The missing data conundrum-Investigation of Data Imputation techniques for sparse feature matrices of varying dimensionality

Priyanka Chakraborti
University of Nebraska-Lincoln
Lincoln, USA
bryan.moore@huskers.unl.edu

Bryan Moore
University of Nebraska-Lincoln
Lincoln, USA
priyanka.chakraborti@huskers.unl.edu

***Abstract*—We investigate multiple techniques for dealing with the extremely common problem of missing massive amounts of data. To this end we utilize two data sets from entirely disparate domains and numbers of entries, exploring the approach of imputing missing data one feature at a time and multiple features at a time. Our investigation shows clearly that imputing missing data can have a significant affect on predictive ability of a machine learning algorithm. Furthermore, some well known ML algorithms can produce worthy placeholder values for target prediction for one dataset, but fail entirely in another. In such cases their performance (evaluated by the mean squared error on the true target variable) is surprisingly sometimes comparable to, or even surpassed, by simple mean imputation or by bare bone removal of the row corresponding to the missing entries.**

***Index Terms*—Missing Data, Imputation**

I. Introduction

Missing data is a very common problem when dealing with real-life datasets. This becomes most acute in cases where the number of missing values in the dataset is significantly large (greater than 10%). This is further complicated when these missing data points belong to features which are strongly correlated with the target. Hence, the layman technique of automatically removing all rows corresponding to any null values can result in the loss of a huge amount of information, making it essential to investigate under what circumstances implementing data imputation techniques is a better approach for preprocessing a dataset, and which imputation methods are most effective.

To successfully implement data imputation one must comprehensively establish as closely as possible the reasons why the data is missing. There are three major classes based on the degree of missingness among the feature columns [1], namely:
(a)MCAR(Missing Completely at Random): This refers to cases where the missingness of the data is absolutely independent of any variables, whether observed in the study or not. As an example, consider a diabetes related obesity survey where the entries were missing because the patients forgot to attend the clinic on the day of the survey. Naturally, throwing away missing data in such cases would be unlikely to introduce any bias.
(b)MAR(Missing at Random): This is the more common case, where the probability of the missing data is determined only by the fully observed features, and not by other missing features as well. Following the above example, consider a case where a subgroup of diabetics could not attend the study because they worked for the same company and were on an official trip. This necessitates the recording of additional missing data but has nothing to do with the observed variables in the study itself.
(c)MNAR (Missing Not at Random): Here both the observed and the missing data will determine the probability of the missingness. As an example, consider a group of diabetics who drop out of the study because one of the survey questions requires them to go through a medical test that is 'uncomfortable'. Here the missingness is implicitly determined by the patient's level of comfort to a question, which in turn would affect the response to a question pertaining to the question. MNAR is the hardest to diagnose since it requires extensive domain knowledge beyond what is just observable.

Often, mean imputation or simple regression techniques are effective in replacing MAR values. However, there is always some uncertainty as to whether the missingness arises from unobserved variables or the missing data themselves. As such, it is judicious to explore clustering mechanisms as unsupervised learners of the data. To this end we chose to implement KNN and Gaussian Mixture Model (GMM) Clustering. It is also possible that the optimal strategy for imputing missing will likely rely on features that may not be very easily separated. We have therefore implemented many imputation approaches which rely on learning non-linear relationships within the data. These are KNN, Random Forest, Gaussian-Mixture Model clustering, and ANN.

That said, it is also possible that the data may be separable without resorting to finding nonlinear trends, which bears the danger of leading to overfitting. It is for this reason that we employ mean imputation and polynomial regression. If the most important relationships between the features are linear or polynomial, in practice we should find that polynomial regression fairs as well, or better, than the nonlinear algorithms.

II. DATA SUMMARY

To assess the efficacy of each imputation model we have obtained two disparate datasets. One dataset is from Zillow and contains 73728 instances, and the other is from Facebook and contains 500 instances. These were purposefully chosen as we wish to study the effectiveness of imputation techniques in varying dataset sizes. For the Zillow dataset one of the features we wish to include in our feature space is missing 65% of its data, which means we do still have 25260 instances which are intact. This may allow some of the imputation strategies discussed below to be quite effective.

For the Facebook dataset we randomly chose 250 instances from the feature most highly correlated with the target and set those instances to NaN. This only leaves us with 250 fully intact instances, which may cause some imputation strategies to be less effective. It is also possible that simply removing rows may be more effective than imputation for creating a predictive model. This will be investigated as well.

For each of these datasets we have defined **'true-target'** and a **'imputer-target'**. These two terms will be used throughout the rest of the article and careful distinction must be made here between them. The true-target is akin to the target variable in a standard machine learning dataset. It is the feature on which we would like to make a prediction. The imputer-target is the feature column that is missing data which we will impute such that it can be effectively used in our predictive model. As such, this feature becomes the target vector for our imputation algorithms. The bulk of our study rests on Single-Column Imputation, where only one column is missing data.

We have also briefly begun exploring Multi-Column Imputation, where multiple columns are missing significant amount of data. To avoid confusion, we will be very clear when discussing Multi-Column Imputation such that there is no confusion with the Single-Column Imputation cases. Thus, unless it is specified, please assume when we refer to an Imputation Target that we are referring to the case where only one feature needs to be imputed.

The Zillow dataset we used in this study can be found on this page *https://www.kaggle.com/c/zillow-prize-1/data*. For this dataset the imputer-target is **garagecounts** and its corresponding true-target is **taxvaluedollarcnt**. 'garagecounts' has a correlation of 0.35 with 'taxvaluedollarcnt'. For simplicity we have considered imputing only one feature vector containing missing data. The rest of features including in the main feature set are not missing any data (or were missing so few we felt it was safe to replace by simple mean imputation). A heatmap of the nullity relationships between the main target features is shown in Fig. 1. From this heatmap we believe that the MAR assumption to be the most likely explanation for the cause of the missing data.

