Hierarchical Model Transfer Methods for Ensemble Learning with Large Amounts of Missing Data

Adam Catto
Capstone Project, M.S. Data Science, CUNY Graduate Center
Advisor: Professor Anita Raja

Fall 2021

1 Introduction

Clinical data are typically very high-dimensional, yet often quite sparse; any given feature may only be available for a relatively small subset of the population. When building prediction models on these data – for instance, in the context of diagnosis, risk stratification, or survival analysis – we must figure out how to handle the missing data. Determining optimal strategies for handling missing data is facilitated by understanding the missingness patterns in the dataset. A popular framework for characterizing missing values is the Rubin's trichotomy [LR19] of "Missing Completely at Random (MCAR)", "Missing at Random (MAR)", and "Missing not at Random (MNAR)".

One method for handling missing values is to simply omit (delete) certain sets of data. This can be deletion of samples with missing values (list-wise deletion), deletion of certain features with large amounts of missingness (test-wise deletion), or a combination of the two. However, list-wise deletion of data that is not MCAR can introduce bias – in this case, the post-deletion sample will be different from (and hence not representative of) the original sample; if the original sample approximated the population's statistics, then the post-deletion sample will not approximate the population's statistics as well as the original sample would have, and in the context of machine learning we would see lower performance on the testing data (i.e. higher bias). In cases where data are MCAR, this may not introduce bias, but it reduces the statistical power. In high-dimensional datasets where many samples are missing at least one value, this can lead to drastic decreases in performance. Another deletion strategy is *pairwise deletion*, which deletes samples only if they are missing values in features that are used in some analysis; if the analysis does not use those features, it is included. This can circumvent some of the issues brought about by list-wise deletion, but it creates different sample sizes across analyses.

Another common strategy with data that is MAR or MCAR is to impute the missing values, i.e. use some mechanism for estimating what a given missing value ought to be, based on either prior knowledge, statistical properties of the feature in question, statistical properties of the relationship between the feature and the other observed features, or some combination thereof. Imputation may be thought of as prediction of the values of certain samples' features, an intermediate step in the final downstream prediction tasks such as classification or regression, or in unsupervised tasks such as clustering. Many sophisticated imputation methods exist, but they run into potential issues of bias, similarly to list-wise deletion. When large amounts of values are missing in high-dimensional data, this can significantly corrupt the distribution. For certain tasks where each of the individual features is very important to the task, such as gene expression profiling , imputing the missing values is unavoidable, but for others where certain features are not absolutely critical, imputation may not be the ideal way of handling missing data.

These problems of distribution corruption and reduction in statistical power posed by high-dimensional, high-missingness datasets are pronounced in clinical / biomedical machine learning tasks. A distribution that is less representative of the population can lead to systematic misclassifications, perhaps skewed towards one class or another, thus negatively affecting prediction metrics. Additionally, utilizing less samples during training can lead to the model's inability to identify correct patterns, thus also misrepresenting the underlying distribution. Given these problems, a major goal for prediction with missing data is overcoming the issues of adding bias posed by imputation on one hand, and the dramatic reduction in statistical power posed by deletion on the other hand. For this purpose, I propose a framework for dealing with missing data that does not require any imputation, and learns from the joint

distribution of each combination of features, aggregating what is learnt over each combination. This avoids (1) the introduction of bias from imputation on any missingness pattern, (2) the introduction of bias from list-wise deletion on MNAR data, and (3) the reduction in statistical power from list-wise deletion on any missingness pattern. In machine learning parlance, I develop a transfer learning algorithm that is essentially a meta-ensemble; it builds a directed acyclic graph (DAG) – that is equivalent to a partially ordered set with subset ordering – where the nodes are each combination of features and the directed arrows are from a node n to each node which has all the features present at n plus one more feature, then trains an ensemble on every combination of features, masking out the features not in the respective combination and deleting the samples at each node which are missing values at any of the relevant features. This is essentially pairwise deletion at each node. Then at each node's trained ensemble, classifiers are sampled from its ancestors' ensembles, thus "inheriting" the statistical power learned on larger sample sizes with less features.

This thesis is structured as follows: in the Background section, I review the details of some methods for handling missing data and machine learning ensemble methods. In the Methods section, I provide a detailed account of the proposed algorithm. In the Experimental Design section, I describe the experiments run to test the validity of my method on synthetic and real-world high-missingness biomedical data. In the Results section, I report the performance of the method in comparison with other state-of-the-art ensemble methods and methods for handling missing data. In the Discussion section, I interpret and discuss the implications of the results, from both a statistical perspective and a clinical decision-making perspective. In the Works in Progress and Future Directions section, I outline a roadmap of next steps.

2 Background

2.1 Data Imputation

A straightforward method of imputing missing values is to use column-wise mean imputation, or other central-tendency statistics such as median. In this case, the mean of the non-missing values of each column in the dataset is calculated, and any missing value in a column c is imputed to the mean of c. However, this approach can severely perturb the distribution of the data, for instance by decreasing variance. Furthermore, if a feature can be modeled as a function of other variables, this dependence is washed away and the conditional / joint distributions become corrupted, in addition to an increase in estimation error as compared to, say, parametric estimation based on the values of other features.

Due to this issue with mean imputation, we may want to approach the problem as a prediction problem using other features. Some options for imputing continuous features are regression and k-nearest neighbors: one can train a (single-variable, multivariate) regression model with the column-to-impute as the target column, and the samples which have the feature present as the training set; the model can be used to predict the column values for those samples which are missing it. For categorical variables, one may consider a similarly simple classifier, such as logistic regression. One particularly useful model for imputing missing tabular data is the MissForest algorithm [SB12], which can simultaneously impute continuous and categorical variables. MissForest tends to outperform k-nearest neighbors and regression imputation models on synthetic imputation tasks (where ground-truth is known and therefore mean error can be calculated), but similarly to other methods, it still has the potential to change the distribution of the data, especially if the other features are not strongly predictive of that feature. If large amounts of samples are missing a given feature, imputing them will cause a significant portion of the data to be synthetic, which may introduce its own set of problems.

2.2 Data Omission

Given the potential drawbacks of imputing missing values under certain circumstances, a natural alternative is to simply delete portions of the data with missing values. This can be done by removing samples with missing values (list-wise deletion), features with missing values (test-wise deletion), or a combination of the two (pairwise deletion) If there are no systematic patterns in the missingness (i.e. the data are missing at least at random), then removing samples which have missing values should not meaningfully change the distribution. However, this process of list-wise deletion can significantly reduce the statistical power of the model if enough samples are missing or if the original dataset is relatively small – in medical datasets and other high-dimensional datasets, samples which do not have any missing values are rare. In these cases, it may make more sense to remove certain features that have

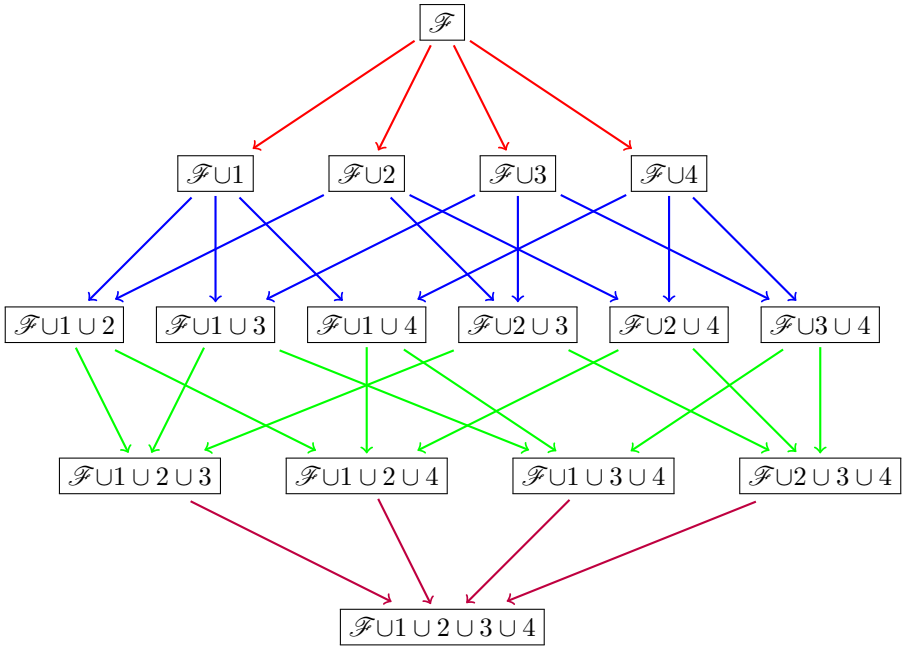

Figure 1: DAG of Ensembles: Illustration

large amounts of missingness. Likewise, if there are only certain features which are necessary for an analysis, it is perhaps a good idea to use pairwise deletion.

3 Methods

In order to mitigate bias brought about by imputation and reduction in statistical power brought about by deletion methods, I propose a method for learning from joint distributions of each combination of features, starting from single-variable distributions all the way up to the joint distribution of all features in the dataset. It is important to note that in the context of medical testing, each test may result in multiple features, thus a sample having missing values in one of the features indicates the rest of the features will be missing at that sample as well. As such, we modify the previous statement from "joint distributions of combinations of features" to "joint distributions of combinations of tests", where all the features associated with a test are included wherever the test is included. The method is sketched as follows. Suppose we are given a set of base features $\mathscr{F}$ and a set of tests enumerated as $\tau = \{1, 2, \cdots, n\}$. First, construct a DAG with nodes representing combinations of tests, and edges representing "immediate inclusion", i.e. given a node ν with tests $\mathscr{F} \wedge 1 \wedge 2 \wedge \cdots \wedge k$, draw an arrow from ν to every node ν^* such that $\nu^* = \nu \wedge t^*$ for any $t \in \tau$ and $t \notin \nu$. A sample illustration on 4 tests is provided in Figure 1.

