# Directly Learning Predictors on Missing Data with Neural Networks

by

Alvina Awwal

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

# Abstract

The problem of missing data is omnipresent in a wide range of real-world datasets. When learning and predicting on this data with neural networks, the typical strategy is to fill-in or complete these missing values in the dataset, called impute-then-regress. Much less common is to attempt to directly learn neural networks on the missing data, without imputing; one such approach, called NeuMiss, introduces a novel layer in the network but can be finnicky to train. In this work, we explore two simple augmentations that make it simple to use standard neural network architectures: augmenting the input by concatenating a missingness indicator and introducing synthetic missingness. Synthetic missingness involves masking additional input attributes; this simple data augmentation technique expands the dataset, but surprisingly has not been explored. We show that both of these augmentations improve prediction performance across several datasets, and levels of missingness.

# Preface

This thesis is an original work by Alvina Awwal. No part of this thesis has been previously published.

# Acknowledgements

I am immensely grateful to Professor Martha White for giving me the opportunity to work with her and her team of brilliant scientists at the RLAI Lab. I have learnt a lot and developed crucial skills under her supervision, needed for both academics and the real-world, during the course of this Masters journey. Special thanks to Andrew Patterson and Lingwei Zhu for helping and guiding me whenever I needed it without any complaints or hesitation.

I will forever be indebted to my mother for giving me the strength and the courage to pursue my dreams in a foreign land, thousands of miles away from home. She was the constant push I needed to achieve this milestone in life and I would not have been able to do it without her by my side at all times.

I would also like to remember and thank my friends here in Edmonton- Najifa, Saidur, Noshin, Tahsin, Saurin, Rafi and many others, who have helped me adjust to life in Canada pretty seamlessly and made it enjoyable.

Finally, I would like to thank Alberta Machine Intelligence Institute (AMII) for providing great funding for our research.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Neural networks (NN) have proven to be very powerful and highly effective at learning complex functions and relationships, especially for high-dimensional datasets with huge amounts of data. But, most NN training procedures assume complete data, where no inputs are missing. However, in many scenarios of practical interest, curated datasets contain missing data. What that entails is that some inputs might contain missing values. The problem setting of inference from inputs with missing values is known as "missing data".

There are a number of ways that missing data have been handled. The most common approach is what is known as the two-stage approach- impute then regress. This means that the dataset is first imputed with a meaningful value using a strategy of choice and then handed over to the neural network for learning and prediction on the complete data. Some of the most used approaches for handling the incomplete dataset include single and multiple imputation methods, partial omission of the data, and complete case analysis, where machine learning models are run on an updated datasets where participants with any missing data are fully excluded [21]. These methods and how effective they are will be discussed in the next chapter in detail.

There is recent work, however, that tries to directly learn on missing data without imputing first. An example would be the NeuMiss architecture which uses a mask element on its first 4 blocks- called Neumiss blocks, which are element-wise multiplied with the neurons in order to perform an operation of layer or neuron skipping across the blocks. Although their technique is clever

and aims to directly learn on incomplete datasets, the architecture itself is quite complex, hard to optimize, and does not generalize well, as seen later in our experiments in Sections 5.1 and 5.2. This could be the reason why it has not yet been widely adopted.

In this work, we explore two simple ideas that make it easy to use any existing neural network architecture with missing data: concatenation and synthetic missingness. Concatenation is essentially stacking a binary indicator with every input vector x and passing this new stacked vector into the neural network for learning. This indicator contains an extra piece of information of whether an input feature is present (0) or missing (1). This naive method can be seen to perform quite well, irrespective of its simplicity. Synthetic missingness is adding missingness to the dataset during training in a way that the neural network learns on a new missingness pattern during every mini-batch and across all epochs. Surprisingly, though this idea is simple, it has not yet been proposed. These two methods combined give us our CatSyn algorithm which can be seen to surpass performance when compared against older common techniques, as well as newer methods and architectures in Chapter 5.

The main contributions of the thesis are as follows:

1. Introducing CatSyn, a novel simple approach to learn on missing data directly with neural networks. In particular, the idea of synthetic missingness is novel, as well as the combination of synthetic missingness with concatenation.

2. An empirical study investigating synthetic missingness and concatenation, compared to a current state-of-the-art approach to learn directly on missing data with neural networks (NeuMiss), as well as a few standard impute-then-regress approaches. We conduct experiments with different levels of missingness, conduct an ablation study to understand the role of concatenation and synthetic missingness, and design two specific studies to particularly highlight when concatenation should help and when synthetic missingness should help.

# Chapter 2

# Overview of Missing Data Strategies

In this chapter, I first introduce the problem setting and then survey existing results.

## 2.1 Problem Setting

Let $X \in \mathcal{R}^{n \times m}$ be a data matrix consisting of $n$ data points and $m$ features. The $j$-th feature of the $i$-th data point is denoted as $X\_\{i,j\}$. Data is labeled as missing when certain values/features of a dataset are not present, denoted as a mask matrix $M \in \{0,1\}^{n \times m}$.

This missingness could be due to a number of reasons which include the user not providing the values intentionally- in the case of a survey; maintenance issues such as mechanical or electronic failures during the acquisition process corrupting the data; or even human error [29].

Therefore, if $M\_\{i,j\}$ is an indicator for the $j$-th feature of the $i$-th sample, $X\_\{i,j\}$ is the observed value and finally, $X^*\_\{i,j\}$ is the 'true' value (possibly unobserved) then, missingness can be formally defined into 3 categories and those are:

- **Missing Completely at Random (MCAR)**- Missingness of data points is completely random and unrelated to both the observed and unobserved data. The probability that a value is missing is not dependent on the values themselves. There is conditional independence between M

and X.

$$\mathcal{P}(M\_\{i,j\} = 1) = \mathcal{P}(M\_\{i,j\} = 1|X\_\{i,:\})$$

- **Missing at Random (MAR)**- This occurs when there is some relationship between the values that are missing and the ones present. There is a pattern and the missingness can be explained by the variables that we have complete information on. Hence, the probability that a value is missing is dependent on the value of the observed variables.

$$\mathcal{P}(M\_\{i,j\} = 1|X\_\{i,:\})$$

- **Missing Not at Random (MNAR)**- When the missingness does not fall in the above 2 categories, it is viewed as MNAR. This is when even though there is a structure to the missingness, the observed values fail to explain it. Thus, the probability that a value is missing is dependent on the underlying true value of the variable that may itself be missing or unobserved.

$$\mathcal{P}(M\_\{i,j\} = 1|X^*\_\{i,:\})$$

Imputation is the process of filling in the missing values, given the observed variables. In our work, our focus is on the prediction part of supervised learning. Supervised learning can be thought of as an instance of imputation, in the sense that the targets are the unobserved variables that we need to infer or predict, given the other available variables. This example is an extremely specific setting since we are assuming that all the variables are present during te training phase and only missing during testing.

Usually solutions dealing with missing data problems go through the two-stage approach which includes:

**Stage 1:** Fill in the missing values.

**Stage 2:** Use the newly completed data for prediction.

However, this work focuses on a more direct method with handling missing data. Our approach and main goal is to learn a classifier that naturally handles incomplete data and can also learn to predict well with it.

## 2.2 Overview of Existing Methods

The problem of missing data is ever-present in most real-world datasets and handling this phenomenon correctly and tactfully is very crucial when working with machine learning models in order to give unbiased and optimum results. Quite a lot of research in the form of surveys and reviews have gone into details about all the approaches that have been implemented so far with this topic in various domains [7], [9]. However, our focus on this paper will be only on a subset of those, most pertinent to this work.

Papageorgiou *et al.* (2018) talks about the steps that are important when dealing with missing data in scientific research. Those are firstly, to take the necessary actions to minimize the amount of missing data wherever possible; secondly, to understand the reason behind the missingness; and finally applying the appropriate methodology to get the best possible conclusion. They later go on to mention how single imputation (SI) and complete case analysis (CCA) are the most preferred methods for handling missing data but they might not always be the best choice since they usually lead to invalid inferences [22]. Single imputation is when the features that are missing are replaced with a single value which best represents the missing value- that could be the mean; the predicted value from a regression equation; or even the value from running a kNN algorithm [37]. Similarly, complete case analysis is another popular approach for this problem wherein the rows/columns with incomplete data are completely removed in order to get a dataset which is complete [4], [24].

Among the most widely used single imputation methods, also called univariate imputation, are the following which we have also implemented in our research. In these strategies, the values that are used as replacement have been either been specified or chosen by us, such as some constant, or are some

statistical measurement of the chosen feature such as mean/median/mode.

Zero imputation- This is the method where the missing values in an input matrix are replaced with 0's since a neural network cannot work with NaN values. In this way, the dataset is completed and can somehow still be able to train and learn on the input. Even though this is a very naive approach and usually produces results that are not too accurate since 0 values do not contribute too much meaning and can also clash with natural 0s in the dataset, it is still widely used in practice due to it's simplicity in application [30], [34].

Mean imputation- This is one such method in which the mean of all the observed values of a single feature is computed and that value is then imputed for all the missing values of that feature. There are quite a few major drawbacks to this dangerously simple algorithm such as it not being able to preserve the relationship among the variables, and that it can lead to an underestimate of the standard error. However, for our case of MCAR, the estimate of the standard error preserves the mean of the observed data and it remains unbiased [11], [16].

