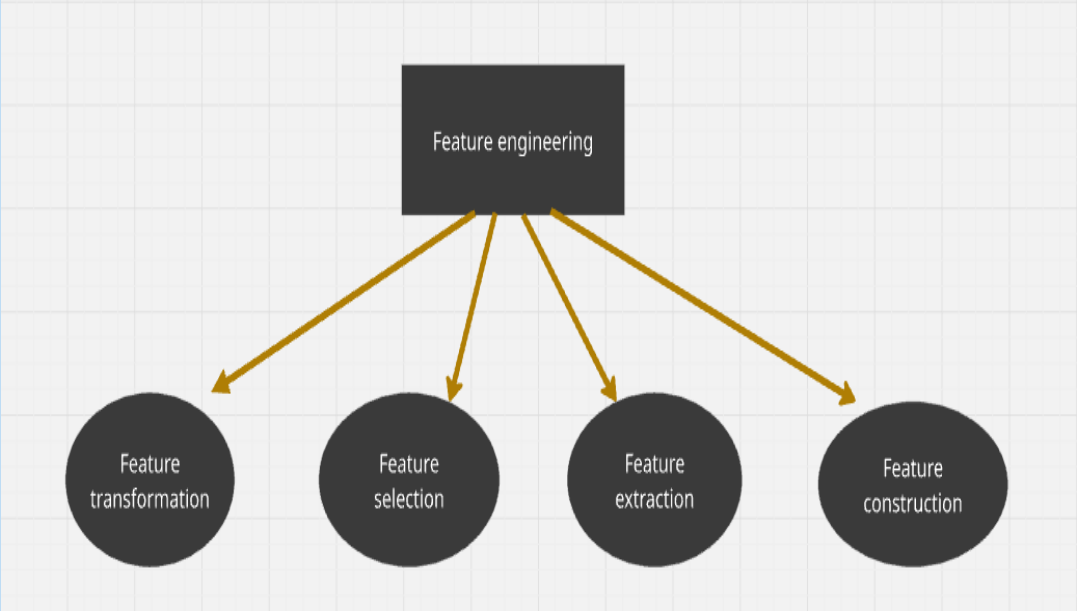
Feature Engineering

Feature engineering is the process of transforming raw data into features that are suitable for machine learning models. It involves selecting, extracting, and transforming the most relevant features from the available data to build more accurate and efficient machine learning models.

Importance of Feature Engineering

1. **Improves Model Accuracy**: By transforming raw data into meaningful features, models can better capture underlying patterns, leading to improved predictive performance.
2. **Reduces Overfitting**: Selecting and creating relevant features helps in minimizing noise and irrelevant information, which reduces the risk of the model overfitting to the training data.
3. **Enhances Interpretability**: Well-engineered features can make models more understandable, allowing stakeholders to gain insights into the factors driving predictions.
4. **Optimizes Computational Efficiency**: By reducing the dimensionality of data and focusing on the most informative features, computational resources are used more effectively, speeding up training and inference.
5. **Facilitates Model Generalization**: Effective feature engineering enables models to generalize better to unseen data, enhancing their applicability in real-world scenarios.



Feature Transformation: Reshaping Your Data

Feature transformation is a key type of feature engineering that involves **modifying existing features to change their scale, distribution, or representation.** The goal is to make the features more suitable for the chosen machine learning model and improve its ability to learn from the data.

Here's a breakdown of the feature transformation techniques you listed:

Encoding Techniques

Encoding is used to convert categorical data into a numerical format that machine learning algorithms can understand.

- **One-Hot Encoding:**
  - **Definition:** Creates new binary columns for each unique category in a categorical feature. If a data point belongs to a category, the corresponding column has a value of 1, and all other category columns are 0.
  - **Steps:** Identify categorical columns. For each unique category in a column, create a new binary column. Iterate through the data, and for each data point, set the value of the corresponding category's binary column to 1 and others to 0.
  - **When to Use:** When the categorical feature has no inherent order (nominal data) and the number of unique categories is not excessively

large. Avoid for high-cardinality categorical features as it can lead to a very large number of new columns.

