Effects of Imputation Strategy on Genetic Algorithms and Neural Networks on a Binary Classification Problem

Esteban Segarra Martinez
University of Central Florida
Orlando, FL, USA
estebansegarra@knights.ucf.edu

Stephen V. Maldonado
University of Central Florida
Orlando, FL, USA
maldonado527@knights.ucf.edu

Annie S. Wu
University of Central Florida
Orlando, FL, USA
aswu@cs.ucf.edu

Ryan P. McMahan
University of Central Florida
Orlando, FL, USA
rpm@ucf.edu

Xinliang Liu
Lehigh University
Bethlehem, PA, USA
xilc21@lehigh.edu

Blake Oakley
University of Central Florida
Orlando, FL, USA
brobersono@Knights.ucf.edu

ABSTRACT

In this paper, we compare the performance of a canonical genetic algorithm (CGA), the Self Adaptive Genetic Algorithm (SAGA), and a feed-forward neural network (FFNN) on a predictive modeling problem with incomplete data. Predictive modeling involves learning relationships between the features and labels of the data points in a dataset. Datasets with missing input values may cause problems for some learning algorithms by biasing the learned models. Imputation refers to techniques for replacing missing data through methods such as statistical probabilities, multivariate analysis, machine learning, or K-nearest neighbors.

We study how imputed datasets impact the ability for CGA, SAGA, and FFNN to learn effective models. Results indicate that imputation method has little effect on CGA and SAGA performance and a noticeable effect on FFNN performance. All three algorithms perform similarly when applied to data imputed by univariate strategies, but FFNN is noticeably worse on data imputed by trained multivariate strategies. With increased quantities of imputed data, test accuracy decreases for all three algorithms while control accuracy remains surprisingly stable in all cases except for FFNN on trained multivariate imputation. Interestingly, CGA and SAGA identify the most relevant input values, even when a large amount of the data is imputed.

KEYWORDS

evolutionary computation, neural networks, real-world applications, data imputation, genetic algorithm

ACM Reference Format:
Esteban Segarra Martinez, Stephen V. Maldonado, Annie S. Wu, Ryan P. McMahan, Xinliang Liu, and Blake Oakley. 2022. Effects of Imputation Strategy on Genetic Algorithms and Neural Networks on a Binary Classification Problem. In *Genetic and Evolutionary Computation Conference (GECCO '22), July 9–13, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3512290.3528863

GECCO '22, July 9–13, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9237-2/22/07…$15.00
https://doi.org/10.1145/3512290.3528863

1 INTRODUCTION

In this work, we compare the abilities of linear prediction models optimized by a self-adaptive genetic algorithm (SAGA) and a canonical genetic algorithm (CGA) to a feed-forward neural network (FFNN) on a binary classification task given different amounts of imputed data and data imputation strategies. The task is to predict whether the total Medicare standardized payment amount received by a physical therapist (PT) is above the median amount [8, 14]. The CGA, the SAGA, and the FFNN are given the same set of training data consisting of labelled data points and a test set to verify performance on data points not seen during training.

In recent years, the importance of data integrity has increased due to the challenging problems faced in big data and prediction modeling, particularly when datasets have missing values. When working with data like medical information, there is a chance that some data may be unfilled or unavailable. Many algorithms used for data analysis and data prediction require completed datasets. This means we need to either rework these algorithms or find a way to complete these datasets. We review the latter, utilizing imputation strategies to infer values to fill in the gaps and study how the different algorithms perform.

Data imputation is the process of repairing missing data in a dataset through connections related to other data in the dataset [3]. An early version of data imputation is multivariate imputation by chained equations (MICE) [13], which imputes data through a chained equation and a seven-choice strategy which depends on a per-value observation. A later variation of this is the DataWig [1] architecture, which builds on top of the MICE architecture but enhances it by accepting multiple types of data types and including imputation strategies that includes deep learning of the dataset. Additional deep-learning techniques have been developed, including deep-generative modeling [7, 15] and autoencoders [5].

We start by analyzing the similarity between the ground truth data and the imputed data created by different imputation strategies using loss functions and comparing the change in independent variables' statistics like median, mean, and standard deviation. We then examine how the prediction models react to different imputation strategies and to growing amounts of imputed data by reviewing

training and test accuracy on both an imputed dataset and the original dataset.

We also examine the weights of the linear models generated by the CGA and SAGA. Norat et al. [8, 14] claim that the CGA and SAGA can identify the relevant input variables in a predictive modelling problem. We investigate whether such an ability persists in datasets with increasing amounts of imputed data.

We test the impact of different imputation strategies on a spectrum of missing data to see the impact imputation strategy selection has on the CGA, SAGA, and FFNN. Our results provide guidance as to which imputation strategies are best to pair with the CGA, SAGA, or FFNN given a percent of missing data.

2 METHODOLOGY

Our primary dataset consists of 40,662 data points split into different sets for training and testing. Each data point consists of 25 features as labelled in Table 1, along with a binary label of whether or not the data point represents a PT office that receives more than the median standardized payment amount.

