Lecture 8

Data Cleaning

[Slides from Fardina Fathmiul Alam]

What we will learn

1. What is Data Cleaning
2. How to clean data
3. Duplicated records
4. Outlier detection, z-score
5. Data missing at random / Data missing not at random
6. Different types of imputation

Today’s Class Notebook File

https://colab.research.google.com/drive/10kS9JcQDsvwTMhX_P5Pj3fnsHQQ6Wrwf?usp=sharing

What is Data Cleaning

Data cleaning is the process that removes data that does not belong in your dataset.

What is Data Cleaning

Data cleaning is the process that removes data that does not belong in your dataset. It does this by:

- fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset.

Why Data Cleaning is Needed?

Oftentimes the data we get will be in some way “messy”:

- Values will be **missing.**
- Columns will need to be combined.
- Multiple tables need to be joined
- **Inconsistency:**
  - E.g.: Date format "MM/DD/YYYY," "YYYY-MM-DD," and "DD-Mon-YY" (e.g., 01/15/2023, 2023-01-15, Jan-15-23)

Why Data Cleaning is Needed?

Our first step is cleaning!

- Sometimes, this will be **simple and obvious**
  - E.g.: Maybe a word is misspelled in some sort of categorical. You can fix this using apply()
- Sometimes, though, the answer is **non-obvious**!
  - Not all data cleaning issues are immediately apparent or easy to solve.

Copy of New York Citi Bikes_Raw Data

Link:https://docs.google.com/spreadsheets/d/1D7y7LfBG-owIkXOiabRfFG0XcEzQS_8Lz5BsnRkVT3E/edit#gid=845159259

File Edit View Insert Format Data Tools Extensions Help Last edit was seconds ago

100% $ % .0 .00 123 Arial 10

1:20401 Start Time

	A	B	C	D	E	F	G	H	I	J	K
1	Start Time	Stop Time	Start Station ID	Start Station Name	End Station ID	End Station Name	Bike ID	User Type	Birth Year	Age	Age Groups
2	01-01-17 00:38	1-1-17 01:03	3194	McGinley Square	3271	Danforth Light Rail	24668	Subscriber	1961	60	55-64
3	01-01-17 01:47	01-01-17 01:58	3183	Exchange Place	3203	Hamilton Park	26167	Subscriber	1993	28	25-34
4	01-01-17 01:47	01-01-17 01:58	3183	Exchange Place	3203	Hamilton Park	26167	Subscriber	1993	28	25-34
5	01-01-17 01:56	01-01-17 02:00	3186	Grove St PATH	3270	Jersey & 6th St	24604	Subscriber	1970	51	45-54
6	1-1-17 02:12	01-01-17 02:23	3270	Jersey & 6th St	3206	Hilltop	24641	Subscriber	1978	43	35-44
7	01-01-17 02:22	1-1-17 02:31	3212	Christ Hospital	3225	Baldwin at Montgomer	24520	Subscriber	1987	34	25-34
8	01-01-17 02:23	01-01-17 02:27	3186	Grove St PATH	3203	Hamilton Park	24512	Subscriber	1984	37	35-44
9	1-1-17 03:24	01-01-17 03:26	3186	Grove St PATH	3213	Van Vorst Park	24513	Subscriber	1985	36	35-44
10	01-01-17 03:52	01-01-17 03:55	3203	Hamilton Park	3213	Van Vorst Park	24442	Subscriber	1986	35	35-44
11	01-01-17 03:52	01-01-17 03:55	3203	Hamilton Park	3213	Van Vorst Park	24442	Subscriber	1986	35	35-44
12	01-01-17 06:29	1-1-17 06:31	3186	Grove St PATH	3211	Newark Ave	24681	Subscriber	1964	57	55-64
13	01-01-17 06:29	01-01-17 06:31	3186	Grove St PATH	3211	Newark Ave	24681	Subscriber	1964	57	55-64
14	01-01-17 07:08	01-01-17 07:15	3195	Sip Ave	3193	Lincoln Park	26163	Subscriber	1983	38	35-44
15	01-01-17 08:24	1-1-17 08:28	3186	Grove St PATH	3211	Newark Ave	26308	Subscriber	1977	44	35-44
16	1-1-17 08:56	01-01-17 09:05	3196	Riverview Park	3210	Pershing Field	26234	Subscriber	1983	38	35-44
17	01-01-17 09:20	01-01-17 09:40	3267	Morris Canal	3267	Morris Canal	24699	Subscriber	1984	37	35-44
18	01-01-17 09:45	01-01-17 09:52	3186	Grove St PATH	3270	Jersey & 6th St	24558	Subscriber	1986	35	35-44
19	01-01-17 09:47	01-01-17 09:50	3279	Dixon Mills	3272	Jersey & 3rd	26260	Subscriber	1975	46	45-54
20	01-01-17 09:55	01-01-17 09:58	3272	Jersey & 3rd	3279	Dixon Mills	26260	Subscriber	1975	46	45-54
21	01-01-17 09:57	01-01-17 10:02	3213	Van Vorst Park	3187	Warren St	24704	Subscriber	1980	41	35-44
22	01-01-17 10:03	01-01-17 10:10	3267	Morris Canal	3273	Manila & 1st	26215	Subscriber	1957	64	55-64
23	01-01-17 10:03	01-01-17 10:10	3267	Morris Canal	3273	Manila & 1st	26215	Subscriber	1957	64	55-64