Another popular approach is multiple imputation, which can be thought of in 3 distinct steps: imputation, analysis and pooling of results. This method is more robust and usually performs better than the previously mentioned simpler ones but it is slightly more complex and needs to be handled with more care [19]. Its sophistication comes from the fact that it generates many datasets, each with its own replacement values for the missing data, and the final imputed value is a combined average of the multiple results. This allows the uncertainty due to missingness to be appropriately considered. However, there are papers that have conflicting thoughts and observations on this particular imputation method. One paper by [10] discusses how multiple imputation is the optimal solution since it does not disregard any information while repairing the problem of systematic dropout at the same time, but is still hardly used because of the reluctance of applied researchers due to all the misconceptions surrounding this method. [15] refutes the previous paper's rebuttals and stresses on those misconceptions and states that CCA is preferable over multiple imputation because they provide unbiased results and is more efficient.

Additionally, there is also multivariate imputation (MI), where algorithms use the entire set of available features to estimate the values that are missing. The MI strategy that we have used in our research is the iterative imputation package from scikit-learn. The reason for using scikit-learn's imputer is that apart from the imputer itself, the machine learning framework provides feature transformation, data manipulation, pipelines, and machine learning algorithms which all integrate smoothly. With just a few lines of code and adjustment, we can impute, normalize, transform, and train our model on any dataset [3]. Their implementation of IterativeImputer was inspired by the R MICE package (Multivariate Imputation by Chained Equations) [31], but differs from it by returning a single imputation instead of multiple imputations.

Iterative imputer from scikit-learn, which is a MI method, works quite differently than the single or multiple imputation methods described above. For our research, we have chosen the order to be ascending, meaning the feature chosen first will be the feature with the least missing values and so on. It initially initializes the missing values with the value passed for initial strategy, which in our case was the mean. The imputer then uses an estimator, which is Bayesian Ridge by default, at each step of the round-robin imputation. During each round of the process, a particular feature column is treated as the target variable $y$ while the rest of the columns are the predictor variables $x$. The NaNs are then predicted for each feature and this process is repeated at most the number of times defined in the algorithm, which in our case is 20. But it is not necessary that it would go on for this many times since there is a built-in early stopping criterion present. This early stopping kicks in when the difference between the previous iteration prediction and the current iteration prediction is smaller than a specified threshold [23] [14].

Apart from these more known approaches, a graph-based framework, called GRAPE, has been proposed that does both feature imputation as well as label prediction using a graph representation learning. Under this particular framework, the feature imputation is formulated as an edge-level prediction task and the label prediction as a node-level prediction task. These tasks are then solved using Graph Neural Networks [36].

Another new distinct method named Generative Adversarial Imputation Nets (GAINS) by [35] uses a generative model for imputation and the most interesting difference in their paper is that they used a "Hint Matrix" when training the Discriminator. This was the only paper we found where the missingness information was somehow used in the training process of the NN.

Similarly, one more paper proposed a principled architecture- NeuMiss Networks [18], whose originality and strength comes from using a new type of mask on top of each of its layers, which is a missingness indicator vector. The paper shows that their approach gives good predictive accuracy with both the number of parameters and computational complexity while being completely independent of missing data patterns. This new architecture is based on a Neuman-series approximation of the optimal predictor. This network creates an indicator vector of [0,1] for every layer, where every layer is equal to the number of features in the dataset. This binary missingness indicator is used as the mask and element-wise multiplied in every single layer to create their novel network.

# Chapter 3

# Proposed Approach

Our problem setting is to learn directly on missing data using neural networks. We believe that we can do better by learning a direct function that inputs the available features plus extra missingness information rather than doing the basic two stage approach of imputation and handing off to the network.

In this work, we propose a simple and easy to use strategy called CatSyn algorithm, where we concatenate the missingness information and also add synthetic missingness into the dataset to help train the network and get even better results than more advanced learning approaches. What the terms concatenation and synthetic missingness mean will be explained in the following segments.

## 3.1 Concatenation Method

Concatenation of two matrices is a naive yet somehow not a very popular approach of dealing with missing data. In this algorithm, the input data is stacked with a binary vector for every $x$ with information about which features are missing, which we are calling the indicator (I). We assume that the data has been marked with the missing features using a NaN. These NaNs are then replaced with a 0, hence ultimately being a case of zero-imputation. This choice was explicitly made so that we are essentially not computing those connections and are instead dropping them for the feature values that are missing. But, before the imputation takes place, the NaNs are also used to detect which values are missing and this knowledge is used in creating the

indicator vector. The indicator is binary since it contains information of which features are missing (1) and which are not (0). In this way, the neural network gets two pieces of information at once inside its network from which it can then learn and train better conditioned on this extra information. This can be better visualized with the help of Figure 3.1.

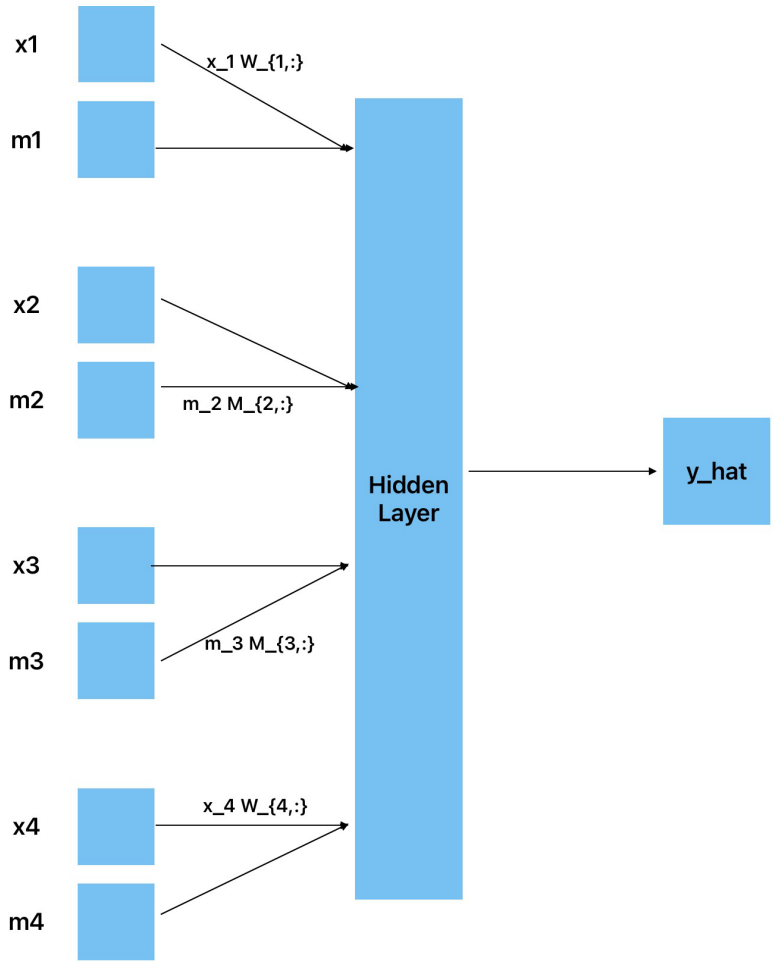


Figure 3.1: A simple diagram of how our Concatenation (Cat) method works

In order to get an insight of how the indicator is working, lets assume that we have a weight matrix $W$ and another one $M$. For every feature, there are 2 values- the original feature input $x\_i$ and its corresponding indicator $m\_i$. If the feature is available, then the indicator is 0 and the weight $W$ of the given feature is being used. Otherwise, since the input value is 0 itself, that

multiplied with a weight $W$ will also amount to 0 and hence the indicator 1 is multiplied with another weight matrix $M$.

Therefore, it is either:

$$(X\_i * W\_ij)$$

if available

or

$$(I\_i * M\_ij)$$

if missing

## 3.2 Synthetic Missingness

Synthetic data is data that has been artificially created instead of acquired from real-world events. They are created algorithmically and used for more robust training of the neural networks and machine learning models. They are hugely beneficial since they help create a diverse and large amount of training data without having to spend massive amounts of money and time [33].

Similar to the above meaning, in our synthetic missingness scenario, we have introduced a function that takes in the training data in batches and introduces an extra artificial missingness of a fixed percentage for every batch. This is repeated for every epoch during the training phase. In this way, the model trains on different patterns of missingness for every run, enabling it to learn on a more diverse training data, leading the model to be more robust and for the learning to be more diverse and thorough.

This can be thought of as a data augmentation strategy where we are adding a fixed percentage of artificial missingness to our data sample every time during its training phase, thus creating and learning on new datasets repeatedly.

**Algorithm 1** Update_Synthetic

**Require:** X_train_batch, y_train_batch, perc_missing

Pass *X_train_batch* and *perc_missing* into *Simulate_Missingness*

Calculate new *gradients* using a standard regression or classification *loss*

Update *parameters* according to the Adam optimizer

**Algorithm 2** Simulate_Missingness

**Require:** X_train_batch, perc_missing

**Ensure:** *perc_missing* = 0.5

For every {i,j} in X:

with 50% probability, set X_{i,j} = 0 and M_{i,j} = 1

Return X_train_new

# Chapter 4

# Experimental Design

Our primary goal in our experiments is to understand the behaviour of CatSyn across a variety of missingness levels. Therefore, for simplicity in this initial study, we take complete datasets and introduce artificial missingness, allowing us to control how much missingness we introduce. Additionally, we start with completely random missingness (MCAR). Note that we actually expect direct methods to be most effective outside the MCAR setting, but nonetheless, MCAR is a sensible first step.

Thus, for the purpose of our research, we have taken complete datasets, with only one of them having some slight natural missingness to it, and manually added in 5 fixed percentage missingness levels in order to have more control over our experiments and environment. The levels are 5, 10, 20, 40 and 80 percent so that we could visually see how the performance of our model dropped with this increasing level of missingness of total features in the dataset. The reason for us choosing greater number of missingness levels in the initial 50% was due to that assumption being more realistic. We would assume to see more real-world datasets with inherent missingness in it with levels closer to and less than 50% rather than more since then those datasets would be quite difficult to work with if the number of missing features exceeded the ones that are available. Another set of experiments had one fixed level of missingness in the dataset, which we kept at 50%, in order to compare our proposed approach against the other baselines and newer ones. This level was a choice made because if we only had to experiment with 1 level, we wanted it to be

a level where there was a 50-50 chance of a feature being present or missing. This missingness information will then be included in a a binary matrix which will be passed into the network as well during training so that the NN can learn conditioned on that extra information. This binary matrix, also called our indicator, consists of 0s and 1s where 0 means non-missing feature and 1 denotes missing.