The dataset comes from the 2014 Medicare Provider Utilization and Payment Data: Physician and Other Supplier Public Use File (PUF) and 2015 - 2016 Area Health Resources File from the 50 states and District of Columbia. The data is sourced from collected Medicare Part B non-institutionalized claims from PUF that are representative of the healthcare market where a healthcare practician is located. For additional details in the distribution of values in the dataset please refer to Norat [8].

The CGA, SAGA, and the FFNN utilize the same training and testing dataset with 20,331 data points for training and 10,016 data points for testing. In addition to the training and test data points, the FFNN is provided a validation set of 10,016 data points for hyper-parameter tuning. The control dataset is the ground truth test set, meaning the values are not imputed and remain the same for the CGA, SAGA, and FFNN for every run and evaluation on true data.

To generate the missing data, we iterate through each data point and each feature in the data point and remove it with a given probability, leaving the dataset to have values missing completely at random. The data points' binary labels are not removed so the labels are never imputed. This probability is the percent of imputed data. We create imputed datasets in which the percentage of missing data ranges from 10% to 90% in increments of 10%. For each combination of missing data percentage and imputation strategy, we create ten unique imputed datasets. Imputation occurs independently of the CGA, SAGA, and FFNN on the datasets and missing values from each set are imputed using one of the nine different imputation strategies. Each set corresponds with a imputation strategy that is distinguished through different percentages of imputed data ranging from 0% to 90%.

The removal of data and the data imputation stages are independent from the training stages of the CGA, SAGA, and FFNN. This means each run has its own unique version of the dataset. Each algorithm for each of the nine imputation strategies and ten different percentages ranging from 0% to 90% of imputed data runs 10 times. In total, there are 900 pairs of training and test data sets per algorithm, and at the end of each run, we evaluate the model on the original ground truth test set which we call the control dataset.

i	Independent Variable	Type
0	Gender is female	Categorical
1	Doctor of PT degree	Categorical
2	# of HCPCS/CPT codes billed	Continuous
3	# of medicare beneficiaries served	Continuous
4	Charge to medicare allowed amt ratio	Continuous
5	Avg medicare standardized payment amt per beneficiary	Continuous
6	Proportion of physical agent	Continuous
7	% of therapeutic procedures	Continuous
8	Proxy for $ of new patients	Continuous
9	Avg age of beneficiaries	Continuous
10	Avg hierarchical condition category (HCC) risk score of beneficiaries	Continuous
11	Small practice location	Categorical
12	Mid-sized metro area practice location	Categorical
13	Non-metro area or missing	Categorical
14	Standardized medicare payment per beneficiaries	Continuous
15	Primary care physicians per 10K pop (2009), county level	Continuous
16	# of PTs per 10K pop (2009), county level	Continuous
17	Beneficiaries as a share of total pop, count level	Continuous
18	Avg age of beneficiaries, count level	Continuous
19	% of female beneficiaries, county level	Continuous
20	Avg HCC risk score of beneficiaries, county level	Continuous
21	% of medicare beneficiaries eligible for medicaid, county level	Continuous
22	Median household income, county level	Continuous
23	% persons 65+ in deep poverty, county level	Continuous
24	# of PTs serving medicare per 10K beneficiaries, county level	Continuous

Table 1: The independent variables in the medicare dataset

2.1 Imputation Strategies

Table 2 lists the imputation strategies that we investigate. Strategies include univariate strategies, which only have access to the same feature from every data point, and multivariate strategies, which have access to every feature from every data point. We will refer to the imputation strategy simply as strategy from this point on unless otherwise prefixed.

Univariate strategies include taking the mode, mean, or median of the missing feature from all data points that did not have that feature removed. For categorical data, we find the mean and median by mapping the categories to a numerical value. Two other univariate strategies for missing features use a combination of using the mode for categorical features and mean or median for continuous features

Strategy	Type	Trained
Mean	Univariate	
Continuous Mean & Categorical Mode	Univariate	
Median	Univariate	
Continuous Median & Categorical Mode	Univariate	
Mode	Univariate	
KNN [10, 12]	Multivariate	
DataWig [1]	Multivariate	X
Iterative Bayesian Ridge [9]	Multivariate	X
Iterative Linear Regression [11]	Multivariate	X

Table 2: List of applied imputation strategies, along with whether they make predictions based off multiple other variables (multivariate) or just itself from other instances (univariate), and whether they are trained methods.

Parameter Name	CGA Values	SAGA Values
Population	100	100
# Gens	200	200
Selection	Tourn	Tourn
Tournament Size	10	SA
Crossover	2-pt	SA
Crossover Rate	0.8	SA
Mutation	Uniform	SA
Mutation Rate	0.2	SA

Table 3: GA parameters used for the CGA and SAGA, where SA means the values are Self-Adapted.

(as demonstrates in Table 1). We call these strategies meanmode and medianmode. Since univariate strategies only have access to a single feature from all the datasets, the imputed values have no relationship with the other features.

More complex multivariate strategies such as SciKit Learn's Bayesian Ridge iterative data imputer [4, 9], SciKit Learn's Linear Regression iterative data imputer [4, 11], or Amazon's DataWig imputer [1] learn from all available information in a dataset to infer missing values based on available ground truth values. We will refer to these three strategies as trained multivariate strategies as they all require training. The iterative imputers start by replacing missing values with the mean of the feature. The iterative data imputer then creates a prediction model to replace missing values trained on data points where the true labels are available. The process for the Bayesian and linear regression is repeated iteratively for ten iterations, no random starting state, and all other settings default to SciKit. Felix et al.[2] shows that DataWig is highly accurate in imputing missing values. The settings we use for DataWig are the default case, we only provide the data in order to impute.

