The many faces of missing data

Tom Breur
28 March 2023

Data comes in three varieties: Good, Bad, and Ugly. I say this tongue-in-cheek, because so much of my time working with data and building models seems to go up in smoke. There are many terms in use like: "data preparation", "data wrangling", "data curation", "data cleansing", and others. Whether you refer to this as *cleaning* or *preparation*, somewhere along the line flaws will need to be addressed. From my experience, the lion's share of those flaws have their root cause in shortcomings during upstream data integration. I sometimes think of this as "technical debt" which remains from imperfect data governance that needs to be paid off. Data management and data stewardship are rarely front and center on the corporate agenda. But in between pulling source data to work with, and tagging your dataset as "ready for analysis", at some point you *have* to "pay the piper."

Here is an interesting paradox: the more diverse your data sources for analysis, the more likely that you will find interesting novel patterns that can be used for commercial gains. However, if data sources cannot readily be joined, it's probably because no one has attempted to consolidate these data until now. Imperfect data integration efforts lead to fallout that partially will need to be deferred: it simply isn't feasible or economical to attempt to fix each and every data quality problem for the purpose of assembling an ad hoc data set for analysis. Therefore, you need to accept the fact that 'some' of your data will be and remain missing.

When I say data are "missing" in the context of this paper, it can mean several things. Missing can mean a value was supposed to be present, but for unknown reasons wasn't available. "Missing" could also mean that a value would not be expected, because it doesn't apply. For instance, when a record contains a Boolean value that indicates if a customer has a savings account, then for people who do not have such an account, a missing value for "Balance" is to be expected. Here missing indicates something like "does not apply." It will be obvious that these two different scenarios, expected or unexpectedly missing data, need to be dealt with in completely different ways.

An example

Let's look at some straightforward examples to clarify. We are joining customer transaction data from disparate source systems. One of the systems lists a column "Trx_date", the other data source has "Transactiondate", and they appear to refer to the same thing. The process of finding records across disparate systems that refer to the same entity is called "record linkage" or sometimes "data linkage" (an entire research field in and of its own). Prof. John Talburt wrote "Entity Resolution and Information Quality" in 2010, and I (still) consider that one of the classics in this field.

When we try to join data from these two source systems, it turns out that the field "Trx_date" has values like 06/20/18, whereas "Transactiondate" lists 20-JUN-2018 08:03 for the corresponding record. For this example, we'll assume that a deterministic record match has been accomplished. Although phenomena like these are quite common, record linkage is severely encumbered when key identifiers for the same entity (like these two "Date" fields) are recorded differently between source data sets.

There are two issues at play here: apart from the fact that "Trx_date" doesn't list the millennium in which the transaction occurred (the two digit value for "millennium" is missing), it is also recorded at a different *grain*: Date versus DateTime. We'll conveniently ignore the missing millennium problem, that is to say we make the simplifying assumption that the records all originated form the same millennium. Since the timestamp in "Trx_date" is missing and cannot easily be backfilled, we proceed with the data "as is": for the sake of discussion we need to decide that the effort to fix this problem upstream isn't realistic given our timeline (i.e. project goals). Effectively, we choose to roll up (aggregating) "Transactiondate" from DateTime to the Date level. Consider this an example where a value was supposed to be present, but not available. After all, the transaction *did* get recorded at a specific point in time (even if the exact time it occurred didn't get logged).

Our example for "Transactiondate" was a fairly simple and straightforward one. It doesn't take too much imagination to extend this single field to multi-field analogies like "Name", "Address", or more elusive compound keys. Address matching is a research field in its own right with intricate standardization issues, and an extra layer of complexity because of different address formats per country. "Standardization" here sometimes also gets called "Normalization", but that is a term I try to avoid because normalization means something completely different in a data modeling context.

Consolidating data silos

The previous data field matching example was caused by siloed data. Invariably, data silos exist for a reason. Sometimes very subtle reasons. And sometimes they are very good and legitimate business reasons. Obviously this doesn't suit our data scientist very well, since he has been tasked to perform cross-functional analyses. One of the recurring patterns you can expect to run into is that the same column *name* will have slightly different meanings across two or more departments (and/or source systems). Superficially, the word "Customer" denotes the same entity, but the devil is in the details: Finance *defines* a customer differently from Marketing, Operations, and Fulfillment. Therefore, (some) elements from those two sets are expected to match, but the sets are only partially overlapping. The boundaries are not defined in the same way.

