Investigation of Multivariate Data Imputation Technique (MICE) for Sparse Data of Varying Dimensionality

Jaelle Kondohoma
University of Nebraska-Lincoln
Lincoln, USA
jaelle.kondohoma@huskers.unl.edu

(Robert) Scott Martin
University of Nebraska-Lincoln
Lincoln, USA
rmartin45@huskers.unl.edu

Richard Mwaba
University of Nebraska-Lincoln
Lincoln, USA
rmwaba2@huskers.unl.edu

Abstract—**We investigate the use of multiple-imputation to impute missing data on sparse data of various dimensions. In particular, we look at two different data sets of different dimension and from diffferent domains and investigate the effect of imputing missing data on downstream machine learning models. We specifically focus on using machine learning models for our imputation and attempt to use hyperparameter tuning to determine an optimal multiple-imputation model for our datasets.**

I. Introduction

It is well-known that machine-learning is a "data hungry" discipline and many ML algorithms require large amounts of data to be effective. However, a sometimes overlooked piece of this problem is that the data must also be a in usable form. In particular, we examine the question of how to handle incomplete datasets. This problem is extremely common particularly in fields in which data is given via a survey or some other user-inputted mechanism as people are prone to not answering questions or not taking the time to input all data properly. As such any practical discourse on machine learning must include a discussion on the methods (and drawbacks of each method) of data imputation.

Before discussing our specific approach to data imputation we will briefly discuss data missingness in general. There are three common types of missing data: missing completely at random (MCAR), missing at random (MAR), and not missing at random (NMAR). The first involves data that is missing for no reason that is explainable within the data itself; that is, there is no correlation between the missing values and any values in the dataset. On the other hand, MAR data has some pattern to its missingness which is present in the rest of that dataset. So MAR data can be imputed well because you can use the rest of the dataset to predict the missing data. Finally, NMAR is data that is missing in a pattern which often times require domain knowledge to deduce. For the purposes of this paper we deal almost exclusively with MAR data.

In this paper we investigate an approach to imputation known as multiple imputation (sometimes called iterative imputation, or, popularly, MICE)[4]. This is the process of imputing multiple missing columns using the previously imputed values of columns to help impute the values of other columns. Doing this in a round-robin technique allows you to impute all missing values in your data set and not introduce significant bias. Our goal in this paper is to apply multiple imputation on sparse datasets using various machine learning models (in particular, ridge regression, KNN, random forests, and ANN) to impute missing data and then, using the newly complete data sets, run a downstream machine learning model to predict the "true target" variable. In this context we will use the term "imputing target" to mean the missing piece of data being imputed and the "true target" to represent the desired target to predict using a ML model. While this is not a new idea (see [2] for a very thorough treatment and [3] for a specific look at using random forests) we hope to specifically focus on hyperparameter tuning the machine learning models that are doing the imputation to optimize the imputation process. We also hope to come to some conclusion about the specific hyperparameters and models that are most useful in different domains and dimensionalities, but, as we will discuss, this process was unsuccessful. We draw strongly from the work of Chakroborti and Moore [1] and hope to extend their work to specifically include multi-column imputation and an investigation into the effects of more thorough hyperparameter tuning.

II. Data Summary

A. Datasets

Two different data sets are used in order to investigate MICE for sparse data of varying dimensionality. The first is a dataset from Kaggle[1] containing information about home sale prices versus Zillow estimates for sale prices. This dataset is specifically used for comparison with (or to expand on) the work of [1]. The second dataset contains US census data about individual's income.[2] The census data has 32561 training instances and 15 variables. This dataset is complete—it's not missing any data—and we will introduce missing-ness to this dataset by randomly dropping values. The Zillow data

[1]https://www.kaggle.com/c/zillow-prize-1/data
[2]https://archive.ics.uci.edu/ml//datasets/Census+Income

Feature	Correlation
finishedsquarefeet15	46159
finishedsquarefeet12	4328
finishedfloor1squarefeet	45471
finishedsquarefeet50	45471
poolsizesum	48883
calculatedfinishedsquarefeet	699
calculatedbathnbr	1543
fullbathcnt	1543
basementsqft	49353
fireplacecnt	44062
garagecarcnt	35203
bathroomcnt	0
garagetotalsqft	35203
threequarterbathnbr	44440
yardbuildingsqft17	47793
unitcnt	15573
finishedsquarefeet6	49055
numberofstories	38227
bedroomcnt	0
finishedsquarefeet13	49306
yearbuilt	800

TABLE I: Missing data per Column-ZILLOW

was parsed down to 50000 entries (with 58 features) and all categorical data was dropped after encoding the categorical variables and seeing negligible correlation values between the categorical variables and the target. Both datasets have a combination of categorical, real valued and integer valued features.

