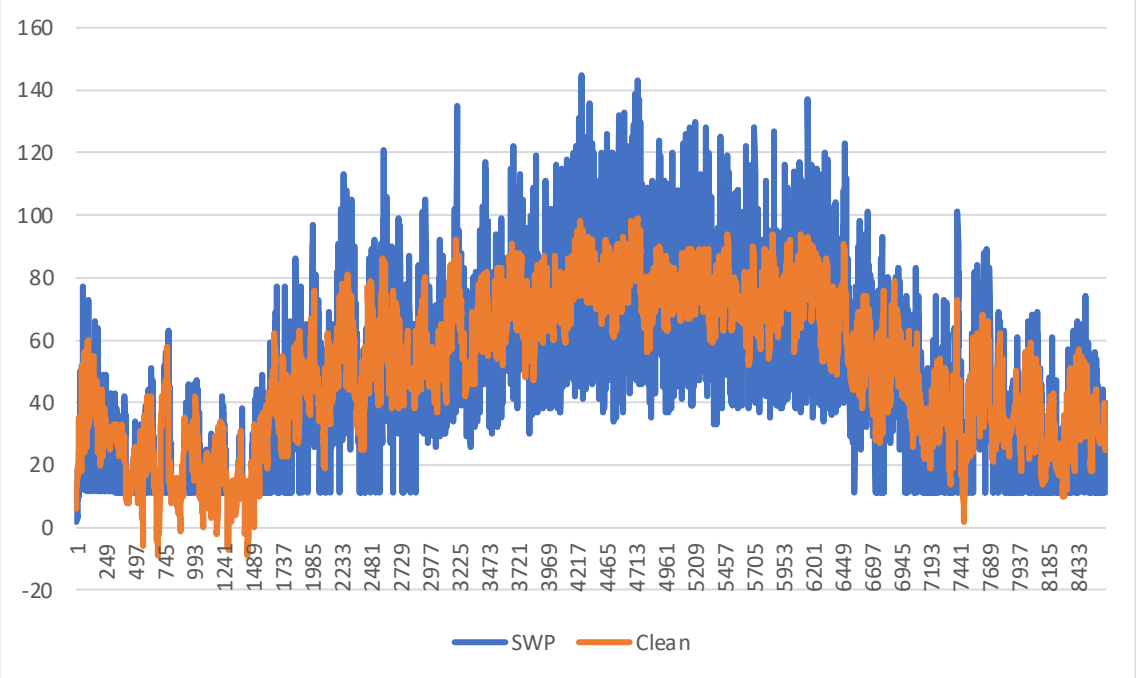
Appendix 43 ARX Output Of High Rate Outlier Repair



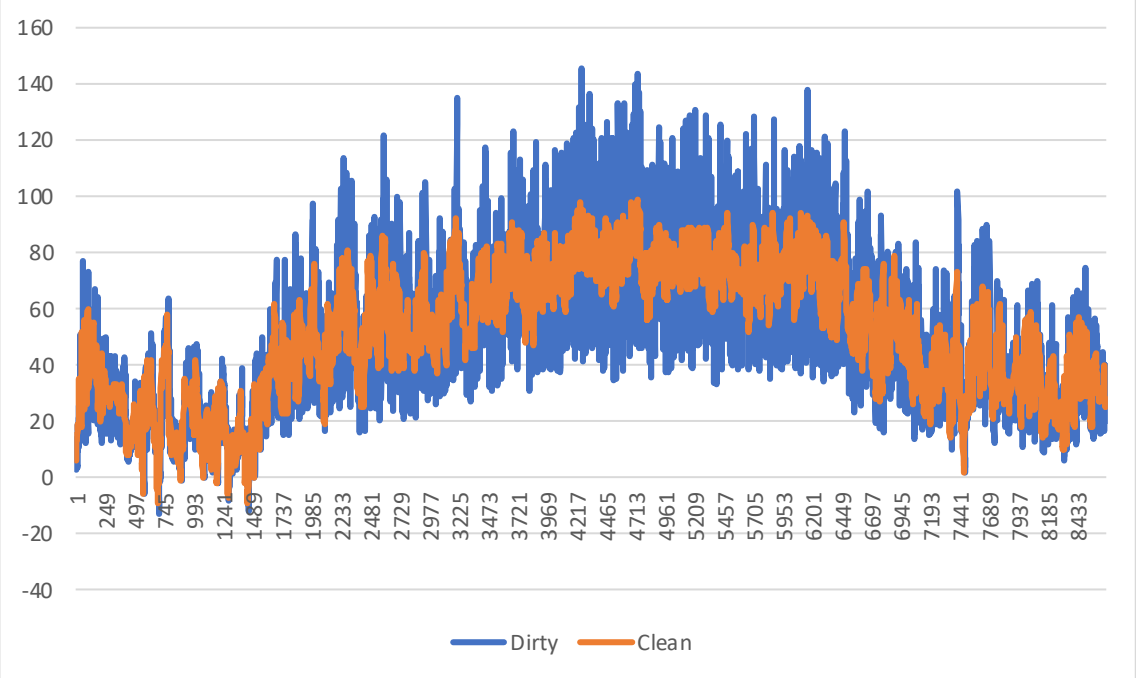
Appendix 44 KNN Output Of High Rate Outlier Repair



