CS 536: Final Project - Missing Value Imputation

Nitin Reddy (nrk60)
Ranjitha Korrapati (rk850)
Siddharth Sundararajan (ss3177)

May 12, 2019

Abstract. The aim of this project is to construct a system for predicting or interpolating missing features (frequently given as empty or NA) from the present features in new records. In this work, we compare five imputation methods: Mean-Mode, Linear and Logistic Regression, PCA (Principal Component Analysis), Neural Networks and EM (Expectation Maximization). Note that EM algorithm was used to impute missing values for numerical variables and not for categorical variables. The analysis was conducted on the dataset from - The Original Many Labs Project (ML1 Dataset). Two evaluation metrics were used for this analysis - MSE (Mean Squared Error) and CE (Classification Error). Our results suggest that linear and logistic regression and mean-mode imputation are two good imputation methods to consider further and experiment with.

1 The Data

For this project, we choose the dataset from the study - The Original Many Labs Project. This data comes from psychology studies in the Many Labs series, an attempt to test the replicability or generalizability of psychological effects. Multiple labs attempted the same experiments, to determine the extent the results of those experiments generalized and could be relied on in diverse circumstances. Being psychology studies, this data set comprises records of personal answers and reports from subjects on a variety of questions relating to topics like perception and mental biases. Additionally, it includes demographic information about the subjects and information about the researchers and experimental environments. All are potentially useful for the problem of prediction and interpolation.

2 Requirements

2.1 Description of Model

What approach did you take? What design choices did you make, and why? How did you represent the data? How can you evaluate your model for goodness of fit? Did you make an effort to identify and exclude irrelevant variables?

How did you handle missing data?

There are three types of data present in this dataset. They are - Numerical, Categorical and Textual data. The classification of data into their corresponding types were done by exploring the data and cleaning it. Pre-processing was done on these types individually after they were categorized (into numerical, categorical and text). Numerical variables were normalized to a scale of 0 to 1. This was done by dividing each value in the column with the maximum value present in that column. Note that this was done only before training. With respect to categorical variables, the categories were changed to start from 0. For example, if *sex* had values *male* and *female*, this would be changed to values 0 and 1. Also, categories that had text, were converted to lower case. Note that no analysis was done on the textual data.

Irrelevant variables were identified and excluded for the purpose of analysis. This was done by going through the data and identifying variables that have timestamp, unique ids, URLs, columns with only one unique value etc. Note that a good number of columns were removed. The total number of columns considered after this were 210. The split of columns considered across different column types (numerical, categorical and textual data) are seen in the table below.

Data type	Numerical	Categorical	Textual	Bad
No. of columns	60	150	6	166

Table 1: Distribution of columns based on column type

In this work, we handled missing data. During basic exploratory data analysis, we identified seven examples of missing values that were present. They are seen in the table below. Note that we replaced all the different missing value types to numpy nans (np.nan).

Missing Value Types	”	.	NaT	n/a	N/A	np.nan

Table 2: Different missing values present in the dataset

It is important to represent data well before building any model. Therefore, we created a dataframe comprising of clean numerical and categorical variables (that were chosen during the data exploration phase). Each numerical column was continuous (and was scaled between 0 and 1 just before training). Similarly, each categorical column was classified again (as mentioned in the example above) for ease of use. Note that this does not affect the analysis in any way.

Our approach for this problem of missing value is as follows. Note that this is a recursive process which runs till there are no missing values.

- Select columns that do not have missing values. Let us assume that these columns are called as *D*. Identify the column with the least number of missing values from original data (Let us call this as *O*), in case it is not yet filled in D.

- The columns that were identified (in the above step) are taken as dependant variables for the algorithm (that we select). We then consider the data D along with this new column and drop the rows with missing values in the identified column to obtain a complete data (which we will call as C).
- Split the data C into two parts. Using the first part, we train the model. With the second part (which is smaller and called the holdout data), we estimate the error to find the best model.
- We then apply a cross validation technique called K-Fold cross validation on the training data. Note that we also scale the data on the training data (as mentioned earlier) before applying the algorithm which will help to identify the best model.
- MSE or Mean Squared Error (in case of numerical variables) and CE or Classification Error (in case of categorical variables) is calculated on the second part of the data (called as the holdout data) to estimate the error of imputation.
- The best model is used to fill the actual missing values in the original data (O).

In this work, three design choice were made. First, min max scaling was performed on the training data. This was done for two reasons - to make comparisons uniform across algorithms and to maintain the same scale for all numerical variables. Second, k-fold cross validation was done to avoid overfitting. Finally, a test data is set aside (as per the method mentioned above) to estimate the error. For goodness of fit, we are using the metrics MSE and CE, corresponding to numerical and categorical variables. Note that these design choices are applied while implementing each algorithm.

2.2 Comparison of Algorithms Implemented

2.2.1 Mean-Mode

Given your model, how did you fit it to the data? What design choices did you make, and why? Were you able to train over all features? What kinds of computational tradeoffs did you face, and how did you settle them?

A simple way to impute missing values is by using the mean-mode method. In this method, if the variable is numerical, the missing value is replaced by the mean of the observed values in that column. Similarly, in case the variable is categorical, the missing variables are replaced by its mode. This technique was used to fit to the data.

