We present a new algorithm for data imputation based on different techniques. For example, if you remove rows containing missing values other algorithms will throw errors complaining about the missing values. The user is expected to handle the missing data and clean it before feeding it to the algorithm. Rows of a data are often not complete, especially when dealing with heterogeneous data sources. Discarding an entire row of a table if just one column has a missing value would often discard a substantial part of the data. Substituting the missing value of a numerical attribute by mean/median of non-missing values of the attribute doesn’t factor the correlations between features. It only works on column level and gives poor results on encoded categorical features. It is also not very accurate, can conflict with other attributes and doesn’t account for the uncertainty in the imputations. Substituting the missing value of a categorical attribute by most frequent value of the attribute doesn’t factor the correlations between features and can introduce bias in the data.

In this paper, we focus on given a dataset with missing values, substitute the missing data with the values which conforms with the rest of the data. Our technique inherently provides explanations for each imputation done, which can be used further to explain the results of the task at hand like drop in accuracy of an AI model or increase in bias in the data.

## 1 INTRODUCTION

Many real-world datasets may contain missing values for various reasons. Training a model with a dataset that has a lot of missing values can drastically impact the machine learning model’s quality. Some algorithms assume that all values are available and hold meaningful value. One way to handle this problem is to get rid of all the observations having any value missing. However, it involves the risk of losing data points with valuable information. The best strategy is to impute these missing values. However, most of the imputation techniques impute values for an attribute that may not conform with other attributes. For example, while imputing salary for an employee, the technique may not consider the designation of the employee and thus, impute a non-conforming value w.r.t designation attribute although the value is valid.

Some naive solutions for this problem are removing the rows containing missing values, substituting the missing values with mean or median of non-missing values of the attribute, or most frequent value for categorical data. There are certain problems with these techniques. For example, if you remove rows containing missing values other algorithms will throw errors complaining about the missing values. The user is expected to handle the missing data and clean it before feeding it to the algorithm. Rows of a data are often not complete, especially when dealing with heterogeneous data sources. Discarding an entire row of a table if just one column has a missing value would often discard a substantial part of the data. Substituting the missing value of a numerical attribute by mean/median of non-missing values of the attribute doesn’t factor the correlations between features. It only works on column level and gives poor results on encoded categorical features. It is also not very accurate, can conflict with other attributes and doesn’t account for the uncertainty in the imputations. Substituting the missing value of a categorical attribute by most frequent value of the attribute doesn’t factor the correlations between features and can introduce bias in the data.

## 2 RELATED WORK

Most research in the field of imputation focuses on imputing missing values in matrices, i.e., numerical value imputation from other numerical values. Popular approaches include k-nearest neighbors (KNN) [2], multivariate imputation by chained equations (MICE) [9], matrix factorization [8, 11, 15] or deep learning methods [3, 4, 6, 10, 18]. While some recent work addresses imputation for heterogeneous data types [12–14, 17], heterogeneous in those studies refers to binary, ordinal or categorical variables, which can be easily transformed into numerical representations.

K-nearest neighbors (KNN) based data imputation [2] replaces the missing data for a given variable by averaging (non-missing) values of its neighbors. This can be quite slow with large datasets and works for numerical data only. For categorical data, data transformation can be used, but it introduces bias. Multiple Imputations by Chained Equations (MICE) [9] is an iterative algorithm based on chained equations uses an imputation model specified separately for each variable, involving the other variables as predictors. This work only considers numerical values on small data sets. Datawig [3, 4] learns Machine Learning models using Deep Neural Networks to impute missing values. It supports both CPU and GPU for training and uses feature encoder to handle categorical data. This works well with categorical and non-numerical features, but needs the columns as input that contain information about the target column to be imputed. This is quite slow, especially with large datasets.

NADEEF: A commodity data cleaning system [5] allows the users to specify data quality rules, defining data deficiencies and how to repair it through writing code that implements predefined classes.
Such rule based systems achieve high precision for imputation, but often require a domain expert in the loop to generate and maintain the set of rules to apply. Other data imputation techniques based on eigen values include singular value decompositions and bayesian principal component analysis. The main drawback of these techniques is that they work well with numerical data only.

3 IMPUTATION MODEL

In this section, we discuss our solution approach. Section 3.1 discusses our constraints inference technique, which computes constraints from the given data. Section 3.2 discusses our imputation technique using the inferred constraints, thereby also generating explanations for better understanding.

