Draft: Comparison of missing data handling methods in building variant pathogenicity metapredictors

Mikko Särkkä[1,2], Sami Myöhänen[1], Kaloyan Marinov[1], Inka Saarinen[1], Leo Lahti[3], Vittorio Fortino[2], and Jussi Paananen[1,2]

[1]Blueprint Genetics Ltd, Helsinki, Finland
[2]University of Eastern Finland, Institute of Biomedicine, Kuopio, Finland
[3]University of Turku, Department of Future Technologies, Turku, Finland

1 Introduction

1.1 Variant pathogenicity prediction

Adoption of next-generation sequencing (NGS) technology has greatly increased the scalability of genetic sequencing in both research and clinical genetics. Whole-exome and whole-genome sequencing (WES and WGS) are now becoming standard methodology, allowing detection of variants in a much broader set of loci than was previously feasible. However, the numbers of detected variants from each sample present problems especially in clinical contexts, for example performing genetic testing to identify pathogenic variants for diagnostic purposes. Manual exploration of even stringently filtered variant lists is time-consuming, while timely diagnosis is often needed.

In-silico variant effect and pathogenicity prediction tools such as SIFT [1], PROVEAN [2], MutationTaster2 [3, 4], LRT [5] and FATHMM [6] have been developed to inform the selection of variants for further testing in research contexts (though also often used in clinical variant interpretation, see ACMG/AMP guidelines [7]). Gene and variant prioritization tools such as VAAST [8], PHEVOR [9], FunSeq [10], PHIVE [11] and Phen-Gen [12] have been developed to rank variants. See reviews by Peterson & al. [13], Niroula & Vihinen [14], and Eilbeck & al. [15].

Many traditional *in-silico* prediction tools are in general based on evolutionary

conservation (e.g. SIFT, FATHMM, PROVEAN[1]), with some adding predicted change in protein function (e.g. PolyPhen2 [16], MutPred [17], MutPred2 [18]).

Another class of predictors, called *metapredictors* or *ensemble predictors*, such as REVEL [19], CADD [20, 21], DANN [22], Eigen [23], PON-P [24] and MetaSVM and MetaLR [25], are developed by using machine learning to combine outputs from several previously developed tools. Metapredictors often report high performance [26, 27]. Since they are able to jointly utilize tools that are aimed at variants with different predicted consequences by, for example, combining missense variant effect prediction tools and splicing variant effect tools, they are often also able to predict for wide ranges of variants.

Since traditional tools have been developed using different data, different methods and at different times, they may differ in the set of variants for which prediction is possible, even if their intended use cases match. That is, SIFT may be able to make a prediction on a missense variant for which MutationTaster is not. In practice, this means that the input features for metapredictors have *missing values*. A similar situation occurs when, for example, incorporating allele frequency information from gnomAD [28]: variants that were not observed in the study will have a missing value as their allele frequency.

1.2 Aim

The high prevalence of missing values in annotated variant data implies that the chosen method for handling missingness will have a major impact on the performance of variant pathogenicity metapredictors. Our aim is to identify missingness handling methods most likely to produce good results in this context. This includes comparing achieved classification performance, difficulty of implementation, and computation time. For this purpose, we present a framework implemented in the `R` language [29] that treats variant data with a variety of methods and then trains and evaluates a classifier on each treated dataset.

We focus on the comparison of imputation methods, as they are the most generally applicable and allow use of any machine learning method on the imputed dataset. These properties are shared by the missingness indicator augmentation method, which we therefore also include. We limit our evaluation to imputation of numerical features, as categorical features have a relatively natural treatment through an additional category denoting missingness.

We perform three experiments based on executing the above-mentioned framework on the ClinGen [30] subset of variants from ClinVar [31, 32, 33]. In the first experiment, we repeatedly simulate additional missing values in the training dataset. We then use thus generated datasets and the framework to evaluate, for each imputation method,

1. the effect of increasing missingness on classification performance,

[1] do I need to cite these again, or is the one-paragraph distance sufficiently short?

2. the relationship between imputation error, which is the difference between original known values and imputed values, and classification performance.

In the second experiment, we perform repeated random sub-sampling cross-validation on the training set to assess whether there are differences in the methods' susceptibility to changes in dataset composition.

In the third, main experiment, we use the full data and wider parameter grids to execute the framework to obtain a realistic estimate of the effect of imputation method choice in variant pathogenicity metapredictor construction.

2 Background

2.1 Missing data

Data with and without missing values are often called *incomplete* and *complete* data, respectively. Consider a matrix of A that represents the unobserved, underlying values that would be obtained by data collection in the absence of any missing data generation mechanisms. The subset of values of A that are observed in data collection is denoted A_{obs}, and the subset of missing values of A is denoted A_{mis}. The values of A_{mis} are not known when analysing any real dataset. M is the missingness indicator matrix whose values are 0 when the corresponding value of A is observed, and 1 when the corresponding value of A is missing.

Traditionally, missing data mechanisms are classified into *missing completely at random* (MCAR), *missing at random* (MAR) and *missing not at random* (MNAR) [34]. In a missing data process with an MCAR mechanism, the probability of a value being missing does not depend on any observed or unobserved values. With an MAR mechanism, the probability of a value being missing may depend on observed values, and with an MNAR mechanism, the probability of a value being missing may depend on both observed and unobserved values. Van Buuren notes data from a truly random sample of a population as an example of MCAR [35, subchapter 1.2]; measurements for individuals that were not selected in the sample do not depend on the underlying values of the individual or the observed values of other individuals. Note however, that missing values for measurements on variants obtained by a random sample of human subjects is not MCAR. The sampling is not over the set of possible variants, but over individuals harboring those variants, and the likelihood of a variant's value being missing thus depends on the variant's allele frequency. [2] Further, for non-MCAR missingness a good example could be gnomAD [28] allele frequency. A missing value may indicate that the variant was not present in the study population, and thus imply that it might also be extremely rare in healthy individuals. Very low allele frequencies

[2] Are there actually study designs that sequence a set of people and then just grab a set of variants (removing duplicates) and then annotate them? IIRC the case is actually that we would count how many times each variant occurs in the sample. Here missingness implies that it was not observed; this does depend on the allele frequency though...

thus end up with missing values, and thus exhibit MNAR behavior. Alternatively, it might indicate the variant is present, but was not observed due to residing in a difficult-to-sequence locus, thus exhibiting MAR behavior dependent on position.

The above classification is essential in statistical inference, and the validity of different methods depends on which mechanisms appear in the data. However, Sarle [36] notes that "The usual characterizations of missing values as *missing at random* or *missing completely at random* are important for estimation but not prediction", and Ding & Simonoff [37] provide evidence in support of this statement in the use of classification trees. In an interesting reversal, the presence of *informative missingness* [36, 37] in the data, i.e. missingness being dependent on the response variable conditional on A_{obs}, may actually lead to improved predictive accuracy compared to complete data [37].

2.1.1 Missingness handling in prediction

Rather than inferring properties of the statistical distribution of variant annotations, our aim is production of predictions of pathogenicity, be it a classification into pathogenic and non-pathogenic, a class probability or some other metric. Our aim is thus not statistical inference, for which most missingness handling methods have been developed. It is not clear how the performance of a missingness handling method in one context relates to its performance in another. Overall, there is a significant distinction between these aims, which leads to a variety of differences in the concerns that one must account for [38, 39].

Most studies introducing imputation methods focus on problems where missingness may occur in the model estimation phase, but where data is assumed to be complete at prediction time. Similarly, comparisons of missingness handling methods in machine learning tend to consider missing values only in the context of the training set. In this context, it suffices to design methods which facilitate model training in a way that maximizes generalization performance. In our use case, missing values proliferate also in the test set, and as such any methods must also allow treatment of the data on which we predict.

This raises additional challenges. Most imputation methods are not implemented in a way that easily allows reuse of learned parameters. That is, it is difficult to first estimate imputation method parameters on the training set, and then impute the test set using the same parameters. This leads to diminished prediction accuracy, as the distributions of imputed data differ in the training and test sets. Even worse, since the parameters for test set imputation are estimated from the test set, the content and size of the test set itself may affect the predictions for individual observations, and make it impossible to predict one observation at time.

2.1.2 Methods for missingness handling

Most of both machine learning models and statistical models are unable to directly use data with missing values. To use such data, we can adopt one of the following strategies.

Removal of incomplete observations. Simple removal of incomplete observations is a valid approach when all data we wish to predict on can be assumed to be complete. This is not a viable alternative for variant pathogenicity prediction, since the predictor would be unable to predict on any variant whose feature vector contains any missing values. The number of such variants grows as more features with missing values are added, and as such would greatly restrict the available set of features.

Model-based approach. There exist classification methods that can train and predict directly on feature vectors with missing values. Such models may require explicit specification of the mechanism behind missing values. E.g. Marlin [40] presented a variation on linear discriminant analysis (LDA) that can both be trained and predict on feature vectors with missing values. Random forest [41] offers a method for handling missing data by iteratively making use of the proximities of observations [42], but is only available for the training phase. CART offers a missingness handling method [43] via so-called surrogate splits, but this method is not suitable for an ensemble of trees like random forest (see [44]). The `randomForestSRC` R package [45] implements a missingness handling strategy that is usable in both training and prediction phases [44]. [3]

Imputation. Imputation is the process of replacing the missing values in data by explicit values. When done exactly once, after which the imputed (or *completed*) dataset is analyzed in the ordinary fashion, the process is termed *single imputation*. Common single imputation strategies are [34]:

- constant imputation, for example imputation with zeroes,
- unconditional mean imputation, where missing values within the same feature are replaced by the mean of the observed values of that feature,
- conditional mean imputation, where replacing values can be means dependent on the observed values of other features, for example by modeling with regression, and
- drawing from a predictive distribution, where values can be estimated by, for example, addition of a random deviate to a conditional mean estimate, or by drawing the value randomly from the set of observed values.

If aiming to perform statistical inference on the imputed data, one has to carefully ensure that, in addition to unbiasedness of any parameter estimates, uncertainty introduced by missingness is correctly reflected in the estimated standard errors. This is in contrast to designing methods simply for accurate prediction of the underlying value (which is yet again distinct from only facilitating the accurate

[3] Should I drop this sentence? We don't use this method due to time constraints and the very large number of moving parts

prediction of the response variable).

Indeed, naïvely imputing data with a single imputation method is misleading as use of even highly accurate single imputation methods will cause underestimation of the standard errors in inference [35, subchapter 2.6]. The uncertainty can be properly incorporated via two main avenues: *likelihood-based approaches* and *multiple imputation* (see [34]). Likelihood-based approaches account for missing values by integrating out the parameter representing the missingness generating process, and do not require explicit imputation of missing values [34].

As opposed to single imputation, the main idea of multiple imputation for statistical inference is to impute the incomplete data multiple times with draws from predictive distributions, fit separate models on each imputed dataset, and *pool* the parameter estimates [34].
In classification, there are several ways to utilize the set of imputed datasets. The first is to obtain estimates of the variability of the classification performance due to the randomness from draws from a predictive distribution. The second would be that one can in principle train a classifier on each, averaging results in the prediction phase [40, 46]. The benefits of this latter approach are not explored in this paper.

Reduced models. A somewhat brute-force approach to missingness handling in classification is to train a separate predictor for each combination of missing features using only the available features for each subset. This approach is called the reduced models [40] or reduced-feature models [47] approach, and is observed by Saar-Tsechansky & Provost to consistently perform well [47]. However, a naïve implementation of reduced models easily leads to very high computational time and storage requirements, and the hybrid and on-demand approaches described by Saar-Tsechansky and Provost [47] are not trivial to implement.

Missingness indicator augmentation. A conceptually simple method is to add, for each original feature, an extra indicator feature that takes the value 1 in observations with the original feature missing, and 0 when the original feature is observed [48]. The missing values of the original feature are then filled with zeroes, and identical indicator features may be removed. The downside with this method is that it may lead to a significant increase in dimensionality of the data (doubling it in the worst case).

2.1.3 Missingness handling in existing tools

Missingness causes and handling in traditional tools. Most missingness in variant annotation data can be attributed to the following causes:

1. Insufficient information related to a sequence
2. Inapplicability to the variant

The first cause describes situations where, for example, protein function change prediction tools based on metrics computed from multiple aligment are unable

to find sufficiently many matches to the input sequence. In these cases, the tool has no information on which to base its estimates. Another example of this cause is values estimated from scientific studies, e.g. allele frequencies, functional properties of proteins or expression levels of transcripts. Variants that are unobserved in the cohorts, excluded from genotyping microarrays or inside hard-to-sequence loci will have no observed allele frequency.

The second cause refers to attempts to annotate a variant with a tool that is inapplicable to its annotated molecular consequence. For example, a tool predicting change in protein function cannot produce an estimate for an intergenic variant.

Missingness handling in existing metapredictors. Strategies for missingness handling vary significantly between different existing metapredictors. REVEL [19] uses k-NN imputation when a variant's missingness is $\leq 50\%$, and mean imputation when missingness is $> 50\%$. CADD [20, 21] and DANN [22] use a mix of manually defined default values to replace missing values, with added missingness indicators for certain features, and mean imputation for genome-wide measures (see Supplementary information for [20]). M-CAP replaces missing values with constants representing maximally pathogenic prediction for each component tool [49]. Eigen [23] utilizes separate strategies for training and test phases. Eigen builds several weighted linear combinations of its features depending on variant type, only requiring annotations applicable to a specific variant to be available. Learning the weights is based on pairwise correlation, which can be estimated in the presence of some missing values. In the test phase, Eigen performs mean imputation for features that are applicable to the specific variant. KGGSeq ignores any variants that has missing values in its features [50]. PRVCS [51] removes variants with missing values in the training phase, and replaces missing values of a feature by its population mean in the test phase.