This dataset has a total of 73728 instances, of which 65% entries appear to be MAR from **garagecounts**. The set contains 60 features, of which we have included 5 in our main model and 7 in our imputation model. Our main model contains the target **taxamount** and the main model features are 'landtaxvaluedollarcnt', 'structuretaxvaluedollarcnt', 'finishedsquarefeet12', 'fullbathcnt', and 'garagetotalsqft'. Our imputation target is **garagetotalsqft** and the imputation model features are 'taxamount', 'propertylandusetypeid', 'bedroomcnt', 'calculatedfinishedsquarefeet', 'bathroomcnt', 'structuretaxvaluedollarcnt', and 'roomcnt'. There is no overlap between the feature spaces of true versus imputer target.

The Facebook dataset we used in this study can be found on this page *https://archive.ics.uci.edu/ml/datasets/Facebook+metrics*. This dataset contains 500 instances, Of which we have randomly replaced 50% of the entries with 'NaN' values to simulate real world missingness for the feature most-correlated with the target **Post Month**. For this dataset we performed one-hot encoding on the 'Type' category. The true-target for this dataset is **Page total likes**. This dataset has 19 features, of which we have included 5 in the feature space for our main model, and 5 in the feature space for our imputation model.

Our main model target is **Page total likes** and the main model features are 'Post Month', 'type_Status', 'Lifetime Post Consumers', 'Post Hour', and 'Lifetime Post Consumptions'. The imputation-target for the single column imputation is **Post Month**, and the features for the imputation model are 'type_Status', 'type_Photo', 'Lifetime Post Consumptions', 'Category', and 'Lifetime Post Consumers'. 'Post Month' has a correlation of 0.94 with the true-target, 'Page total likes'. Here there is small overlap between the two feature spaces post imputation. Seaborn pairwise plots [2] were examined to ensure that the imputer feature is not highly covarying w.r.t the common subset of features that appear in both spaces.

III. METHODS

We have performed experiments using many imputation strategies for replacing missing data in single features or multiple features. The strategies implemented for Single-Feature Imputation are Mean Imputation, Gaussian-Mixture Model Clustering, K-Nearest Neighbors Imputation, Polynomial Regression, Random Forest Regression, and Artificial Neural Network Regression. For assessing the effectiveness of these techniques we calculated the MSE (Mean Squared Error) for the replacement of values in the test set and also the MSE for final predictions made on the main target after imputation. We chose to use MSE as our error

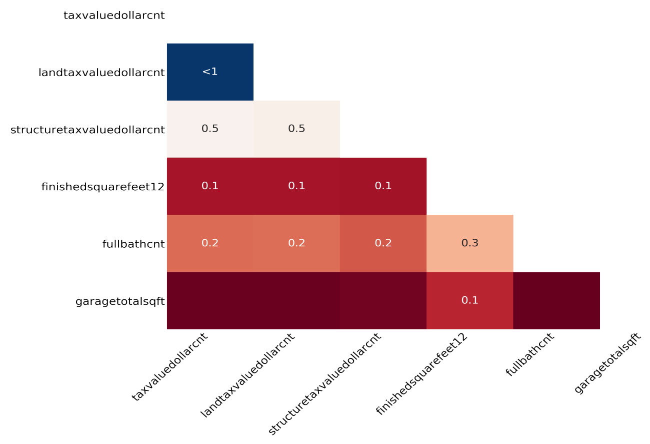

Fig. 1. This shows a simple correlation heatmap for the Zillow dataset. This map describes the degree of nullity relationship between the different features before any imputation was employed. The range of this nullity correlation is -1 R 1. Features with no missing value are excluded in the heatmap. If the nullity correlation is very close to zero (-0.05 ¡ R ¡ 0.05), it will be shown as blank. A perfect positive nullity correlation (R=1) indicates when the first feature and the second feature are always missing values at the same time. A perfect negative nullity correlation (R=-1) means that one of the features is missing and the second is never missing.

metric due in large part to the somewhat normal distribution of many features in our data and also our assumption that the data is MAR (missing at random). We also report the r^2 values for our imputations and predictions, but this is largely done so that the effectiveness of techniques can more easily be compared on an absolute scale across datasets. A flow chart of our Single-Column Imputation Investigation is provided in Fig. 2.

We also begin, to first order, an investigation into Multi-Column Imputation strategies with MICE (Multiple Imputation by Chained Equations) and Mean Imputation. This investigation is only performed on the Facebook dataset, which has 500 total instances. We randomly selected half of the instances for the three features most highly correlated with the main target and removed these values. As these were selected at random for each column separately we are assured of our MAR assumption. The MICE algorithm we implemented is described in detail below.

For this approach we did not perform hyper-parameter optimization on MICE. This is because there were so few rows remaining not missing any data in the available train set to draw any meaningful conclusions from cross-fold validation. We thus used only one pass with MICE as a proof of concept. MICE code was home-built using only the 'Imputer' Python class as a wrapper. A flow chart of our Multi-Column Imputation Investigation is provided in Fig. 3.

