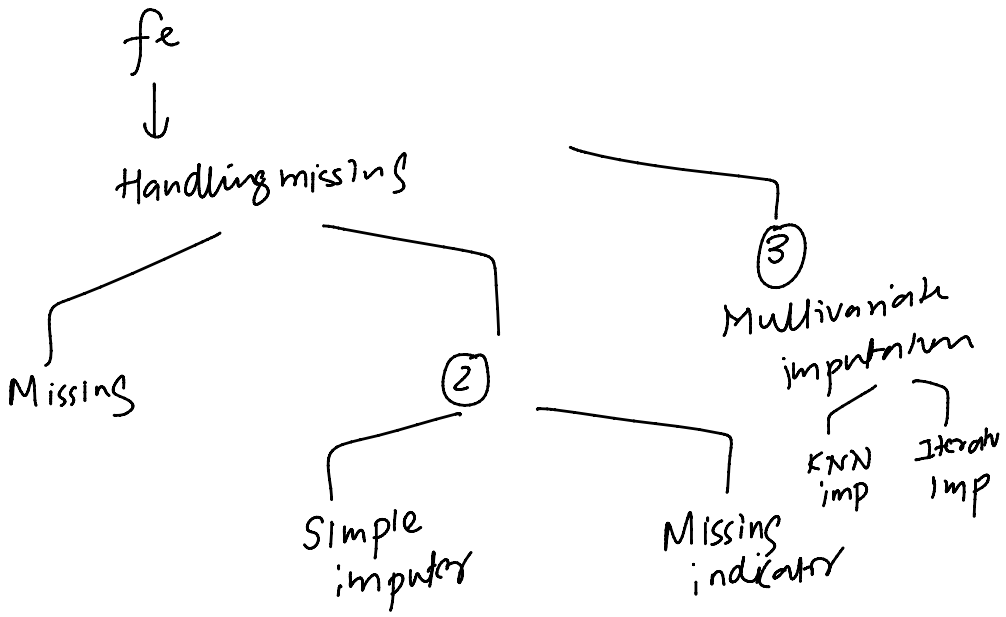
Recap

24 February 2024 14:21



→ Code

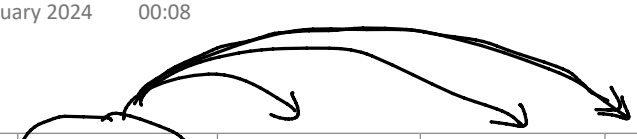
→ comparative study

Multivariate Imputer

25 February 2024 00:08

multivar

Multivariate imp



S NO	Feature 1	Feature 2	Feature 3	Feature 4
1	-----	45	67	21
2	33	------	68	12
3	23	51	71	18
4	40	-----	81	------
5	-----	60	79	------

KNNimp

Iterative imp.

