Women in Data Science Datathon 2022 Series:

The “missing” missing value

Shu-Ting Pi, PhD
Applied Scientist @ Amazon

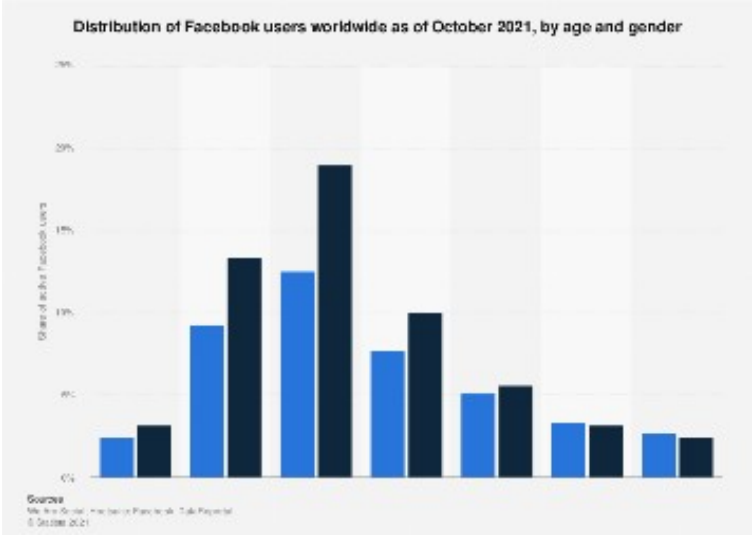
Outline

- Understanding missing value
  - types, origins and impacts of missing values
- Toward MV: Deletion
- Toward MV: Single imputation
- Toward MV: Multiple imputation
- Toward MV: No imputation
- The “good” missing values: A Kaggle Competition
- Conclusion

Understanding missing value

Why Study Missing Value?

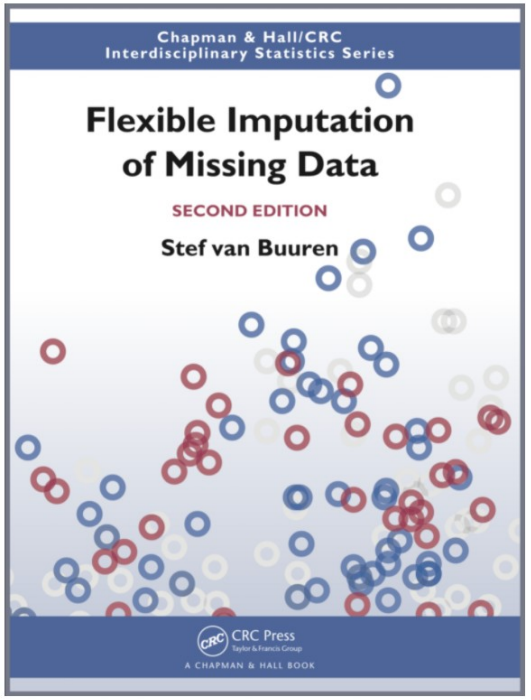
- If you want to develop a machine learning model, missing values may be a minor issue (impute or delete are good for low missing rate cases).
- If you want to perform data analysis, it is critical since it gives you the wrong distribution on features.



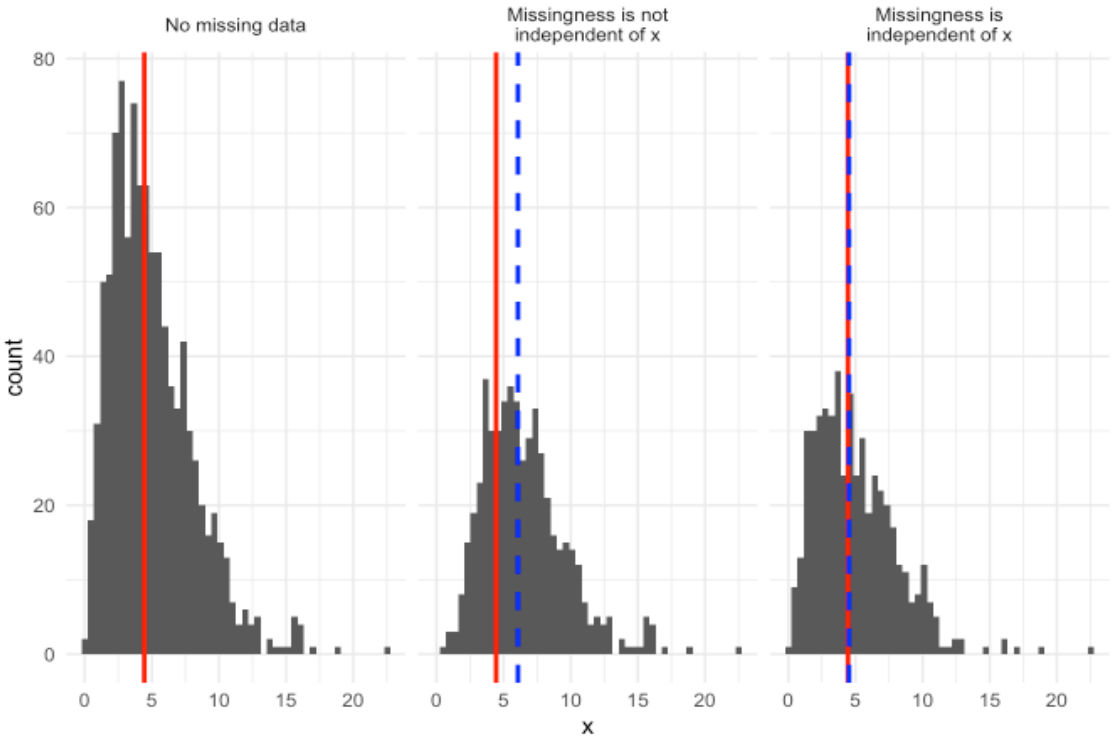
What is missing value?

	name	toy	born
0	Alfred	fdsfa	NaT
1	Batman	Batmobile	1940-04-25
2	Catwoman	Bullwhip	NaT

complete incomplete



Question: What is your annual income?



Missing value could result in bias !
(e.g. low incomes are more likely to reject)

Skewed data !

Types of missing value

- Missing Completely At Random (MCAR):

$$p(y_{obs} = na|y, x) = c$$

- Missing at Random (MAR):

$$p(y_{obs} = na|y, x) = f(x)$$

- Missing not at Random(MNAR):

$$p(y_{obs} = na|y, x) = f(x, y)$$

MCAR: deletion will not result in bias (unrealistic)
MAR: missingness can be predicted from other features (
MNAR: missingness is related to what is missing !