Each node contains a unique set of features. At each node, the features not present at that node are masked out and pairwise deletion is performed: all samples which have at least one missing value in the relevant feature set are removed from the training set at that node. At each node, an ensemble classifier is learned on the unmasked data that passed the pairwise deletion filter. The classifiers learned at each node are stored in a node attribute denoted "classifiers", and the entire model is stored in a node attribute denoted "model". This yields a stratified, partially-ordered set of ensembles (henceforth denoted "stratified meta-ensemble"), with $n+1$ levels for n tests (n levels for tests and one level for base features). The number of classifiers learned at each node is inversely proportional to the number of valid training samples at that node: in other words, given a training set of k_β samples with each of the base features and η_β classifiers learned at the base feature node, for a given node ν with k_ν samples, the ensemble μ_ν will contain $\frac{k_\nu}{k_\beta}$ classifiers (rounded to the nearest whole number). More details on the algorithms for generating the inheritance graph and building the ensembles can be found in Algorithms 1 and 2.

The benefit of this approach is that on one hand, the model at any given node has been learnt using only the information that is fully available using that node's feature set, thus not perturbing the joint distribution as might occur with imputation, and on the other hand a model is learnt at each node, for each possible joint distribution, thus circumventing the statistical power issue posed by list-wise and pairwise deletion – this approach represents an

Algorithm 1 Build Inheritance DAG

```
procedure BuildInheritanceGraph(tests, base_features, data)
    test_combinations ← powerset(tests)
    levels ← [[x for x in test_combinations if len(x) == L] for L in len(tests)]
    poset ← DirectedGraph()
    poset.addNode(level=0, features=base_features, estimators=list(), predictions=dict(),
                  errors=list(), ground_truth=dict())
    for i, level in enumerate(levels) do
        for j, testCombination in level do
            currentFeatures = list()
            currentFeatures += base_features
            for t in testCombination do
                currentFeatures += t.getTestFeatures()
            end for
            currentLevel = i + 1
            currentNode = Node(level=currentLevel, features=currentFeatures, estimators=list(),
                               predictions=dict(), errors=list(), ground_truth=dict())
            poset.addNode(currentNode)
            nodesOneLevelUp = [node for node in poset.nodes if node.level == i]
            for node in nodesOneLevelUp do
                if all node.features in currentNode.features then
                    poset.addEdge(node, currentNode)
                end if
            end for
        end for
    end for
    return poset
end procedure
```

Algorithm 2 Build Stratified Meta-Ensemble

```
procedure BuildStratifiedMetaEnsemble(trainDataset, inheritanceGraph, ensembleType, nEstimBase)
    totalSamples ← len(trainDataset)
    for node in inheritanceGraph.nodes do
        samples ← [sample for sample in trainDataset if each feature in node.features is present]
        model ← initialize ensembleType(n_estim = round(len(samples) / totalSamples))
        model.fit(trainDataset)
        inheritanceGraph.nodes[node].estimators ← model.estimators
        inheritanceGraph.nodes[node].model ← model
    end for
    return inheritanceGraph
end procedure
```

ordering of all possible pairwise deletions. The real power of this approach, however, is not just learning on all possible pairwise deletions and stratifying predictions via this ordering, but from actually propagating the information learned from each pairwise deletion to each of its "descendants" on the graph, i.e. each node "inherits" information from its ancestors.

Once this graph is constructed and the models are trained at each node, it is time to propagate what has been learned from smaller feature sets / larger sample sets to larger feature sets / smaller sample sets. There are two primary methods I've designed for this: inheriting *all* estimators from *all* ancestors, and randomly sampling a certain number of estimators from a node's ancestors – in this paper, given n estimators at a given node, randomly sample n estimators total from a compiled set of all estimators from all ancestors.

4 Experimental Design

To test the validity of my method, I use both synthetic and real-world biomedical datasets with large amounts of missingness. I compare the predictive performance of off-the-shelf, state-of-the-art classifiers commonly used on these kinds of data with my method. Specifically, I use the real-world Wisconsin Breast Cancer Prognosis dataset [WSM94] with a synthetic missingness pattern applied to it, in addition to the NuMom2b dataset [Haa+15], an observational study of nulliparous mothers-to-be which I used to predict which patients would go on to develop hypertensive disorders.

4.1 Synthetic Missingness

The first dataset that I used was the Wisconsin Breast Cancer Prognosis dataset. This dataset contains 194 samples with no missing values over 34 features. The features are characterized over 10 different sorts: (a) radius (mean of distances from center to points on the perimeter) (b) texture (standard deviation of gray-scale values) (c) perimeter (d) area ((e) smoothness (local variation in radius lengths) (f) compactness (perimeter$\hat{2}$ / area - 1.0) (g) concavity (severity of concave portions of the contour) (h) concave points (number of concave portions of the contour) (i) symmetry and (j) fractal dimension ("coastline approximation" - 1), each with three attributes: (1) mean value, (2) standard error, (3) worst value. Additionally, we have features for recurrence time, tumor size and lymph node status. This is a binary classification problem: there are two labels to predict.

To proceed with the experiment, I selected a set of "tests", i.e. groups of features to partly mask out at various rates, and a set of base features to remain fully-observed. I chose smoothness, compactness, concavity, symmetry, tumor size, and lymph node status as the "tests", and all the others as the base features. I used a 70/30 train/test split, first training an XGBoost classifier [CG16] with 100 boosting rounds on the unmasked data, to obtain part of a baseline for comparison. I then randomly masked 10% of the smoothness values, 20% of the compactness, concavity, and tumor size values, and 30% of the symmetry values, and 50% of the lymph node status values. Note that the mean, standard error, and worst values for each test type were masked simultaneously. Three baseline models were tested on the synthetic missingness dataset for comparison: an XGBoost model with no imputation, an XGBoost model with K-nearest neighbor imputation, and an XGBoost model with MICE imputation [VG11; Buc60]. Next, the meta-ensemble I've proposed was trained and tested using three inheritance mechanisms: full inheritance, no inheritance (just node assignment), and at each node with n boosting rounds a random sampling of n boosted trees from the entire set of ancestors (each model therefore has $2n$ estimators respectively). There were a total of 6 tests, and correspondingly $2^6 = 64$ gradient boosting classifiers were trained. The resulting inheritance DAG for the synthetic dataset can be found at the top of Figure 2.

Given the small size of the dataset, to evaluate my method I tracked the sensitivity, specificity, positive predictive value (PPV), negative predictive value (NPV), geometric mean between sensitivity and specificity, and area under the receiver-operator characteristic curve (AUROC) over 50 trials with different train/test splits. I compared the full-inheritance method to the off-the-shelf XGBoost method with no imputation. This dataset is quite heavily imbalanced: 148 negative versus 46 positive samples. The train/test splits reflected this: the test sets had similar imbalances.

4.2 Real-World Dataset: NuMom2b

Next, I tested my method on the NuMom2b dataset [Haa+15]. This data comes from a large multi-site observational study of nulliparous pregnant women st, which aimed to evaluate numerous potentially interrelated mechanisms leading to adverse pregnancy outcomes. The study collected information from over 10,000 participants over 4 visits, spanning data on clinical measurements, blood/urine tests, questionnaires, ultrasound exams, and more. For this evaluation, I used eventual development of hypertensive condition (yes/no) as the label to predict. The base features were the L1 features preprocessed according to [Gor+21], and the tests selected were each of the placental analytes tested at the second visit, for a total of 9 tests. Again, this dataset is highly imbalanced: 2760 positive versus 7279 negative. This time, due to an increase in dataset size, instead of running 50 trials, each of the 6 methods (my proposed method with full inheritance, no inheritance, and random sampling inheritance; XGBoost with no imputation, KNN imputation, and MICE imputation) listed in the previous sub-section were evaluated according to the 6 metrics also listed in the previous section. The performance at each node (in terms of mean classification error per node) is visualized for each of the 6 models as well. Additionally, the ROC, precision-recall, and sensitivity-specificity curves are plotted.

5 Results

5.1 Synthetic Missingness Dataset

On the synthetic missingness dataset, averaged over 50 trials, the meta-ensemble algorithm I proposed tended to outperform the off-the-shelf XGBoost without imputation, on sensitivity, negative predictive value, AUROC, and geometric mean between sensitivity and specificity. The difference was most pronounced on sensitivity; when comparing the average of each metric between XGBoost without imputation and the meta-ensemble with full inheritance, we observed an average sensitivity of 35.2% for XGBoost without imputation, vs 53.7% for the meta-ensemble with full inheritance. See Figure 3 for a visualization of a comparison between the two algorithms over 50 trials (different test sets to test for variability in performance as a result of differences in small test sets).

