An auto-cleaning layer for data management systems

Joana M. F. da Trindade
MIT CSAIL
jmf@csail.mit.edu

Tarfah Alrashed
MIT CSAIL
tarfah@mit.edu

Shruti Banda
MIT SDM
bandas@mit.edu

ABSTRACT

Real-world data tends to be incomplete and inconsistent due to data integration, modification, and exchange. Data cleaning is one of the largest tasks conducted by data scientists due to its complication and inefficiency. An important task when preprocessing the data is to fill in missing values and correct the inconsistencies. There are a number of approaches for dealing with missing values, most of them are domain specific and hence they cannot be applied efficiently universally to different types of datasets. In this project, we propose a machine-learning based approach to automatically perform cleansing tasks for a number of common data errors. Specifically, we evaluate the performance (as cost to train) and prediction quality of different supervised-learning classification and regression algorithms over 10 real-world datasets. We find that training a single-column prediction model for a dataset with over a few thousands of rows can be done in less than a few seconds. Finally, in datasets with strong functional dependencies, our automated data cleaning approach provides highly accurate suggestions, reaching over 0.9 AUC for classifiers and R^2 score for regression algorithms.

1. INTRODUCTION

Both enterprise-grade and publicly available datasets frequently need to be cleaned prior to use due to the presence of errors. In fact, data cleaning is one of the largest tasks conducted by data scientists, often following an 80-20 rule where 80% of the time of data analysis is spent on "janitorial" work [1].

We believe cleaning takes so long because data errors come in many shapes (missing values, mixed formats, typos, and duplicate entries), and the state-of-the-art in automated cleaning systems is lacking. To address some of these issues, we proposing and build a system that automates the correction of a subclass of errors present in dirty data. Specifically, we look at how to correct missing values in datasets using supervised learning machine-learning models that predict values in a column as a function of other columns in the dataset.

In this report, we present our findings obtained during the process of analyzing and comparing the efficiency of five – originally seven, when SVM and SVR are included – supervised-learning algorithms for building single-column prediction models. We measure their execution time and prediction quality over real-world datasets and explain the algorithms' behavior at a high-level. Furthermore, we provide suggestions which algorithm to use when both runtime performance and prediction quality are important. Overall, we were able to confirm experimental results on prediction quality evaluation [30], where available.

Contributions. We give an overview on state-of-the-art developments in data cleaning as it relates to missing value imputation. We design and implement a system that uses supervised machine learning to automatically build single-column prediction models. We select five supervised-learning algorithms for building prediction models, and describe at a high-level how they work. We build prediction models using these algorithms over all columns of 10 different real-world datasets, and evaluate them in terms of both runtime and prediction quality. From our experimental results, we derive suggestions on which algorithm to use, and whether classical machine-learning optimizations are worthwhile when runtime performance and quality trade-offs are considered.

Structure. Section 2 gives an overview of data cleaning in general, discussing common concepts, and alternative approaches. Section 3 describes the design and implementation of our system. Section 4 presents our evaluation results. Section 5 discusses related work in more detail. Finally, Section 6 discusses our findings, and presents further avenues for future work.

2. PRELIMINARIES

The importance of data collection and analysis has increased tremendously in the past few decades, yet data quality remains a pervasive problem. The presence of incorrect or inconsistent data can significantly distort the results of analyses, often negating the potential benefits of information-driven approaches [24]. Errors in data could be of various types such as missing values, typos, mixed formats, replicated entries of the same real-world entity, and even violations of business rules [23, 24].

How data scientists deal with missing values varies depending on their datasets and the problems that they are trying to solve. Several approaches that deal with missing data simplify the problem by discarding the data instances with some missing values [2], by either removing an entire row, or an entire column. Other approaches consist of imputing missing values. These methods keep the full sample size, which can be advantageous for bias and precision; however, they can yield different kinds of bias [3]. Data cleaning in database-related tasks has been a critical issue, thus, several database systems and frameworks have been developed to tackle this problem [21, 22, 31] ranging from domain-specific data cleaning frameworks to more complex systems that deal with different types of errors in which databases often encounter.

2.1 Missing Value Imputation

Many approaches have been used to impute missing values. The simplest approach and least effective one is to discard the tuples with missing values, another approach is to fill in the missing values manually, which is time consuming and may not be feasible given a large dataset with many missing values. Some of the statistical approaches that have been used for missing data imputation [6] are using a global constant to fill in the missing values and/or replacing missing values with random attribute values, another approach is to replace missing values with the attribute mean/median [7]. Apart from using single imputation methods such as mean imputation, multiple imputation techniques have been used [9], such as multiple imputation by chained equations (MICE) [10]. In addition, a number of techniques have used machine learning algorithm, such as random forest, classification trees, k-nearest neighbors, logistic regression, support vector machines, and others to detect and predict missing data [9, 12], though most existing work in this area has only looked at models for a single data domain.

2.1.1 Outliers as Missing Values

Data cleaning include cleaning data from missing values and outliers. Outliers could include data errors such as typos, mixed formats, replicated entries of the same real-world entity, and even violations of business rules. Different approaches have tackled one or more of these issues with data cleaning [16, 18].

We have focused the evaluation of our prediction-model based data cleaning approach on missing values imputation. Nevertheless, we believe that our solution can also be trivially extended to deal with outliers. The idea here is that, since our trained models predict all values for a given target column, once outliers have been detected or labeled, they too can be cleaned using the predictions from the same model.

3. SYSTEM DESIGN

In this section, we introduce the architecture of our auto-cleaning software, how users would interact with the system, and how it would react toward their feedback. We also introduce the flow of how our prediction models impute missing data, and show how it could be extended to outlier detection. Finally, we give an overview of all the classification/regression prediction models that we used as part of our system.

3.1 Architecture

Figure 1 depicts the architecture for a DBMS that uses our data cleaning software layer. We envision that this could be done for multiple DBMS if our layer is implemented as a wrapper around JDBC. Our data cleaning layer is also generic enough that it can integrated with any tabular data management systems, since it does not rely on any specific features of DBMSes.

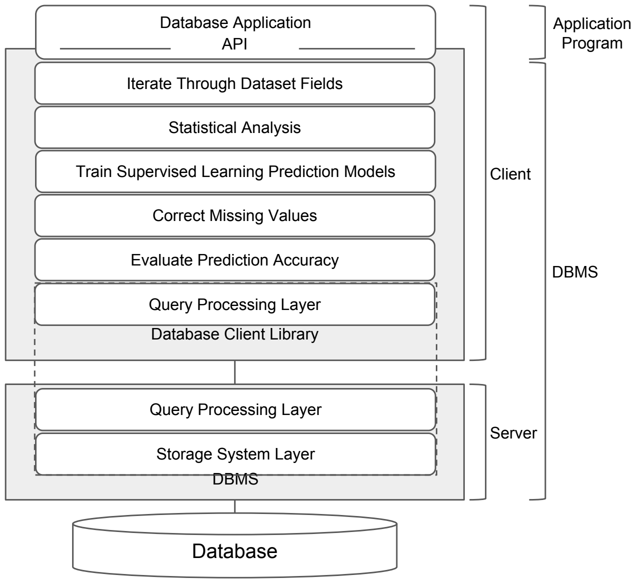


Figure 1. A DBMS architecture that integrates our data cleaning system in its client layer.

3.2 User Interaction

It has been shown in the literature that integrating user feedback in the data cleaning process can improve the performance and efficiency of the automatic data cleaning techniques [20]. Our vision for our auto-cleaning model is to allow users to feed the systems some specifications that could help accelerate the performance of our model. Users should also be allowed to accept/reject the prediction provided and suggested by our model if its accuracy does not satisfy their needs. Accordingly, other statistical predictors (i.e. mean imputation) could also be offered as an alternative to our prediction model.

3.3 Data Pre-Processing

Machine Learning algorithms for predicting values vary based on whether the data being is predicted is categorical or numerical, e.g., logistic regression is used for categorical value classification, while linear regression is used to predict numerical values. To facilitate this choice, our data cleaning system performs data pre-processing to collect various statistics for each of the columns in the dataset. For a data column which is numerical and continuous it uses regression algorithms, while for numerical discrete and categorical columns it uses both regression and classification algorithms to understand which of those methods performs better.

Our system also looks at aggregate statistics such as scaled variance and standard deviation (numerical columns), and the number of unique values (both numerical and categorical columns). These statistics may be useful in further helping the system decide which ML algorithm to use for a particular column. For example, a categorical column with a large number of unique values implies that there are too little number of instances per class, in which case a classification algorithm is likely to provide poor accuracy.

Another aspect of these datasets that can be further examined to determine – and potentially minimize – the best set of features to use when training a column prediction model is the correlation between pairs of columns. Specifically, correlation coefficients can be obtained for the actual values in the columns, and alternatively for the lack thereof (missing value, or otherwise "nullity" correlation). Figure 2 below shows an example nullity correlation heatmap obtained for one of the datasets we use in our evaluation.

