Combining Imputation with Feature Selection

Pepijn Meijer (s2957566), Narendra Setty (s2944200), Wander Stribos (s2398443), Emiel Tempelman (s2765314)

Dept. of Technical Computer Science, University of Twente
Enschede, Netherlands

p.j.meijer@student.utwente.nl, n.setty@student.utwente.nl, w.h.stribos@student.utwente.nl, e.tempelman@student.utwente.nl

QuickSelection is a promising feature selection method, though not much further research is done on it. This study compares the selection method with others in combination with various imputation algorithms to study its efficacy and speed on two datasets with various amounts of missing data. Two datasets were used, one built for a feature selection challenge and a dataset of simulated transactions, some of which are fraudulent.
Results showed that QuickSelected excelled in datasets with more interdependent features, while being outperformed by selection methods such as Lasso selection on datasets with more independent features, especially on training time. Additionally, it showed various niches certain imputation methods (with kNN, GAIN, MissForest, and MICE boasting high accuracies in certain contexts) could still be useful in, even if they might be outperformed on other datasets.

I. Defined Goal and Motivation

The Quick Selection method is a fast, unsupervised feature selection method that utilises the neuron strength of sparsely connected denoising autoencoders. However, little research has been done on the performance of this method on datasets with missing data. [2]

In this research, Quick Selection is tested against other popular feature selection methods in combination with various imputation methods using two datasets with various amounts of MCAR (missing completely at random) data to research the efficacy and speed of each combination.

II. Method

We used two datasets to test the files. First the Madelon dataset was used, an artificial dataset made for the NIPS 2003 feature selection challenge. For this dataset, datapoints were grouped in 32 clusters in a five-dimensional hypercube and randomly given labels +1 or -1 (consistent with each cluster). The dataset consists of each point's five coordinates, fifteen (redundant) combinations of the coordinates, and 480 'probe' distractor features. This created a large dataset with many features (4400 by 200) optimised for a feature selection competition, which thus gave a near-ideal context for particular methods.

However, the fraud database was also used to look into how the various options fared in a more realistic setting wherein the information gained with each feature was more randomly distributed, as opposed to 480 of the features being only distractors. This is a simulated credit card transaction dataset containing legitimate and fraudulent transactions over two years generated using the Sparkov Data Generation tool created by Brandon Harris [7]. To combat computational constraints and make the dataset more balanced, the 7506 samples classified as fraud and an equal number of randomly selected non-fraud cases were used. Additionally, except for gender (which was binary in the dataset and replaced with 1 for male, 0 for female,) the categorical features were also removed. This was because, except for MissForest (an imputation method), the imputation and feature selection methods all required purely numerical data, and the categories consisted of too many independent options for one-hot encoding not to explode the feature count.

We then followed the following step plan for each combination of dataset, amount of missing data, imputation method, and feature selection:

First, the dataset was loaded, and a percentage of the values in the dataset randomly selected to be deleted. Then, the imputation method was used to refill the dataset, after which the feature selection selected the features it deemed best.

Finally, an SKlearn Extra Trees classifier [3] was trained on the imputed data with the selected features, and the accuracy on the test set (which was imputed as a separate dataset) was noted.

This was repeated five times, and the average accuracies of the final decision tree were calculated and noted, along with the total calculation time required for the combination.

For QuickSelection, the clustering accuracy was also noted, as the method was built for unsupervised data. (The other methods did not train unsupervised, so we did not deem clustering accuracy a useful metric.) We also utilised the sparsity by setting epsilon to 13 and the dropout (zeta) to 0.2 (to prevent overfitting), which appeared to result in high accuracy, as can be seen in appendix C. Here, the accuracies and training times were noted of various combinations of epsilon, zeta, and ratios of missing data on the fraud dataset with median imputation.

For each combination, we removed 99%, 50%, 30%, 10%, or 0% of the data. We then applied the mean, median, kNN, soft imputation, MICE, MissForest, Gain, and MIRACLE imputation methods from the HyperImpute [4] and SKlearn [3] libraries. (MIRACLE was only used on Fraud as was unfeasibly slow on Madelon.) Finally, the feature selection

methods Lasso, RFE, GAIN [3], Fischer [16], and Quick selection [2] were used.

III. Technical Depth

Imputation methods

Mean and median imputation address missing data by substituting missing values with the mean and median of the available data, respectively. These techniques are restricted to numerical datasets and do not preserve the inherent relationships between variables. Consequently, they can distort the original data distribution. Hence, they are generally recommended only for datasets with minimal missing values as a provisional measure. [5]

The kNN imputer identifies the k Nearest Neighbours (most similar rows of data) of each row with missing values and fills in the missing value with the mean value for that feature of the neighbours. [3] In our implementation, we set k to five.

The SoftImpute algorithm fills missing data by repeatedly estimating missing values and creating low-rank matrix approximations. It uses spectral regularization by applying a soft-thresholded Singular Value Decomposition (SVD) to reduce reconstruction error while constraining the solution to be low-rank. Therefore, by iteratively replacing missing values with values that are estimated from SVD, the algorithm efficiently computes regularized solutions to approximate large incomplete matrices. [10]

MissForest is an algorithm that initially fills missing data with median or mode values. It then iteratively uses Random Forest models to predict the missing values based on other features, improving the accuracy of the imputation with each iteration until convergence. [11]

GAIN (Generative Adversarial Imputation Nets) utilises a generative adversarial network (GAN) approach, which uses two networks: a generator and a discriminator, that compete against each other. The generator attempts to produce plausible data samples that fill in the missing values, while the discriminator attempts to guess the authenticity of the generated samples compared to the real data using hints regarding the missing values. By iteratively improving both networks, GAIN effectively captures the underlying data distribution, resulting in high-quality imputations. [12]

MIRACLE imputation is a causally aware imputation algorithm that iteratively refines the imputation of missing data by modelling the missingness generating mechanism and also encouraging consistency with the causal structure of the data to improve imputation accuracy by preserving the causal relationships within the dataset. [17]

The MICE (Multiple Imputation by Chained Equations) algorithm generates multiple imputations for incomplete multivariate data. It imputes the missing values using the other variables/features of the dataset called predictors. When predictors have missing values, they are filled with an initial value, after which the most recently generated imputations are used to complete them. [18]

Feature selection methods

Quick Selection uses a Sparse denoising autoencoder (DAE). A denoising autoencoder is trained by adding noise to a copy of the dataset and making it reconstruct the original dataset. This can then be used to remove random noise from new data. Sparsity is introduced in the denoising autoencoder to enhance its ability to select relevant features by focusing on essential signals while ignoring noise. This is achieved by applying a Laplace Prior, which encourages more sparse representations in the hidden layer [19]. In this context, sparsity means adding constraints that require only a small number of neurons in the hidden layer to activate simultaneously. By promoting sparse representations, the model can generalise better, reducing the influence of irrelevant features and emphasising the most critical connections. [20] The method then selects the features with the strongest weights in the neurons. [2]

Recursive Feature Elimination (RFE) is a wrapper-type feature selection algorithm. RFE trains the model on all the features, removes the least important feature, and retrains the model without that feature. This continues until the desired number of features has been reached. [8]

Least Absolute Shrinkage and Selection Operator (LASSO) transforms each coefficient by a constant component λ. It decreases the residual sum of squares subject to the sum of the absolute value of the coefficients being less than a constant. It uses the L1-penalty function to achieve this. [9]

Information Gain Selection works by calculating the dependency between the features and the target of a dataset. [13] The higher the score, the more dependent the target is on the feature. This score is calculated using a Mutual Information Classifier. [14] [3]

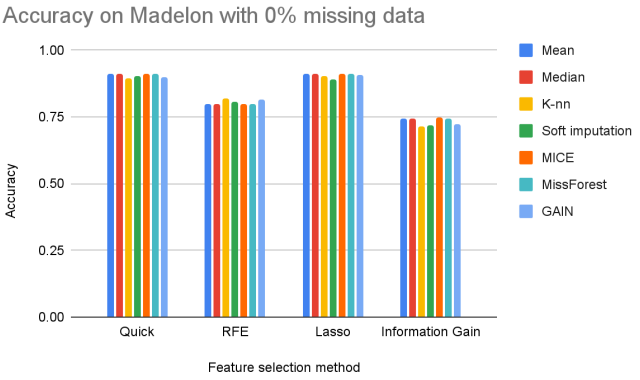
The key idea of using the Fisher score for feature selection is to select features such that in the data space, the distances between data points in different classes are as large as possible, and the distance of data points in the same class is as small as possible [16]. The implementation uses the Laplacian Score as an extension of the Fisher score [6].