Additionally, we have used four such complete real-world datasets for our research- the built-in toy Breastcancer dataset (Wisconsin- diagnostic) from sklearn [23] but it can also be found in the UCI repository [32], Spambase and Letter Recognition datasets from UCI Machine Learning Repository [6] [13] [26], and Pima Indians Diabetes dataset from Kaggle [28] [27].

The first Breast Cancer classification dataset consists of 569 samples in total, 212 of which are male and the rest 357 are female. This simple data has 30 features in total and the target is to identify each sample into 2 classes, either malignant or benign.

The next two UCI datasets of Spam and Letter Recognition have 4601 and 20000 samples respectively, where the former has 57 features and the latter contains 16. The target of the Spambase dataset is to classify a sample as spam or not spam whereas for the letter dataset, the final result is a classification of whether the sample is a letter between A-Z.

The final smaller Diabetes dataset taken from Kaggle consists of 768 samples and 8 features in total. It could primarily be found in the UCI ML repository but is no longer available due to permission restrictions. Several constraints were put on these instances when being chosen from a larger dataset. In particular, all samples were are female patients of 21 years of age or older and of Pima Indian heritage. The objective of this dataset is to diagnostically predict where a patient has diabetes or not. This was our only dataset which contained approximately 10% natural missingness to it when patients, either intentionally or unintentionally, left values empty when asked certain questions such as BMI, Insulin levels, etc. Those values were pre-processed as 0s in the dataset.

A summary of our classification datasets used for this research can be seen

in Table 4.1:

| Dataset | Instances | Attributes | Attribute Type |
|---|---|---|---|
| Breast Cancer | 569 | 30 | Real |
| Diabetes | 768 | 8 | Numeric |
| Spambase | 4601 | 57 | Integer, Real |
| Letter Recognition | 20000 | 16 | Integer |

Table 4.1: Dataset Summary

The research question that we are trying to answer for this thesis is if we can provide a better classifier in the presence of missing data with neural networks. The results that we got while comparing our naive Cat approach when combined with synthetic missingness was encouraging. However, a lot of deductions, decisions and experiments had to be made and run before we were able to fix on this algorithm as our final proposed approach of CatSyn.

For our architecture, the choices that were constant throughout were firstly, in order to create our model, we have used the JAX API [5] to fully customize our NN. Secondly, the layers have ReLU [1] activation function in the hidden layers and the Softmax activation [8] [25] in its final layer. Finally, Adam optimizer [17] has also been added when our datasets seemed to perform better with it. On the other hand, using mini-batches were not our initial plan while starting out with the smaller datasets of Breast cancer and Diabetes but it later seemed to enhance the performance for the slightly bigger Spambase and Letter Recognition data. Hence, mini-batch gradient descent was then used for our neural network's training and learning during our final experiments. All these design choices are also current state-of-the-art for neural network learning.

We carried out two sets of experiments- one had varied levels of 5 missingness percentages and the other had one level fixed at 50%, both sets of experiments ran with 3 step sizes for each dataset. All our algorithms that contained synthetic missingness had a fixed 50% synthetic missingness introduced to them across all experiments. This level for MCAR synthetic was chosen since it was a perfect middle-ground for having equal number of missing and available variables.

The first set of experiments ran for 10 runs each. However, since we were experimenting on 5 different missingness levels and with 3 step sizes of 0.001, 0.0001, and 0.00001, that led to a total of 150 outcome combinations. The step size for each missingness level was chosen according to the average of the last 5% of validation loss values. The learning rate that gave the lowest average validation loss for each missingness level was chosen for that missingness percentage and the accuracy values with that particular learning rate were used for the plotting of the results. This process was used for all the datasets and experiments.

Our second set of experiments for the boxplots had just one single fixed missingness level which was set at 50%. The step sizes for these experiments were also chosen the same way as the previous experiments, in an external queried way. The second set of experiments ran for 40 times each and for every single run, both the neural network (NN) initialization and the dataset splitting were done from scratch.

The network that we chose to create for this research consisted of 2 fully connected hidden layers of size 64 each. 64 as the size of the two hidden layers seemed appropriate and large enough for the datasets that we were using for our research. After seeing an increase in the loss function for a few of the experimental results when running for 2000 epochs each, we made the executive decision of adding in early stopping to our model. This early stopping was based off of the loss of a validation set after the training was completed on the entire training set in mini-batches. However, we made sure to not let early stopping kick in until the model had at-least trained for 50 epochs. This was repeated for every run. Early stopping on the validation loss helped boost the performance of quite a few of our results and improved our accuracies overall. In order to initialize our weights, we used an initializer for uniformly distributed orthogonal matrices with a scale of 0.1, and for the biases we used a constant initializer which we set to 0 [12]. Orthogonal was chosen since they typically tend to have faster convergence and are more consistent across runs [2].

The fixed missingness percentage(s) are introduced into our dataset at the

beginning so that the missingness pattern remained the same throughout and it essentially replicated a dataset that had inherent missingness in it. The missingness is created by randomly selecting features to be NaN. How many of them would be NaN would depend on the percentage that has been decided for that dataset. These NaNs are then used when creating our indicator vector where it locates the NaNs and outs a 1 in place for those features when creating the binary matrix and puts a 0 everywhere else. For our Diabetes dataset which had some 10% natural missingness to it in terms of certain features having 0 values, we made sure that our indicator took those into account and also represented those features as 1s.

Our experiments consisted of imputation methods described in Section 2.2 of this thesis: zero-imputation, mean imputation and an iterative imputation technique. Zero imputation is a single imputation process that involves replacing a NaN with zero. Mean imputation, also known as a uni-variate imputation strategy, involves replacing a NaN with the mean of a feature (column). The multivariate iterative imputation uses the strategy given by scikit-learn which imputes missing values as a function of other features in a round-robin fashion [23]. It estimates each feature from all the others that are present and hence, is a much more practical and realistic way of imputing missing features because the iterative nature of the algorithm gives a close estimate of what the value could have been as opposed to a constant guess. In addition to the details in Section 2.2, it is worthwhile to note that we specifically selected the following options- using the mean of a feature as its initial strategy, Bayesian Ridge as the estimator at each step of the round-robin imputation with maximum iterations of 20 and an early stopping criterion which kicked in if a specified threshold was met.

The novel NeuMiss architecture was also set up and run alongside our imputation strategies for comparison purposes in order to see if our proposed approach's performance would trump these other methods or not. It is essentially a neural network where its first 4 layers are called neumiss blocks and each neumiss block is a standard linear layer. Its only non-linearity comes from using a masking element in the layers. This masking element is put on

top of each layer and element-wise multiplied with the neurons and this creates an algorithm that can either do layer skipping or neuron skipping across the neumiss blocks. Their code can be found on their Github repository and we have taken help duplicating their architecture and results from there [20].

Finally, since we claimed at the start that we believe direct methods to be most effective in settings where data is not completely missing at random (MCAR), we carried out 2 more experiments where data was just missing at random (MAR), meaning that there is a relationship between the values missing and the ones present. Very few papers have tried implementing scenarios where data is MAR for their research, but we took inspiration from the methodologies of those that did and designed our MAR function. Our intuition while creating this function was that MAR often happens at extreme values of a feature. Thus, we chose a certain number of causal features from our dataset randomly, arranged the samples according to that feature and dropped out some of the other features around it in certain pre-determined percentage of samples. In this way, the missingness of the dropped out dependent features is in some way related to the extreme features for that sample.

Algorithm 3 is a simple pseudo-code to help understand our implementation of this setting.

**Algorithm 3** Simulate_feature_MAR

**Require:** X, perc_samples, perc_features
**Ensure:** $perc\_features = 0.2$
Randomly select 20% of features of X to be causal
Label the rest of the features as dependent features
$w$ is sampled from a uniform random distribution between -2 to 1
Dot product of $w$ is applied with causal indices and then values are sorted
Randomly drop out 20% of dependent features from given *perc_samples* of sorted samples
Update the indicator with missingness information in X
Return X_new

The first set of experiments ran for 5 seeds each where the MAR was fixed as a range of 5, 10, 20, 40 and 80 percent. The learning rate from the 3 given- 0.001, 0.0001, and 0.00001 were chosen according to the lowest validation loss for each missingness level. Only the Cat and simple zero-imputed NN was run

for this set since we wanted to see how effective just concatenation is under this MAR setting.

In the second set of experiments, we have carried out another MAR setting where we assumed that data is usually more complete during the training, and hence only applied 10% MAR to the training set. During deployment however, situations are more chaotic and data can usually be more lost and mismanaged, and thus MAR was set to 50% for the test set. Initially, we start off with a complete dataset with no missingness in it. Then we introduce 10 and 50 percent missingness MAR to the train and test set respectively, the same way and with the same intuition that we have done for the previous experiment. After that our algorithms are each run on the NN model where the Syn variants also have an extra 50% MCAR synthetic missingness introduced to them during training. Each of these algorithms ran for 20 seeds each.

These MAR results will be discussed in the next chapter in details along with graphs for better visualization.