B. Data Preprocessing

Preprocessing of the census data consisted of standardizing the data and one-hot encoding categorical features. After experimenting with augmenting the features this idea was abandoned as it was having little affect on the outcome. Before any classification algorithm was run the data was standardized since we used logistic regression as our classifier which is sensitive to non-standardized data. An important note is that the census data was complete in that it had no missing values. However, it was selected for two primary reasons. First, we wanted a dataset that we could manipulate and test different scenarios and sparseness of missing data with. Second, we wanted a dataset with a very clear, fairly low dimensional, classification problem as a sanity check on our imputation results. Unfortunately, these two things interacted in a way we didn't expect; it was difficult to make the census data sensitive to missing values. We will discuss this further in the results section.

For the Zillow dataset, which will be the focus of the remainder of the discussion, all null values in the target columns were dropped. We picked the columns that had correlation coefficient greater than or equal to 0.1 and the rest were dropped. In the beginning there were 58 columns including 5 columns with the categorical data. These 5 columns *hashottuborspa, propertycountylandusecode, propertyzoningdesc, fireplaceflag, taxdelinquencyflag* were dropped. 617 NaN values in the Target collumns were dropped.

The columns with correlation coefficient greater than 0.8 were dropped because they were basically transformations of the target and hence had extremely high correlation; these columns were *taxamount*, *structuretaxvaluedollarcnt*, *landtax-valuedollarcnt*.

Feature	Correlation
taxvaluedollarcnt	1.000000
taxamount	0.963647
landtaxvaluedollarcnt	0.847662
structuretaxvaluedollarcnt	0.834219
finishedsquarefeet15	0.658055
finishedsquarefeet12	0.636250
finishedfloor1squarefeet	0.574428
finishedsquarefeet50	0.565945
poolsizesum	0.520322
calculatedfinishedsquarefeet	0.509469
calculatedbathnbr	0.495457
fullbathcnt	0.481303
basementsqft	0.465104
fireplacecnt	0.454990
garagecarcnt	0.369250
bathroomcnt	0.360499
garagetotalsqft	0.356908
threequarterbathnbr	0.348244
yardbuildingsqft17	0.289622
unitcnt	0.197026
finishedsquarefeet6	0.178706
numberofstories	0.175817
bedroomcnt	0.161504
finishedsquarefeet13	0.158479
yearbuilt	0.156499

TABLE II: Most Correlated variables-ZILLOW

C. Exploratory Data Analysis (EDA)

The only exploratory data analysis done on the Zillow data was dropping NaN values, picking out most correlated values and dropping categorical data.

The EDA for the census data consisted of two main parts. First, the data was largely categorical so determining value counts and valid entries for the categorical data was done first via histograms and then reiterated after the one-hot encoding of the categorical variables. The most important thing for the purpose of our paper was calculating the correlation between each variable and the target since we'd planned to use the highly correlated variables to drop data from and, hopefully, be able to explore the effects of imputing that missing data. Unfortunately, as was mentioned earlier the census data was not used for the main result of this paper since we could never seem to introduce enough missing-ness to drastically affect the performance of the classifier.

III. Methods

We performed experiments with a focus on four imputation strategies, that is, Order Least Squares Regression, K-Nearest Neighbor Regression, Random Forests Regression and Artificial Neural Network Regression. All these exist as modular strategies under the MICE implementation of the multivariate imputation approach. We performed hyper-parameter tuning with Mean Squared Error (MSE) as the chosen score metric to select the best model for imputation. To assess, the imputation effectiveness of each strategy, we employ MSE and r^2

score metrics on the base estimation model. The overall flow of our investigation is show in Fig. 1.