Table 1: Breast Cancer Prognosis Dataset: Classification Comparison: Average Over 50 Trials

Breast Cancer Prognosis Dataset: Classification Comparison: Average Over 50 Trials						
Model	Sensitivity	Specificity	PPV	NPV	G-Mean	AUROC
XGBoost (no imputation)	35.2%	**88.9%**	**48.7%**	82.1%	54.2%	68.4%
Meta-Ensemble Full Inheritance	**53.7%**	78.2%	43.0%	**85.0%**	**64.1%**	**70.0%**

Table 2: Breast Cancer Prognosis Dataset: Classifier Types' Performances

Breast Cancer Prognosis Dataset: Classifier Types' Performances						
Model	Sensitivity	Specificity	PPV	NPV	G-Mean	AUROC
XGBoost (no imputation)	15%	**98%.**	67%	80%	38.8%	81%
XGBoost (KNN imputation)	23%	**98%**	75%	81%.	47.5%	83.9%
XGBoost (MICE imputation)	23%	96%.	60%	81%.	47.0%	80.2%
Meta-Ensemble Full Inheritance	**54%**	96%	**78%**	88%	**71.7%**	**86.5%**
Meta-Ensemble Sample n estimators total	38%	91%	56%	84%	59.2%	77.3%
Meta-Ensemble No Inheritance	44%	84%	57%	**90%**	64.6%	75.3%

5.2 NuMom2b

I have compiled results for the various ensembles evaluated on the NuMom2b dataset. In addition to tabulating the results for the different classifiers, I've also visualized their performances at each of the nodes with placental analyte combinations seen in the testing set. The tabulated results can be seen in Table 3, and the node-wise results can be seen in Figures 4-9. The ROC, Precision-Recall, and Sensitivity-Specificity curves can be seen in Figures 10-12. The stratified meta-ensemble techniques still outperformed the XGBoost models, but the differences were less pronounced on this dataset, because it is a more difficult classification problem overall. Although the gap in performance was not as large, the meta-ensemble methods outperformed the XGBoost methods on every metric.

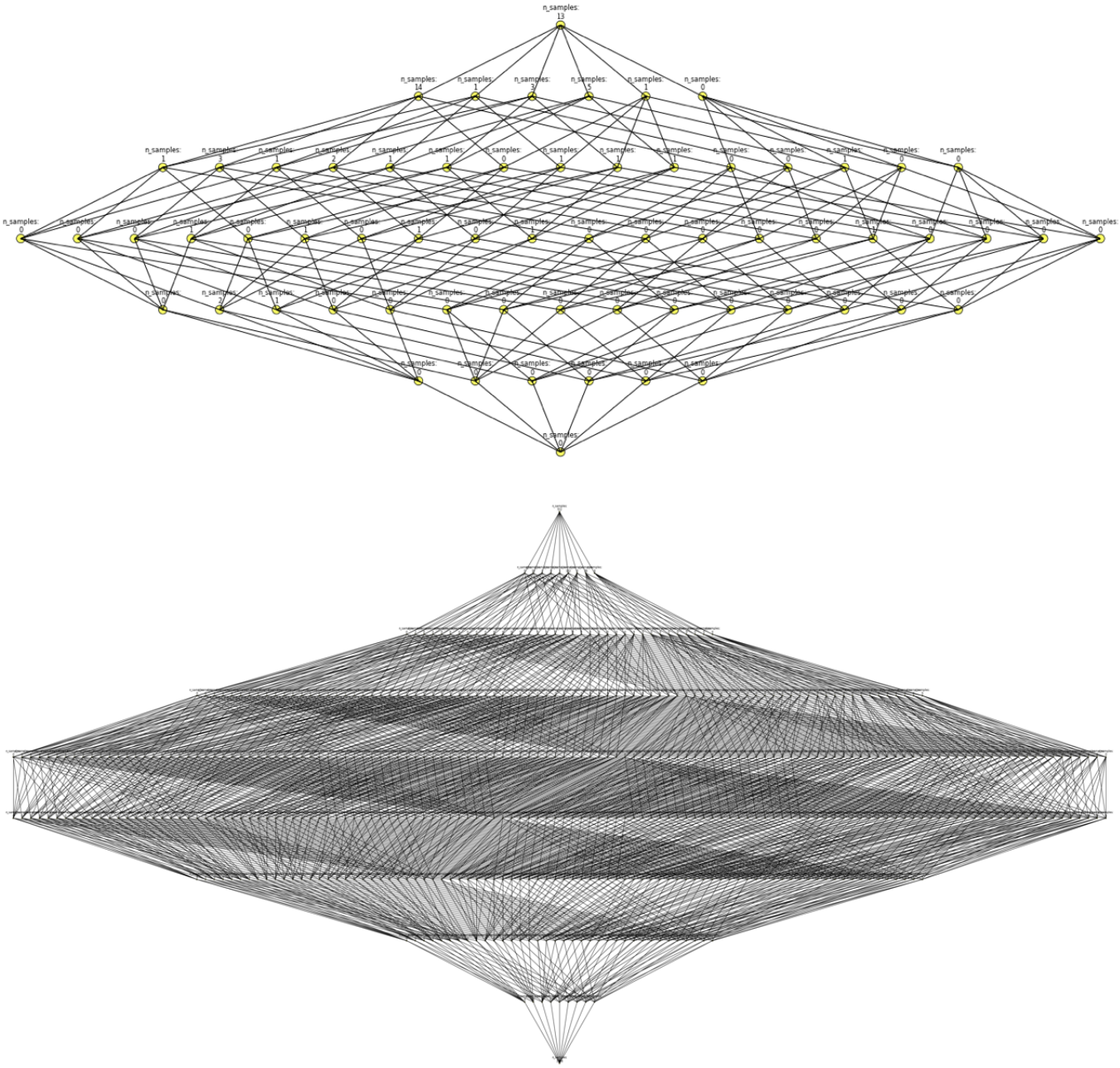

Figure 2: (Top) Synthetic Missingness Inheritance DAG: 64 Ensembles
(Bottom)NuMom2b Inheritance DAG: 512 Ensembles

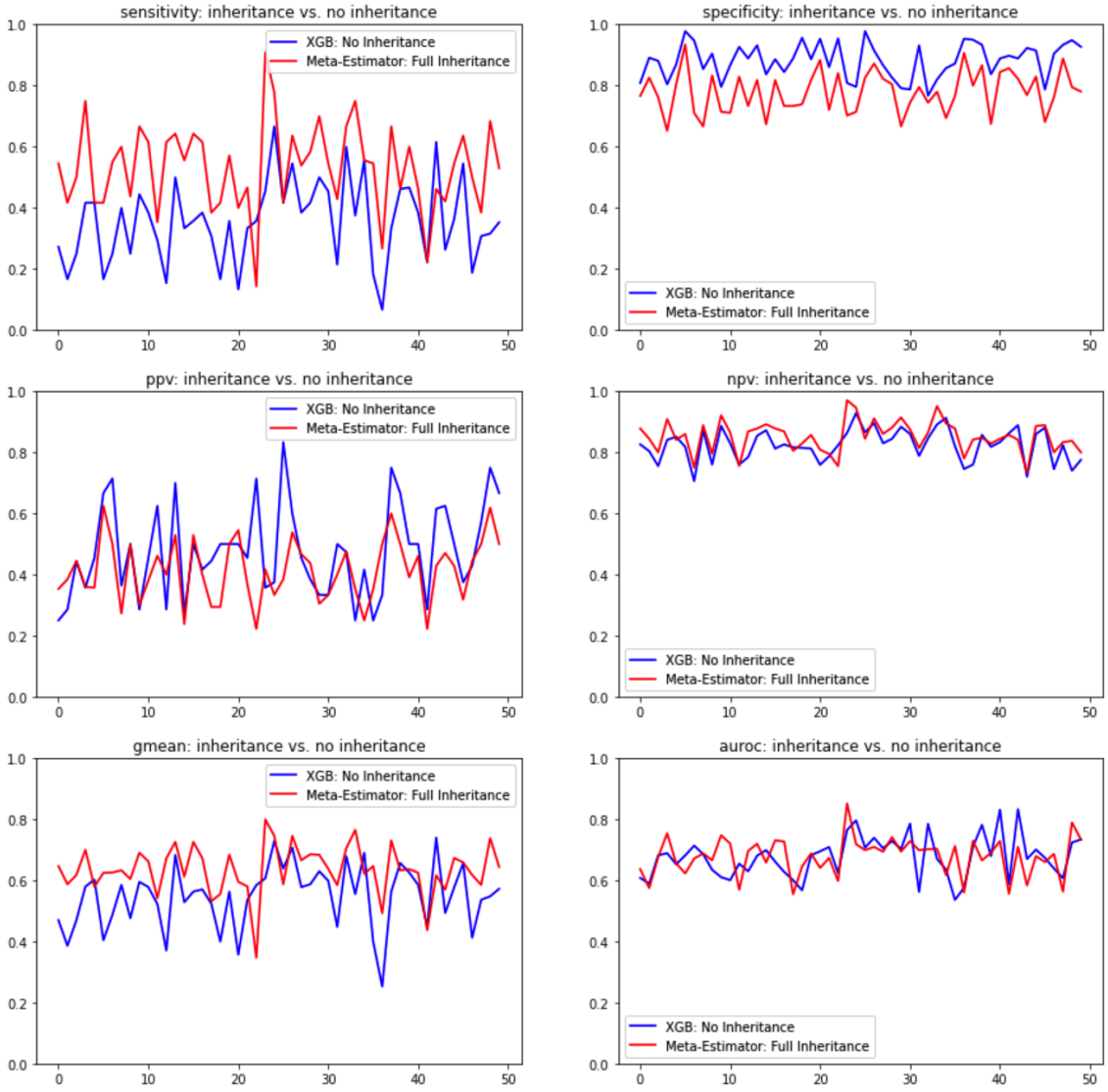

Figure 3: Performance metrics of XGBoost without imputation vs. Stratified Meta-Ensemble with full inheritance over 50 trials