Types of missing value

- Do you support higher hourly rate?
  - MACR: responses are missing due to processing issue.
  - MAR: higher income tends to have missing value
  - MNAR: Silent Spiral, e.g.: Yes are more likely to hide their thoughts since their opinion “looks” weaker.
- In case of MNAR, we use MAR to approximate it ! (Taylor expansion, feature correlations)

Toward MV: deletion

Listwise deletion

Mobile ID	Mobile Package	Download Speed	Data Limit Usage	
1	Fast+	157	80%	
2	Lite	99	70%	
3	Fast+	167	10%	
4	Fast+	N/A	80%	← Delete
5	Lite	76	70%	
6	Fast+	155	10%	
7	N/A	N/A	95%	← Delete
8	Lite	76	77%	
9	Fast+	180	N/A	← Delete

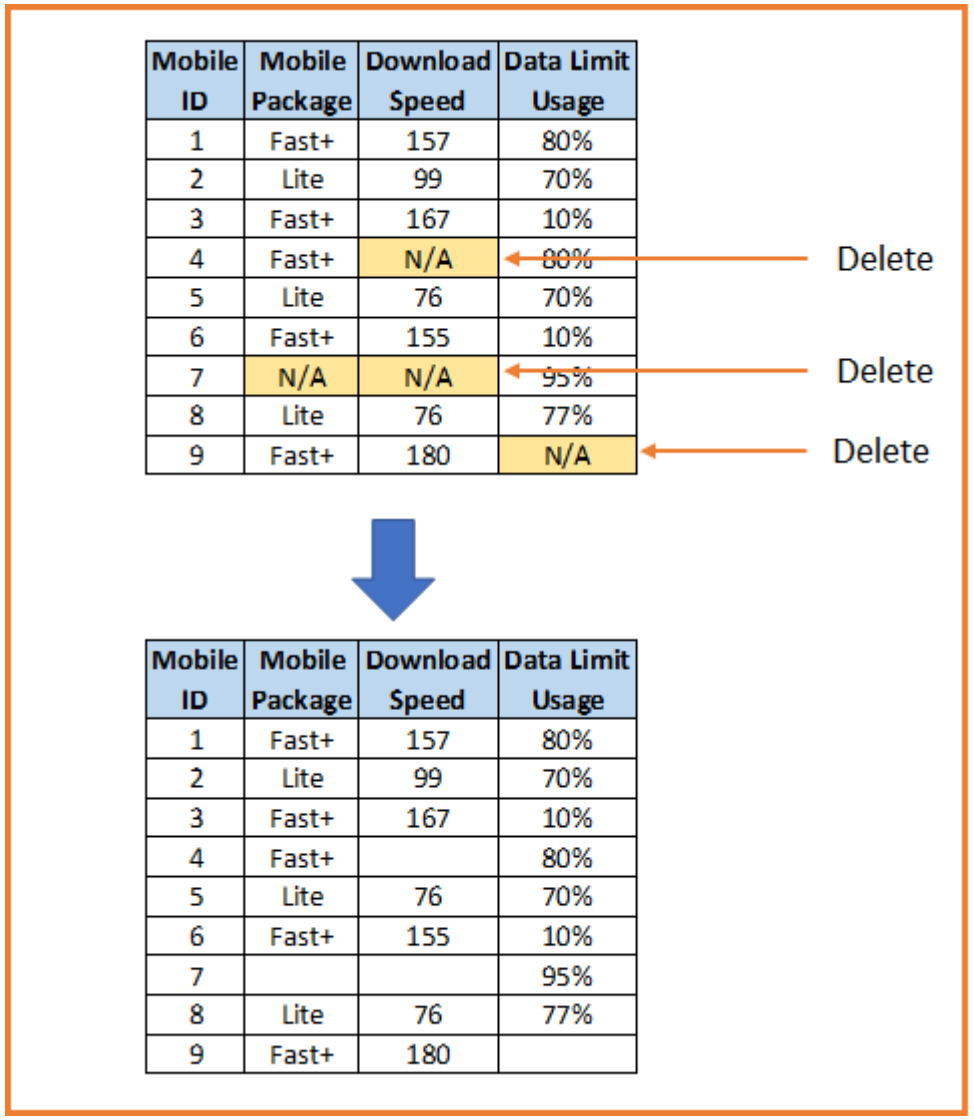
Mobile ID	Mobile Package	Download Speed	Data Limit Usage
1	Fast+	157	80%
2	Lite	99	70%
3	Fast+	167	10%
5	Lite	76	70%
6	Fast+	155	10%
8	Lite	76	77%

Procedure:
1). Delete all cases w/ missing values:

Pros:
1. Super simple ! (most popular)
2. No bias when MACR

Cons:
1. has bias when not MACR
2. Will decrease statistical power

Pairwise deletion



Procedure:

1). When performing an analysis without the need of features with missing values, use all cases.

2) When performing an analysis that requires features with missing values, delete missing cases.

Pro:

- Use more data, more statistical power
- Very simple
- Unbiased on complete variables

Cons:

- bias on some analysis
- unreasonable results on pairwise analysis such as correlation matrix.

Toward MV: Single Imputation

Mean / Median / Mode imputation

	A
0	2.0
1	2.0
2	NaN
3	NaN
4	NaN
5	8.0

mean

	A
0	2.0
1	2.0
2	4.0
3	4.0
4	4.0
5	8.0

	A	B
0	2	4.0
1	3	9.0
2	4	NaN
3	5	25.0

median

	A	B
0	2.0	4.0
1	3.0	9.0
2	4.0	9.0
3	5.0	25.0

16!

Make
Ford
Ford
Fiat
BMW
Ford
Kia
Fiat
Ford
Kia

Mode = Ford

Price
Ford
Ford
Fiat
BMW
Ford
Kia
Ford
Fiat
Ford
Ford
Kia

Pro: simple, quick and low cost
works well when low percentage missing

Con: strong bias when high percentage missing

Hot Deck Imputation

Idea: Search for the “most similar” data point (donor) to impute missing value.

Pro: Simple and straightforward
Con: How to define “similarity”

Affinity Score:
(categorical features)

$$\alpha_{ij} = \frac{k - q_i - z_{ij}}{k - q_i}$$

where k is number of variables, q_i is number of missing values in i-th case, z_{ij} is number of variables for which the potential donor j and the recipient i have different values.

Affinity Score:
(continous features)

$$\alpha_{ij} = \frac{k - q_i - \sum_{p=1}^{k} [|x_{ip} - x_{jp}| > \varepsilon_p]}{k - q_i}$$

Or: rank the values of feature with largest correlation, …, etc.

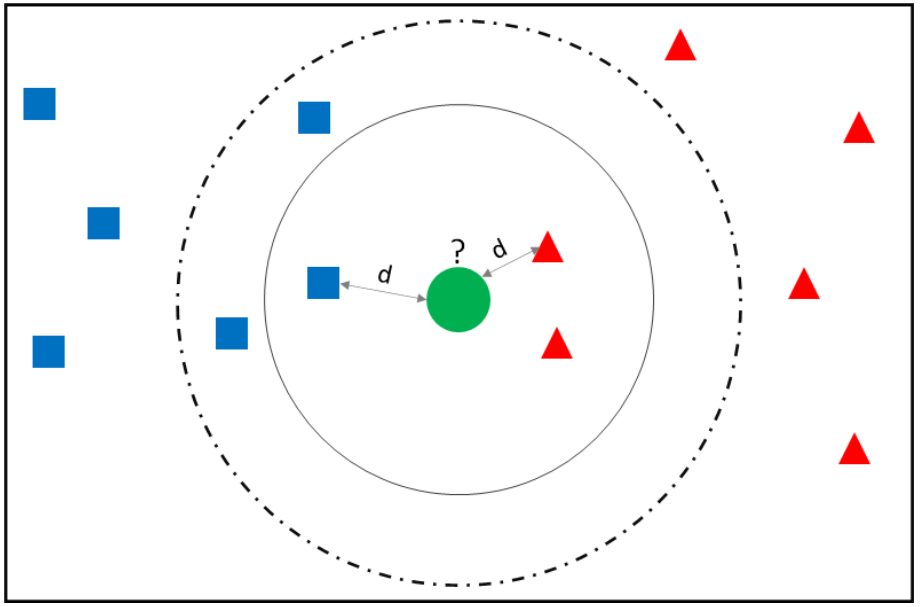
KNN imputation

sklearn.impute.KNNImputer

class sklearn.impute.KNNImputer(*, *missing_values=nan, n_neighbors=5, weights='uniform', metric='nan_euclidean', copy=True, add_indicator=False*) [source]

Imputation for completing missing values using k-Nearest Neighbors.

Each sample's missing values are imputed using the mean value from `n_neighbors` nearest neighbors found in the training set. Two samples are close if the features that neither is missing are close.