An obvious solution to resolve this ambiguity is to add a prefix that indicates the data source: Finance_customer, Marketing_customer, etc. In my younger years, I held high hopes that this process of conforming dimensions would ultimately allow for consistent reporting across departments – the holy grail of a "single version of the truth" (SVOT). Since then, I have come

to appreciate that although working *towards* data unification is noble and worthwhile, it will always remain a *journey*. My current thinking is that it's an illusion you will or even ever *can* arrive at nirvana with consistent definitions across the enterprise. From my experience, you cannot hope to capture every single user perspective present (the *root cause* for these slight differences in definition), while internal and external forces continue to operate: businesses are constantly in flux. Nonetheless, I recommend relentlessly working *towards* unification, as governed by business goals and priorities.

Missing at random – three variations

In statistics, we often refer to "missing at random" (MAR), and use it as a distinct term from "missing completely at random" (MCAR). Those two are usually set against "not missing at random" (NMAR). Methods for handling missing data rest on different sets of assumptions: some very strong, and some less stringent. In many cases these assumptions cannot be tested directly *in* the data, and therefore it is even more important that you understand the requirements imposed on the data – they differ across imputation methods. Unless you are familiar with these assumptions, you cannot make a prudent assessment how plausible they are.

The strongest assumption to make is that the data is MCAR ("missing completely at random"). In this case, the odds for a value Y to be "missing" is neither dependent on the value of Y itself, nor should "missing" be associated with *any* other variable in the dataset (that is: *any* variable that will be included in models). This pattern of missingness corresponds with laypeople's notion of "truly" at random. One way to test for this MCAR assumption is by recoding Y into a Boolean with 0 for "missing" and 1 otherwise, and then regressing all variables on Y. If *any* of the coefficients are significant, then the pattern in Y is *not* MCAR. You cannot test if missing is dependent on the value of Y *itself* (ie "income" is missing more often for the higher values) because that would require knowledge of the missing data elements.

MAR ("missing at random") is a weaker assumption, in that now "missing" values in Y may be associated with predictors X_1 through X_n, but *not* depend on the value of Y itself. For example: if you asked about "annual income", and people with higher income are more likely not to answer that question. Then "missing" is correlated with the value of income. That would be a violation of MAR. Strictly speaking, the assumption is actually relaxed somewhat further in that the pattern of "missing" in Y may not depend on Y *after* controlling for X_1 through X_n. Although this assumption is (much) more relaxed, unfortunately, there is no way for testing it in the data since that would require knowledge of the missing data points in Y.

NMAR ("Not Missing At Random") is the weakest assumption, which occurs when the assumptions for MAR are violated. In real-world datasets, claiming MAR is often unreasonable (obviously incorrect) and then standard imputation mechanisms are problematic because they often lead to biased and inaccurate estimates.

How to fix (impute) missing values
It is often desirable to "fix" records with missing values rather than delete these data. Typically to preserve valuable information contained in incomplete records. Principally, this serves to increase statistical power to the extent possible. Another reason to pursue imputation is to avoid bias associated with listwise deletion when the data are not MCAR.

Much work has been done in this area in the last few decades, and new methods and techniques are still being developed and tested. Besides the "traditional" methods, I will briefly describe three others:

- Traditional imputation methods
- Regression based imputation
- Maximum Likelihood imputation
- Iterative methods of imputation

Traditional imputation methods
A notable advantage of traditional methods is that it's a quick and easy way to replace all of your missing values with one simple and transparent (easy to check) procedure. It's also an approach that is readily available in most statistics packages. So far the good news. The bad news is that it is guaranteed to introduce bias in your dataset (see e.g.: "Missing Data in Regression Analysis" by Yoel Haitovsky, 1968). Also, because an array of constants deflates the observed variance for that field, as a side effect it renders all of the commonly used statistics (like the p-value for significance) unreliable. That is pretty bad news, especially for the unwary.