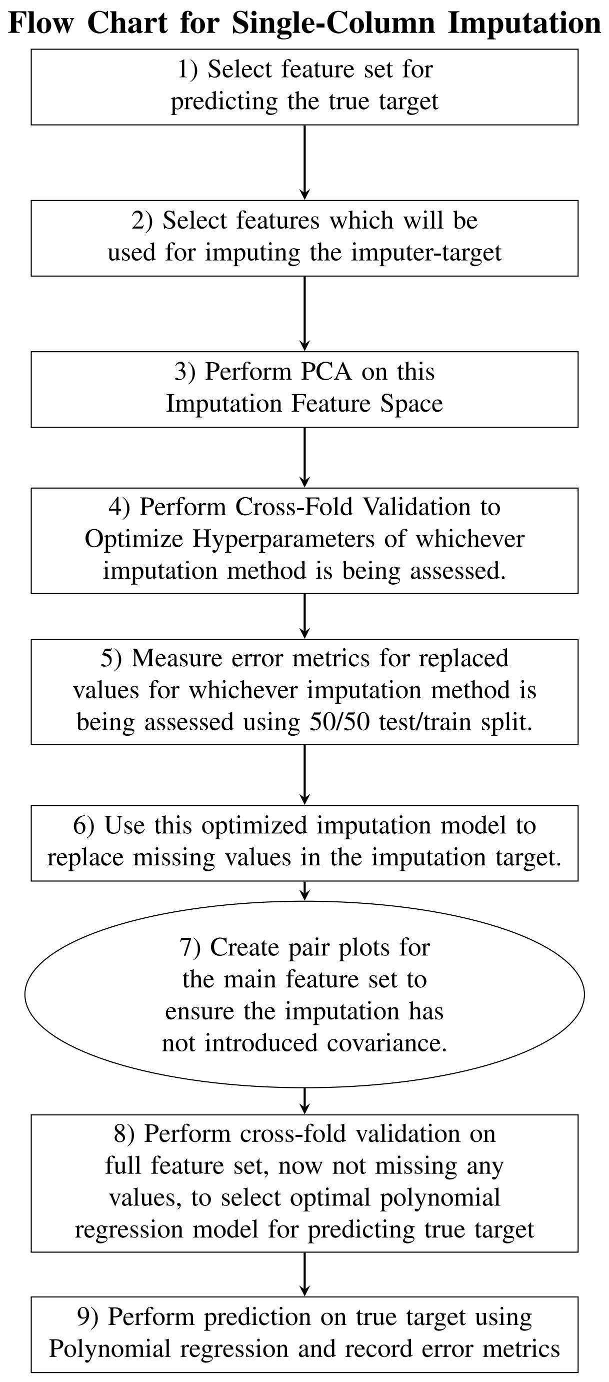
Flow Chart for Single-Column Imputation
1) Select feature set for predicting the true target
2) Select features which will be used for imputing the imputer-target
3) Perform PCA on this Imputation Feature Space
4) Perform Cross-Fold Validation to Optimize Hyperparameters of whichever imputation method is being assessed.
5) Measure error metrics for replaced values for whichever imputation method is being assessed using 50/50 test/train split.
6) Use this optimized imputation model to replace missing values in the imputation target.
7) Create pair plots for the main feature set to ensure the imputation has not introduced covariance.
8) Perform cross-fold validation on full feature set, now not missing any values, to select optimal polynomial regression model for predicting true target
9) Perform prediction on true target using Polynomial regression and record error metrics

Fig. 2. The feature space used for imputing the missing values never includes the main target as a feature. To replicate massive amounts of data missing when training, we split all rows not missing any values equally into test and train portions. For hyperparameter optimization we do retain the standard 80/20 split to provide a full 5-fold averaging of the error metric (MSE). Since we have no labels for validating our prediction on data which is actually missing, we further evaluate the imputation performance by reincorporating this imputed feature into the main feature set and using this to make a prediction on our 'true-target'.

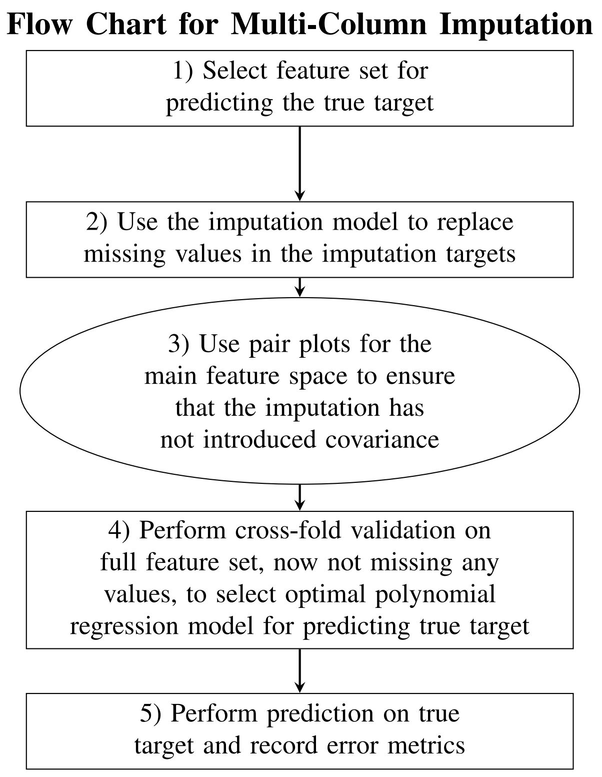
Flow Chart for Multi-Column Imputation
1) Select feature set for predicting the true target
2) Use the imputation model to replace missing values in the imputation targets
3) Use pair plots for the main feature space to ensure that the imputation has not introduced covariance
4) Perform cross-fold validation on full feature set, now not missing any values, to select optimal polynomial regression model for predicting true target
5) Perform prediction on true target and record error metrics

Fig. 3. The approach taken here is very similar to that taken for the investigation into Single-Column Imputation. However, we did not measure the effectiveness of the imputation itself, but were assessing entirely the improvement in predictive power.

A. *Principal Component Analysis (PCA)*

We will skip the detailed theoretical discussion of PCA. Readers are encouraged to refer to this article by Sebastian Raschka [3] for a comprehensive understanding. One of the main advantages of PCA is that it allows us to distillate the most important eigen-modes of our feature matrix. We can choose how many to keep such that we can explain a chosen percentage of the variance of our data without keeping unnecessary noise. It also mathematically guarantees that our 'principal components' are independent of each other in the new space.

We chose not to implement PCA Whitening as we already removed redundant features, and therefore do not believe that significant correlation between features should be a major concern.For both of our datasets we chose to keep just enough components to explain 95% of the variance. In effect, PCA is an unsupervised method to capture the intrinsic variability of the data while also removing covariance.

B. *Mean Imputation*

Mean Imputation is the simplest possible implementation strategy. We replace any missing value in a feature with the mean value of all present values for that feature. This approach has very low time complexity(O(d)) and can often provide reasonable results, although it can severely underfit the data. We will treat Mean Imputation as our baseline imputation strategy and report all other error metrics with respect to the results obtained by Mean Imputation.