The default distance measure is a Euclidean distance measure that is NaN aware, e.g. will not include NaN values when calculating the distance between members of the training dataset.
(You can also use other distances such as cosine, Hamming, …, etc. but it must include NaN aware).

KNN is notorious for poor performance in high dimensional data.

Non-Negative Matrix Factorization

$$X = UV^T \begin{cases} U \in (m \times k) \\ V \in (m \times k) \end{cases}$$

Fill missing values: $X \simeq \widetilde{X} = UV^T$

Loss function: $J = \left\| X - \widetilde{X} \right\|^2 = \left\| X - UV^T \right\|^2 = \sum_{i,j,x_{ij} \neq Nan} (x_{ij} - \sum_k u_{ik} v_{jk})^2$

Keep this method in mind, it is the foundation of multiple imputation denosing autoencoder (MIDA). (This is essential a single layer MIDA)

Toward MV: Multiple Imputation

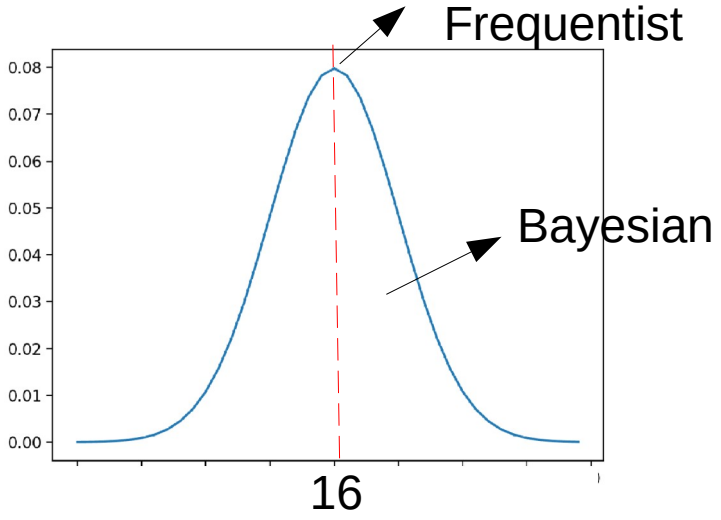
What is Multiple Imputation?

Frequentist: Missing values are well defined numbers.

Bayesian: Missing values are “distributions”.

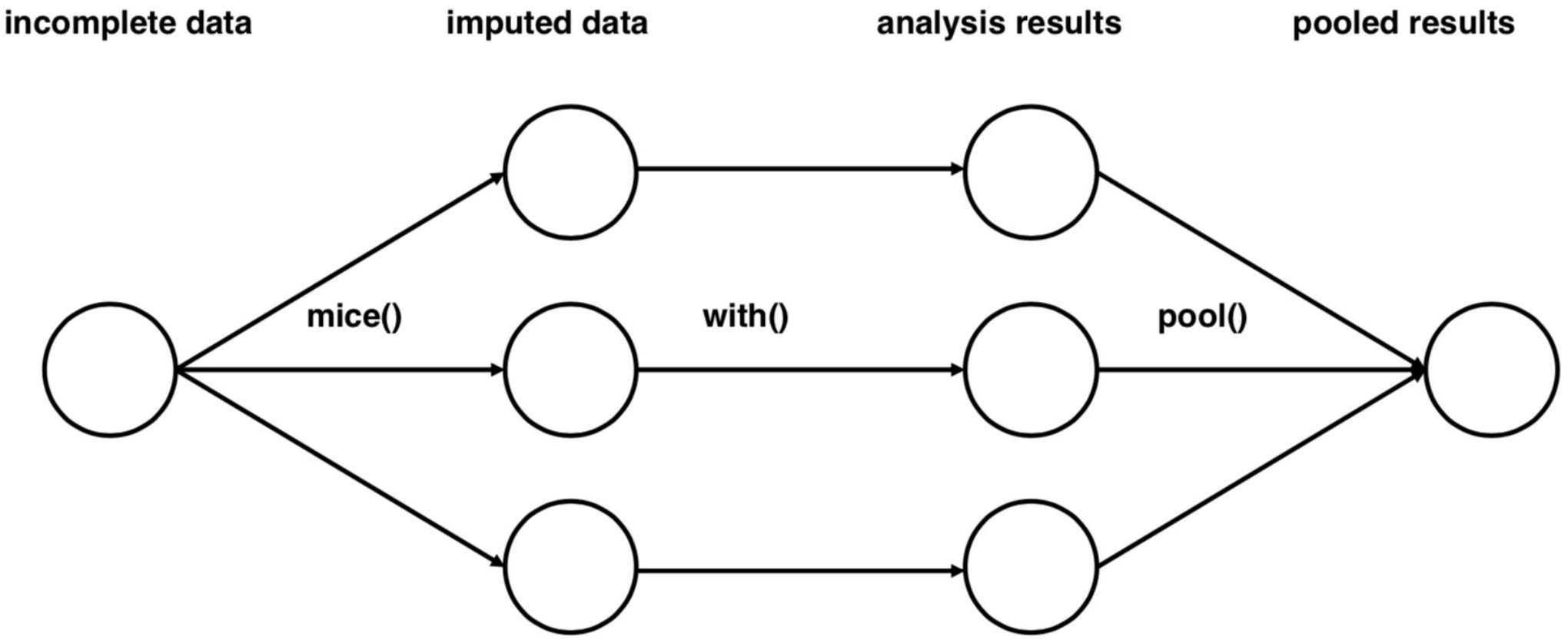
	A	B
0	2	4.0
1	3	9.0
2	4	NaN
3	5	25.0

	A	B
0	2.0	4.0
1	3.0	9.0
2	4.0	16
3	5.0	25.0



Single imputation may be dangerous because we have no idea about uncertainty !