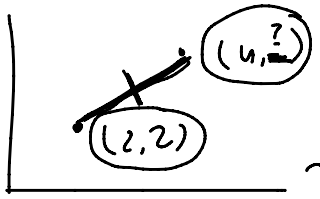
multivar

MCAR | MAR | MNAR

NaN Euclidean distance

26 February 2024 12:17

NaN euclidean dist



$\sqrt{(4-2)^2 + (4-2)^2}$

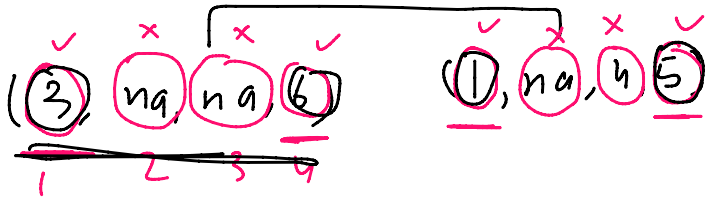
↓

nan euclidean

4 → total axes

2 → avail

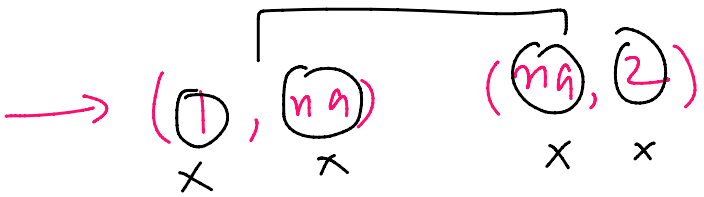
nan



$\frac{4}{2}$

$\sqrt{\frac{4}{2}\left[(3-1)^2 + (6-5)^2\right]}$ → euclidean dis

undefined



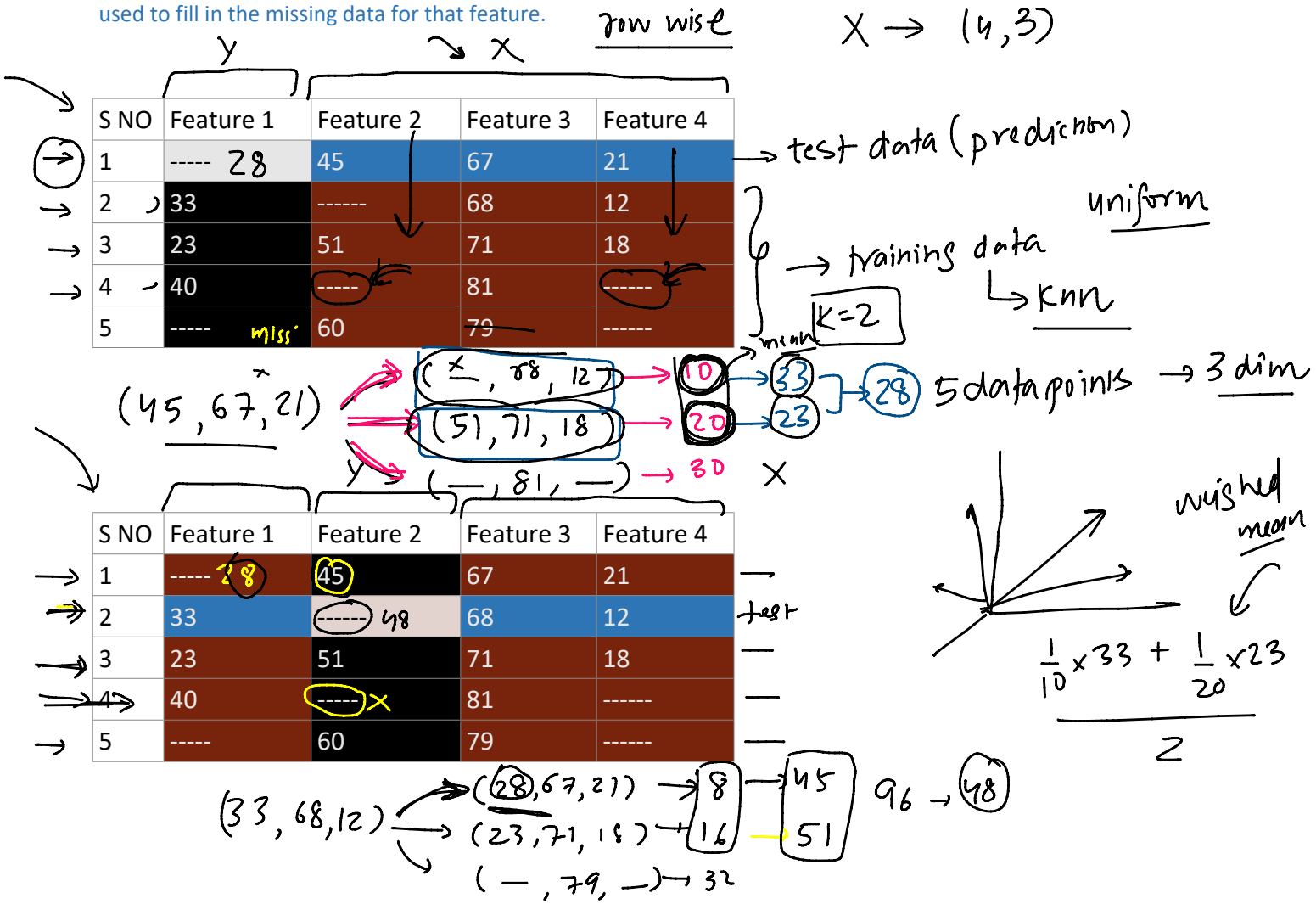
KNN Imputer

24 February 2024 14:22

The KNNImputer is an imputation method provided by the scikit-learn library in Python, designed to fill in missing values in datasets using the k-Nearest Neighbors approach. The core idea behind KNNImputer is to impute missing values in an observation with the mean or median (depending on the configuration) of the nearest neighbors found in the training set. Neighbors are determined based on the similarity of their features, typically measured using a distance metric such as Euclidean distance.

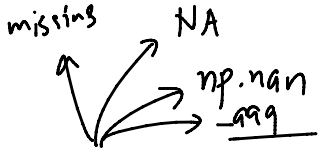
How It Works

- Finding Neighbors: For each sample with missing values, KNNImputer identifies k nearest neighbors using a distance metric across the non-missing dimensions(also known as nan_euclidean distance). The default number of neighbors (k) is 5, but this can be adjusted according to the specific requirements of the dataset and the desired robustness of the imputation.
- Imputation: Once the nearest neighbors are identified, the imputer calculates the mean or median of these neighbors values for each missing feature. This calculated value is then used to fill in the missing data for that feature.



Code and Hyperparameter

24 February 2024 16:28



1. missing_values: Specifies the placeholder for the missing values. The default is np.nan (NumPy's representation of Not a Number), but it can be changed to any other numerical value that represents missing data in your dataset.
2. n_neighbors: The number of neighboring samples to use for imputation. The default value is 5. The imputer selects the n_neighbors closest points in the feature space to use their average (or another specified metric) to fill in the missing value.

# of neighbors

3. weights: This parameter controls the weight function used in prediction. It can be set to:
   - 'uniform': All neighbors are weighted equally. This is the default.
   - 'distance': Weights points by the inverse of their distance. In this case, closer neighbors will have a greater influence on the imputation.
   - Callable: A user-defined function that accepts an array of distances and returns an array of the same shape containing the weights.
4. metric: The distance metric to use for finding nearest neighbors. It can be any metric listed in the documentation of scikit-learn's DistanceMetric class, such as 'euclidean', 'manhattan', 'minkowski', or a callable function that computes the distance. The default is 'nan_euclidean', which is the Euclidean distance that can handle missing values by ignoring them in the distance computation.
5. copy: If True, the imputer will create a copy of the input data before applying imputation. If False, imputation will be done in-place whenever possible. The default is True.

→ X_train

6. add_indicator: When set to True, the imputer will add a binary indicator matrix to the output, indicating the presence of missing values in the input data. Each column in the indicator matrix corresponds to a column in the original data, with 1 indicating a missing value and 0 indicating a non-missing value. This can be useful for downstream models to identify which values were imputed. The default is False.

missing indicator

7. keep_empty_features: This parameter determines whether features (columns) that are entirely missing when the fit method is called should be retained in the output when transform is called.
   - If True, features that consist exclusively of missing values at the time of fitting are included in the output dataset after transformation. These features are imputed with a default value of 0, regardless of the imputation strategy applied to other features. This allows you to keep the original structure of your dataset intact, including columns that were completely missing initially.
   - If False (the default setting), these completely missing features are not included in the output dataset after transformation. This can be useful for reducing the dimensionality of the dataset if certain features provide no information due to their entirely missing values.