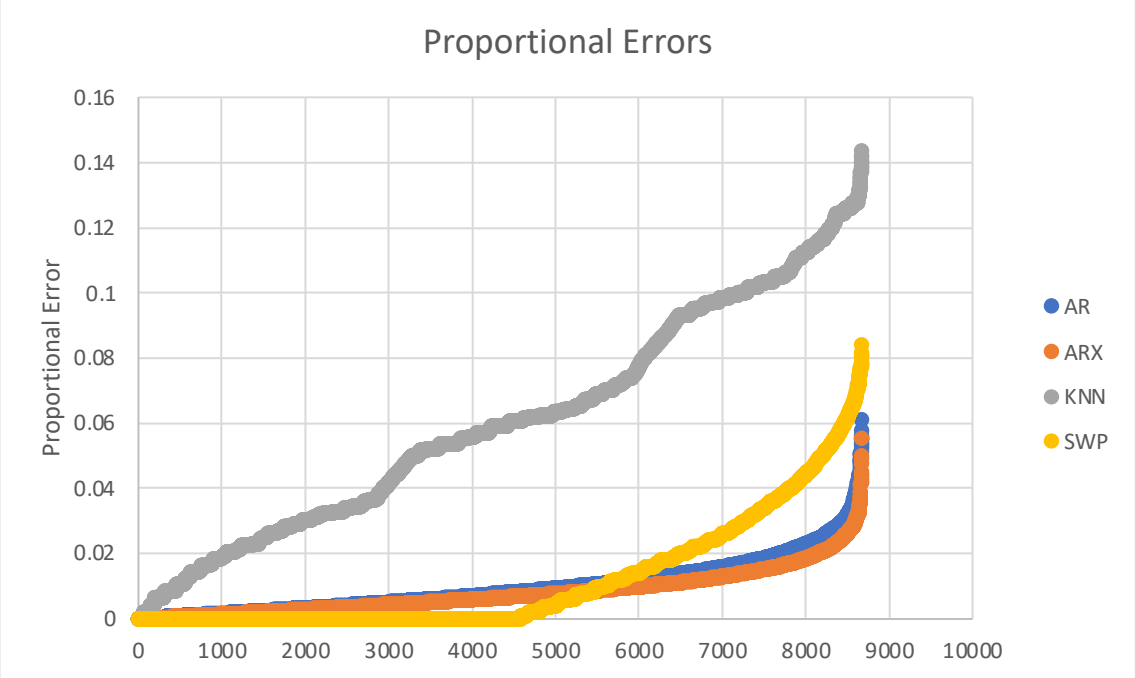
Appendix 45 SWP Output Of High Rate Outlier Repair