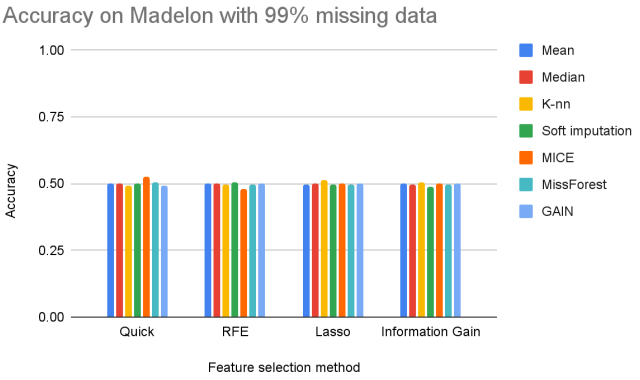
IV. Results

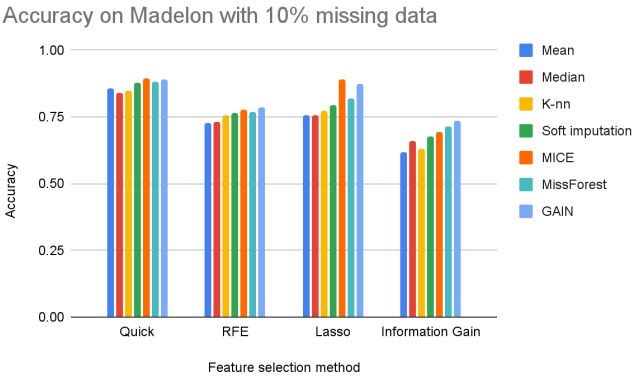
Due to time constraints, the models were run on computers with slightly different specifications. Thus, the calculation times are not perfectly representative, and only significant differences should be compared.

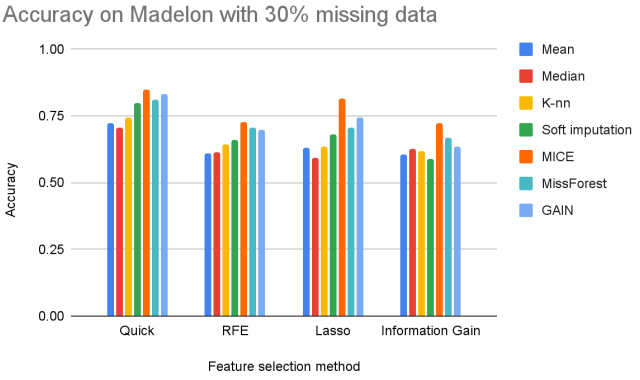
Madelon

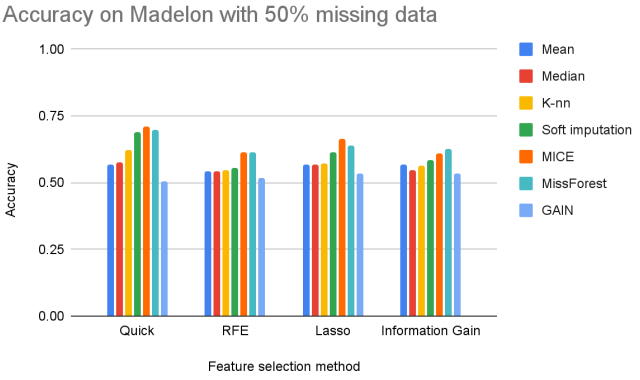
The following graphs are for the Madelon Dataset, the pure data for the Madelon dataset can be seen in Appendix A.











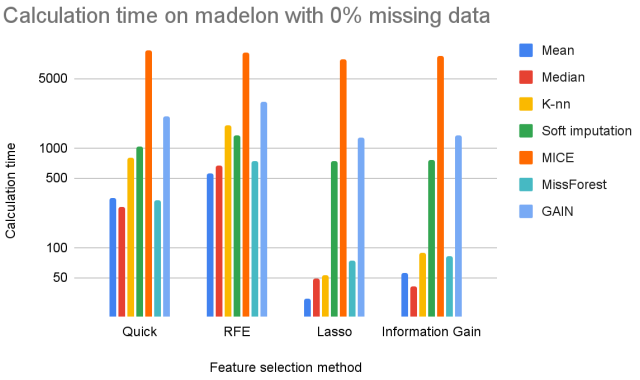
These graphs represent the accuracies for each missing data percentage. From the 0% graph, it can be deduced that Lasso and Quick Selection have the highest average accuracy with them being about the same; the imputation methods all seem to be about the same with a slight advantage for Mean, Median, Mice and MissForest for every feature selection except RFE, which seems to work better with kNN, Soft imputation and GAIN.

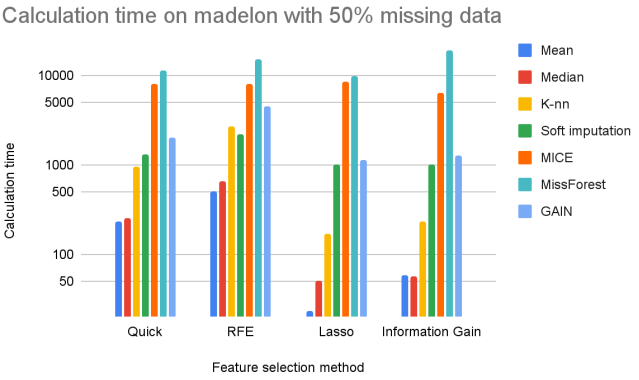
For 10% missing data, the imputation methods start showing differences as MICE and GAIN start to show an improvement over other methods. MissForest also seems to do very well for QuickSelection and Information Gain.

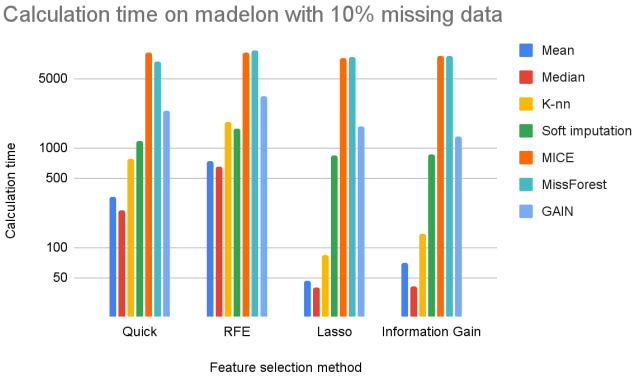
For 30% missing data, this trend does not seem to slow down. The top three imputation methods still seem to be Mice, MissForest and GAIN. It can still be seen that Quick Selection is the most accurate, while LASSO seems to decrease in accuracy to a similar level to RFE and information gain with most imputation methods.

