Multiple Imputation in Machine Learning: Classifier Pooling

Lucas Snijder
l.l.snijder@student.tue.nl
Eindhoven University of Technology
Eindhoven, Netherlands

Tijs Teulings
t.teulings@student.tue.nl
Eindhoven University of Technology
Eindhoven, Netherlands

Joris Smeets
j.e.p.smeets@student.tue.nl
Eindhoven University of Technology
Eindhoven, Netherlands

Maiko Bergman
m.j.n.bergman@student.tue.nl
Eindhoven University of Technology
Eindhoven, Netherlands

Rob Janssen
r.janssen.3@student.tue.nl
Eindhoven University of Technology
Eindhoven, Netherlands

ABSTRACT

The use of multiple imputation in the context of neural networks for classification has been researched minimally. Van Buuren (2018) already suggested some applications for multiple imputation and methods on how (not) to do analysis from, specifically, multiple imputation by chained equations. One of these ways has been to pool the imputed data sets. Another approach that has seen some use cases in the machine learning domain is pooling classification results in a majority-vote bagging approach. This has traditionally been an often used approach. Our paper introduces a way to pool the classifiers resulting from multiple imputation themselves. This paper compares the performance of the three given pooling methods, for various types of missing data on three data sets. The bagging approach appears to be the best pooling method of the three and the average estimator approach appears not to be a good fit with neural networks.

KEYWORDS

Missing Data, Multiple Imputation, Pooling, Machine Learning, Classification, Neural Networks

ACM Reference Format:
Lucas Snijder, Tijs Teulings, Joris Smeets, Maiko Bergman, and Rob Janssen. 2021. Multiple Imputation in Machine Learning: Classifier Pooling. In *Proceedings of Knowledge Discovery and Data Mining Conference (Conference'28).* ACM, Eindhoven, the Netherlands, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In the field of statistics, a lot has been done to tackle the problem of missing data. However, in the domain of machine learning this is not the case. There is no clear framework on how and when to do something regarding missing data. Various approaches exist, but there is no specific gold standard such as is the case in the statistical domain. Scientists either impute data in the whole data set or impute data separately in the train and test set.

Missing data in the machine learning domain has been addressed by a few papers. For instance, single imputation, multiple imputation, and ensemble learning have been compared in this domain [24]. However, there are almost no papers on how to generate a classifier from multiple imputation specifically. Due to the generation of multiple data sets, multiple courses of action can be taken to come to a final classifier.

For instance, a model can be run over a singular data set, which has been created by averaging all data sets that were created by multiple imputation [10, 14]. However, this is not a recommended workflow, according to van Buuren (2018) [25]. He shows that the resulting data set will have incorrect standard errors, confidence intervals and p-values. This can result in incorrect statistical inferences.

Instead of averaging, the recommended and much applied suggestion is to create separate analysis for each separate data set [4, 12, 18, 22]. These resulting estimators can then be pooled to make inferences on the data [25]. For machine learning, this could be translated to training a model on each separate data set, to then pool the resulting models to generate a singular classifier.

A method that is created for machine learning and often applied in the context of missing data, is an ensemble method using bagging [11, 14, 16]. This method trains the same classifier on the different data sets that are generated by multiple imputation. Each trained classifier will then make a prediction. The resulting predictions will then be compared, and by majority vote the best prediction will be selected to be the actual prediction [2].

In this paper, the use of Multiple Imputation by Chained Equations (MICE) will be used to generate complete data sets [3]. These data sets will be used to test a multitude of ways in which neural networks can be trained. The final accuracy of each method will be compared to each other, and the accuracy of the model when using complete data.

2 THEORETICAL BACKGROUND

2.1 Missing Data

Missing data comes in different forms. Rubin identified three primary types of missing data, being: Missing Completely At Random (MCAR), Missing At Random (MAR) and Missing Not At Random (MNAR) [19]. A brief overview of MCAR and MAR are given in the following sections. MNAR is more difficult to work with, and MICE is not the best method of imputation for this kind of data. For the purposes of this paper it was chosen not to take MNAR data sets into account.

2.1.1 Missing Completely At Random. Data that is MCAR is missing independently of the data, meaning that the probability of data being missing is the same for all cases [25]. The events that lead to this missing data occur entirely at random. When data is MCAR, we can safely ignore the missing data instances for a data analysis without introducing bias. However, modelling missing data as MCAR tends to be unrealistic in practice [25].

2.1.2 Missing At Random. If the probability of data being missing is dependent on the value of some observed variables in the data, and within the classes defined by those variables the data is MCAR, then the data is said to be MAR [25]. It is a much more realistic assumption to model missing data as MAR, however it introduces some complexities in dealing with said data. Since the missing data now depends on some of the observed variables, we can now no longer simply delete this data without, potentially, affecting the outcomes of our analysis.