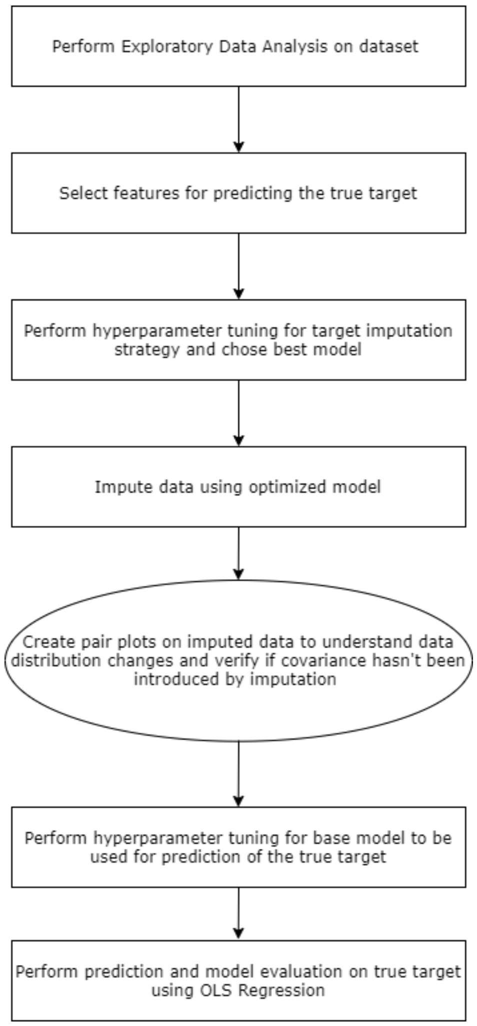
Perform Exploratory Data Analysis on dataset
Select features for predicting the true target
Perform hyperparameter tuning for target imputation strategy and chose best model
Impute data using optimized model
Create pair plots on imputed data to understand data distribution changes and verify if covariance hasn't been introduced by imputation
Perform hyperparameter tuning for base model to be used for prediction of the true target
Perform prediction and model evaluation on true target using OLS Regression

Fig. 1: **Imputation flow chat**.

In addition to the mentioned strategies, we performed Mean Imputation (MI) to act as a baseline for the other strategies and hence determined performance with respect to it.

A. Hyper-parameter Tuning

Before imputing the data with the target strategy, we applied hyperparamater tuning (optimization) to ensure the we chose the best model for imputation. Hyperparameter tuning can be defined as the problem of choosing a set of optimal parameters for a learning algorithm [3]. We ran multiple experiments on every strategy through cross-fold validation and picked the model that performed the best (had the lowest MSE). Except for Mean imputation,each strategy had a distinct set of hyperparameters that were optimized. These parameters were chosen according to what extent each can affect the results of the model in terms of performance. Table III in the APENDIX shows which parameters were optimized for each strategy.

We also applied hyperparameter tuning when choosing the model to be used for prediction on the base model.

B. Mean Imputation

In mean imputation, we replace the missing values with the mean of the available feature values of each column. It has a low time complexity of $O(d)$ where d is the number of columns available. This approach performs well for data that is well distributed and also not sparse. It doesn't perform really well on data that is sparse and is highly likely to underfit it. However, we still chose it as the base model because of its large usability and simplicity.

C. Order Least Squares - Bayesian Ridge

Bayesian Ridge regression is a form of linear regression with a regularization term which helps stabilize the coefficients of the weight matrix. Whereas KNN for example has very fast training time and fairly slow runtime Bayesian Ridge has a low runtime and a high training time as it attempts to find the optimal weight matrix during training. We chose Bayesian Ridge as a ground-level machine learning model in some sense. It is the only linear model we selected (since we suspected that the data was non-linear) and we selected specifically as a comparison to more traditional MICE implementations which use, typically, OLS.