→ f1 → keep

0
0
0
0
0
0

Deep Dive

24 February 2024 16:28

When to use KNN Imputer

1. MAR
2. Small to moderate amount of missing data
3. Where observations can be naturally grouped (Homogeneous Subgroups)

When not to use

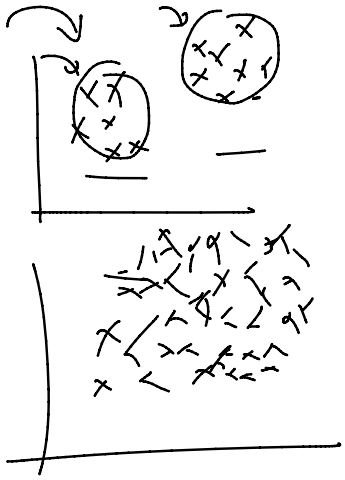
1. Very large dataset
2. High dimensional data
3. MNAR
4. Lots of categorical data

Advantages

1. Simple to implement
2. No assumption about the distribution of data
3. Help preserve distribution

Disadvantages

1. Computational Cost
2. Curse of dimensionality
3. Figuring out the right k is hassle
4. Need to store the entire training set during prediction



Iterative Imputer

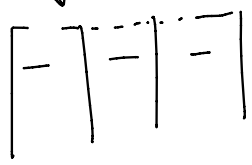
26 February 2024 11:30

MICE → multivariate

flexible → good results

complex

MAR



Iterations → 0 → replaced all the missing value → simple imputer → mean

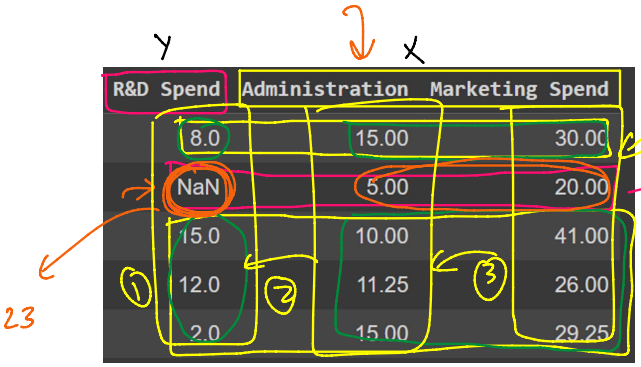
1

2

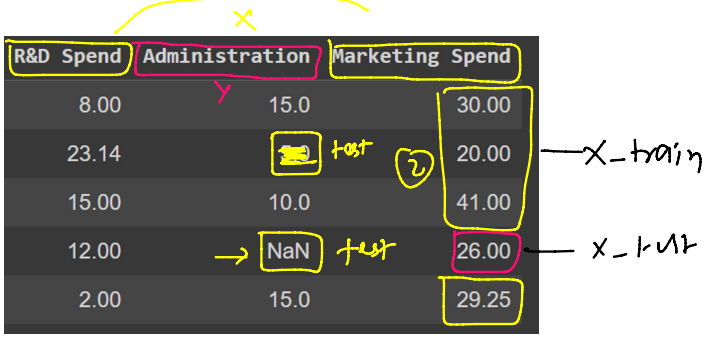
.

.

.



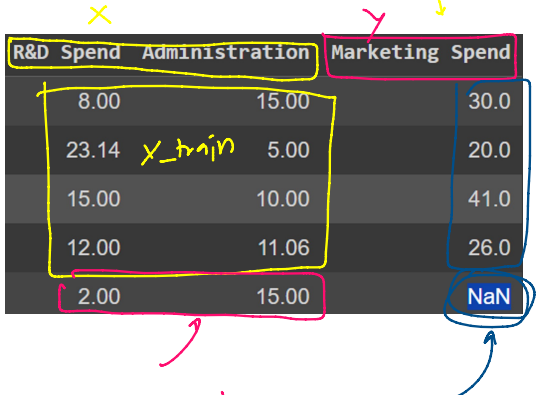
Random forest (X_train, Y_train)



Random (X_train, Y_tr)