2.1.3 Tailed Missing Data. There exist more types of missing data than the primary types that Rubin defined. This section will add right tailed, tailed and centered to these types of missing data. In a data set where the missing data is right-tailed, most of the values that are missing would have had a higher weighted sum of scores value than average weighted sum of scores. In a data set that has centered missing data most data that is missing is around the mean (average weighted sum of scores). Tailed missing data is the opposite of centered missing data. In tailed missing data most of the missing values are far below or above the mean [25].

2.2 Imputation Methods

Depending on the type of missing data, different ways of dealing with missing data might be appropriate. A simple and very common way to handle missing data is called listwise deletion [25]. In listwise deletion, you simply delete all the rows that contain an element of missing data. If the data is MAR or MNAR, doing listwise deletion might skew the distribution of your data and introduce bias [1]. This is undesirable, hence better methods of dealing with this type of data exist. These are called imputation methods. One of the more basic forms of imputation is called mean imputation. It is also one of the most frequently used methods [1]. Mean imputation works by simply replacing the value of a missing attribute by the mean of the non-missing values of that attribute. When a data set is imputed multiple times, and the results of these imputations are averaged or pooled in some way, this is called multiple imputation [20]. One popular multiple imputation algorithm is called MICE [3].

2.2.1 MICE. Nowadays, MICE is one of the most used algorithms for dealing with missing data. It is even used as a benchmark to compare other methods for dealing with missing data [8]. MICE can handle missing values across multiple different variables. At the start, for each of these variables, a simple estimate is calculated using, for instance, mean imputation. Next, the algorithm walks through each of the variables, considering only that variable missing at each pass. Then a regression model is generated, fitting the non-missing instances of that variable as the outcome, and all the other variables in the data set as the dependents. This includes the previously imputed instances of those variables. This regression model is then used to predict the missing values of the current variable. For the next variables, these imputed instances also used in fitting the regression model. Thus, for the first variable which is considered, all the imputed dependent variables will be imputed using the same imputation method. However, for the last variable which is considered, these will all have originated from the regression models and will be more accurate. This process is repeated a number of times with parameters such that the order of variables variate. This is done to, among other things, account for the dependence on which a variable is considered first. After this process has been completed, one imputed data set has been obtained. The entire algorithm is then repeated a number of times, to obtain a number of imputed data sets. Typically, the same data analysis is then performed on all of the imputed data sets. Lastly, the results are pooled according to certain rules called Rubin's Rules. These ensure that the pooled results are better than any of the individual results [20].

2.3 Rubin's Rules

Rubin's rules are widely used in the statistical domain for combining the inferences made on the various imputed models in multiple imputation. These rules have some assumptions, most notably, the parameter to be pooled has to be distributed normally. For some other known distribution types, transformations to normally distributed quantities exist. In such cases, it suffices to transform the statistic to that quantity and then those quantities can simply be pooled using Rubin's Rules. When Rubin's Rules are used, standard errors, confidence intervals and hypothesis tests for the desired quantity are all easily calculated. This makes using these rules attractive [19, 25]. Rubin's Rules state that for normally distributed $\hat{Q}$ that are all estimated from one of the imputed data sets we have that the pooled average $\bar{Q}$ is equal to

$$\bar{Q} = \frac{1}{m}\sum_{t=1}^{m}\hat{Q}_{*t} \tag{1}$$

With the variance of this average being given by:

$$Var(\bar{Q}) = Var_{within}(\bar{Q}) + (1 + \frac{1}{m})Var_{between}(\bar{Q}) \tag{2}$$

with

$$Var_{within}(\bar{Q}) = \frac{1}{m}\sum_{i=1}^{m}Var(\hat{Q}_i) \tag{3}$$

and

$$Var_{between}(\bar{Q}) = \frac{1}{m-1}\sum_{i=1}^{m}(\hat{Q}_i - \bar{Q})^2 \tag{4}$$

2.4 Neural Networks as Classifiers

In this paper, classifiers and neural networks are mentioned. A classifier is any function that is able to, given some input data, determine a class from some set of classes to which the input data best belongs. A neural network is one way to create such a classifier. It consists of consecutive layers of nodes connected to each other via weighted functions. These weights are adapted through a learning process called gradient descent. They determine the function the neural network approximates. Normally, these are not set manually but they can be set to the weights of a different network of the same architecture, to achieve the same classification function. A neural network that is wide or deep enough can approximate any function to an arbitrary degree [13]. This includes any classification function. Neural networks are trained using supervised learning. It learns from properly classified data, and then learns to extend this knowledge to unseen data [28].