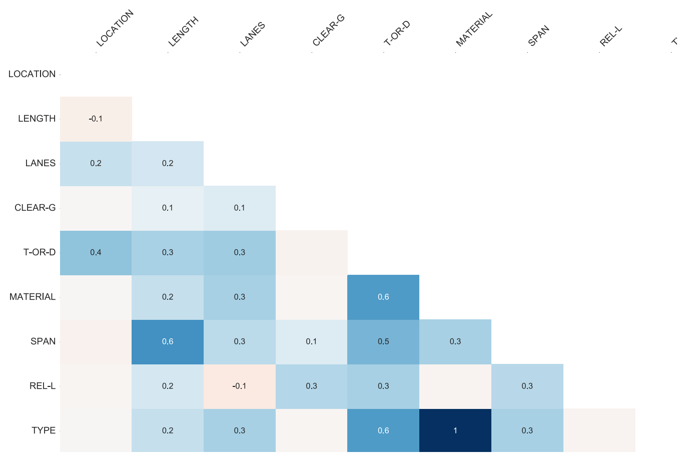


Figure 2. Nullity correlation heatmap for the *bridges* dataset.

While we have implemented ways to assess and visualize column correlations in our datasets, we have not fully explored correlation heuristics during our evaluation when training our supervised learning prediction models. We leave as future work the comparison of accuracy obtained via sequential feature selection – the technique we use in our evaluation – and correlation-based heuristics to prune the feature set space for training models over the data set.

3.4 Outlier Detection

Although we focus on missing data imputation as our primary use case, we note here that the same prediction models trained by our system are also applicable to correction of outliers. The idea is that, so long as outlier values identified or otherwise labeled in the input dataset, predictions from the model can be used to replace them. In fact, since the prediction models in our system are trained over the fraction of data that is considered clean, they can be used to correct virtually any sort of data deemed incorrect by the user, or by existing data error detection algorithms such as those in [16, 17, 32].

3.5 Missing Value Imputation Algorithm

Datasets might contain missing values in any of their columns and our system is designed to correct any such occurrences in any column. For a given dataset, our system iterates through the entire dataset and for each column:

- Determines whether the column is categorical or numerical;
- Finds the data type and aggregate statistics for the column;
- Uses this column as the target variable and all the other columns as independent variables (AKA features);
- Trains a prediction model using supervised machine learning classification algorithms (for categorical or numerical discrete columns), or regression algorithms (for numerical continuous column);
- Uses the model built in the above step to predict the missing values in that column, picking the one with the highest accuracy (classification) or R^2 score (regression).

We currently predict missing values based on the model that is trained using available and complete data, only consider supervised learning algorithms.

3.5.1 Classification

Several machine learning algorithms have been used to predict categorical values. Since searching through the entire algorithms space will be time consuming, we limit ourselves to three algorithms. We choose these algorithms based on running experiments on different datasets, and picking the three algorithms with the best accuracy, while within the reasonable cost to train over data. Our experimental evaluation consists of micro-benchmarks to evaluate the performance of different machine learning classification algorithms in terms of how fast they are as a function of the size of our training and validation datasets, and how accurate they are at the prediction of values for missing data. For evaluating the accuracy of missing value prediction, we plan to use Prediction Accuracy AUC (Area Under the Curve), which plots the fraction of true positive rate over false positive rate. The classifiers with highest AUC scores were chosen for building the system.

The three algorithms chosen were Multinomial Logistic Regression, Decision Tree classifier and Random Forest classifier. These algorithms consistently performed better than other algorithms like Support Vector Machines (SVM) for the datasets we tested. The goal of the supervised classification algorithms is to leverage a set of training samples in order to design a classifier that is capable of distinguishing between the different classes of a column on the basis of the values in the independent variables.

In logistic regression Probability or Odds of the response taking a particular value is modeled based on combination of values taken by the predictors. Multinomial Logistic Regression [27] estimates the probability of the missing value belonging to a particular class.

A Decision Tree is another powerful classification technique. The tree infers the split of the training data based on the values of the available features to produce a generalized model. It is split at each node based on the feature that gives the maximum information gain. Each leaf node corresponds to a class label. A new example is classified by following a path from the root node to a leaf node, where at each node a test is performed on some feature of that example. The leaf node reached is considered the class label for that example. This algorithm can handle both binary or multiclass classification problems. The leaf nodes can refer to either of the K classes that are present in that particular attribute being predicted [28].

Random Forest (RF) is a classification method based on the aggregation of a large number of decision trees. It is a combination of trees constructed from a training dataset and internally validated to yield a prediction of the (missing) response variable. In an RF model, each tree is a standard classification or regression tree (CART) that is built using Gini impurity (DGI) as a splitting criterion and selects the splitting predictor from a randomly selected subset of predictors (the subset is different at each split). Each tree is constructed from a sample drawn with replacement from the original dataset, and the predictions of all trees are finally aggregated through majority voting [29].

Finally, all classifiers used by our auto-cleaning system are calibrated using Platt scaling. We have adopted this calibration step inspired by literature in evaluation of different supervised learning algorithms [30], and after verifying that it indeed reduces overfitting and increases accuracy over test data.

3.5.2 Regression
Similar to classification, we selected two regression algorithms in terms of (i) how fast they are as a function of the size of our training and validation data sets, and (ii) how accurate they are at detection and prediction of values for missing data.

The two regression algorithms chosen for the auto cleaning database are Random forest regression and Linear regression.

Similar to the Random forest classifier, in a RF regression tree, since the target variable is a real valued number, a regression model is fit to the target variable using each of the independent variables. Then for each independent variable, the data is split at several split points and Sum of Squared Error(SSE) is computed at each split point between the predicted value and the actual values. The variable resulting in minimum SSE is selected for the node. and the process is recursively continued through the entire training data set. This model can then be used to predict the real missing values in the target column.

In order to reduce the number of independent variables that are considered for building either a classification or regression model, we use sequential feature selection method that improves the performance on the dependent measures. In this system we implemented a sequential feature selection the search attributes to half the total number of attributes in the dataset being considered. The algorithm iterates through all the attributes and selects the ones that minimizes the mean squared error (in the case of regression) and maximizes accuracy (in the case of classification), and thus improving the quality of predictions. As we later show in the evaluation Section, there is certainly a performance (cost to train) vs prediction quality trade-off when using sequential feature selection. Hence, we evaluate versions of our models with, and without SFS enabled.

4. EMPIRICAL EVALUATION
Our goal with the experiments in this project was two-fold. First, to answer the question "how long does it take to train a prediction model for a given column on a modern server?". This is almost as important as prediction quality, given that the idea is to use these predictions on a DBMS. If the cost to train such models is deemed prohibitive by the user, then even an accuracy of over 90% may be worthless. Second, we also would like to see how good are the predictions made by models without a priori domain-specific knowledge on the dataset, or otherwise manual intervention. To put another way, we want to tell whether our "unsupervised" supervised learning approach fares better than random suggestions and mean/mode imputation.

We initially restricted our supervised learning models to those considered the best performant in terms of prediction quality. In particular, we have started this project with the following set of candidates. For classifiers, we had initially chosen Decision Tree, Logistic Regression, Random Forest and SVM. For regression-based models, we had picked Linear Regression, Random Forest Regression, and SVM Regression. Due to the sub-par performance of SVM-based models – covered in more details on Section 4.3 and the Appendix – we restrict our set of models to the aforementioned ones, minus SVMs.

Note that it is not in the scope of this project to conduct an in-depth evaluation of the accuracy of supervised-learning models in general. For that, we refer the reader to [30], in which the authors conduct an extensive set of experiments to try and measure the maximum possible average accuracy of training supervised-learning classifiers over a set of datasets. Some of the techniques we have used in this project – namely calibration and sequential feature selection – however, have been inspired by that work. While we cover less classifiers than the authors do in [30], we also evaluate the cost to train the models, which they do not. The reason why we measure this attribute is because we believe that measure would be essential when our trained prediction models are used in the context of a DBMS.

4.1 Experimental Settings

Datasets. We evaluate the performance and quality of training supervised learning prediction models on 10 real-world datasets. These datasets have been selected from the literature on evaluation of functional dependency discovery systems [18]. They describe data from a gamut of application domains and provide categorical, numerical continuous, and numerical discrete columns. Furthermore, they all originate from the UCI machine learning repository, have between 100 and 33000 rows, and between 5 and 30 columns, as shown in Table 1. The datasets *abalone*, *balance-scale*, *echocardiogram*, and *hepatitis* contain anonymized, clinical data on diseases and patients observed under the respective studies. The *bridges* dataset has data on bridges located in Pittsburgh, PA. Dataset *iris* contains data on iris plants. The *chess-krk* dataset contains King and Rook versus King end-game chess configurations. The *nursery* dataset has information on applications to nursery schools. The *letter* dataset contains data on the English alphabet used for character recognition. Finally, the *adult* dataset has aggregate data on census conducted in the US.

Table 1. Structural properties of real-world datasets used for evaluation of the prediction models.

Dataset	Columns	Rows	Size	Has MV?
abalone	9	4177	187 K	N
adult	14	32562	3.8 M	Y
balance-scale	5	625	6.2 K	N
bridges	13	108	6.4 K	Y
chess-krk	7	28056	519 K	N
echocardiogram	13	131	6 K	Y
hepatitis	20	155	7.5 K	Y
iris	5	151	4.5 K	N
letter	17	20000	696 K	N
nursery	9	12961	1 M	N

Missing Values. Our system considers any cell with a NULL, NaN, '?' as a missing value. For the real-world datasets that contain missing data, Figures 10 through 17 in the Appendix depict the fraction of missing values in each column, as well as a nullity correlation heatmap. In the four datasets with missing information, four of the datasets have data missing in more than half of their columns. Because we train our prediction models on the fraction of rows without missing values, we evaluate performance in terms of rows left after rows with missing data have been removed.