Training using mean-mode imputation was possible for all the columns that were considered for the analysis. As it is a simple algorithm, it had low running time and computational complexity.

How did you try to avoid overfitting the data? How did you handle the modest (in ML terms) size of the data set?

Data for each column, was split into training and validation in a random manner. The ratio of data that was set aside for validation was 0.1. All the validation data was treated as missing and replaced by mean or mode (depending on the variable type) of the training data. This process of splitting and calculating was done for n iterations. The number of iterations, n was decided to be 5. The error was then estimated by taking the mean of MSE of the true values in the validation data against the imputed values for the numeric variable. Similarly, for the categorical variable the metric was classification error.

Where is your model particularly successful, where does it lack? Does it need a certain amount of features in order to interpolate well? Are there some features it is really good at predicting and some it is really poor at predicting? Why do you think that is?

We observed that the sum of MSE for all the numerical variables was 6.44 and the sum of CE for all the categorical variables was 39.40. Below we see the errors observed for the numerical and categorical variables. It is observed that as the error increases, the number of missing values also increases steadily. Also, it is important to note that initially, the mean imputation technique is able to capture the missing value information well.

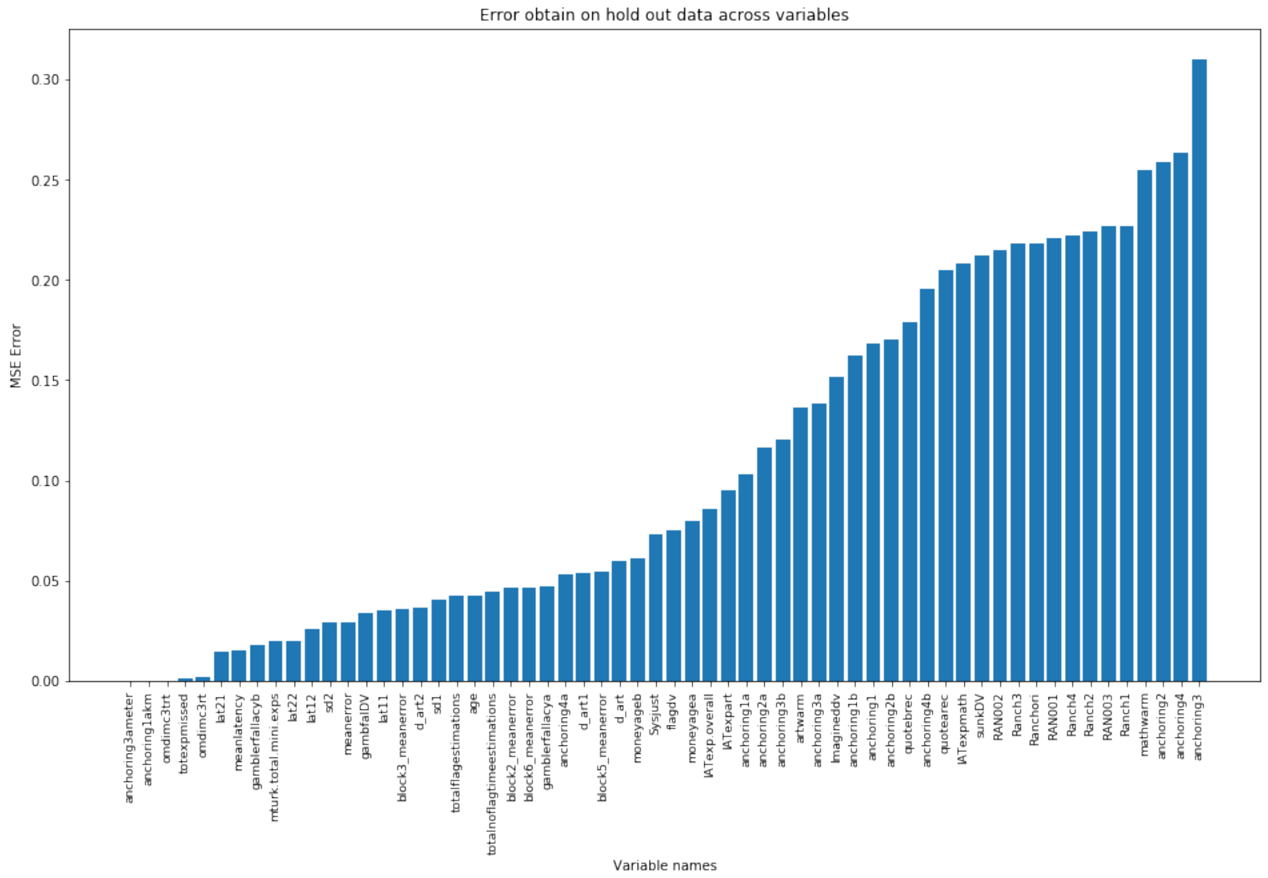

Figure 1: Errors in numerical Variables (Mean)

