Data Analysis Pipeline

1. Introduction

1. **Understand the Problem:** Start by clearly defining the objective of your analysis. Know what questions you're trying to answer and what kind of decision or action will be based on your findings.
2. **Collect the Data:** Once the objective is clear, gather the necessary data. This could come from CSV files, databases, online datasets, APIs, or through scraping websites. Make sure the data is relevant to your problem.
3. **Inspect the Data:** Examine the structure of the dataset. Look at how many rows and columns there are, what types of variables you're dealing with, and get a basic sense of data quality.
4. **Clean the Data:** Handle missing values, remove duplicates, fix inconsistent formatting, and ensure that all data types are correct. Clean data is essential for accurate analysis.
5. **EDA (Exploratory Data Analysis):** Use descriptive statistics and visualizations to understand the underlying patterns, relationships, and distributions in the data. This step often reveals insights or issues that weren't obvious before.
6. **Transform or Engineer Features:** If needed, create new variables that can help in the analysis. This might involve combining columns, categorizing data, or normalizing values for better comparison.
7. **Analyze the Data using visualization:** Now apply the actual analysis methods. This could include calculating correlations, comparing groups, identifying trends over time, or summarizing results using groupings.

---

2. Types of Data:

Quantitative data: deals with measurable quantities and can be further divided into:
1. Discrete data: Takes on distinct, separate values. Example: number of students (0, 1, 2, …). Mathematically, these map to integers ($\mathbb{Z}$).
2. Continuous data: Can take any value within a range. Example: height, weight, temperature. Formally, this maps to $\mathbb{R}$ (real numbers).

Qualitative data: describes attributes and characteristics and is subdivided into:
1. Nominal: Categories with no inherent order. Example: gender, country, color.
2. Ordinal: Categories with a meaningful order but unknown intervals. Example: education level (High School < Bachelor < Master), Likert scales (Strongly Agree, Agree, etc.).

Statistical modeling implications:
1. Discrete and nominal data may require encoding.
2. Continuous variables often need normalization or scaling.
3. Ordinal data can be encoded to preserve ranking (e.g., education level = {0,1,2}).

---

3. Encoding Data:

3.1 Label Based Encoding:

Assigns each category a unique integer. While simple, it assumes an ordinal relationship, which can mislead models if applied to nominal data.

3.2 One-Hot Encoding:

Converts each category into a binary vector. This is ideal for nominal variables but increases dimensionality—known as the "curse of dimensionality."

3.3 Integer Encoding:

Integer encoding is a categorical variable encoding technique where each category label is assigned a unique integer value. It is a simple and memory-efficient approach to one hot-encoding. For Example)

City	Encoded
Mumbai	0
Delhi	1
Bengaluru	2
Mumbai	0

3.3.1 Integer encoding is most useful when:

1. The categorical variable has high cardinality (e.g., thousands of zip codes, user IDs).

2. The model you're using supports embeddings or can learn arbitrary mappings (e.g., deep learning models).
3. You intend to pass the encoded variable into an embedding layer for dense vector representation.
4. Examples: Neural networks (e.g., feedforward, CNN, RNN, Transformers), Gradient Boosting Trees (e.g., CatBoost can use raw categorical features directly with encoding optimizations).

3.3.2 Pitfalls of Integer Encoding:

- Models like Logistic Regression, Linear Regression, or Decision Trees will interpret the integers as ordinal, i.e., with inherent order and magnitude. This creates spurious relationships unless the categorical variable is truly ordinal.
- Example: Encoding ["red", "green", "blue"] as [0, 1, 2] makes "green" appear to lie halfway between "red" and "blue" — which is semantically incorrect.
- This problem is especially harmful in distance-based models (like k-NN, SVMs), where the integer differences affect Euclidean or cosine distances.
- Case Study: In a customer churn prediction model, using integer encoding for “Geography” (India=0, France=1, Germany=2) caused SVM to learn that France lies between India and Germany, creating nonsensical decision boundaries.

---

4. Handling Missing Data:

- Data can often contain values because of some or the other reason

4.1 Missing Quantitative Data:

- Missing values in numeric features can cause major errors during training. The approach depends on the amount and randomness of the missing data.
- Let X be a variable and n be the number of missing values.
- Dropping rows (listwise deletion): If (n / total rows) < 0.05 and the feature is not critical, removing those rows is acceptable. This assumes Missing Completely At Random (MCAR) distribution.
- Mean/Median/Mode Imputation:
    - Mean: $\mu = (\Sigma x_i) / N$, sensitive to outliers, best for symmetric distributions.

    - Median: Robust against outliers, suitable for skewed data.
    - Mode: Used for discrete numerical values.
- Mathematical Note: The imputation method selected should minimize the Mean Squared Error (MSE) of the feature distribution after imputation.
- Advanced: KNN Imputation (based on feature similarity), Regression Imputation, MICE (Multiple Imputation by Chained Equations)

4.2 Missing Qualitative Data:

Categorical missing values are often handled differently:

- Most frequent category (mode) imputation.
- Introduction of a “Missing” category.
- Predictive imputation (e.g., train a classifier to predict missing labels).
- Frequency-based imputation.
- Application: Real-world use: Netflix often uses **latent matrix factorization** to infer missing preferences in user-item matrices (see "Matrix Completion for Netflix Challenge").