The Easy Stuff

Data Typing

Sometimes the data is in the wrong format!

What to do:

- **For "Date Time data"**, this commonly occur with date-times;
    - Pandas offers a robust datetime library.
- **For other data types,** you can use df["column"].astype(someType) for straightforward conversions.
- **For more complex issues,** utilize df["column"].apply(conversion_function).

Combining and Merging Data Sources

Oftentimes you may need to merge various tables.

What to do: Making sure the file formats all match. → check that columns have consistent data types, matching column names

- **This can be a little tricky sometimes.**
    - various data format issues, resolving data conflicts, and deciding on the appropriate type of join (e.g., inner join, outer join) to use when merging tables.
- Remember to sanity check your work!

Evolving Labeling Schemes

Situations where the labels or categories used to classify data change over time. This happens when how things are labeled changes halfway through.

For example, one field gets split into two subfields, or people decide it makes more sense to change how something is recorded midway through.

- **Example Scenario:** Initially, a column that typically records **"failure"** is modified midway through the project to **include "catastrophic failure" and "partial failure".**

You'll have to cope with these as best you can (**Coping strategies**)

- Divide the dataset in two (based on the time of change)?
- Infer old ratings (based on available data)?

Example: evolving labeling scheme for product quality ratings.

SCENARIO: Suppose you are analyzing customer reviews for a product over a period of time, and you notice a change in the way product quality is rated during the analysis.

Initial Labeling Scheme (Phase 1):

Date	Product_ID	Customer_Name	Quality_Rating
2022-01-05	101	Alice	Good
2022-02-10	102	Bob	Excellent
2022-03-15	103	Carol	Fair

Example: evolving labeling scheme for product quality ratings.

However, midway through the project, the company decides to change the labeling scheme to **include more detailed quality ratings**, introducing new subcategories such as "Excellent," "Very Good," "Good," "Fair," and "Poor".

Evolving Labeling Scheme (Phase 2):

Date	Product_ID	Customer_Name	Quality_Rating
2022-04-20	104	Dave	Very Good
2022-05-25	105	Eve	Good
2022-06-30	106	Frank	Excellent
2022-07-05	107	Grace	Fair
2022-08-10	108	Henry	Poor

The updated labeling scheme now includes "Excellent," "Very Good," "Good," "Fair," and "Poor".

Example: evolving labeling scheme for product quality ratings: Managing Data Changes

Possible Coping Strategies:

Step 1: Splitting the Dataset (by Phrases):

To accommodate changes in the labeling scheme, consider dividing the dataset into two separate groups:

1. One containing data from the initial phase (Phase 1).
2. Another for the data collected after the labeling scheme change (Phase 2).

