Data Imperfection & Data Prep

https://github.com/Dr-AlaaKhamis/ISE518/tree/main/6_Data_imperfection

Lectures 7 & 8 – Monday & Wednesday September 22 && 24, 2025

Outline

- Introduction
- Data Imperfection
- Structure Issues
- Inconsistency
- Incompleteness
- Redundancy
- Imbalance
- Outlier

Outline

- **Introduction**
- Data Imperfection
- Structure Issues
- Inconsistency
- Incompleteness
- Redundancy
- Imbalance
- Outlier

Introduction

Data in the real world is imperfect

Introduction

Imperfect data contributes to analysis paralysis

Introduction

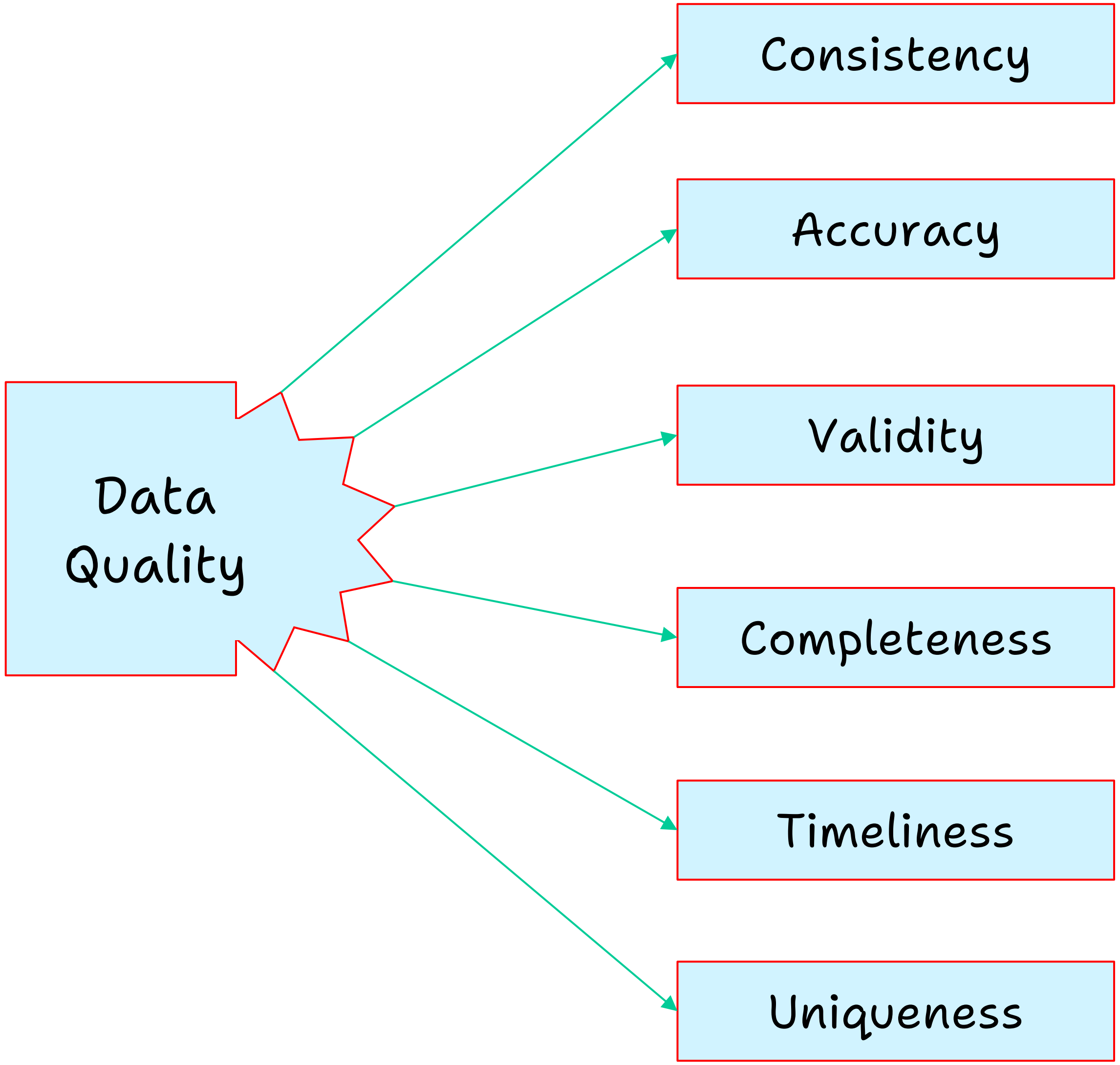
- **Data Quality**

According to Gartner, data quality issues cost the average organization **$12.9 million** every year.