3.1 Constraints Inference

The first step in imputing the missing values is understanding each column in the given data and finding correlations between different type of columns. We have defined seven datatypes for columns - EMPTY, DATE, TEXT, CAT_TEXT, NUMERIC, CAT_NUM and FLOAT. If a column does not have any value, the datatype for that column is EMPTY; if it contains date or time specific data, the datatype is DATE; if it contains string values, the datatype is TEXT or CAT_TEXT; if it contains integer values, the datatype is NUM or CAT_NUM; if it contains float values, the datatype is FLOAT.

Most of these datatypes are standard. The interesting and non-standard ones are CAT_TEXT and CAT_NUM. These datatypes cater to columns with very few unique values. For example, gender is a column with string values with only two or three unique values while person-name is a column with string values but the number of unique values can be of the order of the number of values in the column. In order to differentiate between these two columns, we have defined separate datatypes - TEXT and CAT_TEXT. Similarly we differentiate between NUMERIC and CAT_NUM. This distinction helps in finding specific constraints at the column level and in finding interesting associations. For example, salary of employees may have different distributions based on the gender value.

For each column, we first find the datatype for that column data and then find constraints based on the datatype. We define column level constraints for each column depending on the datatype of that column. The column level constraints for these datatypes include min, max, mean and distribution for NUMERIC, CAT_NUM and FLOAT columns; mindate, maxdate and format for date columns; and frequency distribution for CAT_NUM and CAT_TEXT columns.

In addition to column constraints, we also define multi-column constraints, called associations (Table 1), between column pairs based on their datatypes. Each association has a source and a target column. The associations are - CAT-CAT, CAT-DATE, Specialty-DATE, NUM-DATE, DATE-DATE, DATE-CAT, CAT-TEXT and TEXT-CAT. All associations, except CAT-CAT, describe relations between columns. CAT-TEXT-CAT describes relations between two NUMERIC or FLOAT columns depending on values in a categorical column.

The CAT-CAT association is defined for two categorical columns, and for each value in the source column, the frequency distribution of values in the target column is computed. The CAT_NUM association is defined between a categorical column and a numerical column, and for each value in the source column, we find min, max, mean and distribution of target column. For each source value, min, max, mean and distribution of target column is computed. The CAT-TEXT association, similar to CAT_NUM, in the CAT-TEXT association, we find frequency distribution of the target column for every value of the source column. For the NUM-NUM association, we try to find a polynomial function from the source column to the target column. The target column may not be an exact function of the source column, but an approximate one. So we also find the error of how good fit this polynomial is which helps in imputation.

For example, if multiple NUM-NUM associations are available for a target column, we choose the one with the least error. For the DATE-DATA association, we find the difference between two date columns. A simple example for this association is the difference between order date and delivery date for a product. The complete pseudo-code of all our algorithms is available at [7].

3.2 Imputation using constraints

The imputation is done using the constraints and the values in other columns. The idea is to impute the values of a column using associations first, and if it is not possible to use associations, impute using column level constraints. Note that this can happen either due to non-availability of the required associations or due to the non-availability of the values of other columns required by association.

If there are multiple values missing in the same row, the values are imputed in a particular order. This is done by constructing a graph based on associations and then sorting it topologically. The intuition behind this is that the columns with very few unique values, for example labels in a training dataset, have more valuable information than a column with many values. Thus, the objective is to give preference to categorical columns over numerical or text columns. Note that this is also reflected in the way associations are defined; the source columns for most associations are categorical(CAT-TEXT/CAT-DATE). For example, for a data with categorical attributes (age, marital, gender, salary, education), a sorting of associations based graph may be (gender, education, marital, age, salary) and the imputation is carried out in this order of attributes.

For imputing a categorical column, the CAT-NUM associations are used first and if it fails, CAT-CAT associations are used and if that also fails, most frequent value of the column is chosen. For the imputation using CAT-NUM association, all the possible values for the missing column are computed that conform with the values in other numerical columns range specified in the constraints. The value that conforms with most of the values in numerical columns is chosen. If there are multiple possible values that conform with most numerical columns, the one closest to the mean of the numerical columns is chosen. For example, for imputing education in above

| Association     | Source        | Target        | Constraints                        |
|-----------------|---------------|---------------|------------------------------------|
| CAT-CAT         | CAT-TEXT      | CAT-TEXT      | Frequency distribution             |
| CAT-DATE        | DATE          | DATE          | Frequency distribution             |
| NUM-DATE        | DATE-DATE     | DATE          | Polynomial function               |
| CAT-TEXT-CAT    | NUMERIC       | FLOAT         | Polynomial function, for each category value |

Table 1: Associations
example, the value in age and salary column is used since there are CAT-NUM associations from education to age and salary. Similarly for CAT-CAT association, all the possible values are computed that are most frequent for values in categorical columns, and the one with highest probability is chosen. For example, for imputing value with expected value of the given distribution with least error.