Hardware. All experiments are conducted on a Linux server running Ubuntu Trusty 14.04. The machine contains 8 Intel Xeon E5-2360 v4 (2.20 GHz), and 16 GiB DIMM RAM. All supervised learning classification and regression prediction models are trained using a single threaded configuration, except in cases when the machine learning library allows for more CPUs to be specified, e.g., during cross-validation and sequential feature selection. For Python environment, we use Python 2.7.12 as provided by Anaconda 4.1.1, and Python machine learning library scikit-learn 0.18.1.

Machine Learning Parameters. For each dataset, a 70-30 split is used to obtain training and test data after removal of rows with missing values. We use 5-fold cross-validation whenever there are at least 100 training data rows, and 2-fold otherwise. In addition, we calibrate the classifiers using Platt scaling method (aka 'sigmoid' in Python's scikit-learn machine learning library). With the exception of tree-based models, which are capped at 1024 max levels, and 15 estimators in the case of random forests, all other models use the defaults provided by scikit-learn. Finally, we evaluate both performance and quality of each prediction model before and after using a Sequential Forward Selection (SFS) feature selection algorithm, as made available by Python's mlxtend library. We limit the feature search space for SFS to $\lceil$columns / 2$\rceil$ features per dataset.

4.2 Performance

In our first set of experiments, we evaluate the runtime cost (in number of seconds) of training supervised learning algorithms on each column of the input datasets. Cost to train here is measured as the total number of seconds it takes to fully train one of the supervised learning algorithms under consideration. Since there are several such measurements – one per each column trained – we display the 99th, 95th and 50th (median) percentiles of performance for each of the algorithms. While it is difficult to pinpoint a single winner given that these are distributions, we notice that with the exception of Logistic Regression classifier (*LogRC*), all classifiers and regression-based algorithms take less than 5 seconds in the worst case when sequential feature selection is disabled. Conversely, when SFS is enabled, the models take much longer, though again Logistic Regression is the slowest by orders of magnitude. In addition, when SFS is enabled, Decision Trees and Linear Regression are trained in significantly less time than the other available models.

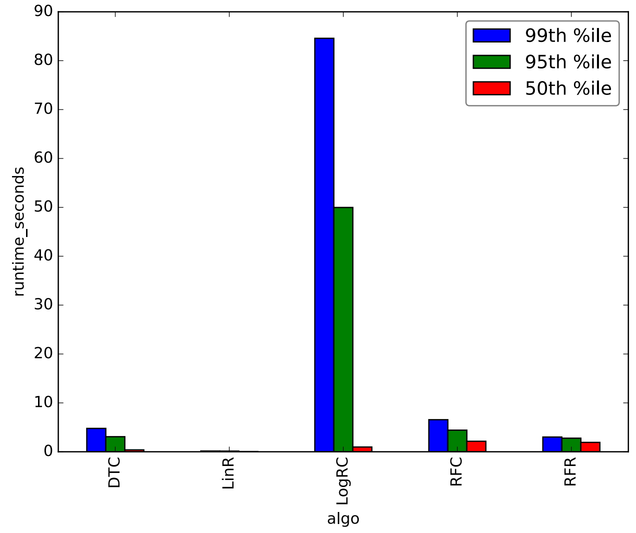


Figure 3. Cost to train a supervised learning prediction model for one of the columns in the dataset, with sequential feature selection disabled.

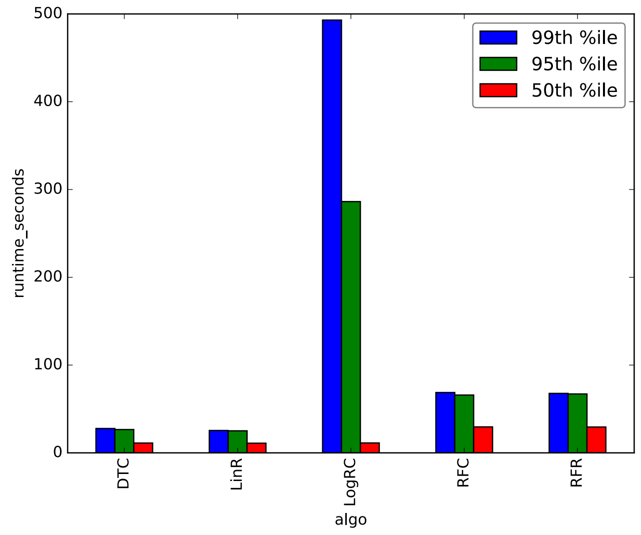


Figure 4. Cost to train a supervised learning prediction model for one of the columns in the dataset, using sequential feature selection.

4.3 Quality

In this experiment, we want to see what is the quality of predictions for each of the models. We use a 70-30 split for training and test data. We use 5-fold cross-validation whenever there are at least 100 training data rows, and 2-fold otherwise. In addition, we calibrate the classifiers using Platt scaling method (aka 'sigmoid' in Python's scikit-learn machine learning library).

We evaluate quality using standard machine learning measures, namely AUC accuracy for classifiers, and R^2 score for regression-based models. Although we originally included SVM-based classifier and regression-based prediction models in our evaluations, their performance was consistently sub-par – up to 2 orders of magnitude slower to train than the other models under consideration, and consistently less accurate. Because of this, we omit SVM and SVR models from our results, and instead make a high-level comparison available in Figure 9 in the Appendix.

4.3.1 Classification

Figures 5 and 6 below depict our results. The classifiers under consideration are LogisticRegression (LogRC), Decision Tree (DTC), and Random Forest (RFC). Surprisingly, the accuracy of the classifier models after applying SFS is reduced in some cases, e.g., the *hepatitis* dataset. While the accuracy of each class of prediction models varies significantly with each dataset, Random Forest prediction models rank at the top for most datasets.

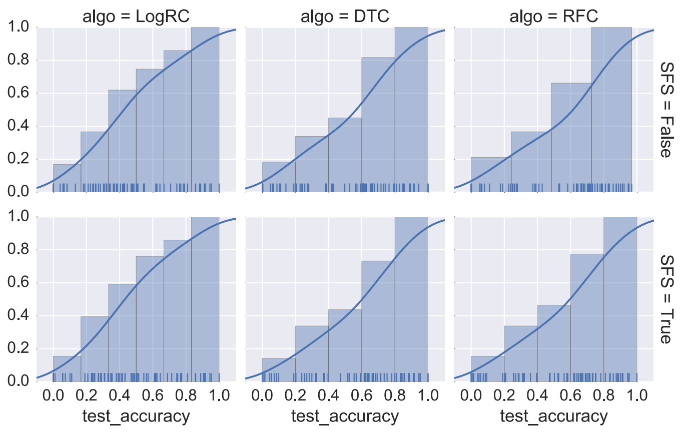


Figure 5. CDF of accuracy on test data for Logistic Regression (LogRC), Decision Tree (DTC), and Random Forest (RFC) classifiers on all columns from all of the 10 datasets.

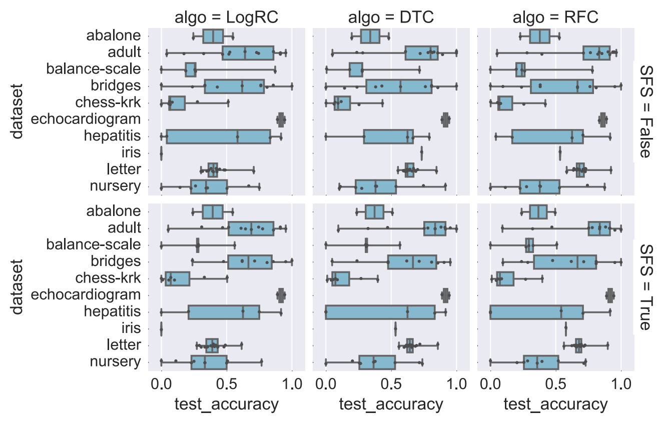


Figure 6. Accuracy on test data for Logistic Regression (LogRC), Decision Tree (DTC), and Random Forest (RFC) classifiers on each of the 10 data sets.

4.3.2 Regression

In Figures 7 and 8 we present our results. The regression-based algorithms under consideration are Linear Regression (LinR), and Random Forest Regression (RFR). While the prediction quality – as measured by R^2 score – of each class of prediction models varies significantly with each dataset, Random Forest Regression prediction models rank at the top for most datasets.

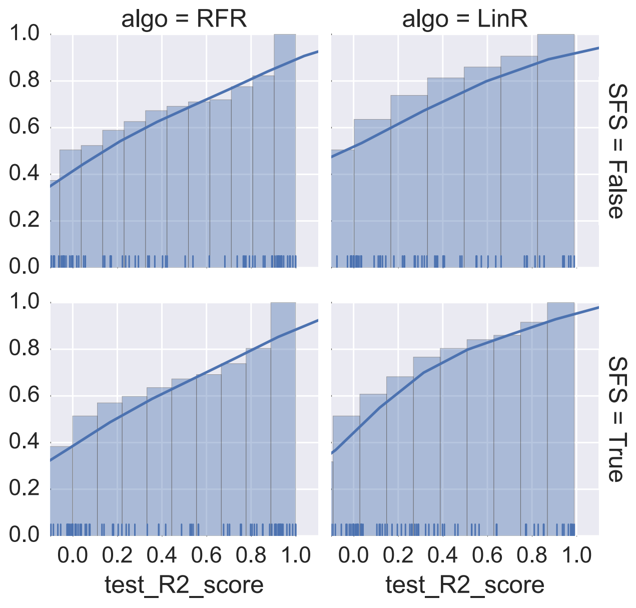


Figure 7. CDF of accuracy on test data for Random Forest Regression (RFR), and Linear Regression (LinR) regression-based models on all columns from all of the 10 datasets.