2.1.4 Multiconsequence predictors

Missingness due to inapplicability could be avoided by training separate models on each molecular consequence, and using only features fully applicable to each consequence. This would still leave the first cause, and thus still require missingness handling, even if to a lesser degree. The dimensionalities of the feature spaces of each consequence-specific classifier would be lower without a loss of information, possibly increasing performance (see also section above on reduced models). However, this requires each consequence category to have sufficiently many observations to train a classifier. In contrast, a single classifier for predicting regardless of consequence class can be trained on the whole data set, possibly allowing the predictor to generalize information between variants with different consequences.

We choose to focus on building a single classifier for single-nucleotide variants and small indels.

3 Materials and methods

We implement a framework that enables comparison of imputation methods by their contribution to the performance of machine-learning classifiers, specifically for prediction of variant pathogenicity. For this purpose, the framework preprocesses variant data to a usable format, performs imputation, trains a classifier and computes relevant performance statistics.

This framework is used to perform three experiments:

1. a simulation experiment, where additional missing values are induced in the dataset several times, and the framework is used on each resulting dataset, described in subsection 3.6,
2. a cross-validation experiment, where the framework is used on datasets produced using repeated random sub-sampling, described in subsection 3.7, and
3. a main experiment, where the framework is used once with more comprehensive hyperparameter ranges for imputation methods, described in subsection 3.8.

3.1 Data

3.1.1 ClinGen dataset

The dataset consists of ClinGen[30] expert-reviewed single-nucleotide variants from ClinVar[31, 32, 33], downloaded on 28 June 2019. We annotated the variants using the Ensembl Variant Effect Predictor (VEP)[52] version 96 with Ensembl transcripts and dbNSFP3.5 [53, 54]. We selected transcripts annotated canonical by VEP for each variant and removed the others, and where values were present for several transcripts, discarded values unrelated to the selected transcript. Any variants whose canonical VEP-annotated transcript ID did not match that from dbNSFP were discarded. In addition, we incorporated annotations used by CADD [21], matching them by transcript to the VEP-annotated Ensembl transcripts. At this stage,the dataset contained 12282 rows.

3.2 Preprocessing

The overall preprocessing process is depicted in Figure 1.

The feature vectors of some sets of variants may be equal (that is, duplicated). We removed variants with duplicated feature vectors (N= 320), retaining only one variant from each equivalence class formed by duplicated feature vectors. We also chose to drop variants of uncertain significance (VUS, N = 1157) from the dataset.

The data was then randomly split into training and test subsets, with 70 % (N= 7536) of variants in the training set and 30 % (N= 3269) of variants in the test set.

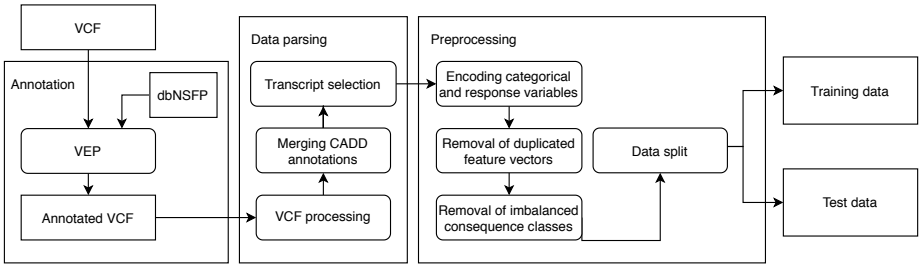

Figure 1: Preprocessing diagram. Data processing is roughly divisible into annotation, data parsing and preprocessing. Some preprocessing actions are deferred to the training phase, as their results may change due to simulated additional missing values.

We formed a binary outcome vector by defining variants classified as pathogenic or likely pathogenic to belong to the positive class (N = 5090 in the training set, N = 2218 in the test set), and variants classified as benign or likely benign to belong to the negative class (N = 2446 in the training set, N = 1051 in the test set).

Categorical features were transformed to dummy variables, with an extra category denoting a missing value. It is important to note that thus no imputation methods designed specifically for categorical features were tested, and categorical features were not treated by the used numerical imputation methods.

Use of categorical features with high class imbalance within certain levels may obfuscate the performance of the imputation methods. This is due to allowing the classifier to learn to classify all variants with that level into either the positive or the negative class, and therefore ignoring all other features upon which imputation may have been performed. VEP-predicted variant consequence is one such feature. For this reason, we removed variants with consequences for which either class had less than 5 % of overall variants of that consequence (removing 4930 variants from the training and 2139 variants from the test sets). A prediction tool developer might prefer to not make such restrictions to retain wide applicability, but it is important that they carefully analyze their results. They should especially pay attention to whether their method is significantly better than simple assignment to the majority class in such cases.
For features where the missingness implied the default value *a priori*, missing values were replaced by default values.

Table 1: Features imputed with default values.

Feature name	Feature interpretation	Default value
`motifDist`	"Reference minus alternate allele difference in nucleotide frequency within an predicted overlapping motif" [20, Supplementary information]	0
`gnomAD_exomes_AF`	gnomAD allele frequency from exomes	0
`gnomAD_genomes_AF`	gnomAD allele frequency from genomes	0

The final processed dataset contains 3736 variants divided into a training set with a total of 2606 variants, of which 1088 are positive and 1518 are negative, and a test set with a total of 1130 variants, of which 476 are positive and 654 are negative.

To minimize issues due to singular matrices with some imputation methods (e.g. MICE linear regression, MICE predictive mean matching), we removed features with fewer than 1 % unique values. For feature pairs with high correlation, we kept only one of the features. Removal of features with few unique values or high correlation is performed as part of the training process, just before imputation, since they may be affected by introduction of additional missing values for simulations (see Simulations). It is possible that some imputation methods would be able to deal with feature sets for which the above treatment was not done, and perhaps gain an advantage due to the additional information. However, we chose to use the same restricted feature set for all imputation methods for simplicity and comparability.

A mock-up illustrating the preprocessed data format is shown in Table 2.

Table 2: Mockup of data after preprocessing. Variant identification information is not used in imputation or training. gnomAD allele frequency has a value exactly 0 on rows 1, 3, 4, and 5 due to *a priori* imputation.

Variant	SIFT	DNase-seq levels	gnomAD AF	mirSVR	Non-synon.	Intronic	...
1:215625828:T:C	NA	0.3126	0.0	NA	0	0	
2:39022774:T:C	0.001	0.2607	4.070e−06	NA	1	0	
2:47410217:G:A	0.033	0.3818	0.0	NA	1	0	
2:47797737:C:T	NA	0.5263	0.0	NA	0	1	
13:32399139:A:G	NA	0.0750	0.0	−0.1513	0	0	

3.2.1 Features

The initial feature set was defined manually to include a variety of traditional tools and annotations from both dbNSFP3.5 and the annotation set for CADD while excluding any metapredictors from the feature set. See supplementary information.

3.2.2 Missingness

Missingness percentages of the features conditional on the predicted variant or transcript[4] consequence are presented in Figure 2. As expected, `INTRONIC`, `UPSTREAM`, `DOWNSTREAM`, `NON-CODING_CHANGE` and `5PRIME_UTR` variants have the most missingness, exhibiting large missingness percentages across protein-related features and splicing predictions, leaving mainly features related to regulation[5]. `INFRAME` variants have similar pattern as previously described variants, but have observed values in features describing variant position in coding sequence and protein codon. Such features are partially observed in `SPLICE_SITE` variants, possibly due to some of them also being interpretable as coding variants. `SPLICE_SITE` variants have a feature describing the distance to a splice site completely observed, but curiously, it is `3PRIME_UTR` variants instead that have observations of splicing predictions[6]. `NON-SYNONYMOUS` variants have, as expected, nearly completely observed feature vectors, except for mostly unobserved splicing predictions and incompletely observed gnomAD allele frequencies, MutPred predictions and REMAP [**remap**] annotations.

Since many features are only applicable to variants of a specific consequence, there will be few complete cases. Further, the only complete cases can be formed by variants that are interpretable as having several consequences, even if only one is assigned for the purposes of analysis. Such variants are, for example, splicing variants. They can often also be interpreted to be non-synonymous or intronic depending on their position[7]. The complete cases are thus the very few variants that have both splicing predictions as well as the features available for non-synonymous variants.

To assess whether there is informative missingness in the data, we computed each feature's correlation to the outcome indicator (see Supplementary information). For some features, there is moderately large correlation, the largest of which is for MutPred (≈ -0.33).

[4] Which? Kind of both?
[5] check with somebody that this is accurate
[6] ask somebody about this
[7] check with somebody that this is accurate

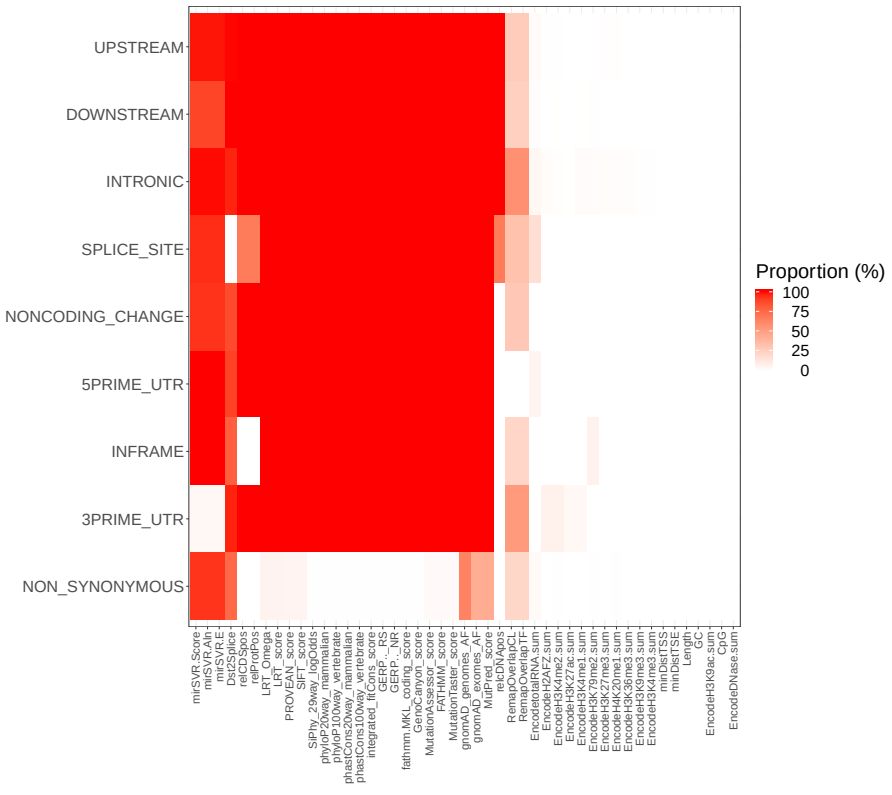

Figure 2: Missingness heatmap of the training set. Each cell displays the proportion of missing values of the indicated feature (horizontal axis) in the predicted molecular consequence class (vertical axis).

3.3 Imputation methods

3.3.1 Univariate imputation

The simplest imputation methods impute every missing value within a feature with the same value, which may either be a constant or a statistic estimated from the observed values of the feature.

In the case of missingness indicators, features with identical missingness patterns produce identical indicator vectors, of which only one is kept.

3.3.2 Multiple imputation by chained equations

Multiple imputation by chained equations (MICE) [55, 56, 57] is an algorithm that iteratively imputes single features conditional on other features. In short, a MICE method

1. uses univariate imputation to sequentially impute each feature conditional on the observed values of other features
2. reimputes each feature conditional on the imputed data from the previous iteration

Step 2 is repeated until some maximum number of iterations or some measure of convergence is reached.

We used the R `mice` package [57] to perform the imputation. The following methods provided by `mice` were used: Each method was run with maximum 10 iterations.

3.3.3 Other imputation methods

Besides unconditional univariate methods and MICE methods, we tested three popular imputation methods.

For k-NN, you must have enough complete cases to start imputation, depending on k. As described above, the data had very few complete cases, and thus the largest k that could be used was 2.

Table 3: Used imputation methods

Imputation method	Description	Implementation
Zero imputation	Replace by 0	Custom
Maximum imputation	Replace by maximum observed value within feature	Custom
Minimum imputation	Replace by minimum observed value within feature	Custom
Median imputation	Replace by median observed value within feature	Custom

Imputation method	Description	Implementation
Mean imputation	Replace by mean observed value within feature	Custom
Outlier imputation	Given observed values F_{obs} of feature F, replace by $\lvert\max(F_{obs}) - \min(F_{obs})\rvert \cdot 10$	Custom
Missingness indicator augmentation	For each feature, perform zero imputation and create a binary feature indicating original missing values	Custom
Predictive mean matching	Sampling from observed values similar to a predicted value	`pmm` in package `mice`
Random forest	Predict values using random forest	`rf` in package `mice`
Linear regression	Predict values from a linear regression model	`norm.predict` in package `mice`
Bayesian linear regression	Predict values from a linear regression model with added uncertainty modeling noise and variability of parameter estimates	`norm` in package `mice`
BPCA [58]	Predict values via Bayesian principal components analysis	`bpca` in package `pcaMethods`[59]
k-NN	Predict values as mean of k nearest neighbors wrt. other features	`knnImputation` in package `DMwR` [60]
MissForest [61]	Predict values using random forest	`missForest` in package `missForest` [62]

3.4 Framework

The framework performs imputation and classifier training and evaluation on a preprocessed training and test dataset pair. The preprocessed training set is used to filter out features that have insufficiently unique values or that are heavily correlated with some other feature, and this filtering is matched on the test set.