For numerical columns, NUM-NUM association is used. If it fails, i.e., if there is no NUM-NUM association for the missing value column, or the source column value is missing, CAT-NUM-NUM association is used. If CAT-NUM-NUM association also fails, then CAT-NUM association is used and if that also fails, mean value of the column is chosen. The imputation using NUM-NUM association is straightforward. If there are multiple NUM-NUM associations in the constraints, the one with least error is used for imputation. The imputation using CAT-NUM-NUM is similar to imputation using NUM-NUM, subject to value in a categorical column. The imputation using CAT-NUM, similar to the imputation of a categorical column using CAT-CAT association, imputes the value with expected value of the given distribution with least error.

For a text columns, CAT-TEXT association is used. If it fails, the most frequent value of the column is used. The imputation of a DATE column uses DATE-DATE association if there are other date columns, otherwise the value is imputed with the median of the column. For example, for imputing order date, delivery date is used.

**Explanations for Imputations** The explanations for an imputation comes directly from the constraints used. For example, if CAT-NUM is used for imputing a numerical value, the value and the name of the categorical column used is the explanation for the imputation. Similarly, for an imputation using NUM-NUM constraint, the value and the name of the numerical column used is the explanation.

### 4 EXPERIMENTAL EVALUATION

We evaluate the performance of our approach as compared to existing ones. We consider three metrics for comparison, namely Data Accuracy, Prediction Accuracy and Fidelity. Please note that most of the prior works have reported only Data Accuracy. We are evaluating on two additional metrics to further assess how the missing value imputation using different approaches impact the model accuracy. A detailed description of these metrics is presented along with the comparative evaluation of different approaches in Section 4.1.

**Benchmarks and Configuration:** We evaluate our approach on open-source data sets and a synthetic data set with three numeric attributes with NUM-NUM polynomial associations. Our code is written in Python and executed in Python 3.7. All the experiments are performed in a machine running macOS 10.14, with 16GB RAM, 2.7Ghz CPU running Intel Core i7.

| Benchmark | Size | #Features | #Num-Features | #Cat-Features | #Date-Features |
|-----------|------|-----------|---------------|--------------|---------------|
| Polynomial | 1000 | 5 5 3 | 0 0 0 | 0 0 0 | 0 0 0 |
| Iris | 150 | 5 4 1 0 | 0 0 0 | 0 0 0 | 0 0 0 |
| Ecoli | 336 | 8 7 1 0 | 0 0 0 | 0 0 0 | 0 0 0 |
| Wine | 178 | 13 13 0 0 | 0 0 0 | 0 0 0 | 0 0 0 |
| Diabetes 2 | 486 | 20 20 0 0 | 0 0 0 | 0 0 0 | 0 0 0 |
| Breast Cancer 2 | 286 | 9 9 0 0 | 0 0 0 | 0 0 0 | 0 0 0 |
| Bank Market 2 | 92114 | 18 18 0 0 | 0 0 0 | 0 0 0 | 0 0 0 |

1 synthetic data with column having polynomial relationships 2 https://archive.ics.uci.edu/ml/datasets.php

**Table 2: Benchmark Characteristics**

We have compared the performance of our approach to fill missing gaps in data as compared to the existing ones, such as mean, k-means, KNN [2], MICE [9], MissForest [14, 16], Datawig [3, 4]. We leverage the existing functionalities in sklearn to implement some of these prior works. The implementations for mean, KNN have been taken from *fancyimpute* package, and *IterativeImputer* with the estimators *RandomForestRegressor* and *LinearRegression* mimics the MissForest and MICE, respectively. For k-means, we use sklearn’s implementation with cluster count set as 4. Further, Datawig API [1] is used to fetch performance numbers for Datawig.

**RMSE and F1 Score.** For different benchmarks, using different approaches, we record the RMSE values obtained for the numerically imputed versions in Figure 1. For the categorical imputations, we record F1-score (shown in Figure 2). The plots in the figures clearly shows that our technique imputes better than the existing techniques by introducing less outliers in the imputed versions.

**NRMSE.** For a column with datatype NUMERIC or CAT_NUM, the normalized root mean square error with a standard deviation $\sigma_c$ is $\text{NRMSE}_c = \text{RMSE}_c / \sigma_c$. Hence, the normalized root mean square for a dataset, with NCOL as the set of all NUMERIC or CAT_NUM columns, is the mean of normalized root mean square error for all such columns, i.e., $\text{NRMSE} = \frac{1}{\text{NCOL}} \sum_c \text{NRMSE}_c$. Table 3 shows NRMSE for benchmarks with different missing values percentage using different imputation techniques. The values here are the average ones across 5 consecutive iterations.