Workflow of Multiple Imputation

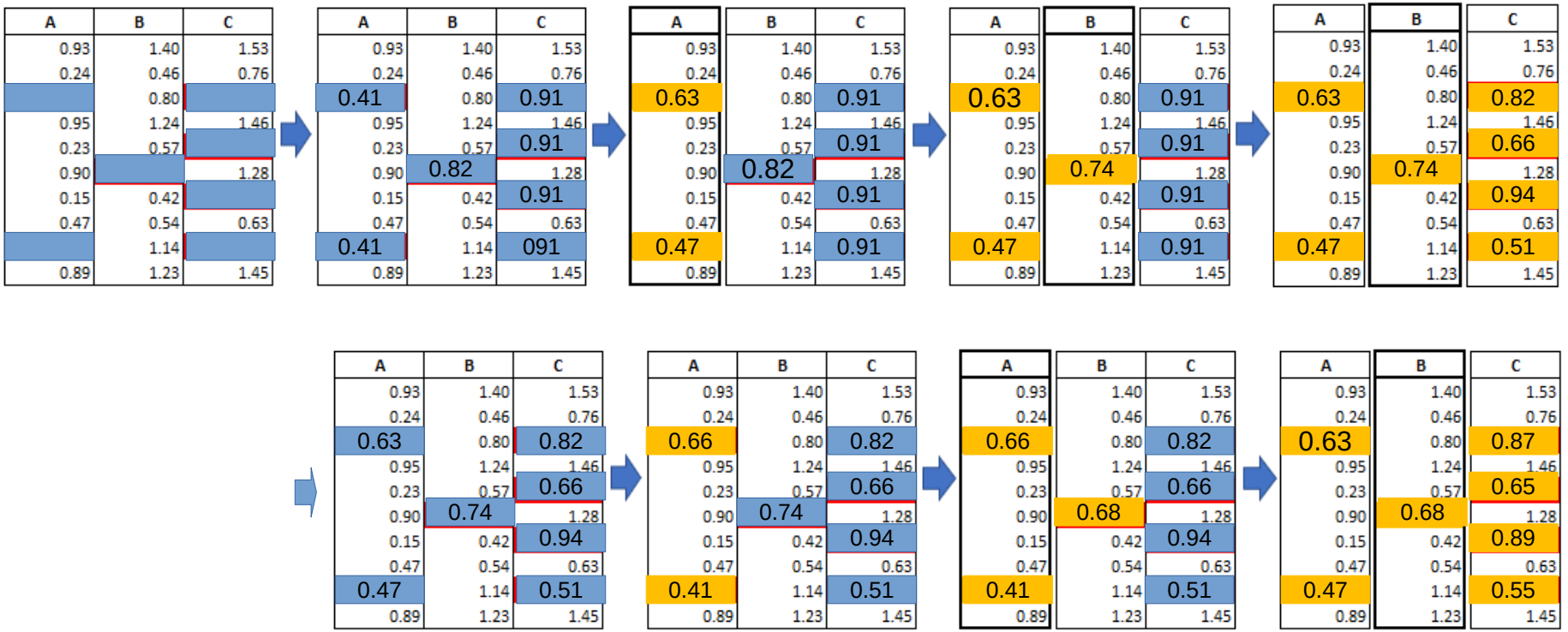


Imputation algorithms must carry uncertainty / Randomness / Variation !

Multiple Imputation => Imputation Algorithms that introduce variation spontaneously !

MICE: Multiple Imputation by Chained Equations (iterative imputation)

MICE = iterative imputation + predictive mean matching



… Repeat the cycle until self-consistency is reached or assigned number of cycles.

Usually Random Forest or ExtraTree were adopted (missForest)

Stef van Buuren, Karin Groothuis-Oudshoorn (2011). “mice: Multivariate Imputation by Chained Equations in R”. Journal of Statistical Software 45: 1-67.

MICE: Multiple Imputation by Chained Equations (predictive mean matching)

The predicted value of A (E[A|B,C]) is shown to the left. We are interested in imputing the bold missing value below

E[A|B,C]	A	B	C
0.73	0.93	1.40	1.53
0.62	0.24	0.46	0.76
0.60		0.80	1.53
1.39	0.95	1.24	1.46
0.36	0.23	0.57	1.28
1.27	0.90	0.46	1.28
0.15	0.15	0.42	1.53
0.65	0.47	0.54	0.63
1.20		1.14	1.28
1.24	0.89	1.23	1.45

Our predicted value for the first missing sample is 0.60. The closest predicted value is 0.62. We find the closest values for all of our missing samples.

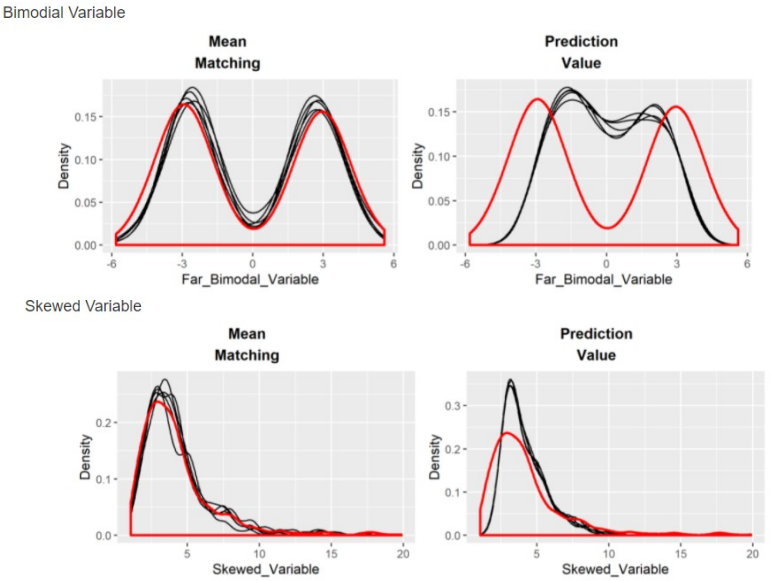
E[A|B,C]	A	B	C
0.73	0.93	1.40	1.53
0.62	0.24	0.46	0.76
0.60		0.80	1.53
1.39	0.95	1.24	1.46
0.36	0.23	0.57	1.28
1.27	0.90	0.46	1.28
0.15	0.15	0.42	1.53
0.65	0.47	0.54	0.63
1.20		1.14	1.28
1.24	0.89	1.23	1.45

We then impute the value corresponding to the original data.

E[A|B,C]	A	B	C
0.73	0.93	1.40	1.53
0.62	0.24	0.46	0.76
0.60	0.24	0.80	1.53
1.39	0.95	1.24	1.46
0.36	0.23	0.57	1.28
1.27	0.90	0.46	1.28
0.15	0.15	0.42	1.53
0.65	0.47	0.54	0.63
1.20	0.89	1.14	1.28
1.24	0.89	1.23	1.45

About PMM:

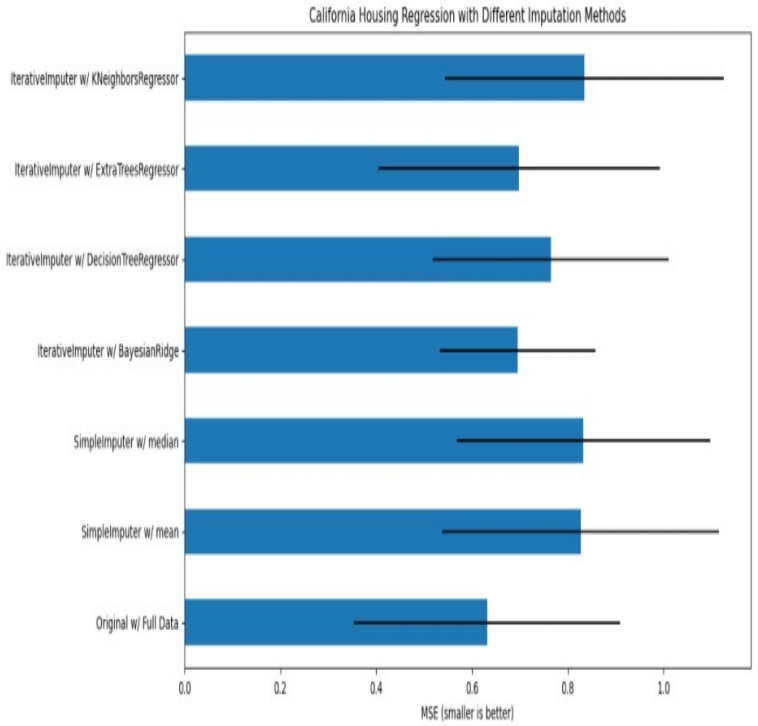
- PMM assumes the distribution of the missing cell is the same as the observed data of the candidate donors.
- PMM is a form of hot-deck
- Imputed values are realistic
- Prevent from unreasonable imputed values
- PMM are useful for 1). multimodal 2). skewed 3). integer