Besides imputing with either the Mean, Mode or Median, in the old days "cold" or "hot" deck methods were used. For reference, "deck" here refers to a stack of punch cards, to give youngsters a sense of how *old* these methods are. By choosing values pseudo randomly from a "deck", you avoid the problems of imputing with a constant. But nowadays these methods have since been replaced by more "intelligent" approaches (see next three methods: Regression, Maximum Likelihood, Iterative).

In summary, although many things may have been (or seemed) 'better' in the old days, that certainly does *not* hold for missing value imputation methods. It is bad practice, and risky at that, because of unreliable p-values, etc. In light of this, as a general rule of thumb casewise deletion –although it is known to add bias when the missing data pattern is MAR or NMAR– is often still preferable over imputation with a constant, *provided you can afford the loss in statistical power*.

Regression based imputation
When a column has missing values, one way to impute those is by building a regression model using the records that are populated, and plugging in the estimated Y ($\hat{Y}$) in the rows that were missing. Although the problems are (much) less severe, this approach –just like imputing with a constant– suffers from underestimating the variance for columns with missings. The reason for this can be intuitively seen because the regression model has an error term, but the $\hat{Y}$ that you

plug in is an "exact" estimate (has *no* error component). This will reduce the variance for that column, proportionate to the error component in the regression model.

Using imputation with regression is a considerable improvement over "traditional" imputation (with constants), but unless you add a stochastic error term to your estimates, it still leads to underestimation of variance. As a side effect of underestimating the error variance are downward biased p-values which carries the risk of inadvertently flagging effects as significant, when in reality they are not (hence inflated Type I error).

Maximum Likelihood (ML) imputation

This is one of the contemporary methods for imputation that has proven to work quite well. Although it still rests on the MAR assumption, practical experience (e.g. "Missing Data" by Paul Allison, 2001) has shown it to be an excellent method under a wide variety of situations. One of the desirable features of this approach is that the estimates for the model parameters are asymptotically unbiased. What that implies is that except for variations due to small sample size, the model specifications (e.g. ß values in a regression model) can generally be trusted except for sampling error. These properties hold when the data are MCAR, MAR, and even NMAR provided you can specify the mechanism ("cause") for patterns in missingness. This is a big advantage compared to regression approaches to missing value imputation that usually lead to underestimating the variance.

These desirable properties for maximum likelihood imputation do not come "free", though: the analyst in return must specify the joint distributions between (all) variables that have missing values in a parametric model. For a few decades now, the Tilburg Faculty of Social Sciences has been providing the LEM freeware that can be leveraged for this approach (that you can download here on Prof. Dr. Jeroen Vermunt's website). Several commercial software packages are available too, like AMOS, EQS, M-PLUS, or MX.

Iterative methods of imputation

Maximum Likelihood imputation is a valuable improvement over regression based imputation. The main downside to maximum likelihood imputation is that it requires (hinges on) specification of the parametric joint probability model for all variables with missing values. The results are also somewhat sensitive to choice of said model. Besides the additional effort, selection of that model can be challenging. A contemporary alternative is multiple imputation (see e.g. Rubin, 1987) which has good statistical properties, too. One could argue *almost* as good as maximum likelihood methods. Like maximum likelihood it works when the data are MCAR as well as MAR, and can (mostly) be made to work for NMAR as well.

There are two main advantages to using iterative methods of imputation:

1. It can be applied to virtually any kind of data or model
2. It can be applied without resorting to packages like LEM or commercial alternatives

One of the main drawbacks is that it does not produce 100% consistent results. I.e. if you rerun analyses on the same dataset, you are bound to get ever so slightly different results. Those variations are the result of the random seeds that will produce comparable but not quite

identical results. Another challenge is that the method is newer, less well known, and hence analysts may be less familiar with it.

Commonly used in the realm of Bayesian data analysis is a method called Monte Carlo Markov Chain (MCMC), an algorithm that –in this context– is an extension of linear regression imputation methods. The main difference is that rather than imputing the $\hat{Y}$, the Monte Carlo procedure is used to determine the empirical distribution of values which avoids the problem of overly optimistic (too low) variance associated with the "ordinary" regression based methods for imputation.

Obviously this approach requires additional computations (to generate empirical distributions), but with much more powerful hardware, and relatively modest requirements with regards to the number of samples that need to be generated, this procedure is usually quite tractable. This approach is embedded, for instance, in SAS PROC MI, but also available in several other commercial products like SPSS or BayesiaLab.

