Joshua Schaaf

# MACHINE LEARNING IMPUTATION ANALYSIS WITH HIGH DIMENSIONAL BIOLOGICAL DATA: METHYLATION ARRAY

**Abstract:** Multiple imputation methods were tested on a chunk of methylation array data (138 samples, 100 probes). Iterative Bayesian ridge regression imputation from sklearn was the best imputer with the lowest RMSE, MAE, and SSE of all the imputation methods. Additional methods for creating and testing the missing data should be explored in the future.

**Introduction:** A large portion of time spent working on a machine learning project is spent on data preparation. This can be in the form of exploratory data analysis, which may lead to initial feature and sample filtering. Some datasets can be incomplete and riddled with missing data. This missing data poses a problem to the rest of the analysis, including most machine learning algorithms/applications. Many solutions exist to the issue of missing data, such as just removing offending samples, replacing missing values with either the mean, median, or mode of that feature, or even neighbor-based approaches like k-Nearest-Neighbors (kNN). Unfortunately, high dimensional data seems to benefit the least from these solutions. Biological data tends to be feature-rich, providing this unique issue of dealing with missing data in high dimensional datasets. For example, methylation data can have up to 860,000 features, or probes per sample. This is the case for Illumina's EPIC methylation chip.

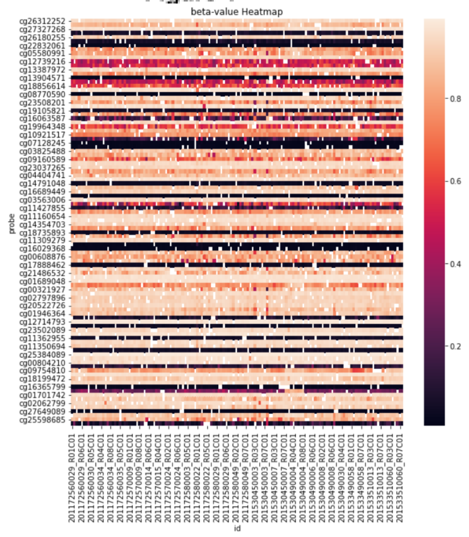
Methylation is a biological phenomenon where the structure of DNA is altered, causing a change in the DNA's function[1]. More specifically, in mammalian genomes, a methyl group is added to cytosine's C5 carbon, turning it to 5-methylcytosine. This causes a change in the physical structure of DNA, changing the rate of transcription if the 5-methylcytosine resides at a transcription factor binding site. Methylation is one of many epigenetic factors to change gene expression in cells. Different genomic methylation patterns are one of the factors for cellular differentiation, and epigenetic clocks based on methylation patterns can predict the biological age of many species of animals, including humans[2]. Unfortunately for scientists, getting high quality methylation data requires just as high quality blood or tissue samples, which can be hard to come by. Buccal cells, or saliva samples, are much easier to acquire, yet saliva samples have been proven to be routinely heterogeneous. This causes problems later in data analysis, because many of the methylation probes are getting differing values, causing their statistical significance to suffer, introducing essentially missing data. This project is focused on achieving high accuracy with imputation with methylation array data found on the Gene Expression Omnibus (GEO)[3] using multiple machine learning methods found in literature. This includes HoloClean AimNet attention-based learning[4], Miss-Forest[5], and iterative learning[6].

**Methodology:** To begin the project, methylation data was required. Methylation data comes in three main forms: 'raw' IDATs, M-values, or beta-values. IDATs are considered the most 'raw' form of beta values, as the Illumina methylation bead-array machines produce these IDAT files as output. IDAT files contain the raw form of signal intensities many times over for each probe, which can be hard to process at times. M-values and beta-values are more commonly used when analyzing methylation array data, as they are more interpretable while still providing the methylation status of some genetic locus. The M-value is calculated as the log2 ratio of the intensities of methylated probe versus unmethylated probe[7]. This provides a value which is close to 0 when the genomic site is neither methylated or unmethylated (partially methylated), with positive values meaning high methylation intensities, and negative values signifying unmethylated regions. M-values have no range limitations. Beta-values on the other hand, are between 0 and 1, with 1 being considered methylated, and 0 being unmethylated. Although M-values may be 'more statistically valid' for analysis than beta-values, beta-values are more easily explainable in most situations[7]. Therefore, this project focused on methylation beta-values.