This division helps maintain consistency in your analysis for each phase.

Example: evolving labeling scheme for product quality ratings: Managing Data Changes

Possible Coping Strategies:

Step 2: Infer/ Interpret Old Ratings:

For records in Phase 1, where you only have "Good," "Excellent," and "Fair" ratings, you can infer how they might map to the new labeling scheme.

- For instance, you might consider mapping "Good" to "Very Good" and "Fair" to "Fair" in the updated scheme.

Keep in mind that this **mapping might depend on what makes the most sense** in your specific situation.

Duplicate Records

Duplicated Records

Sometimes people will put a bunch of duplicate records in your system!

- **If they are exact duplicates**: just do *df.drop_duplicates()*
- **If they are duplicates where some of them are subtly different:** identify the ones that are the true values and drop the rest

ALWAYS CHECK FOR THIS

Example: Duplicated Records

	Customer_ID	Name
0	1	John Doe
1	2	Alice Smith
2	3	Bob Johnson
3	4	johN dOE
4	5	Alice Smith

Example: Duplicated Records

```python
# Convert names to lowercase and remove leading/trailing spaces
df['Cleaned_Name'] = df['Name'].str.lower().str.strip()

# Identify and keep the unique records based on the cleaned names
cleaned_df = df.drop_duplicates(subset='Cleaned_Name', keep='first').drop(columns='Cleaned_Name')

print("\nDataFrame after handling subtle differences:")
print(cleaned_df)
```

	Customer_ID	Name	Email
0	1	John Doe	john@example.com
1	2	Alice Smith	alice@example.com
2	3	Bob Johnson	bob@example.com

Outlier Detection

What defines an outlier?

The formal definition of an outlier may vary depending on the context and statistical method employed.

One common approach is to identify outliers as **data points that deviate significantly from the mean**, often defined as being several standard deviations away.

What defines an outlier?

Outliers are data points that significantly deviate from the majority of the data in a dataset.

- **Example:** unusually high or low values that may be indicative of errors, anomalies, or rare events.

Detecting outliers is important because they can **skew statistical analyses and machine learning models.**

Outlier Detection: Some common approaches

You can find outliers multiple ways. Looking for extreme z-scores is one

$$Z = \frac{x - \mu}{\sigma}$$

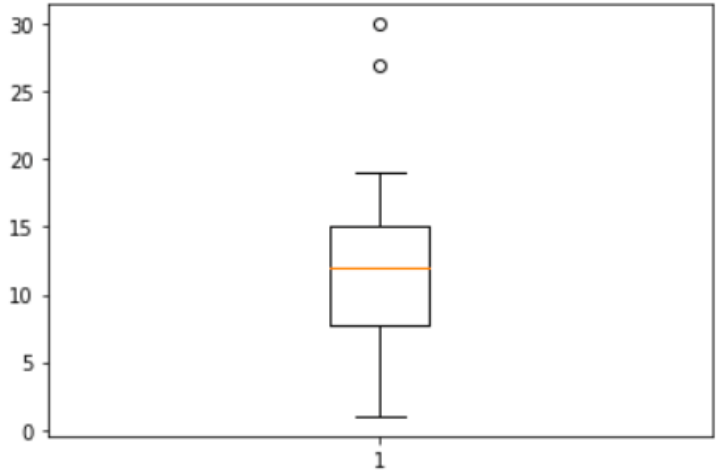
Z = standard score

x = observed value

μ = mean of the sample

σ = standard deviation of the sample

A simple box and whisker plot is another



Outlier Detection using Z-Scores

Z-scores are a statistical measure that quantifies how far a data point is from the mean (average) of a dataset in terms of standard deviations.