Needless to say, when procedures like these can be "run out of the box", without the need to specify a parametric joint distribution for data columns with missing values, there is little objection to (more or less) "always" applying an intelligent approach to missing data imputation.

Project goals: pursuing a "pragmatic" route forward

To wrap up this topic of missing value imputation, for the purpose of "efficient" analysis, we almost always (more or less) *'have'* to assume the distribution pattern of missing values is random (MCAR or MAR). Reality is, however, that the empirical distribution of missing values *as you find it in the data* is rarely (if ever) truly 100% random. In real-world datasets, the distribution of missing values almost always seems to contain at least *some* bias.

Many statistical operations that we perform on data (like regression), *assume* completeness. Therefore, when you encounter a record with missing values, you need to decide on a "strategy" for dealing with it. In social sciences, a procedure called "listwise deletion" (sometimes also called "casewise deletion") is sometimes employed where every record with *a* (any) missing value gets dropped from analysis. In fact, this approach is so common that it's (almost) always available, and often even the default method for statistical analysis tools. An obvious "problem" associated with listwise deletion, since we know that bias in missing data patterns is so prevalent, is that if you drop those records wholesale then you deliberately (although possibly unconsciously) *introduce* bias into your analytic data set.

Another drawback, besides the risk of inadvertently biasing your findings, is that listwise deletion leaves you with a *smaller* data set. It seems a bit of a waste to dispose of the potentially valuable information that was present in the fields that *were* populated. Not everyone seems to appreciate that although we live in the era of "Big Data", for many practical

challenges we often find ourselves short of data. A smaller data set implies an unequivocal loss in statistical power which is obviously undesirable.

Whether you want to "salvage" data (avoiding listwise deletion), or whether you want to avoid unintended bias, in both cases you will need a mechanism for imputing a value where none was present (but *was* expected). This is a domain where recently many advances have been made, to some extent "owing" to the big data revolution. The simplest approach of replacing all missing values in a column with a constant, typically the Mean or Mode, has serious drawbacks, as previously mentioned. So if at all possible and feasible we want to avoid that.

We don't (realistically) assume all missing values to have the same value, so an obvious side-effect of imputing with a constant –whichever one you choose– is that it will 'artificially' lower the variance for that field, an unfortunate artefact of that method with "risky" statistical consequences. When you choose this option, p-values in your models can no longer be trusted since the way those get calculated rests on assumptions that you are willingly and knowingly violating. Because of this, *usually* the p-values come out deceptively small, i.e. you find effects to be significant when a more accurate and equitable estimation method might have flagged the effect as non-significant. Risky business!

Given project goals and schedule pressure, oftentimes you need to settle for a suboptimal ("pragmatic") shortcut. Your choices will be informed by how many records are missing for a given column, and how many columns are missing for each record, as well as their respective distributions. And obviously your need for precision. Columns or rows that are predominantly missing, *and* that appear in far worse condition than the rest of your data may be less representative for extraneous reasons. But sometimes those sparsely populated columns are essential for the purpose of analysis, and you will muster herculean efforts to salvage them. The context for analysis will drive these considerations. "It depends", so to speak.

When you can bear the loss of statistical power, often listwise deletion is a convenient and straightforward option. After you list the number of missing columns for each record, notoriously "empty" rows are a legitimate candidate for exclusion. There could be lots of reasons why this pattern occurs –that you want to learn about– but eliminating these records form analysis may be justified as well as an efficient way to address a large part of the problem. Similar considerations may apply for sparsely populated columns.

Although less than ideal, if the proportion of missing records is less than, say, 5% imputing with a constant like mean or mode might be defensible if this allows you to keep the remaining records. Obviously, you would prefer to opt for a more "intelligent" method like regression, maximum likelihood, or an iterative approach. Sadly, sometimes schedule pressure may not allow you that option.

SQL "NULL" values

"Missing" in databases is (usually) coded as NULL, and since source data often originate in databases, let's have a look at this very special "value" – one that strictly speaking should be thought of as the *absence* of a value. For a number of "technical" reasons, database systems use (or rather: *need*) so-called NULL values. Several authors (e.g. this excellent 2016 paper by Kenneth Baclawski: "The Meaning of NULL in Databases and Programming Languages") have written great contributions on the merits and challenges of the SQL NULL value.