2025 Outlook: Data Integrity Trends and Insight, Drexel LeBow's Center for Applied AI and Business Analytics — Precisely

Introduction

- **Data Quality**



Data values should not conflict with other values across data sets

There should be no errors in the data

The data should conform to a certain format

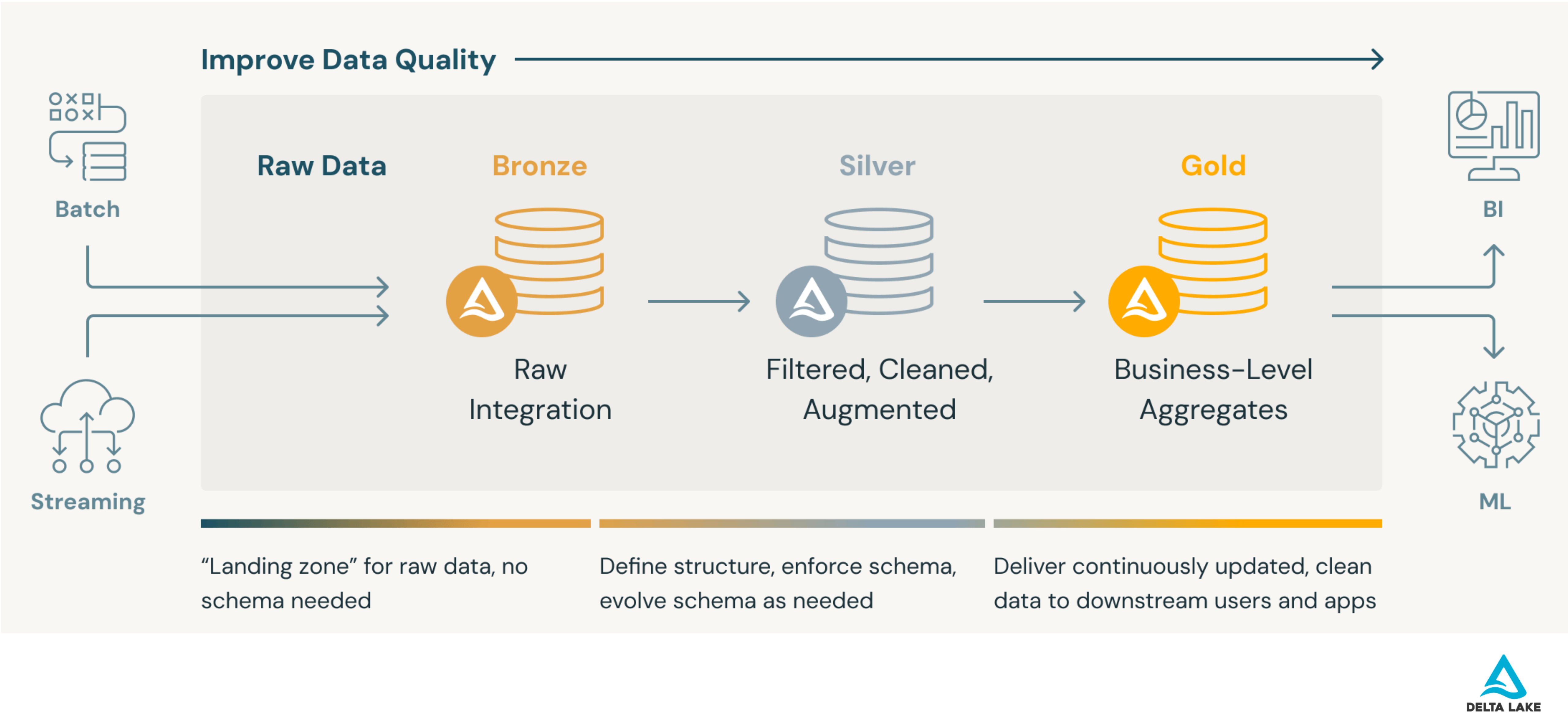
There should be no missing data

The data should be up to date

There should be no duplicates

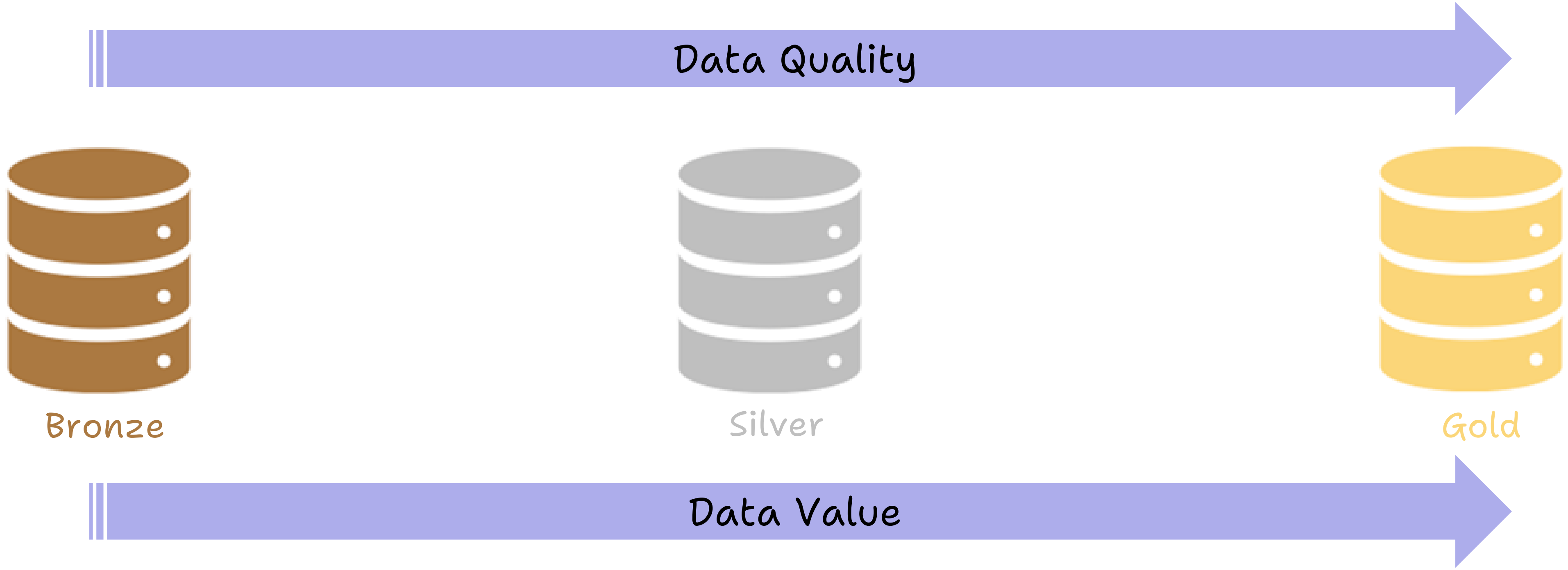
Source: https://www.databricks.com/discover/pages/data-quality-management

- **Medallion Architecture**



Introduction

- **Medallion Architecture**



Bronze	Silver	Gold
Sensor logs and machine data ingested directly from factory floor equipment (e.g., temperature, vibration, error codes, raw PLC messages)	Cleaned and joined data streams with sensor readings mapped to equipment IDs, timestamps corrected, outliers removed, and joined with shift schedules or production orders.	Aggregated production KPIs, such as hourly OEE (Overall Equipment Effectiveness), downtime analysis per machine, predictive maintenance alerts, and executive dashboards for decision-making.

Outline

- Introduction
- **Data Imperfection**
- Structure Issues
- Inconsistency
- Incompleteness
- Redundancy
- Imbalance
- Outlier

Data Imperfection

Data Imperfection Aspects

Structure Issues

- “Equip_ID” vs. “EquipmentID” in different tables.
- Columns labeled “Maint_Date” in one table and “ServiceDt” in another for the same maintenance event, causing misalignment in reliability reports.

Inconsistency

- Dates as “12/05/2025” vs. “05-Dec-2025”.
- Vibration in mm/s vs. in/s.
- Failure types as “Mechanical” vs. “Mech”.
- Run hours as text (“1200 hrs”) vs. numeric (1200).
- Nickname vs. full name (e.g., Moh vs. Mohamed) in equipment operator records or “Aramco” vs. “Saudi Aramco” in company data.

Incompleteness

Missing pump temperature readings.

Redundancy

Duplicate conveyor belt maintenance entries.

Imbalance

90% of equipment sensor readings reflect normal operation (e.g., stable vibration levels), while only 10% capture faulty conditions (e.g., high vibration from bearing wear).

Outlier

- Erratic bearing temperature spikes from interference.
- Downtime outlier of 10,000 hours vs. 10 hours.

Data Imperfection

Data Imperfection Aspects

Structure Issues

Examples:

Header/ column issues

Potential Impact:

- Analytics scripts may fail
- Misaligned data

Mitigation Approach

- Normalize headers
- Remove quotes/newlines
- Standardize names

Inconsistency

Examples:

- Time/date issues, Inconsistent units, Inconsistent categories,
- Incorrect data types
- Noisy data

Potential Impact:

- Misinterpretation of trends
- calculation errors

Mitigation Approach

- Standardize formats
- Convert units
- Enforce correct data types

Incompleteness

Examples:

Missing data

Potential Impact:

Incomplete analysis

Mitigation Approach

Impute missing values

Redundancy

Examples:

Duplicate data

Potential Impact:

Biased results

Mitigation Approach

Remove duplicates

Imbalance

Examples:

Imbalance data

Potential Impact:

Skewed models

Mitigation Approach

Resampling, filtering

Outlier

Examples:

Outliers

Potential Impact:

reduced prediction accuracy

Mitigation Approach

Outlier detection and treatment

Data Imperfection

- **Example: Smart Supply Chain Dataset (DataCo)**

	Type	Days for shipping (real)	Days for shipment (scheduled)	Benefit per order	Sales per customer	Delivery Status	Late_delivery_risk	Category Id	Category Name	Customer City	...	Order Zipcode	Product Card Id	Product Category Id
0	DEBIT	3	4	91.250000	314.640015	Advance shipping	0	73	Sporting Goods	Caguas	...	NaN	1360	73
1	TRANSFER	5	4	-249.089996	311.359985	Late delivery	1	73	Sporting Goods	Caguas	...	NaN	1360	73
2	CASH	4	4	-247.779999	309.720001	Shipping on time	0	73	Sporting Goods	San Jose	...	NaN	1360	73
3	DEBIT	3	4	22.860001	304.809998	Advance shipping	0	73	Sporting Goods	Los Angeles	...	NaN	1360	73
4	PAYMENT	2	4	134.210007	298.250000	Advance shipping	0	73	Sporting Goods	Caguas	...	NaN	1360	73

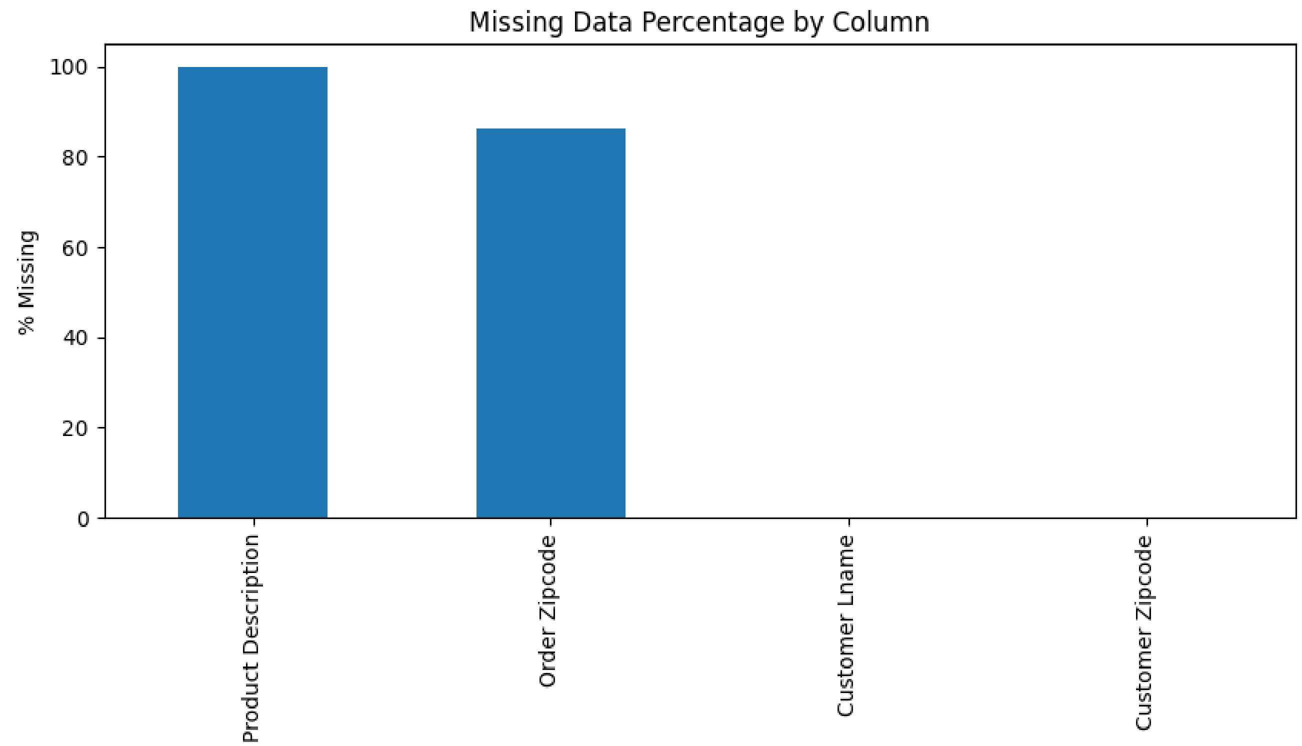
	Type	Days for shipping (real)	Days for shipment (scheduled)	Benefit per order	Sales per customer	Delivery Status	Late_delivery_risk	Category Id	Category Name	Customer City	...	Order Zipcode
count	180519	180519.000000	180519.000000	180519.000000	180519.000000	180519	180519.000000	180519.000000	180519	180519	...	24840.000000
unique	4	NaN	NaN	NaN	NaN	4	NaN	NaN	50	563	...	NaN
top	DEBIT	NaN	NaN	NaN	NaN	Late delivery	NaN	NaN	Cleats	Caguas	...	NaN
freq	69295	NaN	NaN	NaN	NaN	98977	NaN	NaN	24551	66770	...	NaN
mean	NaN	3.497654	2.931847	21.974989	183.107609	NaN	0.548291	31.851451	NaN	NaN	...	55426.132327
std	NaN	1.623722	1.374449	104.433526	120.043670	NaN	0.497664	15.640064	NaN	NaN	...	31919.279101
min	NaN	0.000000	0.000000	-4274.979980	7.490000	NaN	0.000000	2.000000	NaN	NaN	...	1040.000000
25%	NaN	2.000000	2.000000	7.000000	104.379997	NaN	0.000000	18.000000	NaN	NaN	...	23464.000000
50%	NaN	3.000000	4.000000	31.520000	163.990005	NaN	1.000000	29.000000	NaN	NaN	...	59405.000000
75%	NaN	5.000000	4.000000	64.800003	247.399994	NaN	1.000000	45.000000	NaN	NaN	...	90008.000000
max	NaN	6.000000	4.000000	911.799988	1939.989990	NaN	1.000000	76.000000	NaN	NaN	...	99301.000000