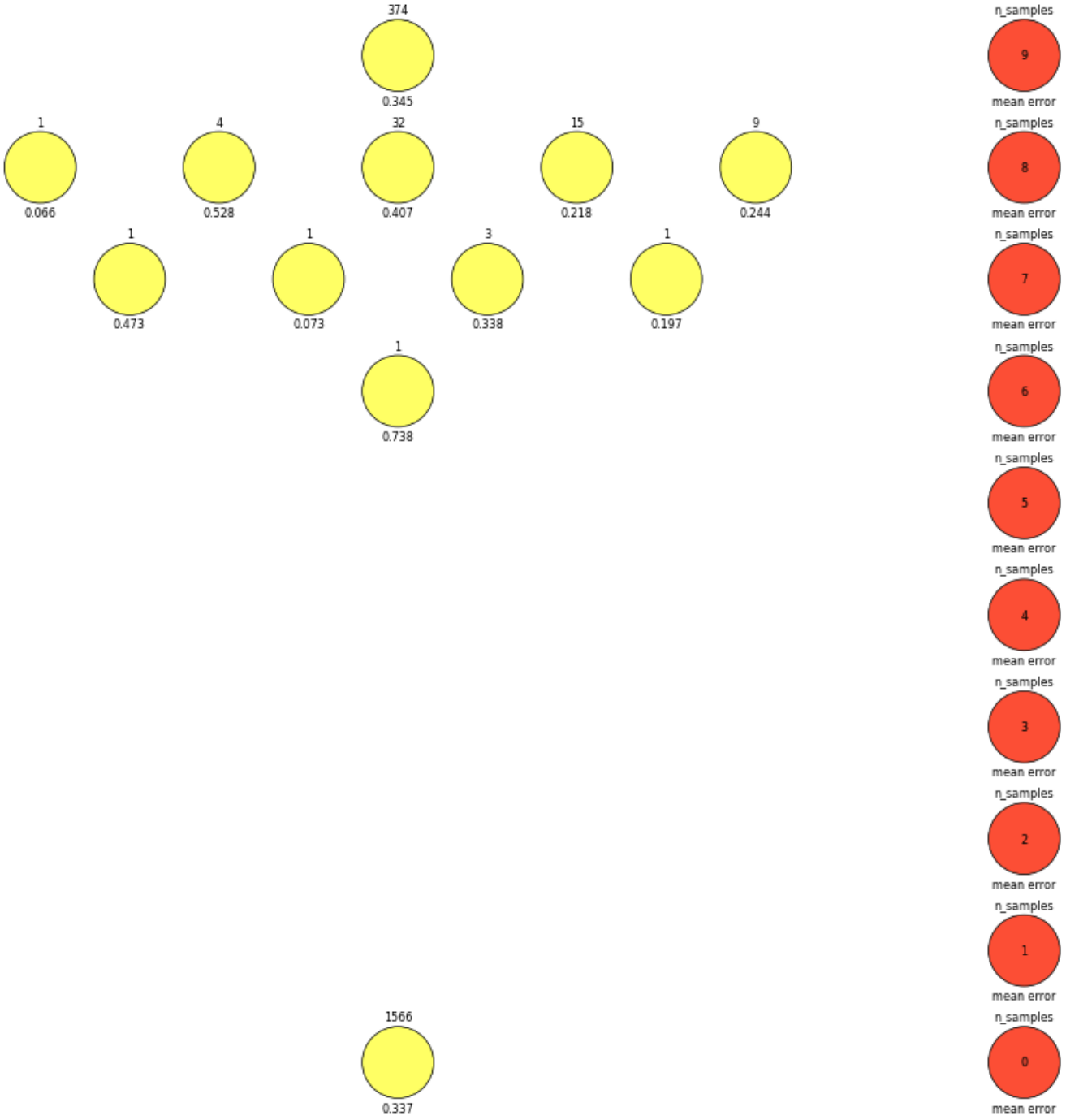

Figure 4: Meta-Ensemble, stratification without inheritance node-wise metrics: number of samples and mean error per node with at least one sample. Levels (corresponding to number of tests) along with text schema are provided by the nodes on the right-hand side.

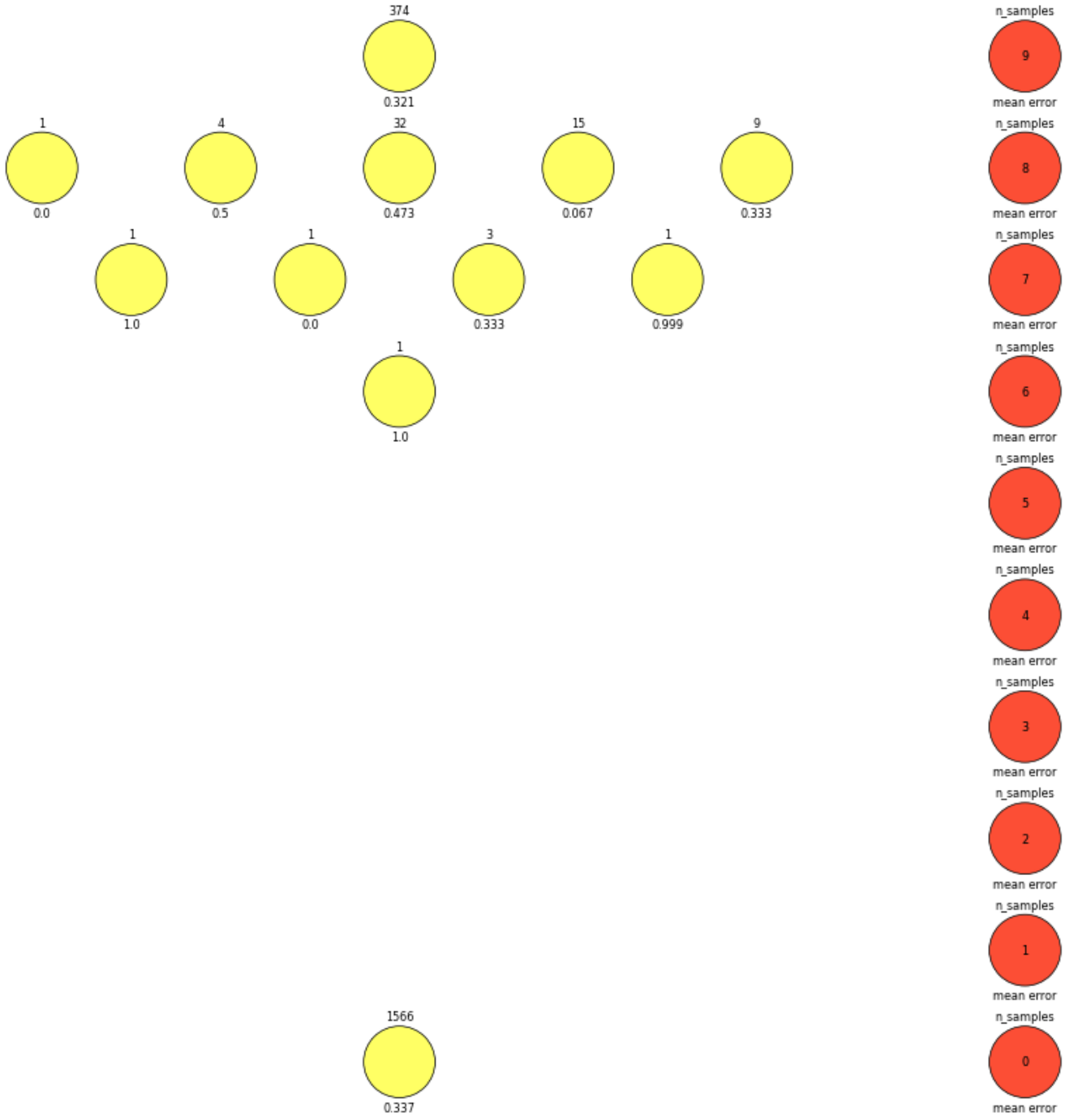

Figure 5: Meta-Ensemble, stratification with full inheritance node-wise metrics: number of samples and mean error per node with at least one sample. Levels (corresponding to number of tests) along with text schema are provided by the nodes on the right-hand side.