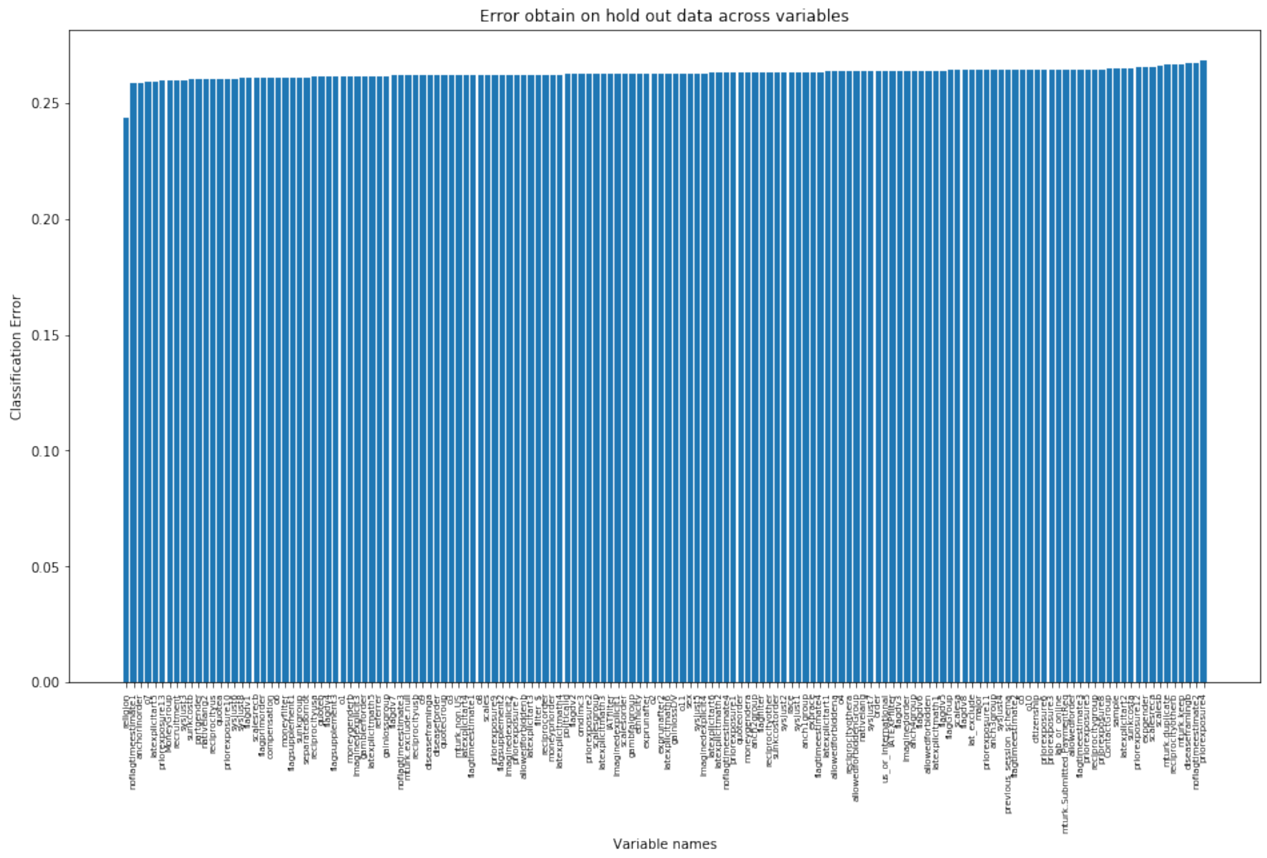

Figure 2: Errors in Categorical Variables (Mode)

The errors observed on the categorical variables when using mode imputation is almost uniformly distributed. Also, a observation to make here is that the maximum error seen for a categorical variable is a little over 0.25. This is good because the maximum error possible is 1.0. But, this is not good for all cases as the number of classes vary for each variable. Therefore, overall the classification error is not the best solution.

Few numerical variable with the least error that we observed are *anchoring3ameter*, *anchoring1akm*, *omdimc3trt*, *totexpmissed*, *omdimc3rt*. Note that the MSE observed for these variables was less than 0.01. Similarly, for categorical variables only one variable *religion* did very well. However, classification error was observed to be almost constant and same for all the variables.

To conclude for the mean-mode imputation technique, it is a simple baseline model without machine learning to begin with. The technique works well. It does not do very well when the number of missing values are more. This is expected as the observed values present may not be actually following an underlying distribution (that can be defined). For interpolation, there is no minimum columns needed but it does well if there are good number of columns with complete data. As seen earlier, this method predicts poorly in cases where observations are less.

2.2.2 EM Algorithm - For Numerical variables

Given your model, how did you fit it to the data? What design choices did you make, and why? Were you able to train over all features? What kinds of computational tradeoffs did you face, and how did you settle them?

EM algorithm or Expectation Maximization algorithm is a technique widely used to impute missing values. The methodology is a probabilistic approach. The advantage here is that with any starting point, the algorithm will converge to its true underlying distribution. In this work, we have only imputed missing values for numerical variables. Note that this concept can be extended to categorical variables. This algorithm is a two step recursive procedure until convergence. The first step is called Expectation, where the missing values are estimated with the distribution followed by the observed data (using its parameters, which are mean and standard deviation in this case). The second step is called Maximization, where the parameters are restimated using the newly imputed values. For convergence, we have set a tolerance of 0.1.

