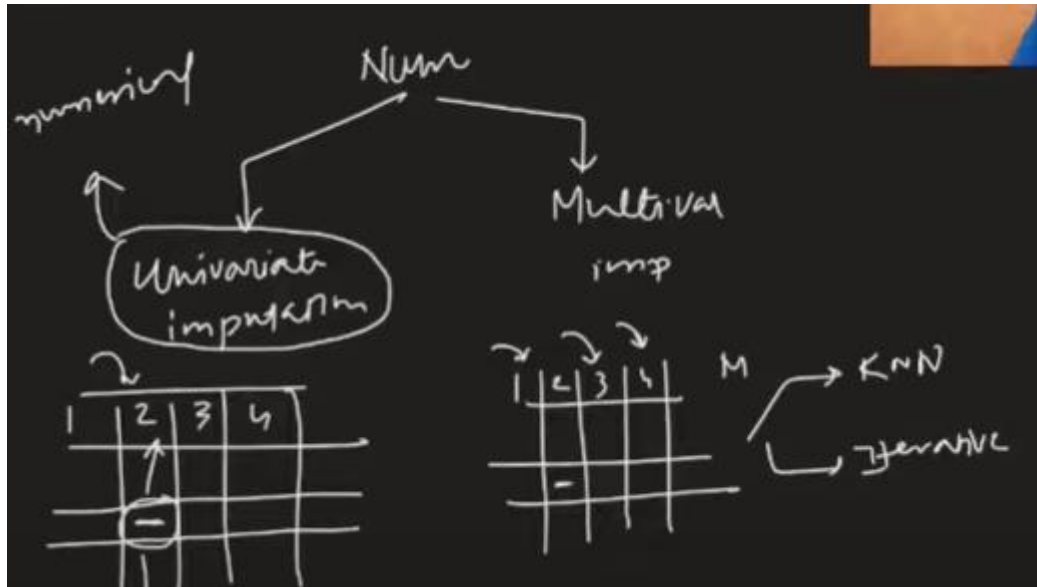


Machine Learning

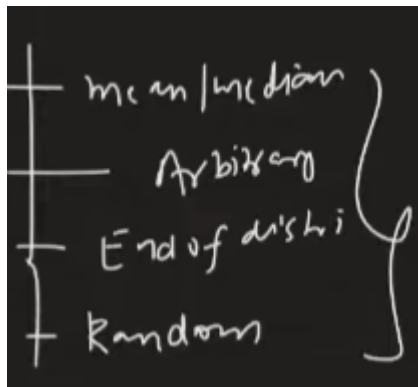
Video 36:

Handling missing data | Numerical Data | Simple Imputer:

How to impute (fill) missing numerical values?



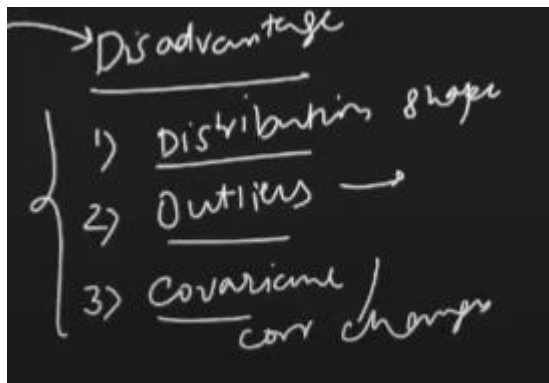
What are some methods to do so?



How to use mean/median imputation?

Mean imputation replaces missing values with the average of the data, best used when the data is normally distributed and doesn't have outliers. Median imputation replaces missing values with the median, suitable for skewed data or when outliers are present, as the median is less affected by extremes. It is used for MCAR or less than 5% data is missing.

Disadvantages: it changes shape of distribution. It also, add outliers. Changes in correlation



Example:

Code link:

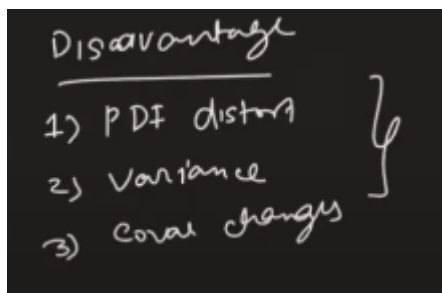
<https://colab.research.google.com/drive/1LHoZNx2Rl84it9DKbrkwzIF1NBDF2ftZ?usp=sharing>

Data link:

<https://github.com/campusx-official/100-days-of-machine-learning/tree/main/day36-imputing-numerical-data>

How to use Arbitrary value imputation?