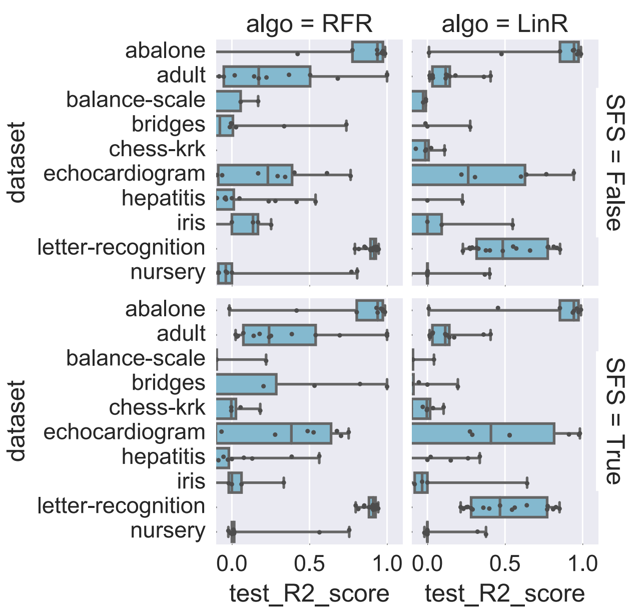


Figure 8. R^2 score on test data for Random Forest Regression (RFR), and Linear Regression (LinR) regression-based models on each of the 10 data sets.

4.4 Interpretation of Experimental Results

We start our discussion by noting that there a machine-learning – or any column correlation-based approach – is only as good to the extent that independent columns ("features") are predictive of the dependent column ("target"). For example, single-column prediction models built for all columns depicted in Table 2 had terrible accuracy. As one can expect, there are not that many features that can predict "age" in a clinical study of hepatitis other than the "age" column itself.

Table 2. The 4 columns where classification prediction models obtained the worst accuracy over test data.

Classifier	Dataset	Column Name	Accuracy
LogRC	hepatitis	AGE	0.0
DTC	hepatitis	AGE	0.0
RFC	hepatitis	AGE	0.0
LogRC	chess-krk	Black-King File	0.056678
DTC	chess-krk	Black-King File	0.056678
RFC	chess-krk	Black-King File	0.056678
LogRC	chess-krk	Black-King Rank	0.010100
DTC	chess-krk	Black-King Rank	0.010100
RFC	chess-krk	Black-King Rank	0.010100
LogRC	bridges	ERECTED	0.238095
DTC	bridges	ERECTED	0.238095
RFC	bridges	ERECTED	0.190476

During our evaluation, we have also obtained a few surprising results. The first one was that SVM-based models – at least in our experiments – take substantially more time to train than any of the other non-SVM classification and regression algorithms we considered. In fact, it was up to 2 orders of magnitude slower than Random Forest models in the worst case.

Another surprising result was the relatively poor trade-off between performance and prediction quality offered by using SFS based sequential feature selection. Overall it is fairly expensive in terms of cost to train given the observed marginal gains in R^2 score for regression-based models in some of our datasets. Not only that, but for some of classification algorithms, employing SFS has actually decrease their accuracy over test data in some of the datasets. We posit that perhaps this may be due to characteristics of our datasets – all of which obtained from the UCI machine learning repository – that may already offer columns that can be predicted by a small number of features without overfitting. Hence, applying sequential feature selection would not significantly change the prediction model.

Based on our findings, and on the fact that we are equally interested in prediction quality as we are in cost to train the models, we recommend skipping sequential feature selection, and choosing Random Forest-based models for both classification and regression. These models consistently offer less variance in their prediction quality than other models, and as our performance experiments indicate, they are fairly inexpensive to train.

5. RELATED WORK

Data Cleaning in DBMS. Data cleaning is a critical problem in many database-related tasks. A number of approaches have been proposed, which use machine learning algorithms for data cleaning and missing data imputation [19], some of these approaches tried to improve the accuracy of the algorithms by integrating users' feedback and evaluation [20]. Several approaches have been proposed for specific domains, one of these is ActiveClean [21], which allows for iterative cleaning in statistical modeling problems and supports some machine learning models such as linear regression and SVMs. Furthermore, some approaches have proposed models for data cleaning that it is general enough to work efficiently and guarantee consistency with different types of dataset with distinct constraints, LLUNATIC is one of these approaches [22]. Others, like BigDansing have tackled the scalability and efficiency issues in data cleaning process [31]. Unlike these approaches, however, our system is not limited to a single domain-specific dataset or single prediction model. In fact, we have evaluated the accuracy and performance of our models in datasets that cover a wide range of application domains. Our system's "unsupervised" way of picking the best supervised-learning prediction model for a given column lets the domain-expert decide whether to accept or reject the suggestions.

Missing Value Imputation. Several statistical methods have been proposed for missing value imputation [4, 5, 6]. The simplest approach is to discard tuples where any attribute is missing. Unfortunately, this method usually performs poorly in datasets where a single column has a large fraction of missing values – since most rows are discarded in that case. Another approach is to have a domain expert manually fill in a missing value, which is time consuming and may not be feasible at all in large datasets. Other single imputation techniques include using global constants, random values, and mean/mode for the distribution of values in the target column [7]. The latter, of course, are limited in their applicability to columns with only a small fraction of values missing. Magnani [8] reviews these and other missing data imputation techniques, showing how, even though they have been useful in survey data management, all of these techniques rely on strong model assumptions.

To overcome some of the drawbacks in single imputation methods, multiple imputation techniques have been proposed [9]. The main idea behind multiple imputation is in reusing values previously imputed for other columns, and repeating this process multiple times to reduce errors introduced due to the order in which columns are imputed. Specifically, each variable with missing data is modeled conditionally using the other variables in the data before filling in the missing values. Because values are reused for the next column imputation process, the order in which columns are imputed introduces biases and significantly affects the final accuracy of values imputed this way. To address this shortcoming, the Multiple Imputation by Chained Equations (MICE) method [10] performs several trials in which the order of columns to impute is randomized. This approach is particularly powerful when features have strong correlation, yet several complications are encountered when actually implementing MICE, particularly with large datasets, including model selection and computing limitations [11]. Research has been conducted to evaluate machine learning algorithms for missing value imputation [9, 12, 13], and studies have shown that machine learning algorithms tend to outperform statistical methods [12, 14]. Most of these studies are conducted over domain specific data, require significant intervention from machine-learning experts, and may not work as efficiently over datasets from other domains. During the course of our project, we have tried to keep our methods as domain-agnostic as possible. Hence, our approach relies only on structural properties of the input dataset (column data type, number of instances, etc) to pick the supervised learning prediction model that provides the best accuracy / R^2 score for that column.

Outlier Detection. A large body of work exists that proposes ways to identify data outliers using statistical methods, though most work in that area is better suited for numerical data [16]. To overcome this limitation, an approach has been developed that leverages tuple expansion to detect outliers in both numerical and heterogeneous datasets [17]. A more recent survey [32] evaluates the precision and recall of different outlier and data error detection methods, and shows that – as always – no single silver-bullet exists. We note here that all of these approaches for outlier and error detection are complementary to the work we have presented in this report. Furthermore, labels produced by outlier detection algorithms can be used to guide the training of prediction models in our system, and the predictions in turn can be used to correct outliers or other classes of errors in the input data.

Functional Dependency Discovery for Data Cleaning. Another way to automate the process of data cleansing is to find the underlying functional dependencies in a dataset, and use that to correct any errors – which are found when rows break one of the discovered functional dependencies. In the database community, several methodologies for mining functional dependencies have been proposed [15, 18]. These methods usually build the lattice of columns for a given dataset, and check whether any correlation exists between those and a target set of dependent columns. Because the size of the lattice is exponential on the number of columns – 2^n, as it considers all subsets of columns – these methods usually suffer from poor runtime and/or memory performance [18]. Our supervised-learning prediction models, when accuracy is above a certain threshold, can also be seen as a way of finding functional dependencies. Since these prediction models do not explore the entire lattice of features, however, their runtime performance is significantly better. It is not clear, however, how the robustness of the column correlations they find compares to those found by functional dependency discovery systems, and we leave this as future work.

6. CONCLUSION AND FUTURE WORK

In this project, we presented an auto-cleaning layer for data management systems. It trains single-column prediction models using the supervised learning classification or regression algorithm that provides the highest prediction quality for that column. We evaluate the accuracy and performance of different such algorithms for the problem of missing value imputation, and find that calibrated random forests – both in its classification and regression forms – consistently rank at the top in terms of runtime performance and prediction quality. Unlike previous work on machine-learning for missing value imputation, our system is not restricted to a single application data domain, and does not rely on any domain-specific knowledge of the input dataset. Furthermore, while we suggest an architecture for using it as part of the client-side layer of a DBMS, e.g., as a JDBC wrapper, our auto-cleaning software system can be integrated into any tabular data management system.

We propose and discuss several promising future research directions:

(1) Trade-offs for data cleaning in higher vs lower levels of the data management system software stack. While we suggest our data cleaning software be used as part of the client-side API of a DBMS, it would be interesting to see what are the trade-offs that exist in terms of client-side vs server-side implementations. For example, what are the indexing data structures and otherwise physical data layout techniques that can be leveraged when cleaning is performed at the lowest layers of the DBMS? In the case of column-oriented data management systems, how can the cleaning layer take into account existing compression schemes in the raw data?