Note that as mentioned earlier, the same design choices are made here. The algorithm was trained on all the numerical variables and not the categorical variables. The method had to be slightly changed for the categorical variables (but this was not implemented due to time). This is a simple algorithm and hence, is easy to run and not computationally expensive. Therefore, hyperparameter tuning in this algorithm was relatively easy.

How did you try to avoid overfitting the data? How did you handle the modest (in ML terms) size of the data set?

As EM algorithm works on each feature, the data split was done at a column level into training and validation in a random manner. The split ratio was 0.1. This was run for n iterations, where, n was decided to be 5. The error was then estimated by taking the mean of MSE of the true values in the validation data against the imputed values for the numeric variable.

Where is your model particularly successful, where does it lack? Does it need a certain amount of features in order to interpolate well? Are there some features it is really good at predicting and some it is really poor at predicting? Why do you think that is?

The total MSE observed was 12.41 for all the numerical variables. In the figure below, we can see that the error is increasing. Also, we observe that it is minimum for the first few variables and them it steadily increases. This trend observed is very similar to the trend seen above with mean imputation. This is expected as the observations for the normal distribution is mainly around the mean. Overall, the MSE is resonably good, except for the last variable.

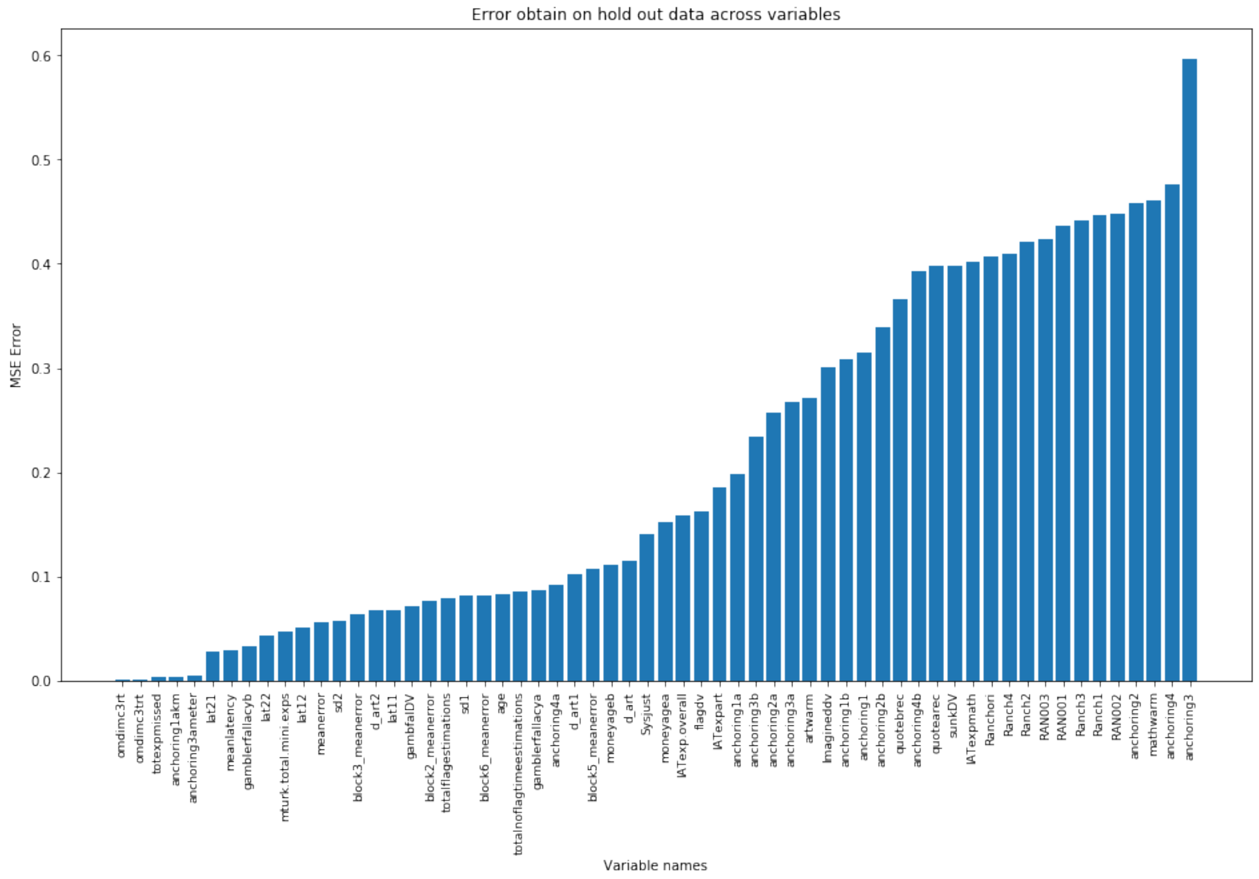

Figure 3: Errors observed in EM Algorithm

To conclude this simple machine learning techniqueis good. It does well even when the missing values are not too much. The model lacks when the column feature does not have an underlying feature. Overall, compared to mean imputation is not good.

2.2.3 PCA Imputation

Given your model, how did you fit it to the data? What design choices did you make, and why? Were you able to train over all features? What kinds of computational tradeoffs did you face, and how did you settle them?

PCA stands for Principal Component Analysis. It is usually used in Dimensionality Reduction to project data to a dimensional space of features that actually are significant. One of the major perks of using PCA imputation is it considers both categorical and numerical features together towards the imputation process.