We also consider another multivariate machine learning (ML) imputation strategy, K-nearest neighbors [12], for data imputation. This strategy looks at similar data points to fill vacant spots by looking at the K data points surrounding a missing data point and averages them. We use 5 for K as it is the default K value from SciKit learn. We measure the similarity between data points by using Euclidean distance between the data points using the values that are not missing from both data points.

2.2 Genetic Algorithm Setup

We use two variations of a genetic algorithm, a canonical genetic algorithm (CGA) and the self-adaptive genetic algorithm (SAGA) developed by Norat et al. [8, 14]. Table 3 gives the CGA and SAGA parameters used in our experiments.

The CGA and the SAGA are both generational genetic algorithms. The CGA uses fixed parameters defined by the user, whereas the SAGA encodes certain parameters inside individuals on a secondary chromosome called the parameter chromosome that allows individuals to evolve its own optimal parameters in addition to evolving a solution to the problem [8]. These parameters include the crossover operator, the crossover rate, the mutation operator, the mutation rate, and tournament selection size.

Including both the CGA and the SAGA allows us to see how introducing self-adaptive parameters affect the ability for the GAs to create a good model. Parameter selection is often a deterrent for using GAs, as there are many parameters which require fine-tuning in order to have the CGA effectively solve the problem.

Mutation operators available for the SAGA to choose from include Uniform Random, Gaussian, Polynomial, and Swap, and the crossover operators available for the SAGA to choose from include Uniform, Simulated Binary, Arithmetic, Linear, Blend, Simplex, and Parent-Centric. Each operator has a value called the usage rate encoded in the SAGA's parameter chromosome. Mutation and crossover operators are selected by entering the operators in a tournament and selecting a winner based on the operators' respective usage rate for that individual, or average of the operators' respective usage rate for the parents involved in the crossover. Crossover rate and mutation rate are directly encoded as a float value between 0 and 1.

Tournament Size is dependent on the individuals in the population who have a "tournament weight" self-adaptive parameter between 0 and 1, where each tournament has a max weight of 2 [8]. As we randomly sample individuals for the tournament, we add the tournament weight parameter from the SAGA parameter chromosome towards the max weight. The tournament is full when that max weight is reached.

The fitness of the CGA and SAGA individuals is determined by the accuracy of the prediction model at correctly classifying a data point. Each individual encodes a constant and a set of weights for a weighted sum that will be the linear prediction model. The CGA chromosome and the SAGA solution chromosome consist of a list of 51 floats ranging between -1 to 1. The first float value represents the constant, which is followed by 25 values representing the weights, one per feature, and then followed by 25 float values that are mapped to an integer between one and four to be exponents applied to those weights. Equation 1 demonstrates the mapping function applied to

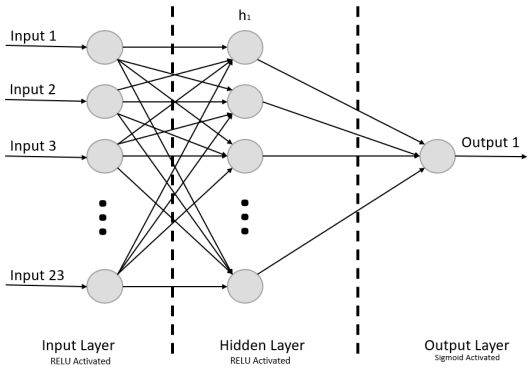

Figure 1: Neural Network Model Layout

the float exponent values to get an integer between one and four.

$$map\_exp(x) = \begin{cases} 1 : x < -0.50, \\ 2 : -0.50 \leq x < 0.00 \\ 3 : 0.00 \leq x < 0.50 \\ 4 : 0.50 \leq x \leq 1.00 \end{cases} \quad (1)$$

Equation 2 demonstrates the weighted sum calculated per data point.

$$w\_sum(x) = chromo[0] + \sum_{i}^{25} X[i] * chromo[i]^{map\_exp(chromo[i+25])} \quad (2)$$

The weighted sum is the constant plus the sum of the features multiplied by their respective weights which were raised to a corresponding exponent. The chromosome (labelled "*chromo*" in Equation 2) is the list of values that encode the solution inside an individual. If the weighted sum is above zero, then we predict the label is True, and otherwise we predict False.

2.3 Feed-Forward Neural Network Setup

We utilize a feed-forward neural network (FFNN) designed around the Medicare dataset as the neural-network approach. Figure 1 shows the basic structure of the FFNN. The FFNN uses the Keras Sequential API that creates three layers, the first one serving as input, the middle layer serving as a hidden rectified linear unit (RELU) layer with 23 nodes each input unique feature from the medicare dataset, and a final sigmoid layer to output a prediction of one or zero. Two nodes are not included in the FFNN since one column of the dataset, the practice location, needs to be expanded into two more columns for the CGA and SAGA to train a model on, which is not required for the FFNN. Table 4 gives the basic parameters of the FFNN.