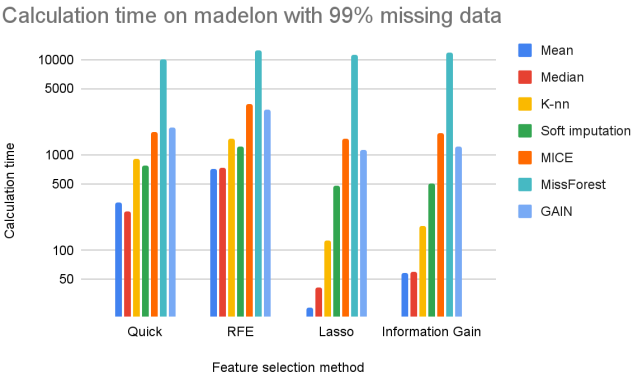
For 50%, GAIN really drops off and gets close to 50% accuracy, Soft imputation seems to catch up and becomes accurate for Quick Selection. Quick Selection remains the most accurate feature selection method.

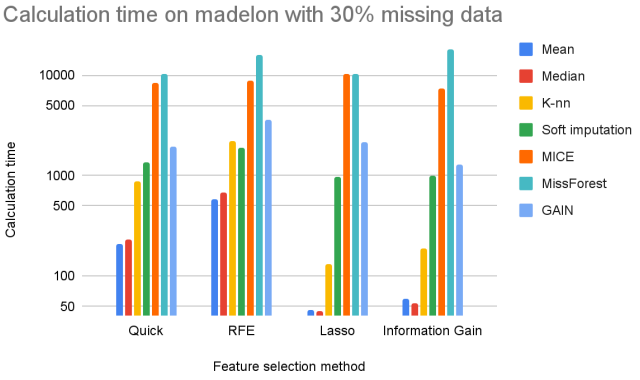
For 99% missing data, no major differences can be seen as almost all methods devolve into over-complicated guessing, with accuracies around 50%.







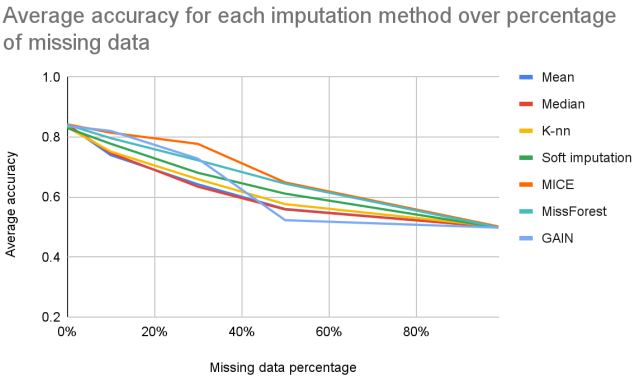


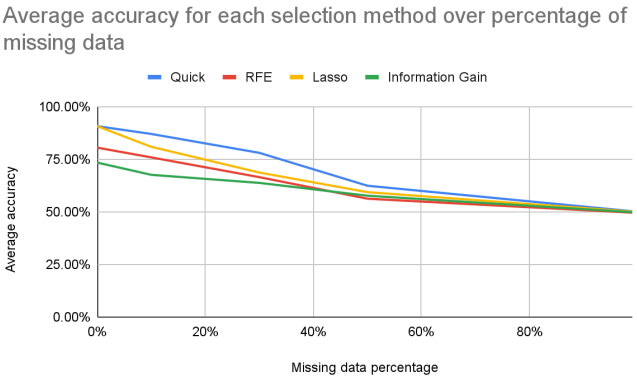


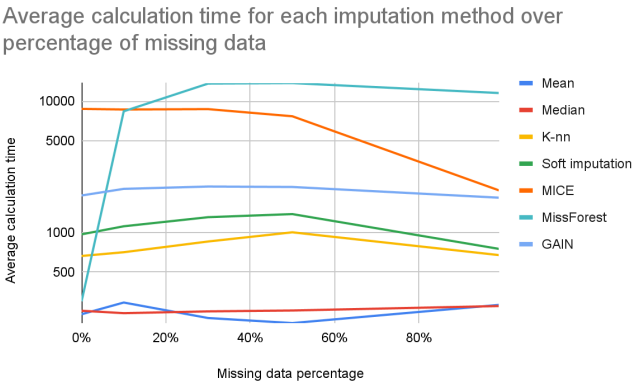
These graphs show the same combinations as the previous set but the time taken to train and use the model instead of the accuracy. For 0% missing data, Mice and GAIN seem to take the longest. The shortest methods seem to be Mean, Median and MissForest. For MissForest, this is interesting as it very quickly increases in calculation time once missing data becomes more than 0%. This could be explained by it simply being the only algorithm that checks if there is data to be imputed before starting the major calculations, while the other methods will still take time fitting to the dataset even if there is nothing to impute.

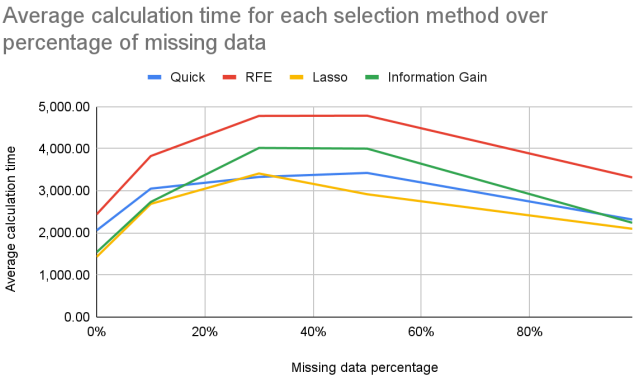
For 10% missing data, MissForest takes a lot more time and joins MICE in being the closest, while kNN comes closer to Mean and Median in speed.

There appear to be not too many differences between 10%, 30%, 50%, and 99%, except for MissForest taking up more and more time compared to the others. Additionally, MICE becomes quicker the more data is missing. This suggests that it simply needs more calculations for the fitting than the imputations compared to other methods and thus becomes relatively quicker the less data there is to train on.









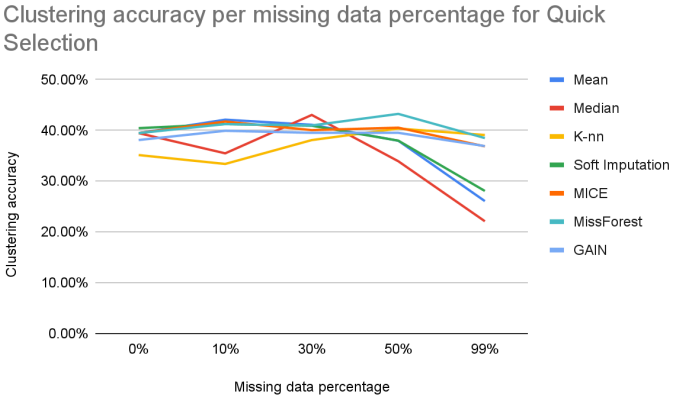
These graphs represent each imputation method's accuracy and calculation time if you take the average of all feature selection methods for that percentage of missing data.