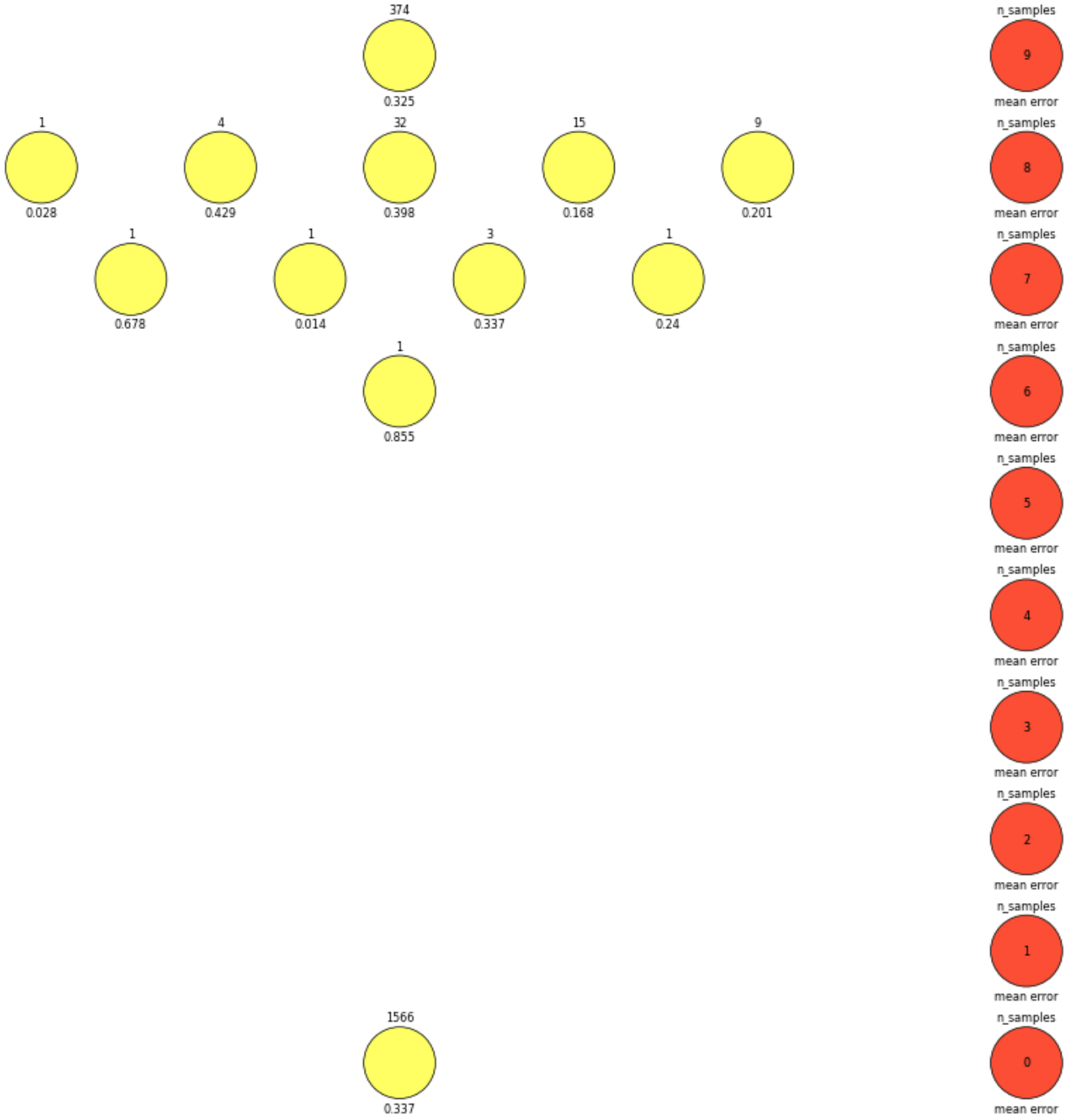

Figure 6: Meta-Ensemble, stratification with inheritance to number of base estimators node-wise metrics: number of samples and mean error per node with at least one sample. Levels (corresponding to number of tests) along with text schema are provided by the nodes on the right-hand side.

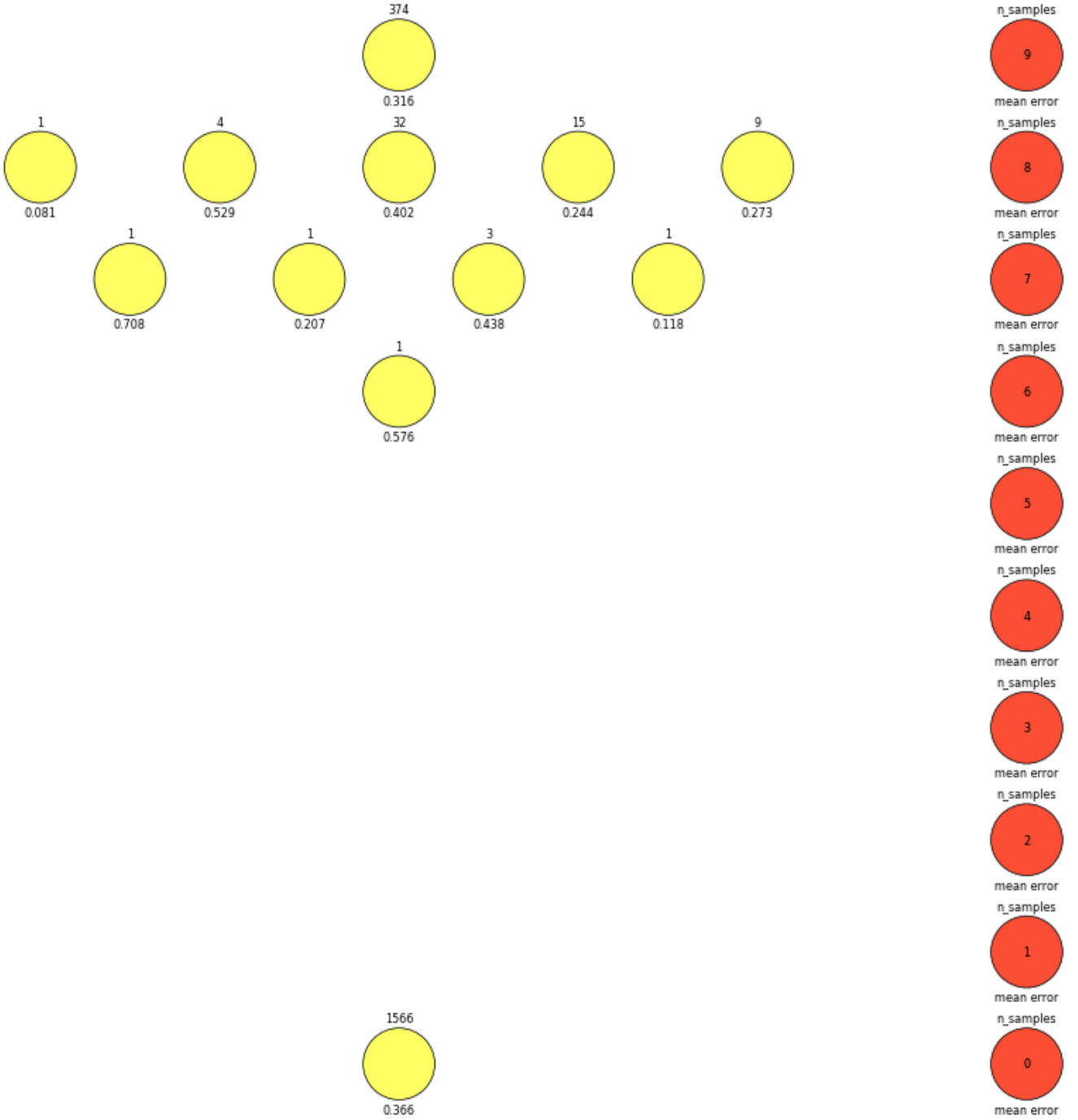

Figure 7: XGBoost, without imputation node-wise metrics: number of samples and mean error per node with at least one sample. Levels (corresponding to number of tests) along with text schema are provided by the nodes on the right-hand side.

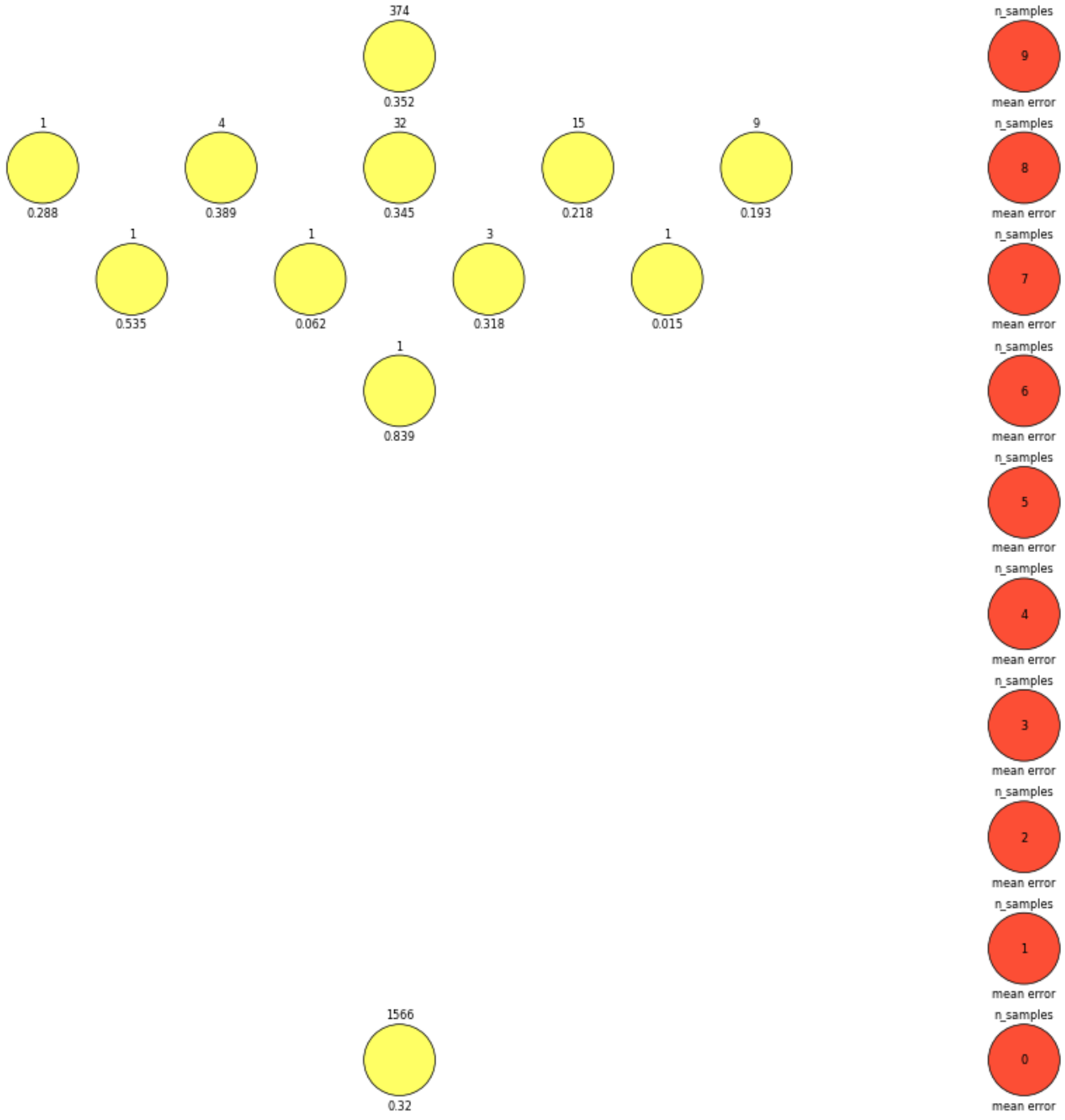

Figure 8: XGBoost, with k-nearest neighbors imputation node-wise metrics: number of samples and mean error per node with at least one sample. Levels (corresponding to number of tests) along with text schema are provided by the nodes on the right-hand side.