The Gene Expression Omnibus (GEO)[3] is a great resource for open-source mostly well-documented data. Data uploaded to GEO can sometimes be pre-processed in which the analysts removed poor quality data. For this project, the most ‘raw’ form of beta values were required. This ensures that no analytical step from another researcher impacts the possible imputation accuracy by introducing possible biases in the data. While searching, preference was made towards data collected from saliva samples, as accurate imputation on saliva data would prove more useful than blood or some tissue sample. The dataset used for this project ended up being GSE111631[8]. The data is entirely saliva samples, with 24 samples being from females and 144 being from males. The samples collected were collected for research regarding immune cell contamination in salival, buccal, and cervix cell samples, so there was no experimental group[9]. Additionally, the data on GEO was raw IDATs, allowing for unhindered analysis for this project.

In order to download and convert the raw IDAT files into useful beta-values, the Python package methylprep was used (00-download-and-process.ipynb)[10, 11]. This took a decent amount of time, as the IDAT files are relatively large. Once downloaded, the option to not use beta-values with pOOBAH values greater than 0.05 was used to create the beta-values. pOOBAH values, or p-value with out-of-band array hybridization values are created with every beta-value. The p-value indicates the probability that the beta-value is significant, and was not created by random chance or by noise. This left the data with a large amount of missing beta-values, with many probes being completely filled with missing values (109,310). 309,046 probes had more than 5% missing values. Additionally, there were 30 samples that had more than 25% missing values. These problem probes (>5% missing) and problem samples (>25% missing) were removed from the data, leaving 138 samples and 685,323 probes/features. Unfortunately, some imputation approaches would take a millenia to compute if provided 685,323 features, so to speed things up, the test data was 100 probes (features) and 138 samples.

**Figure 1**



To test imputation approaches, it was decided that 100 randomly sampled probes with no missing values would be used. From these probes, 10% of them were randomly removed and two files were saved, one with all values and one with the missing values added. The missing values are spread relatively evenly amongst the 100 probes. The lowest number of probes missing per probe for the 138 samples is 8 samples, and the highest is 22. A heatmap of these probe values shows distinct lines, indicating low variance for some probes (**Figure 1**). This low variance allows some imputation methods such as mean/median/mode to work relatively well on methylation data.

The first data imputation approach tested was HoloClean AimNet attention-based learning[4]. This approach took the most time to set up and understand, as it uses PostgreSQL and Python to do database creation and revision. The two main portions of HoloClean AimNet are the attention layer and the prediction/probability layer. The attention layer used in AimNet computes attention weights based on context-target attributes, and starts with three inputs: context attribute keys, target attribute bitmask, and the attribute context vectors. These inputs are in an attention-based autoencoder, producing a context vector as output.

HoloClean AimNet attention-based learning is meant to impute data based on an autoencoder model designed to handle mixed data types. This requires HoloClean AimNet to have two networks for prediction, one for continuous values and the other for discrete. For continuous values, the context vectors are first standardized to have a mean of zero with unit variance. Then, a continuous context projection up to dimension k on the standardized data. Finally, a linear layer followed by a non-linear ReLU layer transformation.

The context-target attributes described for the attention-based autoencoder are created depending on what featurizer you chose when running the Python code. Methylation sites in similar regions of the genome can have correlated beta-values. Additionally, as seen in **Figure 1**, many methylation sites have very low variance, consequentially giving low variance beta-values high correlations. Therefore, the only featurizer chosen was the OccurAttrFeaturizer, which creates new features based on original feature co-occurrence. Additionally, HoloClean needs to decide which values need to be 'repaired' using 'detectors'. Because imputation of completely missing values is the goal, the only error detector used was a NullDetector. Additionally, because there are no linked beta-value constraints that we know of, no linked feature constraints were added. For attention-based learning, I used a learning rate of 0.001, batch size of 10, weight decay of 0.01, all for 10 epochs (additional arguments can be found in holoclean.ipynb). Despite only being 10 epochs, the learning processes took around 75 minutes with GPU acceleration.