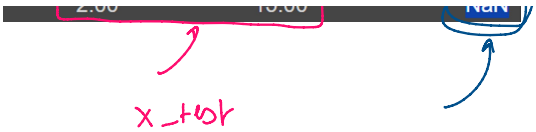
Random (X_test

Y_train



Random (X_train, Y_

Random (X_test



x_test
-Iter 0

simple imputer

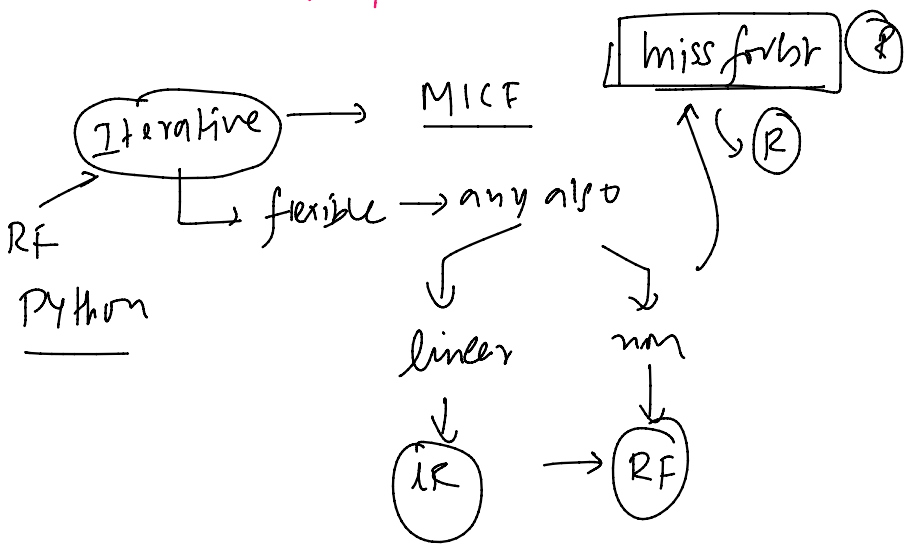
Iter 1
Linear reg

R&D Spend	Administration	Marketing Spend
8.00	15.00	30.00
9.25	5.00	20.00
15.00	10.00	41.00
12.00	11.25	26.00
2.00	15.00	29.25

R&D Spend	Administration	Marketing Spend
8.00	15.00	30.00
[illegible]	5.00	20.00
15.00	10.00	41.00
12.00	[illegible]	26.00
2.00	15.00	[illegible]

→ Iter - (2)

multiple time imputation → Other cols → MICE



Code and Parameters

26 February 2024 11:49

Deep Dive

26 February 2024 11:49

When to use?

1. When the data has missing values across multiple features
2. For datasets where relationship between features is complex and non-linear
3. MAR

When not to use?