SQL NULL suggests explicitly there is *no* value known. In many real-world scenarios, however, there is *some* information known. However, the conflicting data points have not been resolved, yet. Some authors have suggested that the information available about *why* a database value is NULL ought to be codified. It seems crude to lump all causes for NULL into a single bucket, when they all get mapped onto the same value. Even Codd already gave this notion thought as he was quite aware of the (many) problems associated with NULL. Codd (also) considered a multi-valued exception class, but it seemed these solutions entered as many *new* problems as they were supposed to solve. All things considered, most professionals seem to agree that a "single" NULL value is the best of all evils. For the purpose of downstream analysis therefore, the SQL NULL almost always needs to be interpreted as either a "does not apply", or, "value unknown" label.

Summary

Missing data come in many shapes and sizes. The good, the bad, and the ugly. Data sets with a problematic amount of missing data, can be curated to some extent. Depending on the objective of analysis, cost and gains analysis, and time pressure, you can make an evaluation how much effort to put into this. An *informed* assessment requires awareness of the possibilities and limitations of various methods for missing data imputation (or deletion of records with missing data).

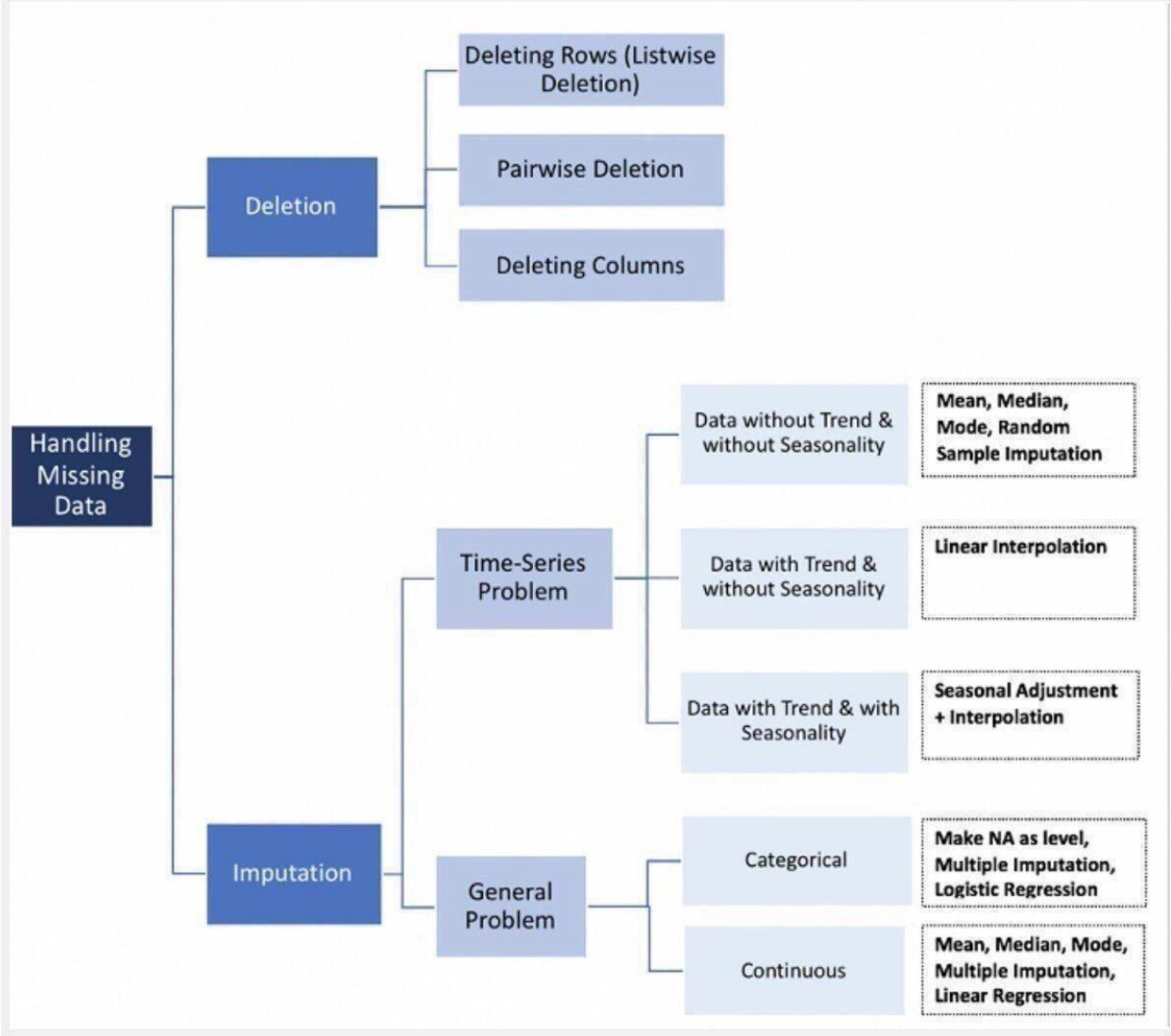
"Quality is Free" Phil Crosby wrote in 1980, and his words seem as true as ever. If you have a chance to *avoid* data quality problems upstream, this is nearly always your best strategy. But for a myriad of reasons, this isn't always feasible. Ad hoc analysis across business silos is often a fruitful strategy to surface gaps in your value stream and hence an opportunity to surface value creating opportunities through analytics. But it isn't until these opportunities have been elucidated, and importantly *quantified*, that you can make an informed assessment whether systems reengineering and probably some change management is worth your while. Therefore ad hoc analyses that offer you a sporting chance of highlighting these pockets of business value will always have their place. And in many (if not all!) of these projects your analytic dataset will be plagued by (at least) some missing data.