Continuing with neural networks, the next approach was with another autoencoder. However, this autoencoder was created in TensorFlow in an article found online[12]. The autoencoder in the article was repurposed to take continuous data as opposed to categorical data as input. It uses the adam optimizer, dropout, relu activation functions, sigmoid as an output activation function, and a batch size of 256 (02-results.ipynb). The network is made of strictly 'dense' layers followed by a dropout layer, with one final dense layer as output. The error for this network is the mean square error, or sometimes called 'reconstruction error' for autoencoders. The idea is that, when 'learning the structure' of the features, the autoencoder will just fill in the missing values with data that 'makes sense' to the network. 50,000 epochs took around half an hour with GPU acceleration in Python.

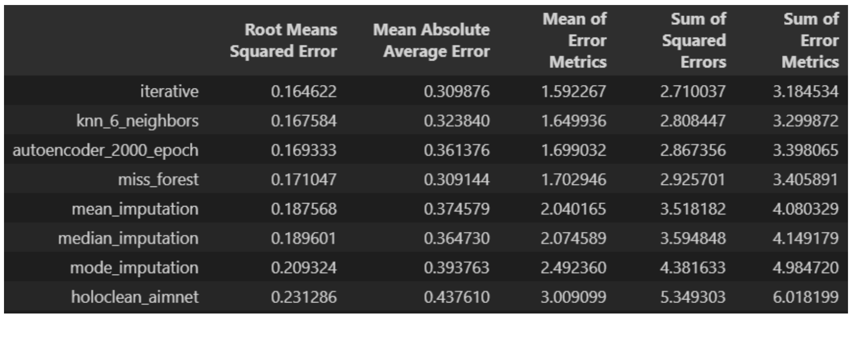
The next imputation approach to the missing data was called missForest[5]. This approach uses a random forest approach for imputing data. First, an initial guess for the missing value is made with some simple imputation method, such as mean imputation. Next, the features are sorted by their number of missing values. Then, for each feature, fit a random forest using the observed values. Once the difference between newly imputed data and the previous imputed increases since the previous iteration for the first time, the training stops. The implementation of missForest used was in missingpy[13], and all the default values were used (02-results.ipynb). It took around 1 minute (5 iterations) for missForest to hit the stopping criterion.

While searching for easily applicable high-accuracy imputers, I came across the IterativeImputer in sklearn, a popular package for data analytics in Python[6, 14]. IterativeImputer is a multivariate imputation method that repetitively models each feature with missing values as a function of other features. Surprisingly, it can be used with any estimator that follows sklearn estimator criteria, so it can actually be used as a missForest imputer with the built-in sklearn RandomForestRegressor. For this instance, the default estimator of Bayesian Ridge Regression was used (02-results.ipynb). The IterativeImputer starts with designating a feature column as output and another feature column as inputs. The regressor of choice (in this case, Bayesian Ridge) is fit on those columns. Then, the regressor is used to predict the missing values. Once the difference of predicted values between iterations falls below the 'tol' parameter, the iterations stop.

In addition to the previously mentioned imputation methods, mean, median, mode, and kNN imputation methods were tested on the data to provide a 'baseline' accuracy (02-results.ipynb).

**Results:** To decide which imputation method was the best, multiple methods of measuring accuracy were implemented. The sum of the squared errors (SSE), the mean absolute

**Table 1**



| | Root Means Squared Error | Mean Absolute Average Error | Mean of Error Metrics | Sum of Squared Errors | Sum of Error Metrics |
|---|---|---|---|---|---|
| iterative | 0.164622 | 0.309876 | 1.592267 | 2.710037 | 3.184534 |
| knn_6_neighbors | 0.167584 | 0.323840 | 1.649936 | 2.808447 | 3.299872 |
| autoencoder_2000_epoch | 0.169333 | 0.361376 | 1.699032 | 2.867356 | 3.398065 |
| miss_forest | 0.171047 | 0.309144 | 1.702946 | 2.925701 | 3.405891 |
| mean_imputation | 0.187568 | 0.374579 | 2.040165 | 3.518182 | 4.080329 |
| median_imputation | 0.189601 | 0.364730 | 2.074589 | 3.594848 | 4.149179 |
| mode_imputation | 0.209324 | 0.393763 | 2.492360 | 4.381633 | 4.984720 |
| holoclean_aimnet | 0.231286 | 0.437610 | 3.009099 | 5.349303 | 6.018199 |

average error (MAE), and the root mean squared error (RMSE) were all found between the imputed data and their actual values. The sum and mean of all three 'accuracies' (SSE, MAE, and RMSE) were also found to try and find the 'best' overall imputation method.