1. In cases where more than 50 percent of data is missing
2. MCAR/MNAR
3. Categorical Data

Advantages

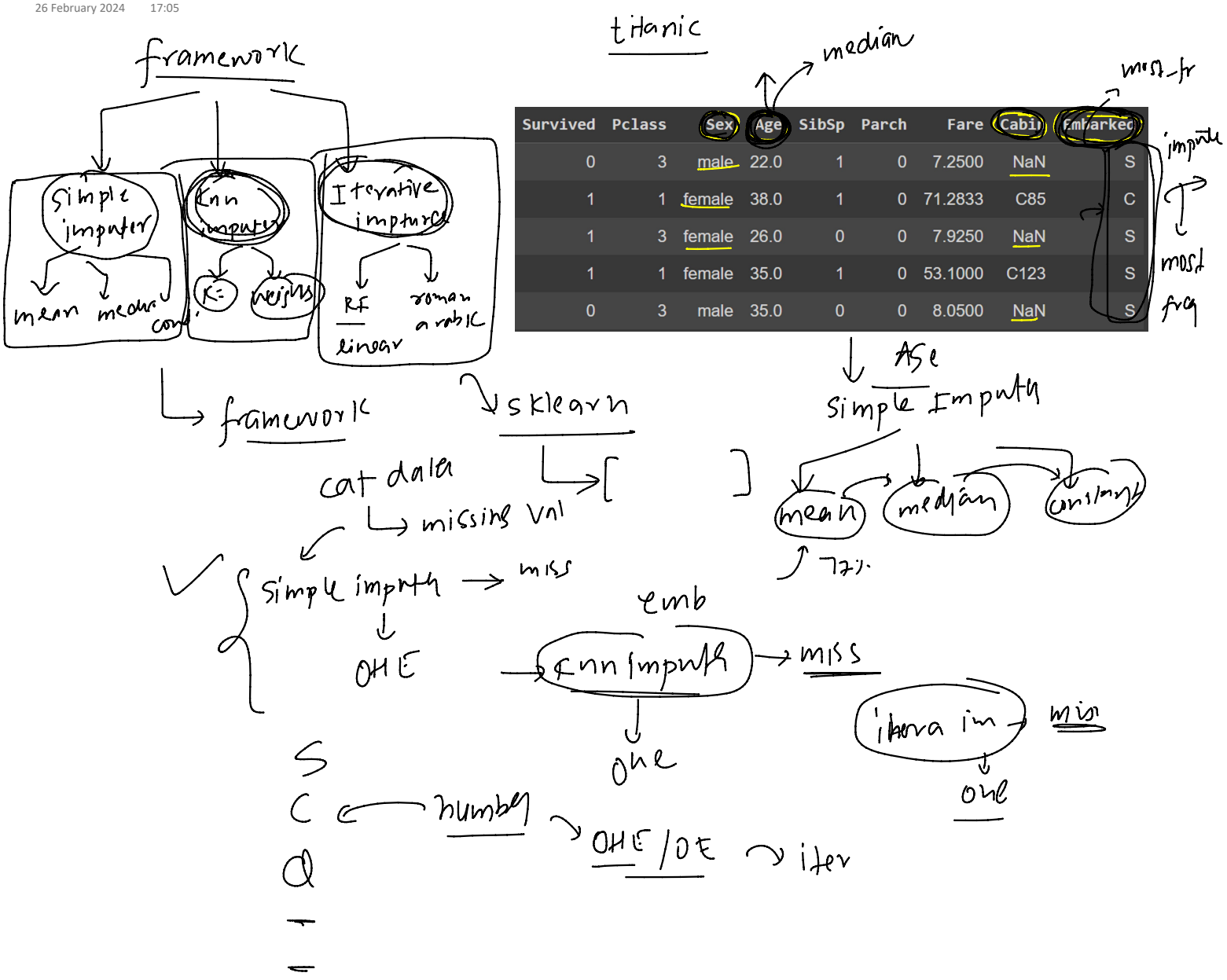
1. Flexible estimator choice

Disadvantages

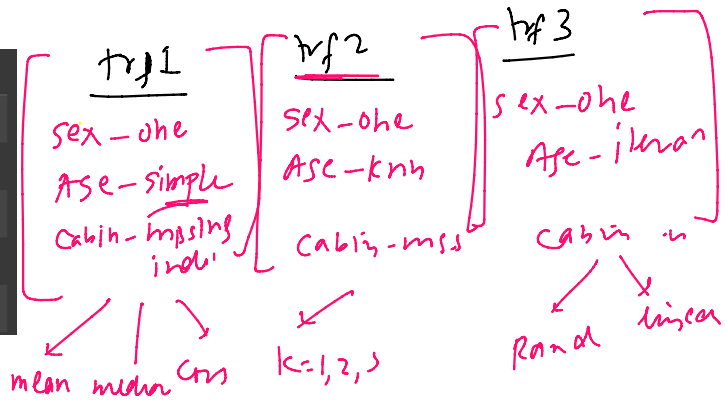
1. Computational Cost
2. Risk of overfitting
3. Convergence issue
4. Sensitive to initialization

Coding Framework to compare different techniques

26 February 2024 17:05



Pclass	Sex	Age	SibSp	Parch	Fare	Cabin
1	male	45.5	0	0	28.5000	C124
2	male	23.0	0	0	13.0000	NaN
3	male	32.0	0	0	7.9250	NaN
3	male	26.0	1	0	7.8542	NaN
3	female	6.0	4	2	31.2750	NaN



Comparison of techniques

26 February 2024 17:05