Data Preprocessing :

1. Data cleaning
2. Handling missing data
3. Encoding categorical data
4. Detecting and handling outliers
5. Handling skewed data
6. Discretization
7. Scaling and normalization

- Feature selection and extraction are considered separate steps from data preprocessing
- the data should be normalized only after it is cleaned and the missing values and outliers have been handled.
- In supervised learning tasks, data preprocessing is usually performed only on the features and not on the target labels.

Data Cleaning :

Data cleaning (or cleansing) involves correcting or removing incorrect, inaccurate, inconsistent, irrelevant, or duplicate data from the data set. These issues can arise from various sources, such as:

- Data entry errors (e.g., an invalid postal code, typographical errors)
- Out-of-range values (e.g., a negative product price)
- Corruption in transmitting or storage of the data
- Merging ambiguous data from different sources (e.g., the same customer was stored in two systems with two different addresses)
- Using inconsistent formats for dates, phone numbers, names of states, etc.
- Using inconsistent unit measures (e.g., using both centimeters and feet to measure length)
- Including features that are irrelevant to the analysis, such as user id.

- Many machine learning algorithms cannot deal with missing values (exceptions include KNN, Naive Bayes, and decision trees)

Missing values :

Scikit-Learn provides three types of imputers:

1. SimpleImputer imputes the missing values using the statistics (e.g., mean, median, or mode) of the feature with the missing values or using a constant value.

   Its important parameters are:

- *missing_values* — which values are considered to be missing values (defaults to np.nan).
- *strategy* — the statistic to use for the imputation. The options are “mean” (the default), “median”, “most_frequent” and “constant”. For categorical features, only the options “most_frequent” and “constant” can be used.

- *fill_value* — which constant to use for replacing the missing values (when the chosen strategy is “constant”).

2. IterativeImputer models each feature with missing values as a function of the other features, in a round-robin fashion. In each iteration, one of the features with missing values is designated as the output y, and the other features are treated as the inputs X. Then, a regression model is trained on (X, y), and used to predict the missing values of y. This process is repeated for *max_iter* imputation rounds.

Important parameters of this transformer:

- *estimator* — the estimator to use for the imputation (the default is BayesianRidge).
- *max_iter* — maximum number of imputation rounds (defaults to 10).

- *initial_strategy* — which strategy to use to initialize the missing values (same as the *strategy* parameter in SimpleImputer).

- *imputation_order* — the order in which the features will be imputed. Defaults to 'ascending', i.e., from features with the fewest missing values to the most.

Since this transformer is still experimental, before using it you need to explicitly import *enable_iterative_imputer*

3. KNNImputer imputes the missing values by using the mean value of the *k*-nearest neighbors that have a value for the missing feature.

Important parameters of this transformer:

- *n_neighbors* — the number of neighbors to use for the imputation (defaults to 5)

- *weights* — whether to weight the neighbors uniformly (the default) or by the inverse of their distance.
- *metric* — the metric to use for computing the distances. Possible values are ‘nan_euclidean’ (an Euclidean distance metric that supports missing values) or a custom function.

Outliers :

1. Density-based clustering methods, such as DBSCAN, can identify outliers based on the density of the data points.
2. Isolation forest is an ensemble-based approach for anomaly detection.

There are also different ways to handle outliers, depending on the nature and extent of the outliers:

1. Remove the outliers. This is the simplest approach, but it should be done judiciously, as it can potentially lead to information loss.

2. Treat the outliers as missing values and then use one of the aforementioned imputation methods to replace them.
3. Capping sets a predefined threshold for extreme values. Any data point that exceeds the threshold is replaced with the threshold value.
4. Winsorization sets all the outliers to a specified percentile of the data. For example, a 90% winsorization replaces all the data points above the 95th percentile with the 95th percentile, and all the points below the 5th percentile with the 5th percentile. This approach limits the impact of outliers without completely removing them.
5. Use discretization to group the data points into bins, and assign the outliers to a separate bin or to the nearest bin.

Skewness Handling :

For power transformations, you can use the class PowerTransformer from Scikit-Learn, which currently provides two transformations:

1. Box-Cox transform, which works only with strictly positive values:

$$x' = \begin{cases} \frac{x^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \ln(x) & \text{if } \lambda = 0 \end{cases}$$

2. Yeo-Johnson transform, which works with any real value:

$$x' = \begin{cases} \frac{(x+1)^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0, x \geq 0 \\ \ln(x+1) & \text{if } \lambda = 0, x \geq 0 \\ -\frac{(x+1)^{2-\lambda} - 1}{2-\lambda} & \text{if } \lambda \neq 2, x < 0 \\ -\ln(-x+1) & \text{if } \lambda = 2, x < 0 \end{cases}$$

Discretization :

Discretization transforms a continuous-valued feature into a discrete one by partitioning the range of its values into a set of intervals or bins. The two main methods for discretization are:

1. Equal width/binning: the range of the variable is divided into equal-width bins. For example, if the range of the variable is 0–20 and we want 5 bins, then each bin will cover a range of 4 units (0–4,4–8,8–12,12–16,16–20).
2. Equal frequency: each bin contains the same number of data points.

Use cases for discretization:

1. Some machine learning algorithms cannot handle continuous values directly, such as some variants of Naive Bayes and the Apriori algorithm for association rule mining.
2. Discretization can make the model more expressive since it allows the model to find a mapping between each interval and the target label. For example, imagine that we need to predict the price of a house given its location, represented by its latitude and longitude. If we use a linear regression model, it can

only find a linear correlation between the exact location of the house and its price. However, if we discretize the latitude and longitude into 10 bins each, the model can find a linear correlation between each one of the 100 areas and the price of the house.

3. Handle outliers or extreme values by placing them in their own category.

The drawback of discretization is that it can lead to a loss of information, and may introduce bias if the number of bins is too small or their edges are not properly chosen, you can use KBinsDiscretizer to perform discretization.

Reduces dimensionality:

Data preprocessing techniques such as principal component analysis (PCA) can be used to reduce the dimensionality of data, making it easier to analyze or model.

Data selection :

Data selection is the process of selecting a subset of data from a larger dataset based on certain criteria. It is an important step in data preprocessing that can help to reduce the size of the dataset and focus on relevant data for analysis or modeling. There are several techniques used for data selection, including:

1. **Random sampling:** Random sampling involves selecting a random subset of data from the larger dataset. This is useful when the dataset is too large to process as a whole and a representative sample is needed.
2. **Stratified sampling:** Stratified sampling involves dividing the dataset into subgroups based on a specific variable and then selecting a random sample from each subgroup. This is useful when the variable is important for analysis or modeling.

3. **Feature selection:** Feature selection involves selecting a subset of features from the dataset based on their relevance to the analysis or modeling task. This is useful for reducing the dimensionality of the dataset and improving the performance of the model.
4. **Instance selection:** Instance selection involves selecting a subset of instances from the dataset based on their relevance to the analysis or modeling task. This is useful for reducing the size of the dataset and focusing on relevant data.

Data Merging and Joining :

When working with multiple datasets, it is often necessary to merge or join them together based on a common column or key. Pandas provides several methods for merging and joining data, including:

- `merge()`: merges two DataFrames based on a common column or key.
- `join()`: joins two DataFrames based on their indices.
- `concat()`: concatenates multiple DataFrames along a specified axis.