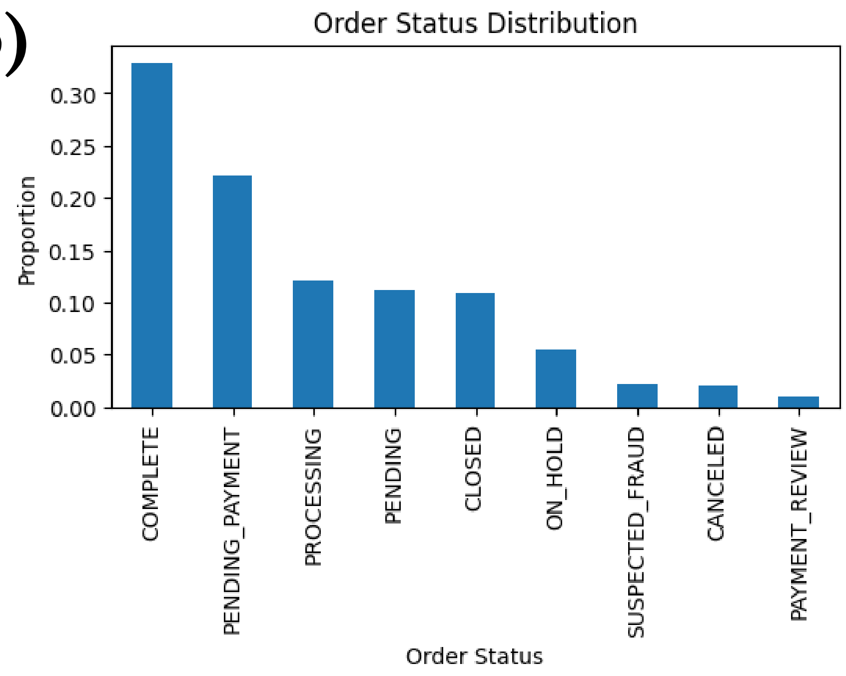
kaggle

https://www.kaggle.com/datasets/shashwatwork/dataco-smart-supply-chain-for-big-data-analysis

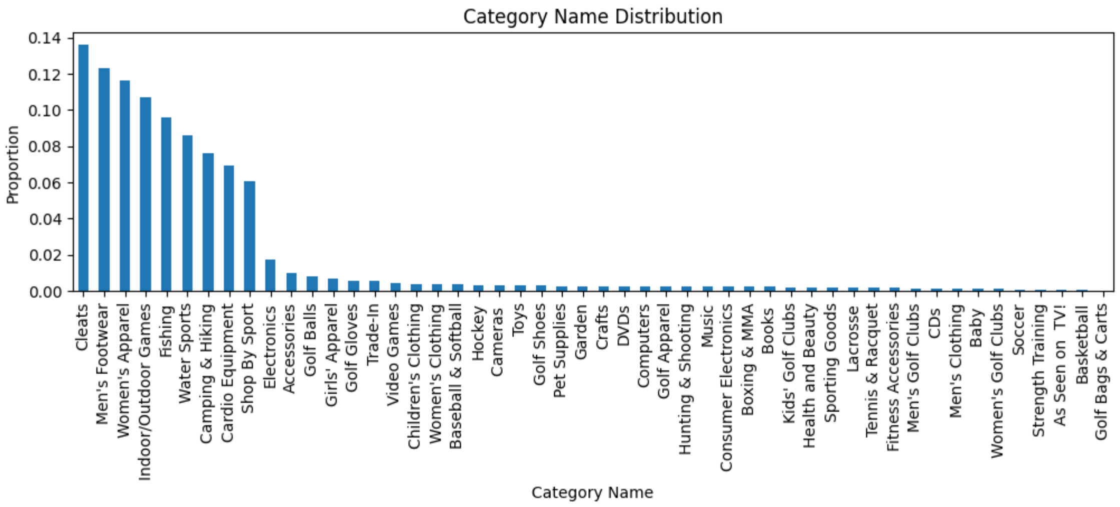
Data Imperfection

- **Example: Smart Supply Chain Dataset (DataCo)**






```
Unique Order Status values: ['COMPLETE' 'PENDING' 'CLOSED' 'PENDING_PAYMENT' 'CANCELED' 'PROCESSING'
 'SUSPECTED_FRAUD' 'ON_HOLD' 'PAYMENT_REVIEW']
Unique Customer Country values: ['Puerto Rico' 'EE. UU.']
Unique Order Region values: ['Southeast Asia' 'South Asia' 'Oceania' 'Eastern Asia' 'West Asia'
 'West of USA ' 'US Center ' 'West Africa' 'Central Africa' 'North Africa'
 'Western Europe' 'Northern Europe' 'Central America' 'Caribbean'
 'South America' 'East Africa' 'Southern Europe' 'East of USA' 'Canada'
 'Southern Africa' 'Central Asia' 'Eastern Europe' 'South of  USA ']
```



https://colab.research.google.com/github/Dr-AlaaKhamis/ISE518/blob/main/6_Data_imperfection/DataCo_imperfection.ipynb

Outline

- Introduction
- Data Imperfection
- **Structure Issues**
- Inconsistency
- Incompleteness
- Redundancy
- Imbalance
- Outlier

Data Structure

Examples	Header/column issues
Potential Impact	Analytics scripts may fail, Misaligned data
Mitigation Approach	Normalize headers, Remove quotes/newlines & Standardize names

```
['line',
 'MCCE\nquipment',
 'MCCDescription',
 'ProblemsItems',
 'Action',
 'jobcompleted',
 'Shift',
 'Month',
 'IssueDate',
 'EndDate',
 'Starttime',
 'Finishtime',
 'NetTime',
 'D.T Time',
 'R.T Time',
 'W O',
 'M C C',
 'PersonFinishJob',
 'SpareStatusandorigin-from',
 'SAPNo',
 'SAPCode',
 'SpareParts',
 'quantity',
 'LE/Uintes',
 'PMCM',
 'Reason']
```

```python
def clean_header(col: str) -> str:
    if not isinstance(col, str):
        col = str(col)
    col = col.replace('"', '')  # remove quotes
    col = col.replace("'", '')  # remove single quotes
    col = col.replace('\r', ' ').replace('\n', ' ')  # remove newlines
    col = re.sub(r'\s+', ' ', col)  # collapse whitespace
    col = col.strip().lower()  # trim + lowercase
    col = col.replace(' ', '_')  # spaces -> underscores
    # remove non-alnum/underscore except Arabic letters
    col = re.sub(r'[^0-9a-zA-Z_\u0600-\u06FF]', '', col)
    # collapse multiple underscores
    col = re.sub(r'_+', '_', col)
    return col

cleaned_cols = [clean_header(c) for c in df.columns]
cleaned_cols
```

```
['line',
 'mcce_quipment',
 'mccdescription',
 'problemsitems',
 'action',
 'jobcompleted',
 'shift',
 'month',
 'issuedate',
 'enddate',
 'starttime',
 'finishtime',
 'nettime',
 'dt_time',
 'rt_time',
 'w_o',
 'm_c_c',
 'personfinishjob',
 'sparestatusandoriginfrom',
 'sapno',
 'sapcode',
 'spareparts',
 'quantity',
 'leuintes',
 'pmcm',
 'reason']
```

https://colab.research.google.com/github/Dr-AlaaKhamis/ISE518/blob/main/6_Data_imperfection/data_structure.ipynb