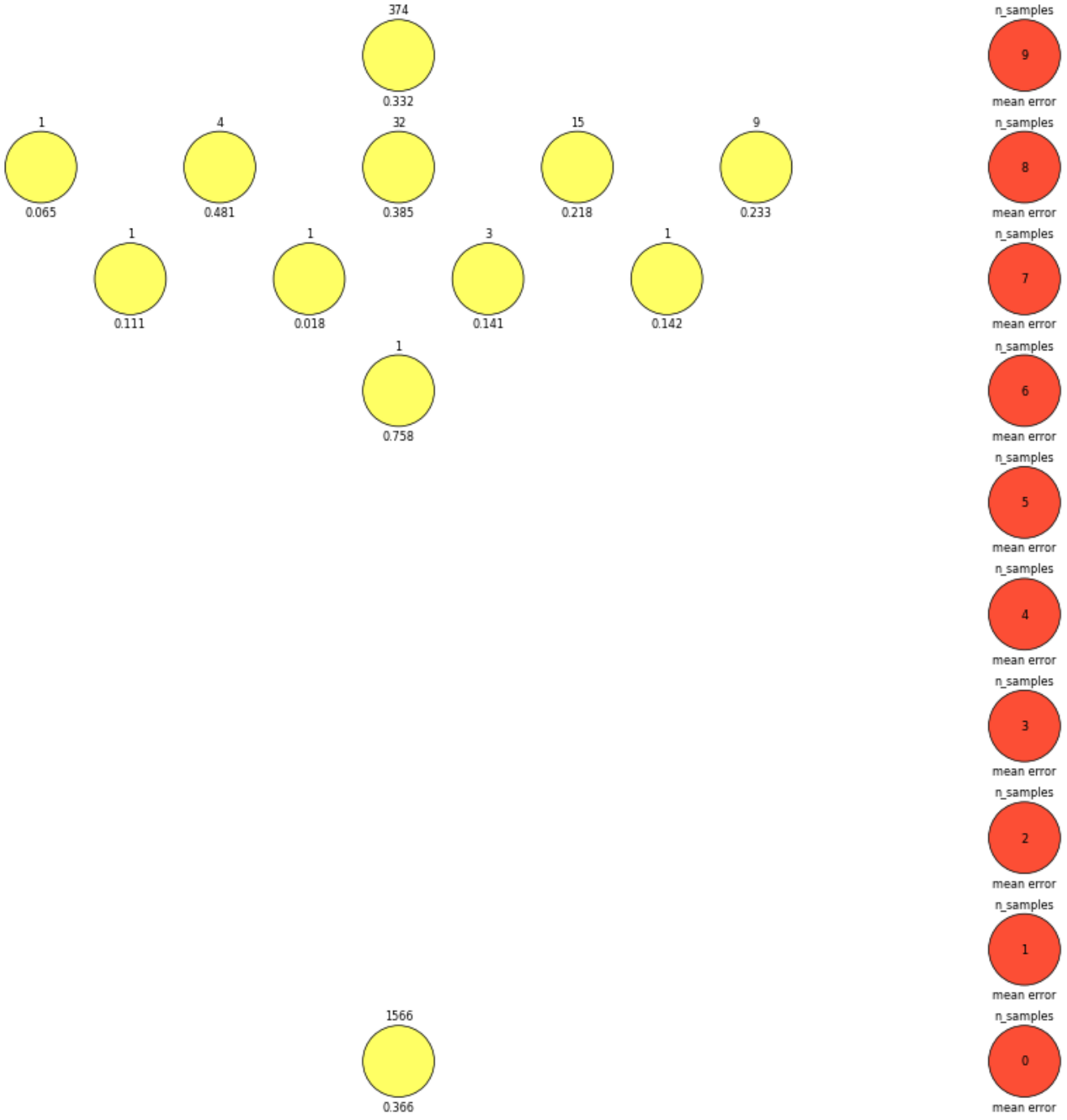

Figure 9: XGBoost, with MICE imputation node-wise metrics: number of samples and mean error per node with at least one sample. Levels (corresponding to number of tests) along with text schema are provided by the nodes on the right-hand side.