Appendix 46 Dirty Data For High Rate Outlier Repair



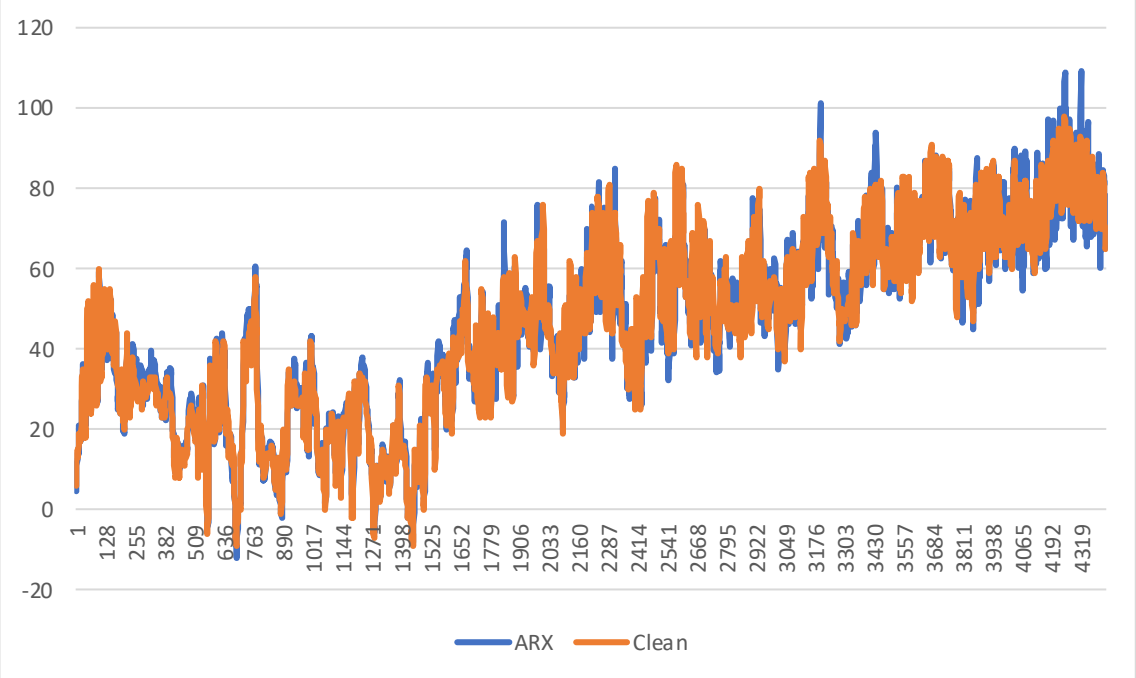
Appendix 47 Proportional Errors Of High Rate Outlier Repair



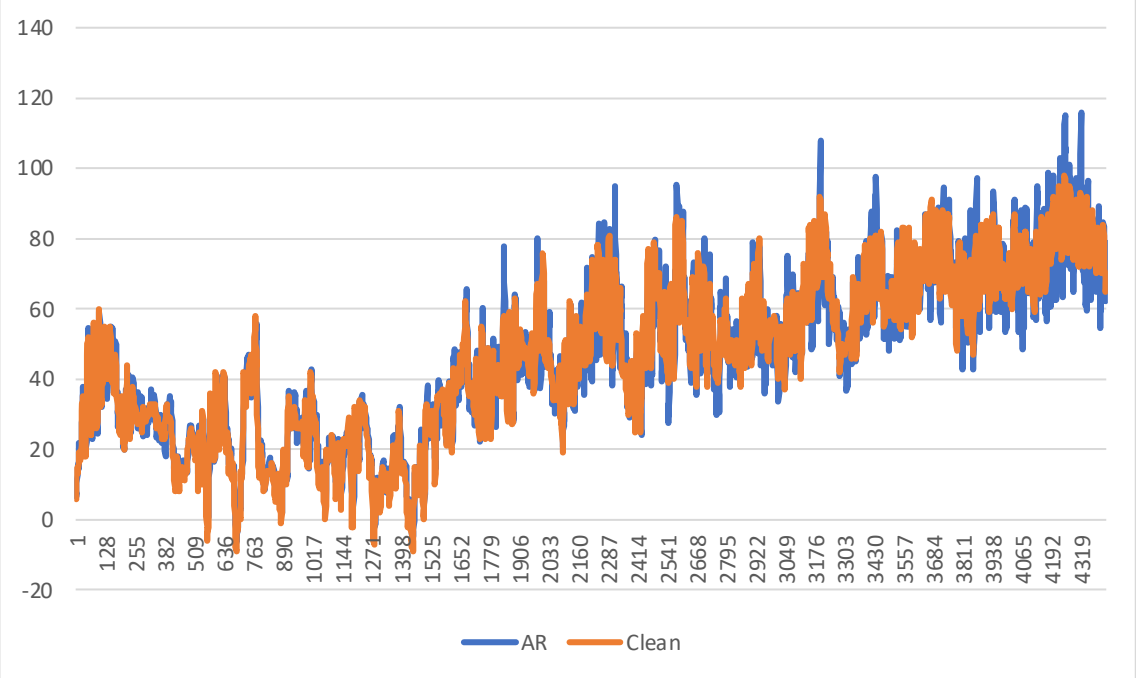
Appendix 48 Table of Outputs Of High Rate Outlier Repair