You can see that MICE is the most accurate overall, followed by GAIN for less than 50% missing data, then MissForest, Soft Imputation, kNN, Mean, Median and lastly, GAIN for more than 50% missing data. The slowest methods are Mice and MissForest, with GAIN not following far behind. This seems to be linearly correlated to the accuracy of the methods. In the second graph, you can also see three distinct zones where imputation methods can be divided. Mean and median are the ‘quick and dirty’ methods, which take up around 250 seconds, of which most is likely to be the selection methods. Afterwards, GAIN, kNN, and soft imputation follow as more rounded methods. Lastly, MICE and MissForest follow up as the more ‘robust’ datasets, which take around twice as long as GAIN the follow-up.

You can see that with less than 50% missing data, GAIN is a very strong option, given that its accuracy is similar to that of the robust methods while taking around half the time. However, at 50% missing its accuracy takes a large drop and it gets surpassed by the other methods. After this, Soft Imputation becomes a strong second option. However, if calculation time can be ignored, MICE reigns supreme as the best method if more than 10% of the data is missing.

These graphs represent each feature selection method's accuracy and calculation time if you take the average of all imputation methods for that percentage of missing data.

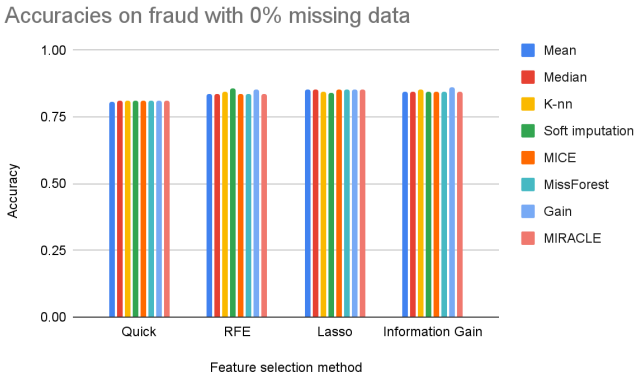
You can see that RFE takes up the most time, followed by Information Gain if larger amounts of data are missing. Since they also have a consistently lower accuracy, these methods can be dismissed in this context. QuickSelection has a consistently higher accuracy, followed by Lasso.

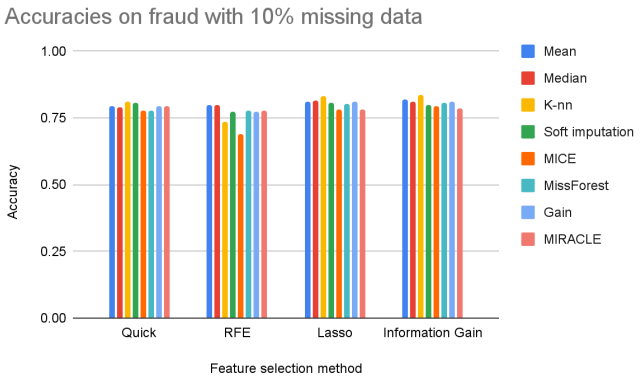


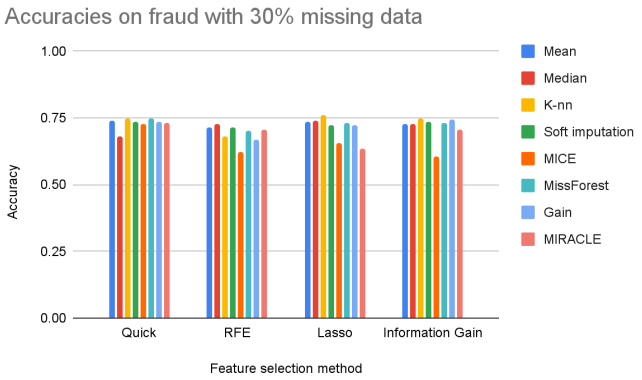
This graph represents the clustering accuracy for Quick Selection. You can see a lot of differences between percentages. However, soft imputation seems to be consistently high.

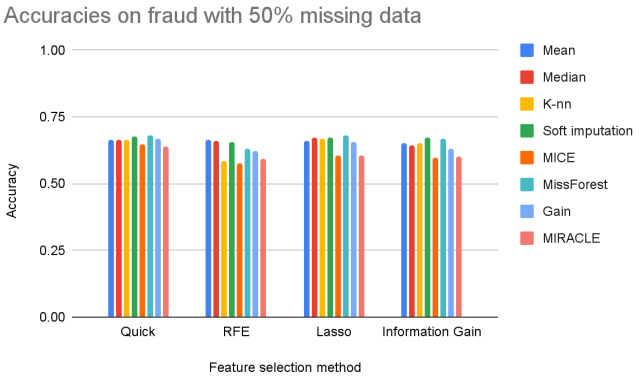
Fraud

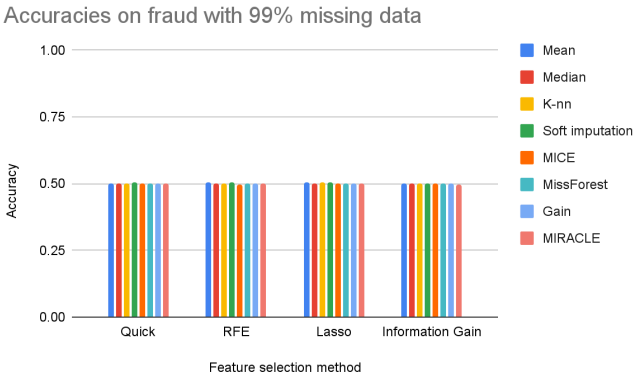
The following graphs are for the Fraud dataset; the pure data for the Fraud dataset can be seen in Appendix B.





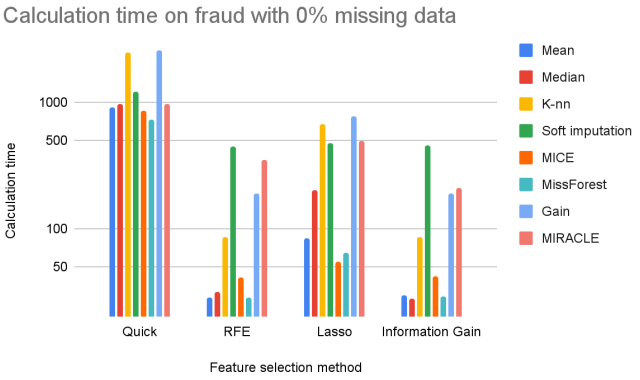


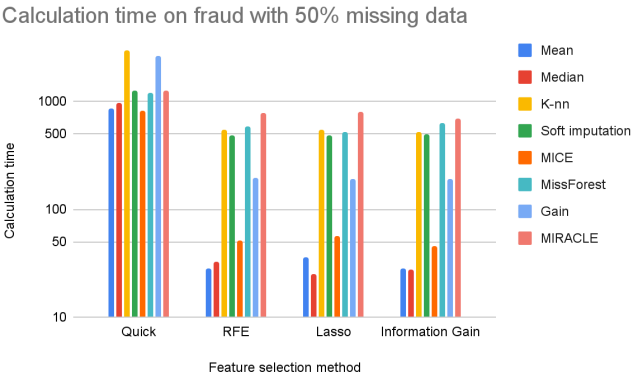


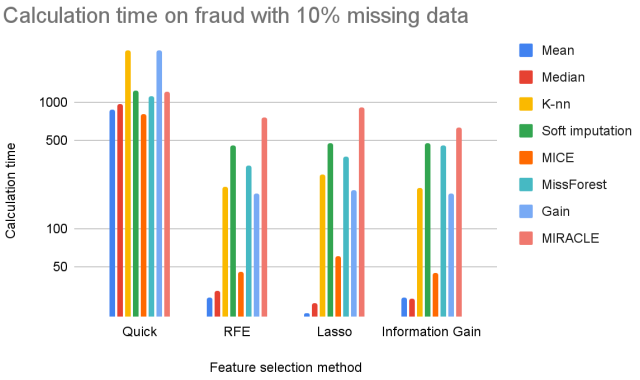


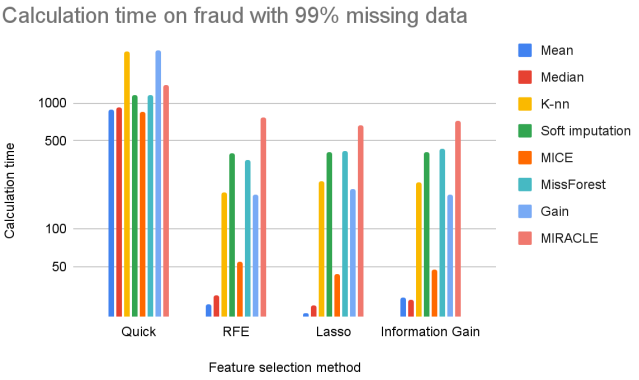
These graphs represent the accuracies for each missing data value on each combination of imputation and selection methods.