The Adam optimization algorithm [6], which is currently the state-of-the-art in optimization as well as computationally fast, optimizes the weights per node in the FFNN. The binary cross entropy loss function locates and optimizes a binary classification for the features per row. The FFNN trains on 315 epochs without an early-stop condition and a batch size of 4096.

Parameter	*Note*
Neuron count	23 + 23 + 1
Layers	3 layers
Training Rate	0.001
Batch size	4096
Epochs	315
Loss Function	Binary Crossentropy
Optimization Function	Adam Optimization
Dropout	No

Table 4: Neural Network Parameters

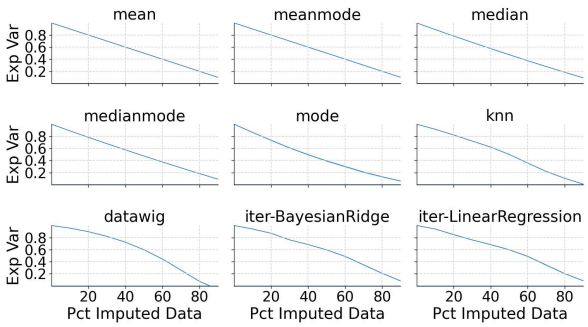

Figure 2: Explained Variance comparison between the imputed data and the original data.

3 COMPARING IMPUTED AND TRUE DATA

We provide statistics to measure the quality of the data imputation. We use loss functions to measure how similar the ground truth values are compared to the imputed values. We then look at how each strategy on average can change the median, mean, and standard deviation of the independent variables' values.

3.1 Similarity

We test several loss metrics to measure the difference between the imputed values and the original values to see how similar they are. We only review the features in this section since the labels are not removed or imputed.

Figure 2 demonstrates the explained variance (exp var) over the percent of imputed data. Each subplot is a different imputation strategy. The band around the line represents the 95% confidence interval (CI).

A higher explained variance suggests imputed values are more similar to the original values. We can see that the univariate strategies, except the mode strategy, have an almost linear decrease in explained variance with an increase in the amount of imputed data. Multivariate strategies, however, have higher explained variance, particularly with smaller amounts of imputed data. The multivariate strategies perform similar to the univariate strategy, if not worse, with larger amounts of imputed data.

Figure 3 demonstrates the mean absolute error (MAE). We can see that the univariate strategies disregarding mode are fairly linear and increasing; however, the multivariate strategies have a lower

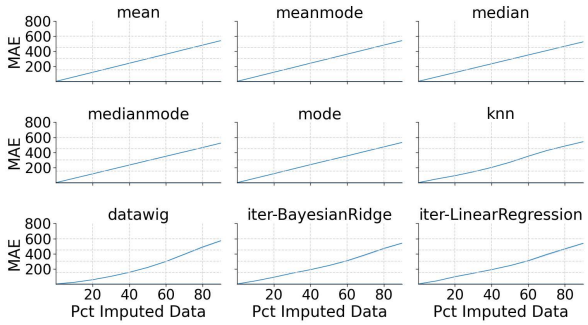

Figure 3: Mean absolute error (MAE) comparison between the imputed data and the original data.

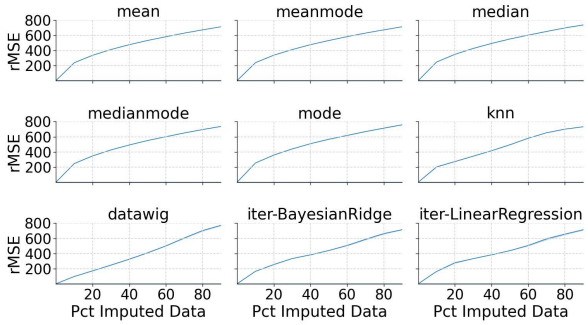

Figure 4: Root mean square error (rMSE) comparison between the imputed data and the original data.

MAE with smaller amounts of imputed data and an equal MAE with larger amounts of imputed data when compared to the univariate strategies.

Figure 4 demonstrates the root mean square error (rMSE). Similar to the MAE, the rMSE is lower for multivariate strategies with lower amounts of imputed data and equal or higher with larger amounts of imputed data. Unlike the with the MAE, the rMSE for univariate strategy is logarithmic-like, whereas the trained multivariate strategy is closer to linear.

The multivariate strategies tend to generate values more similar to the original dataset because they have the ability to predict missing features in data points based off the other features rather than simply aggregate based off the same feature from all the data points. Multivariate strategies work well with small amounts of imputed data, but as more data gets imputed due to greater loss of data, the multivariate strategies are more similar to the univariate strategies. Multivariate strategies perform worse as more data is imputed since there is less data remaining to be trained on and consequently more likely to produce an incorrect value.