- **Label Encoding:**
  - **Definition:** Assigns a unique integer to each unique category in a categorical feature.
  - **Steps:** Identify categorical columns. Map each unique category to an integer. Replace the categorical values with their corresponding integers.
  - **When to Use:** When the categorical feature has a clear ordinal relationship (e.g., 'small', 'medium', 'large') and the integer order reflects this relationship. Using it on nominal data can mislead models into assuming an order that doesn't exist.
- **Ordinal Encoding:**
  - **Definition:** Similar to label encoding, but the integer assignment is done in a way that preserves the order of the categories. Requires domain knowledge to define the correct order.
  - **Steps:** Identify ordinal categorical columns. Define the order of the categories. Map each category to an integer based on its position in the defined order. Replace the categorical values with their corresponding integers.
  - **When to Use:** Specifically when dealing with ordinal categorical data where the order of categories is meaningful.

Scaling Techniques

Scaling is used to normalize the range of numerical features, preventing features with larger values from dominating those with smaller values in algorithms sensitive to feature magnitudes.

- **Standardization (Z-score scaling):**
  - **Definition:** Transforms features to have a mean of 0 and a standard deviation of 1. It centers the data around the mean and scales it by the standard deviation. The formula is $z=(x-\mu)/\sigma$, where x is the original value, μ is the mean, and σ is the standard deviation.
  - **Steps:** Calculate the mean and standard deviation of the feature. For each data point, subtract the mean and divide by the standard deviation.
  - **When to Use:** When the data is approximately normally distributed or when using algorithms that assume a Gaussian distribution or are sensitive to the scale of features (e.g., Support Vector Machines,

Logistic Regression, Linear Regression, K-Nearest Neighbors, principal component analysis). Less affected by outliers than min-max scaling.

- **Normalization (Min-Max scaling):**
  - **Definition:** Scales features to a fixed range, typically between 0 and 1. The formula is $x' = (x - \min(x)) / (\max(x) - \min(x))$, where x is the original value, min(x) is the minimum value, and max(x) is the maximum value.
  - **Steps:** Calculate the minimum and maximum values of the feature. For each data point, subtract the minimum and divide by the range (maximum minus minimum).
  - **When to Use:** When the data does not follow a Gaussian distribution or when using algorithms that require features within a specific range (e.g., Neural Networks, algorithms that use distance calculations). Sensitive to outliers, which can compress the range of the majority of the data.

Outlier Handling

Outliers are data points that significantly differ from other observations. They can negatively impact the performance of many machine learning models.

- **Z-score:**
  - **Definition:** Identifies outliers based on how many standard deviations away from the mean a data point is. A common threshold is a Z-score of 3 or -3.
  - **Steps:** Calculate the mean and standard deviation of the feature. For each data point, calculate its Z-score. Flag data points with a Z-score exceeding a predefined threshold as outliers.
  - **When to Use:** When the data is approximately normally distributed. Less effective for skewed data.
- **IQR (Interquartile Range):**
  - **Definition:** Identifies outliers based on the range between the first quartile (Q1) and the third quartile (Q3). Outliers are typically considered to be below $Q1 - 1.5 \times IQR$ or above $Q3 + 1.5 \times IQR$.
  - **Steps:** Calculate the first quartile (25th percentile) and the third quartile (75th percentile). Calculate the IQR (Q3−Q1). Define the lower and upper bounds for outlier detection. Identify data points falling outside these bounds as outliers.
  - **When to Use:** When the data is not normally distributed or when the presence of outliers makes the mean and standard deviation less representative. More robust to extreme values than the Z-score method.

- **Percentile:**
  - **Definition:** Identifies outliers by setting thresholds based on specific percentiles of the data distribution (e.g., considering data points below the 1st percentile or above the 99th percentile as outliers).
  - **Steps:** Determine the desired lower and upper percentile thresholds. Calculate the values corresponding to these percentiles. Flag data points below the lower percentile threshold or above the upper percentile threshold as outliers.
  - **When to Use:** Provides a flexible way to define outlier boundaries based on the desired proportion of data to consider as outliers. Useful when the data distribution is unknown or complex.
- **Winsorization:**
  - **Definition:** Instead of removing outliers, Winsorization caps extreme values at a specified percentile. Extreme low values are replaced with the value at a lower percentile, and extreme high values are replaced with the value at an upper percentile.
  - **Steps:** Determine the lower and upper percentile thresholds for capping. Calculate the values at these percentiles. For data points below the lower threshold, replace them with the lower percentile value. For data points above the upper threshold, replace them with the upper percentile value.
  - **When to Use:** When you want to reduce the impact of outliers without removing them entirely, preserving the number of data points.