Outline

- Introduction
- Data Imperfection
- Structure Issues
- **Inconsistency**
- Incompleteness
- Redundancy
- Imbalance
- Outlier

Data Consistency

Examples	Time/date issues, Inconsistent units, Inconsistent categories, Incorrect data types
Potential Impact	Misinterpretation of trends, calculation errors
Mitigation Approach	Standardize formats, Convert units, Enforce correct data types

	maintenance_id	equipment_name	equipment_type	last_maintenance	maintenance_interval	status	temperature	cost
0	1	Equip-1	pump	2023-13-45	199 days	active	53.9	6766.31
1	2	Equip-2	Pump	2025-02-20	845 HRS	maint	83.7	$8886
2	3	Equip-3	motor	2025-04-25	573 hours	Maintenance	78.9	1069.3
3	4	Equip-4	Motor	09/07/2025	195 hours	ACTIVE	58.9	$6670
4	5	Equip-5	Motor	2025-09-13	499 HRS	Active	181.1	$9205
5	6	Equip-6	PUMP	2025-06-09	403 hours	ACTIVE	151.4	665.94
6	7	Equip-7	pump	12/17/2024	261 hrs	active	79.5	$9849
7	8	Equip-8	VALVE	2023-13-45	982 hours	active	95.3	493.44
8	9	Equip-9	pump	2023-13-45	773 min	Maint	45.3	$5247
9	10	Equip-10	Valve	2023-13-45	505 hours	ACTIVE	111.2	$1288
10	11	Equip-11	Pump	2023-13-45	736 min	Maint	78.5	$4445
11	12	Equip-12	Pump	2023-13-45	457 hrs	Maint	121.2	$9000
12	13	Equip-13	pump	10/15/2024	641 hrs	Maintenance	147.9	NaN
13	14	Equip-14	Motor	2025-07-22	860 min	maint	42.5	$5717
14	15	Equip-15	Motor	2023-13-45	749 HRS	DOWN	40.1	NaN

https://colab.research.google.com/github/Dr-AlaaKhamis/ISE518/blob/main/6_Data_imperfection/data_consistency.ipynb

Data Consistency

Examples	Time/date issues, Inconsistent units, Inconsistent categories, Incorrect data types
Potential Impact	Misinterpretation of trends, calculation errors
Mitigation Approach	Standardize formats, Convert units, Enforce correct data types

last_maintenance
2023-13-45
2025-02-20
2025-04-25
09/07/2025
2025-09-13
2025-06-09
12/17/2024
2023-13-45
2023-13-45
2023-13-45
2023-13-45
2023-13-45
10/15/2024
2025-07-22
2023-13-45

```python
raw_dates = df['last_maintenance'].copy()
df['last_maintenance'] = pd.to_datetime(df['last_maintenance'], errors='coerce')

pd.DataFrame({'raw': raw_dates, 'parsed': df['last_maintenance']})
```

✓ 0.0s

raw	parsed
2023-13-45	NaT
2025-02-20	2025-02-20
2025-04-25	2025-04-25
09/07/2025	2025-09-07
2025-09-13	2025-09-13
...	...
04/21/2025	2025-04-21
07/06/2025	2025-07-06
2023-13-45	NaT
2023-13-45	NaT
04/16/2025	2025-04-16

```python
import re
def interval_to_hours(s):
    if pd.isna(s): return pd.NA
    s=str(s).lower().strip()
    m=re.search(r'(\d+(?:\.\d+)?)\s*(days|day|d|hours|hrs|hr|h)',s)
    if not m: return pd.NA
    val=float(m.group(1)); unit=m.group(2)
    return val*24 if unit in ['days','day','d'] else val

df['maintenance_interval_hours']=df['maintenance_interval'].apply(interval_to_hours)
df[['maintenance_interval','maintenance_interval_hours']]
```

✓ 0.0s

	maintenance_interval	maintenance_interval_hours
0	199 days	4776.0
1	845 HRS	845.0
2	573 hours	573.0
3	195 hours	195.0
4	499 HRS	499.0
...	...	...
95	894 hours	894.0
96	743 hrs	743.0
97	473 days	11352.0
98	235 min	<NA>

https://colab.research.google.com/github/Dr-AlaaKhamis/ISE518/blob/main/6_Data_imperfection/data_consistency.ipynb

Outline

- Introduction
- Data Imperfection
- Structure Issues
- Inconsistency
- **Incompleteness**
- Redundancy
- Imbalance
- Outliers

Incompleteness

- **Data Imputation**

Data imputation is a technique for handling missing values in a dataset by replacing them with estimated values to create a complete and usable dataset for analysis or modeling.

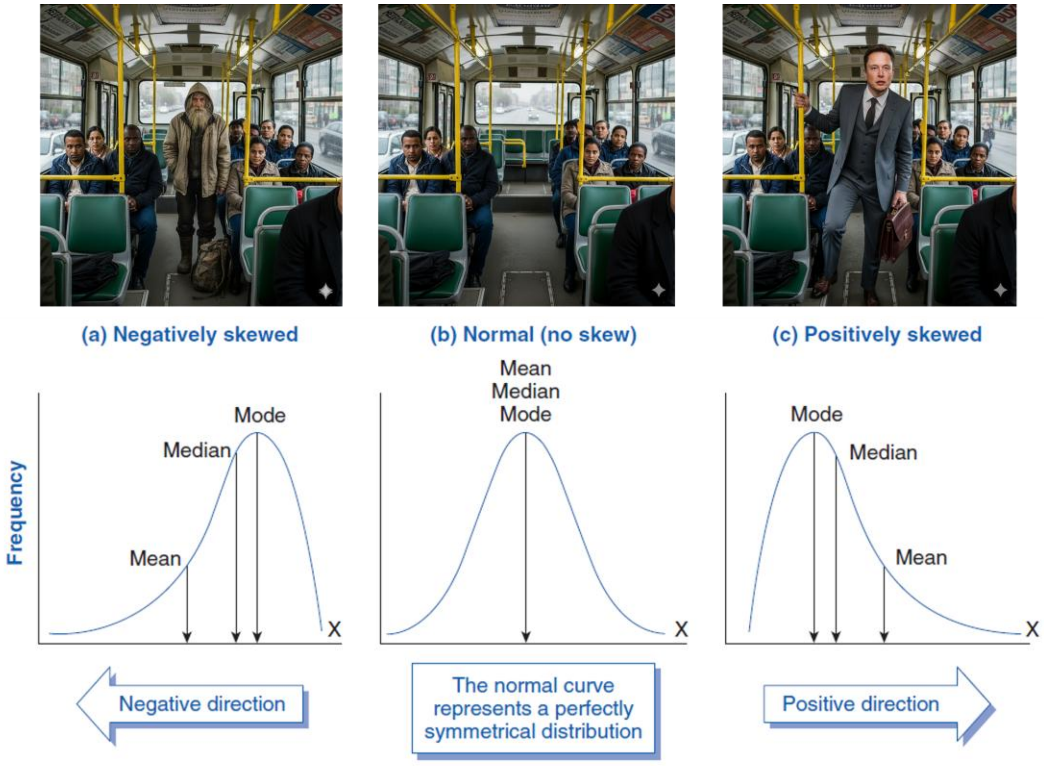
Original Data

F1	F2	F3	F4
3.4	100	65	32
4.0	?	85	?
1.3	110	?	56
?	103	43	63
7.8	198	77	45

Imputed Data

F1	F2	F3	F4
3.4	100	65	32
4.0	**105**	85	**44**
1.3	110	**64**	56
4.5	103	43	63
7.8	**198**	77	45

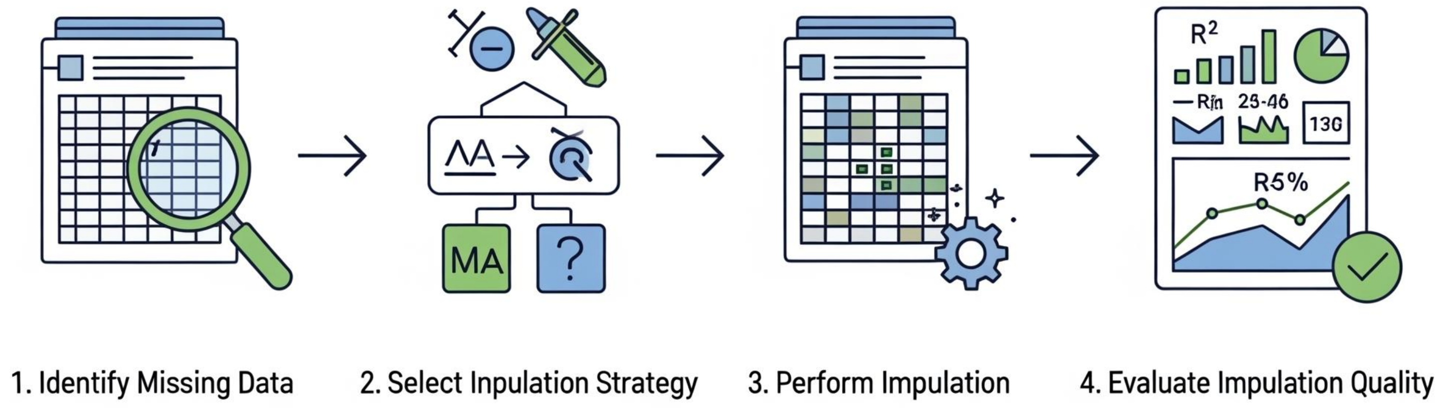
- **Data Imputation Methods**
  - **Mean/Median/Mode imputation:**
    - **Mean:** Best for numerical data that is normally distributed.
    - **Median:** Better for numerical data with outliers, as it is less sensitive to extreme values.
    - **Mode:** Used for categorical data, where the missing value is replaced by the most frequent category.