3.2 Effects on Statistics

We next review the impact of imputation strategy selection on the statistics of the features. Figure 5 demonstrates the change in the median, mean, and standard deviation of the original dataset

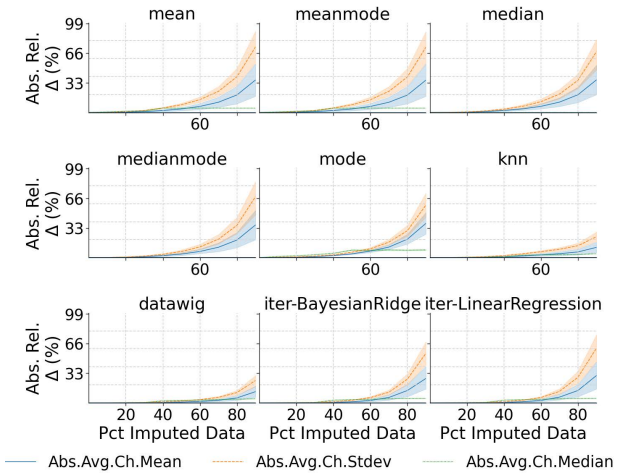

Figure 5: Average absolute relative change of the mean, median, and mode of the independent variables between the imputed data and the original data.

and the imputed dataset. We measure the absolute difference as a ratio per independent variable, then average it to understand the average relative change of the statistics of the whole dataset. Each line represents one of the different statistics, and each subgraph is a different imputation strategy.

Multivariate strategies, especially Datawig and KNN, appear to better maintain the mean, median, and standard deviation when compared to the univariate strategies, especially with lower amounts of imputed data. Having access to multiple variables allows these strategies to impute values more accurately per data point, which makes sense on why the multivariate strategies are better at imputing values that preserve the mean, median, and standard deviation of the original dataset's features.

4 EMPIRICAL RESULTS

This section details the breakdown of the performance of the CGA, SAGA, and FFNN. The breakdown consists of the training, testing, and control accuracy as the amount of imputed data increases. The control accuracy is the performance of the algorithm on the true test data with no imputations. The results per imputation strategy at each percentage of imputed data for each algorithm is an average of ten runs, each with a unique set of missing data. Finally, we provide a breakdown of the weights generated by the CGA and SAGA and how they are impacted by the amount of imputed data.

4.1 Training and Testing Accuracy

We review the algorithms' ability to train models on the dataset given different amounts of imputed data, then review the models' performance on unseen data points by testing the models on a test set with the same amount of imputed data and strategy.

4.1.1 Training Accuracy. Figure 6 plots the training accuracy with respect to the percent of imputed data. Each subplot is a different imputation strategy, and each line in the subplots represents a different algorithm (SAGA, CGA, or FFNN). The band around the

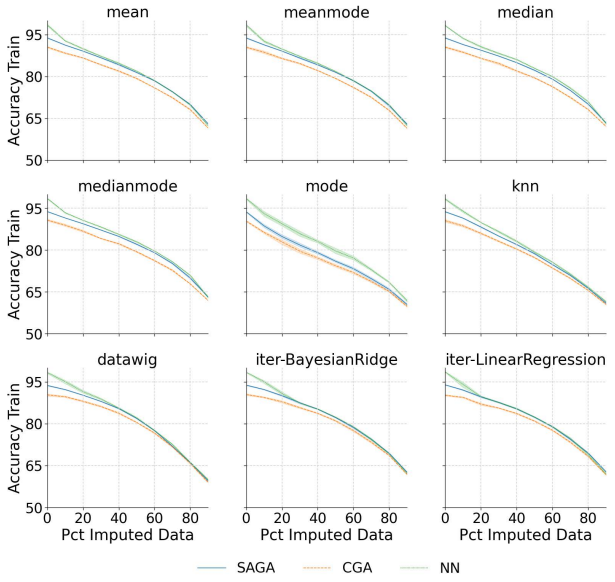

Figure 6: Training Accuracy for the SAGA, CGA, and FFNN for each imputation strategy.

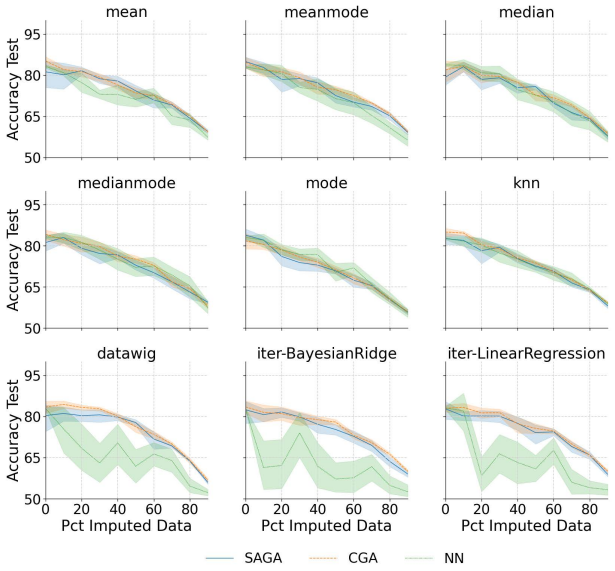

Figure 7: Testing Accuracy for the SAGA, CGA, and FFNN for each imputation strategy

lines represent the 95% confidence interval (CI). The axes' lowest value is 50% since 50% is the expected accuracy when classifying at random in a binary classification task.

Imputation strategy selection does not impact the algorithms' training accuracy significantly. The multivariate strategies allow the algorithms to have a slightly higher training accuracy than the univariate strategies. The mode univariate strategy leads to the largest difference in performance as the FFNN maintains a slightly higher training accuracy than the CGA and SAGA.