Mathematical Transformation

Mathematical transformations apply a mathematical function to a feature to change its distribution or the relationship between features.

- **Log Transformation:**
  - **Definition:** Applies the logarithm function (e.g., natural log or log base 10) to the feature values. Commonly used to reduce right skewness in data.
  - **Steps:** Check if the feature contains zero or negative values (logarithm is undefined for non-positive numbers). If necessary, add a constant to all values to make them positive (e.g., log(x+1)). Apply the logarithm function to each value.
  - **When to Use:** When the data is heavily right-skewed and contains positive values. Can help linearize relationships between variables.
- **Square Root Transformation:**

- **Definition:** Applies the square root function to the feature values. Also used to reduce right skewness, but often less strongly than log transformation.
- **Steps:** Check if the feature contains negative values (square root of negative numbers is not a real number). Apply the square root function to each value.
- **When to Use:** When the data is moderately right-skewed and contains non-negative values.

- **Reciprocal Transformation:**
  - **Definition:** Applies the reciprocal function ($1/x$) to the feature values. Can be used to handle left-skewed data, but it reverses the order of the data.
  - **Steps:** Check if the feature contains zero values (reciprocal of zero is undefined). Apply the reciprocal function to each value.
  - **When to Use:** Less common than log or square root transformations. Can be useful for features where the inverse relationship is more meaningful.
- **Box-Cox Transformation:**
  - **Definition:** A power transformation that transforms the data to be more normally distributed. It is defined as $(x\lambda-1)/\lambda$ for $\lambda=0$ and $\log(x)$ for $\lambda=0$. It requires the data to be positive.
  - **Steps:** Check if the feature contains positive values. Estimate the optimal λ parameter that maximizes the normality of the transformed data (often done using maximum likelihood). Apply the Box-Cox transformation with the estimated λ to each value.
  - **When to Use:** When the data is skewed and strictly positive, and you want to achieve a more normal distribution.
- **Yeo-Johnson Transformation:**
  - **Definition:** Another power transformation technique that is an extension of the Box-Cox transformation. It can be applied to data that includes zero and negative values.
  - **Steps:** Estimate the optimal λ parameter (similar to Box-Cox). Apply the Yeo-Johnson transformation with the estimated λ to each value. The transformation formula is more complex and depends on whether the value is positive, negative, or zero.
  - **When to Use:** When the data is skewed and contains positive, negative, or zero values, and you want to achieve a more normal distribution.

Encoding (Binning/Discretization)

Binning transforms continuous numerical features into categorical or ordinal features by grouping values into bins.

- **Uniform Binning (Equal Width Binning):**
  - **Definition:** Divides the range of the feature into a fixed number of bins of equal width.
  - **Steps:** Determine the number of bins. Calculate the width of each bin (range / number of bins). Assign each data point to a bin based on its value.
  - **When to Use:** Simple to implement and interpret. Can be sensitive to outliers, which can result in most data falling into a few bins.
- **Quantile Binning (Equal Frequency Binning):**
  - **Definition:** Divides the data into bins such that each bin contains approximately the same number of data points. The bin boundaries are determined by quantiles.
  - **Steps:** Determine the number of bins (or quantiles). Calculate the quantile values that define the bin boundaries. Assign each data point to a bin based on which quantile range it falls into.
  - **When to Use:** When the data is not uniformly distributed and you want to ensure each bin has a similar number of observations. Less sensitive to outliers than uniform binning.
- **Custom Binning:**
  - **Definition:** Defines bin boundaries based on domain knowledge or specific requirements.
  - **Steps:** Determine the custom bin boundaries based on expertise or analysis. Assign each data point to a bin based on these custom boundaries.
  - **When to Use:** When there are specific thresholds or ranges that are meaningful in the context of the problem.