2.5 Related Works

When using multiple imputation methods, like MICE, multiple imputed data sets are created. At some point in the analysis process these separate tracks have to be merged or "pooled" into a single classification result [25]. Bagging and stacking are often used in a machine learning context, in contrast to most other imputation methods and their workflow, which have only been tested within the scope of statistics.

However, the imputation methods that might lose statistical value can still be used in the machine learning domain, as the statistical conservation is less relevant. Neural networks, as described in section 2.4, do not depend on the correctness of statistical properties like variance and confidence intervals. Moreover, the final accuracy or recall of the classification result is often regarded as the metric that matters in the machine learning context [9]. The focus on accuracy can be seen in Cao et al. (2016) [24], or recall through an F-score in Garciarena and Santana (2017) [10].

2.5.1 Bagging Approach. A common machine learning approach is to employ ensemble learning. Ensemble learning is a group of meta-algorithm techniques in which multiple classifiers are trained on separate data sets, and their classification results are pooled into a single result. This result has shown to provide improved performance [11]. In the context of missing data stacking and bagging approaches are used to pool the results that classifiers trained on separately imputed data. In a stacking setup the results of multiple classifiers are combined by a higher level (L1) model, while a bagging approach uses a majority vote to decide on the classification [2, 16]. In single classifier context, only a bagging approach is viable as all used classifiers are the same. Figure 1 shows an example of the bagging approach in a multiple imputation context.

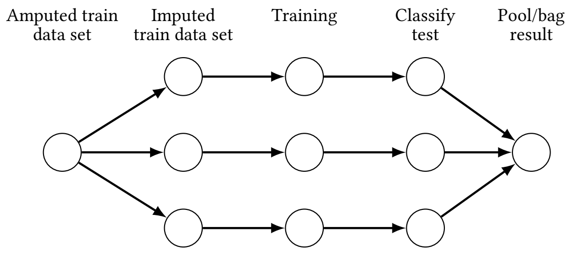

Figure 1: Bagging approach

2.5.2 Pooling Imputed Data Sets. Another option that leads to a single classifier, is to generate a single data set from the multiple data sets that multiple imputation creates. Although sometimes considered in practice [10, 14], more often than not it is heavily discouraged. Van Buuren warns novice users for this approach as combining the imputed data sets into a complete data set gives the false sense that a data set is complete after all [25]. Combining multiple data sets into one data set leads to statistical unsoundness, as any resulting analysis would suffer from an underestimation of uncertainties and therefor incorrect standard errors. Even more complex weighted approaches suffer from this unsoundness [27].

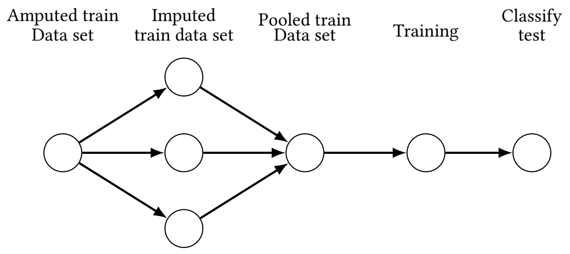

Figure 2: Pooling imputed datasets

2.5.3 Averaging Estimators. In many more recent applications it has become standard practice to use multiple imputation in an approach whereby analysis is run separately on the imputed data sets [12, 18, 22]. Only afterwards are the estimators pooled and the standard errors estimated by following Rubin's Rules, see section 2.3, thereby ensuring sadistically validity. At the same time, this is the workflow as recommended by guidelines for the MICE package [23, 25]. This approach is conventionally applied to combine both linear and logistic regression estimators [4, 12]. In estimators which lack a normal distribution, the general advise is to transform them to a normal distribution before pooling [25].

Pooling has also been applied in cases where no clear precedent was set on the pooling methodology, as is the case for model averaging approaches [18]. A more unprecedented approach is the use of multiple imputation in the context of Principle Component Analysis (PCA). As PCAs do not have estimators equivalent to regression coefficients, either the loading or correlation matrices are pooled. In the process of pooling the loadings, the signs of the loadings re-positioned as to not cancel each other out. As a result standard errors and confidence intervals cannot be calculated, however in practice these are rarely used [26].