- A high positive or negative z-score suggests that a data point is far from the mean and is a potential outlier.

$$Z = \frac{x - \mu}{\sigma}$$

Z = standard score

x = observed value

μ = mean of the sample

σ = standard deviation of the sample

Outlier Detection using Z-Scores

Example:

Test Scores: [80, 85, 88, 90, 92, 95, 98, 100, 150]

Step 01: Calculate the Mean and Standard Deviation:

- Mean (μ) = (80 + 85 + 88 + 90 + 92 + 95 + 98 + 100 + 150) / 9 = 100.44
- Standard Deviation (σ) ≈ 21.11 (rounded for simplicity)

$$Z = \frac{x - \mu}{\sigma}$$

Z = standard score

x = observed value

μ = mean of the sample

σ = standard deviation of the sample

Outlier Detection using Z-Scores

Example:

Test Scores: [80, 85, 88, 90, 92, 95, 98, 100, 150]

Step 02: Calculate Z-scores for each test score using the formula:

- For the test score of 150:
  - Z = (150 - 100.44) / 21.11 ≈ 2.35
- For the other scores, calculate their respective z-scores.

$$Z = \frac{x - \mu}{\sigma}$$

Z = standard score

x = observed value

μ = mean of the sample

σ = standard deviation of the sample

Outlier Detection using Z-Scores

Example:

Test Scores: [80, 85, 88, 90, 92, 95, 98, 100, 150]

$$Z = \frac{x - \mu}{\sigma}$$

Z = standard score

x = observed value

μ = mean of the sample

σ = standard deviation of the sample

Step 03: Identify Extreme Z-scores:

Common threshold used for an extreme z-score: "greater than 2 or less than -2", suggests that a data point is significantly far from the mean.

- Here, the z-score for the score of 150 is approximately 2.35, which is greater than 2.
    - Indicates that the score of 150 is significantly above the mean and is a potential outlier.

Should we remove an outlier always?

It depends!

Ultimately, when doing a data science project, you have some goal in mind. Remove the outlier if it hurts that goal. Consider:

The Context of the Outlier: Is the outlier an **unusual event that will likely never happen again?** (Something like "light intensity during an eclipse").

→If it is, removing the outlier may be appropriate.

Evaluate Impact on Model Performance: Is the outlier something that **will actually *hurt* your model?**

(You don't really need a model factoring in Jeff Bezos when doing financial transactions.

Assess Relevance: Is the outlier **something that is a core element of your data? (Profits during a sale, wait times during traffic);** We should not be remove without careful considerations.

whether to remove an outlier depends on how it affects your project's objectives and if it's an essential or unusual part of your data.

Missing Data

Recap:

Outlier data is a numeric value that is much larger or smaller than other values in the same feature. Outlier data is usually defined as two or three standard deviations from the feature mean.

Duplicate data are two or more identical instances in a dataset. Duplicate instances are usually erroneous and should be removed.