	AR	ARX	KNN	SWP
Max	0.060958	0.055479	0.143462	0.084279
Median	0.00775	0.006468	0.058972	0
Mean	0.009759	0.008035	0.060553	0.0118
Standard Deviation	0.008203	0.006609	0.034055	0.017574
RMSE	6.282781	5.21699	33.27731	10.33632
Mean Time	0.018711	0.024819	0.212992	4.284844

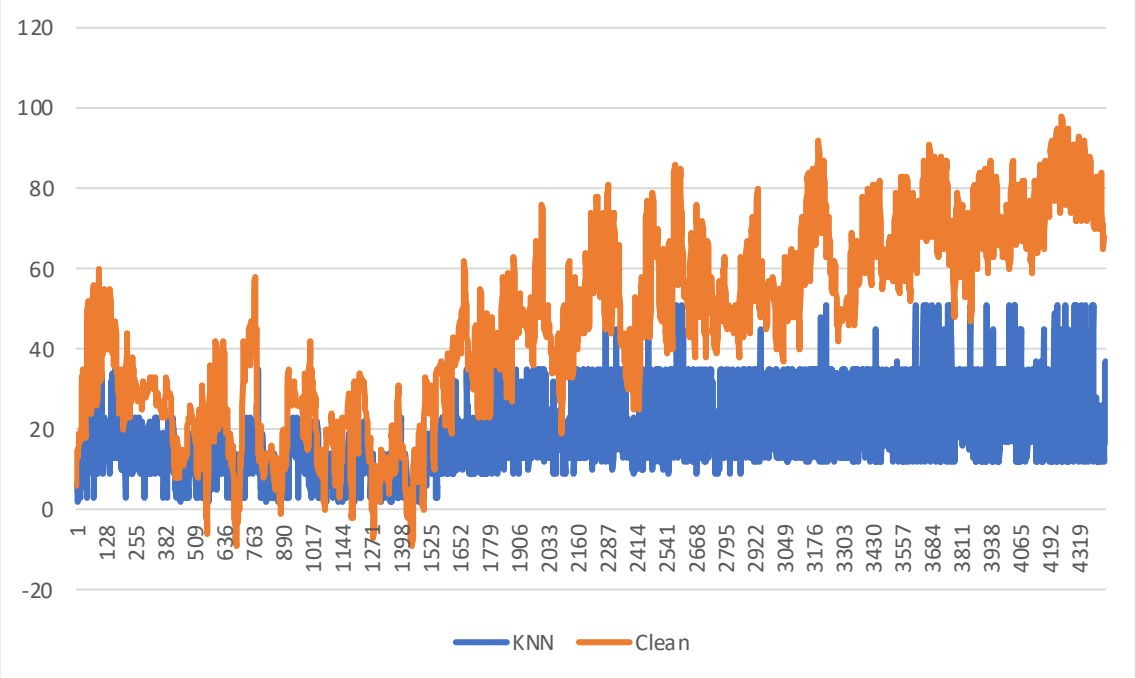
Appendix 49 ARX Output Of High Rate Outlier Repair Half



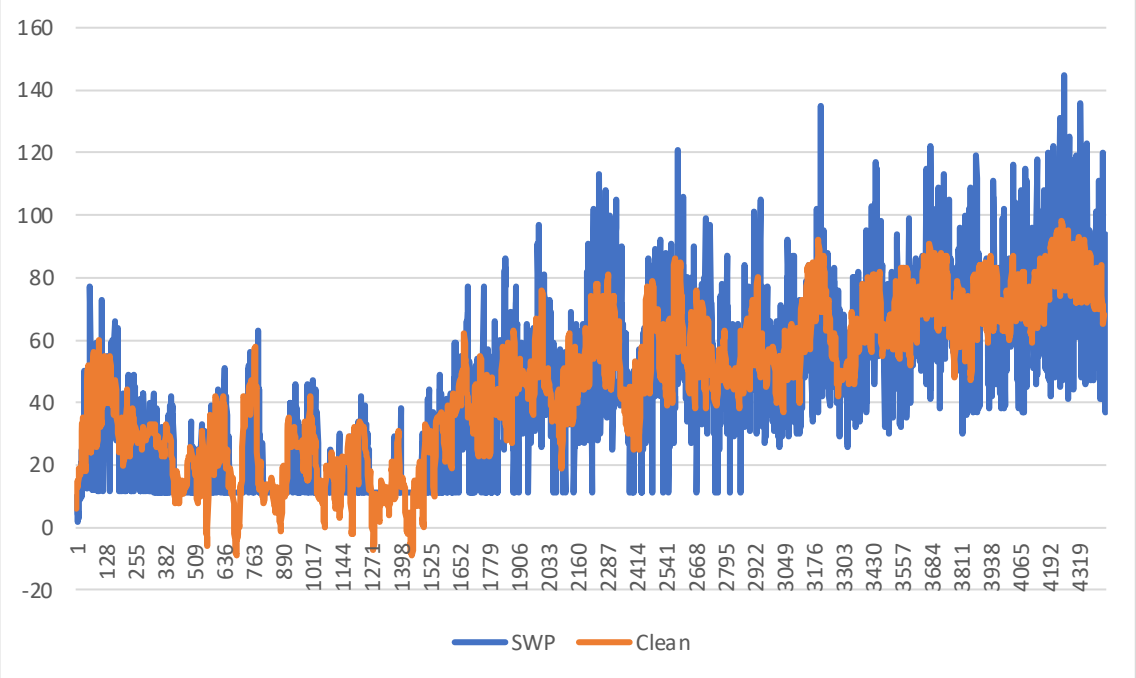
Appendix 50 AR Output Of High Rate Outlier Repair Half