4.3 Algorithms Insensitive to Missing data:

Some machine learning models can handle missing values during training:

- Decision Trees & Random Forest: These models can split on "is null" or handle missing as a separate category.
- XGBoost/LightGBM: These gradient boosting libraries have built-in mechanisms to handle NA by learning optimal splits for missing data.
- Algorithms like Logistic Regression, kNN, and SVM require complete data matrices ($X \in \mathbb{R}^{nxd}$), hence missing values must be imputed.

---

5. Outliers & Noise:

- **Outliers:** Data points significantly different from others but often valid. Example: a person with income ₹10 crores/year in a salary dataset.
- **Noise:** Random errors or variability in data that obscure the underlying signal. Often caused by faulty sensors, transmission errors, or misentry.

Why to Detect Outliers:

- Can skew statistical summaries like mean, median and mode. Ex) If one was to prepare a Gaussian Distribution for the height of people in a sample space, and someone was exceptionally tall like 7'2", the data is valid and not noise, but it is an outlier and can greatly affect the mean.
- Outliers, hence help us to find rare phenomena.
- Many Machine learning models are very sensitive to outliers.

5.1 Checking Outliers:

5.1.1 Z-Score:

- **Formula:**
  $$Z_i = \frac{x_i - \mu}{\sigma}$$
- x_i = value of the observation
- μ = mean of the feature
- σ = standard deviation
  Example: In a dataset of heights with μ = 170 cm, σ = 10 cm, a value of 200 cm gives:
  Z=200−17010=3⇒potential outlierZ = \frac{200 - 170}{10} = 3 \Rightarrow \text{potential outlier}Z=10200−170=3⇒potential outlier
- **Use When:** The data is approximately Gaussian and Simple, fast filtering is required
- **Limitations:** Fails for skewed or non-normal distributions. Sensitive to existing outliers (μ and σ get distorted)

5.1.2 IQR Method

- The Interquartile Range (IQR) method is based on quartiles.
- **Formula:**
  Q1 = 25th percentile
  Q3 = 75th percentile
  IQR = Q3 − Q1
  Outlier thresholds:
  Lower Bound = Q1 − 1.5 × IQR
  Upper Bound = Q3 + 1.5 × IQR

Any data point outside [Q1 − 1.5×IQR, Q3 + 1.5×IQR] is considered an outlier.

- Use When: Data is non-Gaussian or skewed and you want a robust, non-parametric method
- Visualization: Box plots show median, quartiles, and outliers as individual points.

Other Advanced Methods are Isolation Forest, DBS-Scan and M-Estimators

---

6. Normalization Techniques

Feature scaling ensures all features contribute equally to the distance calculations in models like kNN and SVM. Normalization is very important as each data needs to be represented equally. If some value in the data set has very high numerical values, it might be interpreted as more important by models. Hence, Normalization is necessary for unbiased models.

Sometimes the features of our data have vastly different scales. This will cause the learning algorithm to give more importance to certain features, reducing its performance. Data normalization is a method in which we transform the features so that they have similar scales.

Three commonly used feature scaling techniques are rescaling, mean normalization and z-score normalization. Here, we will talk about the simplest one: rescaling.

6.1 Min-max Standardization

$$x' = \frac{x - min(x)}{max(x) - min(x)}$$

All 2D linear transformations can be repreented by a transformation matrix. So what is the matrix associated with the rescaling function? Actually, we cannot represent rescaling with a matrix multiplication, because it is not a linear transform. Rescaling involves shifting the origin of the data, which is not allowed under linear transformations.

We can represent rescaling as a matrix multiplication followed by a vector addition. Let our first feature vector be called X and second feature vector be called Y. Suppose we want to rescale a data point [a,b]

$$\begin{Bmatrix} a' \\ b' \end{Bmatrix} = \begin{Bmatrix} \frac{a - min(X)}{max(X) - min(X)} \\ \frac{b - min(Y)}{max(Y) - min(Y)} \end{Bmatrix} = \begin{Bmatrix} \frac{1}{max(X) - min(X)} & 0 \\ 0 & \frac{1}{max(Y) - min(Y)} \end{Bmatrix} \begin{Bmatrix} a \\ b \end{Bmatrix} + \begin{Bmatrix} \frac{-min(X)}{max(X) - min(X)} \\ \frac{-min(Y)}{max(Y) - min(Y)} \end{Bmatrix}$$

Note

Both (Z-Score and Min-Max) techniques rely on global statistics (min, max, mean, std). Outliers inflate the range or standard deviation, compressing the majority of the data to a small interval, which degrades learning in distance-based or gradient-based models.

6.2 Clipping, Capping, Log Transform

- Feature Clipping: x = min(max(x, lower_bound), upper_bound)
- Winsorizing: Replace extreme percentiles with cutoffs (e.g., 1st and 99th).
- Log Transform: log(x + 1) reduces right-skewed distributions, common in count data.