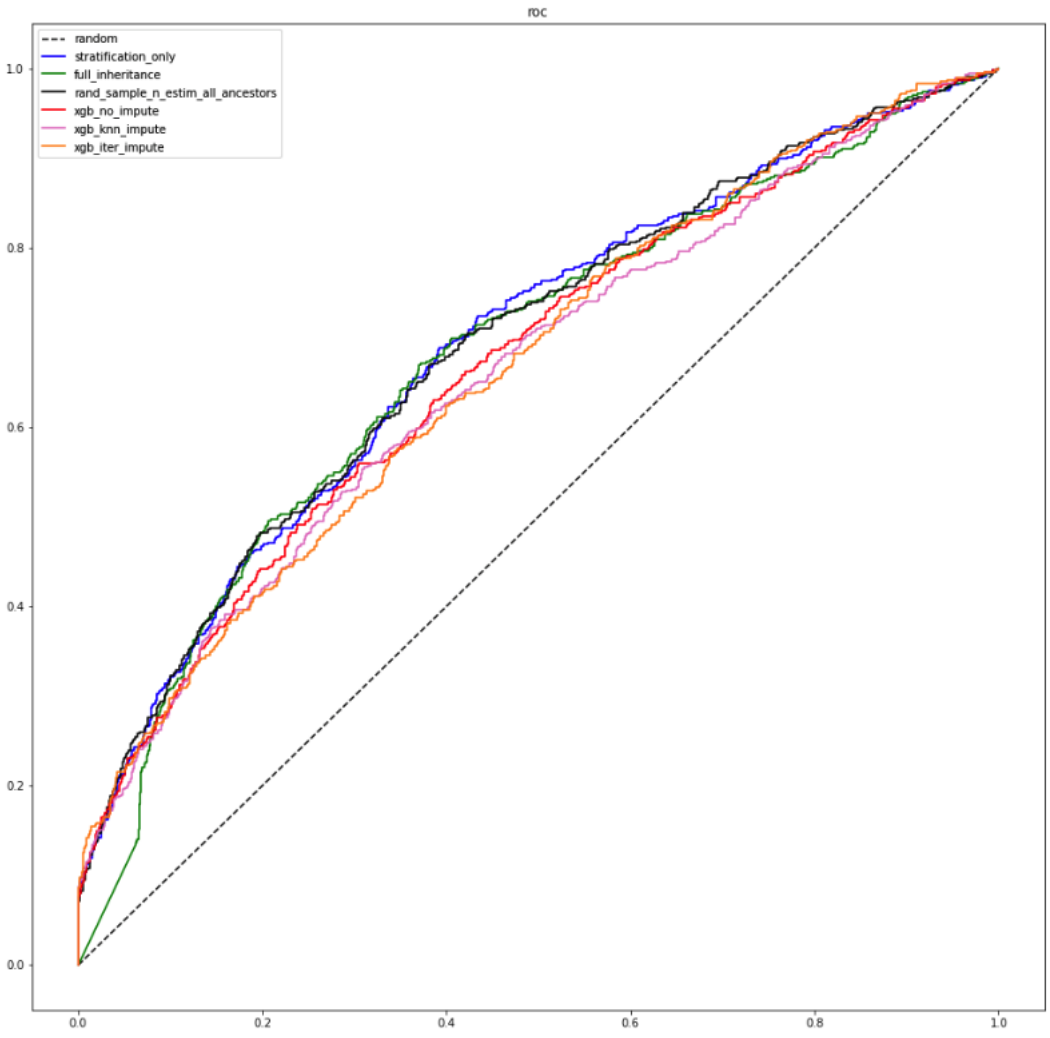

Figure 10: ROC Curve Comparison: NuMom2b Hypertensive vs. Non-Hypertensive

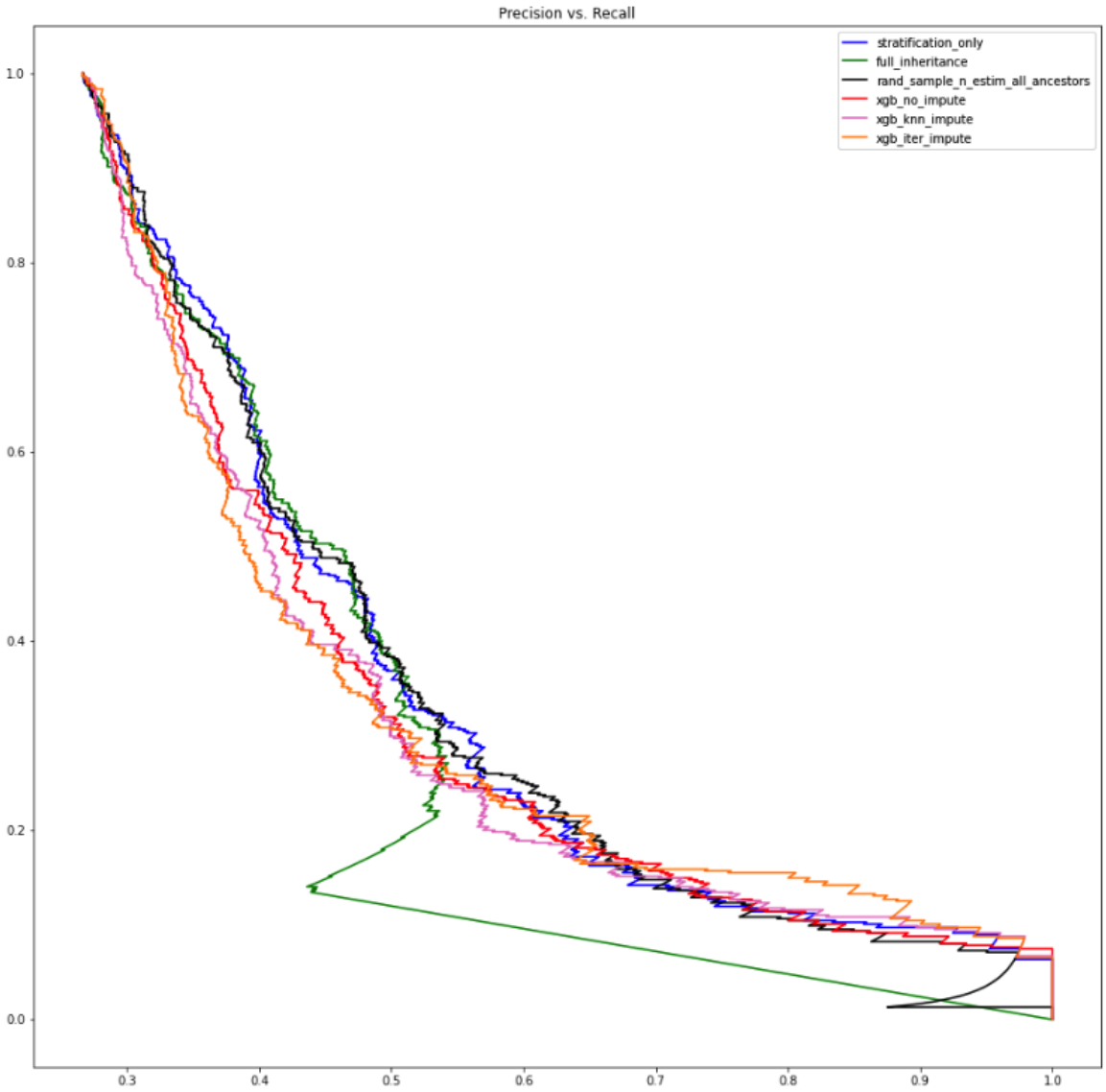

Figure 11: Precision-Recall Curve Comparison: NuMom2b Hypertensive vs. Non-Hypertensive

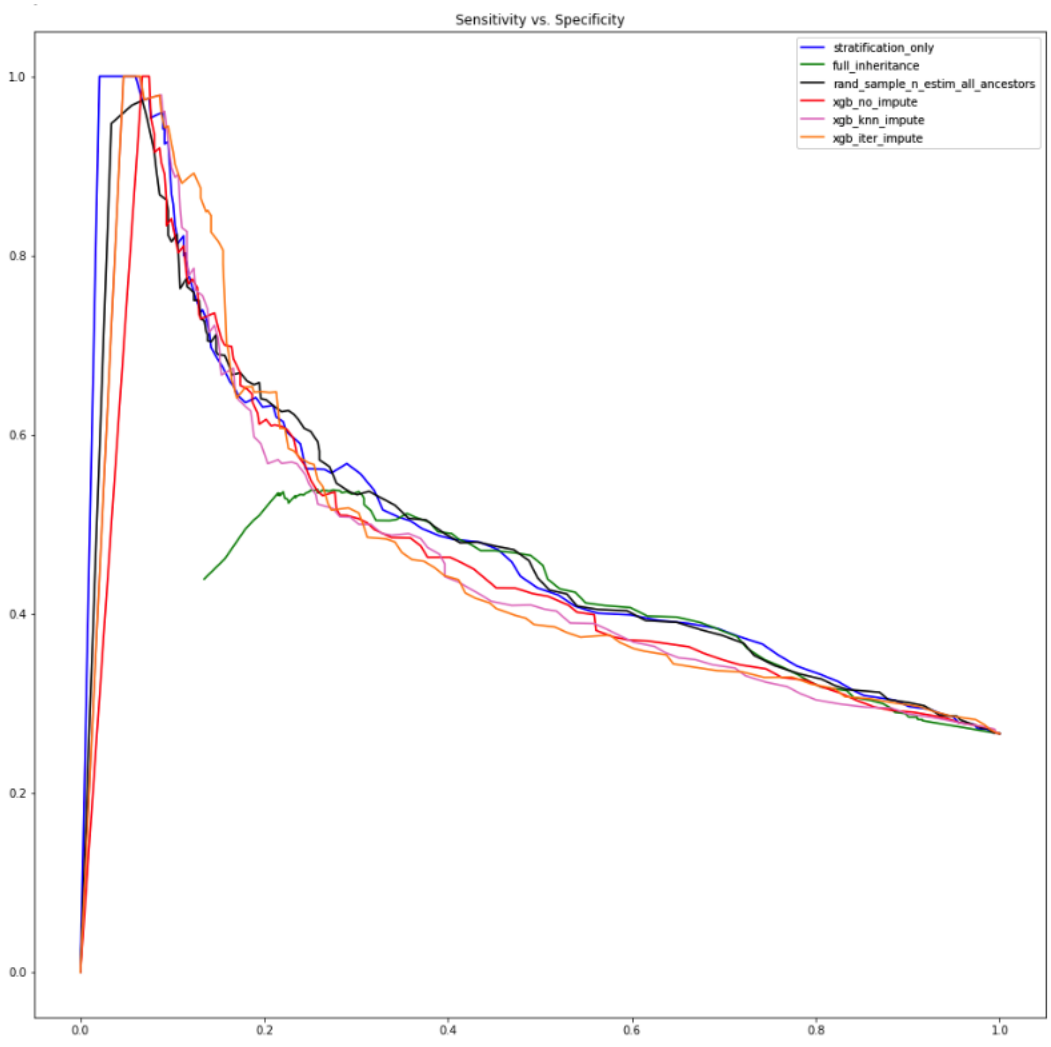

Figure 12: Sensitivity vs. Specificity Curve Comparison: NuMom2b Hypertensive vs. Non-Hypertensive

Table 3: NuMom2b Dataset: Classifier Types' Performances

NuMom2b Dataset: Classifier Types' Performances						
Model	Sensitivity	Specificity	PPV	NPV	G-Mean	AUROC
XGBoost (no imputation)	25%	92%	54%	77%	48.4%	66.9%
XGBoost (KNN imputation)	25%	92%	54%	77%	48.1%	65.9%
XGBoost (MICE imputation)	26%	92%	55%	77%	48.9%	66.4%
Meta-Ensemble Full Inheritance	**31%**	90%	53%	**78%**	**52.6%**	67.8%
Meta-Ensemble Sample n estimators total	26%	93%	59%	**78%**	49.1%	68.9%
Meta-Ensemble No Inheritance	21%	**95%**	**61%**	77%	45.0%	**69.0%**

6 Discussion

On the NuMom2b dataset, the stratified meta-ensemble methods I've proposed tend to outperform the XGBoost methods on every metric; perhaps most notably, the full-inheritance method outperforms the other methods on sensitivity on imbalanced datasets, with acceptable tradeoff in specificity such that my method's g-mean is also higher than the g-mean of any other evaluated method. The inheritance-via-random-sampling method also had the highest PPV of any evaluated method. The three meta-ensemble methods also outperformed the XGBoost methods on AUROC (though this is less important than the other metrics given the severe class imbalance). On the synthetic missingness dataset, the stratified full-inheritance meta-ensemble method still outperforms XGBoost (without imputation) most dramatically on sensitivity and g-mean, in addition to NPV and similar AUROC, getting outperformed in specificity and PPV.

One hypothesized takeaway is that propagating learnt information from each joint distribution can lead to increased performance, though it remains to be investigated more systematically. Perhaps the key takeaway here, though, is that the meta-ensembles appear to have different strengths and weaknesses, and correspondingly different clinical applications; there are certain clinical cases in which you'd want to optimize for sensitivity or specificity, or PPV or NPV, and so choice of machine learning technique depends on the problem at hand. Sensitivity is a problem for learning on imbalanced datasets, and when not much data is available it can be costly to undersample. As a result, building a model that non-trivially improves sensitivity on imbalanced datasets can potentially open up new clinical application areas for prediction modeling.

7 Works In Progress and Future Directions

Building on the momentum of the discussion section, a line of research I am following up with is an investigation of why the sensitivity increases so much when the inheritance mechanism is applied. Relatedly, it is to be seen whether these changes in performance (increase in sensitivity and g-mean especially) remain when working with balanced datasets. There are two primary ways to investigate this: (1) an empirical, data-driven way, where the individual nodes are investigated; (2) a probability / statistical learning theory approach, where expectations on the predictions are calculated.