If there are k1 numerical variables and k2 categorical variables, then there are k1+k2 features in total. Before proceeding to implementing PCA imputation, the data has to be processed for the model to give accurate results. So , the we perform one hot encoding of the k2 categorical variables , each dummy variable representing a particular category of a categorical variable.

How did you try to avoid overfitting the data? How did you handle the modest (in ML terms) size of the data set?

After the encoding of categorical variables, we standardize / normalize the data in order to ensure that all the variables are confined to a particular scale and become comparable. Numerical variables are divided by the standard deviation of the variable and the categorical variables are divided by the proportion of the number of times that particular category appeared. These ratios are taken after each point is made to be centered at their respective column means.

After the standardization, we can perform Singular Value Decomposition on the data matrix. Singular Value Decomposition is the process of factor a matrix such that it can analyzed to get the components or features that are significant. After the decomposition, the data is reconstructed and only the values that are missing, are imputed. This process is iteratively performed until we get a tolerance error of a particular threshold.

Where is your model particularly successful, where does it lack? Does it need a certain amount of features in order to interpolate well? Are there some features it is really good at predicting and some it is really poor at predicting? Why do you think that is?

The model can be evaluated for the goodness of fit by creating a masked data set out of our actual data set. This can be achieved by hiding few variables that are present in the data set and basically predicting them back to check if the values are the same or close enough.

The features were looked thoroughly to see which features might contribute to decision making process or imputation in our case. In this way, we removed our irrelevant features.

The missing data was initially imputed with mean for numerical variables and proportions for categorical variables. The other steps have been discussed earlier.

2.2.4 Linear and Logistic Regression

Given your model, how did you fit it to the data? What design choices did you make, and why? Were you able to train over all features? What kinds of computational tradeoffs did you face, and how did you settle them?

Using Linear-logistic Regression is one of the ways to handle both categorical and numerical data. In Linear-Logistic regression, we take the set of columns that are totally filled and make it a training set. Ten percent of the data is set aside as test data to test how our model works. Initially, the columns with the least number of missing values is taken and made as the dependent variable. If the variable turns out to be numerical, Ridge Regression is applied else if it is a categorical with binary classes, logistic regression is applied. If there are multiple classes Soft max is performed in order to impute the missing values. Once the values of a particular column were imputed, this column was added to set of independent variables and the process was continued to impute other columns.

The design choices were made in a way to accommodate numerical and categorical variables in the best possible ways. In Ridge Regression, we are trying to perform Hyper Parameter Tuning in order to find the model with parameters that best performs our task. For this sake, we have considered the parameters in Ridge to be 0.1, 0.25, 0.5, 0.75, 1, 1.5, 2, 5. Similar hyper parameter tuning was performed on logistic Regression. All combinations of the epochs and learning rates wee built to find the best model. The considered epochs were 30, 50 ,100, 150, 200, 500, 1000 and the learning rates were considered to be 0.001, 0.01, 0.05, 0.1 and 0.5. We have considered batch sizes up to 100 in columns with multiple classes.

How did you try to avoid overfitting the data? How did you handle the modest (in ML terms) size of the data set?

We trained the data over the features incrementally adding as they are filled. So initially , only the columns with all values present were taken and combined with other columns as they are being filled.

Iterating through all the combinations of parameters was computationally challenging. All the combination took around 26 hours. We could not try more parameters and fine tune the model. So we had to choose parameters wisely and run less number of epochs to ensure computation feasibility.

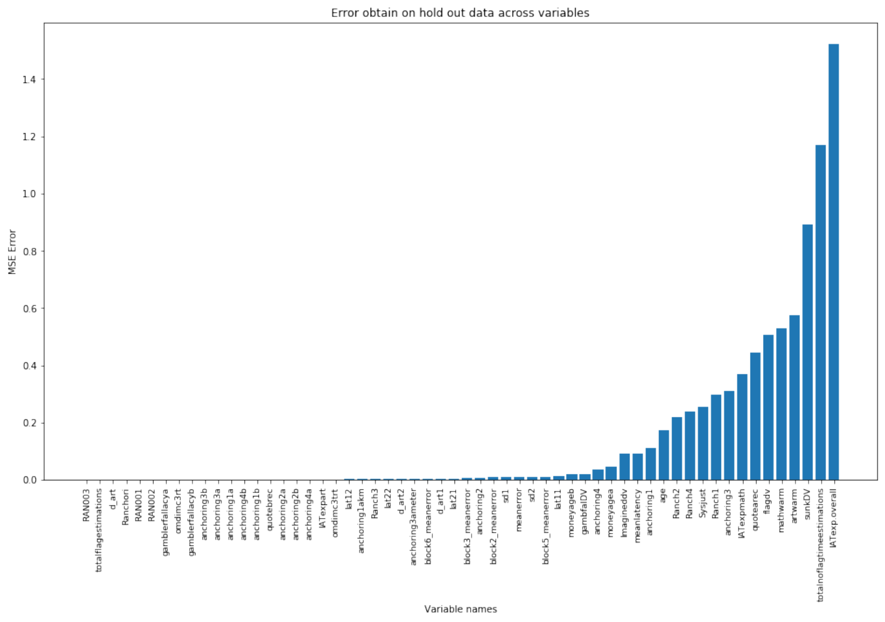

Figure 4: Errors observed in numerical variables using Linear Regression

Where is your model particularly successful, where does it lack? Does it need a certain amount of features in order to interpolate well? Are there some features it is really good at predicting and some it is really poor at predicting? Why do you think that is?