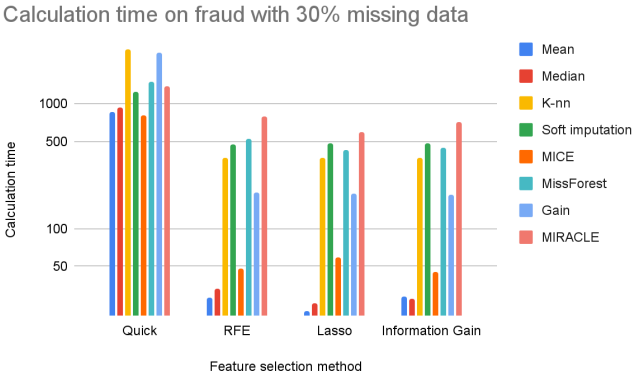
For the fraud dataset, the differences between feature selection methods and imputation methods seem to be much smaller. The differences are not significant enough to make conclusions. However, it can be observed that MICE and MIRACLE are often lower than others, while some of the quicker methods (especially mean and kNN) have more accurate results.







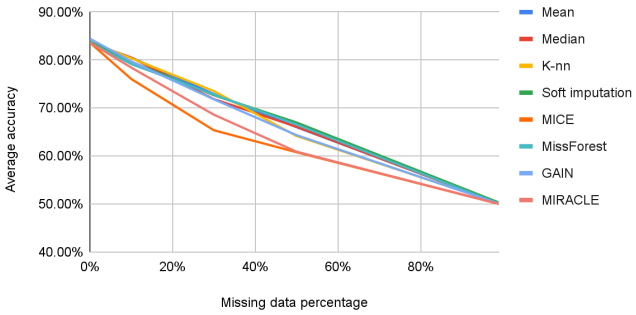




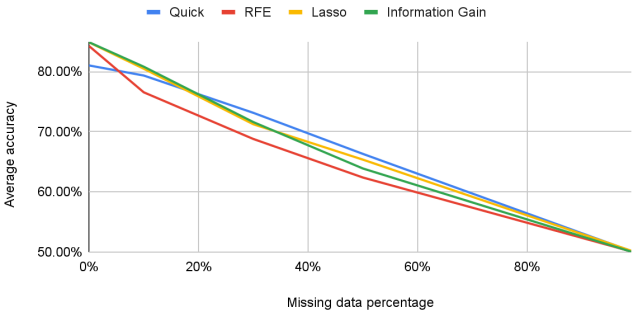
These graphs represent the time taken per percentage of missing data. It is clear that, once again, specific methods take significantly longer than others. However, the correlation between accuracy and time spent calculating is considerably less present.

On this dataset, QuickSelection takes substantially longer than other selection methods. For Quick Selection, kNN and Gain seem to be consistently the slowest, while MICE, mean and median seem to be the quickest, even with less missing data, where MICE would take longer on Madelon. On the other hand, Soft imputation, MissForest and MIRACLE seem to be the slowest.

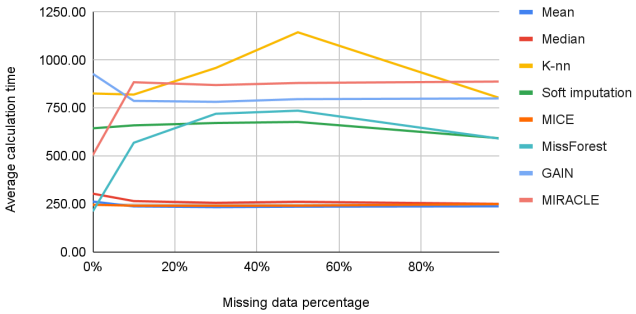
Average accuracy for each imputation method over percentage of missing data



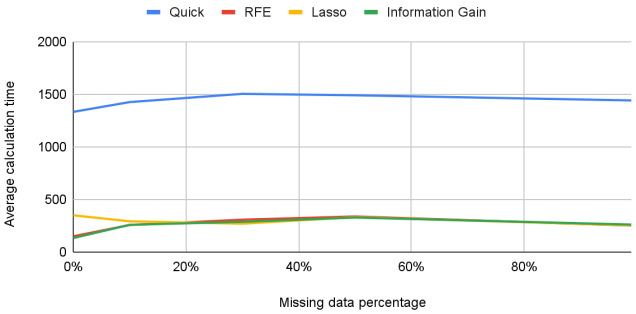
Average accuracy for each selection method over percentage of missing data



Average calculation time for each imputation method over percentage of missing data



Average calculation time for each selection method over percentage of missing data

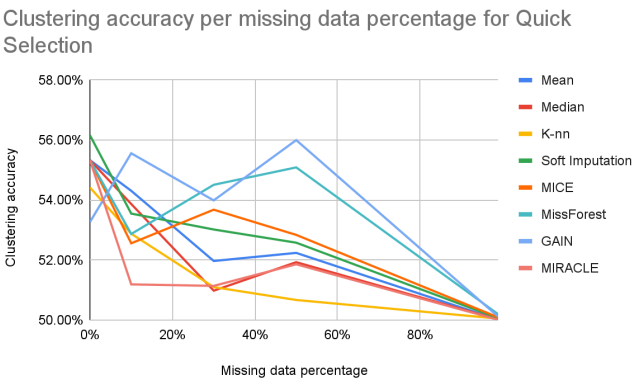


These graphs represent each imputation method's accuracy and calculation time if you take the average of all feature selection methods for that percentage of missing data.

The graph shows that while most methods have similar accuracies, MICE and MIRACLE visibly underperform. At the same time, Mean, Median, and MICE are the fastest, followed by Soft imputation and MissForest.

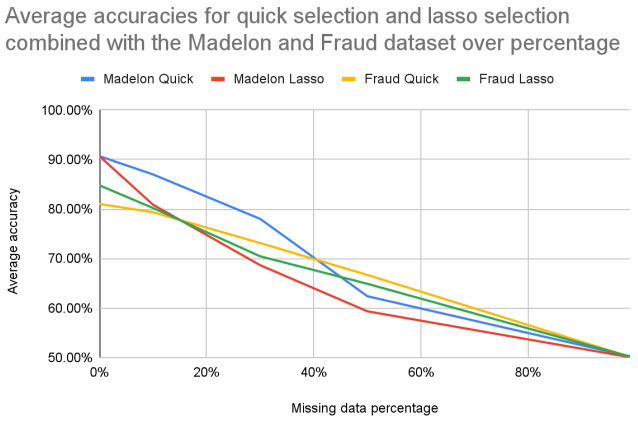
These graphs represent each feature selection method's accuracy and calculation time if you take the average of all imputation methods for that percentage of missing data.