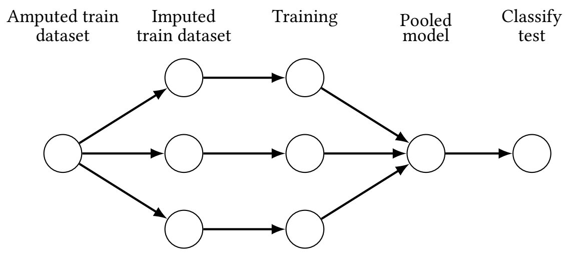

Figure 3: Average estimators approach

3 PROBLEM STATEMENT

There are several pooling methods that can be used to move from multiple imputed data sets to a single classification. As shown in figures 1, 2 and 3, one can respectively bag the classifications, pool the imputed data or pool the estimators. In the context of machine learning bagging is a well-established approach, however the other two approaches are not. Pooling the imputed data sets in a statistical context has been discouraged due to the lack of statistical soundness (see section 2.5.3). However in a machine learning context, this statistical correctness is much less relevant (see section 2.5). As such, the approach of pooling the data sets, shown in figure 2, could well result in positive results in a neural network classifier.

Similarly, pooling the estimators of neural network, as shown in figure 3, has to our knowledge not yet been tested while this has been the recommended approach with other models. As with multiple imputation in the context of PCAs there is no methodological precedence of pooling machine learning estimators through Rubin's Rules. We feel that in a neural network its weights are the most logical candidates to be seen as the estimators (see section 2.4). Thus the weight matrices would be pooled similar to Rubin's Rules like eq. 5 where W is the weight matrix corresponding to each imputed data set.

$$\bar{W} = \frac{1}{m}\sum_{t=1}^{m}\hat{W}_{*t} \tag{5}$$

However, pooling these weights may introduce statistical unsoundness. As it is not guaranteed that the weights are normally distributed, nor are there any techniques that transform them to normality, there is no certainty that Rubin's rules will hold. In the following analysis confidence intervals and p-values are not guaranteed [5, 25]. As statistical soundness, is less relevant, all three methods can not ruled out in advance on this ground, as classification accuracy is the most important metric.

Thus we compare the three pooling methods that can be used after multiple imputation through MICE. Such a comparison is novel to apply in the context of machine learning and specifically neural networks, while it tests some classic pooling methods.

4 SOLUTION APPROACH

A multitude of approaches have been mentioned to solve the given problem. Sections 2.5.1, 2.5.2 and 2.5.3 show some solutions when using MICE. These methods will be explained in more detail, and converted to proper methods for machine learning in the following subsections. The can be viewed our git repository. [1].

4.1 Creating Imputed Data Sets

4.1.1 Train test split. The complete data set is first divided in a train and a test data set. Of the complete data set 70% of the rows will be in the train data set and 30% of the rows will be in the test data set. These rows are chosen randomly. For the test set we will not create missing data. This test set is used to evaluate the performance of each of the resulting classifiers.

4.1.2 Amputing the Data Set. In the train data set missing values will be generated by the use of amputation methods. For the experiment we used a Monte Carlo approach, this will be explained later in section 5.1.2. For this purpose we repeated the creation of imputed data sets. We used the function *ampute* [21] in R that has been implemented in the *MICE* [25] package to ampute this data. This function created 10%, 20% and 50% missing data in our train data set, with the arguments *RIGHT* to create right-tailed missing data and *MID* to create centered missing data. Right-tailed missing data is tailed to one side so the right-tail (highest weighted sum) would have the most missing data, while the left-tail (lowest weighted sum) would have almost no missing data. Centered missing data would have most missing values around the average and only a few of the missing values would be around a tail. We chose to use centered and right-tailed missing data because this are two completely different types of of missing data so it could give us a broader understanding of effects of missing data.

We used the MAR as argument for the type of missing data to create MAR data sets. We have chosen not to use MCAR because it is not realistic in our approach. To have 10%, 20% and 50% of the cell missing in the data set and not missing values in 10%, 20% and 50% off the rows we need to have bycases off in the argument. In order to be able to set the bycases argument to False we needed to feed custom missing data patterns to the function. In the patterns there is at least one value in the data that is never missing in the data sets with less than six columns and at least two values that are never missing in the data sets with six or more columns. For less than six columns it is the maximum to have one value that is never missing in every pattern, because with two values that would never be missing it is not possible to reach the 50% missing cells in a normal way.

4.1.3 Imputing the Data Set. The missing values of 10%, 20% and 50% in these train data sets are filled back in with the use of the *MICE* [25] package in R. This MICE imputation method creates five different imputed data set for each train data set. We used the default arguments for MICE that use the mean as method for imputation. The pseudo-code for this method is shown in the Algorithm 1 codeblock.