In linear regression , we can see that the error is almost negligible for most of the initial variables. The errors start to increase towards the end and reach maximum. So we can conclude that linear regression gives best performance as far as numerical variables are concerned. The average error in linear regression is 104 and we can see that it performs poorly only for a few variables, removing the four poorly performing variable we can see that the average error drastically drops to 4.43.

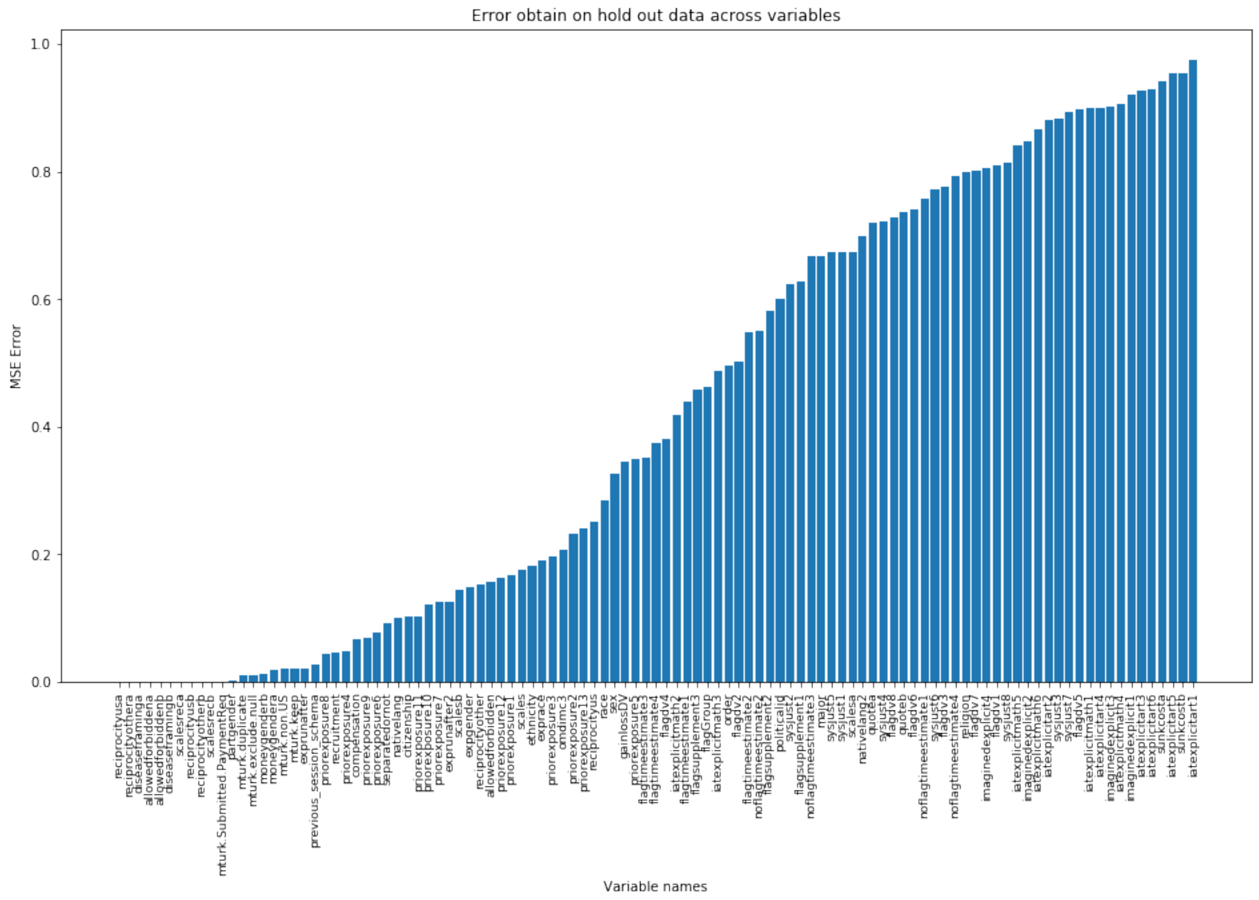

Figure 5: Errors observed in categorical variables using Logistic Regression

The errors in categorical variables show an increasing trend. Few of the initial variables have almost negligible errors. The increase starts rapidly as compared to linear regression in numerical variables and then reaches maximum towards the end. The average error in logistic regression is found to be 44.2.

2.2.5 Neural Networks

Given your model, how did you fit it to the data? What design choices did you make, and why? Were you able to train over all features? What kinds of computational trade offs did you face, and how did you settle them?

Same pipeline as mentioned earlier was used to pick the dependant and independent data. Once a model was trained on particular variable, it was used back to predict the missing values in the same variable. The model used features that had no missing values. Finding the right set of parameters of neural network model was challenging task. There was lot of parameter which required tuning and various options for them made the neural network models to be computationally challenging. We had just 2 layers and 4 set of learning rate to find a best model among these parameters. Various epochs, batch size were not experimented with. Also, other combinations of layers and neurons were also not looked into.