Handling Missing Values

Missing values are common in real-world datasets and need to be addressed before training a machine learning model.

- **Dropping:**
  - **Definition:** Removing rows or columns that contain missing values.
  - **Steps:** Identify rows or columns with missing values. Remove the identified rows or columns from the dataset.

    - **When to Use:** When the number of missing values is small relative to the dataset size and dropping them does not lead to significant loss of information. Also, consider dropping columns with a very high percentage of missing values.
- **Imputation:**
    - **Definition:** Replacing missing values with estimated values. Common imputation strategies include using the mean, median, mode, or more advanced techniques.
    - **Steps:** Identify missing values. Choose an imputation strategy (e.g., mean, median, mode). Calculate the chosen statistic from the available data in the feature. Replace the missing values with the calculated statistic.
    - **When to Use:** When dropping missing values would result in a significant loss of data. The choice of imputation strategy depends on the data distribution and the nature of the missingness. Mean imputation is sensitive to outliers, while median imputation is more robust. Mode imputation is suitable for categorical or discrete numerical data. More advanced techniques (e.g., K-Nearest Neighbors imputation, regression imputation) can capture more complex relationships.

Feature Selection: Identifying the Most Relevant Variables

Feature selection is the process of **choosing a subset of the most relevant features (variables, predictors) from the original dataset to be used in building a machine learning model.** Instead of using all available features, feature selection aims to identify and keep only those that are most informative and contribute most to the model's performance.

The Importance of Feature Selection

Including irrelevant or redundant features in your machine learning model can have several negative consequences. Feature selection is important because it helps to:

- **Reduce Overfitting:** By removing irrelevant or noisy features, feature selection helps the model to focus on the true patterns in the data and avoid learning from random fluctuations, thus improving its ability to generalize to unseen data.

- **Improve Model Accuracy:** Selecting the most relevant features can lead to simpler and more accurate models by removing noise and focusing on the most predictive signals.
- **Decrease Training Time:** Training a model on a smaller subset of features is computationally less expensive and faster.
- **Enhance Model Interpretability:** Models with fewer features are often easier to understand and interpret, making it easier to explain the model's predictions.
- **Mitigate the Curse of Dimensionality:** In high-dimensional datasets, the volume of the data space increases rapidly with the number of features, making it sparse and difficult for algorithms to find meaningful patterns. Feature selection helps to reduce dimensionality.

Feature selection techniques can be broadly categorized into three main types: Filter Methods, Wrapper Methods, and Embedded Methods. Hybrid methods combine aspects of these approaches.

Types of Feature Selection

1. Filter Methods

Filter methods select features based on their statistical properties or their relationship with the target variable, independently of any specific machine learning algorithm. They "filter" out features based on a scoring metric.

- **Correlation Coefficient Techniques:**
  - **Definition:** Measure the linear relationship between two continuous variables. In feature selection, it's commonly used to assess the correlation between each independent numerical feature and the continuous target variable (Pearson correlation) or the correlation between independent numerical features to identify multicollinearity.
  - **Steps to Calculate:**
    - For Pearson correlation between a feature (X) and the target (Y): Calculate the covariance of X and Y, and divide it by the product of their standard deviations. The formula is: $r=\sum i=1n(xi-x^-)2\sum i=1n(yi-y^-)2\sum i=1n(xi-x^-)(yi-y^-)$ where n is the number of data points, xi and yi are the individual data points, and x^- and y^- are the means.
    - For correlation between independent features: Calculate the correlation coefficient for each pair of independent numerical features.
  - **Uses:**