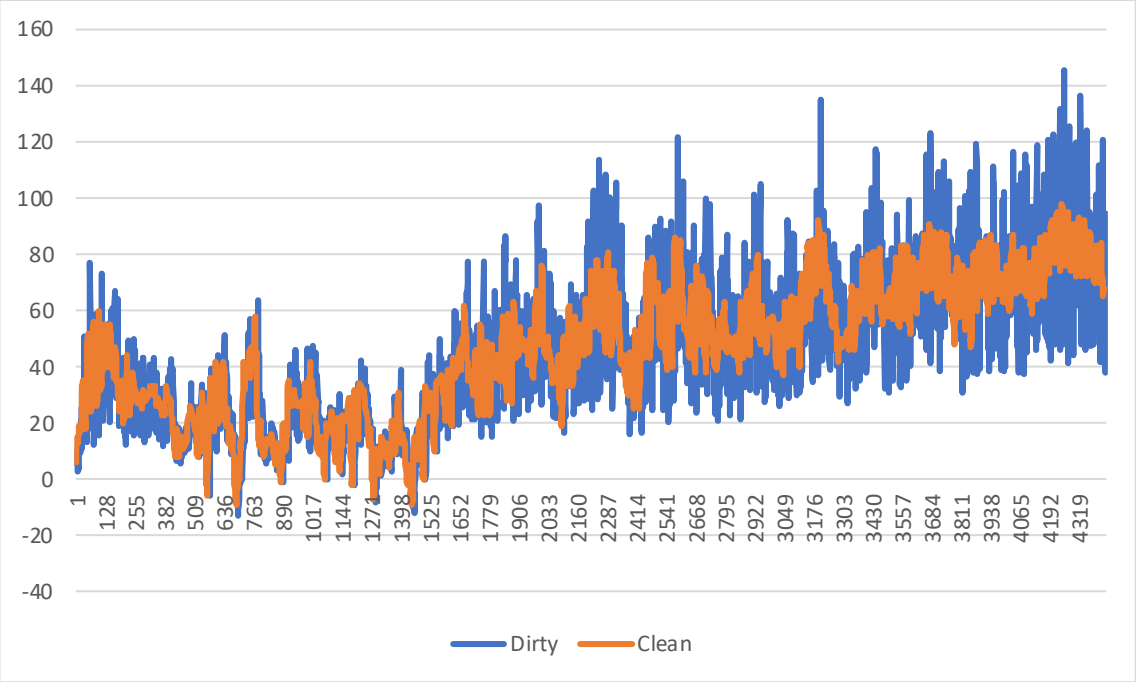
Appendix 51 KNN Output Of High Rate Outlier Repair Half