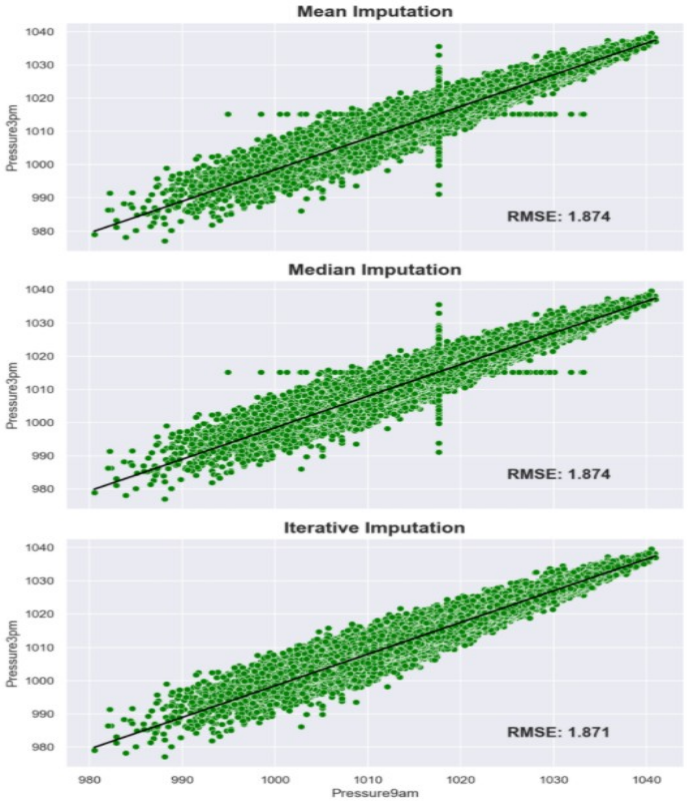


MICE: Multiple Imputation by Chained Equations

sklearn.impute.IterativeImputer

class sklearn.impute.IterativeImputer(*estimator=None, *, missing_values=nan, sample_posterior=False, max_iter=10, tol=0.001, n_nearest_features=None, initial_strategy='mean', imputation_order='ascending', skip_complete=False, min_value=- inf, max_value=inf, verbose=0, random_state=None, add_indicator=False*) [source]





	A	B
0	2	4.0
1	3	9.0
2	4	NaN
3	5	25.0

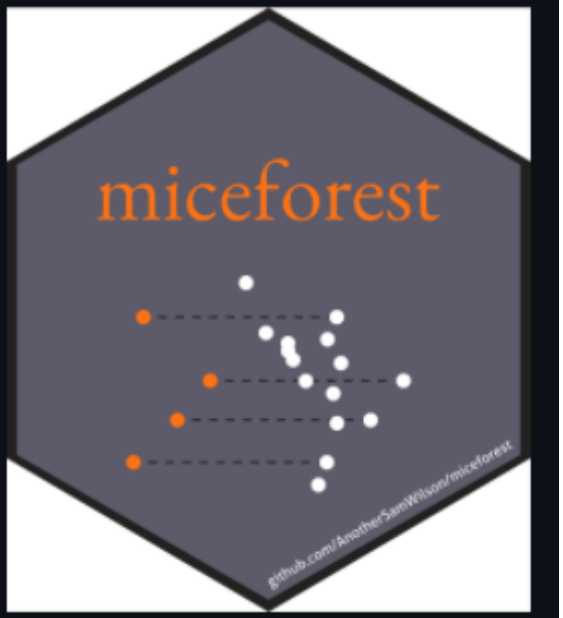
	A	B
0	2.0	4.0
1	3.0	9.0
2	4.0	17.4
3	5.0	25.0

Mean: 12.7
Median: 9

Useful Tool: miceForest

miceforest: Fast, Memory Efficient Imputation with lightgbm

Fast, memory efficient Multiple Imputation by Chained Equations (MICE) with lightgbm. The R version of this package may be found here.



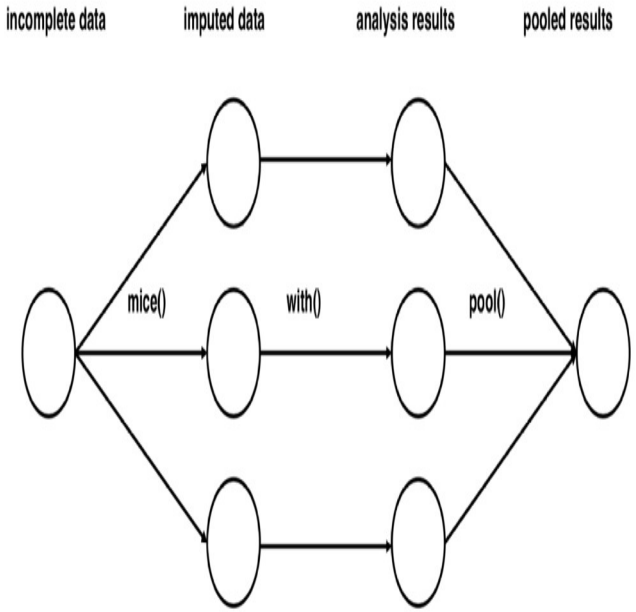
`miceforest` was designed to be:

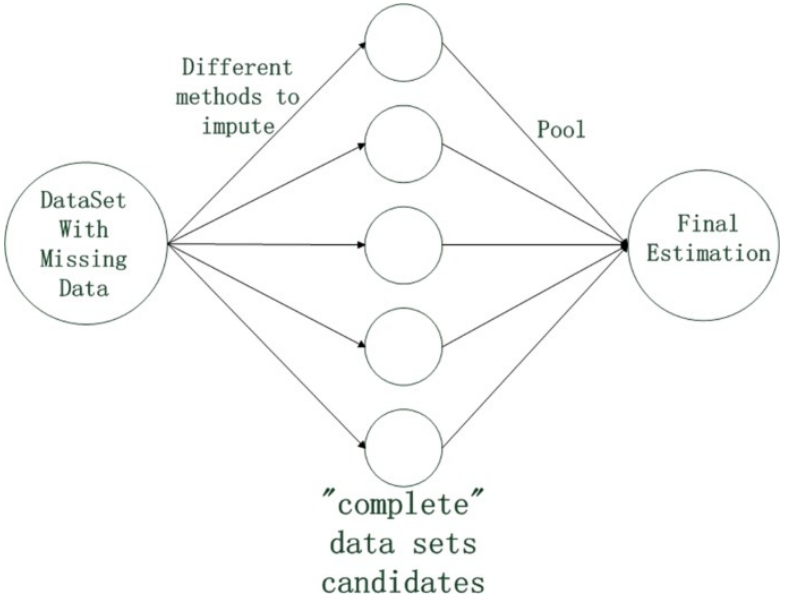
- **Fast** Uses lightgbm as a backend, and has efficient mean matching solutions.
- **Memory Efficient** Capable of performing multiple imputation without copying the dataset. If the dataset can fit in memory, it can (probably) be imputed.
- **Flexible** Can handle pandas DataFrames and numpy arrays. The imputation process can be completely customized. Can handle categorical data automatically.
- **Used In Production** Kernels can be saved and impute new, unseen datasets. Imputing new data is often orders of magnitude faster than including the new data in a new `mice` procedure. Imputation models can be built off of a kernel dataset, even if there are no missing values. New data can also be imputed in place.

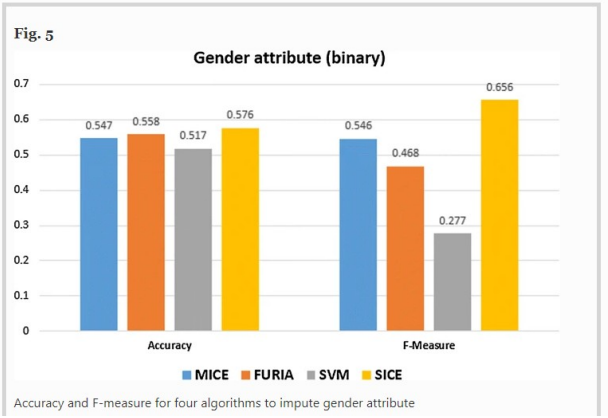
This document contains a thorough walkthrough of the package, benchmarks, and an introduction to multiple imputation. More information on MICE can be found in Stef van Buuren's excellent online book, which you can find here.