As the amount of imputed data increases, all three algorithms decline in training accuracy. The FFNN has the highest training accuracy for all the strategies with ground truth data with accuracy as high as 98%, and continues to do so with smaller amounts of imputed data. The three algorithms' training accuracies also became more similar in terms of training accuracy as the amount of imputed data increases, however the CGA always has a somewhat lower training accuracy than the other two algorithms.

4.1.2 Testing Accuracy. Figure 7 plots the test accuracy with respect to the percent of imputed data. The test accuracy represents the accuracy of the algorithms' models on unseen imputed data points.

The importance of imputation strategy selection on the FFNN becomes apparent when reviewing the test accuracy of the three algorithms. All univariate strategies and the KNN strategy seem to lead to similar results where none of the three algorithms are particularly better or worse. General FFNN performance appears to be stable across univariate and KNN strategies, including some significant outliers in mean, median, continuous median and categorical mode, mode, KKN, and Linear Regression, but has wider sway in accuracy when compared to the CGA or SAGA.

As observed in Figure 7, we can infer that for the FFNN that trained multi-variate datasets impacts the performance much more than the univariate and KNN datasets. Since the FFNN is a neural network, understanding why it performs poorly on ML imputated data is difficult since neural networks lack human-interpretability. We speculate that there are two reasons for the neural network to be impacted. The first is that ML-imputation transforms the range of data that is in the dataset - a number that would be replaced by an integer would be replaced by a double number with multiple decimal points. This transformation changes the data representation presented to the ML, which causes it to train and expect different representations when attempting to predict a number. This leads to the second problem - overfitting in predictions that produces highly inaccurate predictions. It is important to note that the GA implementations do not share this same behavior, which means that the GA is resilient against high data transformation.

All three algorithms demonstrate similar test accuracy as the amount of imputed data increases. The test accuracy for all three algorithms start around 85%, and all decrease to just below 60% with 90% imputed data if we disregard the FFNN's performance on trained multivariate algorithms.

4.2 Control Accuracy

Figure 8 plots the control accuracy with respect to the percent of data that was imputed for training. The control accuracy is what we call the accuracy of the algorithms' models on the unmodified, ground truth test set that contains no imputed values. The control accuracy represents how these models work on true data, despite training on imputed values. We find that the FFNN can actually predict more accurately on ground truth data when trained on imputed data. We also find that, despite the large amounts of imputed data, CGA and SAGA maintain a control accuracy similar to the test accuracy on ground truth data. We speculate that the CGA and SAGA are capable of maintaining a high control accuracy because

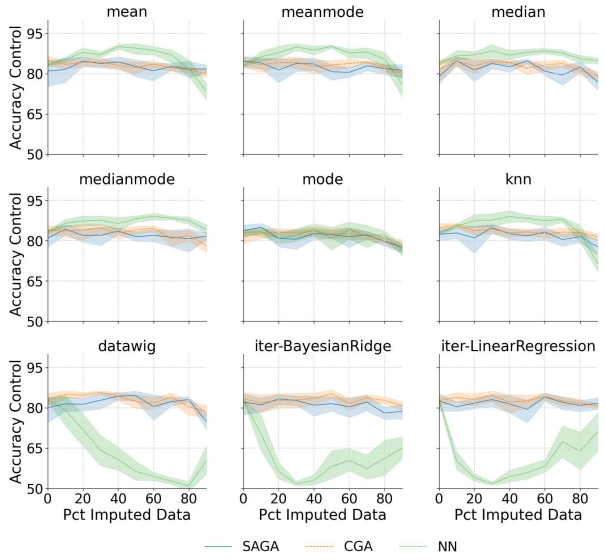

Figure 8: Control Accuracy for the SAGA, CGA, and FFNN for each imputation strategy

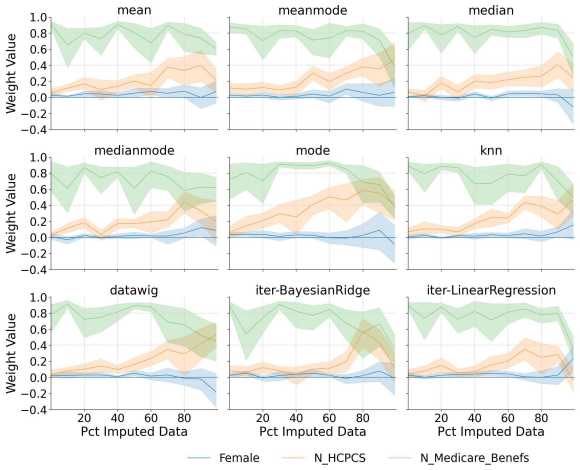

Figure 9: Weights generated by the SAGA and the CGA for "Female", "Num HCPCS", and "Num Medicare Beneficiaries" as the amount of data imputed increases.

the CGA and SAGA are still finding relevant relationships between the features and the labels of the data points.

The sensitivity of FFNN to imputation strategy observed in the testing accuracy results persists in the control accuracy results. Where the CGA and SAGA seem to maintain a control accuracy equal to the test accuracy of models training on ground truth data, the control accuracy of the FFNN varies between strategies.