Algorithm 1 Creating imputed datasets

```
1:  while $i \leq 20$ do
2:      data_set ⟵ load_dataset()
3:      create train test split
4:      test_csv<-write_csv(test split)
5:      train_missing <- ampute(train_set)
6:      impute_train <- MICE(train_missing)
7:      while $j \leq 5$ do
8:          train[j] <-complete(impute_missing,j)
9:          train_csv[i,j] <-write_csv(train[j])
10:         $j+ = 1$
11:     $i+ = 1$
12: Return ensemble_prediction
```

4.2 Classifier: Neural Network

The classifier we used is a neural network. Neural networks are a good candidate for our proposed method because it can be (re)built easily from its output. A model can be rebuilt based on the structure and a matrix of the weights. Therefore, the proposed approach of pooling classifiers can be done in a straightforward manner. We

[1]https://github.com/11036354/Research-Topics

also looked into Support Vector Machines as a second classifier, since it can also be built up from its previous output. The problem with Support Vector Machines is the over-parametrization of the output structure, the support vectors. Minor changes in the input can result in a whole different structure of these support vectors, making it hard to perform model pooling by averaging. Therefore, we decided that Support Vector Machines were not suitable for our proposed method.

The hyper-parameters of the neural network have a great effect on the prediction accuracy of the model. These hyper-parameters are set by experimenting with different combinations. A more structured approach would be to use hyper-parameter tuning [7]. However, due to computational limitations this was not possible. For the different data sets we used two fully connected layers with the following hyper-parameter settings. For the banknote data set we used a batch-size of 20 with 3 epochs. For the scale data set we used a batch-size of 32 with 10 epochs. For the user data set we used a batch-size of 3 with 25 epochs.

4.3 Pooling Imputed Data Sets

After MICE has generated five different data sets, these data sets will be averaged into an "average data set". This data set will then be used to train a neural network. This neural network is then evaluated on the test set.

4.4 Averaging Classifier

Instead of generating an average data set it is also possible to run a neural network over all generated data sets. The weight matrix coming from these networks can than be averaged into an "average model". The performance of the average model is then tested on the test set. The workflow of this method can be seen in figure 4.

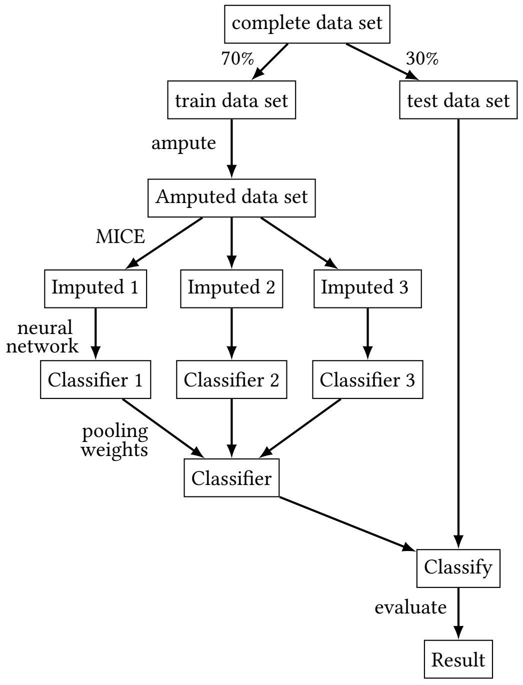

Figure 4: Workflow averaging weights neural network

The pseudo-code for this method is shown in the Algorithm 2 code-block. The result is the prediction of the pooled classifier on the test set. These results are visualized in the result section.

Algorithm 2 Pooling classifier

```
1: imputed_data_sets ← load_datasets()
2: test_dataset ← load_test_data()
3: for each imputed data set $i$ do
4:     classifier[i], weights_matrix[i] ← train_classifier_get_weights(data_set[i])
5: mean_weight_matrix ← get_mean_weight_matrix(weights_matrix)
6: pooled_model ← build_model(classifier[1], mean_weight_matrix)
7: prediction ← predict(pooled_model, test_data)
8: Return prediction
```

4.5 Bagging Approach

For the bagging approach, the algorithm first consecutively initializes and trains the classification neural networks on the various imputed data sets. Then, the test set is used to retrieve predictions for each of the imputed data sets. Then majority voting is used to get the overall prediction. This is done by taking the statistical mode for each of the predictions of all the classifiers. These statistical modes represent the outputs of the bagging ensemble. The pseudo-code for this method is shown in the Algorithm 3 code-block.

Algorithm 3 Bagging Method

```
1: imputed_data_sets ← load_datasets()
2: for each imputed data set $i$ do
3:     classifier[i] ← train_classifier(data_set[i])
4:     prediction[i] ← predict(classifier[i], test_data)
5: ensemble_prediction ← mode(predictions)
6: Return ensemble_prediction
```

5 EXPERIMENTAL RESULTS

5.1 Experimental setup

5.1.1 Reproducibility. Training a neural network involves multiple sources of randomness, such as initialization. Neural networks are highly over-parameterized, meaning that randomness causes differences between predictions of two models trained by the same algorithm on the same data. To deal with this, the random number generator of the TensorFlow package is seeded.