**Table 3: NRMSE for ours vs related works**

**Missing data generation:** The data sets we consider for our experiments have no missing value in their original forms. We take a random approach to pick indices in any data set to discard their values and treat them as missing ones. We consider a missing data percentage variable, perc. to define how many values in the data are treated as missing. Also note that we first encode the categorical text values present in the data sets using an appropriate encoder before feeding it to the data imputation engine. Such values which were treated as CAT_TEXT in their original form are now marked with a datatype CAT_NUM by our constraint inference module.

**Experiment Runs:** We take an iterative approach while running our experiments to augment the reliability and correctness of our results. We have set the variable iter as 5 for all our experimental runs, i.e., 5 consecutive imputation rounds were performed to replace the missing values in an input data set. The numbers reported for different metrics in the subsequent subsections are the average of all the iterations in a single experiment run.

**4.1 Comparison with the Related Works**

We have compared the performance of our approach to fill missing gaps in data as compared to the existing ones, such as mean, k-means, KNN [2], MICE [9], MissForest [14, 16], Datawig [3, 4]. We leverage the existing functionalities in sklearn to implement some of these prior works. The implementations for mean, KNN have been taken from *fancyimpute* package, and *IterativeImputer* with the estimators *RandomForestRegressor* and *LinearRegression* mimics the MissForest and MICE, respectively. For k-means, we use sklearn’s implementation with cluster count set as 4. Further, Datawig API [1] is used to fetch performance numbers for Datawig.

**Next, we discuss evaluation metrics and the experimental results.**

**RMSE and F1 Score.** For different benchmarks, using different approaches, we record the RMSE values obtained for the numerically imputed versions in Figure 1. For the categorical imputations, we record F1-score (shown in Figure 2). The plots in the figures clearly shows that our technique imputes better than the existing techniques by introducing less outliers in the imputed versions.
Bench. perc. | (Prediction Accuracy, Fidelity) for iter=5
---|---
Wine 5 | 0.85, 0.81 | 0.83, 0.79 | 0.81, 0.79 | 0.80, 0.78 | 0.80, 0.78
10 | 0.85, 0.72 | 0.83, 0.74 | 0.81, 0.73 | 0.80, 0.72 | 0.80, 0.72
20 | 0.84, 0.72 | 0.83, 0.73 | 0.81, 0.73 | 0.80, 0.72 | 0.80, 0.72
30 | 0.82, 0.72 | 0.81, 0.73 | 0.80, 0.72 | 0.80, 0.72 | 0.80, 0.72
Iris 5 | 0.95, 0.95 | 0.95, 0.95 | 1.1 | 1.1 | 1.1
10 | 0.95, 0.95 | 0.95, 0.95 | 1.1 | 1.1 | 1.1
20 | 0.95, 0.95 | 0.95, 0.95 | 1.1 | 1.1 | 1.1
30 | 0.95, 0.95 | 0.95, 0.95 | 1.1 | 1.1 | 1.1

Table 4: Prediction Accuracy and Fidelity

Prediction Accuracy and Fidelity. We split the dataset as train_inputs and test_inputs in 70:30 and train a Decision Tree Classifier or Regressor (based on label datatype) using train_inputs, and record the accuracy on test_inputs. Using train_inputs, we generate training inputs, say train_impute, with missing values while varying missing data percentage and impute them using different approaches. We also train a Decision Tree Classifier or Regressor using every train_impute and record its accuracy on test_inputs, and the fidelity of the trained models using imputed trained data for categorical label datasets. Fidelity is the fraction of test inputs for which both models (trained on train_inputs and train_impute) output same decision. Table 4 shows the accuracy and fidelity of models trained using imputed training data using different methods for Wine and Iris datasets. Note that both datasets have categorical labels, hence, fidelity scoring was possible. Our approach performs better than the state-of-the-art methods while offering higher accuracy without compromising much on fidelity.