(2) An user interaction model for auto-cleaning. The main goal of this system is to automate as much as possible the laborious task of data cleaning so that data analysis can focus their time on conducting actual data analysis. Therefore, it is crucial that users of this system have an appropriate interaction model that allows them to (i) partially accept or rollback predictions, (ii) specify thresholds for performance and prediction quality, and (iii) label known outliers or data errors in efficient ways.

(3) Comparison with existing functional dependency discovery systems. While it is clear that a single-column prediction model's output can be used as a functional dependency, we have not conducted a comparison between the column correlations found by our prediction models, and the FDs found by TANE, FD_Mine, and others functional dependency discovery systems. At a first glance [18], our ML-based approach is faster, though further empirical evaluation is required to validate whether our functional dependencies would be as accurate as those from the FD discovery systems.

(4) Integration with ML-based outlier and error detection algorithms. Currently our system performs error correction, and only rudimentary missing value detection. Since our prediction models can be used for correcting any error, it would be interesting to see how well its error correction performs when actually integrated with outlier and otherwise machine-learning based error detection systems.

(5) Accuracy of single vs multiple imputation. While we initially randomly injected missing values using MCAR (Missing Completely At Random), the best prediction quality was obtained when data pre-processing removes rows with missing values. Possible future work here would be to investigate the accuracy of multiple imputation (i.e., reusing imputed values from previous columns) when using our prediction models.

7. REFERENCES

[1] Lohr, S. For Big-Data Scientists, 'Janitor Work' Is Key Hurdle to Insights. Retrieved October 10, 2016, from New York Times: http://www.nytimes.com/2014/08/18/technology/for-big-data-scientists-hurdle-to-insights-is-janitor-work.html.

[2] Friedman, J. H., Kohavi, R., & Yun, Y. 1996. Lazy Decision Trees. In Proceedings of the 13th National Conference on Artificial Intelligence, (1996, August), 717-724.

[3] Shawe-Taylor, J. and Cristianini, N., 2004. *Kernel methods for pattern analysis*. Cambridge university press.

[4] Han, J., Pei, J. and Kamber, M., 2011. *Data mining: concepts and techniques*. Morgan Kaufmann Publishers, 2006, 2nd edition.

[5] Zhang, S., Qin, Y., Zhu, X., Zhang, J. and Zhang, C. August. Optimized parameters for missing data imputation. In *Pacific Rim International Conference on Artificial Intelligence*, (2006) 1010-1016).

[6] Van Buuren, S. Flexible imputation of missing data. *CRC press*, 2012.

[7] Somasundaram, R. S., and Nedunchezhian, R. Missing Value Imputation using Refined Mean Substitution. *International Journal of Computer Science* Issues 9.4, (2012), 1694-0814.

[8] Magnani, M. Techniques for Dealing with Missing Data in Knowledge Discovery Tasks. Department of Computer Science, *University of Bologna*, 2004.

[9] Shah, A. D., Bartlett, J. W., Carpenter, J., Nicholas, O., and Hemingway, H. Comparison of random forest and parametric imputation models for imputing missing data using MICE: a CALIBER study. *In American Journal of* Epidemiology*, 179*(6), (2014, April), 764-774.

[10] Raghunathan, T.E., Lepkowski, J.M., Van Hoewyk, J. and Solenberger, P. A multivariate technique for multiply imputing missing values using a sequence of regression models. *Survey methodology*, *27*(1), (2001, June), 85-96.

[11] Stuart, E.A., Azur, M., Frangakis, C. and Leaf, P. Multiple imputation with large data sets: a case study of the Children's Mental Health Initiative. *American journal of epidemiology*, (2009, January).

[12] Jerez, J. M., Molina, I., García-Laencina, P. J., Alba, E., Ribelles, N., Martín, M., and Franco, L. Missing data imputation using statistical and machine learning methods in a real breast cancer problem. *Artificial intelligence in medicine* 50, no. 2, (2010), 105-115.

[13] Song, S., Zhang, A., Chen, L., and Wang, J. Enriching data imputation with extensive similarity neighbors." *Proceedings of the VLDB Endowment* 8, no. 11, (2015), 1286-1297.

[14] Schmitt, P., Mandel, J., and Guedj, M. A comparison of six methods for missing data imputation. *Journal of Biometrics & Biostatistics*, 2015.

[15] I. F. Ilyas, V. Markl, P. Haas, P. Brown, and A. Aboulnaga. CORDS: Automatic discovery of correlations and soft functional dependencies. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data - SIGMOD '04*, page 647, (New York, USA, June 2004). ACM Press.

[16] Hodge, V.J. and Austin, J. A survey of outlier detection methodologies. *Artificial intelligence review*, *22*(2), (2004, October), 85-126.

[17] C. Pit-Claudel, Z. Mariet, R. Harding, and S. Madden. Outlier detection in heterogeneous datasets using automatic tuple expansion. Technical Report MIT-CSAIL-TR-2016-002, CSAIL, MIT, 32 Vassar Street, Cambridge MA 02139, February 2016.

[18] Papenbrock, T., Ehrlich, J., Marten, J., Neubert, T., Rudolph, J.P., Schönberg, M., Zwiener, J. and Naumann, F. Functional dependency discovery: An experimental evaluation of seven algorithms. *Proceedings of the VLDB Endowment*, *8*(10), (2015), 1082-1093.

[19] Yakout, M., Berti-Équille, L. and Elmagarmid, A.K. Don't be SCAREd: use SCalable Automatic REpairing with maximal likelihood and bounded changes. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, (2013, June), 553-564.

[20] Yakout, M., Elmagarmid, A.K., Neville, J., Ouzzani, M. and Ilyas, I.F. Guided data repair. *Proceedings of the VLDB Endowment*, *4*(5), (2011), 279-289.

[21] Krishnan, S., Wang, J., Wu, E., Franklin, M.J. and Goldberg, K. ActiveClean: interactive data cleaning for statistical modeling. *Proceedings of the VLDB Endowment*, *9*(12), (2016), 948-959.

[22] Geerts, F., Mecca, G., Papotti, P. and Santoro, D. The LLUNATIC data-cleaning framework. *Proceedings of the VLDB Endowment*, *6*(9), (2013), 625-636.

[23] Ilyas, I.F.. Effective Data Cleaning with Continuous Evaluation. *Data Engineering*, (2016), 38.

[24] Hellerstein, J.M.. Quantitative data cleaning for large databases. *United Nations Economic Commission for Europe (UNECE), (*2008).

[25] Chen, Z. Computational intelligence for decision support. *CRC Press*, (1999).

[26] Janssen, R., Spronck, P., Dibbets, P. and Arntz, A. Frequency Ratio: a method for dealing with missing values within nearest neighbour search. *Journal of Systems Integration, 6(3),* (2015), 3.

[27] Krishnapuram, B., Carin, L., Figueiredo, M. A., & Hartemink, A. J. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE transactions on pattern analysis and machine intelligence*,*27*(6), (2005), 957-968.

[28] Aly, M. Survey on multiclass classification methods. *Neural Netw*, (2005), 1-9.

[29] Boulesteix, A. L., Janitza, S., Kruppa, J., & König, I. R. Overview of random forest methodology and practical guidance with emphasis on computational

biology and bioinformatics. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *2*(6), (2012), 493-507.

[30] Rich Caruana and Alexandru Niculescu-Mizil. 2006. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning* (ICML '06). ACM, New York, NY, USA, 161-168.

[31] Khayyat, Z., Ilyas, I.F., Jindal, A., Madden, S., Ouzzani, M., Papotti, P., Quiané-Ruiz, J.A., Tang, N. and Yin, S., 2015, May. BigDansing: A system for big data cleansing. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, 1215-1230.

[32] Ziawasch Abedjan, Xu Chu, Dong Deng, Raul Castro Fernandez, Ihab F. Ilyas, Mourad Ouzzani, Paolo Papotti, Michael Stonebraker, and Nan Tang. 2016. Detecting data errors: where are we and what needs to be done?. *Proc. VLDB Endow.* 9, 12 (August 2016), 993-1004.

Appendix

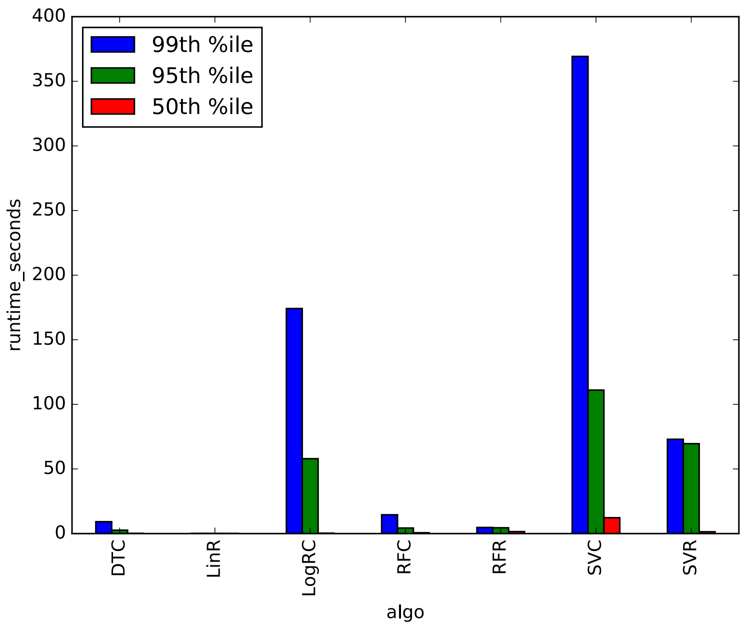


Figure 9. Initially obtained runtime for the different classification and regression algorithms, including SVM-based classification and regression (without SFS enabled). We have excluded these models from our subsequent evaluation experiments due to poor performance of SVM/SVR compared to that of other classification/regression algorithms.