  - **Forward/Backward fill:** For time-series or ordered data, a missing value is replaced by the last or next observed value.
  - **Regression imputation:** Uses a regression model to predict the missing value of one variable based on other variables in the dataset.

Incompleteness

- **Performing Data Imputation**
  - **Identify missing data:** detect missing values in the dataset using tools in programming languages like Python (e.g., `isnull()` or `isna()` in pandas).
  - **Select an imputation strategy:** Choose a method based on the data type, the amount of missing data, the underlying missingness mechanism, and the desired level of accuracy.
  - **Perform the imputation:** Implement the chosen technique using a relevant software library.
  - **Evaluate imputation quality:** Check the quality of the imputed data by comparing its distribution to the original data and evaluating the performance of a downstream model



Incompleteness

- **Data Imputation Methods**

```python
import pandas as pd

# Sample DataFrame with missing values
df=pd.DataFrame({
    'A':[1,2,None,4],
    'B':[None,2,3,4],
    'C':[1,None,None,4]
})

df.head()
```

	A	B	C
0	1.0	NaN	1.0
1	2.0	2.0	NaN
2	NaN	3.0	NaN
3	4.0	4.0	4.0

```python
# Drop rows with any missing values
df_cleaned = df.dropna()
df_cleaned.head()
```

	A	B	C
3	4.0	4.0	4.0

```python
# Replace missing values with a specific value (e.g., 0)
df_filled = df.fillna(0)
df_filled.head()
```

	A	B	C
0	1.0	0.0	1.0
1	2.0	2.0	0.0
2	0.0	3.0	0.0
3	4.0	4.0	4.0

```python
import numpy as np

# Sample array with missing values
arr = np.array([[1, 2, np.nan], [4, np.nan, 6], [7, 8, 9]])
arr
```

```
array([[ 1.,  2., nan],
       [ 4., nan,  6.],
       [ 7.,  8.,  9.]])
```

```python
# Identify missing values
missing_mask = np.isnan(arr)
missing_mask
```

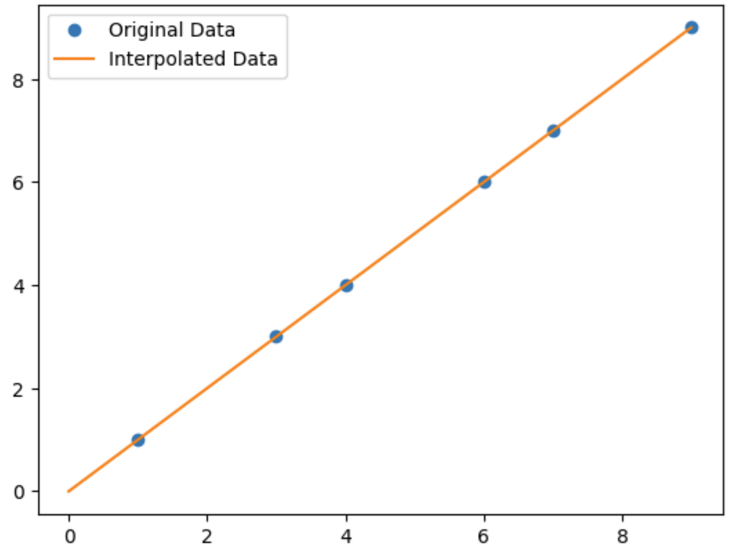
```
array([[False, False,  True],
       [False,  True, False],
       [False, False, False]])
```

```python
# Handle missing values by replacing them with a specific value (e.g., 0)
arr_filled = np.where(missing_mask, 0, arr)
arr_filled
```

```
array([[1., 2., 0.],
       [4., 0., 6.],
       [7., 8., 9.]])
```

```python
# Handle missing values by replacing them with the mean of the columns
col_means = np.nanmean(arr, axis=0)
```

```python
import numpy as np
from scipy.interpolate import interp1d
import matplotlib.pyplot as plt

# Sample array with missing values
x = np.array([0,1,2,3,4,5,6,7,8,9])
y=np.array([np.nan,1,np.nan,3,4,np.nan,6,7,np.nan,9])

# Interpolate to fill missing values
mask = ~np.isnan(y)
interp_func = interp1d(x[mask], y[mask], kind='linear', fill_value='extrapolate')
y_filled = interp_func(x)

# Plot original and filled data
plt.plot(x, y, 'o', label='Original Data')
plt.plot(x, y_filled, '-', label='Interpolated Data')
plt.legend()
plt.show()
```



https://colab.research.google.com/github/Dr-AlaaKhamis/ISE518/blob/main/6_Data_imperfection/data_imputation_1.ipynb

Incompleteness

- **Data Imputation: Maintenance Data**

	maintenance_id	equipment_name	equipment_type	last_maintenance	maintenance_interval	status	temperature	cost
0	1	Equip-1	pump	2023-13-45	199 days	active	53.9	6766.31
1	2	Equip-2	Pump	2025-02-20	845 HRS	maint	NaN	$8886
2	3	Equip-3	motor	2025-04-25	573 hours	NaN	78.9	1069.3
3	4	Equip-4	Motor	09/07/2025	195 hours	ACTIVE	58.9	$6670
4	5	Equip-5	Motor	2025-09-13	499 HRS	Active	181.1	NaN
5	6	Equip-6	PUMP	2025-06-09	403 hours	ACTIVE	151.4	665.94
6	7	Equip-7	pump	12/17/2024	261 hrs	active	79.5	$9849
7	8	Equip-8	VALVE	2023-13-45	982 hours	active	95.3	493.44
8	9	Equip-9	pump	2023-13-45	773 min	Maint	45.3	$5247
9	10	Equip-10	Valve	2023-13-45	505 hours	ACTIVE	111.2	$1288

```
demo.isna().sum()
```

```
maintenance_id          0
equipment_name          0
equipment_type          0
last_maintenance        0
maintenance_interval    0
status                  1
temperature             1
cost                   13
```

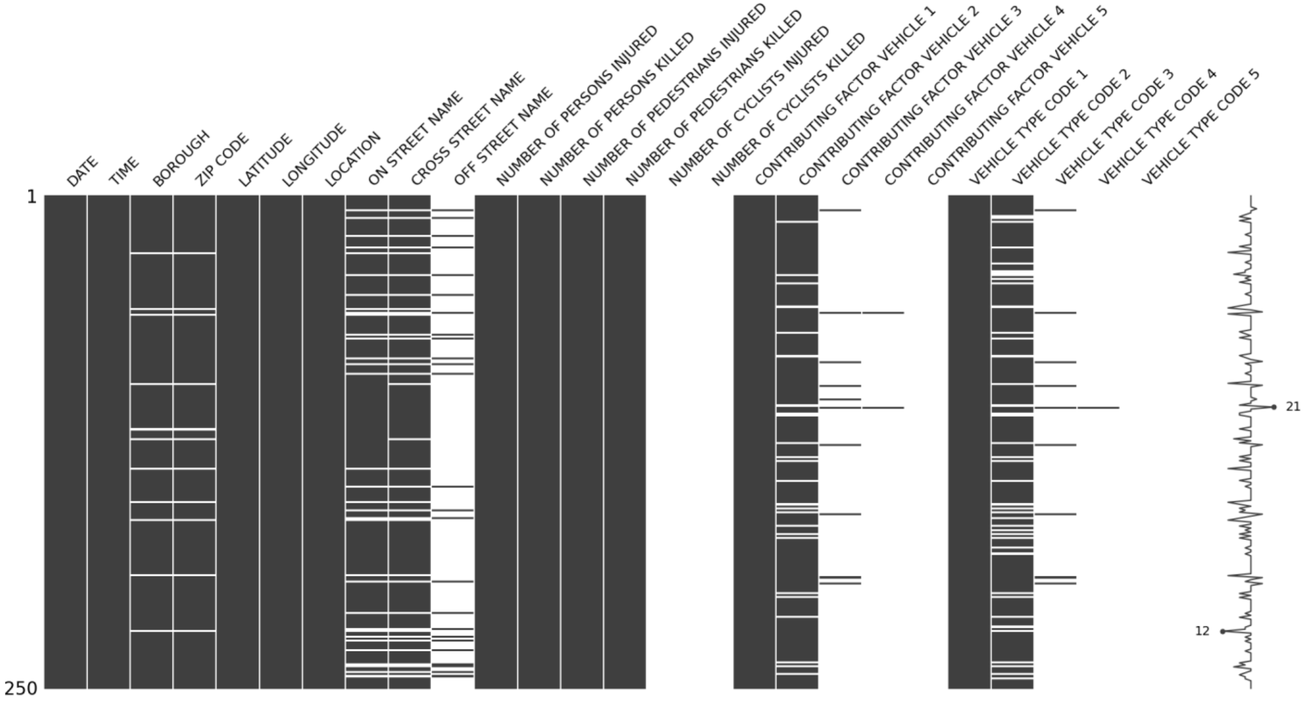
```
numeric_cols = demo.select_dtypes(include=['number']).columns
categorical_cols = [c for c in demo.columns if c not in numeric_cols]

# For numeric columns
for col in numeric_cols:
    if demo[col].isna().any():
        demo[col] = demo[col].fillna(demo[col].median())

# For categorical columns
for col in categorical_cols:
    if demo[col].isna().any():
        mode_val = demo[col].mode().iloc[0]
        demo[col] = demo[col].fillna(mode_val)
```