D. K-Nearest Neighbor Regression

K-Nearest Neighbor (KNN) is a basic machine learning algorithm which predicts the class of some test value based on the classes of the test values nearest neighbors. The KNNRegressor, as implemented in scikit-learn, transfers this to a regression task by interpolating the value of the test data based on the values of its neighbors. The k in KNN is one of several hyperparameters we optimized. This value determines the number of neighbors to investigate during the execution of the algorithm. We also optimized the weights measure (which determines how much weight to assign to neighbors), the number of features to use in the prediction, and which metric to use. The time complexity of the KNN algorithm depends on the implementation however at best it is $O(d \log N)$ where d is the number of features and N is the number of records. In practice, the values of the hyperparameters will also affect KNN runtime. We selected this model as one of our choices since it's a non-linear model and there was uncertainty on whether our dataset was linearly classifiable or not due to its sparsity. Typically KNN is not a good choice for high-dimensional data, however since we dropped non-correlated columns our data in the end was fairly low-dimensional and hence KNN was a logical choice of model.

E. Random Forests Regression

Random forests are a common technique to be used with multiple imputation [3]. They use many decision trees averaged to make predictions. These are non-linear models which is part of the reason we picked them to use in our investigation. As we have mentioned, we suspected our data was non-linear and as such we wanted to see how several different non-linear models worked on the data. Similar to ANN, which we'll talk about next, the random forest implementation was slow to train based on the size of our dataset.

BayesianRidge	**KNeighbors**	**Random Forest**	**ANN**
n_iter	n_neighbours	n_estimators	hidden_layer_sizes
tol	weights	min_sample_split	alpha
alpaha_1	p	max_depth	tol
lambda_1			

TABLE III: Hyperparameters that were optimized for each strategy

F. Artificial Neural Network Regression - MLP

The last method we implemented was artificial neural networks. Artificial neural networks are extremely powerful non-linear classifiers. However, they have the disadvantage of being particularly slow to train, especially on large datasets like ours. Because of this we had to limit our hyperparameter tuning to the number of neurons in the single hidden layer. This constraint meant we always used ReLU as our activation function and didn't experiment with different solvers (we used the default ADAM solver) or the various parameters that can help optimize the solvers. We selected ANN as a model for several reasons: the most practical was that in [1] Chakraborti and Moore found that for single column imputation ANN was the most effective model they tried. Moreover, as discussed earlier we suspected the data was non-linear and ANN is one of the best non-linear models available. However, in hindsight, the overhead of training time for ANN was a detriment and would need to be improved upon for ANN to be a practical imputation model for the everyday data scientist.

G. Census Data Methods

We introduced missing-ness in the census data by dropping various percentages of random values from highly correlated sets of columns. For example, we would find the n columns most correlated with our target variable (in this case the binary value of "$income > 50K$") and randomly change p percent of values in each of those n columns to NaN. We had hoped to then use an imputation strategy to impute the missing data and then compare the performance of a classification algorithm (we chose logistic regression for this purpose since the data was linear) on the full dataset to performance on the imputed dataset. However, as we will discuss in the results section this proved to be unfruitful.

IV. Results

A. Zillow Dataset

The results are summarized in Table IV and visualized in Fig 2. The table contains the $r^2 score$ of the base model prediction on train and test data from the Zillow dataset. Despite the data being highly sparse, some columns missing as high as 90% of data, every strategy imputed missing data successfully which matched our expectation. However, it does not end at only having a successful imputation. It goes further in trying to understand the impact each imputation had on the predictive ability of the model. To understand the impact, we report the $r^2 score$ only on the base model for each strategy.

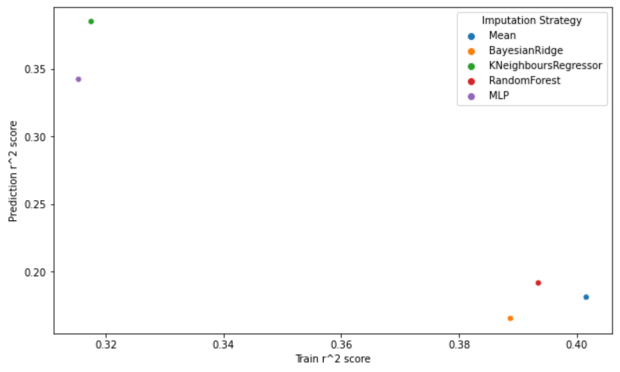

Fig. 2: Plot of performance of imputation using $r^2 score$ on test and validation data.