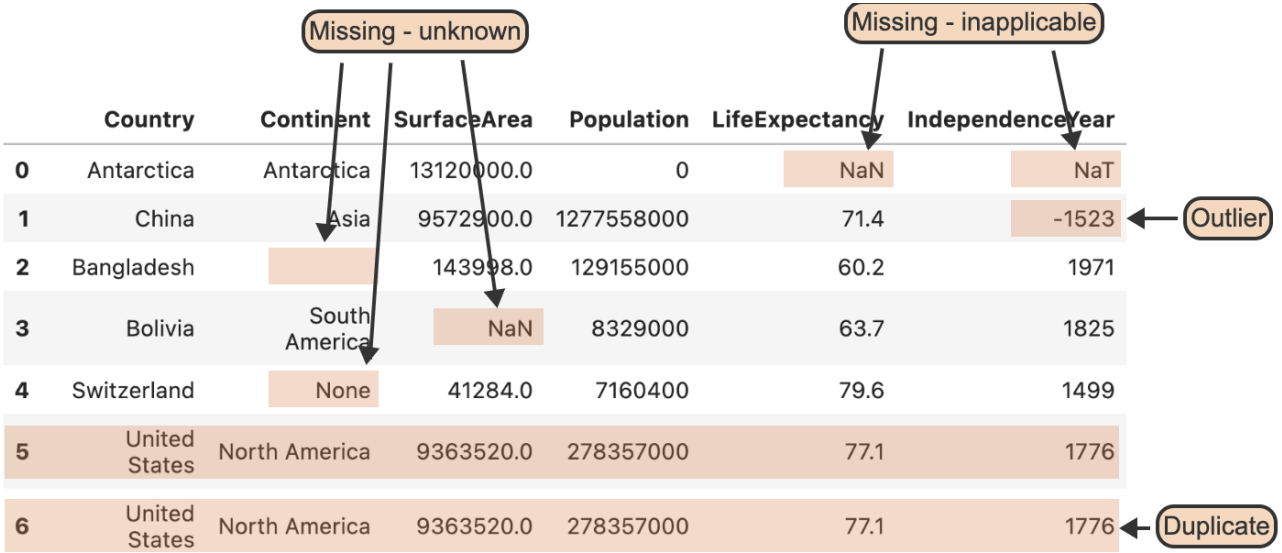
Missing data

The term "missing data" in data cleaning refers to any instances within a dataset where certain values or observations are not recorded or are incomplete.

In a database, missing data is represented as NULL. In Python, missing data is represented as NaN (not a number), NaT (not a time), None (an unspecified object), or a blank value.

Dirty Data

Missing, outlier, and duplicate data are collectively called dirty data. (Ref: Zybook 5.4)



	Country	Continent	SurfaceArea	Population	LifeExpectancy	IndependenceYear
0	Antarctica	Antarctica	13120000.0	0	NaN	NaT
1	China	Asia	9572900.0	1277558000	71.4	-1523
2	Bangladesh		143998.0	129155000	60.2	1971
3	Bolivia	South America	NaN	8329000	63.7	1825
4	Switzerland	None	41284.0	7160400	79.6	1499
5	United States	North America	9363520.0	278357000	77.1	1776
6	United States	North America	9363520.0	278357000	77.1	1776

Dirty data creates bias and inefficiencies in data analysis.

Back to: Missing Data

Why is our data missing?

- Non-responses on a survey
- Sensor malfunction
- Data not provided by a third party
- Some idiot forgot to enter it

Age
21
19
NaN
22
20
21

Missing data can be categorized as:

- Data Missing Completely at Random (MCAR)
- Data missing at random (MAR)
- Data missing not at random (MNAR)

Missing data can be categorized as:

Data missing Completely at random (MCAR):

- There is no pattern to what data is missing
- In practical terms, for all possible values in that column, there is an equal chance of that value being missing
- P(Row R missing | Row R actual value) = P(Row R missing)

For example: a drive is corrupted and a bunch of bits are dropped at random

Data missing at random (MAR):

- Missing at Random (MAR) means that the missingness of data is not entirely random, but can be explained by other observed variables.
- It is a broader category than MCAR, as it allows for the missingness to be related to other variables, as long as it is not directly related to the missing variable itself.
- P(Missing|Missing Data,Observed Data)=P(Missing|Observed Data)

For example: in a health survey, males might be less likely to report certain health issues, leading to missing data that depends on gender but not on the specific health issue itself (the actual health issue is not directly causing the missingness).

Missing data can be categorized as:

Data missing not at random (MNAR):

- **Either: One particular *value* is more likely to be missing, or one particular *range* is more likely to be missing.**
- P(Row R missing | Row R actual value) != P(Row R missing)

For example: An entire day's worth of data got deleted intentionally. Or, on a survey that asks about GPA, students with lower GPA might choose not answer.

Example: MCAR

Age	Gender	Income	Favorite Color
25	Male		Blue
30	Female	50,000	Red
22	Male	45,000	Red
28	Female	60,000	Green
	Male	55,000	Yellow

The missing data (Income and Age) happened randomly due to technical issues (some survey forms didn't save properly) and has no connection to the observed data (Age, Gender, Favorite Color) or the actual missing values.

Example: MAR

In a medical study, older patients are more likely to skip questions about their mental health status, but their actual mental health condition does not affect whether they skip the question.