It can be seen that QuickSelection is not as good here as with Madelon, also given the fact that it was the slowest method by quite a margin. The method with the highest accuracy was either Lasso or Information Gain depending on how much missing data is present. RFE performed the worst. This difference in performance from QuickSelection is not inexplicable: Autoencoders use context acquired from other features to perform their denoising. This means that in Madelon, if the foundation and supporting features are found and identified, they can be used to remove noise from each other (up to a certain level). However, in the fraud dataset, most features do not have much effect on each other, and a denoising autoencoder would not perform as well.



This graph shows the clustering accuracy for Quick Selection.

The imputation method that shows the highest average clustering accuracy is GAIN followed by MissForest and MICE. The methods with the lowest clustering accuracy are kNN, MIRACLE and Median.



This graph represents the average accuracy of all imputation methods of QuickSelection and Lasso on both the Madelon and Fraud datasets.

On low amounts of missing data, Madelon gets a higher average accuracy, especially when QuickSelection is used. However, if a large amount (50%) of the data has to be imputed, the accuracies start to drop and Fraud gets higher results.

V. Conclusions

As all our imputation methods except MissForest require numerical data, one hot encoding needs to be used to change categorical data to numerical data. Because of the size of the fraud dataset and the many options for categorical values for features like “Job”, “Street”, “City”, and “Name”, one-hot encoding becomes too computationally expensive and time-consuming.

QuickSelection worked significantly better on the Madelon dataset. We are unsure if this can be pinned on the higher degree of realism in the fraud database (as the Fraud dataset was based on a simulation while the Madelon dataset was less mathematically complex in its construction.) or the lack of context which could be used for the denoising autoencoder, as mentioned in the results. (E.G., if noise was applied to a feature such as the date of birth, it might be more difficult to figure out which data can be used to deduce the original data.) More research could be done to clear this up, specifically by testing the QuickSelection on image datasets such as the MNIST database of handwritten digits, [21] a dataset of realistic images where context can be used in the autoencoder.

We also saw while comparing the results of the two datasets that with QuickSelection on fraud, changing the imputation method would not result in major differences in accuracies compared to other feature selection methods and QuickSelection on Madelon. We speculate that this might be caused by the noise cancelling out much of the accuracies on the imputed values without sufficient context to return to the original values.

Another thing we concluded from comparing the results of the two datasets is that QuickSelection is efficient on smaller datasets, as it takes significantly more time than other feature selection methods, while it even becomes faster than some on a larger database. (Madelon)

When most of the data is missing, it does not matter what combination of imputation method and feature selection method you use, as there is so little usable data it will not get much better than a coin toss.

The accuracies and speeds of imputation methods varied depending on the size of the dataset. Computationally heavy imputation methods (MICE, MissForest) created better results when working on bigger datasets. We can also see from the differences between the Madelon and Fraud datasets that MICE imputation becomes much quicker on small datasets but less accurate due to MICE having less data to work on.

MissForest seems to be best used on smaller datasets as it uses less computation time while retaining the same accuracy.

While there might be a specific context in which MIRACLE is effective, we did not find it as it usually gave worse accuracies than competing methods whilst taking similar times as the most accurate methods.

We set out to research which imputation methods were best suited for QuickSelection compared to other methods. However, it was slightly more complex than that. Lasso gave better results in more complex datasets where features are more independent if not too much data was missing, and imputation methods could be chosen based on the computational time provided to the user. While MissForest or

MICE did give slightly better results, a method such as mean also gave acceptable results for a fraction of the calculation time. kNN was, however, a good middle ground. However, in datasets where variables were more dependent on each other (Madelon), Quickselection had significantly better results while also being faster than some of the methods tested. In these situations, GAIN was surprisingly effective with little missing data without taking too much time, while MICE or MissForest gave more consistently high results, especially when more data was missing.

VI. Critical Reflection

While testing the different combinations of feature selection and imputation methods, we saw a high accuracy (99.64%), which did not change, so we started to look at what could have caused this. This search started by looking at our dataset using the knowledge we got from the Data Quality course. We found that our data was mostly non-fraud (99%), so it was a very unbalanced data set, and we concluded from this that our algorithm just guesses that everything is non-fraud and thus would be very accurate. We used knowledge from the Data Quality course to generate and impute missing data. The Explainable AI and advanced courses for Machine Learning for Data Types also helped us understand the internals of the various imputation and selection methods and theories we had certain results.

VII. Ethical

Feature selection has been proposed as a way to decrease the computational costs of high-dimensional data [2] and, thus, the carbon footprint, to which large-scale data science is a significant contributor. [3] However, care should always be given to the optimisation of machine learning through feature selection, as it does, by definition, simplify the calculation and ways the model views the world. Especially when building systems with strong effects in life, one should pay attention to not simplifying data too much. As a (slightly obvious) example, predictive policing algorithms, which are already known for their discriminating biases [1], could get an even more black-and-white dataset and learn even stronger biases. This is increased even more if large amounts of data are missing and imputed, especially if one of the simpler imputation methods is used as they make strong generalisations on the data.

References

[1] Will Douglas Heaven, Predictive policing algorithms are racist. They need to be dismantled. MIT Technology Review, 2020: https://www.technologyreview.com/2020/07/17/1005396/predictive-policing-algorithms-racist-dismantled-machine-learning-bias-criminal-justice/

[2] Atashgahi, Z., Sokar, G., van der Lee, T. et al. Quick and robust feature selection: the strength of energy-efficient sparse training for autoencoders. Mach Learn 111, 377–414 (2022). https://doi.org/10.1007/s10994-021-06063-x

[3] Pedregosa et al, Scikit-learn: Machine Learning in Python, JMLR 12, pp. 2825-2830, 2011, https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

[4] D. Jarrett et al, HyperImpute: Generalized Iterative Imputation with Automatic Model Selection, 39th International Conference on Machine Learning (2022) https://arxiv.org/abs/2206.07769

[5] Raheem, E. Statistical Approaches for Epidemiology, Springer, pp. 293-316, 2023, https://doi.org/10.1007/978-3-031-41784-9_18

[6] Q. Gu, Z. Li and J. Han, Generalized Fisher Score for Feature Selection (2012 feb, 14), Cornell University. https://doi.org/10.48550/arXiv.1202.3725 Accessed 29 October 2024.