K-Nearest Neighbor(KNN) Regressor proved the most effective with a score of 0.38 on the validation data. This was an interesting result as it was not expected. When the majority of data surrounding the imputer target is missing, KNN Regressor suffers due to the distance between the target and the neighbors increasing. This affects the imputed values in every iteration of the imputer hence diverging from the possible optimal values. In this case however, the random nature of the missing data gives KNN Regressor a great chance at predicting optimal values. If the closest neighbors found in the first few iterations have their full values, this will continue to be the case as imputation proceeds due to the imputer reusing imputed values to determine other missing values. Compared to Mean imputation, KNN Regressor performs significantly better.

The second most effective strategy was ANN Regressor. It scored 0.34 on validation data. This was expected to be the best model as it is not highly affected by the nature of the data. Due to its great optimal and iterative nature, ANN is expected to perform considerable better in every iteration which leads to possibly optimal imputed values. However, the overhead of training the ANN model cannot be overlooked. It takes a considerably huge amount of time to train the ANN imputer which led to not using enough parameters to perform hyperparameter tuning. There is a higher chance ANN performing even better if more parameters are explored during training of the model.

Random Forest and Bayesian Ridge were in the same range of $r^2 score$ values of 0.19 and 0.16 respectively. There was no great expectation on Bayesian Ridge as it largely relies on the availability of a good number of values for computation. In this case where a lot of values are missing, the computation goes way off the possibly optimal values causing it to perform poorly. Random Forests on the other hand needs as many values as possible to form and split trees. It is therefore highly affected by the orientation of the data. In the case that the data is shuffled and there are less missing values in the first few iterations, it is expected to perform better.

Imputation Strategy	**Train r^2 score**	**Prediction r^2 score**
Mean	0.401598	0.181020
BayesianRidge	0.388733	0.165326
KNeighboursRegressor	0.317524	0.384981
RandomForest	0.393469	0.191547
MLP	0.315377	0.342243

TABLE IV: Prediction results

B. Census Dataset

After applying our procedure to simulate missing data with the census dataset we assumed the accuracy of any classification algorithm–in our case logistic regression in particular–would drop significantly. However, that interestingly proved to be false. After letting n–the number of highly correlated columns to drop data from–range from 1 to 10 and p–the percentage of values in the column to drop–range from 0.1 to 0.7 the difference in model classification accuracy from the full dataset (missing no values) to the dataset with all rows with NaN values dropped was never more than 12% (and the classification accuracy on the reduced dataset still $>$70%). Since even the most naive approach to missing data–dropping the records with missing values–seemed to not drastically decrease the effectiveness of the classification algorithm on the dataset we abandoned the use of the census data for testing imputation and instead focused on the Zillow dataset for imputation. While we do not know for sure the cause of the census data confusion we hypothesize that the data distribution is fairly narrow and hence it doesn't take much data to recognize the distribution and predict it.

V. Conclusion

The main goal was to investigate the multivariate imputation of missing data using Machine Learning models on sparse datasets. We studied four different Machine Learning models whose parameters were optimized prior to applying them on the imputation of missing values. Our study indicates that ANN performs very well in multivariate imputation. However, this lacks in many ways to be a strong conclusion because of the many challenges that we faced. The time period for the project was really compressed and this did not allow us to do a thorough investigation. There was also some randomness noticed in the results when the dataset is shuffled. When the available data form clusters after the shuffle, Random Forest and KNN Regressor are seen to perform better while in many cases ANN was not affected. One way to address is to ensure that the dataset is shuffled the same by employing the necessary parameters which can nonetheless be a hindrance to a deeper investigation.

It is also important to note that sparsity of data differs. Our investigation was carried out on a near extreme case and it would be ideal to investigate further with different levels of sparsity.This would help understand an existence of a correlation between data sparsity and imputation effectiveness. In the future, we would like to extend the study to applying more advanced Machine Learning models and be able to infer the level of missingness of the data.

References

[1] P. Chakraborti, B. Moore. "The missing data conundrum". UNL CSCE 878 Project-Unpublished manuscript. 2019.

[2] D. Bertsimas, C. Pawlowski, Y. Zhou. "From Predictive Methods to Missing Data Imputation: An Optimization Approach". *Journal of Machine Learning Research* 18 (2018) 1-39.

[3] A. Shah, J. Bartlett, J. Carpenter, O. Nicholas, H. Hemingway. "Comparison of Random Forest and Parametric Imputation Models for Imputing Missing Data Using MICE: A CALIBER Study". *American Journal of Epidemiology*. 179(6): 764-774. 2015.