# Chapter 5

# Results and Discussion

In this chapter, we go on to discuss the results that we got from our experiments and show graphs in the form of box and whisker plots and line graphs to better visualize them. We have divided the chapter into 3 sections, each with their own set of experimental results. The first part is a comparison study between our proposed algorithm CatSyn versus some of the previous approaches that have been used and defined as state-of-the-art in an MCAR setting. The second section discusses our findings in an ablation study, again under MCAR, that we created on the utility of concatenation when combined with synthetic missingness. The final section is based on an MAR setting, where we discuss the basic concatenation approach under MAR and dig further and also investigate our proposed approach CatSyn under our simulated extreme-feature based MAR.

## 5.1 Overall Results

At a glance, our proposed algorithm CatSyn can be seen to perform the best among the other previous ones when the fixed missingness level has been set at 50 percent as seen in figure 5.1. The red boxes of BreastCancer and Letter specifically, can be seen to achieve a much higher accuracy than the other algorithms, as seen in green and blue boxes. Even for the other remaining datasets, our proposed approach is drawn higher than the older approaches. These box and whisker plots are drawn against the baseline which is our simple zero imputed NN and hence, the test accuracies are relative.

A statistical test has also been carried out on these results and the paired T-test results can been found in tables 5.1 5.2 5.3 and 5.4. A paired t-test is a statistical test used to compare the means of two related groups or conditions. It's often used when the aforementioned two sets of data are somehow paired or matched and we want to determine if there is a significant different between the means of those two. The tables here show that with a p-value less than 0.0001 for all the algorithms, the results are highly unlikely to have occurred by random choice alone, thus the null hypothesis is rejected.

A box and whisker plot is essentially a graphical and standardized way of displaying the distribution of quantitative data. They can be used to identify outliers or anomalous data points; to understand the range of data; and to determine if data is skewed or not. The box itself is formed between the 25th and 75th percentiles of the data- which in our case are the test accuracies, with the median being shown as a horizontal line across the middle of the rectangle. The two extended lines from this box, called whiskers, are basically the minimum and maximum values of the inter-quartile range and anything outside of this range are called outliers which are shown as diamonds.

| **CatSyn VS** | **P value** | **Mean Difference** | **Conclusion** |
|---|---|---|---|
| NN | $< 0.0001$ | 0.155 | statistically significant |
| Iterative | $< 0.0001$ | 0.060 | statistically significant |
| Neumiss | $< 0.0001$ | 0.355 | statistically significant |

Table 5.1: Paired T-tests between CatSyn and older approaches on the Breast-Cancer dataset. Mean difference is CatSyn accuracy minus algorithm accuracy.

| **CatSyn VS** | **P value** | **Mean Difference** | **Conclusion** |
|---|---|---|---|
| NN | $< 0.0001$ | 0.032 | statistically significant |
| Iterative | 0.0009 | 0.021 | statistically significant |
| Neumiss | $< 0.0001$ | 0.036 | statistically significant |

Table 5.2: Paired T-tests between CatSyn and older approaches on the Diabetes dataset. Mean difference is CatSyn accuracy minus algorithm accuracy.

Quite a few different kinds of experiments were carried out with the 4 classification datasets mentioned throughout this thesis. At first, our main target was looking at how all the different algorithms performed when compared to

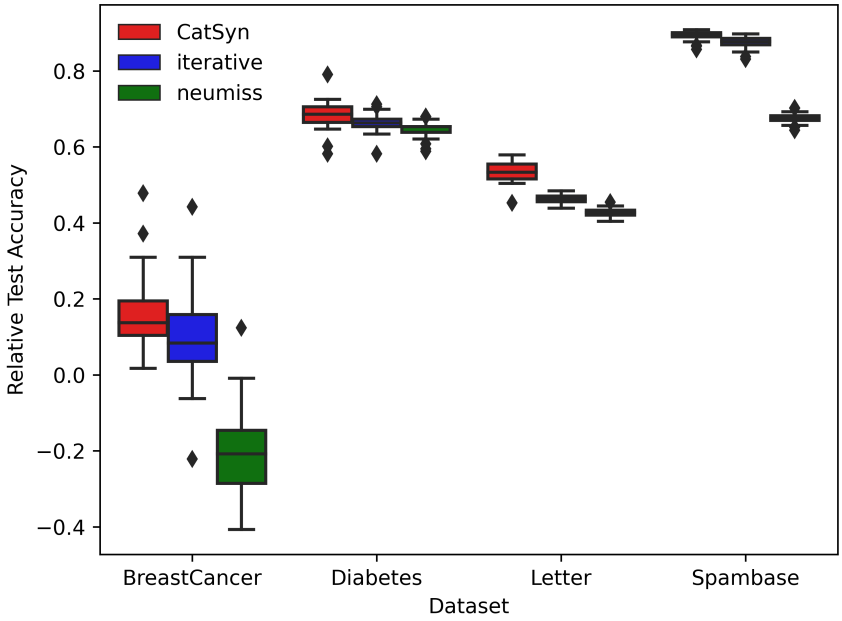


Figure 5.1: Comparison Study of our proposed algorithm CatSyn against previous methods. The datasets all have 50% fixed missingness in them. The box plots are drawn against the baseline which is our simple zero-imputed NN. CatSyn can be seen to outperform the rest of the methods.

| **CatSyn VS** | **P value** | **Mean Difference** | **Conclusion** |
|---|---|---|---|
| NN | $< 0.0001$ | 0.014 | statistically significant |
| Iterative | $< 0.0001$ | 0.018 | statistically significant |
| Neumiss | $< 0.0001$ | 0.217 | statistically significant |

Table 5.3: Paired T-tests between CatSyn and older approaches on the Spambase dataset. Mean difference is CatSyn accuracy minus algorithm accuracy.

| **CatSyn VS** | **P value** | **Mean Difference** | **Conclusion** |
|---|---|---|---|
| NN | $< 0.0001$ | 0.108 | statistically significant |
| Iterative | $< 0.0001$ | 0.073 | statistically significant |
| Neumiss | $< 0.0001$ | 0.107 | statistically significant |

Table 5.4: Paired T-tests between CatSyn and older approaches on the Letter dataset. Mean difference is CatSyn accuracy minus algorithm accuracy.

each other and on the same machine learning model. That included the naive zero and mean imputation approaches along with a fairly new iterative imputation and the concatenation method. Added with those four baselines was a reasonably new state-of-the-art method called NeuMiss architecture. The

mean imputation performed almost exactly the same as our iterative imputer and hence will not be visualized in the main results but can be found in the Appendix A section of this research.

The first set of experiments that we carried out was to answer our hypothesis of whether our simple CatSyn algorithm could perform better in comparison to the previous approaches of zero and iterative, which are our baselines, and the novel architecture NeuMiss. Therefore, the experiment consisted of 5 fixed missingness levels that were manually introduced into the datasets. Then, the 4 algorithms were run on the same machine learning model and the results were recorded and can be seen in figure 5.2.

The methods can be seen to often perform quite similarly, but sometimes can be quite different as well. The latter case would be for the 2 smaller datasets of BreastCancer and Diabetes. The standard errors for these graphs are much higher than the other larger two datasets of Letter and Spambase. The algorithms, except NeuMiss, train quite similarly for BreastCancer but during testing, CatSyn performs far better than the basic zero-imputed NN and scikit-learn's iterative imputer. The second smallest dataset of Diabetes, has the most differences among the algorithms during training as well as testing. This might be due to the fact that it's the only dataset among the 4 with some natural missingness to it, around 10%. However, CatSyn can still be seen to outperform all the others. That is also the case for the 2 larger datasets which train and test very closely with much overlapping but CatSyn still trumps over the rest.

Neumiss, even though claimed to be a state-of-the-art architecture for dealing with missing data, can be seen to be the most difficult to optimize among all of the datasets during both the training and testing phase.

The train accuracies for these algorithms with our proposed approach CatSyn can be visualized in figure 5.3. It can be very clearly seen from the 4 newer graphs with CatSyn that our 2 augmentations have a highly promising improvement to results.

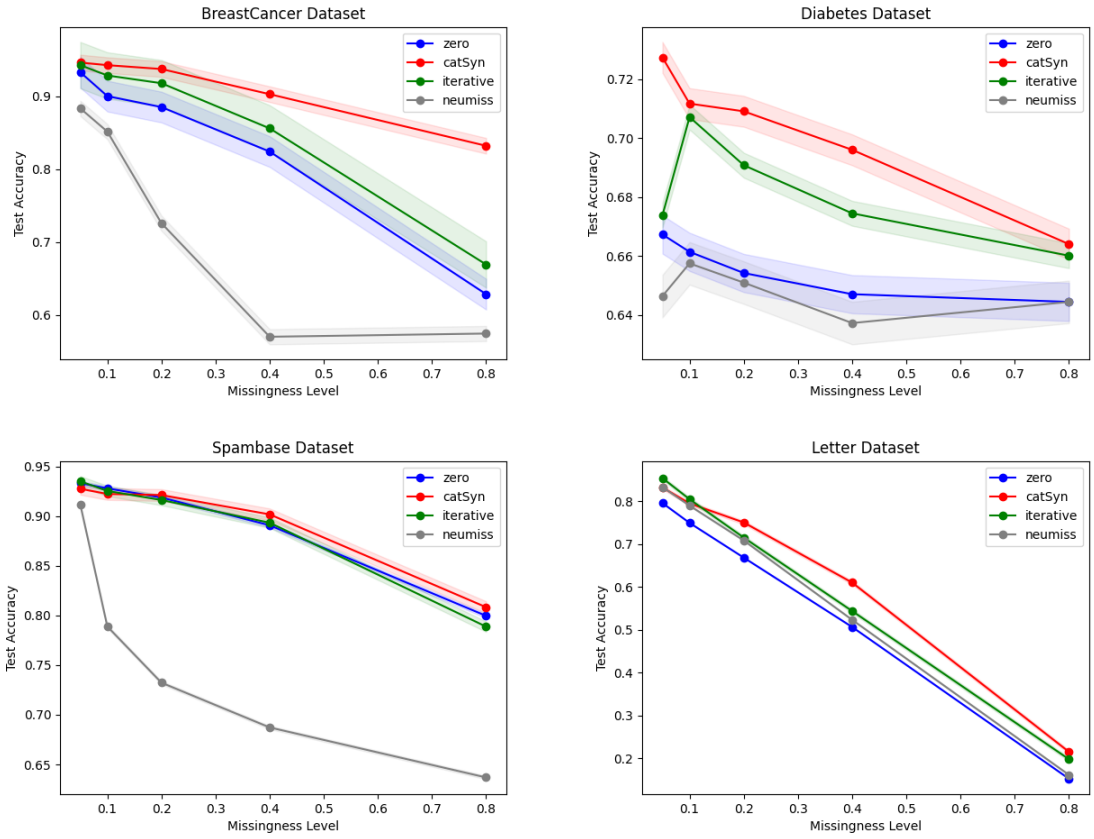


Figure 5.2: Comparison Study of our proposed algorithm CatSyn against previous approaches. These test accuracy vs missingness level graphs show that CatSyn performs far better than the older methods plus Neumiss on these four datasets.