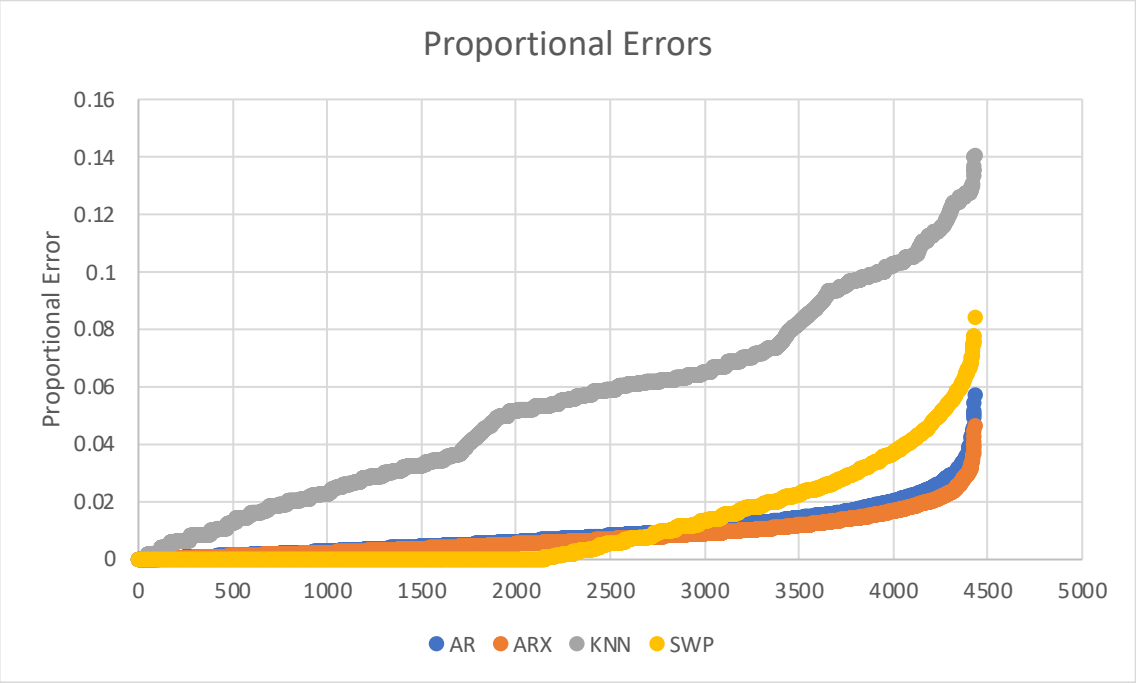
Appendix 52 SWP Output Of High Rate Outlier Repair Half



Appendix 53 Dirty Data Of High Rate Outlier Repair Half



Appendix 54 Proportional Errors Of High Rate Outlier Repair Half



Appendix 55 Outputs Of High Rate Outlier Repair Half

	AR	ARX	KNN	SWP
Max	0.057345	0.046503	0.140624	0.084279
Median	0.0071	0.006082	0.05382	0.001264
Mean	0.009195	0.007696	0.053904	0.011479
Standard Deviation	0.00798	0.006503	0.033067	0.016429
RMSE	6.299919	5.226742	33.27731	10.33632
Time	0.021026	0.010232	0.113337	1.98722

Appendix 56 Outputs of Outlier Repair

	Metric	AR	ARX	KNN	SWP
Full Dataset With Low Error rate	Median Inaccuracy	0.004294	0.00449	0.049051	0
	Highest Inaccuracy	0.059191	0.055479	0.138323	0.086227
	Mean Inaccuracy	0.006203	0.00578	0.050289	0.003187
	Standard Deviation	0.006363	0.00531	0.031716	0.009785
	RMSE	4.349757	3.912273	28.34284	5.223187
	Time	0.0185153	0.016745933	0.210156333	4.012226267
Half Dataset With Low Error rate	Median Inaccuracy	0.003949	0.004226	0.03963	0
	Highest Inaccuracy	0.050431	0.042936	0.125759	0.077111
	Mean Inaccuracy	0.005855	0.005563	0.043466	0.003633
	Standard Deviation	0.006107	0.00516	0.029921	0.009629
	RMSE	4.354195	3.905678	27.75466	5.209444
	Time	0.011954	0.020929	0.113882	1.801031
Full Dataset With High Error rate	Median Inaccuracy	0.00775	0.006468	0.058972	0
	Highest Inaccuracy	0.060958	0.055479	0.143462	0.084279
	Mean Inaccuracy	0.009759	0.008035	0.060553	0.0118
	Standard Deviation	0.008203	0.006609	0.034055	0.017574
	RMSE	6.282781	5.21699	33.27731	10.33632
	Time	0.018711	0.024819	0.212992	4.284844
Half Dataset With High Error rate	Median Inaccuracy	0.0071	0.006082	0.05382	0.001264
	Highest Inaccuracy	0.057345	0.046503	0.140624	0.084279
	Mean Inaccuracy	0.009195	0.007696	0.053904	0.011479
	Standard Deviation	0.00798	0.006503	0.033067	0.016429
	RMSE	6.299919	5.226742	33.27731	10.33632
	Time	0.021026	0.010232	0.113337	1.98722