Each imputation method is used on the training set, producing at least one imputed dataset for each combination of hyperparameters (see below), and classifiers are trained on each dataset.

Multiple imputation methods can be used to produce several imputed datasets using the same hyperparameter configuration, and we make use of this in the main experiment. For probabilistic single imputation methods, the method can be run multiple times with different seeds, producing a set of imputed datasets analogous to that generated via multiple imputation. We apply this approach to

MissForest.
For each completed dataset from a probabilistic or multiple imputation method, we train a separate classifier (performing its usual hyperparameter search and model selection procedure separately on each dataset).

In order to maximize the performance of each imputation method for fair comparison, hyperparameter grids were defined for each method for which different hyperparameters[8] could be passed. For the simulation and cross-validation experiments we decided to save time by using fewer hyperparameter configuration, sampling a maximum of 8 hyperparameter configurations for each imputation method used on a dataset. The hyperparameter grids are described in table 4.

Table 4: Statistics on hyperparameter grids for imputation methods. For k-NN, only two values of `k` succeed, see subsection Other imputation methods.

Method	Varied hyperparameters	Number of combinations
MICE PMM	`donors`, `ridge`, `matchtype`	180
MICE regression	None	1
MICE Bayes regression	None	1
MICE Random forest	`ntree`	36
k-NN	`k`	5 (effectively 2)
BPCA	`nPcs`, `maxSteps`	112
MissForest:	`mtry`, `ntree`	12
Simple methods	None	1 each

After imputations of the training set and classifier training, the hyperparameter configuration with highest downstream classifier performance (or highest mean performance, in case of multiple imputation methods), and its associated classifiers, are selected and stored for each method, and that hyperparameter configuration is used when imputing the test set. Finally, performance is computed using the associated classifiers and associated test set (or test sets, for probabilistic and multiple imputation) of each imputation method.

The process is depicted visually in figure 3.

3.4.1 Classifiers

We restricted our analysis to two common classification methods. We chose to study both a simpler, less flexible baseline method, and a more complex and highly flexible machine learning method in order to see whether classifier flexibility affects the choice of best imputation method. In this study, we refer

[8] Are these usually called hyperparameters in imputation methods? Technically I think they are hyperparameters even if not called that, but it might be better to conform to common terminology.

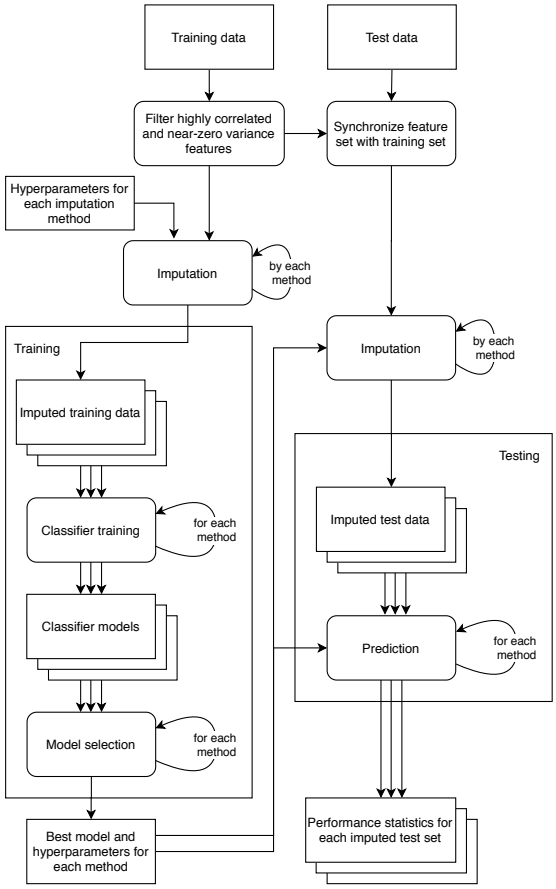

Figure 3: Framework flowchart. Features are filtered on the training data according to their correlations and variance, and the feature set of the test data is synchronized to match the filtered feature set. Afterwards, the training data is imputed using each imputation method, with each hyperparameter configuration. In the main experiment, multiple imputation methods are set to produce 10 imputed datasets per hyperparameter configuration. Only one dataset is set to be produced in the simulation and cross-validation experiments. Every dataset is then used to train a classifier, and for each imputation method the best-performing hyperparameter configuration is selected by the training-set performance of the corresponding classifier. For multiple imputation methods, the mean performance of the corresponding classifier set is used. The test set is then imputed with the selected hyperparameter configuration for each method, and corresponding classifier is used to predict on the imputed test set(s). Classifier performance is then evaluated on each set of predictions.

to classifiers that are trained and predict on datasets imputed by an imputation method as *downstream classifiers* of that imputation method. For the baseline downstream classifier, we chose logistic regression (LR), a standard statistical method for two-class problems; for the more complex method, we chose random forest (RF) [41], a highly flexible machine-learning method. Both methods have been used in existing variant pathogenicity predictors (e.g. KGGSeq [50] and later versions of CADD [21] utilize logistic regression, while e.g. MutPred [17], PON-P [24], REVEL [19] and Meta-SNP [63] utilize random forest).

Logistic regression is used with the base R `glm`, and Random Forest is used via the package `randomForest` [64]. Both are trained and applied to test data via functionality from the `caret` package [65].

The random forest was trained using the out-of-bag (OOB) performance for model selection. `glm` does not offer any tuning parameters.

3.5 Metrics

We use *Matthews' correlation coefficient* (MCC) as our main evaluation metric since it is less misleading than other common classification performance metrics in imbalanced datasets [66].

MCC is defined as

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FN)(TP + FP)(TN + FP)(TN + FN)}}.$$

We also present results with the *area under ROC curve* (AUC-ROC, or just AUC) metric, defined as the area under the receiver operating characteristic curve.

Finally, we compute the root-mean-square error (RMSE) to compare imputed and original values when using simulated additional missing values. RMSE is defined as

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}$$

where y_i is the ith true value, $\hat{y}_i$ is the ith prediction (in this case, the imputed value), N is the number of predictions (in this case, the number of imputed values).

3.6 Simulation

We performed simulations with additional missing values in order to

1. compare the sensitivity of imputation methods to different levels of missingness,

2. relate downstream classifier performance and an imputation method's predictive performance, i.e. the capability of the imputation method to predict the original value underlying a missing value.

A common strategy for studying imputation methods' performance is simulation of missing values either on fully simulated data, or on the complete subsets of real datasets. In our context of application, the number of complete cases in the dataset is very low and thus could not be used as the basis for simulating additional missing values.
Instead, we chose to take the full, incomplete dataset as the basis of simulations, and used the following strategy:

1. Create many simulated datasets based on the full dataset with additional missing values using `ampute` [67] on the dataset while varying missingness percentage
2. Impute each simulated dataset with each imputation method
3. Compute RMSE for each simulated dataset with respect to values that were observed in the original dataset but missing in the simulated dataset
4. Train a classifier on each imputed simulated training dataset, and evaluate performance on imputed original test set

We use nine percentages of additional missingness, from 10 % to 90 %, each producing 100 datasets with the respective amount of additional missingness. We therefore have 900 simulated datasets on which the framework is executed. To reduce the computational requirements, we downsample the hyperparameter grids of each imputation method (in comparison to the main experiment). In addition, since MissForest is significantly more computationally expensive than other imputation methods, it is not included in the simulation experiment.

3.6.1 Amputing additional missing values

The input matrix of `ampute` is required to be complete, so we partitioned the data according to missingness patterns. This forms a set of matrices for which every feature is either fully observed or fully missing. We then used each of the fully observed submatrices as inputs to `ampute`, and combined the resulting output back to form a matrix of the original size.

3.6.2 Simulation evaluation

Many imputation studies are specifically built to assess an imputation method's capability to predict the original values from the observed values of a dataset with missing data. Thus, they use the RMSE between the original dataset and an imputed dataset as a metric of performance of the imputation method. However, a model with the lowest RMSE is not in general the best one in the statistical inference context [35, chapter 2.6]. In short, the best model wrt. RMSE is linear regression estimated via least squares, and the deterministic nature of a regression prediction necessarily ignores uncertainty due to the missing data.

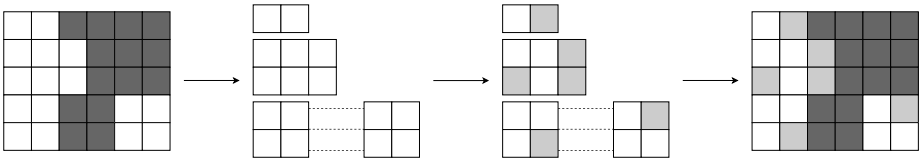

Figure 4: Partitioning and simulating missingness. White blocks represent available values in a data matrix, while dark gray blocks represent original missing values and light gray blocks represent additional missing values. The rows of the original dataset are partitioned to matrices with equal missingness patterns. Each feature in such matrices is now either completely observed or completely missing, and the complete features can be treated together as a fully observed matrix. Additional missingness is simulated separately using `ampute` on the fully observed matrices, which are then combined.

The same argument cannot be applied to the predictive context, but the same end result may apply when missingness is informative.

Consider a situation where missingness is highly informative. Then a perfect imputation method (with respect to RMSE; i.e. one where RMSE would equal 0) would impute the dataset with the original values. However, the informativeness in the missing values would be lost. The loss of information could, in principle, be avoided by adding missingness indicators before imputing, but this comes with the increase in dimensionality (doubling the number of features in the worst case) and thus cannot be seen as a universal solution.

Since RMSE cannot be used as a universal metric of imputation method performance, we perform a lean version of the framework on each simulated dataset. That is, we impute each simulated dataset using a sparser hyperparameter grid (downsampled to eight hyperparameter configurations separately for each simulated dataset) and producing only one dataset each when using probabilistic methods. A classifier of each type is trained on the imputed simulated datasets, the best performing imputation hyperconfiguration is chosen by the highest performing classifier trained on a dataset imputed via that configuration. Performance is estimated on the test set imputed with the winning configuration. We compute, in addition, RMSE of each imputation method on each dataset, and can thus investigate whether low RMSE on the training set predicts high downstream classifier performance on the test set. [9]

As mentioned in the Preprocessing section, due to the removal of features with fewer than 1 % unique values and features that highly correlate with another feature, the addition of missing values may lead to differing feature sets between different simulated datasets. Especially large numbers of additional (MCAR) missing values may lead to fewer unique values, and both increase or decrease

[9] Is this block necessary? Does it suffice to say (with a single sentence) that we run the framework on the datasets?

correlation between features by chance.

3.7 Cross-validation experiment

In the cross-validation experiment, the training data is used to produce 100 training/test dataset pairs via repeated random sub-sampling with a 70 % split. The framework is run separately on each pair, after which the results can be used to estimate imputation methods' relative robustness to variation from sampling. For speed, hyperparameters for imputation methods are downsampled to eight hyperparameter configurations each, and multiple imputation methods are set to produce only a single dataset. However, due to the fewer required executions of the framework (when compared to the simulations), it was feasible to also include MissForest in the set of methods.

3.8 Main experiment

In the main experiment, the framework is run once with the full hyperparameter grid for each imputation method on the full training and test datasets, and MissForest is included in the set of methods. We also utilize the multiple datasets generated by multiple imputation methods to estimate the performance variability that arises from randomness in the imputation of both training and test sets.

3.9 Challenges

3.9.1 Dataset composition

Even after *a-priori* imputation, there is a very high proportion of missing values in the dataset ($\sim$ 31 %). The missingness exhibits a general pattern, and is not e.g. monotone (see [34, pp. 6–7]). In addition, missingness appears in most features, and the number of complete cases ends up minuscule (2 rows in training data), and removal of any single feature would not significantly increase the number of available complete cases (see figure 2). The missingness can also not in general be assumed to be MCAR, though due to the arguments by Sarle [36] and Ding & Simonoff [37] mentioned earlier, we expect this has little effect on classification performance.

3.9.2 Studying imputation methods developed for statistical inference in a predictive context

As mentioned in subsection 2.1.1, the differences in intended usage between statistical inference and prediction make use of existing implementations–many of which were designed for the former purpose–difficult. The first and main issue is that out-of-the-box implementations often do not provide an easy way to reuse learned parameters from an earlier run. This makes it difficult to use parameters from the training set on the test set.