5.1.2 Monte Carlo Simulation. Another consequence of the over-parametrization is that the results can be widely distributed. In order to be able to draw general conclusions, some form of repeated experimentation is desirable. In this paper we have used the widely used Monte Carlo simulation technique [17]. With this approach the experiment is repeated M number of times. This repetition makes sure that the variation in the results is dealt with. For this experiment we used a M of 20.

5.1.3 Evaluation Metric. For the evaluation of the used methods we use the accuracy. Accuracy is the most widely used metric for evaluating classification models. Accuracy is the number of correct predictions divided by the total number of predictions times 100%.

5.2 Data

The User Knowledge Modeling data set uses the knowledge level of the user as classifier. The different values for this classifier are very low, low, medium and high. The five attributes in this data set are: the degree of study time for goal object materials, the degree of repetition number of user for goal object materials, the degree of study time of user for related objects with goal object, the exam performance of user for related objects with goal object and the exam performance of user for goal objects. There are no missing values in this data set and there are 403 rows. The data set banknote authentication uses if a banknote is genuine or forged as classifier. The attributes for this data set are the variance, skewness and kurtosis of Wavelet Transformed image and the entropy of image. This data set has no missing values and there are 1372 rows. In the data set balance scale each example is classified by having a balanced scale tip to the left, tip to the right or be balanced. The attributes in this data set are left weight, left distance, right weight and left distance. There are no missing values in the original data set and there are 625 rows.

Data set MD	N.Feats	N.Cases	N.Classes
Banknote	4	1375	2
Scales	4	625	3
User	5	402	5

Table 1: Description of data sets used

We used data sets that have a maximum of 5 attributes. The reason for this is that MICE and neural networks are computationally heavy and we want at least twenty iterations, to lower that randomness of our prediction, so it takes a long time to run our model.

5.3 Results

These tables show the accuracy of classification on the test data set for each of the pooling methods. The baseline represents the accuracy of a classifier trained on the complete train data set. Complete meaning the train data set before creating missing values.

	10% MD	20 % MD	50% MD	Complete data
Classifier pooling	97.82	97.39	95.85	-
Train data pooling	99.87	99.55	98.11	-
Prediction pooling	99.95	99.64	98.46	-
Baseline	-	-	-	99.96

Table 2: Accuracy of the classifier on the Banknote data set MAR MID

	10 % MD	20% MD	50% MD	Complete data
Classifier pooling	41.20	23.96	13.19	
Train data pooling	90.82	90.66	84.23	
Prediction pooling	91.28	88.14	73	
Baseline	-	-	-	91.41

Table 3: Accuracy of the classifier on the Scales data set MAR MID

	10 % MD	20% MD	50% MD	Complete data
Classifier pooling	42.93	41.07	34.96	
Train data pooling	91.78	90.66	85.41	
Prediction pooling	91.07	89.13	81.86	
Baseline	-	-	-	92.11

Table 4: Accuracy of the classifier on the User data set MAR MID

5.3.1 *Centered Missing Data.*

As one can imagine, the more missing data is present in the data set, the less accurate the classifier will be. This is true in general across all methods and data sets. Furthermore, accuracy varies wildly depending on the chosen method. The bagging method seems to be the best across the board, with the weight pooling method performing the worst. It manages to get a score of only 13% in one of the instances. As this data set consists of only three classes, simply always guessing the biggest class would have vastly outperformed this classifier. The bagging classifier gets very close to the benchmark accuracy of the data set without missing data.

	10% MD	20 % MD	50% MD	Complete data
Classifier pooling	98.30	97.77	96.57	-
Train data pooling	100	99.84	98.25	-
Prediction pooling	100	99.88	98.41	-
Baseline	-	-	-	100

Table 5: Accuracy of the classifier on the Banknote data set MAR RIGHT

	10% MD	20 % MD	50% MD	Complete data
Classifier pooling	38.75	23.43	14.76	-
Train data pooling	91.04	90.74	84.28	-
Prediction pooling	90.59	87.69	73.64	-
Baseline	-	-	-	91.57

Table 6: Accuracy of the classifier on the Scale data set MAR RIGHT

	10% MD	20 % MD	50% MD	Complete data
Classifier pooling	46.78	44.88	34.01	-
Train data pooling	91.12	90.37	84.22	-
Prediction pooling	90.87	88.97	81.57	-
Baseline	-	-	-	91.53

Table 7: Accuracy of the classifier on the User data set MAR RIGHT

5.3.2 *Right-Tailed Missing Data.*

Like the centered missing data the accuracy of the classifier of right-tailed missing data depends on the percentage of missing data in the data set. The more missing data the less accurate the classifiers are. Train data pooling and prediction pooling perform better than classifier pooling in every data set tested. Like centered missing data, the classifier pooling of 50% missing data in the scale data set performs worse than random guessing.