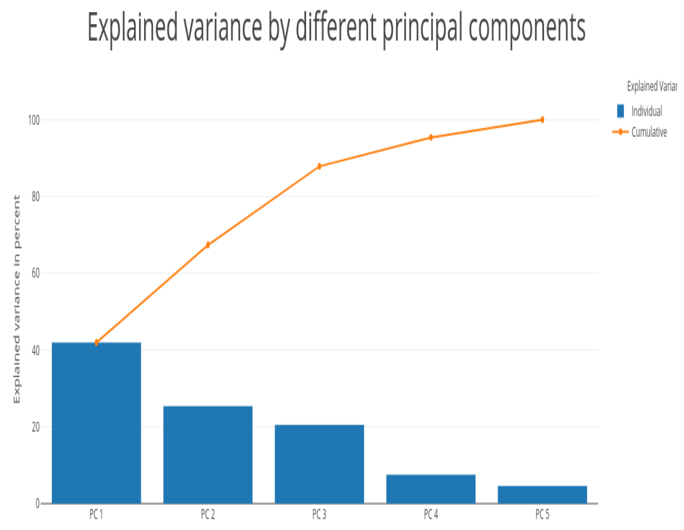

Fig. 4. This shows the variance of principle components in the Zillow dataset after PCA. The X-axis holds the features in the new space and the Y-axis holds the total variance in our data explained by each feature. We chose to keep only the first 4 features as these collectively explain 95% of the variance in our data. We were therefore able to remove one dimension from our feature space.

C. *Unsupervised Non Parametric Strategies : KNN and Gaussian Mixture Model Clustering*

There is always the possibility that our imputer-target may not actually be missing at random with respect to the other observed features. To verify this requires specialized domain knowledge in locating and including additional features, or we can instead let the data speak for itself by employing a class of non parametric unsupervised learning algorithms which make no a priori hypothesis about the data. If these methods prove to be very effective this may also tell us something about the underlying distribution of our data, and whether MAR is a reasonable assumption.

KNN can be thought of as an Ocam's razor solution to this problem. It is automatically non-linear, and given that we have a fairly large number of examples (especially for our Zillow target dataset) and a small number of principle components, KNN will not suffer from the 'Curse of Dimensionality'. It has a time complexity of O(nd), which has been found to be tractable on both datasets. This algorithm works by finding the instances in the training set 'closest' to itself. It then returns the weighted average, or unweighted mean, value for those rows. The hyperparameters we optimized over are the number of neighbors to include, the order of the Minkowski distance metric, and whether we should weight the contribution of each row to the returned value based on how 'close' each vector is to our imputation target instance.

We do note that for the Facebook dataset we expect a degradation of performance because post artificial NaN replacement it has only 250 instances remaining not missing any information. A graphical node-based nearest neighbour search, such as KD-Tree, may be a better choice to circumvent this limitation, but as we are interested in how KNN will perform on a small dataset like this we will implement KNN and study the performance. Also, note that the 'Mahalanobis' distance metric is not a part of our cross validation grid search distance metric parameter. This was chosen intentionally as implementing PCA already isolates high variance components and transforms us to a space where features are less correlated, similar to the Mahalanobis distance approach.

A Gaussian mixture model is typically used for binning data into meaningful clusters of similar data. Each cluster is modelled by a different Gaussian distribution. This approach assigns a Bayesian soft assignment probability to each cluster. An easy way to visualize this is to imagine that each hidden Gaussian distribution has some responsibility towards generating a particular data instance D. Since our feature matrix is d dimensional the random variables for a multivariate Gaussian distribution are mean and variance. In the spirit of MAR we can introduce a latent variable Z, which we can think of as a feature variable for which data was never collected, but which directly influences our missingness [1]. The joint distribution of our feature set **X** with that of **Z** is given in equation 1 [4].

$$ln\{P(X \mid \Theta)\} = ln\{\sum_{Z} P(X, Z \mid \Theta)\} \tag{1}$$

where Θ represents our parameters (μ, σ).

We maximize this joint probability in the Gaussian-Mixture Model Clustering algorithm and update our mean and variance for each Gaussian at each step of the maximization. After the iterations are complete we thus have a soft classification of probabilities for each instance to be in a particular Gaussian cluster. The subtlety here is that GM is an unsupervised learning model, while in this case our imputer target class is actually labelled. We address this by making a slight change to this process such that we transform this approach into a 'Semi-supervised' learning problem.

We use the classification output from the clustering to calculate the mean value of the target column for each cluster found in our training set. When predicting target values for our imputer class we assign this mean value to each instance of the train/test set belonging to a particular cluster. We consider this to be an improvement over mean imputation, and similarly we do still expect many predictions to be fraught with bias due to underfitting. Although this is a semi-classification algorithm we still report 'MSE' as our main error metric as our final prediction is on a continuous variable.

The time complexity for this approach is O(d^3*I*C+N), where I is the number of iterations, C is the number of clusters, d is the number of features, and N is the number of instances. This has been shown to be tractable on both of our datasets. It must be noted here that GMM does not lead to an optimal closed form solution for the likelihood function. At best we are estimating a local maximum for the missing data. Also, due to the data distribution on the Facebook dataset (which includes multiple multinomial distributions) we do not expect GMM Clustering to be very effective.

There is a natural question which arises to this GMM Clustering approach. Due to the significant amount of data missing would an MLE maximization approach perform well and still give good convergence? Dellaleau et al, have shown that we can still achieve good accuracy given that we evaluate the full covariance matrices for each cluster along with applying regularization to them [5]. In our vanilla implementation of GMM we were able to implement the former, but not the latter. We thus note that there is significant room for improvement for this method.

D. Parametric Supervised Strategies: Polynomial Regression and Artificial Neural Network

Regression models are some of the simplest, and most easily scalable, algorithms in Machine Learning. Polynomial Regression can be tailored to work with both small and large datasets without significant overfitting by using a simple regularization parameter. We have chosen to use polynomial regression with Stochastic Gradient Descent. While the asymptotic properties of SGD ensure that we are more likely to arrive at a global minimum, even for a non-convex optimization problem, it also generally has a faster convergence speed.

Polynomial regression works by projecting the feature space to a higher dimension but still carving a linear decision boundary. To find the optimal order complexity for our feature space we first utilize OLS linear regression to find the polynomial order which gives the lowest MSE on the test set, and the best generalization error between test and train.