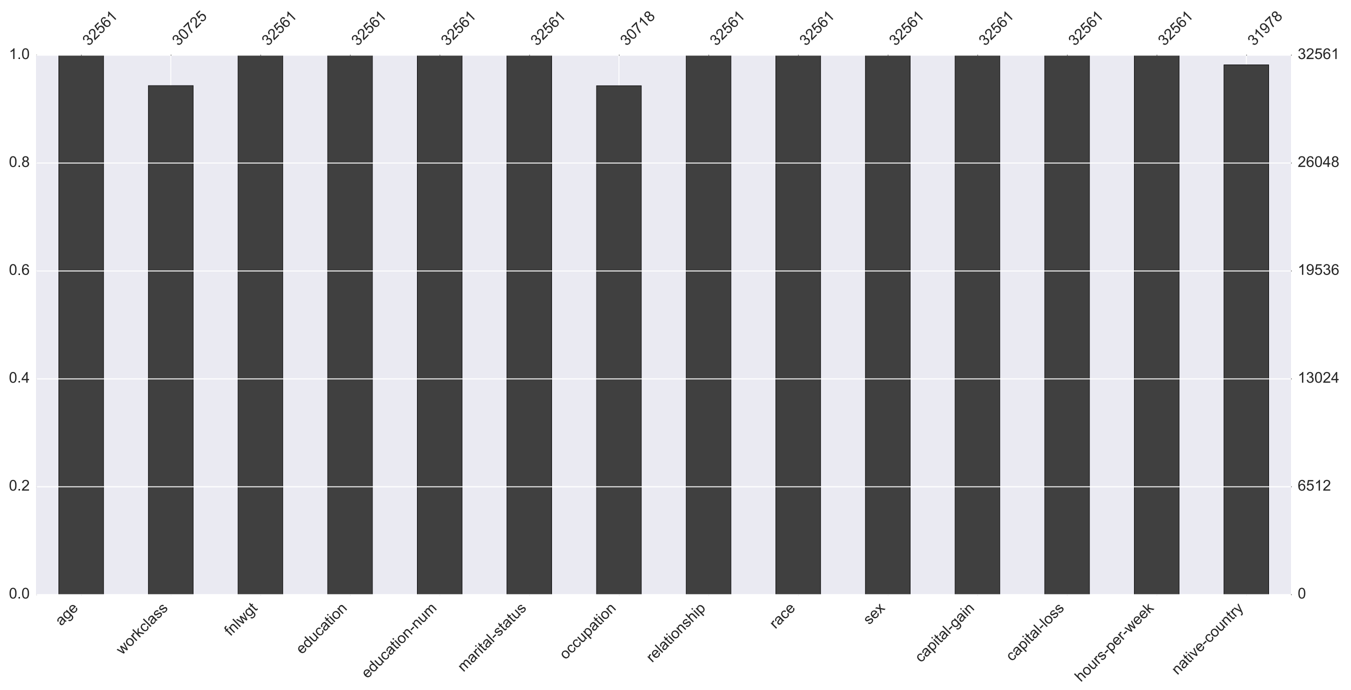


Figure 10. Fraction of nulls on each column for the *adult* dataset.

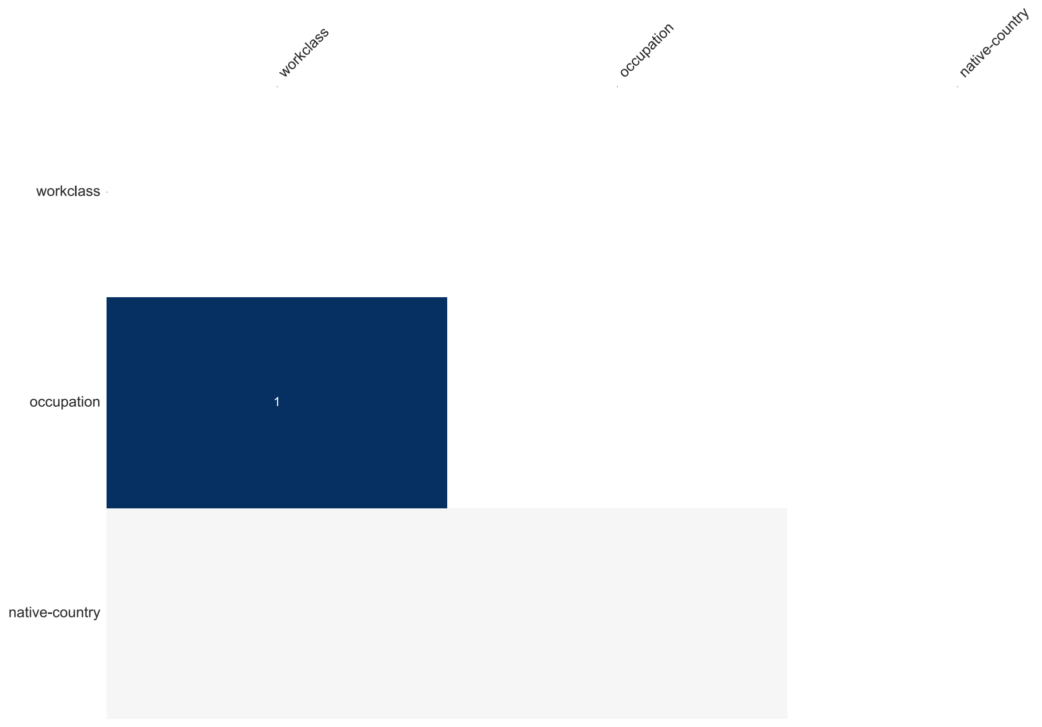


Figure 11. Nullity correlation between columns in the *adult* dataset.

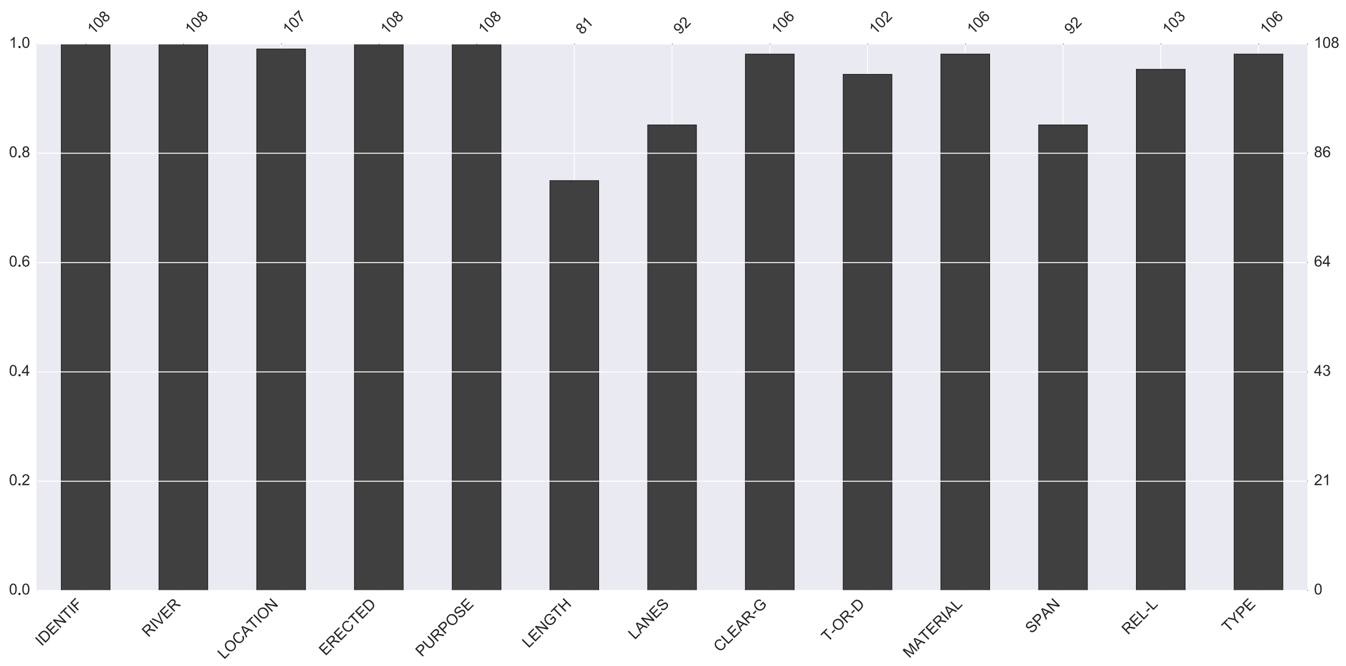


Figure 12. Fraction of nulls on each column for the *bridges* dataset.