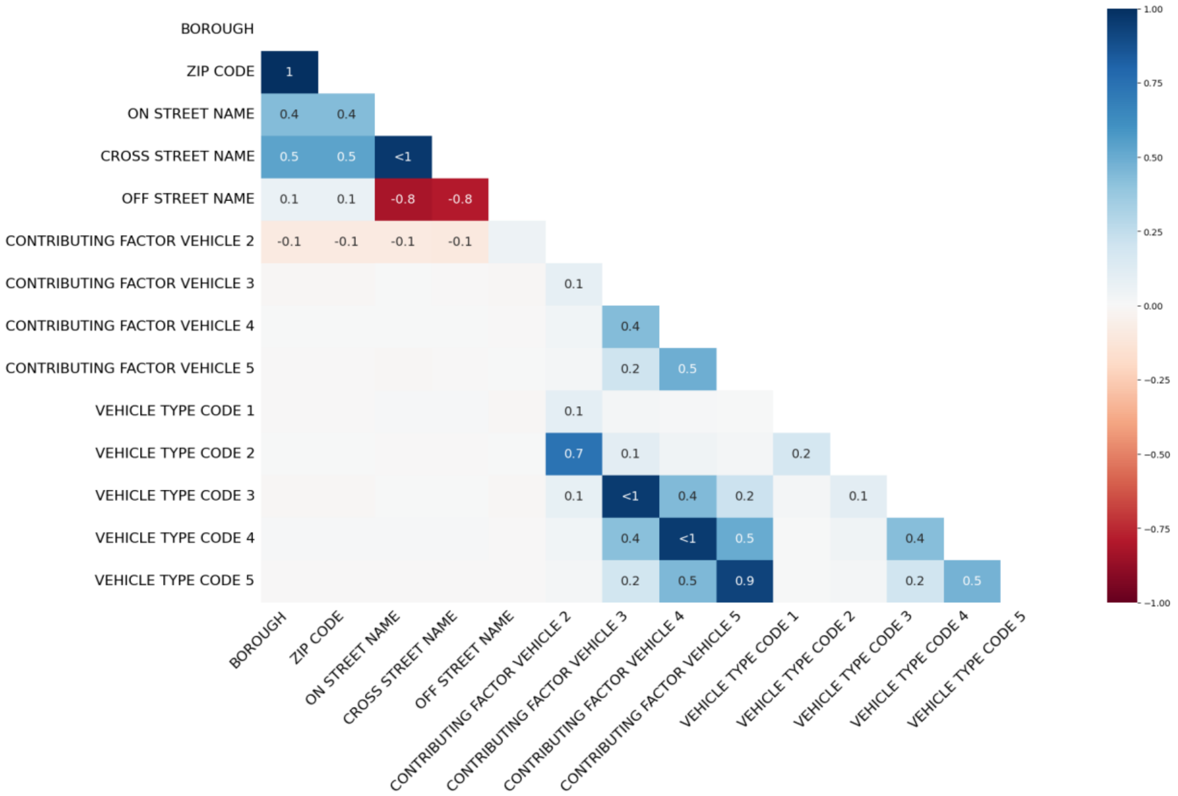
	maintenance_id	equipment_name	equipment_type	last_maintenance	maintenance_interval	status	temperature	cost
0	1	Equip-1	pump	2023-13-45	199 days	active	53.90	6766.31
1	2	Equip-2	Pump	2025-02-20	845 HRS	maint	81.95	$8886
2	3	Equip-3	motor	2025-04-25	573 hours	maint	78.90	1069.3
3	4	Equip-4	Motor	09/07/2025	195 hours	ACTIVE	58.90	$6670
4	5	Equip-5	Motor	2025-09-13	499 HRS	Active	181.10	6766.31
5	6	Equip-6	PUMP	2025-06-09	403 hours	ACTIVE	151.40	665.94
6	7	Equip-7	pump	12/17/2024	261 hrs	active	79.50	$9849
7	8	Equip-8	VALVE	2023-13-45	982 hours	active	95.30	493.44
8	9	Equip-9	pump	2023-13-45	773 min	Maint	45.30	$5247
9	10	Equip-10	Valve	2023-13-45	505 hours	ACTIVE	111.20	$1288

https://colab.research.google.com/github/Dr-AlaaKhamis/ISE518/blob/main/6_Data_imperfection/data_imputation_2.ipynb

Incompleteness

- **Data Imputation: NYC Collision Data**





https://colab.research.google.com/github/Dr-AlaaKhamis/ISE518/blob/main/6_Data_imperfection/data_imputation_3.ipynb

Outline

- Introduction
- Data Imperfection
- Structure Issues
- Inconsistency
- Incompleteness
- **Redundancy**
- Imbalance
- Outliers

- **Data Imputation: Maintenance Data**

```
# Create DataFrame with duplicate rows
df = pd.DataFrame({
    'A': [1, 2, 2, 4],
    'B': [5, 6, 6, 8],
    'C': [9, 10, 10, 12]
})
df.head()
```

	A	B	C
0	1	5	9
1	2	6	10
2	2	6	10
3	4	8	12

```
# Remove duplicate rows
df_no_duplicates = df.drop_duplicates()
df_no_duplicates.head()
```

	A	B	C
0	1	5	9
1	2	6	10
3	4	8	12

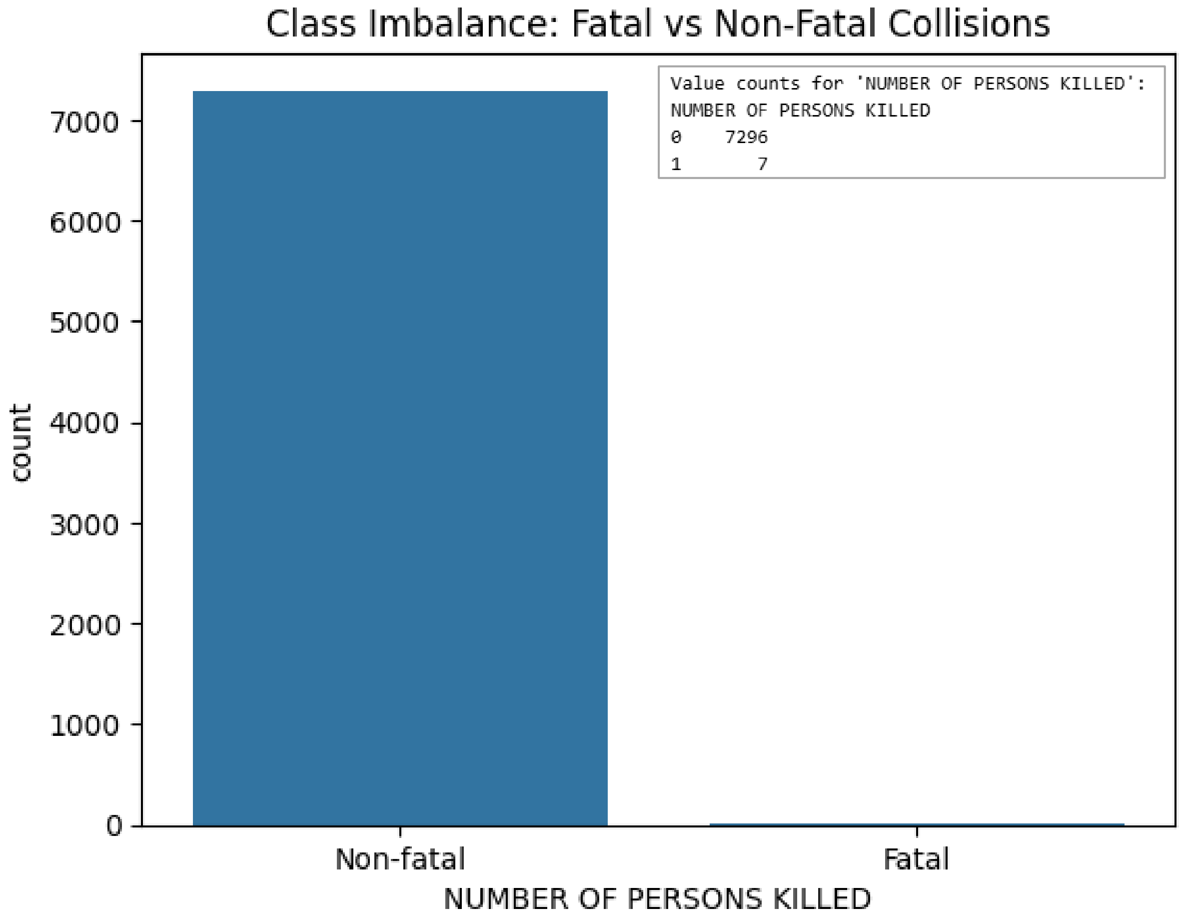
https://colab.research.google.com/github/Dr-AlaaKhamis/ISE518/blob/main/6_Data_imperfection/data_imputation_1.ipynb

