Explore Data Processing: A critical step in any machine learning or data science pipeline. Its goal is to clean, transform, and organize raw data into a suitable format for modeling.

1. Feature Scaling: It is essential when features in a dataset have different units or magnitudes. Many machine learning algorithms (especially those based on distance, like KNN or SVM) assume that all features contribute equally.
2. Standardization (Z-score Normalization): Rescales data to have a mean of 0 and a standard deviation of 1.

When to use:

* When the distribution is **not bounded** (can have outliers).
* Common with **linear regression**, **logistic regression**, **SVM**, **PCA**.

Ex:

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaled\_data = scaler.fit\_transform(data)

1. Normalization (Min-Max Scaling): Scales values to a fixed range, typically [0, 1].

When to use:

* When you know the **range of your data**.
* Often used in **neural networks**.

Ex:

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

normalized\_data = scaler.fit\_transform(data)

2) Encoding Techniques: Many machine learning models can only work with numeric data, so **encoding categorical variables** is essential.

1. Label Encoding:Converts each category to a unique integer.

**Limitation:** Imposes ordinal relationship where it might not exist.

Ex:

Red -> 0

Green -> 1

Blue -> 2

from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()

encoded = encoder.fit\_transform(data['Color'])

1. One-Hot Encoding:Converts each category into a new binary column.  
   **Advantage:** No ordinal relationship is implied.

Ex:

Color -> Red, Green, Blue

Red => [1, 0, 0]

Green => [0, 1, 0]

Blue => [0, 0, 1]

import pandas as pd

encoded\_df = pd.get\_dummies(data, columns=['Color'])

1. Ordinal Encoding: Categorical data with a meaningful order (e.g., education level).

Ex:

from sklearn.preprocessing import OrdinalEncoder

encoder = OrdinalEncoder(categories=[['Low', 'Medium', 'High']])

data[['Education']] = encoder.fit\_transform(data[['Education']])

1. Target/Mean Encoding (Caution: Can lead to leakage): Replaces categories with the mean of the target for each category.

**Needs Cross-validation** to avoid data leakage.

**Used In:** High-cardinality categorical variables (e.g., zip codes).

3) Imputation Techniques: Missing data can bias your model if not handled properly. Imputation refers to replacing missing values with estimated ones.

1. Simple Imputation:

**Mean / Median / Mode Imputation**

Best for numerical (mean/median) or categorical (mode) data.

Ex:

from sklearn.impute import SimpleImputer

imputer = SimpleImputer(strategy='mean') # or 'median', 'most\_frequent'

data\_imputed = imputer.fit\_transform(data)

1. K-NN Imputation: Uses **K-Nearest Neighbors** to impute missing values based on similarity with other samples.

Ex:

from sklearn.impute import KNNImputer

imputer = KNNImputer(n\_neighbors=5)

data\_imputed = imputer.fit\_transform(data)

1. Multivariate Imputation: More advanced: models each feature with missing values as a function of other features.

Ex:

from sklearn.experimental import enable\_iterative\_imputer

from sklearn.impute import IterativeImputer

imputer = IterativeImputer()

data\_imputed = imputer.fit\_transform(data)

4) Feature Scaling: Feature selection helps to reduce overfitting, improve model performance, and speed up training.

1. Filter Method: Use statistical tests independent of the model.

Examples:

* Pearson correlation
* Chi-square test
* ANOVA F-test

1. Wrapper Method: Use a predictive model to evaluate combinations of features.

Examples:Recursive Feature Elimination (RFE)

Ex:

from sklearn.feature\_selection import RFE

from sklearn.linear\_model import LogisticRegression

rfe = RFE(estimator=LogisticRegression(), n\_features\_to\_select=5)

X\_new = rfe.fit\_transform(X, y)

1. Embedded Method: Perform feature selection as part of the model training.

Examples:

* Lasso (L1 regularization)
* Tree-based models (feature importances)

Ex:

from sklearn.linear\_model import Lasso

model = Lasso(alpha=0.01)

model.fit(X, y)

importance = model.coef\_

3) Handling Duplicates: Duplicates can distort statistical summaries and introduce bias.

1. Deleting Duplicates:

Ex:

duplicates = data.duplicated()

print(f"Number of duplicates: {duplicates.sum()}")

1. Removing Duplicates:

Ex:

data\_cleaned = data.drop\_duplicates()

1. Subset-Based Duplicates:

Ex:

data.drop\_duplicates(subset=['Name', 'Age'], keep='first', inplace=True)