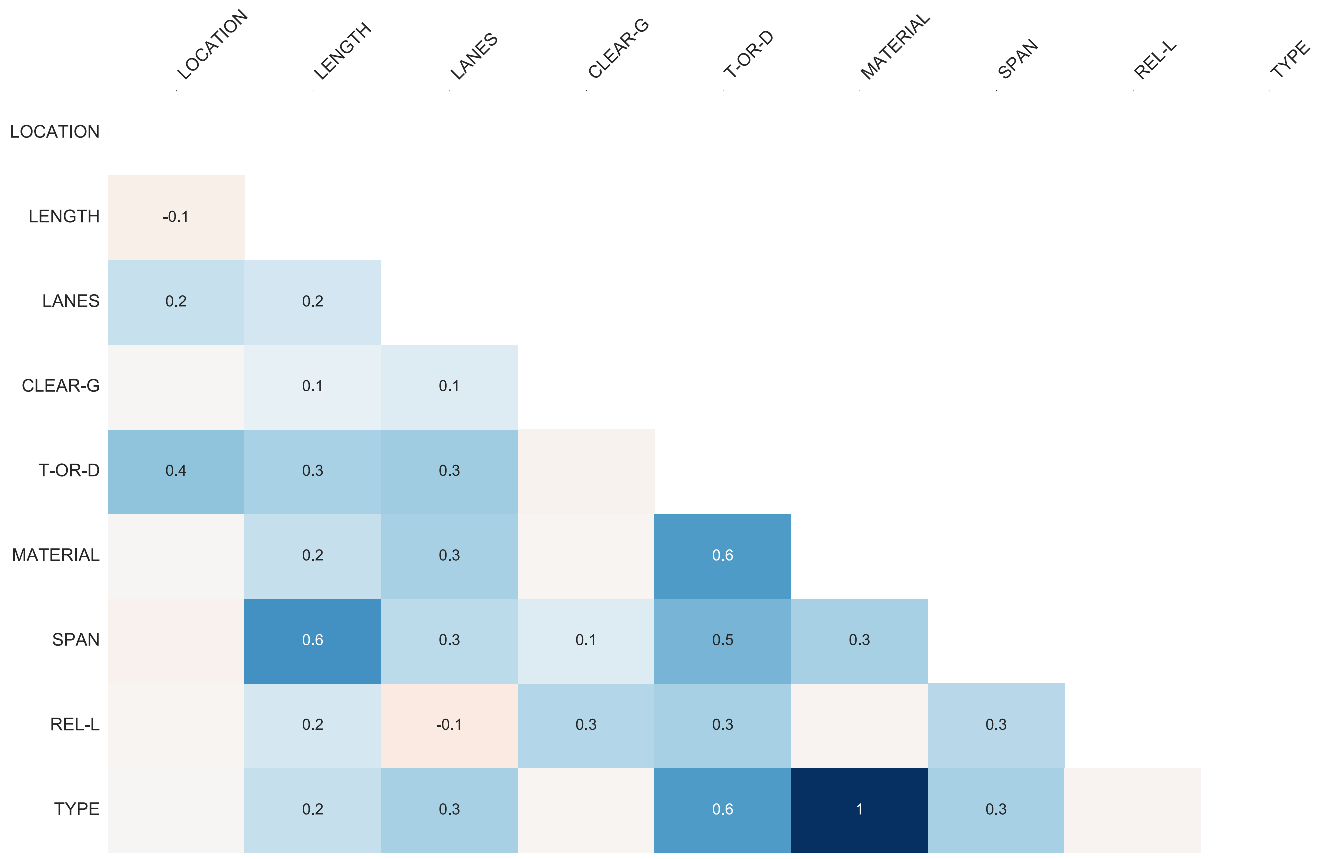


Figure 13. Nullity correlation between columns in the *bridges* dataset.

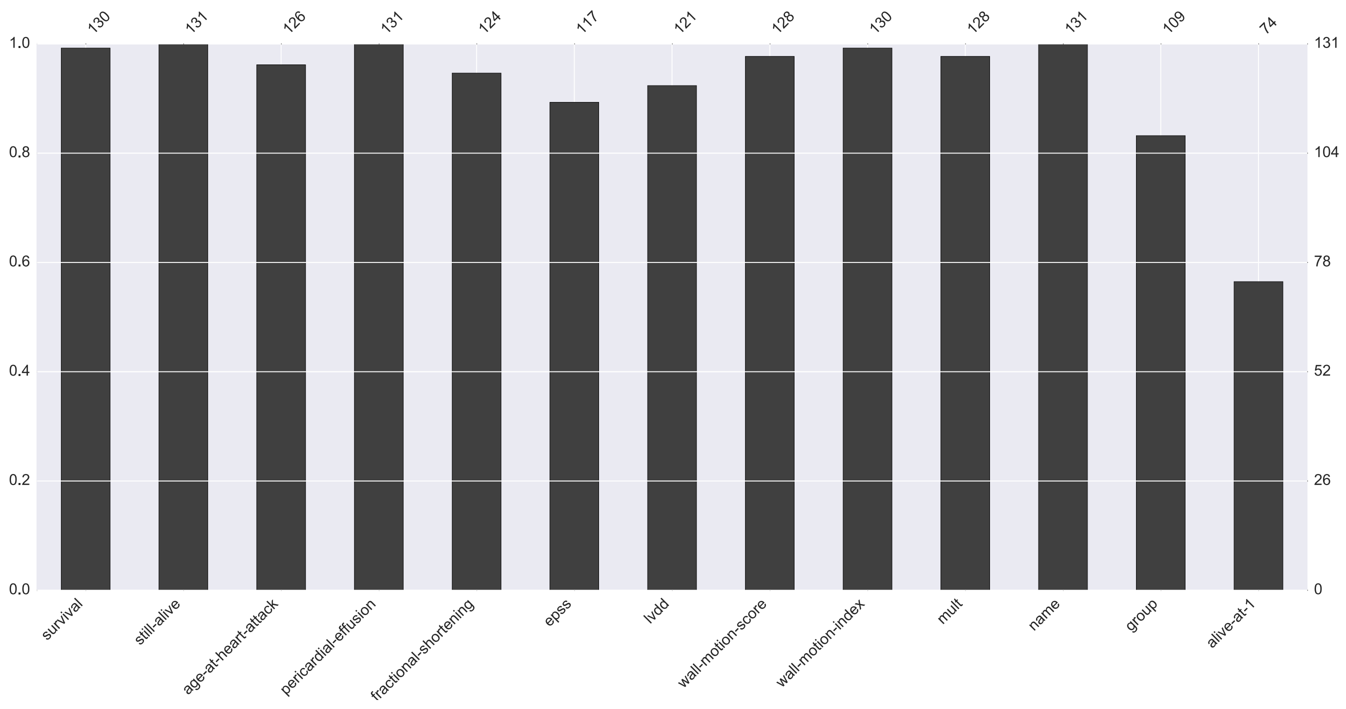


Figure 14. Fraction of nulls on each column for the *echocardiogram* dataset.

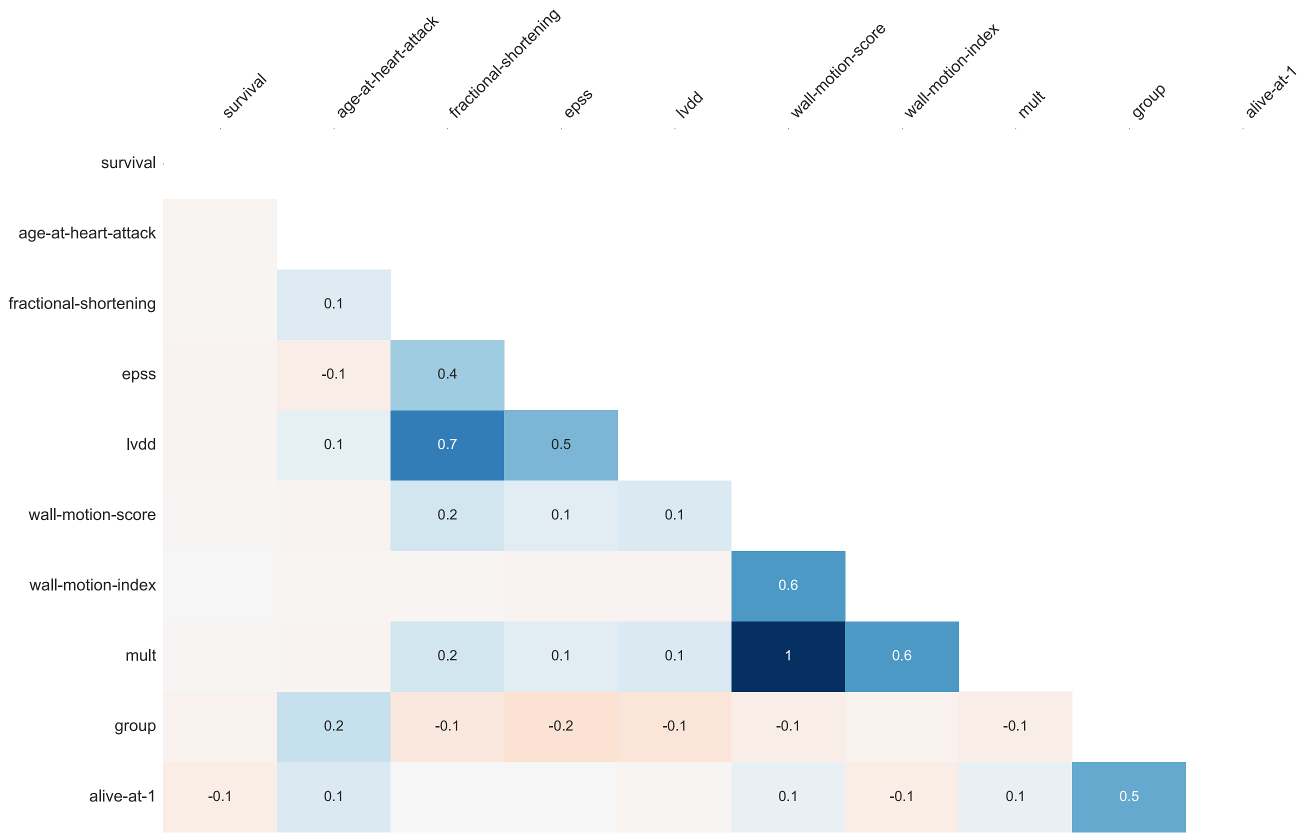


Figure 15. Nullity correlation between columns in the *echocardiogram* dataset.