## 5.2 Utility of Concatenation and Synthetic Missingness

Now we test if both concatenation and synthetic missingness improved performance, by doing an ablation study. We compare the vanilla NN with zero impute (NN), adding concatenation only (Cat), adding synthetic missingness only (NNSyn) and both (CatSyn). These test and train accuracy against missingness level graphs graphs can be seen in graphs 5.5 and 5.6.

These box and whisker plots show that overall adding synthetic missingness helps boost results significantly and then combining in the concatenation with that gives our algorithm that slight edge needed to outperform the other methods in 3 out of 4 datasets as seen in 5.4. Again, these plots are drawn

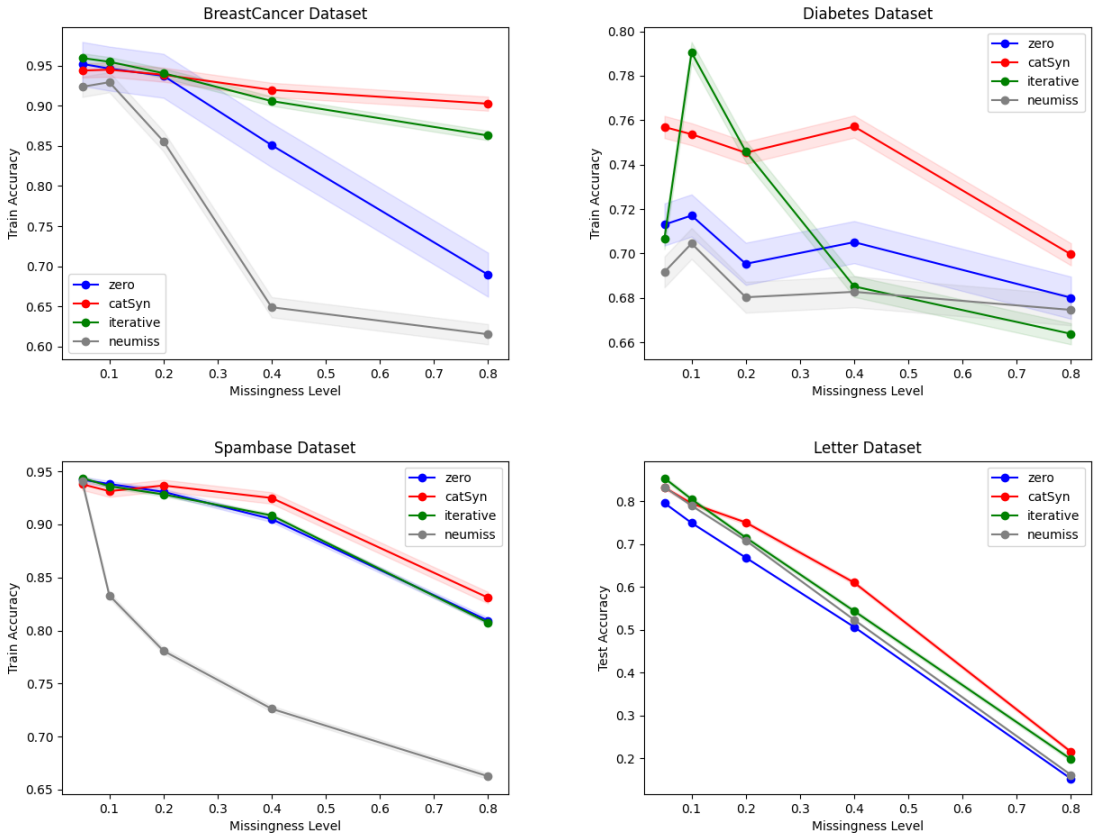


Figure 5.3: Train Accuracies of our Comparison Study with CatSyn

against the baseline which is our simple zero imputed NN.

Their corresponding paired T-tests can be found in tables 5.5 5.6 5.7 and 5.8 to prove their statistical significance. NNSyn can be seen to perform quite similarly to CatSyn in some datasets and this can be seen in two of the tables where their p-value between CatSyn and NNSyn can be seen to not be not quite and even not at all statistically significant for the BreastCancer and Diabetes datasets respectively. However, CatSyn is quite different and superior in its results with the harder and larger Spambase and Letter datasets where their statistical difference is extremely significant.

Even though the train accuracies of the first 2 datasets- BreastCancer and Diabetes, can be seen to be quite uniform, the test accuracies prove that overall synthetic missingness has the most impact in their performances. For the first row, NNSyn can be seen to perform slightly better than CatSyn for

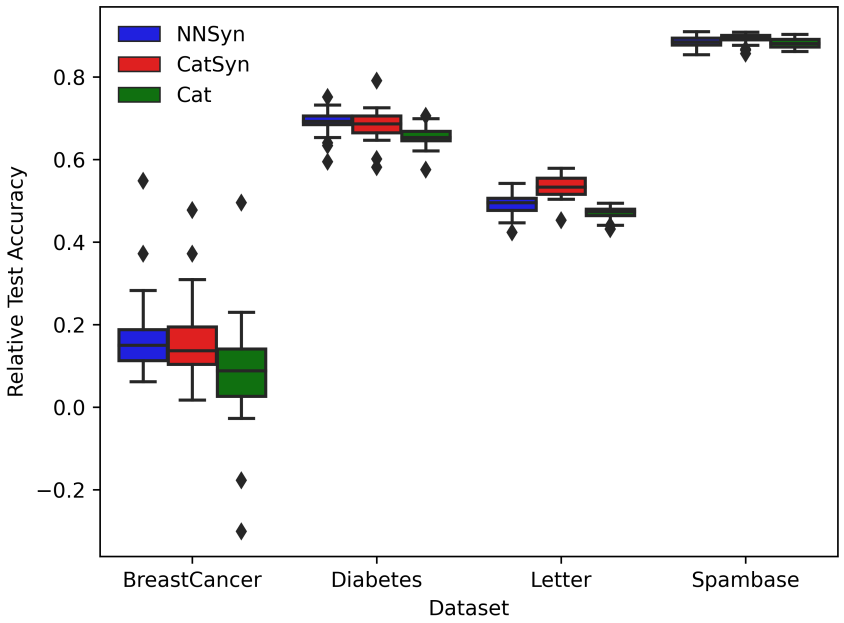


Figure 5.4: Ablation Study of our proposed algorithm CatSyn against the naive concatenation (Cat) approach and a basic synthetic variant (NNSyn). The datasets all have 50% fixed missingness in them. The box plots are drawn against the baseline which is our simple zero-imputed NN which does not contain any synthetic missingness in it. CatSyn can be seen to perform better and have a slight edge in 3 out of 4 datasets.

| **CatSyn VS** | **P value** | **Mean Difference** | **Conclusion** |
|---|---|---|---|
| NN | $< 0.0001$ | 0.155 | statistically significant |
| Cat | $< 0.0001$ | 0.073 | statistically significant |
| NNSyn | 0.0791 | -0.011 | not quite statistically significant |

Table 5.5: Paired T-tests between CatSyn and other approaches in our Ablation Study on the BreastCancer dataset. Mean difference is CatSyn accuracy minus algorithm accuracy.

| **CatSyn VS** | **P value** | **Mean Difference** | **Conclusion** |
|---|---|---|---|
| NN | $< 0.0001$ | 0.032 | statistically significant |
| Cat | $< 0.0001$ | 0.029 | statistically significant |
| NNSyn | 0.4068 | -0.006 | not statistically significant |

Table 5.6: Paired T-tests between CatSyn and other approaches in our Ablation Study on the Diabetes dataset. Mean difference is CatSyn accuracy minus algorithm accuracy.

| CatSyn VS | P value | Mean Difference | Conclusion |
|---|---|---|---|
| NN | $< 0.0001$ | 0.014 | statistically significant |
| Cat | $< 0.0001$ | 0.012 | statistically significant |
| NNSyn | 0.0003 | 0.009 | statistically significant |

Table 5.7: Paired T-tests between CatSyn and other approaches in our Ablation Study on the Spambase dataset. Mean difference is CatSyn accuracy minus algorithm accuracy.

| CatSyn VS | P value | Mean Difference | Conclusion |
|---|---|---|---|
| NN | $< 0.0001$ | 0.108 | statistically significant |
| Cat | $< 0.0001$ | 0.064 | statistically significant |
| NNSyn | $< 0.0001$ | 0.043 | statistically significant |

Table 5.8: Paired T-tests between CatSyn and other approaches in our Ablation Study on the Letter dataset. Mean difference is CatSyn accuracy minus algorithm accuracy.

most of the missingness levels. However, the standard error bars for these 2 algorithms, meaning the shaded regions for red and blue, are overlapping enough for these smaller datasets that we can infer that they are basically performing quite similarly. Moreover, for the bottom row larger datasets, all the methods seem to train and test comparably, but adding concatenation to synthetic missingness seems to very important, especially in the Letter dataset where CatSyn outperforms the rest by quite a large margin. Neumiss, once again, proves to be the hardest to train and tests the poorest.