Some ways to deal with this are

- *Reimplementation.* One can reimplement the method in a way that allows reuse of parameters, fully solving the problem. However, this is work-intensive, and requires deep understanding of each imputation method.
- *Ignore.* One can ignore the issue, and allow the imputation method to re-estimate its parameters when imputing the test set. However, this may lead to diminished classifier performance on the test set, as the distributions of imputed values may differ between training and test sets and thus may confound the classifier.
- *Concatenation of incomplete datasets.* When imputing the test set, concatenate the training set and test set, impute the combined dataset and then remove rows belonging to the training set. This reduces the difference between the distributions compared to *Ignore*, but may be computationally expensive.
- *Concatenation to imputed training data.* One can make a variation on *Concatenation of incomplete datasets* by imputing the training set before concatenation. This is faster than *Concatenation of incomplete datasets* if the imputation method does not run in linear time with respect to the size of the input dataset. However, it may introduce additional issues, since it will use the imputed values in the training set to estimate imputations for the test set.
- *Concatenation of single observation*: A second variation of *Concatenation of incomplete datasets* is to concatenate only one observation from the test set at a time, and repeating this until the full test set is imputed. This variation makes it possible to predict on a single observation at a time. This is computationally expensive.

Reimplementation and *Concatenation of single observation* are the only options that allow imputation of test observations independently from each other. The others perform imputation with parameters estimated from other observations that are being predicted on at the same time.

Imputation method / family	Implementation	Out-of-the-box parameter reuse
MICE	`mice`	No
BPCA	`pcaMethods`	No
k-NN	`DMwR`	Yes
Simple methods	custom	Yes
MissForest	`missForest`	No

The package `mlr`[68] offers wrapper functionality that allows use of any prediction method offered by the package also for univariate imputation, along with functionality for correct reimputing data with previously learned parameters, but using this option is difficult in a dataset with very few complete cases. We did not explore this possibility in this work. Investigation of the imputation

performance of methods originally intended for prediction might merit further study.

We choose to implement option *Ignore* due to its simplicity for methods where out-of-the-box parameter reuse is not available. There is a possibility that this will give an advantage to simple methods and k-NN even if MICE, BPCA and MissForest would otherwise outperform them. However, this comparison still represents the situation as it shows itself to the practitioner that may not have the time or expertise to make use of the other options.

3.10 Implementation

The framework and experiments were implemented with R [29] and organized into 11 executable scripts:

Number	Description	File name
1.	Parsing of VCF file	`01_parse_vcf.R`
2.	Preprocessing	`02_preprocess_data.R`
3.	Computation of descriptive statistics	`03_descriptive_stats.R`
4.	Execution of training set imputation and classifier training for main experiment	`04_run_impute_and_train.R`
5.	Execution of test set imputation and prediction for main experiment	`05_run_prediction.R`
6.	Generation of datasets with additional missing values	`06_generate_simulated_data.R`
7.	Execution of imputation and classifier training on datasets with additional missing values	`07_run_simulations.R`
8.	Process and plot test set performance statistics for main experiment	`08_analyze_results.R`
9.	Process test set performance statistics for classifiers trained on datasets with additional missing values	`09_analyze_simulation_results.R`
10.	Plot test set performance statistics for classifiers trained on datasets with additional missing values	`10_simulations_plots.R`

Number	Description	File name
11.	Perform repeated random sub-sampling cross-validation and process and plot its results	`11_crossvalidation.R`

The scripts are intended to be executed in order, but users may choose only run a subset if they are only interested in a subset of the results. To run the main experiment, one must run 1., 2., 4., 5., and 8.; to run the simulations, one must run 1., 2., 6., 7., 9. and 10.; to run the crossvalidation experiment, one must run 1., 2. and 11.

3.11 Availability of source code and requirements

Project name: AMISS

Project home page: https://github.com/blueprint-genetics/amiss

Operating system(s): Linux

Programming language: R [29]

Other requirements: The software was run with R 3.6.0 with packages vcfR [69], futile.logger [70], tidyr [71], here [72], magrittr [73], ggcorrplot [74], mice [57], foreach [75], doParallel [76], ggplot2 [77], iterators [78], missForest [62, 61], DMwR [60], doRNG [79], rngtools [80], lattice [81], itertools [82], randomForest [64], ModelMetrics [83], stringr [84], gridExtra [85], digest [86], purrr [87], caret [65, 88], testthat [89] and e1071 [90], and pcaMethods [59] via BioConductor[91] BiocManager [92].

License: MIT

Any restrictions to use by non-academics: CADD annotations[10] require commercial users to contact authors for licensing. dbNSFP [54] annotations may require licenses for commercial use and must be reviewed individually.

3.12 Availability of supporting data and materials

The data sets supporting the results of this article are available in the Zenodo repository, DOI [**amiss-zenodo-doi**].

4 Results

This section is arranged as follows. First we present results from the added-missingness simulations. We evaluate both the relation of downstream classifier

[10] I am not sure whether this refers to all annotations or just the score, which I don't use.

performance and RMSE, and how increasing missingness percentage affects downstream classifier performance. Next, we present results from the repeated random sub-sampling cross-validation experiment, shedding light on robustness of imputation methods to dataset composition. Finally, we present results pertaining the main experiment, comparing the downstream classifier performances of imputation methods. Finally, we compare the imputation methods with respect to their running times.

4.1 Simulation experiments

4.1.1 RMSE

We found that RMSE and classification performance as measured by MCC did not correlate in datasets with simulated additional missing values (see supplementary information). This is especially clear in the case of outlier imputation, where RMSE is–as expected–much higher than with any other method, while MCC was comparable to other methods.

4.1.2 Missingness percentage

The effect of additional MCAR missingness on MCC performance of downstream classifiers is displayed in figures 5 and 6. Fitted LOESS (locally estimated scatterplot smoothing) curves are shown.

When the downstream classifier is a random forest, missingness indicator augmentation as well as mean, minimum, zero and median imputations show similar curves, with their average performances dropping from slightly below MCC = 0.85 at 30 % missingness to slightly above MCC = 0.75 at 70 % missingness. MICE random forest and MICE PMM have overall slightly lower mean performance than the previously mentioned methods, but otherwise show similar shapes. Outlier and maximum imputations suffer more drastically, with mean performances dropping to just above MCC = 0.70 at 70 % missingness. MICE regression and MICE Bayes regression demonstrate a curious effect where average downstream classifier performance increases at first as missingness increases, before starting their descent. BPCA shows hints of a similar but more muted trend. This may be related to a phenomenon noted by Poulos & Valle [93] in the context of prediction on categorical variables, where introduction of additional missing values prior to imputation may improve classifier performance.

When the downstream classifier is logistic regression, outlier imputation shows a dramatic drop immediately between 30 % and 40 % missingness, and stabilises slightly above MCC = 0.20. Maximum imputation shows a clear linear downward trend, while minimum imputation, zero imputation and BPCA show a much smaller one. Missingness augmentation and all MICE methods show very light reductions in average performance as missing value percentage grows. Median and especially mean imputation show practically no performance reduction due to increasing MCAR missingness. MICE Bayes regression, BPCA and PMM and

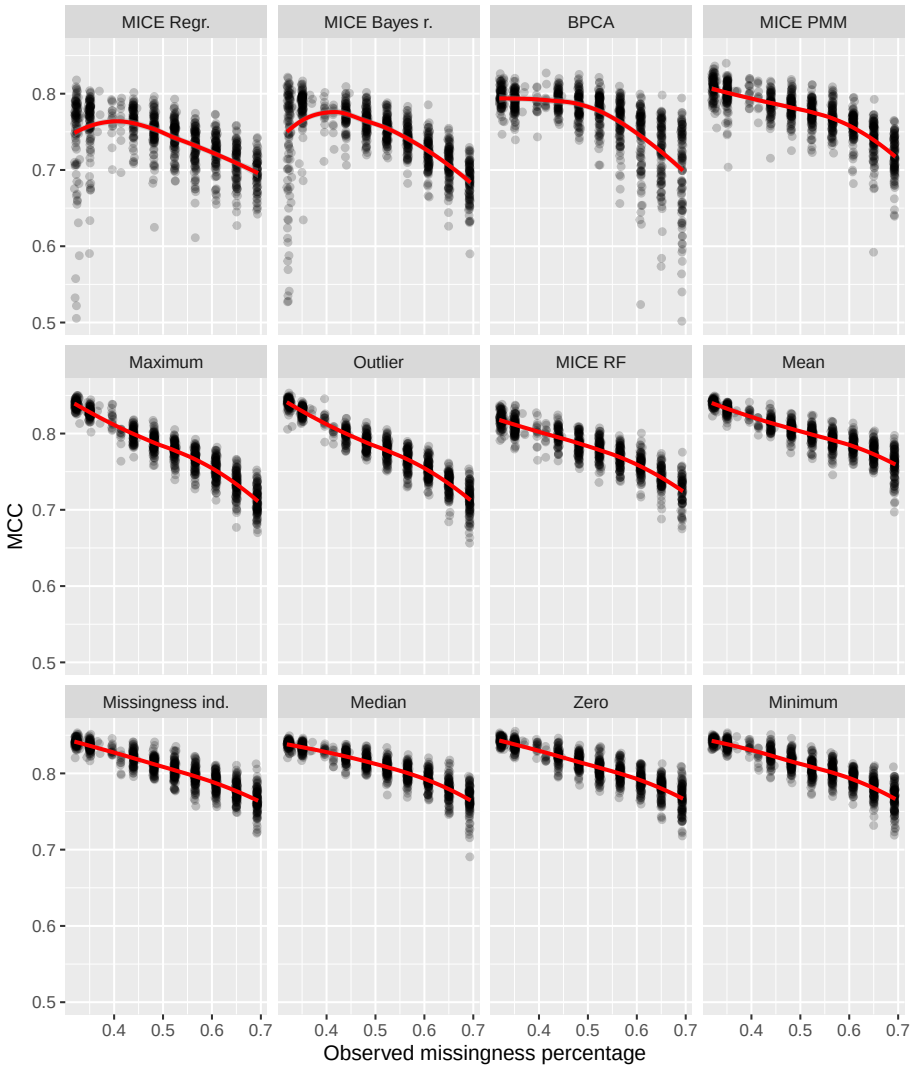

Figure 5: Random forest MCC against actual missingness percentage, with fitted LOESS curves (red).

especially MICE regression all show much larger variability in their performance than other methods.

4.2 Cross-validation experiment

The variability of downstream classifier performance evaluated via repeated random sub-sampling cross-validation is displayed in figure 7.

The random forest classifier outperforms logistic regression regardless of imputation method, and the lowest performing imputation method wrt. random forest classifiers (MissForest) has higher mean performance than the highest performing imputation method wrt. logistic regression (missingness indicators).

For both logistic regression and random forest, single imputation methods and k-NN appear to have equivalent performance, though in conjunction with logistic regression outlier imputation seems to perform slightly worse. With regard to random forest classifiers MICE random forest, BPCA, MICE Bayes regression and MICE PMM perform slightly or somewhat worse than simple imputation methods, but display greater differences in conjunction with logistic regression, where MICE random forest and BPCA are clearly preferable to MICE Bayes regression and MICE PMM. MICE ordinary regression and MissForest perform worse on average with both classifier types, but when combined with logistic regression, MICE ordinary regression brings mean classification performance down close to that of a coin flip.

4.3 Main experiment

In order to assess the differences in downstream classifier performance due to selection of imputation method, we show mean performance metrics for classifiers trained on datasets produced by each missingness handling method, sorted by MCC, for random forest classifiers (table 7) and for logistic regression classifiers (table 8). Single imputation methods (mean, missingness indicator, median, k-NN, zero, minimum, maximum, outlier) produced only a single dataset, and as such also produced only a single set of performance statistics. MICE ordinary regression imputation was unable to produce imputations in the main experiment, and is thus excluded.

In the case of random forest classification, maximum imputation has the best average performance with respect to MCC ($= 0.848$). It is very closely followed by missingness indicator augmentation, k-NN imputation and mean imputation, and then zero, median, minimum and outlier imputation methods. MICE random forest imputation is the highest performing MICE method (MCC $= 0.817$), with slightly better MCC than Bayes regression and PMM. The final group, with distinctly worse downstream classifier MCC, consists of BPCA (MCC $= 0.781$) and MissForest (MCC $= 0.756$).