Age	Gender	Physical Health Score	Mental Health Status
65	Female	80	
70	Male	75	Mild Depression
30	Female	90	No Issues
50	Male	85	
45	Female	88	Moderate Anxiety

*Reason for Missingness: Older age is associated with the likelihood of missing data on Mental Health Status, but the actual Mental Health Status is not directly causing the missingness.

* Implication: The missing data is related to observed data (Age), but not to the missing data itself (Mental Health Status).

Example: MNAR

In a financial survey, people with very high incomes are less likely to report their income because they prefer to keep it confidential.

Age	Gender	Occupation	Income
25	Male	Engineer	70,000
30	Female	Doctor	
22	Male	Student	20,000
28	Female	Lawyer	
45	Male	Executive	

*Reason for Missingness: The actual value of the income **itself influences** the likelihood of the data being missing.

* **Implication:** The missing data is directly related to the value of the missing data itself.

HOW TO HANDLE

Data Missing at Random/ Completely Random

Data Missing at Random

For data that is categorical, it's fine! "Missing" can just become a new category.

Ex: If our category is major, we could have { "Computer Science", "Philosophy", "Did not answer" }

For numerical data, most algorithms we use **cannot accept NaN**. So what do we do?

Solution 1: Drop it

If 1% of your data has missing rows and you have a terabyte of it, just drop all rows that have missing stuff.

df.dropna()

Listwise Deletion: Remove any rows with missing data. This works well when the proportion of missing data is small.

Pairwise Deletion: Use only the data points where both variables being analyzed (e.g., income and education) have non-missing values, excluding cases with missing values for either variable from that specific analysis.

Solution 2: Apply Imputation!

Imputation is the process of replacing missing values with estimates (new values).

Data may be imputed in several ways:

- **Mean/ Median/Mode Imputation:** Imputation Using (Mean/Median/Mode) Values
- **Hot Imputation:** the value is selected from other instances in the same dataset.
- **Cold Imputation:** the value is selected from other instances in the different dataset.
- etc.

Imputation Using (Mean/Median/Mode) Values:

Usually good enough if you're in a hurry

- Calculate the mean/median (or mode sometimes) of the non-missing values in a column and then replacing the missing values within each column separately and independently from the others.
- REMEMBER: only be used with numeric data.
- Therefore, these methods are suitable for continuous and discrete numerical variables only.

Example: Mean Imputation

Mean Imputation: Set the value to the mean of that column

For example, if a height is missing, fill in 5'5"

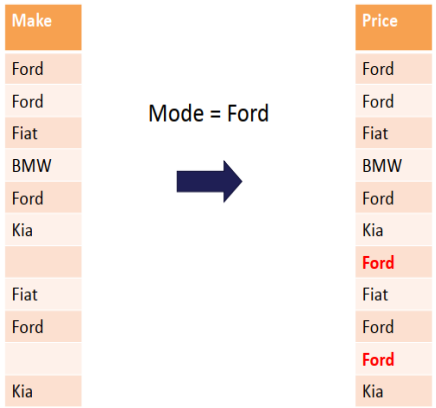
	col1	col2	col3	col4	col5
0	2	5.0	3.0	6	NaN
1	9	NaN	9.0	0	7.0
2	19	17.0	NaN	9	NaN

	col1	col2	col3	col4	col5
0	2.0	5.0	3.0	6.0	7.0
1	9.0	11.0	9.0	0.0	7.0
2	19.0	17.0	6.0	9.0	7.0

Example: Mode Imputation

Mode Imputation: Fill in with the most common value

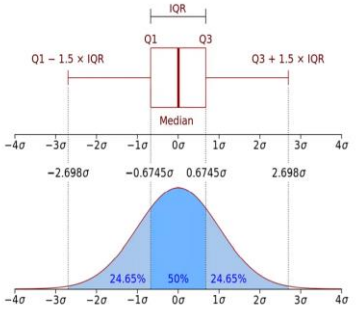
- If someone in this class is missing their major, fill in “Computer Science”