5 CONCLUSION

For datasets with no or few associations, i.e., attributes are independent, other techniques give better results than imputation using constraints. But when the attributes are related, which is more often than not in most real datasets, constraints based imputation outperforms other techniques.
REFERENCES

[1] Last accessed 12th Jul, 2022. DataWig - Imputation for Tables. https://prpi.org/project/datawig

[2] Gustavo E. A. P. A. Batista and Maria Carolina Monard. 2003. An Analysis of Four Missing Data Treatment Methods for Supervised Learning. Applied Artificial Intelligence 17, 5-6 (2003), 519–533. https://doi.org/10.1080/713827181

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

[4] Felix Biessmann, David Salinas, Sebastian Schelter, Philipp Schmidt, and Dustin Lange. 2018. "Deep" Learning for Missing Value Imputation in Tables with Non-Numerical Data. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018, Alfredo Cuzzocrea, James Allan, Norman W. Paton, Divesh Srivastava, Rakesh Agrawal, Andri Z. Broder, Mohammed J. Zaki, K. Selçuk Candan, Alexandros Labrinidis, Assaf Schuster, and Haixun Wang (Eds.). ACM, 2017–2025. https://doi.org/10.1145/3269206.3272005

[5] Michele Dallachiesa, Amir Ebrahimi, Ahmed Eldawy, Ahmed K. Elmagarmid, Ihab F. Phung, Vincent S. Tseng, Geoffrey I. Webb, Bao Ho, Mohadeseh Ganji, and Lida Rashidi (Eds.). Springer, 260–272. https://doi.org/10.1007/978-3-319-93040-4_21

[6] Lovedeep Gondara and Ke Wang. 2018. MDIA: Multiple Imputation Using Denoising Autoencoders. In Advances in Knowledge Discovery and Data Mining - 22nd Pacific-Asia Conference, PAKDD 2018, Melbourne, VIC, Australia, June 3-6, 2018, Proceedings, Part III (Lecture Notes in Computer Science, Vol. 10899), Dinh Q. Phung, Vincent S. Tseng, Geoffrey I. Webb, Bao H. Ho, Madahesh Gani, and Lida Rashidi (Eds.). Springer, 260–272. https://doi.org/10.1007/978-3-319-93840-4_21

[7] Sandeep Hans, Deptikalyan Saha, and Aniya Aggarwal. 2022. Explainable Data Imputation using Constraints. CoRR abs/2205.04731 (2022). https://doi.org/10.48550/arXiv.2205.04731 arXiv:2205.04731

[8] Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. IEEE Computer 42, 8 (2009), 30–37. https://doi.org/10.1109/MC.2009.263

[9] R.J.A. Little and D.B. Rubin. 2002. Statistical analysis with missing data. Wiley. http://books.google.com/books?id=ayPwAAMAAJ

[10] Pierre-Alexandre Mattei and Jes Frellsen. 2019. MIWAE: Deep Generative Modeling and Imputation of Incomplete Data Sets. In Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research, Vol. 97), Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 4413–4423. http://proceedings.mlr.press/v97/mattei19a.html

[11] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. 2010. Spectral Regularization Algorithms for Learning Large Incomplete Matrices. J. Mach. Learn. Res. 11 (2010), 2287–2322. http://portal.acm.org/citation.cfm?id=1859931

[12] Alfredo Nazábal, Pablo M. Olmos, Zoubin Ghahramani, and Isabel Valera. 2018. Handling Incomplete Heterogeneous Data using VAEs. CoRR abs/1807.03653 (2018). arXiv:1807.03653 http://arxiv.org/abs/1807.03653

[13] Shaoshu Song, Yu Sun, Aqiqan Zhang, Lei Chen, and Jianmin Wang. 2018. Enriching data imputation under similarity rule constraints. IEEE transactions on knowledge and data engineering 32, 2 (2018), 275–287.

[14] Daniel J. Stekhoven and Peter Bühlmann. 2012. MissForest - non-parametric missing value imputation for mixed-type data. Bioinformatics 28, 1 (2012), 112–118. https://doi.org/10.1093/bioinformatics/btr597

[15] Olga G. Troyanskaya, Michael N. Cantor, Gavin Sherlock, Patrick O. Brown, Trevor Hastie, Robert Tibshirani, David Botstein, and Russ B. Altman. 2001. Missing value estimation methods for DNA microarrays. Bioinformatics 17, 6 (2001), 520–525. https://doi.org/10.1093/bioinformatics/17.6.520

[16] S. van Buuren. 2018. Flexible Imputation of Missing Data. CRC Press, Taylor & Francis Group. https://books.google.com/books?id=bLmItgEACAAJ

[17] Jinsung Yoon, James Jordan, and Mihaela van der Schaar. 2018. GAIN: Missing data Imputation using Generative Adversarial Nets. In Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholm, Sweden, July 10-15, 2018 (Proceedings of Machine Learning Research, Vol. 80), Jennifer G. Dy and Andreas Krause (Eds.). PMLR, 5675–5684. http://proceedings.mlr.press/v80/yoon18a.html

[18] Hongbao Zhang, Pengtao Xie, and Eric P. Xing. 2018. Missing Value Imputation Based on Deep Generative Models. CoRR abs/1808.01684 (2018). arXiv:1808.01684 http://arxiv.org/abs/1808.01684