SICE: Single Center Imputation from Multiple Chained Equation

SICE:
If you still want a single imputation with better performance, take imputation ensemble !







- **Don't do analysis on multiple complete datasets.**
- **Use the "average" for continuous values (m=7).**
- **Use "mode" for categorical values to impute the miss values (m=7).**
- **It forms a single imputation method that outperforms most algorithms. (other algorithms use the "best" result only. SICE use average / mode instead.)**

Shahidul Islam Khan & Abu Sayed Md Latiful Hoque , Journal of Big Data volume 7, Article number: 37 (2020)

MIDAS: Multiple Imputation of

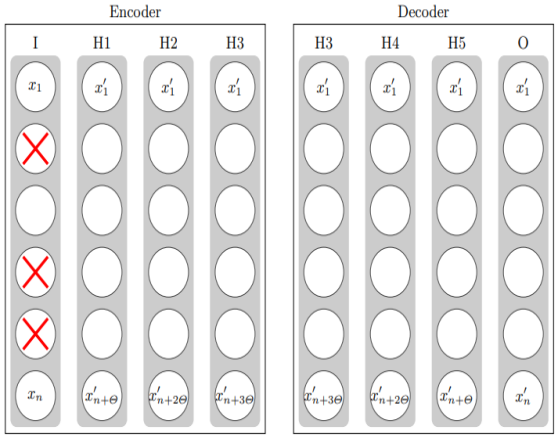


Fig. 1: Our basic architecture, encoder block increases dimensionality at every hidden layer by adding Θ units with decoder symmetrically scaling it back to original dimensions. Crossed out inputs represent stochastic corruption at input by setting random inputs to zero. H_1, H_2, H_3, H_4, H_5 are hidden layers with I and O being the input and output layers respectively. Encoder and decoder are constructed using fully connected artificial neural networks.

	Observations	Attributes
Boston Housing (BH)	506	14
Breast Cancer (BC)	699	11
DNA (DN)	3186	180
Glass (GL)	214	10
House votes (HV)	435	17
Ionosphere (IS)	351	35
Ozone (ON)	366	13
Satellite (SL)	6435	37
Servo (SR)	167	5
Shuttle (ST)	58000	9
Sonar (SN)	208	61
Soybean (SB)	683	36
Vehicle (VC)	846	19
Vowel (VW)	990	10
Zoo (ZO)	101	17

	Data	Uniform missingness		Random missingness	
		DAE	MICE	DAE	MICE
MCAR	BH	2.9(2.9,3)	3.7(3.5,3.8)	0.9(0.9,1)	0.9(0.7,1)
	BC	2.9(2.9,2.9)	3.9(3.6,4.2)	1.2(1.2,1.3)	1.3(1.1,1.4)
	DN	25.7(25.7,25.7)	36.5(36.3,36.6)	13.1(13.1,13.2)	16.9(16.9,17)
	GL	1.1(1,1.1)	1.5(1.3,1.7)	1.3(1.2,1.4)	1.4(1.3,1.6)
	HV	2.4(2.4,2.4)	3.4(3.1,3.7)	1.1(1.1,1.2)	1.2(0.9,1.3)
	IS	13(12.9,13.1)	17.1(16.2,17.7)	5.8(5.6,6.2)	7(6.7,7.5)
	ON	2.1(2.1,2.1)	3.1(3,3.3)	0.9(0.9,1)	1(1,1.2)
	SL	3.6(3.6,3.7)	4.5(4.4,4.6)	1.8(1.7,1.8)	0.7(0.7,0.7)
	SR	1.2(1.,1.2)	1.5(1.4,1.7)	0.4(0.4,0.5)	0.4(0.4,0.5)
	ST	16.5(16.5,16.7)	27.9(27.5,28.2)	6.5(6.4,6.7)	13(12.5,13.8)
	SN	5.1(5,5.1)	7.3(7.2,7.5)	2.3(2.2,2.3)	3.2(3.2,3.3)
	SB	1.8(1.8,1.8)	2.4(2.3,2.4)	1.2(1.1,1.2)	1.1(1,1.1)
	VC	4.1(4,4.1)	5.6(5.5,5.7)	1.6(1.6,1.6)	2.2(2.1,2.3)
	VW	5.8(5.7,6.2)	7.7(7,8.1)	2.6(2.4,2.9)	3.8(3.3,4.2)
	ZO	2.1(2.1,2.1)	3.4(3.1,4.3)	1.1(1.1,1.2)	1.1(1.1,1.1)
MNAR	BH	2.3(2.2,2.4)	3.2(2.9,3.4)	0.9(0.8,1)	0.7(0.7,0.8)
	BC	2.9(2.8,3)	3.6(3.4,3.8)	1.7(1.7,1.8)	1.4(1.3,1.5)
	DN	25.3(25.2,25.3)	34.5(34.5,34.7)	5.7(5.7,5.8)	7.2(7.1,7.2)
	GL	1.3(1.3,1.4)	1.5(1.3,1.8)	0.4(0.3,0.4)	0.2(0.11,0.2)
	HV	2.6(2.6,2.6)	3.5(3.3,3.7)	1.3(1.2,1.3)	1.3(1.3,1.4)
	IS	11.7(11.5,11.8)	15.4(14.9,16.5)	4.8(4.5,5.1)	6.3(5.6,6.8)
	ON	1.5(1.5,1.5)	2.2(2,2.4)	1.2(1.1,1.2)	1.3(1.1,1.5)
	SL	3.4(3.4,3.4)	3.8(3.8,3.9)	1.6(1.6,1.6)	0.5(0.5,0.5)
	SR	1.2(1.2,1.2)	1.6(1.5,1.7)	0.4(0.3,0.4)	0.3(0.2,0.3)
	ST	11.8(11.7,11.9)	22.4(22.1,22.7)	4.5(4.3,4.7)	9.5(8.4,10.3)
	SN	4.6(4.6,4.6)	6.8(6.5,7.1)	2.3(2.3,2.4)	3.1(3,3.2)
	SB	1.7(1.7,1.7)	2.3(2.2,2.4)	0.6(0.6,0.6)	0.9(0.9,0.9)
	VC	3.5(3.4,3.7)	4.6(4.4,4.8)	1.7(1.7,1.8)	2.4(2.3,2.4)
	VW	5.9(5.9,5.9)	NA	2.3(2.1,2.5)	NA
	ZO	3.3(2.8,5.5)	3.9(3.6,4.6)	0.9(0.8,1.0)	1.1(0.7,1.7)

- Use mean or mode as missing values placeholder
- Use Denoising AE with dropout to recover
- on one assumption: enough complete data to train the model, so the model learns to recover true data and is not learning to map placeholders as valid imputations.
- When prediction, keep dropout alive (**Monte Carlo integration approximation of Bayesian Network !**)
- Similar idea has been applied to impute image or text data

Lovedeep Gondara, Ke Wang; MIDA: Multiple Imputation using Denoising Autoencoders; arXiv:1705.02737