We then use this resultant order to expand our feature space and subsequently scale each feature. After this we perform hyperparameter optimization over the regularization parameter, the learning rate, the ratio of L1 regularization to L2 regularization, the maximum number of iterations, and whether the learning rate is constant or updated during the descent. The time complexity of Stochastic Gradient Descent followed by applying the weights for predictions is O(d*I+d), which is tractable for both of our datasets.

To incorporate the concept of a non-linear decision boundary within a parametric regression class we can use an artificial neural network. ANN is a parametric model as we set the architecture, specifying the number of layers, number of nodes per layer, etc... For our ANN model we incorporated several intermediate outputs at each node of a unidirectional graph. These nodes are referred to as neurons. By combining several layers of such neurons we construct a neural network, where each node from a single layer broadcasts its result to all nodes at the following layer. If we have 0 hidden layers this should only be capable of representing linearly separable functions. 1 hidden layer can approximate any function that contains a continuous mapping from one finite space to another. 2 layers can represent any arbitrary decision boundary to reasonable accuracy with rational activation functions. [6]

For a visual representation of how this work see Fig. 5. The weights associated with each calculation are assigned by tracking the error in the calculation backward through the layers and figuring out which node calculation contributed more or less to that error. The weights for each neuron are thus adjusted. first compute the derivatives of the error with respect to the weights of the second to last layer. In other words, we backpropagate the derivative computations in the network to adjust these weights.

For the forward part of the calculation, note that each of these neurons outputs a result only after it passes through the activation function. We used the RELU (Rectified Linear Units) activation function as this will not pass on any results from a particular node if the value of this function is less than or equal to zero, but will pass on the exact calculated function (along with the weight) otherwise. This is less computationally expensive than other activation functions, such as a sigmoid function, and still provides very good results within an artificial neural network (ANN). As we will see, computational complexity is a major drawback of utilizing an ANN for this type of imputation.

Because of this network approach overfitting can easily occur if we do not restrict ourselves to a shallow enough network. After creating learning curves over number of hidden layers we chose to utilize 2 hidden layers. After this the hyperparameters we optimized over are the maximum number of iterations, the batch size, the number of neurons in layer 1, and the number of neurons in layer 2. The time complexity for this algorithm is O(N*I*[Ne1*Ne2+Ne2*Out]), where N is the number of training instances, I is the number of iterations, Ne1 is the number of neurons in hidden layer 1, Ne2 is the number of neurons in hidden layer 2, and Out is the number of neurons in the output layer. This was found to be tractable for both datasets, but is significantly slower than all other tested imputation approaches.

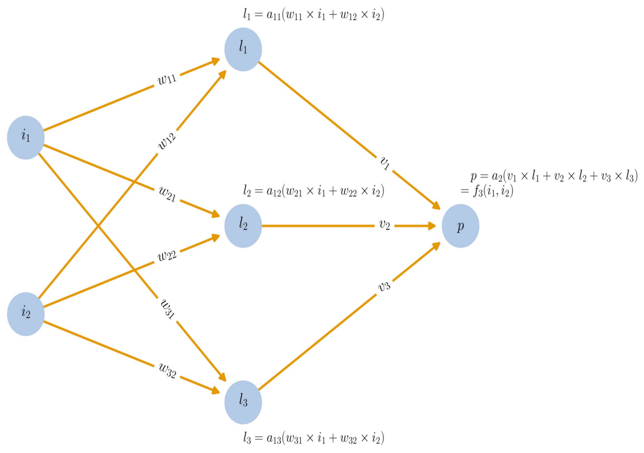
$l_1 = a_{11}(w_{11} \times i_1 + w_{12} \times i_2)$
l_1
w_{11}
w_{12}
i_1
v_1
$p = a_2(v_1 \times l_1 + v_2 \times l_2 + v_3 \times l_3)$
$= f_3(i_1, i_2)$
w_{21}
$l_2 = a_{12}(w_{21} \times i_1 + w_{22} \times i_2)$
l_2
v_2
p
w_{22}
v_3
i_2
w_{31}
w_{32}
l_3
$l_3 = a_{13}(w_{31} \times i_1 + w_{32} \times i_2)$

Fig. 5. This figure is taken from a very useful article by Rocca [7]. Here we have a neural network with 2 inputs, 1 hidden layer with 3 neurons, and 1 output. $w_{i,j}$ are the weights for each neuron, l_i are the neurons at each layer, and p is the final output. Incorporating multiple hidden layers this architecture should allow us to capture any nonlinearity in our data. Neural networks do employ SGD to arrive at the optimal weights, but through a layer by layer approach known as back-propagation.

E. Non-Parametric Supervised Strategy / Ensemble Learning: Random Forest

As a final approach we explore the possibility of training ML algorithms on different subsets of our data, sampled with replacement. The intuition is that if random subsets of our data are trained by separate learners we will form an ensemble of learners which each make individual predictions on random portions of the dataset. The average of all predictions may thus be more accurate than any one prediction made by a single learner, and will certainly be more robust to noise.

The approach we implemented is called Random Forest, where each 'tree' in this 'forest' is actually a Decision Tree. Each Decision Tree receives a randomly chosen number of features and a randomly chosen subset of instances with those features. At each node of this tree the dataset passed to that node is split, using a threshold, such that 2 pieces are formed which are as homogeneous as possible. The metric used for this is to minimize the average squared error between group average and individual values of elements in the group. The splitting process continues until maximal separation has been achieved, or the maximum number of splits is struck.

These final groups are referred to as 'leaf' nodes, and hold the decided upon target values for those final grouped instances. To prevent overfitting it is also possible to prune the tree by removing 'branches' which provide little predictive power. This can help with overfitting, but may introduce bias. We did not incorporate this in our study, but it is worth exploring in the future. Our model thus has an ensemble of decision trees which are trained on different subsets of the data and random selections of features.

For making predictions on a single instance we use the same thresholds decided upon for each tree during the learning process. We then find the predicted target values for a particular instance in every tree which received that instance. The value returned to the user is the mean value of the prediction made across all trees. For hyperparameter optimization we explored whether it was worth employing bootstrapping to sample the data, the maximum depth allowed, the maximum features to send into any one tree, the minimum samples allowed for a cluster to be considered a leaf, the minimum samples allowed for a split to occur at a node, and the number of trees to include in our forest. The time complexity of Random Forest is O(N^2*d*T+d*T), where N are the number of training instances, d the number of features, and T the number of trees in the forest. For both of our datasets this was tractable.