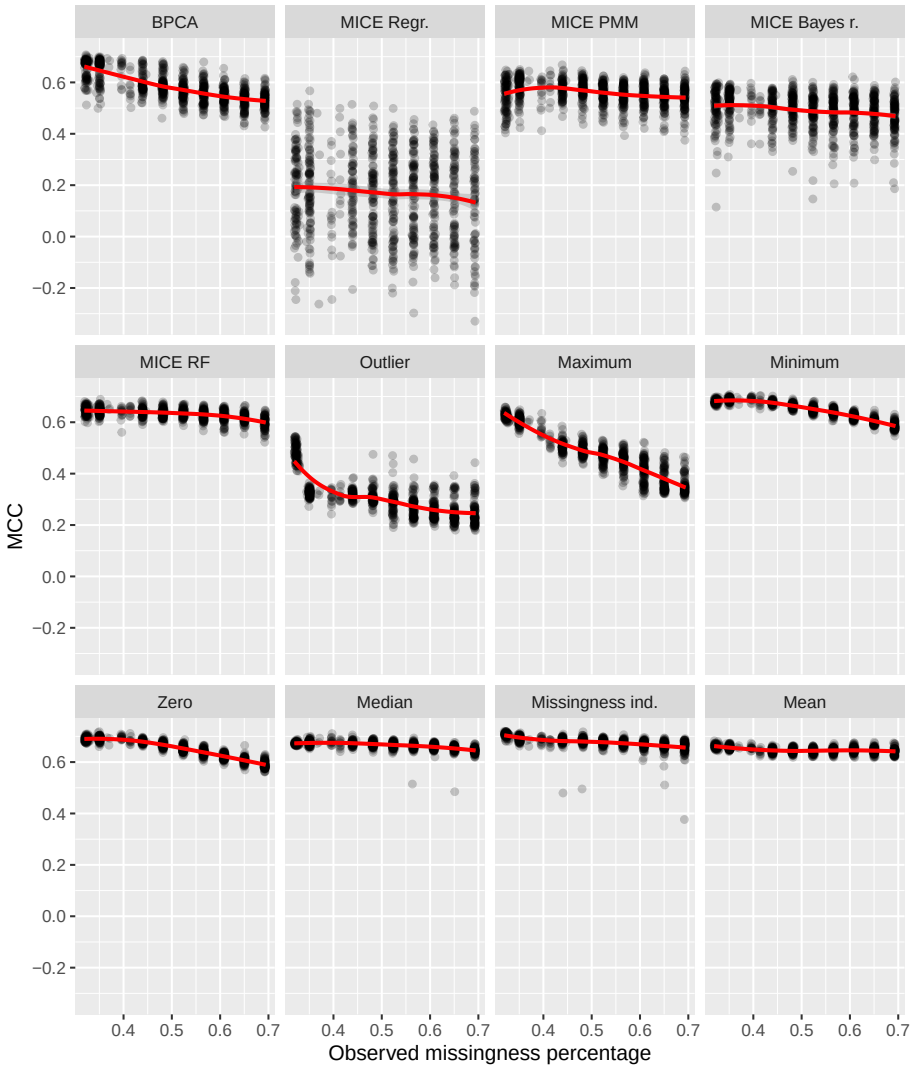

Figure 6: Logistic regression MCC against actual missingness percentage, with fitted LOESS curves (red).

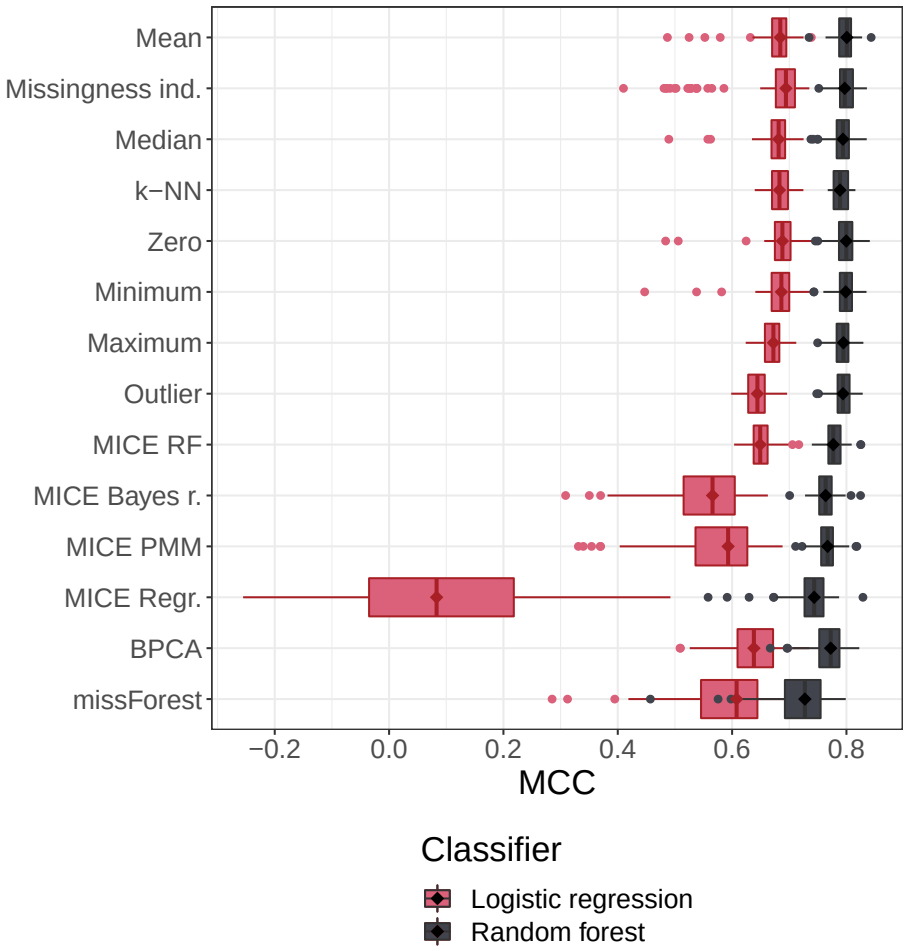

Figure 7: Boxplots describing down-stream classifier performance estimated via repeated random sub-sampling cross-validation. Diamonds were added to emphasize median values. The training set was further randomly split 100 times into 70%/30% subsets, and the framework was executed for each of the resulting pairs.

Table 7: Mean test set performance metrics for a random forest classifier trained and tested on data sets treated with each missingness handling method

Method	MCC	AUC-ROC	Sensitivity	Specificity	F_1	Precision
Maximum	0.848	0.975	0.912	0.937	0.913	0.914
Missingness indicators	0.846	0.975	0.908	0.937	0.910	0.913
k-NN imputation	0.844	0.973	0.918	0.930	0.911	0.905
Mean	0.844	0.974	0.914	0.934	0.912	0.910
Zero	0.840	0.976	0.905	0.933	0.906	0.907
Median	0.839	0.975	0.916	0.925	0.907	0.899
Minimum	0.838	0.975	0.903	0.934	0.906	0.909
Outlier	0.837	0.974	0.908	0.930	0.906	0.904
MICE RF	0.817	0.976	0.889	0.927	0.894	0.898
MICE Bayes regression	0.812	0.975	0.876	0.932	0.890	0.904
MICE PMM	0.812	0.975	0.869	0.936	0.888	0.909
BPCA	0.781	0.974	0.779	0.976	0.860	0.959
MissForest	0.756	0.952	0.818	0.927	0.853	0.892

Mean classification performance is lower across the board for logistic regression. Missingness indicator augmentation is the winner in this scenario with MCC = 0.700. k-NN imputation (MCC = 0.682) and BPCA (MCC = 0.680) receive second and third place, after which mean imputation, zero imputation, minimum imputation, maximum imputation, outlier and median imputations have essentially equal performance (MCC ranging from 0.674 to 0.669). MissForest and MICE methods (MCC between 0.645 and 0.591) perform noticeably worse.

Table 8: Mean test-set performance metrics for a logistic regression classifier trained and tested on data sets treated with each missingness handling method

Method	MCC	AUC-ROC	Sensitivity	Specificity	F_1	Precision
Missingness indicators	0.700	0.925	0.840	0.862	0.828	0.816
k-NN	0.682	0.924	0.817	0.865	0.816	0.816
BPCA	0.680	0.923	0.828	0.855	0.817	0.806
Mean	0.674	0.920	0.815	0.859	0.812	0.808
Zero	0.673	0.923	0.809	0.864	0.811	0.812
Minimum	0.672	0.922	0.813	0.859	0.810	0.808
Maximum	0.670	0.914	0.828	0.846	0.812	0.796
Outlier	0.670	0.906	0.863	0.813	0.815	0.771
Median	0.669	0.921	0.815	0.855	0.809	0.803
MissForest	0.645	0.929	0.695	0.921	0.769	0.868
MICE RF	0.642	0.909	0.761	0.874	0.787	0.815

Method	MCC	AUC-ROC	Sensitivity	Specificity	F_1	Precision
MICE PMM	0.601	0.898	0.709	0.874	0.751	0.810
MICE Bayes regression	0.591	0.879	0.785	0.805	0.763	0.750

The MCC and AUC-ROC performances of downstream classifiers are illustrated in figures 8 and 9, respectively. For multiple imputation and MissForest the variability of performance due to randomness in the imputation is also visible. Variability in downstream classifier performance is seen to be smaller for random forest compared to logistic regression in all methods except for MissForest, and variability in general larger in methods with lower mean performance.

Using AUC-ROC to measure performance makes it more difficult to compare imputation methods due to the very small absolute differences. For a random forest classifier (table 7), the mean AUC-ROC of all methods, with the exception of MissForest, is within 0.002 of each other. For logistic regression (table 8), the range is wider, and is also visually distinguishable in figure 9. Interestingly, here MissForest has the upper hand. However, looking back at table 8 we notice that MissForest shows the highest specificity but also the lowest sensitivity of all methods. The imbalance is reflected in the relatively poor MCC and F_1 scores, but ignored by AUC-ROC.

The results are not very different from the cross-validation experiment results. When compared using MCC, simple imputation methods are largely interchangeable, though missingness indicators dominate when using a logistic regression classifier, and maximum imputation is dominant when using a random forest classifier. However, when considering the variance shown in the cross-validation experiment, the differences between simple imputation methods are likely to be due to chance. It is important to note that here the variance for multiple imputation methods is due to the multiple imputation (or multiple runs of a probabilistic imputation method) of training and test sets, and thus is not directly comparable to that of the cross-validation experiments. MICE ordinary regression is not included here due to it failing in the main experiment.

4.3.1 Results grouped by consequence

To assess whether the strongly distinct missingness patterns exhibited within different variant consequence classes, we also computed performance statistics separately within certain consequence classes, specifically `DOWNSTREAM`, `UPSTREAM`, `INTRONIC`, and `Other`, an aggregate of all remaining consequence classes. `Other` is formed by the consequence classes for which either 1) there are overall few variants, and the class indicator gets filtered out due to non-zero variance, or 2) there is high correlation with another feature. Results are shown in figure 10.

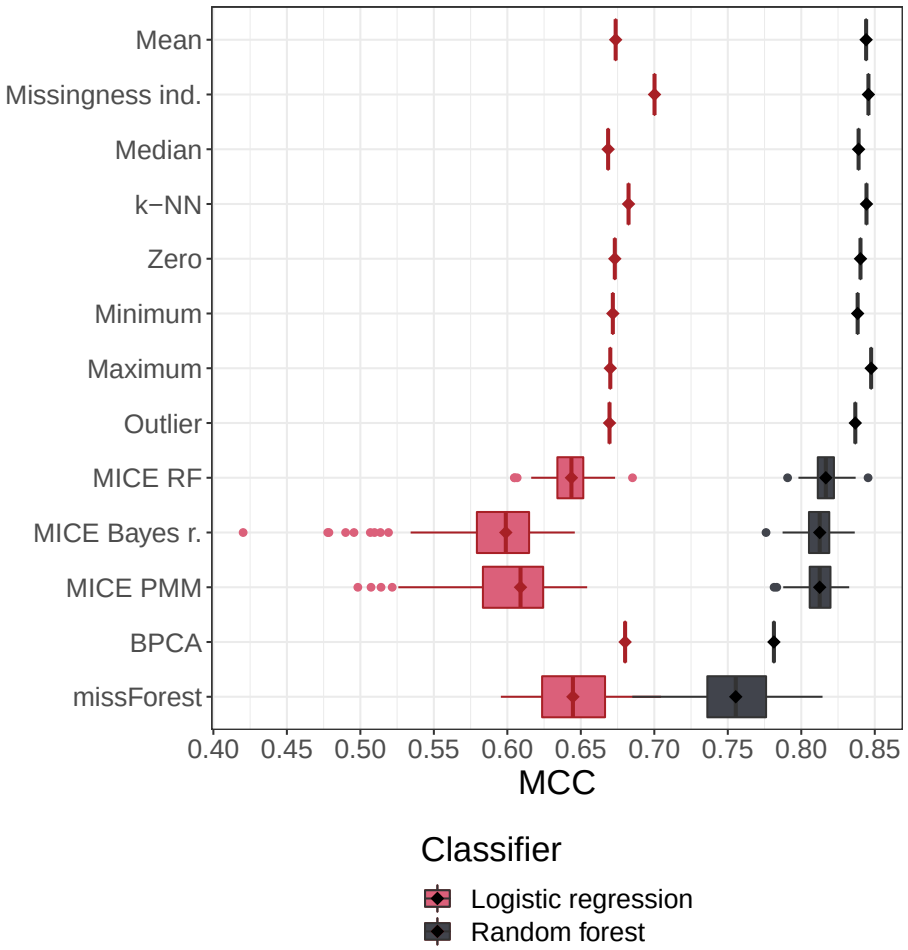

Figure 8: Boxplots describing down-stream classifier performance with respect to MCC. Diamonds were added to emphasize median values. Variance represents variation in probabilistic or multiple imputation, with 10 classifiers, each trained on a training set imputed with the same method, used to predict on 10 test sets imputed using the same method.