Toward MV: No Imputation

Sparsity Aware

XGBoost: A Scalable Tree Boosting System

Tianqi Chen
University of Washington
tqchen@cs.washington.edu

Carlos Guestrin
University of Washington
guestrin@cs.washington.edu

Algorithm 3: Sparsity-aware Split Finding

Input: I, instance set of current node
Input: $I_k = \{i \in I | x_{ik} \neq \text{missing}\}$
Input: d, feature dimension
Also applies to the approximate setting, only collect statistics of non-missing entries into buckets
$gain \leftarrow 0$
$G \leftarrow \sum_{i \in I}, g_i, H \leftarrow \sum_{i \in I} h_i$
for $k = 1$ ***to*** m **do**
  // enumerate missing value goto right
  $G_L \leftarrow 0,\ H_L \leftarrow 0$
  for j *in sorted*$(I_k,$ *ascent order by* $\mathbf{x}_{jk})$ **do**
    $G_L \leftarrow G_L + g_j,\ H_L \leftarrow H_L + h_j$
    $G_R \leftarrow G - G_L,\ H_R \leftarrow H - H_L$
    $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  end
  // enumerate missing value goto left
  $G_R \leftarrow 0,\ H_R \leftarrow 0$
  for j *in sorted*$(I_k,$ *descent order by* $\mathbf{x}_{jk})$ **do**
    $G_R \leftarrow G_R + g_j,\ H_R \leftarrow H_R + h_j$
    $G_L \leftarrow G - G_R,\ H_L \leftarrow H - H_R$
    $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  end
end
Output: Split and default directions with max gain

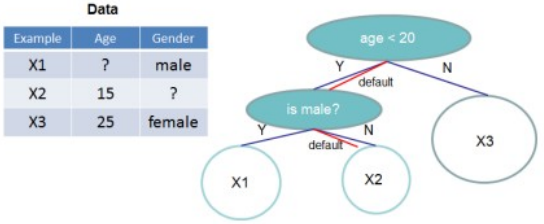


Figure 4: Tree structure with default directions. A example will be classified into the default directio when the feature needed for the split is missing.

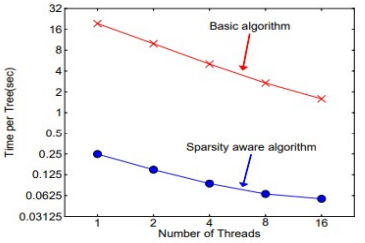


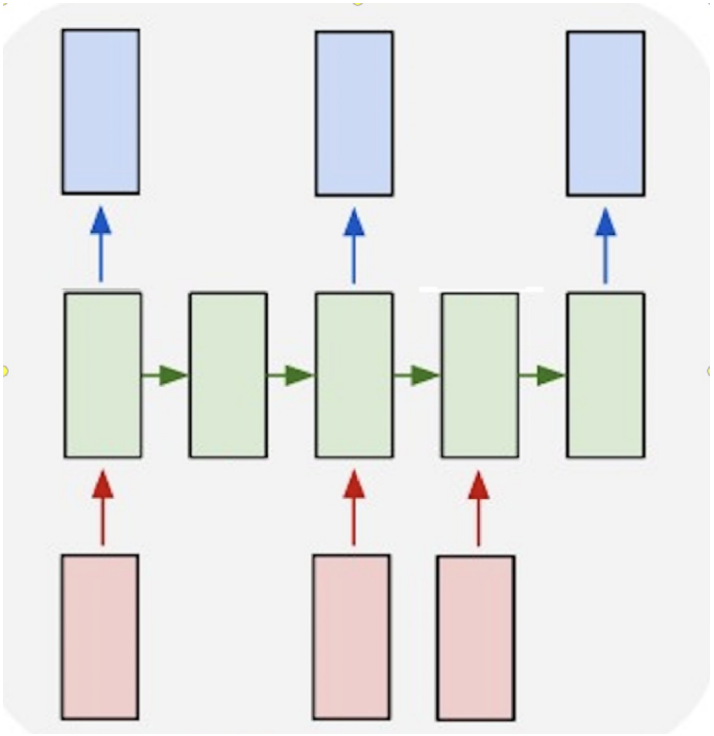
Figure 5: Impact of the sparsity aware algorithm on Allstate-10K. The dataset is sparse mainly due to one-hot encoding. The sparsity aware algorithm is more than 50 times faster than the naive version that does not take sparsity into consideration.

- Examples with missing values will be put to the left and right. The one with max gain will be chosen as default direction.
- LightGBM uses the same algorithm to handle missing values.
- Catboost doesn't include sparsity-aware. Users must specify default direction.

Deep Learning Masking

Deep Learning can learn new representation for missing values !

- Sequence data: Use masking layer
- Continuous : use mean or median
  (don’t use 0 or arbitrary number, it may create discontinuity)
- Categorical: set as a new category.



- If the feature is categorical, it’s safe to create a new category that means “the value is missing.” The model will automatically learn what this implies with respect to the targets.
- If the feature is numerical, avoid inputting an arbitrary value like `"0"`, because it may create a discontinuity in the latent space formed by your features, making it harder for a model trained on it to generalize. Instead, consider replacing the missing value with the average or median value for the feature in the dataset. You could also train a model to predict the feature value given the values of other features.

Francois Chollet, Deep Learning with Python 2ed

Comments on No Imputation

- It’s still unclear whether no-imputation is better or not.
- No-imputation model is convenient but provides no information for statistical analysis.
- Caution: If the training set contains no missing values but real-world may have, create “fake” missing values in your training set ! (e.g. use miceForest.ampute())

- **The “Good” Missing Values**

A Kaggle Competition

27	▲6	No more stressed		0.92283	100	3Y
28	▼1	Belinda Trotta		0.92281	32	3Y
29	▼4	Wearecoming		0.92278	144	3Y
30	▼2	.92496		0.92271	92	3Y
31	▲3	[ods.ai] DmitryS		0.92268	48	3Y
32	▲3	LV		0.92261	8	3Y
33	▲4	mgchbot		0.92261	111	3Y
34	▼12	big data is cool	Top 0.38% !	0.92259	47	3Y

Description of the Data

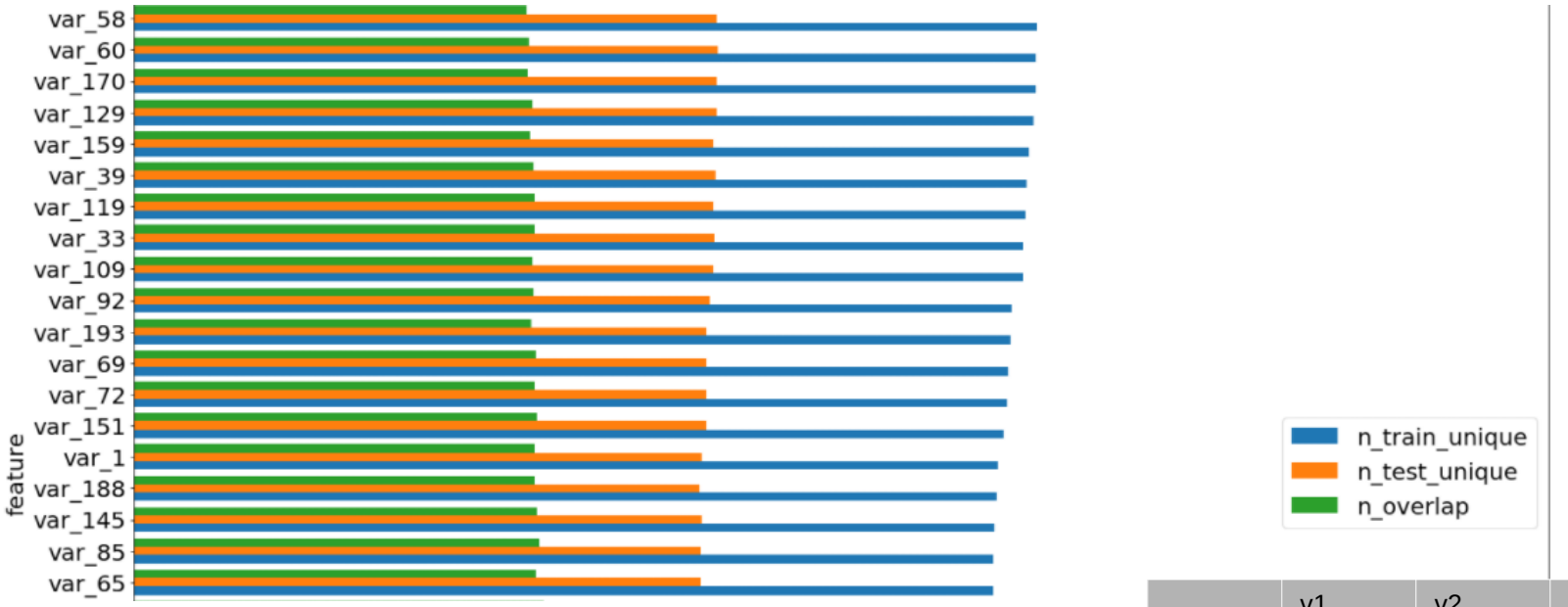
train.csv (302.13 MB)