Another important thread is that of computational efficiency. While I only used 9 tests on the NuMom2b dataset and 6 tests on the Wisconsin Breast Cancer dataset, scaling up to larger and larger amounts of tests is prohibitively difficult considering the current approach of training 2^n ensembles for n tests. For instance, if I had also included the placental analytes at visit 1, the number of models to train would jump from 2^9 to 2^18; it took 403 seconds to train the entire set of 2^9 ensembles, so we could expect that it would take at least 403×2^9 seconds (two and a half days) to train this ensemble, and correspondingly larger amount of memory needed for the models. Given the high dimensionality of medical datasets, a more pragmatic training methodology is necessary. One such approach is to only build nodes for those combinations we have seen; when using the meta-ensemble to make predictions on a test-set, we can assign test-set nodes to their closest ancestor and proceed in the usual way. Another approach is to only build nodes for those combinations which have significantly more samples than their children. We can also consider designing a policy for deciding which nodes to build, introducing a certain amount of exploration away from the policy as well.

Given issues related to undersampling and list-wise deletion in general, it might make sense to try oversampling.

However, state-of-the-art oversampling techniques such as SMOTE or ADASYN [Cha+02; He+08] require missing values to be imputed. Since we want to avoid imputation, another next step is to design a missingness-aware variant of, say, SMOTE, to generate synthetic samples that also contain missing values.

Because the different inheritance mechanisms seem to have different strengths and weaknesses, in order to get the best predictions out of them, we may consider combining them with yet another meta-ensemble technique, such as stacked generalization [Wol92]; we may even consider including the XGBoost methods in here, as they tend to perform better on specificity, at least on the Wisconsin dataset. We may also consider testing with different types of estimators in the ensembles, as opposed to gradient boosting only. Another direction that appears fruitful is extending the model to do multi-output learning: in this way, we can predict multiple potential outcomes at once, utilizing what we've learned from the prediction of one outcome for the next.

Finally, this entire project was motivated in large part by a further goal of optimizing medical test-selection. Test-selection is a sequential decision-making problem, which can be formulated as optimizing for uncertainty reduction in some target variables of interest. The initial ideas for this meta-ensemble technique were spawned out of an investigation of screening tests for preeclampsia, namely utilizing the first-trimester uterine artery Doppler ultrasound exam as a screening test, and determining who might benefit most from this test. Only a small subpopulation actually received this test, and when we tried training machine learning models on only the samples who received this test, the model performed worse than a model trained on the whole population but without including the ultrasound. In this case, some patients who may have otherwise gotten a reduction in uncertainty of development of preeclampsia from the ultrasound may have been determined to have an *increase* in uncertainty as a result. This led to the idea of retaining the information learnt from the larger sample set with less features and propagating it to the sub-populations who received the test, which is more natural in a sequential decision-making context: there are certain features already observed previously, and knowledge is built on top of that. One reason for using ensembles in this context especially is that one can define a notion of *confidence interval* (or simplex) with ensembles: order the class probability predictions of all the estimators for a given sample and take some central interval/simplex (e.g. the 25th percentile to the 75th percentile) of values as the confidence simplex. Reduction in uncertainty can be seen as a combination of (1) moving the probability prediction closer to one class or another (reduction in data uncertainty), (2) shrinking the size of the confidence simplex (reduction in knowledge uncertainty), and (3) shrinking the degree to which the confidence simplex overlaps with multiple classes (combination of data uncertainty and knowledge uncertainty) [MMG19]. See Figure 13 for an illustration of what such confidence intervals might look like (this figure is taken from a preliminary experiment with confidence intervals on the NuMom2b dataset).

References

[Buc60] Samuel F Buck. "A method of estimation of missing values in multivariate data suitable for use with an electronic computer". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 22.2 (1960), pp. 302–306.

[Wol92] David H Wolpert. "Stacked generalization". In: *Neural networks* 5.2 (1992), pp. 241–259.

[WSM94] William H Wolberg, W Nick Street, and Olvi L Mangasarian. "Machine learning techniques to diagnose breast cancer from image-processed nuclear features of fine needle aspirates". In: *Cancer letters* 77.2-3 (1994), pp. 163–171.

[Cha+02] Nitesh V Chawla et al. "SMOTE: synthetic minority over-sampling technique". In: *Journal of artificial intelligence research* 16 (2002), pp. 321–357.

[He+08] Haibo He et al. "ADASYN: Adaptive synthetic sampling approach for imbalanced learning". In: *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*. IEEE. 2008, pp. 1322–1328.

[VG11] Stef Van Buuren and Karin Groothuis-Oudshoorn. "mice: Multivariate imputation by chained equations in R". In: *Journal of statistical software* 45 (2011), pp. 1–67.

[SB12] Daniel J Stekhoven and Peter Bühlmann. "MissForest—non-parametric missing value imputation for mixed-type data". In: *Bioinformatics* 28.1 (2012), pp. 112–118.

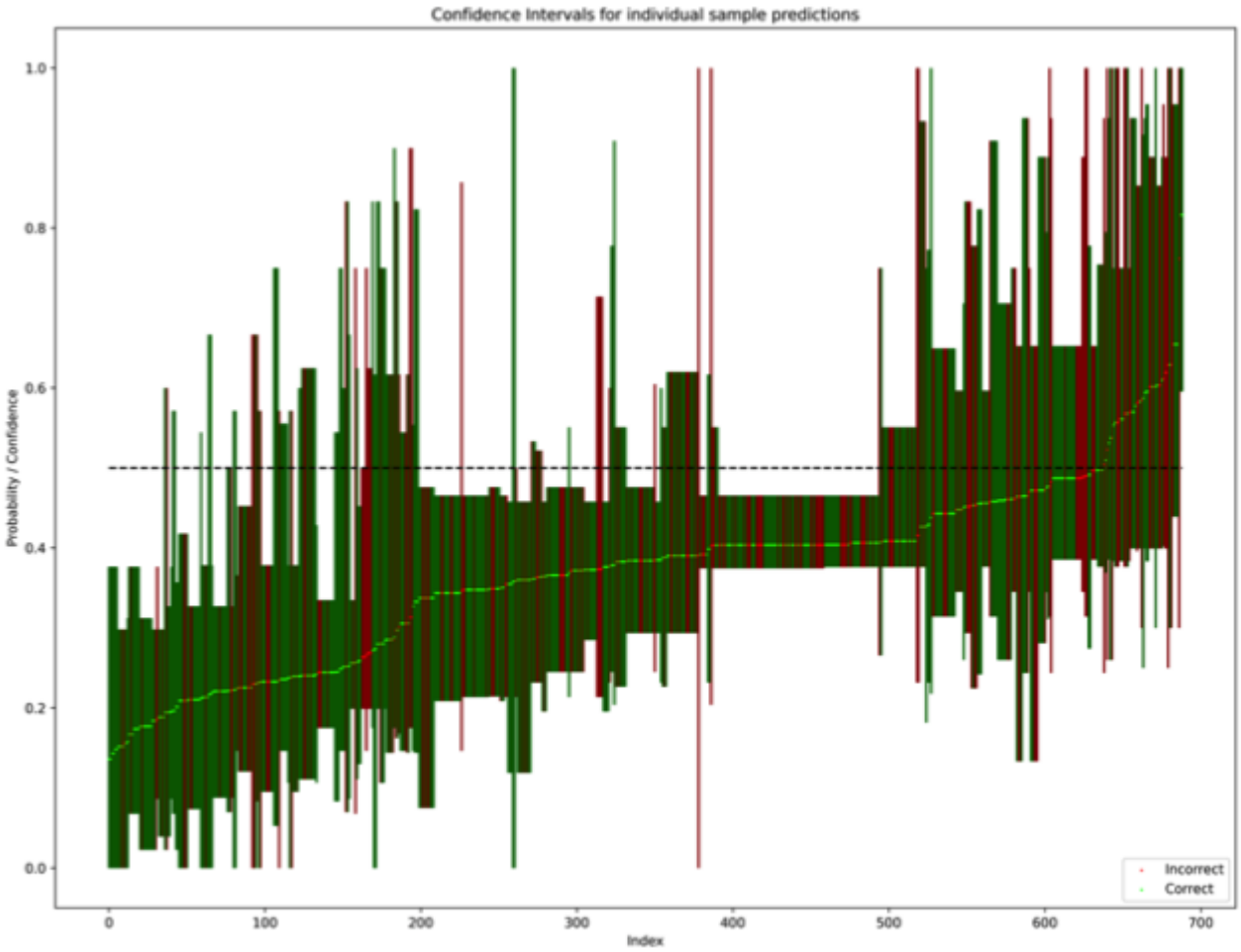

Figure 13: Meta-Ensemble, stratification with full inheritance node-wise metrics: number of samples and mean error per node with at least one sample. Levels (corresponding to number of tests) along with text schema are provided by the nodes on the right-hand side. Bright-green dots are correct predictions, bright-red dots are incorrect predictions; green and red bars are confidence intervals (25th-75th percentile of ordered predictions over all estimators) of correct and incorrect predictions, respectively.

[Haa+15] David M Haas et al. “A description of the methods of the Nulliparous Pregnancy Outcomes Study: monitoring mothers-to-be (nuMoM2b)”. In: *American journal of obstetrics and gynecology* 212.4 (2015), 539–e1.

[CG16] Tianqi Chen and Carlos Guestrin. “Xgboost: A scalable tree boosting system”. In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794.

[LR19] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*. Vol. 793. John Wiley & Sons, 2019.

[MMG19] Andrey Malinin, Bruno Mlodozeniec, and Mark Gales. “Ensemble distribution distillation”. In: *arXiv preprint arXiv:1905.00076* (2019).

[Gor+21] Anton Goretsky et al. “Data Preparation of the nuMoM2b Dataset”. In: *medRxiv* (2021).