6 CONCLUSIONS

As is seen in section 5.3, the results for data sets and methods vary wildly. The two classical methods (data pooling and bagging) were able to achieve above the 50% threshold for accuracy for all data sets, levels of missing data, and types of missing data. Thus, as a benchmark, this should be a possible value to reach for our classifier pooling method. However, the pooling of the classifiers seemed to perform the worst on average out of all the techniques, independent of the type of missing data. It even performed far below this benchmark in some cases. This likely has to do with the fact that each of the classifiers trained on one imputed data set converges to a local minimum as opposed to necessarily converging to the global minimum. It is not obvious then, that somehow averaging these classifiers would end up in a better minimum. Indeed, simply averaging the weights as we did does not need to end in a local minimum, as there is no convergence criterion applied to the classifier that ends up with the averaged weights. It simply has the averaged weight of the converged classifiers, but need not be in an optimal state itself. The bagging method conversely seemed to perform the best. This is the one that is used the most in practice, and we have demonstrated empirically that, indeed, it is preferable to pooling the classifiers and pooling the data sets.

7 DISCUSSION

There are some improvements that could be made to increase the accuracy of the results in this paper. Hyper-parameter optimization by, for instance, performing grid search was not done in these experiments due to computational constraints. This could further improve accuracy [15]. However, we experimented with different hyper-parameters, and doing hyper-parameter optimization is unlikely to change the ordering of the accuracy values of the pooling methods, merely the absolute values. In addition, the number of data sets we used is relatively small, and so are the data sets themselves. This is also the case for the maximum number of iterations used to achieve convergence. All of these constraints were due to limited computing power and time. However, dealing with these issues would increase the accuracy of the results found. To summarize, we think the pooling of classifiers method through averaging neural network weights is of little value, and should not be considered for real-life applications.

7.1 Future Research

As mentioned in the conclusion (section 6), the specific classifier pooling method we used does not work. Averaging the weights in neural networks of imputed data sets does not lead to a better classifier. However, there might be other methods to pool classifiers that could be researched. For instance, when gradient descent is used to train the individual classifiers, the gradients could be pooled and a central update step could then be used on the final neural network. This is already used in a federated learning context, but could be useful for working with missing data. [6]. In the end, this paper has not shown that classifier pooling cannot be useful. Instead, it has merely shown that one particular approach was not useful, and as such, a lot of research still remains to be done.

Like mentioned before in section 7, our specific analysis could also be improved to increase accuracy in places, most notably through hyper-parameter optimization. While unlikely to lead to different conclusions, it might lead to more accurate results.

REFERENCES

[1] Edgar Acu~na and Caroline Rodriguez2. 2004. The treatment of missing values and its effect in the classifier accuracy, classification, clustering, and data mining applications. (2004).

[2] Aliya Aleryani, Wenjia Wang, and Beatriz de la Iglesia. 2019. Multiple Imputation Ensembles (MIE) for Dealing with Missing Data. (2019).

[3] Melissa J. Azur, Elizabeth A. Stuart, Constantine Frangakis, and Philip J. Leaf. 2011. Multiple imputation by chained equations: what is it and how does it work? (2011).

[4] MR Baneshi, H Faramarzi, and M Marzban. 1. Prevention of Disease Complications through Diagnostic Models: How to Tackle the Problem of Missing Data? *Iranian Journal of Public Health* 41, 1 (1 1), 66–72. https://ijph.tums.ac.ir/index.php/ijph/article/view/2635

[5] John Barnard and Donald B. Rubin. 1999. Small-Sample Degrees of Freedom with Multiple Imputation. *Biometrika* 86, 4 (1999), 948–955. http://www.jstor.org/stable/2673599

[6] Robert Carlsson. 2020. Privacy-Preserved Federated Learning.

[7] Marc Claesen and Bart De Moor. 2015. Hyperparameter Search in Machine Learning. *CoRR* abs/1502.02127 (2015). arXiv:1502.02127 http://arxiv.org/abs/1502.02127

[8] Bruno Crémilleux, Arnaud Giacometti, and Arnaud Soulet. 2018. How Your Supporters and Opponents Define Your Interestingness. In *ECML-PKDD The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*. Dublin, Ireland, 373–389. https://hal.archives-ouvertes.fr/hal-01889234

[9] Pedro J. García-Laencina, José-Luis Sancho-Gómez, and Aníbal R. Figueiras-Vidal. 2009. Pattern classification with missing data: a review. *Neural Computing and Applications* 19 (2009), 263–282.