Arbitrary value imputation in machine learning involves replacing missing data with a predefined constant value, such as the mean, median, or a user-defined number. It's useful for filling gaps in datasets, ensuring models can process complete data, though it may introduce bias depending on the chosen value. Used when data is not missing at random.



Code link:

<https://colab.research.google.com/drive/1LHoZNx2Rl84it9DKbrkwzIF1NBDF2ftZ?usp=sharing>

How to use End of distribution imputation?

End of distribution imputation involves replacing missing values with the extreme values from the dataset, either the minimum or maximum value. This method is useful when the missing data may fall outside the general distribution, preserving data integrity while avoiding artificial data points. It's typically applied to numerical features.

Video 37:

Handling Missing Categorical Data | Simple Imputer | Most Frequent Imputation | Missing Category Imp:

Techniques involved here:



What do you mean by Most frequent Value Imputation?

Most frequent value imputation in machine learning refers to replacing missing or null values in a dataset with the most frequent (mode) value of that feature. This technique is commonly used for categorical variables to maintain data consistency and prevent the loss of information from missing entries.

What do you mean by Missing Category Imputation?

Missing category imputation involves replacing missing values in a categorical feature with a new, distinct category (e.g., "Missing" or "Unknown"). This approach helps preserve the integrity of the dataset by acknowledging the absence of data, rather than assuming it aligns with any existing category. It is particularly useful when the missingness is informative.

Example:

Code link:

https://colab.research.google.com/drive/1m7ABwvW_arHNDP5kabpSQMMVGOTouPf5?usp=sharing

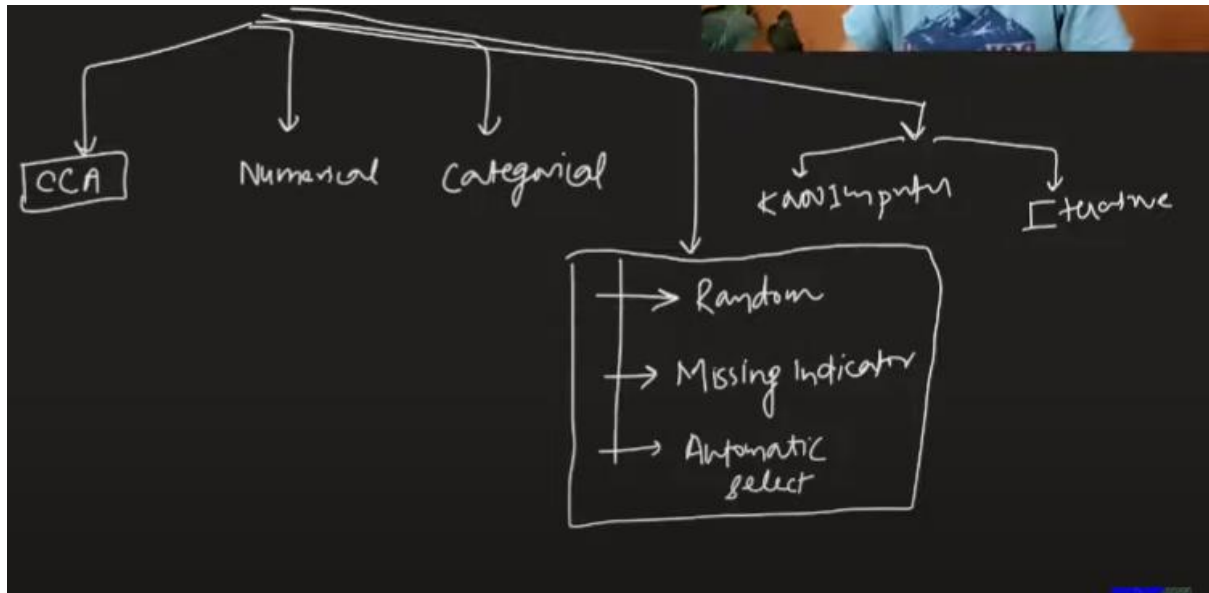
Data link:

<https://github.com/campusx-official/100-days-of-machine-learning/tree/main/day37-handling-missing-categorical-data>

Video 38:

Missing Indicator | Random Sample Imputation | Handling Missing Data:

Till now we have covered the left 3 of these 6:



What is Random Imputation?