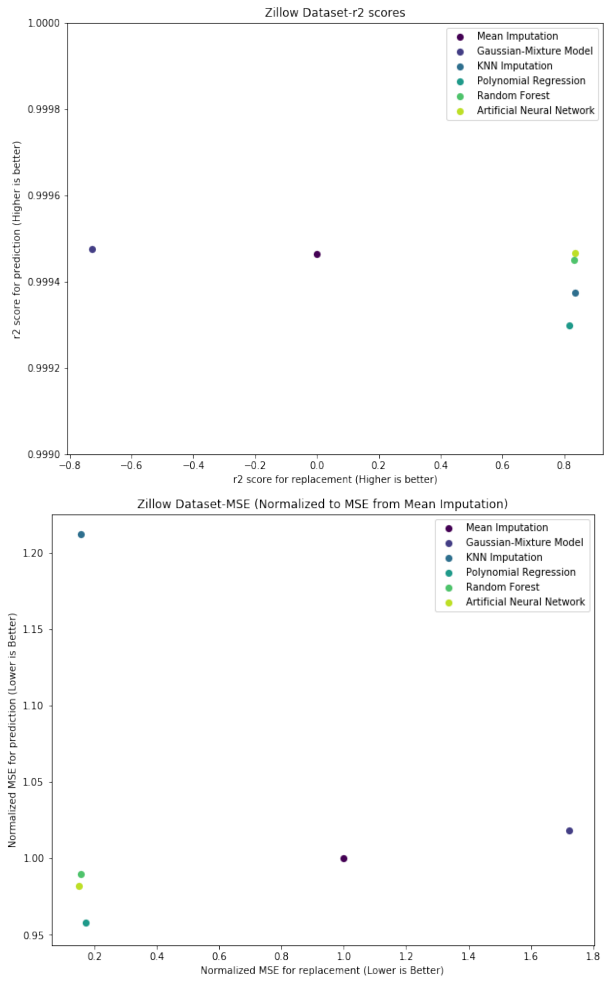

Fig. 6. This shows the overall results achieved by all imputation methods for the Zillow Dataset. Note that the x-axis always corresponds to the imputation effectiveness while the y-axis always corresponds to the prediction effectiveness. We see that although there were small differences between the approaches, all did very well, as is easily seen in the above plot (showing the r^2 scores).

F. Multi-Column Imputation Strategy MICE

As described by Buuren and Groothuis-Oudshoorn [8] MICE (Multiple Imputation by Chained Equations), works by imputing the missing values with the mean value of its corresponding column. It then centers itself on a single column which has missing values. It uses a train set composed of instances for which we have full data for that column and trains a model (we used linear regression) to find the optimal weights for predicting values in the column we are centered on. It then replaces all missing values for that column with the predicted values. It then moves to the next column missing values, centers itself there, learns the weights, and replaces those with predictions.

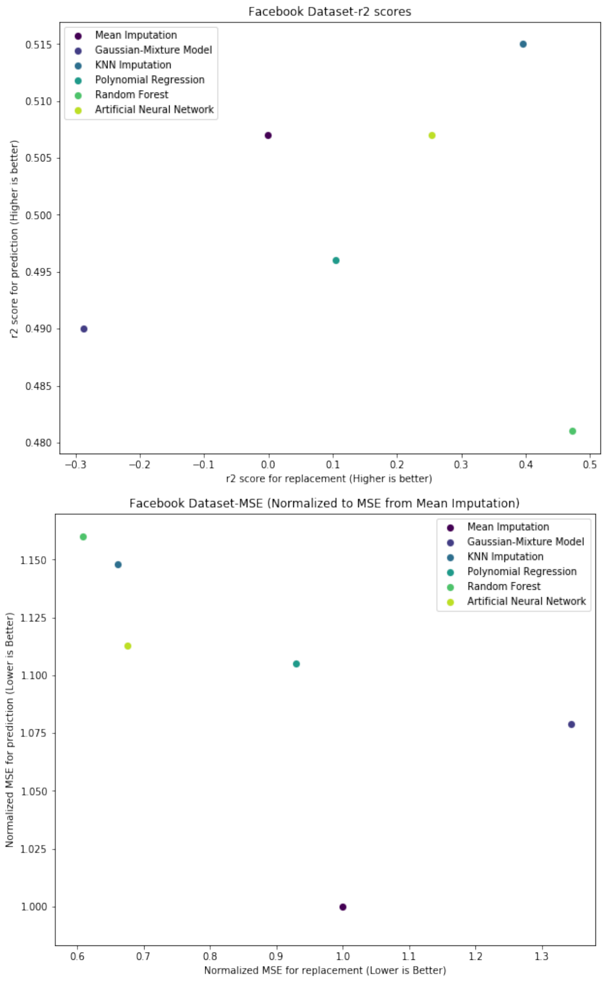

Fig. 7. This shows the overall results achieved by all imputation methods for the Facebook Datasets. Note that the x-axis always corresponds to the imputation effectiveness while the y-axis always corresponds to the prediction effectiveness. We can see what look like possible trends of increased predictive performance versus increased replacement performance in the plot showing r^2 scores, although as none had good performance in either category it is hard to argue that this possible trend is not merely due to noise.

This process generally continues for multiple iterations, although for our proof of concept implementation we only utilized a single iteration. The time complexity of this algorithm is similar to OLS linear regression, as we did not implement gradient descent, except that it must be done for all features missing values and iterated over multiple times. We thus have a time complexity of O((($d-1)^2$*N+($d-1)^3$+(d-1))*M*I), where I is the number of iterations, d is the number of features, N is the number of features in the training set for each calculation, and M is the number of columns missing data. On the Facebook dataset, where we ran this, this was absolutely tractable. As mentioned above, we did not perform hyperparameter optimization over this method due to the low number of instances not missing any data.