        - To identify features highly correlated with the target variable (potentially good predictors).
        - To identify highly correlated independent features (multicollinearity). You might remove one of a pair of highly correlated features to avoid redundancy.
        - Suitable for numerical features and continuous target variables (Pearson). Other correlation measures like Spearman or Kendall can be used for non-linear relationships or ordinal data.
- **Chi-Square Test:**
    - **Definition:** A statistical test used to determine if there is a significant association between two categorical variables. In feature selection, it's used to assess the independence between a categorical independent feature and a categorical target variable.
    - **Steps to Calculate:**
        - Create a contingency table showing the frequencies of each category combination for the two variables.
        - Calculate the expected frequencies for each cell in the table, assuming independence between the variables.
        - Calculate the Chi-Square statistic using the formula: $\chi 2 = \sum i \sum j Eij (Oij - Eij)2$ where Oij is the observed frequency and Eij is the expected frequency for each cell (i,j).
        - Compare the calculated $\chi 2$ statistic to a critical value from the Chi-Square distribution or use the p-value to determine statistical significance.
    - **Uses:** To select categorical features that are statistically dependent on a categorical target variable. Higher Chi-Square values indicate a stronger association.
- **ANOVA (Analysis of Variance):**
    - **Definition:** A statistical test used to compare the means of three or more groups to see if there is a statistically significant difference between them. In feature selection, it's used to assess if there is a significant difference in the means of a continuous feature across different categories of a categorical target variable.
    - **Steps to Calculate:**
        - Calculate the variance within each group (categories of the target variable) and the variance between the groups.
        - Calculate the F-statistic, which is the ratio of the between-group variance to the within-group variance.
        - Compare the F-statistic to a critical value from the F-distribution or use the p-value to determine statistical significance.

    - **Uses:** To select continuous features whose means differ significantly across the different classes of a categorical target variable. Higher F-statistics indicate a stronger relationship.
- **Mutual Information:**
    - **Definition:** Measures the dependency between two variables. It quantifies the amount of information obtained about one variable through the other variable. Unlike correlation, it can capture non-linear relationships.
    - **Steps to Calculate:** Based on the concept of entropy. For discrete variables X and Y, it's calculated as: $I(X;Y)=\sum_{y\in Y}\sum_{x\in X} p(x,y)\log(p(x)p(y)p(x,y))$ where $p(x,y)$ is the joint probability distribution and $p(x)$ and $p(y)$ are the marginal probability distributions. For continuous variables, it involves probability density functions and integration.
    - **Uses:** To select features that share a high degree of mutual information with the target variable. Can be used for both discrete and continuous features and target variables. Higher mutual information values indicate stronger dependency.
- **Variance Threshold:**
    - **Definition:** Removes features whose variance does not meet a certain threshold. Features with low variance have values that are very similar across all data points, meaning they carry little information.
    - **Steps to Calculate:**
        - Calculate the variance of each feature.
        - Set a variance threshold.
        - Remove features with a variance less than the threshold.
    - **Uses:** A simple baseline method to remove constant or near-constant features that are unlikely to be informative for a model. Does not consider the relationship with the target variable.

2. Wrapper Methods

Wrapper methods evaluate subsets of features by training and evaluating a machine learning model on each subset. The feature selection process is "wrapped" around the model training.

- **Forward Selection:**
    - **Definition:** An iterative method that starts with an empty set of features and adds one feature at a time that best improves the model's performance until a stopping criterion is met.
    - **Steps to Calculate:**

        - Start with an empty set of selected features.
        - Train the chosen machine learning model on each individual feature and evaluate its performance (e.g., using cross-validation). Select the single best feature.
        - In subsequent steps, consider adding each of the remaining features to the current set of selected features.
        - Train the model on each of these new subsets (current features + one additional feature) and evaluate performance.
        - Add the feature that results in the greatest improvement in model performance to the set of selected features.
        - Repeat until adding no more features significantly improves performance or a predefined number of features is reached.
    - **Uses:** When the number of features is not too large. It's computationally less expensive than Exhaustive Feature Selection (which checks all possible subsets) but can be computationally intensive for a large number of features. It considers feature interactions to some extent.
- **RFE (Recursive Feature Elimination):**
    - **Definition:** A backward elimination method that starts with all features and recursively removes the least important features based on the model's coefficients or feature importance until the desired number of features is reached.
    - **Steps to Calculate:**
        - Train the chosen machine learning model on the full set of features.
        - Obtain the importance of each feature (e.g., from coefficients in linear models, or feature importance attributes in tree-based models).
        - Remove the least important feature(s).
        - Retrain the model on the remaining features.
        - Repeat the process of ranking and removing features until the desired number of features is left.
    - **Uses:** Effective for models that provide a measure of feature importance. It's a widely used technique and can be combined with various models. Can be computationally expensive for a large number of features and require careful tuning of the number of features to select.

