Missing Data

Contents

Theory

Within a data set there are three mechanisms by which data can be missing:

- Missing Completely at Random (MCAR)
- Missing at Random (MAR)
- Not Missing at Random (NMAR)

An example:

The effect of missing mechanisms can be illustrated via an example - consider we are doing an experiment with a group of students. We wish to measure their heights to introduce the concept of the mean, μ. Start by imagining the true heights, h_{true}, follow a normal distribution:

$$h_{true} \sim N(1.6, 0.15)$$

And without missing data we would observations 1.71, 1.48, 1.48, 1.24, 1.44, 1.48, 1.6, 1.43, 1.4, 1.34, 1.73, 1.86, 1.63, 1.74, 1.84, 1.71, 1.88, 1.59, 1.56, 1.75, with mean 1.59.

MCAR

Now, imagine some students are missing - say due to sickness. Every student has the same chance of being missing, resulting in data 1.71, NA, 1.48, NA, 1.44, 1.48, 1.6, 1.43, 1.4, 1.34, NA, 1.86, 1.63, NA, 1.84, 1.71, 1.88, 1.59, 1.56, 1.75 and a mean 1.61. As every student has the same chance to be missing the method is MCAR and the mean estimate is similar to the true mean.

NMAR

Instead of being missing due to sickness, imagine some students simply refuse to give data - say if they are shorter than 1.45m and don't want to participate. Instead of being normal, the data follows a truncated normal distribution:

$$h_{observed} = \begin{cases} h_{true} & \text{if } h > 1.5 \\ \text{Missing} & \text{Otherwise} \end{cases}$$

and we would have values 1.71, NA, NA, NA, NA, NA, 1.6, NA, NA, NA, 1.73, 1.86, 1.63, 1.74, 1.84, 1.71, 1.88, 1.59, 1.56, 1.75 and mean value 1.72 - greater than the MCAR estimate, and hence missingness has introduced bias.

Table 1: Heights as a function of gender in a MAR model.

Gender	Actual Height	MAR Height	Imputed Height
M	1.71	1.71	1.710
M	1.48	NA	1.474
M	1.48	1.48	1.480
M	1.24	NA	1.474
M	1.44	1.44	1.440
M	1.48	NA	1.474
M	1.60	NA	1.474
M	1.43	NA	1.474
M	1.40	1.40	1.400
M	1.34	1.34	1.340
F	1.73	1.73	1.730
F	1.86	1.86	1.860
F	1.63	1.63	1.630
F	1.74	1.74	1.740
F	1.84	1.84	1.840
F	1.71	1.71	1.710
F	1.88	1.88	1.880
F	1.59	1.59	1.590
F	1.56	1.56	1.560
F	1.75	1.75	1.750

MAR

For MAR the situation needs to be more complex. This time imagine as well as measuring heights, we measure gender. Height and gender might be related, but also lets imagine that no all the guys want to to give their heights - not because of the height but lets imagine half of the men didn't have readings taken.

We'd end up with data as shown in Table 1 - where the `MAR Height` column is the values collected. because height depends on gender **and** height is missing because of gender there is a bias and $\mu_{MAR} = 1.64$, greater than the true value.

Unlike **NMAR** bias - **MAR** bias can be corrected via imputation. Here we apply mean imputation, creating the `Imputed Height` column, with average value $\mu_{Imputed} = 1.6$.

Detecting Missingness

Imputation can allow us to deal with MAR - but the problem is how do we detect it? The first step is to make a new variable, Z, that represents the missingness of a variable, Y:

$$Z_i = \begin{cases} 1 & \text{if } Y_i \text{ is missing} \\ 0 & \text{otherwise} \end{cases}$$

Which then lends itself to analysis via:

- Logistic Regression (which will assume linearity between the features and response)
- Classification algorithms (which can be more flexible)

One of the easiest classification algorithms to trial is a Binary Decision Tree (BDT) - in which we construct a tree out of rules with two results. The BDT algorithm has two key benefits:

- Does not require linearity of response
- Quantifies the contribution of each feature via 'Feature Importance' measures

and hence can be used as a high level pass through the data to quickly build interpretable models.