Random imputation involves replacing missing values in a dataset with randomly selected values from the existing observed values in that feature. This technique helps maintain the variability and distribution of the data, but it can introduce some randomness and may not always preserve the underlying relationships in the dataset as effectively as other imputation methods.

Advantage:

- Random imputation helps maintain the variability and distribution of the data, which can prevent the dataset from becoming overly biased or skewed.

Disadvantage:

- It can introduce noise and inconsistency, as the imputed values might not reflect the true relationships between features, potentially leading to less reliable model performance.

Example:

Code Link:

<https://colab.research.google.com/drive/1oM1BTu6rtbqNUGRdty0DMDeFPaTzfLCR?usp=sharing>

Data link:

<https://github.com/campusx-official/100-days-of-machine-learning/tree/main/day38-missing-indicator>

What is Missing indicator?

A **missing indicator** is a binary variable (often called a flag) that is added to a dataset to indicate whether a value is missing for a particular feature. It takes the value 1 if the data is missing and 0 if the data is present. This helps machine learning models to recognize patterns related to the absence of data, potentially improving the handling of missing values.

How to automatically select value for imputation?

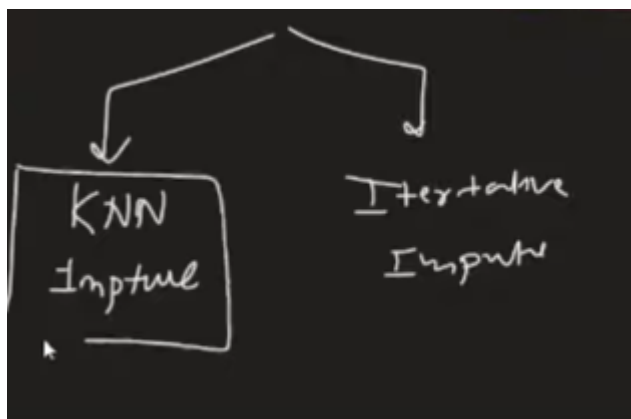
We will use gridsearchcv

Video 39:

KNN Imputer | Multivariate Imputation | Handling Missing Data:

Univariate Imputation: It involves filling missing data in a single variable using information from the same variable. Common methods include mean, median, or mode imputation, where missing values are replaced by a summary statistic from the available data for that feature.

Multivariate Imputation: This method imputes missing data by considering relationships between multiple variables. It uses algorithms like regression, k-NN, or models like MICE (Multiple Imputation by Chained Equations) to predict missing values based on other variables in the dataset.



How does KNN imputer work?

KNN Imputation (K-Nearest Neighbors Imputation) is a technique used to fill missing values in a dataset. It works by finding the 'k' nearest neighbors (data points) to the missing value based on the similarity of other features. The missing value is then imputed using the mean (or weighted mean) of the nearest neighbors' corresponding values. KNN imputation leverages patterns and relationships between data points, making it more accurate than simple imputation methods like mean or median imputation.

The term "**NaN Euclidean Distance**" typically refers to the calculation of Euclidean distance in datasets where some values are missing (represented as NaN, which stands for "Not a Number").