The design choices were as follows:

Neural network was implemented using the back propagation algorithm. The algorithm had following parameter to tune :

- Hidden Layers - Configuration of neural network tried were [3,3], [5,5]
- Learning Rate - 0.003, 0.05, 0.1
- Activation function for hidden layer - Sigmoid function
- Activation function for outter layer - Sigmoid for binary class, Linear for numeric and Softmax for multiclass
- Batch size - 250
- Number of Epochs - 10

The Neural network model was trained for all the combinations of Hidden Layers and Learning Rate mentioned above.

How did you try to avoid overfitting the data? How did you handle the modest (in ML terms) size of the data set?

As part of pipeline, Cross validation technique KFold was used which helped in avoiding over fitting of the model. The error was calculated on hold out data, which was model had never seen before.

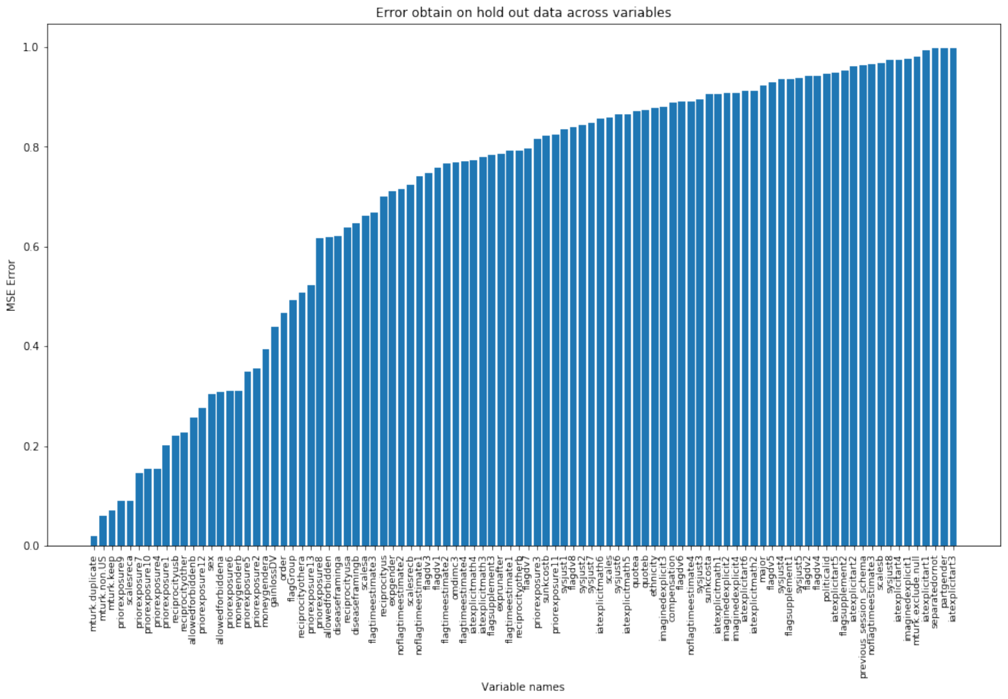

Figure 6: Errors observed in categorical variables using Neural Networks

When the missing values in categorical variables are imputed using Neural Networks , the error shows increasing trend. We can see that the error is less than 0.6 for about 30 percent of the data. The average error for Neural Networks in categorical variables is 67.005.

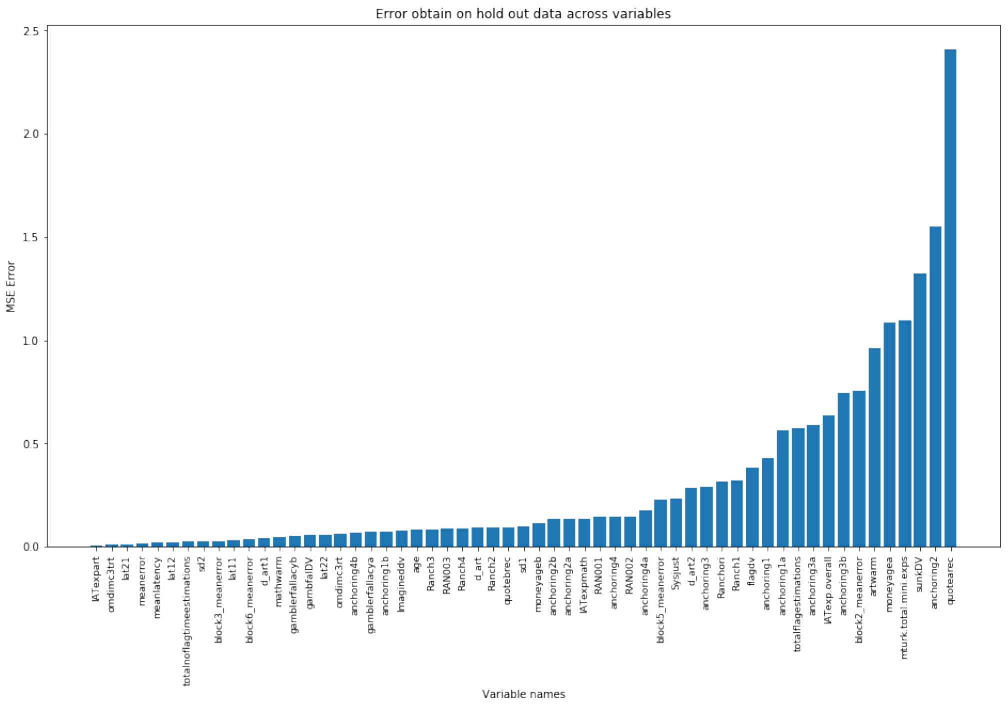

Figure 7: Errors observed in numerical variables using Neural Networks

We can see that Neural Networks performed well for around 75 percent of the numerical variables with error less than 0.5 . The average error is found to be 17.4.