Quality of Missingness Model

The ability to build a BDT does not guarantee that the model will accurately reflect the model of missingness. Instead of learning some generalizable pattern - it may instead simple memorize the training data. To counteract this, we can hold back a quantity of the data as a validation set - testing to what extent the proposed model can accurately predict the pattern of missingness in effectively unseen data.

Quantifying how well a classifier performs requires the consideration of two terms:

- Precision: $\frac{\text{TP}}{\text{TP+FP}}$
- Recall: $\frac{\text{TP}}{\text{TP+FN}}$

i.e. precision is the rate at which positive predictions are true and recall is the rate at which positive cases are detected. So a Precision of 0.9 means 9 out of 10 cases labelled Missing were missing - and a Recall of 0.9 means 9 out of 10 cases of Missing were labelled accurately.

The two terms can be combined to form the **F1-score**:

$$F1 = 2\frac{\text{precision x recall}}{\text{precision} + \text{recall}}$$

Embedding of data

The last consideration is how to prepare the data we will use to predict missingness - i.e. the other variables in the data set. Typically data will break down into two forms, numeric and categorical, and to feed this data into the BDT algorithm it has to be preprocessed:

- Numeric data - missing values need to be imputed. Typically we will be working with a broad data set, with multiple variables each with their own chance of having a small amount of missing data. Consider if we have 20 variables, each with a low quantity of data missing e.g. 5%. If the missingness of one variable does not overlap with the missingness of any other variables - there is not a single compete case. Hence, we need a way to impute missing values - the simplest approach to which is mean imputation.
- Categorical data - variables need to be converted to a numeric representation. Here we use a standard approach called the One-Hot vector embedding, treating missing as a possible value of the categorical variable.

Pseudo code of the approach

To aid the reader - we outline here the pseudo code for the analysis being carried out:

```
load a rectangular data set to memory, assign to data
for column Y in data:
    if Y contains a missing observation:
        set Z equal to a binarized version of Y
        set X equal to all non-Y columns of data

        split X into numeric_X and categorical_X

        set imputed_X equal to numeric_X with imputed missing values
        set one_hot_encoded_X equal to an embeded version of categorical_X
        set expanded_X equal to the column bind of imputed_X and one_hot_encoded_X

        divide (expanded_X, Z) into (train_X, train_Z) and (test_X, test_Z)

        set model equal to a BDT trained with (train_X, train_Z)
```

```
        test model on (test_X, test_Z) and extract the F1 score
        extract feature importances from model
```

Which can be done for a csv file as:

```
data = pd.read_csv("file_path.csv")
missing = missing_classifier.MissingClassifier(data)
result = missing.test_all_columns()
result.to_markdown()
```

Individual jobs break down as

```
load a rectangular data set to memory, assign to data
[pd.read_csv(...) or pd.read_excel(...)]
for column Y in data:
    if Y contains a missing observation:
        set Z equal to a binarized version of Y
        [Z = data[Y].isna()]

        set X equal to all non-Y columns of data
        [X = data.drop(columns=[Y])]

        split X into numeric_X and categorical_X
        [see missing_classifier.MissingClassifier.divide_by_data_type]

        set imputed_X equal to numeric_X with imputed missing values
        [see missing_classifier.MissingClassifier.prepare_numeric_data]

        set one_hot_encoded_X equal to an embeded version of categorical_X
        [see missing_classifier.MissingClassifier.prepare_categorical_data]

        set expanded_X equal to the column bind of imputed_X and one_hot_encoded_X
        [expanded_X = pd.concat([numeric_X, categorical_X], axis=1)]

        divide (expanded_X, Z) into (train_X, train_Z) and (test_X, test_Z)
        set model equal to a BDT trained with (train_X, train_Z)
        test model on (test_X, test_Z) and extract the F1 score
        extract feature importances from model
        [see missing_classifier.MissingClassifier.test_column]
```