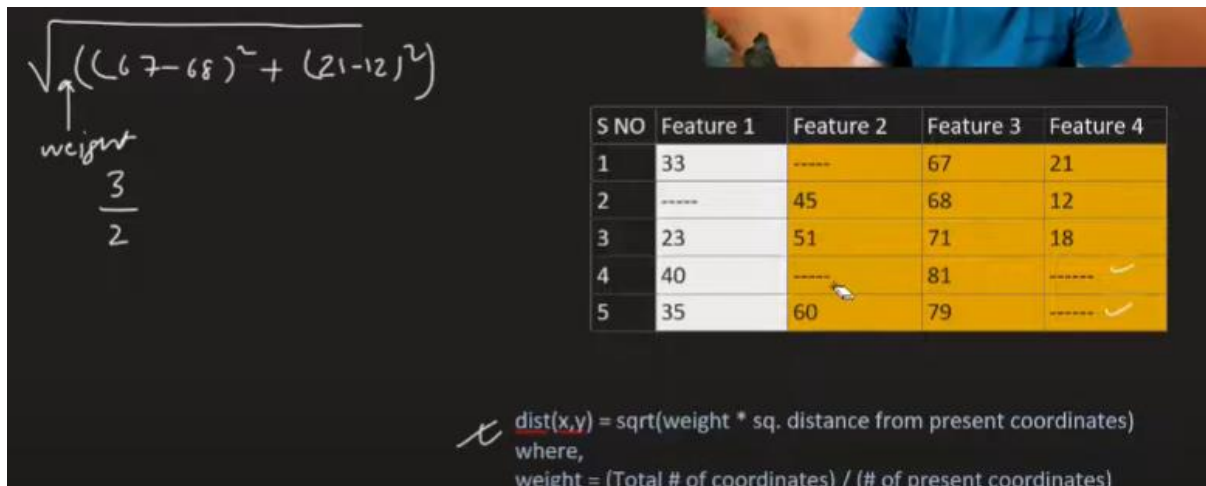
In Euclidean distance calculations, NaN values are often excluded or handled in a special way to avoid errors. For example, when calculating the distance between two points, any NaN values in the

corresponding dimensions might be ignored, or the distance could be computed based on only the available (non-missing) data points.

In practical terms, some libraries or functions may provide specific methods or options to handle NaNs during distance calculations, such as:

- Ignoring dimensions with NaN values
- Replacing NaNs with a predefined value (like zero or the mean of the feature)

The concept is generally applied when you're using distance-based algorithms, such as KNN, in datasets with missing values.



Handwritten formula for weighted distance:

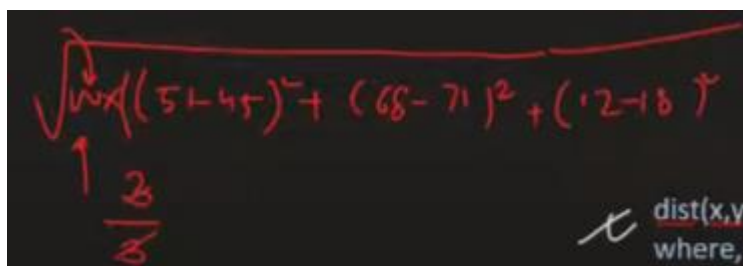
$$\sqrt{\frac{((67-68)^2 + (21-12)^2)}{2}}$$

where, weight = $\frac{3}{2}$

S NO	Feature 1	Feature 2	Feature 3	Feature 4
1	33	-----	67	21
2	-----	45	68	12
3	23	51	71	18
4	40	-----	81	----- ✓
5	35	60	79	----- ✓

where, $\text{dist}(x,y) = \text{sqrt}(\text{weight} * \text{sq. distance from present coordinates})$
 where, $\text{weight} = (\text{Total \# of coordinates}) / (\# \text{ of present coordinates})$

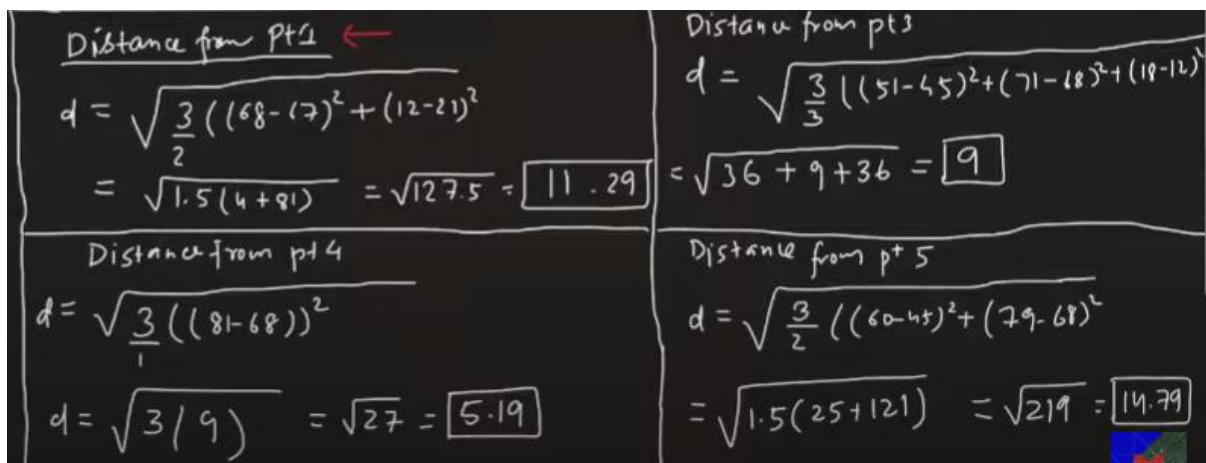
Distance between row 2 and row 3:



$$\sqrt{\frac{((51-45)^2 + (68-71)^2 + (12-18)^2)}{3}}$$

where, $\text{dist}(x,y)$

Say $k = 2$, then



Distance from pt 1

$$d = \sqrt{\frac{3}{2} ((68-17)^2 + (12-21)^2)}$$

$$= \sqrt{1.5(4+81)} = \sqrt{127.5} = \boxed{11.29}$$

Distance from pt 3

$$d = \sqrt{\frac{3}{3} ((51-45)^2 + (71-68)^2 + (12-18)^2)}$$

$$= \sqrt{36 + 9 + 36} = \boxed{9}$$

Distance from pt 4

$$d = \sqrt{\frac{3}{1} ((81-68)^2)}$$

$$= \sqrt{3(9)} = \sqrt{27} = \boxed{5.19}$$

Distance from pt 5

$$d = \sqrt{\frac{3}{2} ((60-45)^2 + (79-68)^2)}$$

$$= \sqrt{1.5(25+121)} = \sqrt{219} = \boxed{14.79}$$

Clearly, nearest are 3 and 4 rows.

Now, advantage and disadvantages:

Advantages of KNN:

1. **Simple and Intuitive:** KNN is easy to understand and implement. It doesn't require a complex model-building process, making it user-friendly.
2. **No Assumptions:** KNN makes no assumptions about the underlying data distribution (non-parametric), which is useful when dealing with complex, real-world data.
3. **Versatile:** Can be used for both classification and regression tasks, making it adaptable to different types of problems.
4. **Works well with smaller datasets:** Effective when the dataset is small or moderate in size, where it can efficiently compute the nearest neighbors.

Disadvantages of KNN:

1. **Computationally Expensive:** For large datasets, KNN can be slow since it computes the distance to all points for each prediction, leading to high computational cost.
2. **Sensitive to Irrelevant Features:** KNN can perform poorly if the dataset contains irrelevant features or noise, as it relies heavily on distance calculations.
3. **Curse of Dimensionality:** As the number of features (dimensions) increases, the algorithm's performance can degrade due to sparsity in the data space.
4. **Storage and Memory Intensive:** Since KNN stores the entire dataset for future predictions, it can require a lot of memory, especially for large datasets.

Example:

Code link:

https://colab.research.google.com/drive/1ynS8eyPbCtkFIIBhZH3OkkArWuH_PQZQ?usp=sharing

Data link:

<https://github.com/campusx-official/100-days-of-machine-learning/tree/main/day39-knn-imputer>

Video 40:

Multivariate Imputation by Chained Equations for Missing Value | MICE Algorithm | Iterative Imputer:

MICE stand for Multivariate imputation by Chained Equations.

MICE (Multiple Imputation by Chained Equations) is a method for handling missing data by creating multiple imputations for each missing value using regression models. It then averages the results to estimate the final value, ensuring more accurate data analysis.

MCAR, MAR, MNAR:

- **MCAR (Missing Completely at Random):** Data missing entirely by chance, unrelated to observed or unobserved data.
- **MAR (Missing at Random):** Missing data related to observed data but not the missing values.
- **MNAR (Missing Not at Random):** Missing data related to the unobserved values themselves.

Example:

Code Link:

<https://colab.research.google.com/drive/1vxW6PMiUH9wQFZyXCdIS9WuLST7U5jhg?usp=sharing>

Data link:

<https://github.com/campusx-official/100-days-of-machine-learning/tree/main/day40-iterative-imputer>