## 5.3 Effects of Missing-at-Random (MAR) on proposed algorithms

In this section, we look at some other experiments with our most complex and largest dataset- the Letter dataset. In order to thoroughly understand the performance of our different approaches under the effects of MAR with different experimental settings in place, we carry out 2 new experiments and investigate further.

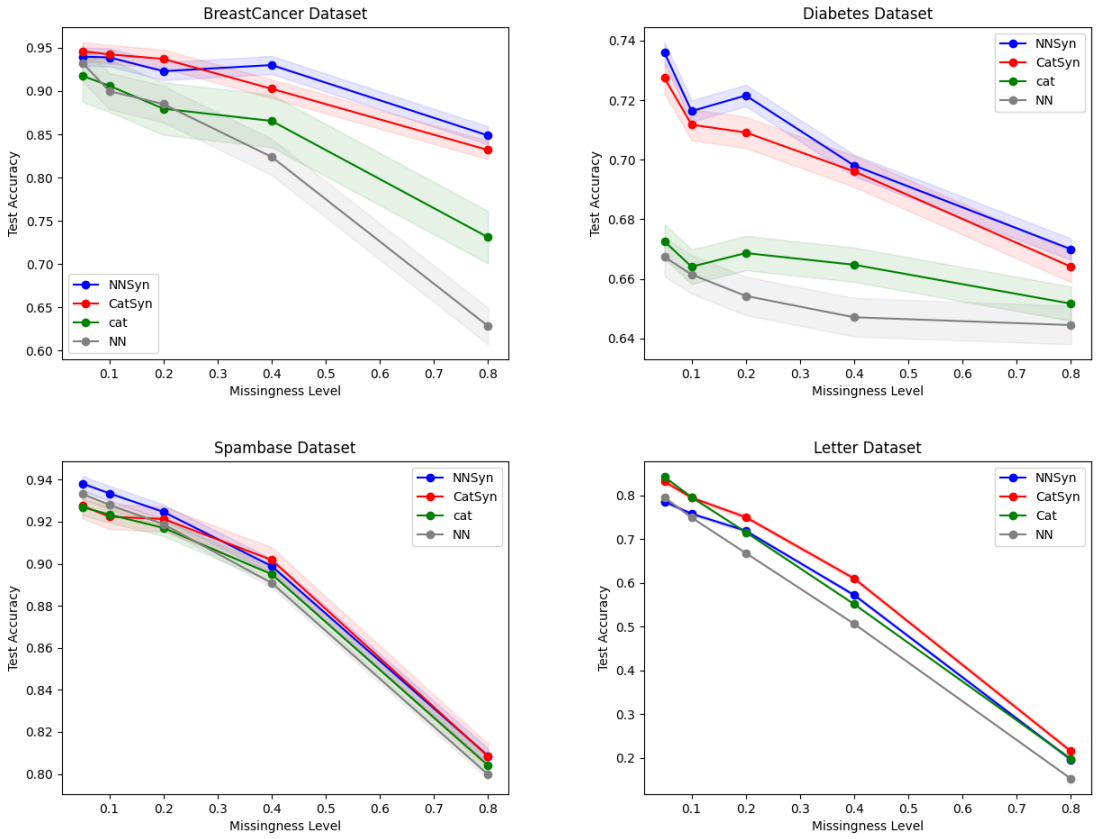


Figure 5.5: Ablation Study of our proposed algorithm CatSyn against other basic and synthetic variants. These test accuracy vs missingness level graphs prove that adding synthetic missingness to datasets overall boosts performances significantly and adding concatenation approach on top of that (CatSyn) can be seen to help improve results further for more meaningful and larger datasets.

### 5.3.1 Investigating Concatenation under MAR

Here we investigate the effects of a Missing-at-Random (MAR) setting on two of our algorithms- the zero-imputed neural network and the concatenation method. Our intuition behind this particular setting of MAR was that missing-at-random often happens at extreme values of a feature. Therefore, the missingness of the dependent features are somehow related to the causal features for each sample.

We omitted synthetic missingness from this particular experiment because our current approach introduced an MCAR synthetic missingness, and so introduces a distribution mismatch. We expect this MCAR synthetic missingness

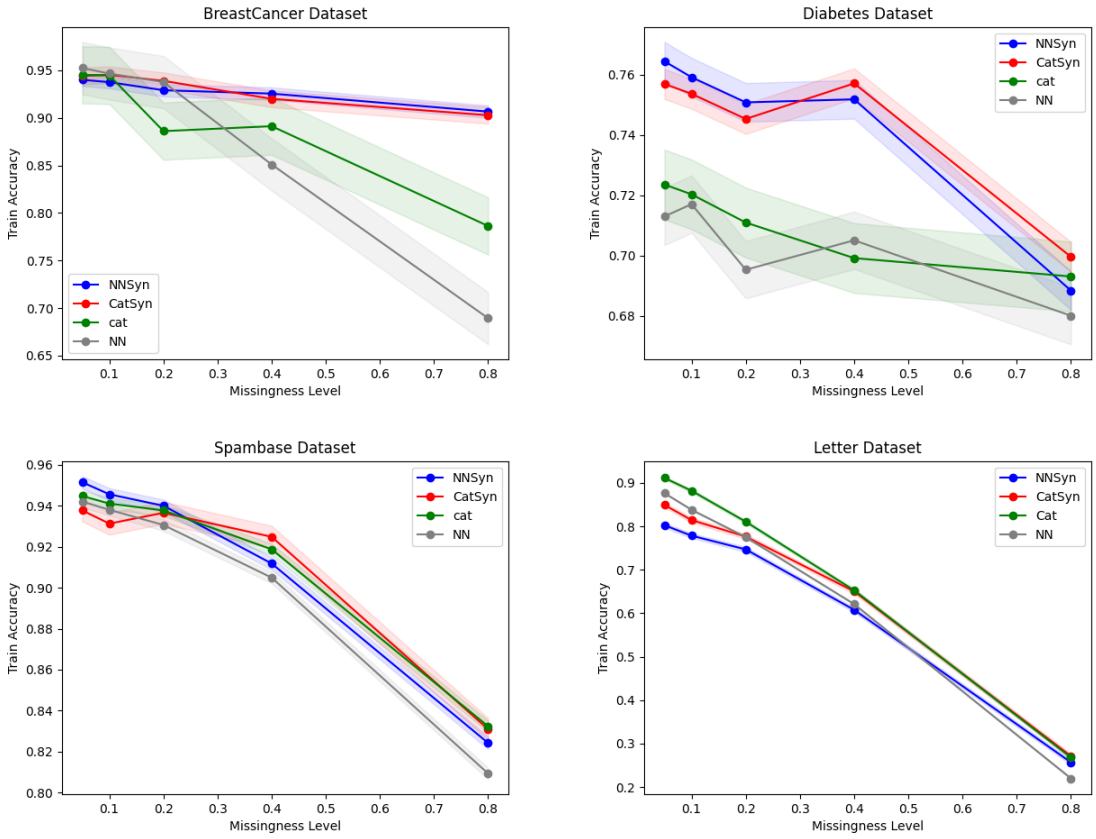


Figure 5.6: Train Accuracies of our Ablation Study with CatSyn

to do more poorly than adding no synthetic missingness to address this MAR setting.

It can be clearly seen in the graphs 5.7 that Cat performs far better at all levels of missingness than the basic zero-imputed NN. Cat strictly dominates and gets even better with higher missingness levels when data is MAR. The x-axis of the line graphs define percentage of total samples that have fixed MAR missingness introduced in them.

### 5.3.2 Investigating CatSyn under MAR

The results for these experiments are surprisingly clear and different for each of the algorithm. Even though there is a difference in data distributions here due to there being both MCAR and MAR missingness combined in the Syn variants, the difference in the MAR missingness proportions seemed to have

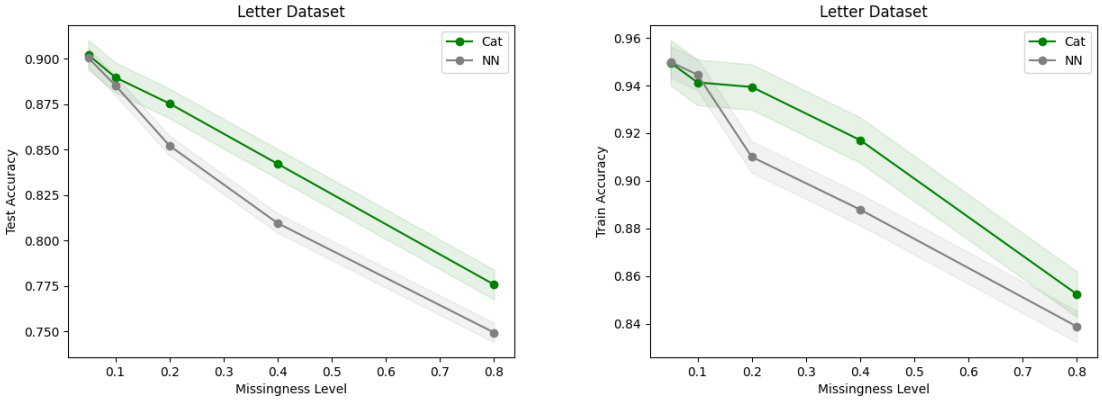


Figure 5.7: Comparison study of basic zero-imputed dataset (NN) against the naive concatenation (Cat) approach. The simple yet deceptively uncommon Cat approach seems to perform far better than the baseline under the Missing at Random (MAR) setting with increasing levels of missingness.

trumped that abnormality and overall improved results. Concatenation seems to have a big impact on performance, meaning our NN is actually learning on the missingness information well and can somewhat detect the MAR pattern. With the Syn variants, the higher MAR percentage in the test data seems to not be an issue for the neural network, since MCAR synthetic missingness patterns combined with a low fixed MAR missingness in the training data have made it susceptible to recognizing newer and higher missingness levels. The results shown in figure 5.8 are what we would've expected and proves that our proposed algorithm CatSyn can outperform other methods in more complex and meaningful missing data problems.