IV. Results

The results for Single Column Imputation are summarized in Table 1 and Table 2 (in Supplemental Materials section), and shown in Fig. 6 and Fig. 7. Table 1 contains the results for the analysis of the Zillow dataset, which had a total of 73728 instances, of which 48468 are missing data in the feature column 'garagecounts'. This feature has a correlation of only 0.328 with the main target, but we surmised that this was significant enough that with suitable imputation we should be able to improve our predictive ability.

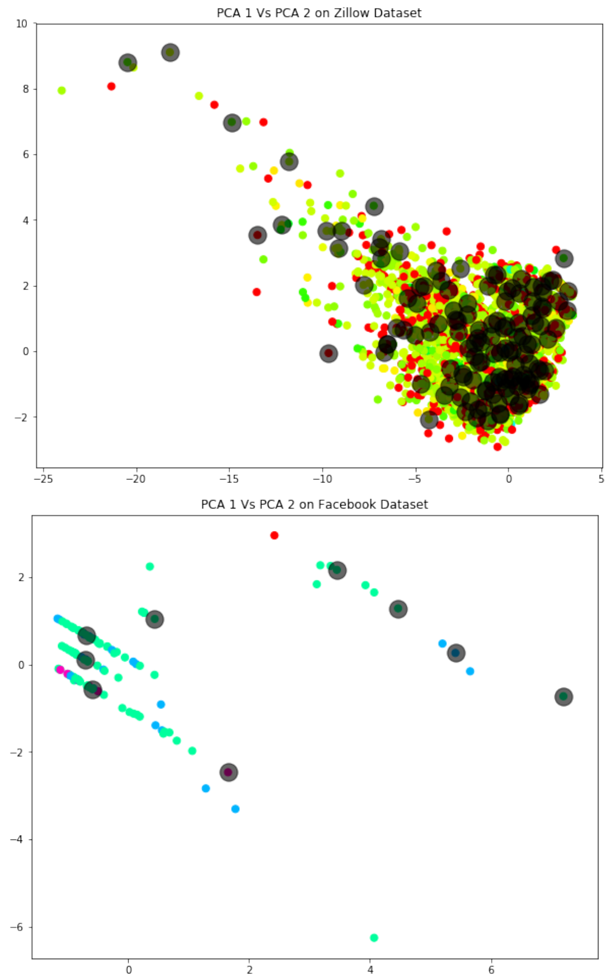

Fig. 8. In these plots each color represents a different class. For the Zillow dataset we actually have 131 classes, and we can visually see that the clusters are very overlapped. For the Facebook dataset we also see that it looks like the data cannot be separated, although here we have only 9 classes.

The results of our experiment did uphold this hypothesis. The prediction r^2 score for the rows without missing any values was about 0.97, and after imputation all approaches had raised this predictive r^2 score to 0.999. To better understand the performance of the imputation techniques we instead look at the adjusted MSE values. These have all been normalized with respect to the MSE obtained using Mean Imputation, as this is our baseline approach.

For these results we report the normalized MSE and r^2 score metrics on the imputation-target as well as the final prediction on the true-target. The results arising from the Zillow dataset are noteworthy and interesting. In terms of overall target prediction the most effective imputation methods are Polynomial Regression, ANN, and Random Forest. However, they are all only slightly more effective than bare bone mean imputation, which is in turn much more effective than removal of data entirely. The other two clustering algorithms, GMM and KNN, actually led to worse predictive results than mean imputation. This is especially strange as KNN was very effective at imputing missing data.

The most effective techniques for replacing data in the Zillow dataset were ANN, followed by Random Forest, followed by KNN. All of these are much more effective than simple Mean Imputation at replacing the missing values by more than a factor of 5. The only algorithm to have a less effective imputation score than Mean Imputation is GMM Clustering. In Fig. 8 we see that for the Zillow dataset (showing only the relationship between the top two most important features) it is not possible to truly separate the data into classes. The data is very overlapped, making this algorithm ineffective. This further reinforces the underlying MAR assumption.

Regularizing the covariance matrix in the imputing stage could perhaps help with this, although this has not been explored in this investigation. Alternatively we could perhaps augment the feature space to a higher order and see if that can help to uncoil some of the tight clustering. This said, excluding GMM Clustering, it appears that unsupervised learning models do just as well as supervised learning models, if given a large enough dataset. For those situations it appears that it is worth employing one of the above mentioned techniques for imputation, even at the risk of introducing some bias.

Another interesting finding is that for the Zillow dataset the ability to more accurately impute data was not directly proportional to the increase in predictive power gained after imputation. We are not sure why this is, but hypothesize that it may be due to the relatively low correlation which this feature has with the main target. We will note that for the Facebook dataset, which has 500 instances of which 250 are missing in the most highly-correlated column, that we also do not see a clear relationship between the replacement score and the prediction score. However, for that dataset none of the methods were able to effectively impute missing data, so this potentially small relationship could simply be due to noise.

We also investigate how important the number of rows not missing any data are to the effectiveness of these imputation models. The details of this investigation can be seen in Table II (found in the Supplemental Materials section). The smaller Facebook dataset, of which we artificially removed half from the feature 'Post Month', has a correlation of 0.941 with the main target. For such low number of available instances none of the models were able to generalize suitably. The results for rows not requiring imputation were able to achieve an r^2 score of 0.94, while the highest r^2 score from imputed data was only 0.52. We thus conclude that for a small enough dataset one might consider deleting missing instances, and increasing predictive power instead through incorporation of domain knowledge in Bayesian learning.

Although we did not explore multi-column imputation very deeply in this study, we did conduct a proof of concept experiment. On the Facebook dataset we removed half of all instances from each of the top 3-most correlated features with the main target. These results are recorded in Table III (found in the supplemental material section). Here we see that MICE actually performed slightly worse than Mean Imputation or No Imputation at predicting the main-target. However, it had significantly better standard deviation, although none of the methods are truly acceptable. Mean Imputation led to the highest predictive power, with a r^2 score of 0.41. No Imputation and MICE both had r^2 scores of 0.39. Thus, we were unable to capture the true power of MICE, but have shown that it is not able to increase predictive power on such a small dataset with such massive amounts of data removed.