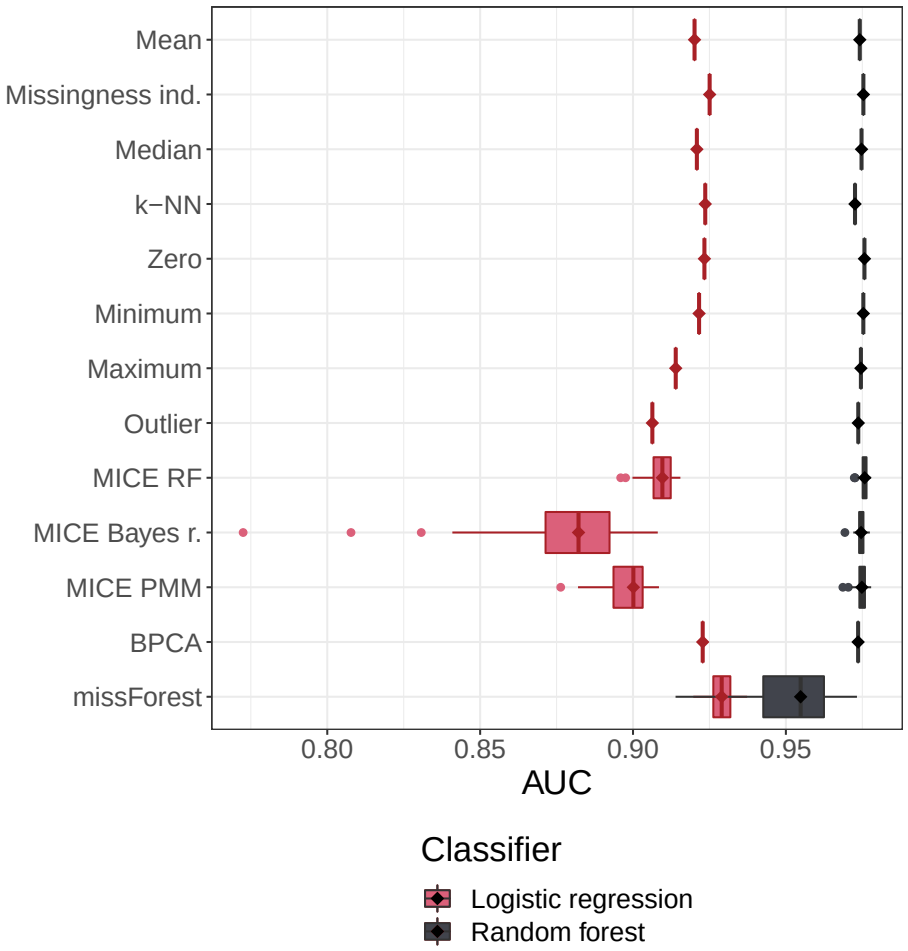

Figure 9: Boxplots describing down-stream classifier performance with respect to AUC-ROC. Diamonds were added to emphasize median values. Variance represents variation in probabilistic or multiple imputation, with 10 classifiers, each trained on a training set imputed with the same method, used to predict on 10 test sets imputed using the same method.

Table 9: Numbers of occurrence of variant consequence classes.

Consequence	Occurrences
INFRAME	14
NONCODING_CHANGE	14
5PRIME_UTR	18
3PRIME_UTR	27
SPLICE_SITE	53
DOWNSTREAM	464
NON_SYNONYMOUS	578
UPSTREAM	632
INTRONIC	806

Some consequence classes were indeed very sparse, as seen in table 9, and the pre-imputation filtering process eliminated the dummy variables encoding class membership of INFRAME, NONCODING_CHANGE, 5PRIME_UTR, 3PRIME_UTR, and SPLICE_SITE. In addition, membership to class NON_SYNONYMOUS was strongly (negatively) correlated to the missingness of LRT predictions, and was also eliminated by the pre-imputation feature filtering. Thus these classes were grouped into Other.

Performances show large differences in different consequence classes. Other is mostly non-synonymous variants, as described above, and shows good performance for both classifier types, though random forest is always superior. Mean imputation performs best with the random forest classifier, while missingness indicator augmentation and minimum imputation maximize logistic regression performance. The results are essentially equal to the general results depicted in figure 8 where consequence classes are not distinguished.

The consequence class INTRONIC shows a very different and interesting situation. First it is important to note that even after the preprocessing steps which removed variants in consequence classes with high class imbalance, only ≈ 8.1 % of INTRONIC variants are positive.However, the robustness of MCC to class imbalance allows us to compare the performances. In this consequence class, logistic regression is virtually useless, though missingness indicator augmentation and maximum imputation seem to allow some discriminatory power. It seems that in INTRONIC, logistic regression learns to classify almost everything as negative (see Supplementary information). With most imputation methods, MCC is barely above zero. In contrast, the random forest classifier performs well with any imputation method, though the high variation in MissForest imputations seems to allow also for instances of poor performance.

Compared to Other, prediction on variants of DOWNSTREAM consequence shows lower performance in both classifiers. For random forest, the relative orders of imputation methods are basically the same as in the main experiment and the cross-validation experiment. Unlike in all other consequence classes, logistic

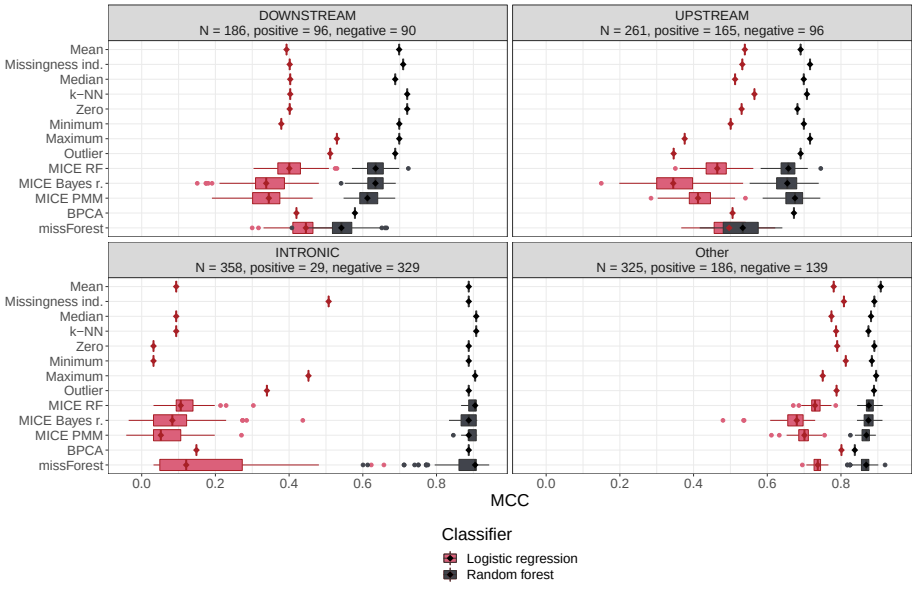

Figure 10: Boxplots describing down-stream classifier performance with respect to MCC, conditional on variant consequence. Diamonds were added to emphasize median values. Variance represents variation in probabilistic or multiple imputation, with 10 classifiers, each trained on a training set imputed with the same method, used to predict on 10 test sets imputed using the same method.

regression seems to perform better with MICE methods and MissForest than most simple imputation methods and k-NN, though maximum and outlier imputation do seem to outperform them even here.

In `UPSTREAM` variants, performance is again lower than in `Other`, but not as low as in `DOWNSTREAM`. Random forest classification shows the common pattern, where simple imputation methods and k-NN seem slightly preferable to MICE methods, and MissForest performs noticeably worse. Logistic regression classification looks, in this case, to actually be disadvantaged by the same methods that dominated `DOWNSTREAM` (that is, maximum and outlier imputation).

As mentioned earlier, minor differences between simple imputation methods have a good chance of being random.

4.4 Running time

Running times were recorded in the main experiment for the best hyperparameter configurations of all imputation methods, and are presented in table 10. For methods that were used to produce multiple datasets (MICE methods and MissForest) the overall time was recorded and then divided by the number of produced datasets (10). The machine used to run the software was a CentOS Linux server with two Intel Xeon CPU E5-2650 v4 @ 2.20GHz processors and 512 GiB[11] of memory. The analysis was run with parallelization using 24 processes. Each imputation was run inside a single process, and thus running times should not be affected based on whether an imputation method itself offers parallelization features.

Table 10: Running time for imputing the training set with the best hyperparameter configurations in the main experiment.

Method	LR (elapsed (s))	RF (elapsed (s))
Zero	0.007	0.007
Minimum	0.008	0.008
Maximum	0.009	0.009
Mean	0.009	0.009
Median	0.013	0.013
Outlier	0.014	0.014
Missingness indicator augmentation	0.553	0.553
MICE Bayes regression	5.475	5.475
MICE regression	6.033	6.033
k-NN	12.386	12.211
BPCA	14.461	19.15
MICE PMM	15.464	15.429
MICE RF	107.919	103.632
MissForest	363.018	363.018

[11] Report GiB or GB?

MICE regression and MICE Bayes regression show similar running times and finish around 6 seconds, while k-NN, MICE PMM and BPCA take somewhat longer. MICE random forest and especially MissForest take much longer. Simple imputation methods are much faster than all other methods.

5 Discussion

Random forest classifiers defeated logistic regression overall, with the worst-performing combination of imputation method and random forest still reaching higher performance than best-performing combination of imputation method and logistic regression. This was expected, due to both the inherent capability of tree predictors such as those in random forest to ignore irrelevant features, and the more flexible decision boundary that random forests are able to form.

The high comparative performance of k-NN with both classifiers is surprising. The depletion of complete cases greatly limits the possible values for k. Even more importantly, the few complete cases are the only possible neighbors to any point; when k equals the number of complete cases, every missing value in a specific feature will be imputed with the average value of that feature in the set of complete cases. k-NN imputation with k equaling the number of complete cases can be thus be interpreted to be a form of unconditional mean imputation, since in such a case every missing value will be imputed with a single value, irrespective of the values of other features.

The computational cost of different methods varies dramatically. Many of the highest-performing methods take a minuscule portion of time spent during the overall training and prediction process, while more costly methods dominate the time requirements. The difference between zero imputation (0.007 second runtime) to MissForest (363.018 seconds) may be inconsequential for a single sample, but is compounded with large cohorts or high-throughput diagnostic work. For example, for 100 000 WGS samples the difference in total computational time with the simplest and most complex method will be over a year, translating into hundreds of thousands of euros[12] of additional cloud computing costs.

Simple imputation methods overall could be considered the winning group of the comparison presented in this paper, but mean imputation might further be pointed out as the favorite in that group. In addition to the consistently high MCC in all experiments, it also displays high resilience to performance degradation due to increasing MCAR missingness, and low variance in relation to sampling. It is also one of the fastest methods to compute, trivial to implement and is always applicable.

Considering the complexity of the prediction task, the features, the large degree of missingness, and the sophistication of available imputation strategies, it is

[12]This originally referred to tens of years. 100 000 samples at 0,007 s per imputation gives 700 s = ~ 12 min, while 100 000 samples at 363 s per imputation gives 36300000 s = ~ 420 days. Should this then be changed to “tens of thousands of euros”?

somewhat surprising that the best performance is gained using simple univariate strategies.

5.1 Possible explanations to simple imputation superiority

5.1.1 Informativeness

A putative explanation is the presence of some informativeness in the missingness mechanism. If the presence of a missing value in certain variables correlates with the pathogenicity of the variant, simple imputation methods would be given an advantage: when every imputed value (within a given feature) is replaced by a single specific value, the classifier may learn to correlate that single value with the outcome. This is less likely to happen when using more sophisticated imputation methods, which make it harder for the classifier to learn which values were likely imputed. [13] However, logistic regression is less flexible and thus less capable of representing such potentially discontinuous dependencies, and yet missingness indicator augmentation did not perform dramatically better than most simple imputation methods. This would imply that informativeness does not hold a large role, even though we did find it to be present in the data. It may also be that the dimensionality increase due to the additional indicator features disadvantages the classifier enough to neutralize the benefit from informativeness.

5.1.2 Inability to transfer training parameters to test set

As described in Methods and materials, many available implementations of imputation methods do not allow reuse of parameters between imputation of training and test sets. Thus the estimated distributions on which imputed values are based will differ at least slightly, and thus disadvantage any methods for which parameter reuse was not possible.

5.2 Additional challenges

Grimm et al. [94] described several biasing factors in variant effect predictor training and performance evaluation using data from commonly used variant databases, e.g. the tendency for variants within the same gene being classified as all pathogenic or all neutral, or simply due to difficulty of finding datasets completely disjoint with the training set. Mahmood et al. [95] further analysed existing variant effect predictors using datasets generated from functional assays, and found drastically lower performance compared to earlier reported estimates.

Our approach is not immune to these biases, and we expect that any reported performance metrics will be overoptimistic. However, we expect that the main result of the study, the relative performance rankings of imputation methods, is not affected by the biases. The classifiers built described in this work are not

[13] Nasty anthropomorphism. How can I rephrase this more accurately?

intended to outperform earlier approaches or be directly used for variant effect prediction.

5.3 Conclusions

It appears that it is unnecessary to use sophisticated imputation methods to treat missing values when building variant pathogenicity metapredictors. Instead, simple univariate methods and even zero imputation give higher performance and save significant computational time, leading to considerable cost savings if adopted. This highlights the conceptual separation between imputation for prediction and imputation for statistical inference, the latter of which requires carefully constructed techniques to reach correct conclusions.

5.4 Further work

There are several ways to improve and expand on this work. The dataset could be extended to include variants from wider sources, and the effect of circularity could be estimated using additional datasets. It would likely also be possible to make changes to or reimplement methods whose implementation does not currently support reuse of parameters. Reduced models and its hybrid variants would make an interesting point of comparison if implemented.

6 Potential implications

We compared a variety of commonly used imputation methods in order to assess their suitability for using machine learning to build variant pathogenicity metapredictors. The analysis will help pathogenicity predictor researchers choose imputation methods to maximize the performance of their tools.

7 Competing interests

MS, SM, KM, IS and JP are employed by Blueprint Genetics. [14] LL has received compensation from and is a scientific advisor for Blueprint Genetics.