In statistics, we use refer to the distinction between "missing at random" (MAR), "missing completely at random" (MCAR), and "not missing at random" (NMAR). I would love to have some MCAR data, one day, but mine never look like that. Until then I usually need to make assumptions that data are MAR, and hope they aren't NMAR. That's the twilight zone I am most interested in, because it's the only consideration that is relevant to me from a practical

perspective. When you know that –due to unmeasured confounding– there is an effect on the variable of interest from that mechanism that caused your patterns in missingness, you need to resort to contemporary (and rather advanced) statistical methods to try and quantify how "bad" the situation might be. These methods like calculating e-values, or quantitative bias analysis (QBA), would merit a white paper of their own. For now, I will leave that to the interested reader to explore himself.

Methods for handling missing data rest on different sets of assumptions: some very strong, and some less stringent. In many cases these assumptions cannot be tested directly *in* the data, and therefore it is even more important that you understand the requirements imposed on the data – they differ across imputation methods. Unless you are familiar with these assumptions, you cannot make a prudent assessment how plausible it is your assumptions will hold, and when and where they might be violated. Along with testing your assumptions, I also recommend some causal analysis, preferably illustrated with causal diagrams (DAG's: directed acyclical graphs) to support your thinking. Robust study design helps avoid missing data, and also prepares imputation strategies for those likely to be encountered.

At the end of the day, missing data can rarely if ever be avoided altogether. For projects where the amount of missing data and their non-randomness becomes a serious threat to validity, it's on the analyst or data scientist to make an informed data assay: can these data be salvaged? Are they fit for purpose, provided the efforts to impute worth your while? The onus is on you to make an *informed* decision about which data to discard, and which data you will repair. Your knowledge of imputation methods will also drive responsible data governance to properly inform all downstream information consumers of limitations and caveats. The efforts to impute can be onerous, but often worth your while, provided you go about it in the right way. As Charles Babbage was quoted saying: "Errors using inadequate data are much less than using no data at all."


Handling Missing Data
Deletion
Deleting Rows (Listwise Deletion)
Pairwise Deletion
Deleting Columns
Imputation
Time-Series Problem
Data without Trend & without Seasonality
Mean, Median, Mode, Random Sample Imputation
Data with Trend & without Seasonality
Linear Interpolation
Data with Trend & with Seasonality
Seasonal Adjustment + Interpolation
General Problem
Categorical
Make NA as level, Multiple Imputation, Logistic Regression
Continuous
Mean, Median, Mode, Multiple Imputation, Linear Regression