3. Hybrid Methods

Hybrid methods combine the strengths of both filter and wrapper methods. They often use a filter method to quickly reduce the initial set of features and then employ a wrapper method on the reduced set.

- **Boruta:**
  - **Definition:** An all-relevant feature selection method built around the Random Forest algorithm. It works by creating shuffled copies of the features (shadow features) and iteratively removing features that are statistically less important than these random probes.
  - **Steps to Calculate:**
    - Create shuffled copies of all original features (shadow features).
    - Combine the original features with the shadow features.
    - Train a Random Forest model on this extended dataset.
    - Calculate the Z-scores of the feature importance for both original and shadow features.
    - Find the maximum Z-score among the shadow features (MZSA).
    - For each original feature, check if its Z-score is significantly higher than the MZSA using a statistical test.
    - Features with significantly higher importance are tentatively marked as important. Features with significantly lower importance than MZSA are deemed unimportant and removed.
    - Repeat the process until all features are confirmed as important or deemed unimportant.
  - **Uses:** To find all features that are statistically relevant to the target variable, not just a minimal optimal set. Robust to multicollinearity and can capture complex interactions. Can be computationally more expensive than simple filter methods.

4. Embedded Methods

Embedded methods perform feature selection as part of the model training process. The feature selection logic is "embedded" within the learning algorithm itself.

- **Definition:** These algorithms have built-in mechanisms for feature selection during the model construction. They leverage the model's internal properties (like coefficients or feature importance) to identify and select the most relevant features.

- **How it Works:** The learning algorithm itself penalizes or assigns importance scores to features during training. Features that are less important for the model's performance are either assigned zero or very small coefficients (as in regularization) or are given lower importance scores (as in tree-based models).
- **Uses:**
    - Often computationally more efficient than wrapper methods because feature selection is integrated into the training process.
    - Considers feature interactions.
    - Examples of algorithms with embedded feature selection:
        - **Lasso Regression (L1 Regularization):** Adds a penalty to the absolute values of the coefficients, which can shrink some coefficients to exactly zero, effectively performing feature selection.
        - **Ridge Regression (L2 Regularization):** Adds a penalty to the squared values of the coefficients. While it shrinks coefficients, it typically doesn't set them exactly to zero, so it's less aggressive for feature selection than Lasso.
        - **Elastic Net:** Combines L1 and L2 regularization.
        - **Tree-based Models (Decision Trees, Random Forests, Gradient Boosting):** These models inherently rank features based on how much they contribute to reducing impurity (or variance) during the tree building process. Feature importance scores can be extracted from these models to select features.

Feature Construction:

Feature construction, also known as feature creation or feature synthesis, is a vital part of feature engineering. It involves **creating new features from the existing raw data or from features that have already been transformed or selected.** Unlike feature transformation which modifies existing features, feature construction actively builds entirely new ones, often by combining or manipulating the original attributes based on domain knowledge or data exploration. The goal is to generate features that are more informative and have a stronger relationship with the target variable.

Importance of Feature Construction:

Feature construction is important because raw data often doesn't directly provide the most useful information for a machine learning model. Important patterns and

relationships might be hidden and can only be revealed by combining or manipulating existing features. Its importance stems from its ability to:

- **Uncover Hidden Information:** It allows you to extract valuable insights that are not immediately apparent in the raw features.
- **Improve Model Performance:** Well-constructed features can significantly boost the accuracy and predictive power of your models by providing them with more relevant information.
- **Capture Complex Relationships:** By creating interaction terms or combining features, you can represent non-linear and complex relationships that a model might struggle to learn from individual raw features.
- **Incorporate Domain Expertise:** Domain knowledge is crucial in identifying how existing features can be combined or transformed to create meaningful new features.
- **Simplify the Problem:** Sometimes, a well-constructed feature can simplify the learning task for the model.

Feature extraction

Feature extraction is another important type of feature engineering. Unlike feature selection, which chooses a subset of original features, and feature construction, which creates new features by combining or manipulating existing ones, **feature extraction transforms the data from a high-dimensional space to a lower-dimensional space.** The goal is to create a new set of features (the extracted features) that capture the most important information or variance from the original features, while significantly reducing the number of features.