The statistical test results are also shown in table 5.9. All the methods are quite different from each other and when compared against CatSyn, also extremely statistically significant.

| **CatSyn VS** | **P value** | **Mean Difference** | **Conclusion** |
|---|---|---|---|
| NN | $< 0.0001$ | 0.159 | statistically significant |
| Cat | $< 0.0001$ | 0.101 | statistically significant |
| NNSyn | 0.0056 | 0.031 | statistically significant |

Table 5.9: Paired T-tests between CatSyn and other approaches in our Ablation Study on the Letter dataset under MAR. Mean difference is CatSyn accuracy minus algorithm accuracy.

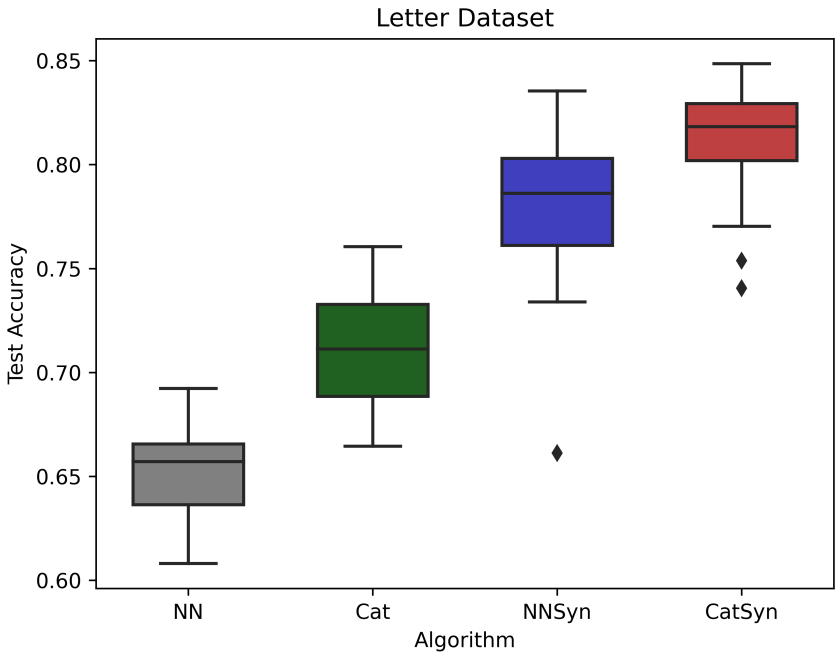


Figure 5.8: Comparison study of our proposed algorithm CatSyn against our other methods in an ablation study under the MAR setting. It is evident from the plot that both synthetic and concatenation helps improve results quite significantly when data is missing-at-random and have different missingness levels during training and deployment.

# Chapter 6

# Conclusion and Future Work

We propose a new algorithm CatSyn in this research, which works well as a direct approach to handling missing data with neural networks. Our research also focuses on understanding previous approaches that have been widely used and implemented and compares them against our own approach in both a comparison and ablation study. We have shown that not only does introducing synthetic missingness in the training data alone helps boost performance in the presence of missing data in small to large datasets, but that these surprisingly naive ideas of concatenation and synthetic missingness together also improve neural network performances significantly.

However, there are some limitations to this work that we believe can be explored further in the future. Apart from MCAR, we have only exploited the MAR setting briefly, but we believe a thorough investigation of MAR would lead to some interesting discoveries. We have also only used datasets which are more common, almost complete, and used in previous missing data works, but experimenting with more diverse and larger real-world datasets which have natural missingness in them might make this research more interesting and detailed. Future work can also focus on experimenting with different levels of synthetic missingness introduced during training, as opposed to just 50%, and looking at results across a range. The size of the neural network can also be enlarged in terms of number and size of hidden layers.

## 6.1 Ethical Implications of our Work

Working with missing data across a range of domains such as machine learning, statistics, and even general research, can raise quite a few ethical implications. Addressing these implications properly is essential to ensure that analyses and decision-making processes are fair and just while also being accurate and unbiased. To address these ethical implications, professionals working with missing data should follow established best practices, communicate openly about their methods and assumptions, and be mindful of the potential consequences of their decisions on individuals and communities.

More specifically, imputing values when working with missing data, can be concerning and be ethically unacceptable in certain situations and domains. Some examples of such domains include medical diagnosis and treatment, criminal justice and sentencing, national security and intelligence or even child welfare and protection. In such cases, the key is to thoroughly assess the ethical risks associated to the imputation, consider any alternatives, and make sure that imputation practices align with ethical principles, that it is transparent and can be accounted for. Collaborating with experts in both ethics and the domain can help navigate these sophisticated ethical considerations.

Our approach CatSyn and use of zero-imputation as the imputation strategy can help mitigate these ethical problems for quite a few of these domains since we are not essentially trying to guess any of the information that is missing and/or may be sensitive. Since our main goal is not to fill in the missing information with a meaningful value, thus essentially giving away any highly sensitive or private information, but just make predictions with whatever data we do have available, it takes away a lot of the dangers of unfair or harmful outcomes. However, this approach should still be used with a lot of caution and after careful consideration of the nature of the data, its context, and the intended use of the predictions.

# References


[1] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[2] T. W. Anderson, I. Olkin, and L. G. Underhill, "Generation of random orthogonal matrices," *SIAM Journal on Scientific and Statistical Computing*, vol. 8, no. 4, pp. 625–629, 1987.

[3] A. A. Awan, *Using scikit-learn's imputer*, Jul. 2022. [Online]. Available: `https://www.kdnuggets.com/2022/07/scikitlearn-imputer.html`.

[4] J. Bartlett, *When is complete case analysis unbiased?* Sep. 2020. [Online]. Available: `https://thestatsgeek.com/2013/07/06/when-is-complete-case-analysis-unbiased/`.

[5] J. Bradbury, R. Frostig, P. Hawkins, *et al.*, *JAX: Composable transformations of Python+NumPy programs*, version 0.3.13, 2018. [Online]. Available: `http://github.com/google/jax`.

[6] D. Dua and C. Graff, *UCI machine learning repository*, 2017. [Online]. Available: `http://archive.ics.uci.edu/ml`.

[7] T. Emmanuel, T. Maupong, D. Mpoeleng, T. Semong, B. Mphago, and O. Tabona, "A survey on missing data in machine learning," *Journal of Big Data*, vol. 8, no. 1, pp. 1–37, 2021.

[8] B. Gao and L. Pavel, "On the properties of the softmax function with application in game theory and reinforcement learning," *arXiv preprint arXiv:1704.00805*, 2017.

[9] P. J. Garcıa-Laencina, J.-L. Sancho-Gómez, and A. R. Figueiras-Vidal, "Pattern classification with missing data: A review," *Neural Computing and Applications*, vol. 19, no. 2, pp. 263–282, 2010.

[10] J. R. van Ginkel, M. Linting, R. C. Rippe, and A. van der Voort, "Rebutting existing misconceptions about multiple imputation as a method for handling missing data," *Journal of personality assessment*, vol. 102, no. 3, pp. 297–308, 2020.

[11] K. Grace-Martin, *Missing data: Two big problems with mean imputation*, Nov. 2020. [Online]. Available: `https://www.theanalysisfactor.com/mean-imputation/`.

[12] T. Hennigan, T. Cai, T. Norman, and I. Babuschkin, *Haiku: Sonnet for JAX*, version 0.0.9, 2020. [Online]. Available: `http://github.com/deepmind/dm-haiku`.

[13] M. Hopkins, E. Reeber, G. Forman, and J. Suermondt, "Spambase data set," 1999.

[14] G. Hoque, *A better way to handle missing values in your dataset: Using iterativeimputer (part i)*, Nov. 2021. [Online]. Available: `https://towardsdatascience.com/a-better-way-to-handle-missing-values-in-your-dataset-using-iterativeimputer`.

[15] R. A. Hughes, J. Heron, J. A. Sterne, and K. Tilling, "Accounting for missing data in statistical analyses: Multiple imputation is not always the answer," *International journal of epidemiology*, vol. 48, no. 4, pp. 1294–1304, 2019.

[16] J. Josse, N. Prost, E. Scornet, and G. Varoquaux, "On the consistency of supervised learning with missing values," *arXiv preprint arXiv:1902.06931*, 2019.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[18] M. Le Morvan, J. Josse, T. Moreau, E. Scornet, and G. Varoquaux, "Neumiss networks: Differentiable programming for supervised learning with missing values.," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5980–5990, 2020.

[19] P. Li, E. A. Stuart, and D. B. Allison, "Multiple imputation: A flexible tool for handling missing data," *Jama*, vol. 314, no. 18, pp. 1966–1967, 2015.

[20] m. LM and G. Varoquaux, *Marinelm/neumiss*, Sep. 2020. [Online]. Available: `https://github.com/marineLM/NeuMiss`.