**Figure 2**

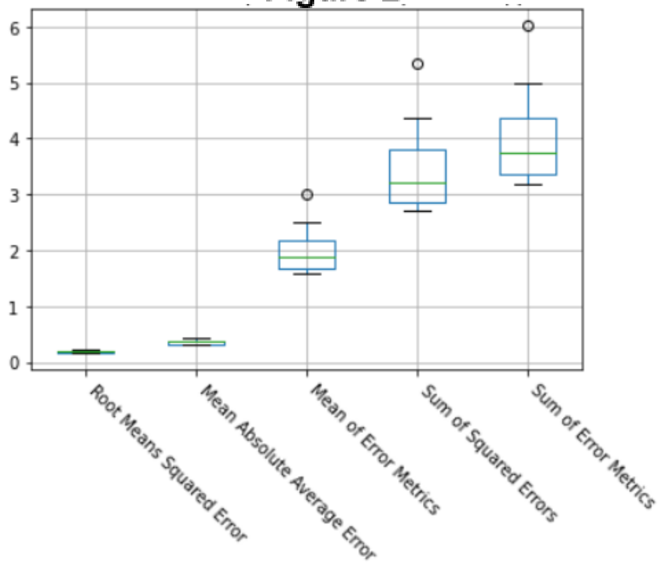
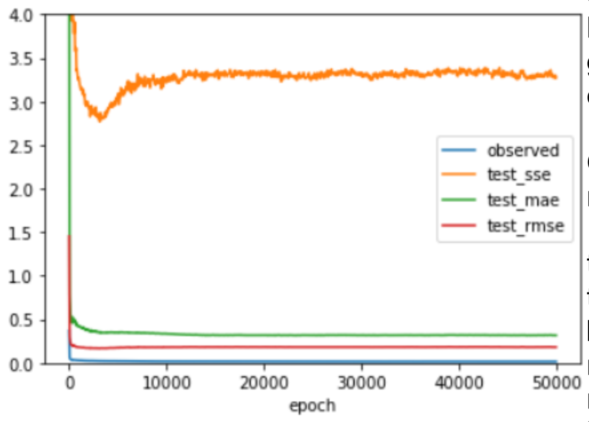


**Table 1** and **Figure 4** shows these accuracies calculated. The index (rows) were sorted ascending by the sum of all the error metrics (SSE, MAE, and RMSE). This means that the Iterative Bayesian ridge regression imputation from sklearn was the best imputer, and the worst imputation method was HoloClean AimNet with its attention-based autoencoder.

As seen in **Figure 2**, the spread of these metrics (RMSE, MAE, and SSE) had similar distributions, yet their ranges varied from 0.066 (RMSE) to 2.639 (SSE).

The Tensorflow autoencoder was run for 50,000 epochs to see if there was any overfitting[15]. **Figure 3** shows the observed (training) MAE, test SSE, test MAE, and test RMSE. Around 2000 epochs the SSE has a minimum, and then the autoencoder begins to overfit. To get the best model, the autoencoder was stopped at 2000 epochs.

**Figure 3**

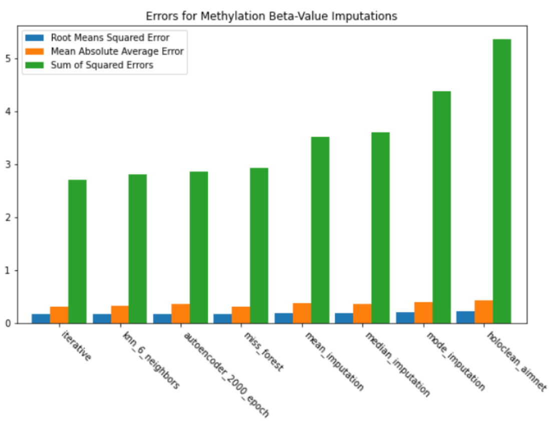


**Conclusion:** Multiple imputation methods of a small chunk of methylation data (132 samples, 100 probes) were tested. Iterative Bayesian ridge regression imputation from sklearn was the best imputer with the lowest RMSE, MAE, and SSE of all the imputation methods[6, 14]. Bayesian ridge regression is a linear model similar to that of ridge regression, but the regularization parameters are estimated by maximizing the log marginal likelihood. It is possible that this was the best imputation method because the selected probes had linear relationships, and the other methods were possibly too nonspecific to get high enough accuracy.