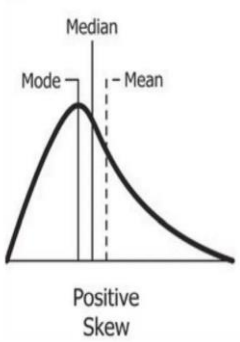
When to use Mean/Median/Mode Imputation

- **Mean-**It is preferred if data is numeric and not skewed.
- **Median-**It is preferred if data is numeric and skewed.
- **Mode-**It is preferred if the data is a string(object) . Suitable for both string and numeric but it's less commonly used for numeric data. It's typically more appropriate for categorical data or situations where there's a clear mode in the distribution.

Use Mean/Median/Mode Imputation when no more than 5% of the variable contains missing data.



- If the variable is normally distributed the mean and median are approximately the same



- If the variable is skewed, the median is a better representation

Hot-Deck Imputation

Hot-deck imputation: is a method used to fill in missing values in a dataset by replacing them with values from similar record/row.

How it works? Find a row that is most similar to the one with the missing value, and copy that value. We can also average over several similar data-points.

- If someone is missing a grade for CMSC 320, find a student who has taken all the same classes as them and copy over their CMSC 320 grades.

Example: Hot-Deck Imputation

Hot-deck imputation Example: Suppose you have a dataset containing information about the ages of individuals, but some ages are missing

ID	Gender	Age
101	Male	32
102	Female	NaN
103	Male	NaN
104	Female	45
105	Male	NaN
106	Female	28

Example: Hot-Deck Imputation

Identify Similar Cases: determine which cases (rows) are similar (such as gender or other relevant characteristics) or close to the one with missing data.

Impute with Similar Values: For each missing age value, find a similar individual (a "donor") from the dataset. In this example, let's use the same gender as a criterion for similarity.

ID	Gender	Age
101	Male	32
102	Female	NaN
103	Male	NaN
104	Female	45
105	Male	NaN
106	Female	28

Example: Hot-Deck Imputation

Hot-deck imputation: After applying hot-deck imputation, your table might look like this

ID	Gender	Age
101	Male	32
102	Female	28
103	Male	32
104	Female	45
105	Male	32
106	Female	28

Bayesian Imputation

A statistical technique used for imputing missing data in a dataset using Bayesian methods.

- **Bayesian statistics is based on Bayes' theorem**, which allows for the estimation of unknown parameters by combining prior information (prior beliefs) and observed data.

- Multiple Possibilities: Unlike simpler imputation methods that rely on point estimates, Bayesian imputation, **it thinks about many values (a range of possible values) for missing data, not just one guess (single estimate point).**

Bayesian Imputation

For categorical data, for example:

$$p(\text{possible\_value} \mid \text{features}) = \frac{p(\text{features} \mid \text{possible value}) * p(\text{possible\_value})}{p(\text{features})}$$

Evaluate each of these for each feature and assign!

Bayesian Imputation: Example

Let's say you have a dataset containing information about the ages of individuals, but some ages are missing. You want to impute (fill in) the missing ages using Bayesian imputation.

- Using Bayesian imputation, you **estimate the missing ages based on the observed data and the relationships between age, gender, and other relevant variables.**
- For instance, if most females in the dataset are in their 30s, and a female individual with a missing age has similar characteristics to those in their 30s, the algorithm may impute an age around 30 for that individual.

Bayesian Imputation: another example

Suppose you have a dataset about fruits with two features: color and shape. You want to impute the missing values for the "Type of Fruit" category.

Color	Shape	Type of Fruit
Red	Round	Apple
Yellow	Long	Banana
Green	Round	NaN
Yellow	Round	NaN
Red	Round	NaN

You want to fill in the missing 'Type of Fruit' values using Bayesian imputation, which involves estimating the probabilities of each fruit type based on the observed data.

Bayesian Imputation

Bayesian imputation can get even more complicated if we want to use Bayes nets, or have to deal with continuous values.

You'll need to measure how much of an impact the missing data has on your dataset

Data Missing Not at Random

NEXT Part