References

[1] Pauline C Ng and Steven Henikoff. "Predicting deleterious amino acid substitutions". In: *Genome research* 11.5 (2001), pp. 863–874.

[2] Yongwook Choi et al. "Predicting the Functional Effect of Amino Acid Substitutions and Indels". In: *PLOS ONE* 7.10 (Oct. 2012), pp. 1–13. DOI: 10.1371/journal.pone.0046688. URL: https://doi.org/10.1371/journal.pone.0046688.

[14] Do we need to still also have the funding section?

[3] Jana Marie Schwarz et al. “MutationTaster evaluates disease-causing potential of sequence alterations”. In: *Nature Methods* 7.8 (2010), pp. 575–576. ISSN: 1548-7105. DOI: 10.1038/nmeth0810-575. URL: https://doi.org/10.1038/nmeth0810-575.

[4] Jana Marie Schwarz et al. “MutationTaster2: mutation prediction for the deep-sequencing age”. In: *Nature Methods* 11.4 (2014), pp. 361–362. ISSN: 1548-7105. DOI: 10.1038/nmeth.2890. URL: https://doi.org/10.1038/nmeth.2890.

[5] Sung Chun and Justin C. Fay. “Identification of deleterious mutations within three human genomes”. In: *Genome Research* 19.9 (2009), pp. 1553–1561. DOI: 10.1101/gr.092619.109. eprint: http://genome.cshlp.org/content/19/9/1553.full.pdf+html. URL: http://genome.cshlp.org/content/19/9/1553.abstract.

[6] Hashem A. Shihab et al. “Predicting the Functional, Molecular, and Phenotypic Consequences of Amino Acid Substitutions using Hidden Markov Models”. In: *Human Mutation* 34.1 (2013), pp. 57–65. DOI: 10.1002/humu.22225. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/humu.22225. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/humu.22225.

[7] Sue Richards et al. “Standards and guidelines for the interpretation of sequence variants: a joint consensus recommendation of the American College of Medical Genetics and Genomics and the Association for Molecular Pathology”. In: *Genetics in Medicine* 17.5 (2015), pp. 405–423. ISSN: 1530-0366. DOI: 10.1038/gim.2015.30. URL: https://doi.org/10.1038/gim.2015.30.

[8] Mark Yandell et al. “A probabilistic disease-gene finder for personal genomes”. In: *Genome Research* 21.9 (2011), pp. 1529–1542. DOI: 10.1101/gr.123158.111. eprint: http://genome.cshlp.org/content/21/9/1529.full.pdf+html. URL: http://genome.cshlp.org/content/21/9/1529.abstract.

[9] Marc V. Singleton et al. “Phevor Combines Multiple Biomedical Ontologies for Accurate Identification of Disease-Causing Alleles in Single Individuals and Small Nuclear Families”. In: *The American Journal of Human Genetics* 94.4 (May 2014), pp. 599–610. ISSN: 0002-9297. DOI: 10.1016/j.ajhg.2014.03.010. URL: https://doi.org/10.1016/j.ajhg.2014.03.010.

[10] Ekta Khurana et al. “Integrative Annotation of Variants from 1092 Humans: Application to Cancer Genomics”. In: *Science* 342.6154 (2013). ISSN: 0036-8075. DOI: 10.1126/science.1235587. eprint: https://science.sciencemag.org/content/342/6154/1235587.full.pdf. URL: https://science.sciencemag.org/content/342/6154/1235587.

[11] Peter N Robinson et al. “Improved exome prioritization of disease genes through cross-species phenotype comparison”. In: *Genome research* 24.2 (Feb. 2014), pp. 340–348. ISSN: 1088-9051. DOI: 10.1101/gr.160325.113. URL: https://europepmc.org/articles/PMC3912424.

[12] Asif Javed, Saloni Agrawal, and Pauline C Ng. “Phen-Gen: combining phenotype and genotype to analyze rare disorders”. In: *Nature Methods* 11.9 (2014), pp. 935–937. ISSN: 1548-7105. DOI: 10.1038/nmeth.3046. URL: https://doi.org/10.1038/nmeth.3046.

[13] Thomas A. Peterson, Emily Doughty, and Maricel G. Kann. "Towards Precision Medicine: Advances in Computational Approaches for the Analysis of Human Variants". In: *Journal of Molecular Biology* 425.21 (2013). Understanding Molecular Effects of Naturally Occurring Genetic Differences, pp. 4047–4063. ISSN: 0022-2836. DOI: https://doi.org/10.1016/j.jmb.2013.08.008. URL: http://www.sciencedirect.com/science/article/pii/S0022283613005111.

[14] Abhishek Niroula and Mauno Vihinen. "Variation Interpretation Predictors: Principles, Types, Performance, and Choice". In: *Human Mutation* 37.6 (2016), pp. 579–597. DOI: 10.1002/humu.22987. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/humu.22987. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/humu.22987.

[15] Karen Eilbeck, Aaron Quinlan, and Mark Yandell. "Settling the score: variant prioritization and Mendelian disease". In: *Nature Reviews Genetics* 18.10 (2017), pp. 599–612. ISSN: 1471-0064. DOI: 10.1038/nrg.2017.52. URL: https://doi.org/10.1038/nrg.2017.52.

[16] Ivan A Adzhubei et al. "A method and server for predicting damaging missense mutations". In: *Nature Methods* 7.4 (2010), pp. 248–249. ISSN: 1548-7105. DOI: 10.1038/nmeth0410-248. URL: https://doi.org/10.1038/nmeth0410-248.

[17] Biao Li et al. "Automated inference of molecular mechanisms of disease from amino acid substitutions". In: *Bioinformatics* 25.21 (Sept. 2009), pp. 2744–2750. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btp528. eprint: https://academic.oup.com/bioinformatics/article-pdf/25/21/2744/6063223/btp528.pdf. URL: https://doi.org/10.1093/bioinformatics/btp528.

[18] Vikas Pejaver et al. "MutPred2: inferring the molecular and phenotypic impact of amino acid variants". In: *bioRxiv* (2017). DOI: 10.1101/134981. eprint: https://www.biorxiv.org/content/early/2017/05/09/134981.full.pdf. URL: https://www.biorxiv.org/content/early/2017/05/09/134981.

[19] Nilah M. Ioannidis et al. "REVEL: An Ensemble Method for Predicting the Pathogenicity of Rare Missense Variants". In: *The American Journal of Human Genetics* 99.4 (2016), pp. 877–885. ISSN: 0002-9297. DOI: https://doi.org/10.1016/j.ajhg.2016.08.016. URL: http://www.sciencedirect.com/science/article/pii/S0002929716303706.

[20] Martin Kircher et al. "A general framework for estimating the relative pathogenicity of human genetic variants". In: *Nature Genetics* 46.3 (2014), pp. 310–315. ISSN: 1546-1718. DOI: 10.1038/ng.2892. URL: https://doi.org/10.1038/ng.2892.

[21] Philipp Rentzsch et al. "CADD: predicting the deleteriousness of variants throughout the human genome". In: *Nucleic Acids Research* 47.D1 (Oct. 2018), pp. D886–D894. ISSN: 0305-1048. DOI: 10.1093/nar/gky1016. eprint: http://oup.prod.sis.lan/nar/article-pdf/47/D1/D886/27436395/gky1016.pdf. URL: https://doi.org/10.1093/nar/gky1016.

[22] Daniel Quang, Yifei Chen, and Xiaohui Xie. "DANN: a deep learning approach for annotating the pathogenicity of genetic variants". In: *Bioinformatics* 31.5 (Oct. 2014), pp. 761–763. ISSN: 1367-4803. DOI: 10.1093/

bioinformatics/btu703. eprint: https://academic.oup.com/bioinformatics/article-pdf/31/5/761/13201318/btu703.pdf. URL: https://doi.org/10.1093/bioinformatics/btu703.

[23] Iuliana Ionita-Laza et al. "A spectral approach integrating functional genomic annotations for coding and noncoding variants". In: *Nature Genetics* 48.2 (2016), pp. 214–220. ISSN: 1546-1718. DOI: 10.1038/ng.3477. URL: https://doi.org/10.1038/ng.3477.

[24] Ayodeji Olatubosun et al. "PON-P: Integrated predictor for pathogenicity of missense variants". In: *Human Mutation* 33.8 (2012), pp. 1166–1174. DOI: 10.1002/humu.22102. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/humu.22102. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/humu.22102.

[25] Chengliang Dong et al. "Comparison and integration of deleteriousness prediction methods for nonsynonymous SNVs in whole exome sequencing studies". In: *Human Molecular Genetics* 24.8 (Dec. 2014), pp. 2125–2137. ISSN: 0964-6906. DOI: 10.1093/hmg/ddu733. eprint: http://oup.prod.sis.lan/hmg/article-pdf/24/8/2125/14147495/ddu733.pdf. URL: https://doi.org/10.1093/hmg/ddu733.

[26] Jinchen Li et al. "Performance evaluation of pathogenicity-computation methods for missense variants". In: *Nucleic Acids Research* 46.15 (July 2018), pp. 7793–7804. ISSN: 0305-1048. DOI: 10.1093/nar/gky678. eprint: https://academic.oup.com/nar/article-pdf/46/15/7793/25690336/gky678.pdf. URL: https://doi.org/10.1093/nar/gky678.

[27] Rajarshi Ghosh, Ninad Oak, and Sharon E. Plon. "Evaluation of in silico algorithms for use with ACMG/AMP clinical variant interpretation guidelines". In: *Genome Biology* 18.1 (2017), p. 225. ISSN: 1474-760X. DOI: 10.1186/s13059-017-1353-5. URL: https://doi.org/10.1186/s13059-017-1353-5.

[28] Konrad J. Karczewski et al. "The mutational constraint spectrum quantified from variation in 141,456 humans". In: *bioRxiv* (2020). Ed. by Carlos A Aguilar Salinas et al. DOI: 10.1101/531210. eprint: https://www.biorxiv.org/content/early/2020/04/08/531210.full.pdf. URL: https://www.biorxiv.org/content/early/2020/04/08/531210.

[29] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2019. URL: https://www.R-project.org/.

[30] Heidi L. Rehm et al. "ClinGen — The Clinical Genome Resource". In: *New England Journal of Medicine* 372.23 (2015). PMID: 26014595, pp. 2235–2242. DOI: 10.1056/NEJMsr1406261. eprint: https://doi.org/10.1056/NEJMsr1406261. URL: https://doi.org/10.1056/NEJMsr1406261.

[31] Melissa J. Landrum et al. "ClinVar: public archive of relationships among sequence variation and human phenotype". In: *Nucleic Acids Research* 42.D1 (Nov. 2013), pp. D980–D985. ISSN: 0305-1048. DOI: 10.1093/nar/gkt1113. eprint: http://oup.prod.sis.lan/nar/article-pdf/42/D1/D980/3584314/gkt1113.pdf. URL: https://doi.org/10.1093/nar/gkt1113.

[32] Melissa J. Landrum et al. “ClinVar: public archive of interpretations of clinically relevant variants”. In: *Nucleic Acids Research* 44.D1 (Nov. 2015), pp. D862–D868. ISSN: 0305-1048. DOI: 10.1093/nar/gkv1222. eprint: http://oup.prod.sis.lan/nar/article-pdf/44/D1/D862/9483060/gkv1222.pdf. URL: https://doi.org/10.1093/nar/gkv1222.

[33] Melissa J. Landrum et al. “ClinVar: improving access to variant interpretations and supporting evidence”. In: *Nucleic Acids Research* 46.D1 (Nov. 2017), pp. D1062–D1067. ISSN: 0305-1048. DOI: 10.1093/nar/gkx1153. eprint: http://oup.prod.sis.lan/nar/article-pdf/46/D1/D1062/23162472/gkx1153.pdf. URL: https://doi.org/10.1093/nar/gkx1153.

[34] Roderick J. A. Little and Donald B. Rubin. *Statistical analysis with missing data*. eng. 2nd ed. Hoboken, New Jersey: John Wiley & Sons, Inc., 2002. ISBN: 9781118625866.

[35] S. van Buuren. *Flexible Imputation of Missing Data*. 2nd ed. Chapman & Hall/CRC Interdisciplinary Statistics. CRC Press LLC, 2018. ISBN: 9781138588318. URL: https://www.crcpress.com/Flexible-Imputation-of-Missing-Data-Second-Edition/Buuren/p/book/9781138588318.

[36] Warren S. Sarle. “Prediction with missing inputs”. In: *3rd, International conference on computational intelligence and neurosciences*. JCIS ’98 proceedings (Research Triangle Park, NC). Ed. by Paul P. Wang. Vol. 2. Association for Intelligent Machinery, 1998, pp. 399–402.

[37] Yufeng Ding and Jeffrey S. Simonoff. “An Investigation of Missing Data Methods for Classification Trees Applied to Binary Response Data”. In: *J. Mach. Learn. Res.* 11 (Mar. 2010), pp. 131–170. ISSN: 1532-4435.

[38] Galit Shmueli. “To Explain or to Predict?” In: *Statist. Sci.* 25.3 (Aug. 2010), pp. 289–310. DOI: 10.1214/10-STS330. URL: https://doi.org/10.1214/10-STS330.

[39] Leo Breiman. “Statistical Modeling: The Two Cultures (with comments and a rejoinder by the author)”. In: *Statistical Science* 16 (Aug. 2001), pp. 199–231. DOI: 10.1214/ss/1009213726.