The KNN strategy and the univariate strategies excluding mode tend to give the FFNN an advantage over the CGA and SAGA with medium amounts of imputed data. The FFNN actually has an increase in control accuracy with these strategies when the amount of imputed data initially increases. This improvement could be due to increasing generalization initially that occurs when random features are replaced with the univariate strategies and KNN. With moderate data loss, diversity of data points decreases and imputation will more likely impute data points that make it slightly easier for the FFNN to obtain a higher accuracy. Another factor that can improve the prediction is that since the control set has data points that are correct, it makes it easier for a trained FFNN model to correctly predict values, an observable factor that can be seen in the CGA and SAGA.

FFNN has the higher accuracy in the univariate and KNN strategies and maintains a slightly higher accuracy until a gap chance of 80%, where in mean, mode, KNN, and continuous mean and categorical mode has the bigger decreases in accuracy. We can see that similar to the test accuracy, the control accuracy is much lower for the FFNN when trained on data points imputed by the trained multivariate strategies.

As the amount of imputed data increases, the CGA and SAGA remain at slightly above 80% control accuracy, whereas the FFNN changes. As previously mentioned, the FFNN has a higher control accuracy with non-trained strategies (Univariate strategies and KNN) than they did when training on ground truth data. This is particularly true with amounts of imputed data between 20% and 80%.

4.3 Weight Extraction

One benefit of a linear model over a neural network is the ability to interpret the relationship found between the features and the labels by reviewing the weights. Seeing that the CGA and SAGA maintain a high control accuracy, we further examine the weights CGA and SAGA generate and how the imputation strategy selection and amount of imputed data impacts those weights.

The CGA and SAGA's weights demonstrate the relationship between features and the label of the data points. Weights closer to 1 and -1 demonstrate a stronger and more relevant relationship, where 1 is a positively correlated relationship and -1 is a negatively correlated relationship. Weights closer to zero are deemed to be irrelevant to predicting the label.

We find that even as the amount of imputed data increases, the CGA and SAGA are capable of finding the most relevant and most irrelevant weights; however, weights that have medium or inconsistent values in the original experiments [8, 14], change in importance as the amount of imputed data increases.

We provide three examples of weights in Figure 9, showing the weight values encoded in the best individual of the each run for the "Female", "Num HCPCS", and "Num Medicare Beneficiaries" features.

We can see that regardless of strategy selection, the most important feature and least important feature stay relatively the same up until about 80% of the data being imputed. All tested strategies keep "Female" low and "Num Medicare Beneficiaries" high up until 80% of the data is imputed.

The weight for "Female" (whether the PT is female or not) is not an important factor in determining the correct label [14], and stays irrelevant regardless of the amount of imputed data. The weight for "Number of Medicare Beneficiaries" is found to be important [14] and stays important until there is a large amounts of imputed data at

which point the weight starts to decrease. The weight for "Number of HCPCS", however, is found to have growing significance as the amount of data that was imputed increases. Wu et al. [14] and Norat et al. [8] finds that the "Number of HCPCS" had some importance, but not as significant as the features like "Number of Medicare Beneficiaries".

These results demonstrate that the CGA and SAGA are still capable of finding relevant relationships between features and labels despite large amounts of imputed data. This explains why the control accuracy remains high despite a lower training accuracy and test accuracy. Having a low training and test accuracy but high control accuracy demonstrates that these relationships may not be as applicable on imputed data, but still work on true data. It is important to note again from Figure 7 and Figure 8 and how the accuracy in FFNN is highly impacted by using multivariate trained imputation methods. This provides evidence to support our speculation that the FFNN overfits its model when training and generally results in low accuracy when predicting data from ML imputed datasets.

Using the linear models optimized by the CGA and SAGA for linear regression analysis may be less reliable with large amounts of imputed data. Most relevant and irrelevant weights remain fairly similar to the weights extracted on the original dataset, but some weights change with increasing amounts of imputed data.

5 CONCLUSION

In this study, we compare the performance of a Canonical Genetic Algorithm (CGA), a Self-Adaptive Genetic Algorithm (SAGA), and a Feed-Forward Neural Network (FFNN) on a binary classification problem given datasets with different amounts of imputed data and imputation strategies. We show the impact of data imputation strategy selection has on these algorithms and compare how they perform.

Given that many algorithms, including the algorithms in this paper, require complete datasets to build prediction models, we must have a good understanding of data imputation and its effects on prediction modeling in order to effectively use datasets that may have missing data.

We remove data completely at random in our dataset and impute the missing values using nine different imputation strategies and review the differences between the imputed data and the ground truth data. We then compare how the different algorithms are able to perform on the imputed data based on the different accuracy on the binary prediction task. We evaluate the algorithms by reviewing their ability to train a model on imputed data (training accuracy), their ability to classify data on unseen imputed data (test accuracy), and their ability to classify data on unseen ground truth data despite being trained on imputed data (control accuracy).

The CGA and SAGA both are genetic algorithms training a linear prediction model. We find the performance is similar between the two algorithms despite the SAGA having self-adaptive parameters and the CGA using fixed pre-tuned parameters. The FFNN is different as it trains on the data using neural networks, utilizing Adam to improve the learning of the algorithm and loss functions such as binary cross-entropy to predict values. The FFNN, while functionally different to an evolutionary algorithm, can obtain higher accuracy with moderate levels of imputed data in the test and control datasets.