Unfortunately, the worst imputation method was HoloClean AimNet with its attention-based autoencoder, despite taking the longest to train by almost an hour[4]. The slow train time and low accuracy could be the case due to the large amount of engineered features from the co-occurrence 'featurizing'.

In the future, systematically missing values should be imputed in addition to randomly missing to find the robustness of each imputation method. Additionally, multiple percentages of missing values should be found, instead of just 10%. Finally, using the paired Wilcoxon test for error rates between methods may be better than the accuracies measured here.

**Figure 4**



I followed the schedule I put in the proposal for this project. I believe I spent ~30 hours in total, and the part that took the longest was figuring out how to get HoloClean and PostgreSQL to work. The shortest was getting the accuracy of each method.

Joshua Schaaf

## Works Cited

1. Moore, L. D., Le, T., & Fan, G. (2012). DNA methylation and its basic function. *Neuropsychopharmacology*, *38*(1), 23–38. https://doi.org/10.1038/npp.2012.112

2. Wikimedia Foundation. (2021, December 8). *Epigenetic clock*. Wikipedia. Retrieved December 14, 2021, from https://en.wikipedia.org/wiki/Epigenetic_clock.

3. Barrett T, Wilhite SE, Ledoux P, Evangelista C, Kim IF, Tomashevsky M, Marshall KA, Phillippy KH, Sherman PM, Holko M, Yefanov A, Lee H, Zhang N, Robertson CL, Serova N, Davis S, Soboleva A. NCBI GEO: archive for functional genomics data sets--update. Nucleic Acids Res. 2013 Jan;41(Database issue):D991-5.

4. Wu, R. et al. ATTENTION-BASED LEARNING FOR MISSING DATA IMPUTATION IN HOLOCLEAN, https://cs.uwaterloo.ca/~ilyas/papers/WuMLSys2020.pdf

5. Stekhoven, D. J., & Buhlmann, P. (2011). Missforest--non-parametric missing value imputation for mixed-type data. *Bioinformatics*, *28*(1), 112–118. https://doi.org/10.1093/bioinformatics/btr597

6. *Sklearn.impute.IterativeImputer*. scikit. (n.d.). Retrieved December 14, 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.impute.IterativeImputer.html.

7. Du, P., Zhang, X., Huang, C.-C., Jafari, N., Kibbe, W. A., Hou, L., & Lin, S. M. (2010). Comparison of beta-value and m-value methods for quantifying methylation levels by microarray analysis. *BMC Bioinformatics*, *11*(1). https://doi.org/10.1186/1471-2105-11-587

8. U.S. National Library of Medicine. (n.d.). *Geo accession viewer*. National Center for Biotechnology Information. Retrieved December 14, 2021, from https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE111631.

9. Zheng, S. C., Webster, A. P., Dong, D., Feber, A., Graham, D. G., Sullivan, R., Jevons, S., Lovat, L. B., Beck, S., Widschwendter, M., & Teschendorff, A. E. (2018). A novel cell-type deconvolution algorithm reveals substantial contamination by immune cells in saliva, buccal and cervix. *Epigenomics*, *10*(7), 925–940. https://doi.org/10.2217/epi-2018-0037

10. *Welcome to Python.org*. Python.org. (n.d.). Retrieved December 14, 2021, from https://www.python.org/.

11. *Methylprep*. PyPI. (n.d.). Retrieved December 14, 2021, from https://pypi.org/project/methylprep/.

12. *Data imputation using autoencoders: What to do when data is missing? (part II)*. Curiousily. (n.d.). Retrieved December 14, 2021, from https://curiousily.com/posts/data-imputation-using-autoencoders/.

13. *Missingpy*. PyPI. (n.d.). Retrieved December 14, 2021, from https://pypi.org/project/missingpy/.

14. *Sklearn*. scikit. (n.d.). Retrieved December 14, 2021, from https://scikit-learn.org/stable/.

15. *Tensorflow*. TensorFlow. (n.d.). Retrieved December 14, 2021, from https://www.tensorflow.org/.