V. Conclusion and Future work

Our overarching goal is to study the imputation of missing data in two disparate datasets using supervised and unsupervised learning algorithms. Assuming the missingness of our data can be attributed to the MAR assumption, our study indicates unsupervised learning should work quite well as they are able to draw flexible decision boundaries around the data. However on smaller datasets with large missing values both supervised and unsupervised ML algorithms found it difficult to generate reasonable results. This makes sense, as on smaller training set it is harder to obtain enough samples to adequately model the underlying distribution. There is also the additional trouble that for clustering algorithms the nuances of semi-supervised learning were not able to capture the nuances due to the close clustering of the data.

One possible way to address this would be to exercise greater granular control on the semi-supervised learning approaches at a more grass-root level. For instance, we could apply a learning rate to this clustering update process, which may provide improvements in class prediction. We could also implement weighting on the covariance matrices for each class of the Gaussian Mixture. Assigning these without specialized domain knowledge would require a thresholding mechanism to know what weight should apply to each cluster, thus making it somewhat similar to the thresholding at each node of a Decision Tree. It is also possible that augmenting the feature matrix before performing GMM Clustering could improve the classification ability. This should be checked in future studies. Within Random Forest we could see if we can improve performance by having the assigned prediction be weighted by the confidence each tree has in its result.

Figuring out the hidden nuances of semi-supervised learning imputation techniques will help us improve our MICE approach as well. Within the limited duration of the project this study has successfully established the potential of semi-supervised learning as a tool to impute data MAR, and potentially also diagnose the MAR assumption in missing datasets. This study has also found that there is not necessarily a direct relationship between imputation effectiveness and predictive effectiveness. This should certainly be investigated in future studies. It is also suggested that the effectiveness of imputation techniques which rely on the MAR assumption can potentially be used as a diagnostic to determine whether the data is MAR or potentially MNAR, as decided by the effectiveness of strategies which require this assumption and those which do not. To further generalize these results future explorations should be conducted on ensemble approaches with different amounts of missing data, and also with the features missing the data having different correlation strengths with the main-target.

References

[1] P. D. Allison, "Handling missing data by maximum likelihood estimation," *SAS Global Forum*, vol. 312-2012, 2012.
[2] PyData, "https://seaborn.pydata.org."
[3] S. Raschka, "A step by step tutorial to principal component analysis, a simple yet powerful transformation technique."
[4] C. M. Bishop, *Pattern Recognition and Machine learning*. Springer, 978-0387310732 ed., 2011.
[5] A. C. Olivier Delalleau and Y. Bengio, "Efficient em training of gaussian mixtures with missing data," *arxiv*, vol. 1209.0521v2, 2018.
[6] Y. B. Yann LeCun, Leon Bottou and P. Haffner, "Gradient-based learning applied to document recognition," *Procedures of the IEEE*, 1988.
[7] J. Rocca, "A gentle journey from linear regression to neural networks."
[8] S. van Buuren and K. Groothuis-Oudshoorn, "mice: Multivariate imputation by chained equations in r," 2011.

VI. Supplemental Materials

TABLE I
Overall Results For Zillow Dataset Imputation of Single Feature

Imputation Approach	Runtime (s)	Replacement MSE	Replacement r^2 Score	Prediction MSE	Prediction r^2 Score
Mean Imputation	0.004	1.000	-0.000	1.000	0.999463
No Imputation	NA	NA	NA	5.821	0.973449
Gaussian-Mixture Model	0.336	1.723	-0.729	1.018	0.999476
KNN Imputation	1.201	0.158	0.836	1.212	0.999374
Polynomial Regression	9.793	0.171	0.817	0.958	0.999298
Random Forest	25.722	0.157	0.833	0.990	0.999450
Artificial Neural Network	388.990	0.152	0.836	0.982	0.999467

There are 73728 total instances. 25260 of those are missing no data, and 48468 are missing data only in a single instance. Note all MSE values are normalized with respect to those for predictions using Mean Imputation, as this is taken to be our baseline approach. The r^2 scores are provided to make the results easier to visualize and compare between datasets. The more natural error metric for considering these replacements and predictions is the MSE ratio.

TABLE II
Overall Results For Facebook Dataset Imputation of Single Feature

Imputation Approach	Runtime (s)	Replacement MSE	Replacement r^2 Score	Prediction MSE	Prediction r^2 Score
Mean Imputation	0.001	1.000	-0.001	1.000	0.507
No Imputation	NA	NA	NA	0.145	0.943
Gaussian-Mixture Model	0.063	1.343	-0.287	1.079	0.490
KNN Imputation	0.005	0.661	0.396	1.148	0.515
Polynomial Regression	0.024	0.929	0.105	1.105	0.496
Random Forest	0.101	0.608	0.473	1.160	0.481
Artificial Neural Network	4.756	0.675	0.254	1.113	0.507

There are 500 total instances. 250 of these are missing no data, and 250 are missing data only in a single feature. All MSE values are normalized with respect to those for predictions using Mean Imputation, as this is taken to be our baseline approach. The r^2 scores are provided to make the results easier to visualize. The more natural error metric for considering these replacements and predictions is the MSE ratio.

TABLE III
Results For Facebook Dataset Imputation of 3 Feature of 5 each missing half of data

Imputation Approach	Runtine (s)	Prediction MSE	Prediction MSE-Std Dev	Prediction r^2	Prediction r^2-Std Dev
Mean Imputation	0.004	1.000	1.000	0.41	0.08
No Imputation	NA	1.106	1.419	0.39	0.10
MICE	0.063	1.063	0.990	0.39	0.06

There are 500 total instances. Of the 5 features in this model we have randomly removed half of data from top 3 most correlated features. This leaves only 62 instances missing no data, and using an 80/20 split only 13 instances in our test set for evaluation with no imputation. What is reported is the average MSE over 10 runs of the code. This is done to remove bias arising from selection of such a small test set for the no imputation case. Note all MSE values are normalized with respect to those for predictions using Mean Imputation, as this is taken to be our baseline approach. The r^2 scores are provided to make the results easier to visualize. The more natural error metric for considering these replacements and predictions is the MSE ratio.