Detail Compact Column 10 of 202 columns

ID_code	target	var_0	var_1	var_2	var_3	var_4	var_5	var_6
200000 unique values	0 1	0.41 20.3	-15 10.4	2.12 19.4	-0.04 13.2	5.07 16.7	-32.6 17.3	2.35 8.45
train_0	0	8.9255	-6.7863	11.9081	5.093	11.4607	-9.2834	5.1187
train_1	0	11.5006	-4.1473	13.8588	5.389	12.3622	7.0433	5.6208
train_2	0	8.6093	-2.7457	12.0805	7.8928	10.5825	-9.0837	6.9427
train_3	0	11.0604	-2.1518	8.9522	7.1957	12.5846	-1.8361	5.8428
train_4	0	9.8369	-1.4834	12.8746	6.6375	12.2772	2.4486	5.9405
train_5	0	11.4763	-2.3182	12.608	8.6264	10.9621	3.5609	4.5322
train_6	0	11.8091	-0.0832	9.3494	4.2916	11.1355	-8.0198	6.1961
train_7	0	13.558	-7.9881	13.8776	7.5985	8.6543	0.831	5.689
train_8	0	16.1071	2.4426	13.9307	5.6327	8.8014	6.163	4.4514
train_9	0	12.5088	1.9743	8.896	5.4508	13.6043	-16.2859	6.0637
train_10	0	5.0702	-0.5447	9.59	4.2987	12.391	-18.8687	6.0382
train_11	0	12.7188	-7.975	10.3757	9.0101	12.857	-12.0852	5.6464
train_12	0	8.7671	-4.6154	9.7242	7.4242	9.0254	1.4247	6.2815
train_13	1	16.3699	1.5934	16.7395	7.333	12.145	5.9004	4.8222
train_14	0	13.808	5.0514	17.2611	8.512	12.8517	-9.1622	5.7327
train_15	0	3.9416	2.6562	13.3633	6.8895	12.2806	-16.162	5.6979

- 200K examples in training set, 200K examples in test set
- 200 features (X), 90%:0 & 10%: 1 as target (Y)
- Data masking, no meaning, each feature tends to follow Gaussian distribution
- **No missing values !**
- **LightGBM / SVM / NaiveBayes => AUC: 0.87~0.89**

Our Finding: Duplicated Values



We found:

- Each feature has some duplicated values / unique values
- Duplicated values up to 5 digits? There must be some story!

	v1	v2	v3
s1	**0.1234**	17.3412	9.5137
s2	0.4125	11.8451	4.3351
s3	1.3857	8.1761	1.6897
s4	**0.1234**	8.1761	9.5137
s5	**0.1234**	6.3518	2.4114

Our Hypothesis

Hypothesis:
Duplicated values and unique values originate from two weakly-correlated distributions !

Question:
Two distribution in one feature, how to tell our machine learning model this hypothesis?

Refer Friends	Brand	Size (cm)	Style
0	addidas	42.5	Running
1	Nike	38.5	Tennis
1	Reebok	36	Running
0	Timberland	40	Casual
0	New Balance	34	Running

- Female shoes and male shoes are very different.
- Their customers and shopping behaviors are different also.
- Nearly-independent (weakly-correlated) distrubitions.

Solution 1: Extra Column

Put an extra column to tell the model their originated distribution.

Refer Friends	Brand	Size (cm)	Style	Sex
0	addidas	42.5	Running	M
1	Nike	38.5	Tennis	M
1	Reebok	36	Running	F
0	Timberland	40	Casual	M
0	New Balance	34	Running	F

adding extra columns to label whether the numbers are duplicated / unique values

	v1	v2	v3	v1'	v2'	v3'
s1	**0.1234**	17.3412	9.5137	**0**	1	0
s2	0.4125	11.8451	4.3351	1	1	1
s3	1.3857	8.1761	1.6897	1	0	1
s4	**0.1234**	8.1761	9.5137	**0**	0	0
s5	**0.1234**	6.3518	2.4114	**0**	1	1

AUC in LightGBM: 0.90 !

Solution 2: Add missing Values

Separate all features to Female and Male

Refer Friends	Brand-M	Size-M	Style-M	Brand-F	Size-F	Style-F
0	adidas	42.5	Running	N/A	N/A	N/A
1	Nike	38.5	Tennis	N/A	N/A	N/A
1	N/A	N/A	N/A	Reebok	36	Running
0	Timberland	40	Casual	N/A	N/A	N/A
0	N/A	N/A	N/A	New Balance	34	Running

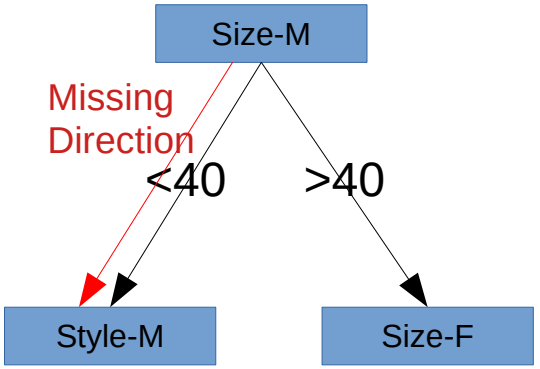
Separate all features to duplicated and unique

	v1-dup	v2-dup	v3-dup	v1-uni	v2-uni	v3-uni
s1	**0.1234**	N/A	9.5137	**N/A**	17.3412	N/A
s2	N/A	N/A	N/A	0.4125	11.8451	4.3351
s3	N/A	8.1761	N/A	1.3857	N/A	1.6897
s4	**0.1234**	8.1761	9.5137	**N/A**	N/A	N/A
s5	**0.1234**	N/A	N/A	**N/A**	6.3518	2.4114

AUC in LightGBM: 0.92 ! (Silver Medal !)

Deep Dive Sparsity-Aware

Refer Friends	Brand-M	Size-M	Style-M	Brand-F	Size-F	Style-F
0	addidas	42.5	Running	N/A	N/A	N/A
1	Nike	38.5	Tennis	N/A	N/A	N/A
1	N/A	N/A	N/A	Reebok	36	Running
0	Timberland	40	Casual	N/A	N/A	N/A
0	N/A	N/A	N/A	New Balance	34	Running



What happened:
1. Every split are based on F or M distribution only
2. F or M examples are always grouped in each split
3. M and F distribution are interacted in an indirected way.
4. Such an idea only works in sparsity-aware

Question: Are missing values always bad ?
Answer: Not really !

Conclusion

- Type of Missing Values: MCAR, MAR, MNAR
- Single Imputation:
  mean/mdeian/mode, hot-deck, KNN, Matrix Factorization
- Multiple Imputation:
  MICE(iterative+PMM), SICE, DAE
- No Imputation:
  Sparsity-Aware, Deep Learning,not for analysis, remember to **add MV in training!**
- Dummy MV + Sparsity Aware could improve performance in some cases (two weakly interacted distributions on the same feature).