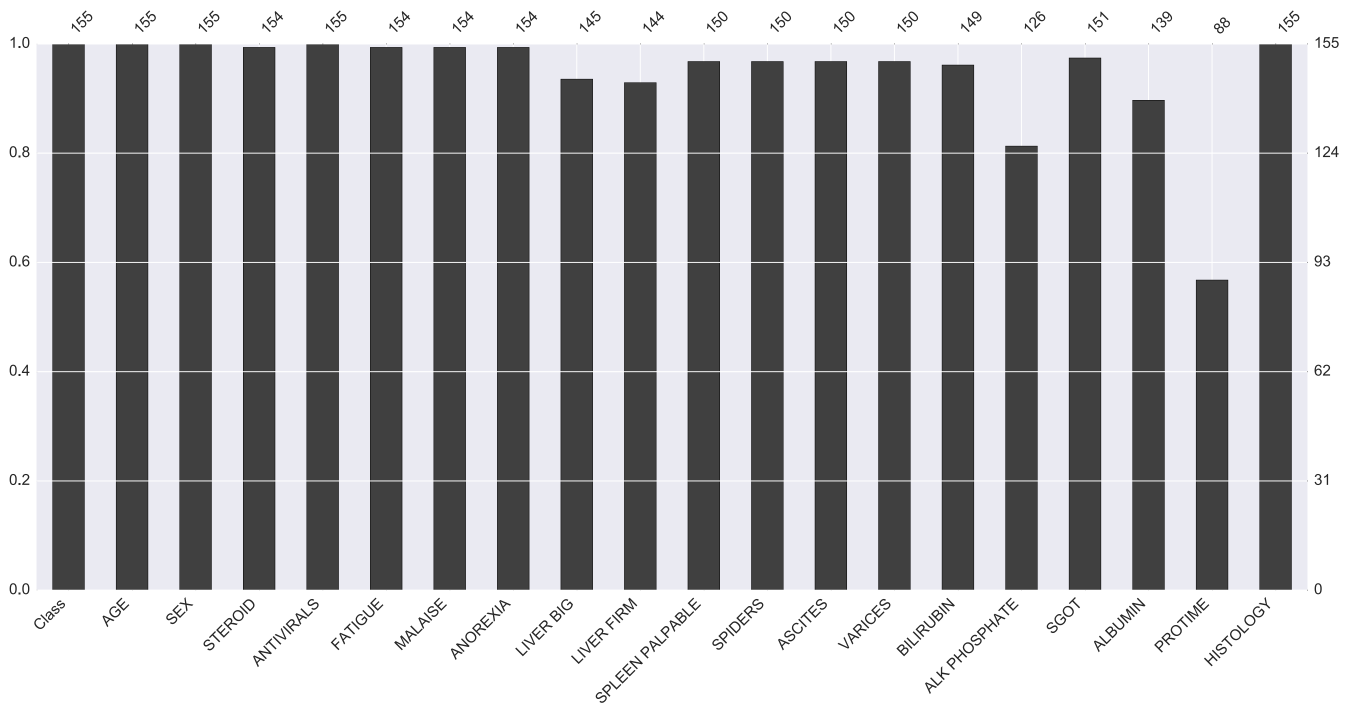
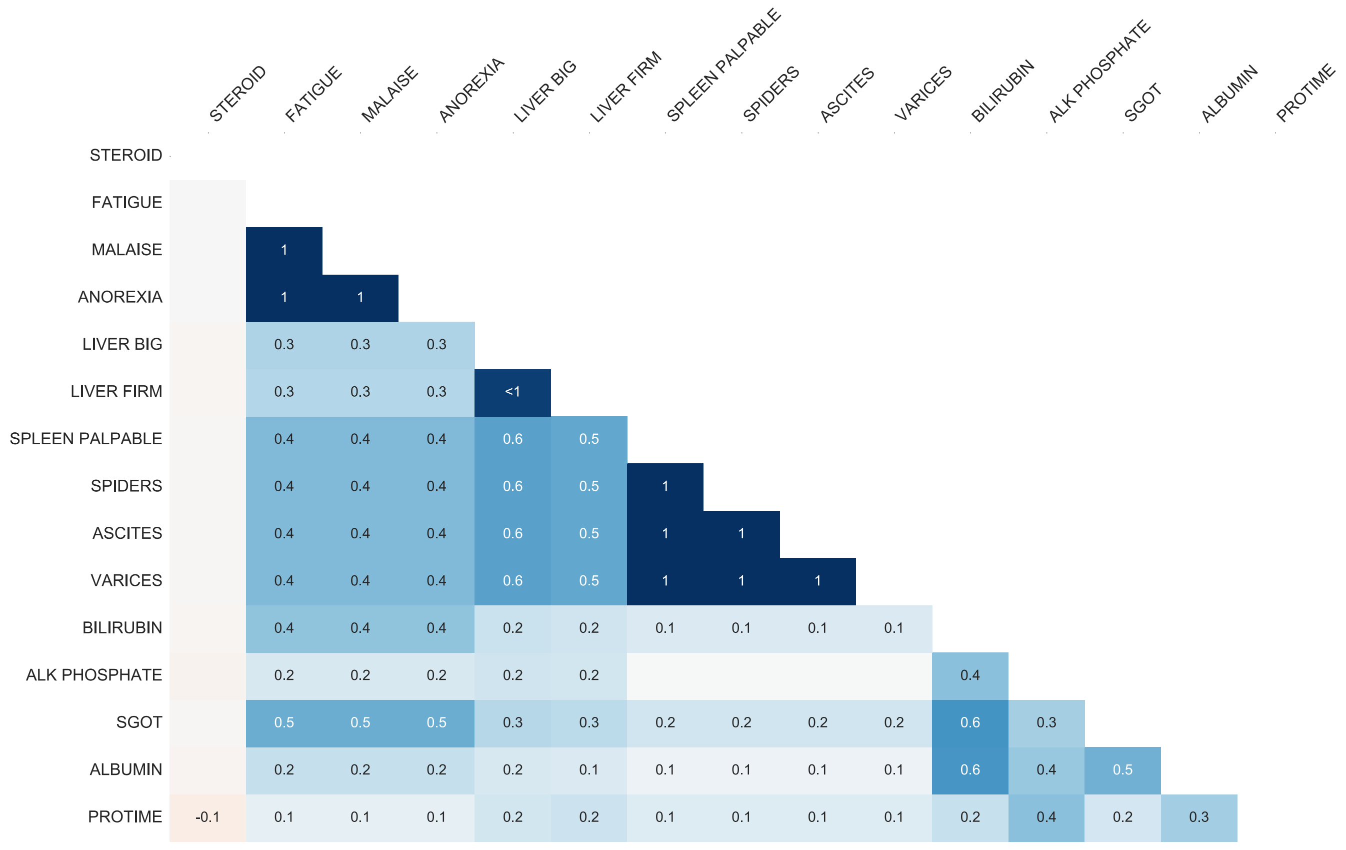


Figure 16. Fraction of nulls on each column for the *hepatitis* dataset.



	STEROID	FATIGUE	MALAISE	ANOREXIA	LIVER BIG	LIVER FIRM	SPLEEN PALPABLE	SPIDERS	ASCITES	VARICES	BILIRUBIN	ALK PHOSPHATE	SGOT	ALBUMIN	PROTIME
STEROID															
FATIGUE															
MALAISE		1													
ANOREXIA		1	1												
LIVER BIG		0.3	0.3	0.3											
LIVER FIRM		0.3	0.3	0.3	<1										
SPLEEN PALPABLE		0.4	0.4	0.4	0.6	0.5									
SPIDERS		0.4	0.4	0.4	0.6	0.5	1								
ASCITES		0.4	0.4	0.4	0.6	0.5	1	1							
VARICES		0.4	0.4	0.4	0.6	0.5	1	1	1						
BILIRUBIN		0.4	0.4	0.4	0.2	0.2	0.1	0.1	0.1	0.1					
ALK PHOSPHATE		0.2	0.2	0.2	0.2	0.2					0.4				
SGOT		0.5	0.5	0.5	0.3	0.3	0.2	0.2	0.2	0.2	0.6	0.3			
ALBUMIN		0.2	0.2	0.2	0.2	0.1	0.1	0.1	0.1	0.1	0.6	0.4	0.5		
PROTIME	-0.1	0.1	0.1	0.1	0.2	0.2	0.1	0.1	0.1	0.1	0.2	0.4	0.2	0.3	

Figure 17. Nullity correlation between columns in the *hepatitis* dataset.