Outline

- Introduction
- Data Imperfection
- Structure Issues
- Inconsistency
- Incompleteness
- Redundancy
- **Imbalance**
- Outliers

Imbalance

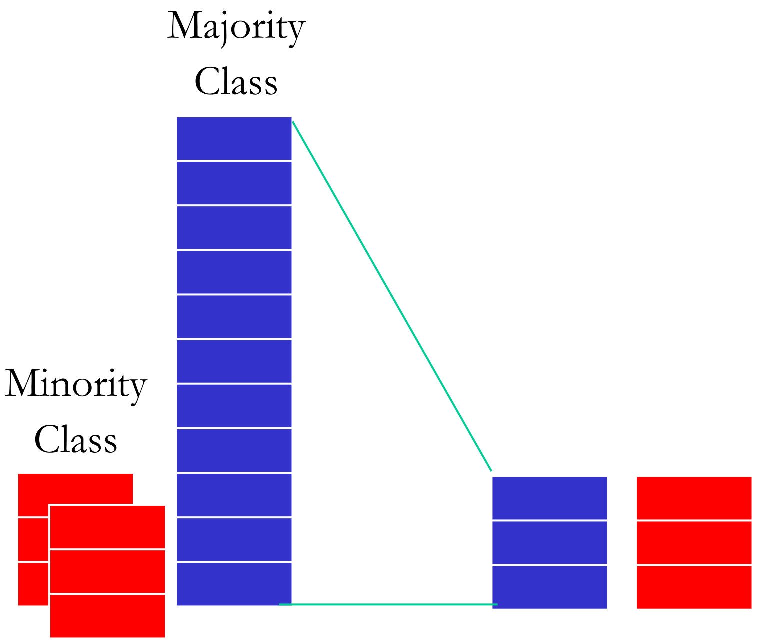
Imbalanced data occurs when certain classes or values appear much more frequently than others in a dataset. This can cause predictive models to be biased toward the majority class and perform poorly on the minority class, which is often of greater interest (such as fraud detection or rare diseases).



- One or more classes greatly outnumber others.
- Leads to biased or misleading model performance.
- Requires special techniques for detection and correction (e.g., resampling, balanced metrics).

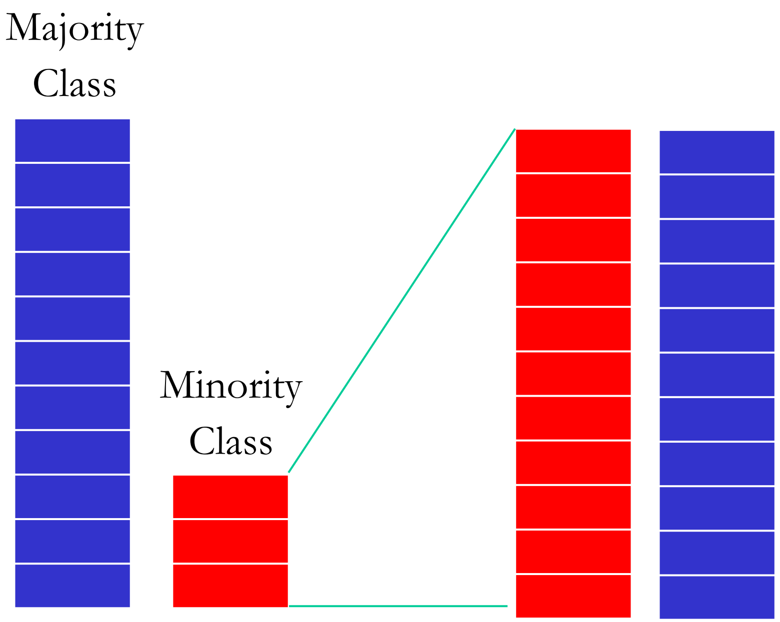
Imbalance

- **Data Sampling**



Under-sampling

In under-sampling, we reduce the number of observations from all classes but the minority class



Over-sampling

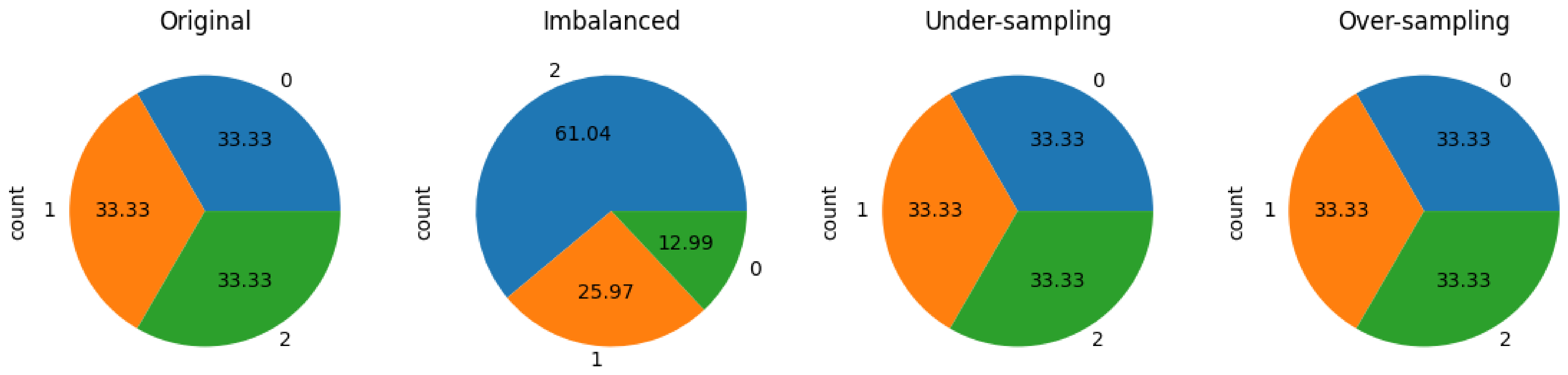
In over-sampling, we generate new samples in the classes which are under-represented

Imbalance

- **Data Sampling: Iris Dataset**

Iris dataset includes three iris species with 50 samples each as well as some properties about each flower. One flower species is linearly separable from the other two, but the other two are not linearly separable from each other.