Where is your model particularly successful, where does it lack? Does it need a certain amount of features in order to interpolate well? Are there some features it is really good at predicting and some it is really poor at predicting? Why do you think that is?

The model seems to do a good job with numeric variables, but the error on the categorical variables was observed to be very high.

2.3 Analysis of Data

What features were particularly valuable in predicting/interpolating? What features weren't particularly useful at all? Were there any features about the researcher/experimental environment that were particularly relevant to predicting answers - and if so, what conclusions can you draw about the replicability of those effects?

The features with the lowest number of missing values were particularly valuable in making predictions or interpolating. As the percentage of missing values increases , the variables tend to become less expressive and less contributing towards the predictions.

We can see that in Linear Regression Model for numerical variables, the variables that have less than 0 percent of missing values have almost 0 to negligible error. Some of those variables include *Ranchori, anchoring2a, anchoring2b, Ranch3* have very less percentage of missing values and also showed very less errors and contributed the most.

The variables with less number of missing values also help better in finding the underlying distribution of the data. If the percentage of missing data points lie within 10 - 15 percentage, the features tend to contribute considerably with low errors.

There were 166 columns that were not useful in in interpolation. These variables included time stamps, id's and other variables. Some of these variables have the same value throughout the column or they have unique values. They also do not convey much to the interpolation of other values.

Method	**MSE**	**CE**
Mean-Mode	6.44	**39.4**
Linear Regression*	**4.43**	-
Linear Regression	**104**	-
Logistic Regression	-	44.2
Expectation Maximization	12.41	-
Neural Networks	17.4	67

Table 3: Comparison of the average MSE and CE for algorithms implemented. * is the MSE without the top 4 variables.

The above table is the comparison of the different algorithms implemented in this work. It is good to see that the average MSE if the best for linear regression, without including the top 4 variables. The top 4 variables contribute to high error as the percentage of missing values are very high (more than 80%). With regards to CE, the best model is the mode imputation. But, the CE for logistic regression is close to that of the mode imputation.

2.4 Data Generation - Missing Value Imputation using ML models

Use your system to try to generate realistic data, and compare your generated data to the real data. How good does it look? What does it mean for it to 'look good'?

3 Bonus

3.1 Bonus 1

Build your system to include processing and analysis of natural language answers, to try to use them to inform prediction of other features. How can you represent natural language answers in a useful way? What language processing is necessary to be able to use them? Do they actually provide useful information for the prediction problem?

Text column like 'expcomments' and 'imagedescribe' will play good role in determining the outcome in some of the columns of the data. These comments mention the state of experiment was conducted or the thoughts process of the survey taker. A bit of processing and cleaning of this column can be a good predictor.

Stop word removal, converting the string to lower case, and then applying stemming / lemmatization will lead to the words to their base or root form. One we have words cleaned and to their root form. We can apply term- frequency and inverse document frequency to identify important words in occurring the comments. These words can then be used to create the one hot encoding. While the same one hot encoding can be used in any of the above algorithms to fill in the missing values.

Also, apart from one hot encoding, word embeddings can also be used to get a vector representation of the words with context. In this case, we tf-idf multiple by the word vector of each word will give use the weighted vector. Then average of all these weighted vector for the important words in the column will lead to word vector which represent the whole column. This vector can then be used as an predictor.

Same goes with the other columns - ’feedback’, ’text’. However, these columns have very few unique values (59) and can be reduced to much lower by cleaning (less than 20). Then these columns can be treated same as categorical variable.

3.2 Bonus 2 and 3

Build your system to include prediction about the natural language answers. Based on available features, what can you predict about a person’s answers to natural language questions (if not the actual answers themselves)? How can you assess the quality of these predictions? Build your system to include generation of natural language answers to questions, based on available features. How can you model this generation problem, and how can you evaluate the quality of the answers?

Column ’imagedescribe’ is an interesting column to predict. A deep learning model like Sequence models can help in generating such text present in the ’imagedescribe’. Based on available columns from the pipeline can then later be used for generating the missing values in the ’imagedescribe’.

A dense to sequence model will comprise of encoder and decoder. Encoder take in the available columns as inputs and passed a latent vector to the decoder. Ecoder here is a just an dense layer neural network. Where as in case of decoder it will be a recurrent neural network (LSTM / GRU). The output from the encoder to decoder will try to generate the word one by one.

Word embedding and one hot encoding both can be used to represent the words. We might not achieve great accuracy as the amount of data is less and the method mentioned above generally requires large amount of data.

Same method can be applied to predict a person’s answers to natural language questions.