[7] Shenoy, K., Credit Card Transactions Fraud Detection Dataset. (2020, August 5). Kaggle https://www.kaggle.com/datasets/kartik2112/fraud-detection

[8] Brownlee, J., Recursive Feature Elimination (RFE) for Feature Selection in Python, https://machinelearningmastery.com/rfe-feature-selection-in-python/

[9] Muthukrishnan, R., Rohini, R., LASSO: A feature selection technique in predictive modeling for machine learning. (2016, October 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/7887916

[10] Mazumder, R., Hastie, T., Tibshirani, R., Spectral Regularization Algorithms for Learning Large Incomplete Matrices. (2010, August 1). ACM Digital Library, pp. 2287-2322, https://dl.acm.org/doi/10.5555/1756006.1859931

[11] Ye, A., towardsdatascience 31 August 2020, https://towardsdatascience.com/missforest-the-best-missing-data-imputation-algorithm-4d01182aed3 Accessed 29 October 2024

[12] Jinsung, Y., Jordan, J., van der Schaar, Michaela., GAIN: Missing Data Imputation using Generative Adversarial Nets, (2018, June 7), Arxiv, https://doi.org/10.48550/arXiv.1806.02920

[13] Azhagusundari, B., Thanamani, A. S., Feature selection based on information gain. International Journal of Innovative Technology and Exploring Engineering (2013). IJITEE, 2(2), 18-21. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=e17df473c25cccd8435839c9b6150ee61bec146a

[14] Kraskov, A., Stögbauer, H., & Grassberger, P., Estimating mutual information. (2004). Physical Review E, 69(6). https://doi.org/10.1103/physreve.69.066138

[15] I. Guyon. "Madelon," UCI Machine Learning Repository, 2004. [Online]. Available: https://doi.org/10.24432/C5602H

[16] J. Li, (2019), open-source feature selection repository in python [Online], Available: https://github.com/jundongl/scikit-feature
[17] Kyono, T. et al, MIRACLE: Casually-Aware Imputation via Learning Missing Data Mechanisms, Arxiv, (2021) https://arxiv.org/abs/2111.03187
[18] Platias, C. and Petasis, G., A comparison of Machine Learning Methods for Data imputation, Machine Learning Plus, SETN 2020: 11th Hellenic Conference on Artificial Intelligence (2020), pp 150-159, https://doi.org/10.1145/3411408.3411465
[19] Goodfellow, I. et al., Deep Learning book, MIT Press, 2016, https://www.deeplearningbook.org/
[20] Geeks for Geeks, Sparse Autoencoders in Deep Learning, geeksforgeeks, https://www.geeksforgeeks.org/sparse-autoencoders-in-deep-learning/ (Accessed 30th October 2024)
[21] Yann LeCun, The MNIST Database of handwritten digits, Courant Institute, NYU Corinna Cortes, Google Labs, New York Christopher J.C. Burges, Microsoft Research, Redmond. http://yann.lecun.com/exdb/mnist/

VIII. Appendix A Madelon

TABLE I
Quick Selection

Imputation Method	Percentage of Missing data				
	0 %	**10 %**	**30 %**	**50 %**	**99 %**
Mean	c: 39.47% 91.23% 311.03s 1.40s	c: 42.12% 85.80% 319.01s 1.17s	c: 41.11% 72.47% 207.28s 0.96s	c: 37.97% 56.67% 229.83s 1.01s	c: 26.08% 50.00% 319.78s 1.04s
Median	c: 39.47% 91.23% 253.51s 0.91s	c: 35.49% 84.17% 235.18s 0.98s	c: 43.05% 70.43% 228.39s 0.91s	c: 33.93% 57.47% 254.06s 0.95s	c: 22.11% 50.13% 258.09s 0.83s
kNN	c: 35.15% 89.53% 794.88s 2.36s	c: 33.40% 84.70% 770.61s 2.27s	c: 38.10% 74.37% 876.73s 2.42s	c: 40.37% 62.33% 947.01s 2.33s	c: 39.10% 49.10% 909.01s 2.44s
Soft imputation	c: 40.45% 90.37% 1023.81s 1.24s	c: 41.25% 87.60% 1182.58s 1.20s	c: 40.92% 79.87% 1353.99s 1.19s	c: 37.99% 68.87% 1305.37s 1.11s	c: 28.08% 50.23% 774.65s 1.12s
MICE	c: 39.47% 91.23% 9623.74s 1.58s	c: 41.75% 89.53% 9106.51s 1.51s	c: 40.06% 84.70% 8393.70s 1.54s	c: 40.54% 71.10% 7924.33s 1.59s	c: 36.86% 52.43% 1761.42s 1.44s
MissForest	c: 39.47% 91.23% 298.06s 0.95s	c: 41.25% 88.17% 7385.65s 1.02	c: 40.98% 81.00% 10288.55s 1.30s	c: 43.26% 69.97% 11286.82s 1.00s	c: 38.50% 50.67% 10207.00s 5.71s
GAIN	c: 38.11% 90.07% 2097.03s 2.55s	c: 39.93% 89.07% 2347.67s 2.35s	c: 39.53% 83.33% 1943.79s 2.29s	c: 39.53% 50.40% 2011.00s 2.36s	c: 36.92% 49.37% 1975.62s *Missing, shown in graphs as 2.36 seconds.*
MIRACLE	c: 40.93% 91.13% 10421.61s 7.08s				

TABLE II
RFE Selection

Imputation Method	Percentage of Missing data				
	0 %	**10 %**	**30 %**	**50 %**	**99 %**
Mean	79.83% 555.77s	72.57% 733.90s	60.90% 578.57s	54.17% 504.36s	49.87% 719.50sx
Median	79.83% 669.81s	73.23% 655.10s	61.33% 674.65s	54.43% 655.10s	49.93% 742.16s
kNN	81.80% 1711.70s	75.57% 1837.11s	64.23% 2214.64s	54.80% 2665.45s	49.70% 1469.64s
Soft imputation	80.83% 1356.02s	76.53% 1567.50s	65.80% 1900.35s	55.57% 2200.16s	50.40% 1239.96s
MICE	79.83%	77.83%	72.60%	61.43%	48.00%

	9109.13s	9125.18s	8743.33s	8038.96s	3410.71s
MissForest	79.83% 741.96s	76.87% 9568.52s	70.73% 15784.05s	61.37% 14958.73s	49.47% 12631.74s
GAIN	81.50% 2951.70s	78.47% 3290.56s	69.77% 3556.72s	51.90% 4465.73s	49.87% 3012.01s
MIRACLE					

TABLE III
LASSO SELECTION

Imputation Method	**Percentage of Missing data**				
	0 %	**10 %**	**30 %**	**50 %**	**99 %**
Mean	91.17% 30.33s	75.73% 46.69s	63.07% 45.90s	56.67% 23.24s	49.67% 25.06s
Median	91.17% 48.43s	75.53% 39.73s	59.50% 45.37s	56.90% 50.83s	49.93% 40.74s
kNN	90.37% 52.96s	77.20% 84.53s	63.53% 132.71s	57.07% 168.34s	51.27% 127.68s
Soft imputation	89.17% 741.39s	79.23% 849.38s	68.00% 981.75s	61.53% 1004.48s	49.63% 481.00s
MICE	91.17% 7802.87s	89.10% 7949.88s	81.50% 10311.99s	66.23% 8396.80s	50.23% 1508.08s
MissForest	91.17% 73.58s	82.07% 8208.73s	70.53% 10206.24	63.80% 9658.35s	49.73% 11370.59s
GAIN	90.67% 1263.06s	87.37% 1643.21s	74.53% 2153.46s	53.30% 1121.37s	50.17% 1130.77s
MIRACLE	91.17% 11620.57s				

TABLE IV
INFORMATION GAIN SELECTION

Imputation Method	**Percentage of Missing data**				
	0 %	**10 %**	**30 %**	**50 %**	**99 %**
Mean	74.33% 55.51s	61.90% 70.72s	60.60% 60.12s	56.63% 58.53s	50.23% 58.41s
Median	74.33% 41.23s	65.80% 41.16s	62.60% 53.74s	54.73% 57.84s	49.73% 59.80s
kNN	71.63% 88.48s	63.27% 135.59s	61.77% 189.15s	56.43% 230.89s	50.33% 182.69s
Soft imputation	71.83% 750.79s	67.80% 859.91s	58.73% 994.30s	58.63% 1012.73s	48.97% 505.01s
MICE	74.33% 8427.83s	69.40% 8382.16s	72.13% 7356.92s	60.83 6344.20s	50.03% 1693.80s
MissForest	74.33% 82.17s	71.53% 8336.90s	66.83% 18171.25s	62.77% 19020.25s	49.60% 11938.00s
GAIN	72.50% 1341.81s	73.37% 1313.85s	63.60% 1298.63s	53.60% 1286.73s	49.87% 1244.22s
MIRACLE	74.33% 10938.87s				