[10] Unai Garciarena and Roberto Santana. 2017. An extensive analysis of the interaction between missing data types, imputation methods, and supervised classifiers. *Expert Systems with Applications* 89 (2017), 52–65. https://doi.org/10.1016/j.eswa.2017.07.026

[11] Magdalena Graczyk, Tadeusz Lasota, Bogdan Trawiński, and Krzysztof Trawiński. 2010. Comparison of Bagging, Boosting and Stacking Ensembles Applied to Real Estate Appraisal. In *Intelligent Information and Database Systems*, Ngoc Thanh Nguyen, Manh Thanh Le, and Jerzy Świątek (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 340–350.

[12] Jeff Gruenewald and William Alex Pridemore. 2012. A Comparison of Ideologically-Motivated Homicides from the New Extremist Crime Database and Homicides from the Supplementary Homicide Reports Using Multiple Imputation by Chained Equations to Handle Missing Values. *Journal of Quantitative Criminology* 28, 1 (2012), 141–162. https://doi.org/10.1007/s10940-011-9155-5

[13] KURT HORNIK, MAXWELL TINCHCOMBE, and HALBERTWHITE. 1989. Multilayer Feedforward Networks are Universal Approximators. (1989).

[14] Shehroz S Khan, Amir Ahmad, and Alex Mihailidis. 2019. Bootstrapping and multiple imputation ensemble approaches for classification problems. *Journal of Intelligent & Fuzzy Systems* 37, 6 (2019), 7769–7783.

[15] Petro Liashchynskyi and Pavlo Liashchynskyi. 2019. Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS. (2019).

[16] Xiaoling Lu, Jiesheng Si, Lanfeng Pan, and Yanyun Zhao. 2011. Imputation of missing data using ensemble algorithms. In *2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, Vol. 2. 1312–1315. https://doi.org/10.1109/FSKD.2011.6019647

[17] Christopher Z. Mooney. 1997. *Monte carlo simulation*. Sage.

[18] Shinichi Nakagawa and Robert Freckleton. 2011. Model averaging, missing data and multiple imputation: a case study for behavioural ecology. *Behavioral Ecology and Sociobiology* 65, 1 (2011), 103–116. https://doi.org/10.1007/s00265-010-1044-7

[19] Donald. B. Rubin. 1976. Inferences and Missing Data. (1976).

[20] Donald. B. Rubin. 1987. *Multiple imputation for nonresponse in surveys*.

[21] Rianne Margaretha Schouten, Peter Lugtig, and Gerko Vink. 2018. Generating missing values for simulation purposes: a multivariate amputation procedure. *Journal of Statistical Computation and Simulation* 88, 15 (2018), 2909–2930. https://doi.org/10.1080/00949655.2018.1491577

[22] Anoop D. Shah, Jonathan W. Bartlett, James Carpenter, Owen Nicholas, and Harry Hemingway. 2014. Comparison of Random Forest and Parametric Imputation Models for Imputing Missing Data Using MICE: A CALIBER Study. *American Journal of Epidemiology* 179, 6 (01 2014), 764–774. https://doi.org/10.1093/aje/kwt312 arXiv:https://academic.oup.com/aje/article-pdf/179/6/764/17341607/kwt312.pdf

[23] Jonathan A C Sterne, Ian R White, John B Carlin, Michael Spratt, Patrick Royston, Michael G Kenward, Angela M Wood, and James R Carpenter. 2009. Multiple imputation for missing data in epidemiological and clinical research: potential and pitfalls. *BMJ* 338 (2009). https://doi.org/10.1136/bmj.b2393 arXiv:https://www.bmj.com/content

[24] Cao Truong Tran, Mengjie Zhang, Peter Andreae, Bing Xue, and Lam Thu Bui. 2017. Multiple Imputation and Ensemble Learning for Classification with Incomplete Data. (2017), 401–415.

[25] Stef van Buuren. 2018. *Flexible Imputation of Missing Data*. CRC Press.

[26] Joost R Van Ginkel and Pieter M Kroonenberg. 2014. Using generalized procrustes analysis for multiple imputation in principal component analysis. *Journal of classification* 31, 2 (2014), 242–269.

[27] Angela M. Wood, Ian R. White, and Patrick Royston. 2008. How should variable selection be performed with multiply imputed data? *Statistics in Medicine* 27, 17 (2008), 3227–3246. https://doi.org/10.1002/sim.3177 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/sim.3177

[28] Guoqiang Peter Zhang. 2000. Neural Networks for Classification: A Survey. (2000).