[40] Benjamin M. Marlin. “Missing data problems in machine learning”. PhD thesis. University of Toronto, 2008.

[41] Leo Breiman. “Random Forests”. In: *Machine Learning* 45.1 (2001), pp. 5–32. ISSN: 1573-0565. DOI: 10.1023/A:1010933404324. URL: https://doi.org/10.1023/A:1010933404324.

[42] Leo Breiman and Adele Cutler. *Random forests - classification description*. URL: https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm (visited on 01/27/2020).

[43] Leo Breiman et al. *Classification and regression trees*. CRC press, 1984.

[44] Hemant Ishwaran et al. “Random survival forests”. In: *Ann. Appl. Stat.* 2.3 (Sept. 2008), pp. 841–860. DOI: 10.1214/08-AOAS169. URL: https://doi.org/10.1214/08-AOAS169.

[45] H. Ishwaran and U.B. Kogalur. *Fast Unified Random Forests for Survival, Regression, and Classification (RF-SRC)*. R package version 2.9.3. manual, 2020. URL: https://cran.r-project.org/package=randomForestSRC.

[46] D. Williams et al. “On Classification with Incomplete Data”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.3 (2007), pp. 427–436.

[47] Maytal Saar-Tsechansky and Foster Provost. “Handling missing values when applying classification models”. In: *Journal of machine learning research* 8.Jul (2007), pp. 1623–1657.

[48] Michael P. Jones. “Indicator and Stratification Methods for Missing Explanatory Variables in Multiple Linear Regression”. In: *Journal of the American Statistical Association* 91.433 (1996), pp. 222–230. ISSN: 01621459. URL: http://www.jstor.org/stable/2291399.

[49] Karthik A Jagadeesh et al. “M-CAP eliminates a majority of variants of uncertain significance in clinical exomes at high sensitivity”. In: *Nature Genetics* 48.12 (2016), pp. 1581–1586. ISSN: 1546-1718. DOI: 10.1038/ng.3703. URL: https://doi.org/10.1038/ng.3703.

[50] Miao-Xin Li et al. “A comprehensive framework for prioritizing variants in exome sequencing studies of Mendelian diseases”. In: *Nucleic Acids Research* 40.7 (Jan. 2012), e53–e53. ISSN: 0305-1048. DOI: 10.1093/nar/gkr1257. eprint: https://academic.oup.com/nar/article-pdf/40/7/e53/25334663/gkr1257.pdf. URL: https://doi.org/10.1093/nar/gkr1257.

[51] Mulin Jun Li et al. “Predicting regulatory variants with composite statistic”. In: *Bioinformatics* 32.18 (June 2016), pp. 2729–2736. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btw288. eprint: https://academic.oup.com/bioinformatics/article-pdf/32/18/2729/24406781/btw288.pdf. URL: https://doi.org/10.1093/bioinformatics/btw288.

[52] William McLaren et al. “The Ensembl Variant Effect Predictor”. In: *Genome Biology* 17.122 (2016). DOI: 10.1186/s13059-016-0974-4.

[53] Xiaoming Liu, Xueqiu Jian, and Eric Boerwinkle. “dbNSFP: A lightweight database of human nonsynonymous SNPs and their functional predictions”. In: *Human Mutation* 32.8 (2011), pp. 894–899. DOI: 10.1002/humu.21517. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/humu.21517. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/humu.21517.

[54] Xiaoming Liu et al. “dbNSFP v3.0: A One-Stop Database of Functional Predictions and Annotations for Human Nonsynonymous and Splice-Site SNVs”. In: *Human Mutation* 37.3 (2016), pp. 235–241. DOI: 10.1002/humu.22932. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/humu.22932. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/humu.22932.

[55] S. Van Buuren et al. “Fully conditional specification in multivariate imputation”. In: *Journal of Statistical Computation and Simulation* 76.12 (2006), pp. 1049–1064. DOI: 10.1080/10629360600810434. eprint: https://doi.org/10.1080/10629360600810434. URL: https://doi.org/10.1080/10629360600810434.

[56] Stef van Buuren. “Multiple imputation of discrete and continuous data by fully conditional specification”. In: *Statistical Methods in Medical Research* 16.3 (2007). PMID: 17621469, pp. 219–242. DOI: 10.1177/0962280206074463. eprint: https://doi.org/10.1177/0962280206074463. URL: https://doi.org/10.1177/0962280206074463.

[57] Stef van Buuren and Karin Groothuis-Oudshoorn. "mice: Multivariate Imputation by Chained Equations in R". In: *Journal of Statistical Software, Articles* 45.3 (2011), pp. 1–67. ISSN: 1548-7660. DOI: 10.18637/jss.v045.i03. URL: https://www.jstatsoft.org/v045/i03.

[58] Shigeyuki Oba et al. "A Bayesian missing value estimation method for gene expression profile data". In: *Bioinformatics* 19.16 (Nov. 2003), pp. 2088–2096. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btg287. eprint: https://academic.oup.com/bioinformatics/article-pdf/19/16/2088/667802/btg287.pdf. URL: https://doi.org/10.1093/bioinformatics/btg287.

[59] Wolfram Stacklies et al. "pcaMethods—a bioconductor package providing PCA methods for incomplete data". In: *Bioinformatics* 23.9 (Mar. 2007). R package version 1.78.0, pp. 1164–1167. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btm069. eprint: https://academic.oup.com/bioinformatics/article-pdf/23/9/1164/761822/btm069.pdf. URL: https://doi.org/10.1093/bioinformatics/btm069.

[60] L. Torgo. *Data Mining with R, learning with case studies*. R package version 0.4.1. Chapman and Hall/CRC, 2010. URL: http://www.dcc.fc.up.pt/~ltorgo/DataMiningWithR.

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

[62] Daniel J. Stekhoven. *missForest: Nonparametric Missing Value Imputation using Random Forest*. R package version 1.4. 2013.

[63] Emidio Capriotti, Russ B. Altman, and Yana Bromberg. "Collective judgment predicts disease-associated single nucleotide variants". In: *BMC Genomics* 14.3 (2013), S2. ISSN: 1471-2164. DOI: 10.1186/1471-2164-14-S3-S2. URL: https://doi.org/10.1186/1471-2164-14-S3-S2.

[64] Andy Liaw and Matthew Wiener. "Classification and Regression by randomForest". In: *R News* 2.3 (2002), pp. 18–22. URL: https://CRAN.R-project.org/doc/Rnews/.

[65] Max Kuhn. "Building Predictive Models in R Using the caret Package". In: *Journal of Statistical Software, Articles* 28.5 (2008), pp. 1–26. ISSN: 1548-7660. DOI: 10.18637/jss.v028.i05. URL: https://www.jstatsoft.org/v028/i05.

[66] Davide Chicco and Giuseppe Jurman. "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation". In: *BMC Genomics* 21.1 (2020), p. 6. ISSN: 1471-2164. DOI: 10.1186/s12864-019-6413-7. URL: https://doi.org/10.1186/s12864-019-6413-7.

[67] Rianne Margaretha Schouten, Peter Lugtig, and Gerko Vink. "Generating missing values for simulation purposes: a multivariate amputation procedure". In: *Journal of Statistical Computation and Simulation* 88.15 (2018), pp. 2909–2930. DOI: 10.1080/00949655.2018.1491577. eprint: https://doi.org/10.1080/00949655.2018.1491577. URL: https://doi.org/10.1080/00949655.2018.1491577.

[68] Bernd Bischl et al. “mlr: Machine Learning in R”. In: *Journal of Machine Learning Research* 17.170 (2016), pp. 1–5. URL: http://jmlr.org/papers/v17/15-066.html.

[69] Brian J. Knaus and Niklaus J. Grünwald. “VCFR: a package to manipulate and visualize variant call format data in R”. In: *Molecular Ecology Resources* 17.1 (2017). R package version 1.8.0, pp. 44–53. ISSN: 757. URL: http://dx.doi.org/10.1111/1755-0998.12549.

[70] Brian Lee Yung Rowe. *futile.logger: A Logging Utility for R*. R package version 1.4.3. 2016. URL: https://CRAN.R-project.org/package=futile.logger.

[71] Hadley Wickham and Lionel Henry. *tidyr: Easily Tidy Data with 'spread()' and 'gather()' Functions*. R package version 0.8.3. 2019. URL: https://CRAN.R-project.org/package=tidyr.

[72] Kirill Müller. *here: A Simpler Way to Find Your Files*. R package version 0.1. 2017. URL: https://CRAN.R-project.org/package=here.

[73] Stefan Milton Bache and Hadley Wickham. *magrittr: A Forward-Pipe Operator for R*. R package version 1.5. 2014. URL: https://CRAN.R-project.org/package=magrittr.

[74] Alboukadel Kassambara. *ggcorrplot: Visualization of a Correlation Matrix using 'ggplot2'*. R package version 0.1.3. 2019. URL: https://CRAN.R-project.org/package=ggcorrplot.

[75] Microsoft and Steve Weston. *foreach: Provides Foreach Looping Construct for R*. R package version 1.4.4. 2017. URL: https://CRAN.R-project.org/package=foreach.

[76] Microsoft Corporation and Steve Weston. *doParallel: Foreach Parallel Adaptor for the 'parallel' Package*. R package version 1.0.14. 2018. URL: https://CRAN.R-project.org/package=doParallel.

[77] Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. R package version 3.2.0. Springer-Verlag New York, 2016. ISBN: 978-3-319-24277-4. URL: https://ggplot2.tidyverse.org.

[78] Revolution Analytics and Steve Weston. *iterators: Provides Iterator Construct for R*. R package version 1.0.10. 2018. URL: https://CRAN.R-project.org/package=iterators.

[79] Renaud Gaujoux. *doRNG: Generic Reproducible Parallel Backend for 'foreach' Loops*. R package version 1.7.1. 2018. URL: https://CRAN.R-project.org/package=doRNG.

[80] Renaud Gaujoux. *rngtools: Utility Functions for Working with Random Number Generators*. R package version 1.4. 2019. URL: https://CRAN.R-project.org/package=rngtools.

[81] Deepayan Sarkar. *Lattice: Multivariate Data Visualization with R*. R package version 0.20-38. New York: Springer, 2008. ISBN: ISBN 978-0-387-75968-5. URL: http://lmdvr.r-forge.r-project.org.

[82] Steve Weston and Hadley Wickham. *itertools: Iterator Tools*. R package version 0.1-3. 2014. URL: https://CRAN.R-project.org/package=itertools.

[83] Tyler Hunt. *ModelMetrics: Rapid Calculation of Model Metrics*. R package version 1.2.2. 2018. URL: https://CRAN.R-project.org/package=ModelMetrics.

[84] Hadley Wickham. *stringr: Simple, Consistent Wrappers for Common String Operations*. R package version 1.4.0. 2019. URL: https://CRAN.R-project.org/package=stringr.

[85] Baptiste Auguie. *gridExtra: Miscellaneous Functions for "Grid" Graphics*. R package version 2.3. 2017. URL: https://CRAN.R-project.org/package=gridExtra.

[86] Dirk Eddelbuettel with contributions by Antoine Lucas et al. *digest: Create Compact Hash Digests of R Objects*. R package version 0.6.20. 2019. URL: https://CRAN.R-project.org/package=digest.

[87] Lionel Henry and Hadley Wickham. *purrr: Functional Programming Tools*. R package version 0.3.2. 2019. URL: https://CRAN.R-project.org/package=purrr.

[88] Max Kuhn. Contributions from Jed Wing et al. *caret: Classification and Regression Training*. R package version 6.0-84. 2019. URL: https://CRAN.R-project.org/package=caret.

[89] Hadley Wickham. "testthat: Get Started with Testing". In: *The R Journal* 3 (2011). R package version 2.1.1, pp. 5–10. URL: https://journal.r-project.org/archive/2011-1/RJournal_2011-1_Wickham.pdf.

[90] David Meyer et al. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*. R package version 1.7-2. 2019. URL: https://CRAN.R-project.org/package=e1071.

[91] W. Huber et al. "Orchestrating high-throughput genomic analysis with Bioconductor". In: *Nature Methods* 12.2 (2015), pp. 115–121. URL: http://www.nature.com/nmeth/journal/v12/n2/full/nmeth.3252.html.

[92] Martin Morgan. *BiocManager: Access the Bioconductor Project Package Repository*. R package version 1.30.4. 2018. URL: https://CRAN.R-project.org/package=BiocManager.

[93] Jason Poulos and Rafael Valle. "Missing Data Imputation for Supervised Learning". In: *Applied Artificial Intelligence* 32.2 (2018), pp. 186–196. DOI: 10.1080/08839514.2018.1448143. eprint: https://doi.org/10.1080/08839514.2018.1448143. URL: https://doi.org/10.1080/08839514.2018.1448143.

[94] Dominik G. Grimm et al. "The Evaluation of Tools Used to Predict the Impact of Missense Variants Is Hindered by Two Types of Circularity". In: *Human Mutation* 36.5 (2015), pp. 513–523. DOI: 10.1002/humu.22768. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/humu.22768. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/humu.22768.

[95] Khalid Mahmood et al. "Variant effect prediction tools assessed using independent, functional assay-based datasets: implications for discovery and diagnostics". In: *Human Genomics* 11.1 (2017), p. 10. ISSN: 1479-7364. DOI: 10.1186/s40246-017-0104-8. URL: https://doi.org/10.1186/s40246-017-0104-8.