IX. Appendix B Fraud

TABLE I
Quick Selection

Imputation Method	Percentage of Missing data				
	0 %	**10 %**	**30 %**	**50 %**	**99 %**
Mean	c: 55.33% 80.89% 909.09s 1.39s	c: 54.31% 79.45% 869.82s 3.21s	c: 51.97% 74.15% 854.04s 3.09s	c: 52.24% 66.28% 849.48s 3.36s	c: 50.01% 50.06% 875.28s 3.00s
Median	c: 55.33% 80.98% 956.59s 1.39s	c: 53.87% 79.19% 974.98s 1.29s	c:50.98% 68.17% 939.12s 1.32s	c: 51.93% 66.56% 961.73s 1.42s	c: 50.02% 50.21% 921.60s 1.14s
kNN	c: 54.42% 81.10% 2463.85s 2.56s	c: 52.88% 80.95% 2585.94s 2.66s	c: 51.10% 74.93% 2728.88s 2.81s	c: 50.67% 66.41% 2972.37s 3.21s	c: 50.05% 50.27% 2538.94s 2.27s
Soft imputation	c: 56.16% 81.14% 1211.42s 1.50s	c: 53.55% 80.61% 1231.19s 1.46s	c: 53.02% 73.49% 1246.26s 1.51s	c: 52.58% 67.82% 1246.52s 1.60s	c: 50.06% 50.35% 1156.88s 1.27s
MICE	c: 55.33% 80.98% 845.59s 1.55s	c: 52.56% 77.95% 812.99s 1.49s	c: 53.68% 72.91% 811.75s 1.62s	c: 52.84% 64.93% 811.60s 1.83s	c: 50.09% 50.00% 850.81s 2.07s
MissForest	c:55.33% 80.98% 727.06s 1.16s	c:52.87% 77.95 1121.84s 1.17s	c:54.51% 74.71% 1485.89s 1.18s	c:55.09% 68.27% 1211.57s 1.17s	c:50.19% 50.15% 1160.43s 0.93s
GAIN	c: 53.27% 81.25% 2568.73s 2.76s	c: 55.56% 79.48% 2567.56s 2.77s	c: 53.99% 73.65% 2554.78s 2.92s	c: 56.00% 66.66% 2607.07 3.21s	c: 50.11% 50.09% 2616.34s 2.95s
MIRACLE	c:55.33% 80.98% 963.20s 1.22s	c:51.19% 79.29% 1222.26s 1.25s	c:51.14% 73.25% 1391.21s 1.25s	c:51.86% 63.72% 1241.31s 1.44s	c:50.01% 50.12% 1392.29s 1.11s

TABLE II
RFE Selection

Imputation Method	Percentage of Missing data				
	0 %	**10 %**	**30 %**	**50 %**	**99 %**
Mean	83.80% 28.56s	79.80% 28.41s	71.59% 28.25s	66.62% 28.16s	50.41% 25.33s
Median	83.80% 31.56s	79.99% 32.11s	72.70% 33.19s	66.05% 32.36s	50.14% 29.97s
kNN	84.26% 86.36s	73.71% 215.20s	68.08% 372.21s	58.28% 542.30s	50.03% 196.97s
Soft imputation	85.64% 441.59s	77.51% 460.47s	71.59% 477.12s	65.74% 482.74s	50.29% 399.12s
MICE	83.80% 41.30s	68.90% 46.06s	62.44% 48.15s	57.77% 51.71s	49.86% 55.51s
MissForest	83.80%	77.75%	70.21%	63.10%	50.03%

	28.29	319.32s	523.89s	585.20	353.93s
GAIN	85.27% 190.02s	77.30% 191.03s	66.85% 194.42s	62.21% 196.25s	49.94% 186.28s
MIRACLE	83.80% 352.54s	77.58% 765.35s	67.02% 784.67s	59.44% 788.63s	50.14% 770.43s

TABLE III
LASSO SELECTION

Imputation Method	**Percentage of Missing data**				
	0 %	**10 %**	**30 %**	**50 %**	**99 %**
Mean	85.10% 84.61s	80.99% 21.61s	73.38% 21.81s	66.13% 36.01s	50.33% 21.31s
Median	85.10% 199.89s	81.58% 25.92s	73.91% 25.13s	67.38% 25.21s	50.24% 24.94s
kNN	84.35% 666.28	83.28% 267.25s	76.22% 367.54s	66.92% 539.98s	50.32% 240.85s
Soft imputation	83.89% 469.58s	80.51% 472.08s	72.10% 480.22s	67.36% 486.73s	50.40% 407.01s
MICE	85.10% 55.27s	78.19% 61.21s	65.43% 59.30s	60.69% 56.62s	50.07% 43.55s
MissForest	85.10% 63.77s	80.10% 375.04s	73.11% 422.73s	68.26% 523.68s	50.01% 416.57s
GAIN	85.29% 765.03s	81.12% 200.37s	72.47% 189.98s	65.66% 190.57s	50.19% 207.70s
MIRACLE	85.10% 492.11s	78.15% 914.69s	63.46% 590.38s	60.59% 797.98s	50.20% 667.33s

TABLE IV
INFORMATION GAIN SELECTION

Imputation Method	**Percentage of Missing data**				
	0 %	**10 %**	**30 %**	**50 %**	**99 %**
Mean	84.64% 29.90s	81.90% 28.44s	72.62% 28.40s	65.07% 28.34s	50.06% 28.44s
Median	84.64% 28.17	81.27% 27.92s	72.71% 27.50s	64.43% 27.62s	50.05% 27.18s
kNN	85.37% 84.95s	83.58% 210.80s	74.97% 365.50s	65.16% 521.78s	50.11% 235.75s
Soft imputation	84.66% 454.81s	80.02% 475.15s	73.54% 484.53s	67.11% 493.82s	50.24% 408.50s
MICE	84.64% 41.82s	79.22% 44.86s	60.75% 45.40s	59.83% 45.52s	50.18% 47.66s
MissForest	84.64% 29.04s	80.87% 460.68s	73.27% 448.30s	66.73% 624.57s	50.02% 433.40s
GAIN	85.99% 188.31s	81.02% 188.46s	74.35% 188.43s	62.94% 189.16s	50.17% 188.20s
MIRACLE	84.64% 209.40s	78.73% 635.08s	70.75% 712.53s	59.94% 693.90s	49.67% 721.99s

X. Appendix C different values of sparsity on fraud with median imputation

Missing data %	epsilon	zeta	accuracy	time
0.5	5	0.2	47.60	1552.03
0.3	5	0.2	49.10	1685.14
0.3	9	0.2	74.41	1797.91
0.5	9	0.2	50.45	1790.58
0.3	17	0.2	69.99	2848.88
0.5	17	0.2	66.65	2968.80
0.3	13	0.2	70.21	2543.13
0.5	13	0.2	64.02	2499.57
0.3	13	0.8	49.38	2405.45
0.5	13	0.8	48.70	2364.68
0.3	3	0.8	49.04	1461.64
0.5	3	0.8	48.75	1553.15
0.3	3	0.5	49.02	1460.22
0.5	3	0.5	48.51	1405.72
0.3	2	0.1	48.42	1424.77
0.5	2	0.1	48.22	1440.16
0.3	9	0.8	49.72	2210.76
0.5	9	0.8	55.52	2109.54