[4] S. van Buuren, K. Oudshoorn. "Flexible Multivariate imputation by MICE". TNO Prevention and Health. October, 1999.

VI. Appendix

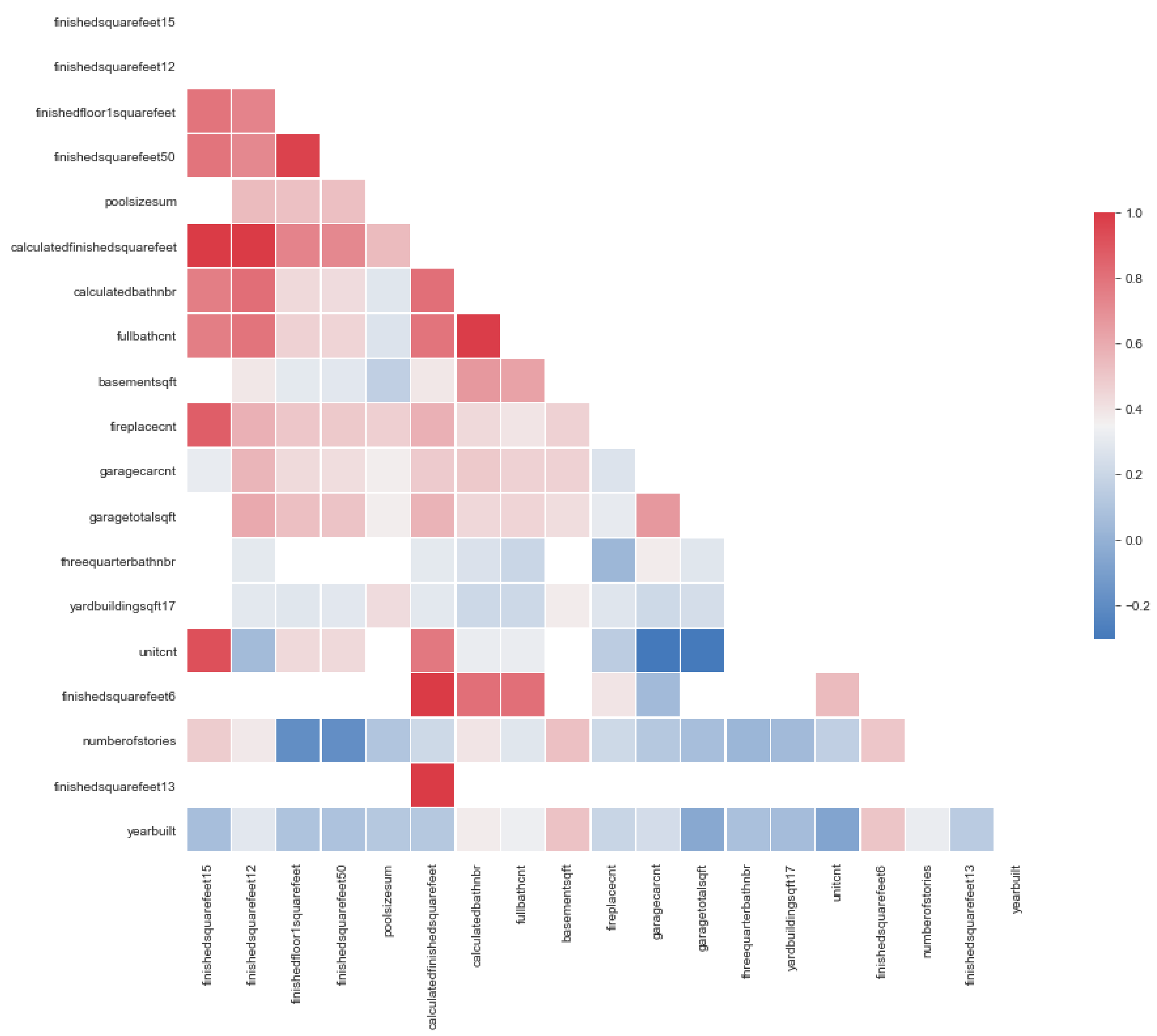

Fig. 3: Correlation matrix for the variables missing values