We find that the multivariate strategies impute values more similar to the original dataset by using loss functions between the original and imputed datasets. We also find that multivariate strategies, particularly Datawig and KNN, are more effective at maintaining the mean, median, and standard deviation of the features in the dataset.

The algorithms' results demonstrate that imputation strategy selection has a minimal impact on both the CGA and SAGA; however, the FFNN has a much lower test and control accuracy when trained on data imputed by a trained multivariate strategy. This is unexpected as the multivariate strategies impute data more similarly to the original data according to our analysis. With univariate strategies, the FFNN seems to perform similar to the CGA and SAGA in terms of training and test accuracy; however, it seems to actually improve in control accuracy. Understanding the impact of imputation strategy selection on predictive modeling is important, as we have shown that different algorithms like the FFNN may perform differently given imputation strategy selection, where others like our CGA and SAGA are not as sensitive.

All three algorithms have a decrease in training and test accuracy as the amount of imputed data increases, and the difference between the accuracy of the algorithms decreases as the amount of imputed data increases.

Control accuracy, however, seems to not be affected much by the amount of imputed data. We find that, even with large amounts of imputed data, all three algorithms are capable of maintaining a control accuracy similar to the test accuracy on the ground truth data. We find these algorithms on our dataset are capable of creating a model that is just as effective on unseen true data when trained on 80% imputed data versus when trained on ground truth data assuming that the missing data is missing completely at random and there are a significant number of data points.

The CGA and SAGA have overall very similar performance with training, test, and control accuracy. Given that both algorithms are genetic algorithms, it is not surprising that their performance is similar. The SAGA has a slightly higher training accuracy than the CGA, however the difference between the SAGA and CGA in regards to test and control accuracy are not notable. Given the CGA and SAGA's similar performance, it seems advantageous to look more into self-adaptive parameters, as parameter selection is often a deterrent for using Genetic Algorithms.

Comparing the FFNN to the CGA and SAGA, we can see that the FFNN performs similar to the CGA and SAGA on unseen imputed data; however, FFNN has a higher control accuracy on moderately imputed data when not using a trained multivariate imputation strategy. The CGA and SAGA are better suited for datasets imputed with trained multivariate strategies or for whenever understanding the relationships between the features and the labels is important.

REFERENCES

[1] Felix Biessmann, Tammo Rukat, Phillipp Schmidt, Prathik Naidu, Sebastian Schelter, Andrey Taptunov, Dustin Lange, and David Salinas. 2019. DataWig: Missing Value Imputation for Tables. *Journal of Machine Learning Research* 20, 175 (2019), 1–6. http://jmlr.org/papers/v20/18-753.html

[2] Felix Biessmann, David Salinas, Sebastian Schelter, Philipp Schmidt, and Dustin Lange. 2018. "Deep" Learning for Missing Value Imputationin Tables with Non-Numerical Data. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 2017–2025.
[3] Marvin L Brown and John F Kros. 2003. Data mining and the Impact of Missing Data. *Industrial Management & Data Systems* (2003).
[4] Samuel F Buck. 1960. A Method of Estimation of Missing Values in Multivariate Data Suitable for Use with an Electronic Computer. *Journal of the Royal Statistical Society: Series B (Methodological)* 22, 2 (1960), 302–306.
[5] Lovedeep Gondara and Ke Wang. 2018. MIDA: Multiple Imputation Using Denoising Autoencoders. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 260–272.
[6] Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *ArXiv Preprint* (2014).
[7] Pierre-Alexandre Mattei and Jes Frellsen. 2019. MIWAE: Deep Generative Modelling and Imputation of Incomplete Data Sets. In *International Conference on Machine Learning*. PMLR, 4413–4423.
[8] Reamonn Norat. 2020. Improving Usability of Genetic Algorithms through Self Adaptation on Static and Dynamic Environments. *Electronic Theses and Dissertations* (2020).
[9] Sklearn. 2021. Bayesian Ridge Regression. *Scikit-Learn Developers* (2021).
[10] Sklearn. 2021. K-Nearest-Neighbors. *Scikit-Learn Developers* (2021).
[11] Sklearn. 2021. Linear Regression. *Scikit-Learn Developers* (2021).
[12] Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Botstein, and Russ B Altman. 2001. Missing value estimation methods for DNA microarrays. *Bioinformatics* 17, 6 (2001), 520–525.
[13] Stef Van Buuren and Karin Groothuis-Oudshoorn. 2011. MICE: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software* 45, 1 (2011), 1–67.
[14] Annie Wu, Xinliang Liu, and Reamonn Norat. 2019. A Genetic Algorithm Approach to Predictive Modeling of Medicare Payments to Physical Therapists, In The Thirty-Second International Flairs Conference. *The Thirty-Second International Florida Artificial Intelligence Research Society Conference (FLAIRS-32)*.
[15] Hongbao Zhang, Pengtao Xie, and Eric Xing. 2018. Missing Value Imputation Based on Deep Generative Models. *ArXiv Preprint* (2018).