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

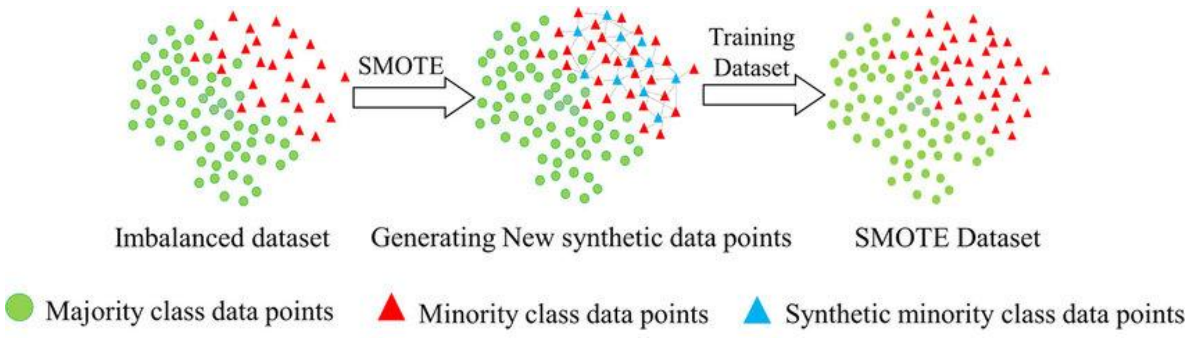


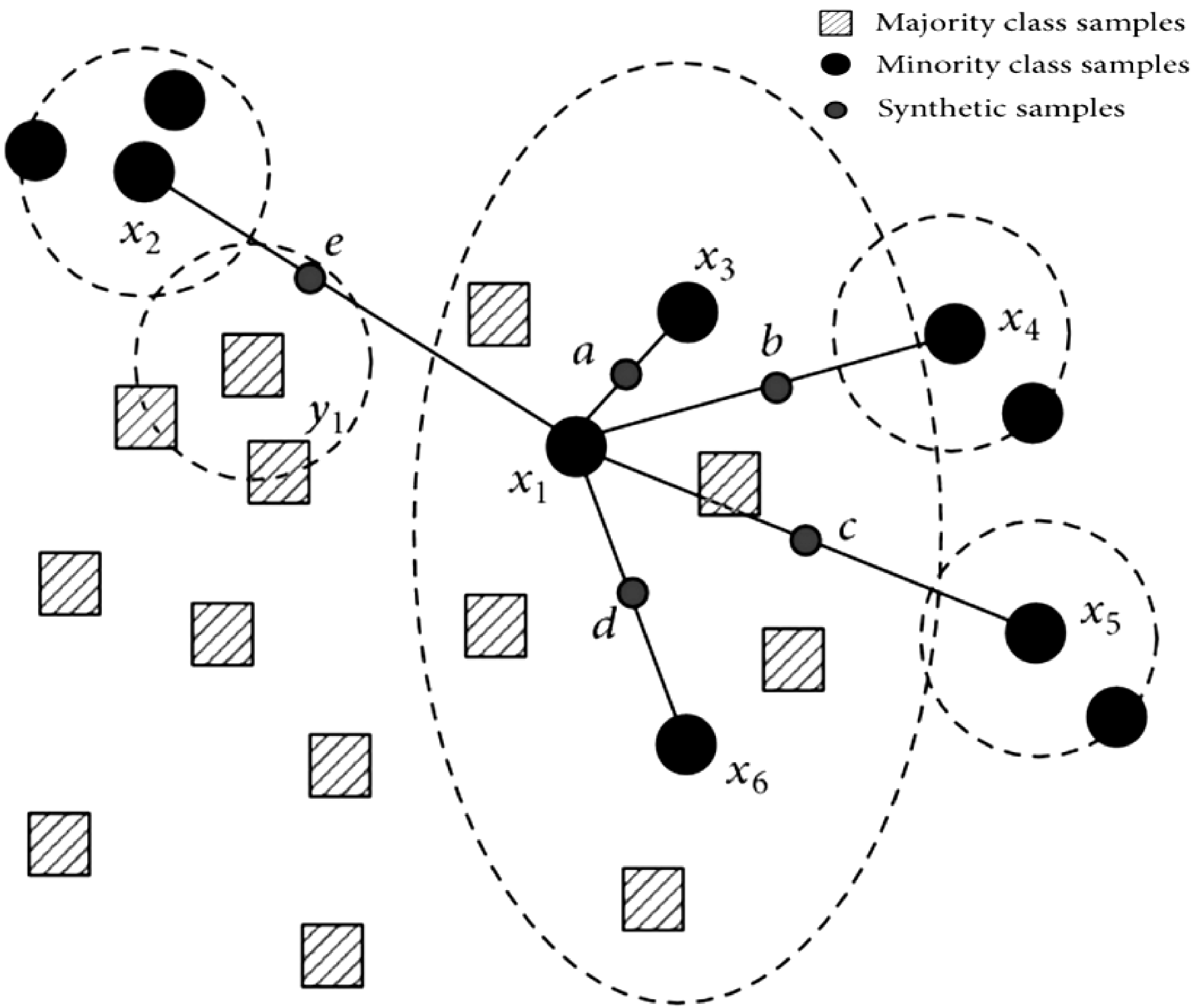
https://colab.research.google.com/github/Dr-AlaaKhamis/ISE518/blob/main/6_Data_imperfection/data_integrity_1.ipynb

Imbalance

- **Sampling using SMOTE**

  Using SMOTE, the minority class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors. Depending upon the amount of over-sampling required, neighbors from the k nearest neighbors are randomly chosen.





SMOTE (Synthetic Minority Over-sampling Technique)

Imbalance

- **Handling data imbalance using SMOTE**

```
from imblearn.over_sampling import SMOTE

# Create binary target: 1 for fatal, 0 for non-fatal
collisions["fatal"] = (collisions["NUMBER OF PERSONS KILLED"] > 0).astype(int)

# Use correct feature names
X = collisions[[
    "NUMBER OF PERSONS INJURED",
    "NUMBER OF PEDESTRIANS INJURED",
    "NUMBER OF CYCLISTS INJURED"  # <-- fixed here
]].fillna(0)
y = collisions["fatal"]

print("Original class distribution:")
print(y.value_counts())

# Apply SMOTE
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)

print("\nAfter SMOTE class distribution:")
print(pd.Series(y_resampled).value_counts())
```

```
Original class distribution:
fatal
0    7296
1       7
Name: count, dtype: int64

After SMOTE class distribution:
fatal
0    7296
1    7296
Name: count, dtype: int64
```

https://colab.research.google.com/github/Dr-AlaaKhamis/ISE518/blob/main/6_Data_imperfection/data_integrity_2.ipynb

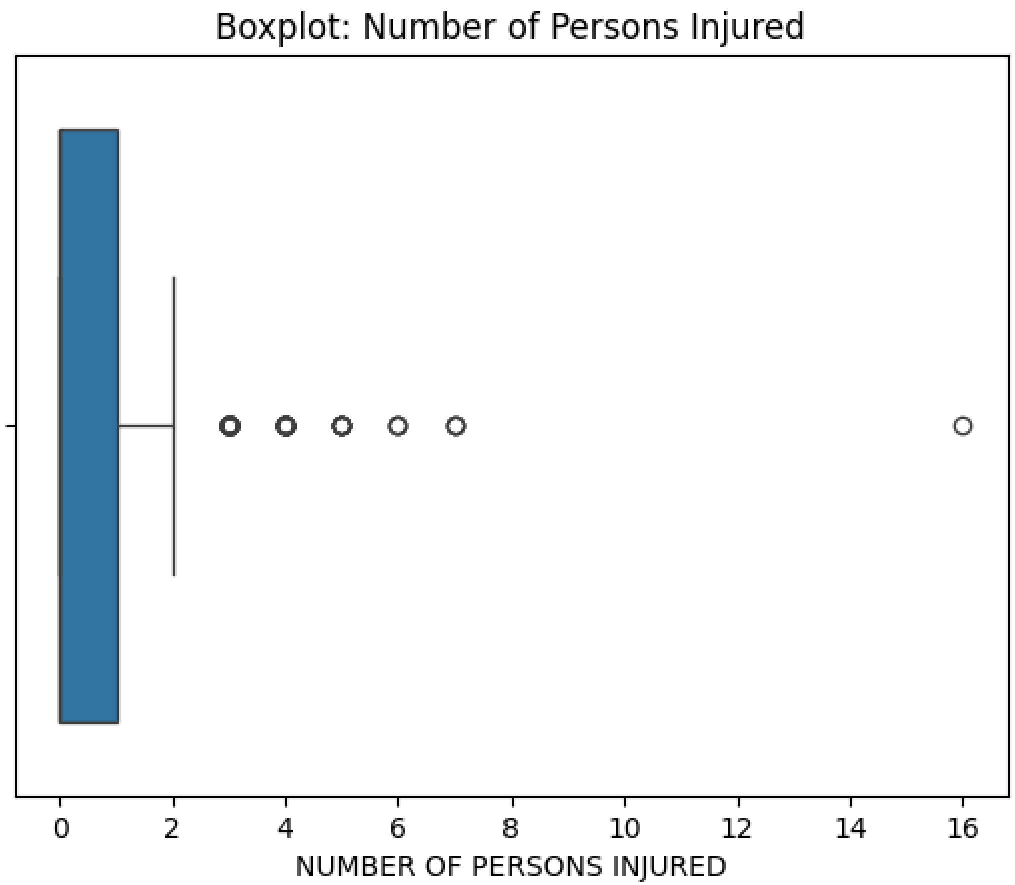
Outline

- Introduction
- Data Imperfection
- Structure Issues
- Inconsistency
- Incompleteness
- Redundancy
- Imbalance
- **<u>Outliers</u>**

Outliers

Outliers are data points that differ significantly from other observations. They may result from data entry errors, measurement variability, or genuine rare events. Outliers can distort statistical analyses and model training if not properly addressed.

- Values much higher or lower than most other data.
- Can indicate errors or rare, important cases.
- Affect summary statistics and model training; may require detection and handling.




```
Rows with most persons injured:
      NUMBER OF PERSONS INJURED  ZIP CODE  CONTRIBUTING FACTOR VEHICLE 1
2626                         16   11435.0  Failure to Yield Right-of-Way
4797                          7       NaN  Failure to Yield Right-of-Way
696                           7   10454.0  Failure to Yield Right-of-Way
6940                          7   11208.0  Failure to Yield Right-of-Way
3046                          6   11434.0  Failure to Yield Right-of-Way
```