The Importance of Feature Extraction

Feature extraction is particularly valuable when dealing with high-dimensional data, such as images, text, or time series. Its importance lies in its ability to:

- **Reduce Dimensionality:** It effectively reduces the number of features, which helps to combat the curse of dimensionality, making models less prone to overfitting and improving their generalization ability.
- **Reduce Computational Cost:** Training models on a lower-dimensional dataset requires less memory and computational power, leading to faster training and inference times.

- **Remove Redundancy and Noise:** Feature extraction techniques often identify and combine highly correlated features, effectively removing redundancy and potentially filtering out noise in the data.
- **Improve Model Performance:** By representing the data in a more compact and informative way, feature extraction can sometimes improve the performance of machine learning algorithms.
- **Aid Visualization:** Reducing data to two or three dimensions allows for easier visualization and exploration of the data's structure and patterns.

Techniques for Feature Extraction

Here are two common techniques used for feature extraction:

Principal Component Analysis (PCA)

- **Definition:** PCA is a linear dimensionality reduction technique that transforms the data to a new coordinate system such that the greatest variance in the data is captured along the first axis (the first principal component), the second greatest variance along the second axis, and so on. The principal components are orthogonal to each other.
- **Steps to Perform:**
    1. **Standardize the Data:** Scale the features so that they have zero mean and unit variance. This is crucial because PCA is sensitive to the scale of the features.
    2. **Compute the Covariance Matrix:** Calculate the covariance matrix of the standardized data to understand the relationships and variance between different features.
    3. **Compute the Eigenvectors and Eigenvalues:** Calculate the eigenvectors and eigenvalues of the covariance matrix. Eigenvectors represent the directions of maximum variance (the principal components), and eigenvalues represent the magnitude of the variance along those directions.
    4. **Sort Eigenvalues and Corresponding Eigenvectors:** Sort the eigenvalues in descending order and arrange the corresponding eigenvectors accordingly. The eigenvector with the highest eigenvalue is the first principal component, and so on.
    5. **Select Principal Components:** Choose the top k eigenvectors (principal components) that capture a desired amount of variance (e.g., 95%). The number of components k will be the new dimensionality of the data, where k≤original number of features.
    6. **Project Data onto the New Subspace:** Create a projection matrix from the selected k eigenvectors. Multiply the original standardized

data by this projection matrix to transform the data into the new k-dimensional space.

- **Uses:**
  - Dimensionality reduction while retaining most of the variance.
  - Noise reduction by discarding components with low variance.
  - Data visualization by reducing data to 2 or 3 dimensions.
  - As a preprocessing step for other machine learning algorithms.
  - Identifying the most important directions of variation in the data.

UMAP (Uniform Manifold Approximation and Projection)

- **Definition:** UMAP is a non-linear dimensionality reduction technique that aims to preserve both the local and global structure of the data in a lower-dimensional embedding. It is based on the theory of Riemannian geometry and topological data analysis.
- **Steps to Perform:**
  1. **Construct a High-Dimensional Graph:** Build a weighted graph in the high-dimensional space where edge weights represent the likelihood that two points are connected (based on distance and local neighborhood density). This process involves building a fuzzy simplicial set representation of the data's topology.
  2. **Construct a Low-Dimensional Graph:** Create a graph in the target lower-dimensional space (e.g., 2D or 3D). Initially, points are placed randomly.
  3. **Optimize the Low-Dimensional Embedding:** Use an optimization process (typically stochastic gradient descent) to adjust the positions of the points in the low-dimensional space such that the structure of the low-dimensional graph is as similar as possible to the structure of the high-dimensional graph. This involves minimizing the difference between the high and low-dimensional graph representations.
- **Uses:**
  - Dimensionality reduction for visualization and exploratory data analysis, particularly for complex, non-linear data structures.
  - Revealing clusters and the overall shape (manifold) of the data in a lower-dimensional space.
  - As a preprocessing step for clustering or classification tasks, providing a lower-dimensional representation that captures important data structure.
  - Analyzing and visualizing high-dimensional data from various domains, including single-cell RNA sequencing, image data, and text data embeddings.