[21] *Missing data*, Feb. 2022. [Online]. Available: `https://en.wikipedia.org/wiki/Missing_data`.

[22] G. Papageorgiou, S. W. Grant, J. J. Takkenberg, and M. M. Mokhles, "Statistical primer: How to deal with missing data in scientific research?" *Interactive cardiovascular and thoracic surgery*, vol. 27, no. 2, pp. 153–158, 2018.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[24] R. K. Ross, A. Breskin, and D. Westreich, "When is a complete-case approach to missing data valid? the importance of effect-measure modification," *American journal of epidemiology*, vol. 189, no. 12, pp. 1583–1589, 2020.

[25] Y. Sako, *"is the term 'softmax' driving you nuts?"* Aug. 2018. [Online]. Available: https://medium.com/@u39kun/is-the-term-softmax-driving-you-nuts-ee232ab4f6bd.

[26] D. J. Slate, "Letter recognition data set," 1991.

[27] J. Smith, J. Everhart, W. Dickson, W. Knowler, and R. Johannes, "Pima indians diabetes database," 1988.

[28] J. W. Smith, J. E. Everhart, W. Dickson, W. C. Knowler, and R. S. Johannes, "Using the adap learning algorithm to forecast the onset of diabetes mellitus," in *Proceedings of the annual symposium on computer application in medical care*, American Medical Informatics Association, 1988, p. 261.

[29] N. Tamboli, *Tackling missing value in dataset*, Oct. 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/10/handling-missing-value/.

[30] H. F. Tay, *When is it ok to impute missing values with a zero?* Aug. 2021. [Online]. Available: https://towardsdatascience.com/when-is-it-ok-to-impute-missing-values-with-a-zero-6d94b3bf1352.

[31] S. Van Buuren and K. Groothuis-Oudshoorn, "Mice: Multivariate imputation by chained equations in r," *Journal of statistical software*, vol. 45, pp. 1–67, 2011.

[32] D. W. H. Wolberg, W. N. Street, and O. L. Mangasarian, "Breast cancer wisconsin (diagnostic) data set," 1995.

[33] K. Yasar and N. Laskowski, *What is synthetic data?: Definition from techtarget*, Mar. 2023. [Online]. Available: https://www.techtarget.com/searchcio/definition/synthetic-data.

[34] J. Yi, J. Lee, K. J. Kim, S. J. Hwang, and E. Yang, "Why not to use zero imputation? correcting sparsity bias in training neural networks," *arXiv preprint arXiv:1906.00150*, 2019.

[35] J. Yoon, J. Jordon, and M. Schaar, "Gain: Missing data imputation using generative adversarial nets," in *International conference on machine learning*, PMLR, 2018, pp. 5689–5698.

[36] J. You, X. Ma, Y. Ding, M. J. Kochenderfer, and J. Leskovec, "Handling missing data with graph representation learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 19 075–19 087, 2020.

[37] Z. Zhang, "Missing data imputation: Focusing on single imputation," *Annals of Translational Medicine*, vol. 4, no. 1, 2016, ISSN: 2305-5847. [Online]. Available: https://atm.amegroups.com/article/view/8839.

# Appendix A

# Background Material

Mean imputation was also carried out for all the experiments but with the simple datasets, it was often seen to perform similarly, if not identical, to our iterative imputer and hence the results can be found here in this section of my thesis. The optimum step size for each missingness level for these results have been chosen according to the training loss.

For the comparison experiments with our proposed algorithm CatSyn and the other previous approaches, the test and train accuracies against our 5 missingness levels can be found in figures A.1 and A.2.

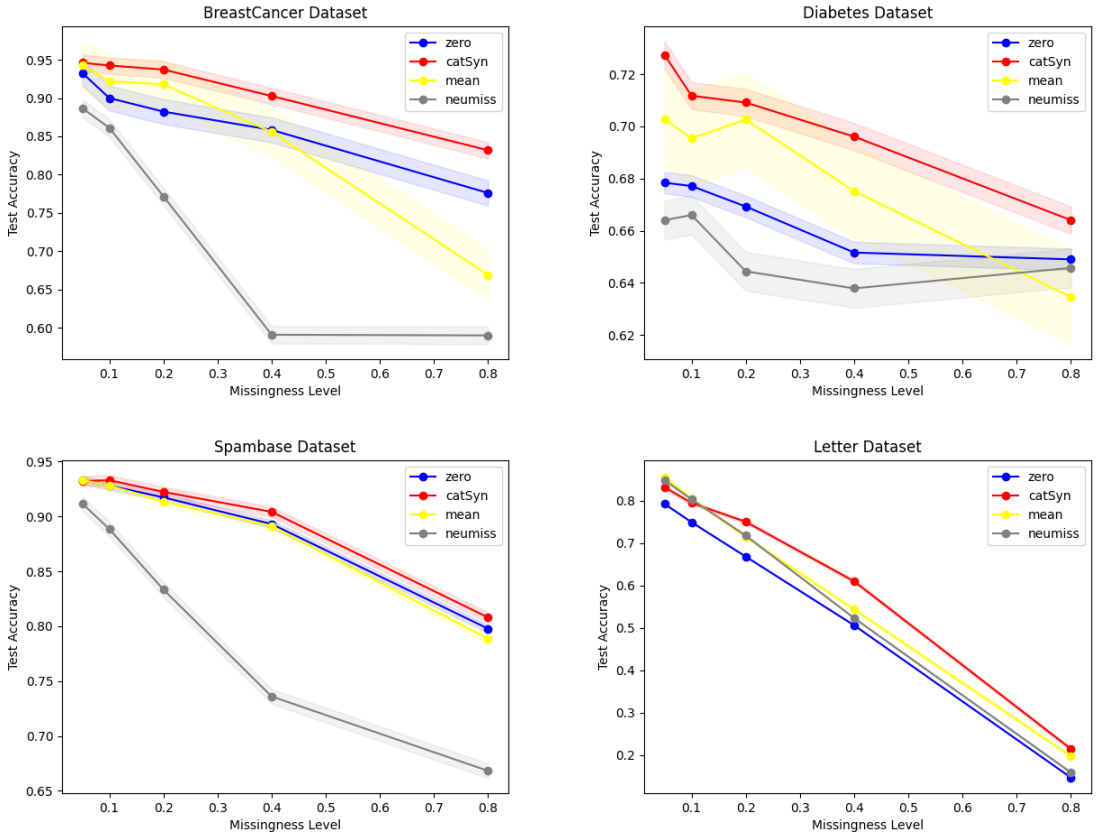


Figure A.1: This test accuracy vs missingness level graph shows how CatSyn performs against other more commonly used methods used for handling missing data. Mean imputation has been added here in place of iterative imputation since they can be seen to perform almost identically for these datasets.

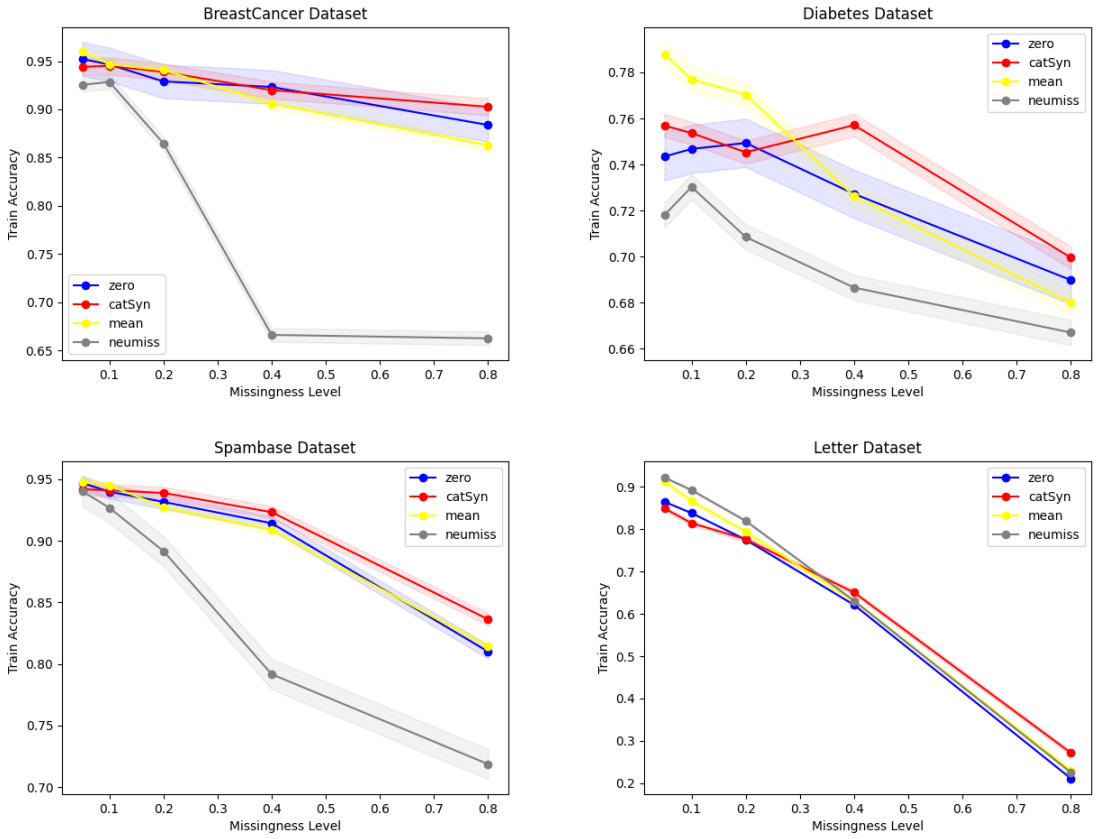


Figure A.2: Train Accuracies with our proposed algorithm CatSyn