Efficient Data Cleaning and Preprocessing Techniques for Robust Machine Learning Models

Dr Charalambos Chrysostomou
Associate Research Scientist
The Cyprus Institute

Efficient Data Cleaning and Preprocessing Techniques for Robust Machine Learning Models

Dr. Charalambos Chrysostomou

Dr. Chrysostomou is an Associate Research Scientist with background in machine learning and data-driven applications. With expertise in big data methodologies, he has contributed to diverse applications like data classification, genomic analysis, and medical imaging.

Email: c.chrysostomou@cyi.ac.cy

Contact Number: +357 22 208 792.

Brief overview of the importance of data quality in machine learning

1. **Garbage In, Garbage Out:** The quality of input data directly influences the reliability of the model's predictions. Clean, high-quality data is crucial.
2. **Bias and Fairness:** Low-quality or unrepresentative data can introduce bias, leading to unfair model predictions for underrepresented groups.
3. **Performance and Accuracy:** Clean data enhances model performance and accuracy by facilitating the recognition of underlying patterns.
4. **Trust and Usability:** Stakeholders' trust in a model is tied to their confidence in the quality of the underlying data.
5. **Efficiency:** Preprocessed data improves model efficiency by reducing noise, managing missing values, and removing irrelevant features.

Introduction to data cleaning and preprocessing techniques

Data Cleaning

This process involves identifying and correcting errors in the dataset, such as dealing with missing or incorrect data. Data cleaning ensures that your dataset is accurate, consistent, and usable.

Data Preprocessing

Data preprocessing involves transforming raw data into an understandable format for algorithms. It prepares inputs for machine learning models.

Introduction to data cleaning and preprocessing techniques

Data Cleaning

1. **Missing Values:** Addressed by deletion, filling with statistical values, or predictive methods.
2. **Outliers:** Identified using statistical techniques like Z-score or IQR, and handled by truncation or imputation.
3. **Duplicates:** Detected and removed to prevent skewed analysis results.
4. **Inconsistencies:** Corrected for uniformity in recording, such as consistent units or date formats.

Handling Missing Values - Overview

Handling missing values is a critical step in data cleaning. Missing data can lead to biased or incorrect results, so it's important to handle them appropriately. The three primary strategies we'll discuss are:

- Dropping missing values
- Imputation techniques
- Interpolation

Handling Missing Values - Dropping Missing Values

Dropping missing values is the simplest method, where we remove the rows or columns containing missing data.

Pros:

- Easy to implement
- Doesn't introduce additional bias

Cons:

- Can result in loss of valuable data
- Not ideal when missing data is not random

Use when:

- The dataset is large and a small proportion of data is missing
- Missing data is likely to be random

Handling Missing Values - Imputation Techniques

Imputation involves filling missing values with estimated ones. The method can vary depending on the nature of the data and the type of variable.

- **Mean/Median/Mode Imputation:** Replace missing values with the mean (for continuous variables) or median/mode (for categorical variables).
- **Predictive Imputation:** Use statistical or machine learning algorithms to predict missing values based on other data.

Handling Missing Values - Imputation Techniques

Pros:

- Prevents data loss
- Can handle non-random missingness if done correctly

Cons:

- Can introduce bias if assumptions are incorrect
- More complex to implement

Use when:

- Missingness is not random
- Enough non-missing data is available to make reasonable estimates

Handling Missing Values - Interpolation

Interpolation involves estimating missing values using other adjacent observed values. This method assumes a specific relationship between samples.

- **Linear Interpolation:** Assumes a straight-line relationship between points.
- **Polynomial/ Spline Interpolation:** Assumes a polynomial or flexible curve relationship between points.

Handling Missing Values - Interpolation

Pros:

- Can provide a good estimate for time-series data
- Can handle non-linear relationships

Cons:

- Doesn't work well if assumption about relationship between samples is incorrect
- Not suitable for categorical data

Use when:

- Data is sequential (e.g., time-series)
- Missing values are likely to be a function of their nearest values

Handling Outliers - Overview

Outliers are data points that significantly differ from the rest of the dataset. They can affect the performance of machine learning models and lead to incorrect conclusions. In this section, we will discuss three methods for detecting and handling outliers:

- Z-score method
- IQR method
- Winsorization

Handling Outliers - Z-score Method

The Z-score method calculates the number of standard deviations a data point is from the mean. Data points with a Z-score above a specific threshold (e.g., 2 or 3) are considered outliers.

$$Z = (x - \mu) / \sigma$$

Where:

- **x** is the value you are interested in.
- **μ** (mu) is the mean of the dataset.
- **σ** (sigma) is the standard deviation of the dataset.

Handling Outliers - Z-score Method

Pros:

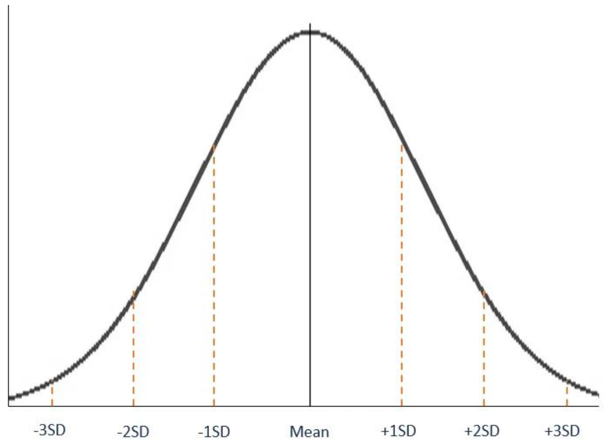
- Works well for normally distributed data
- Simple to calculate and implement

Cons:

- Sensitive to extreme values
- Assumes data follows a normal distribution

Use when:

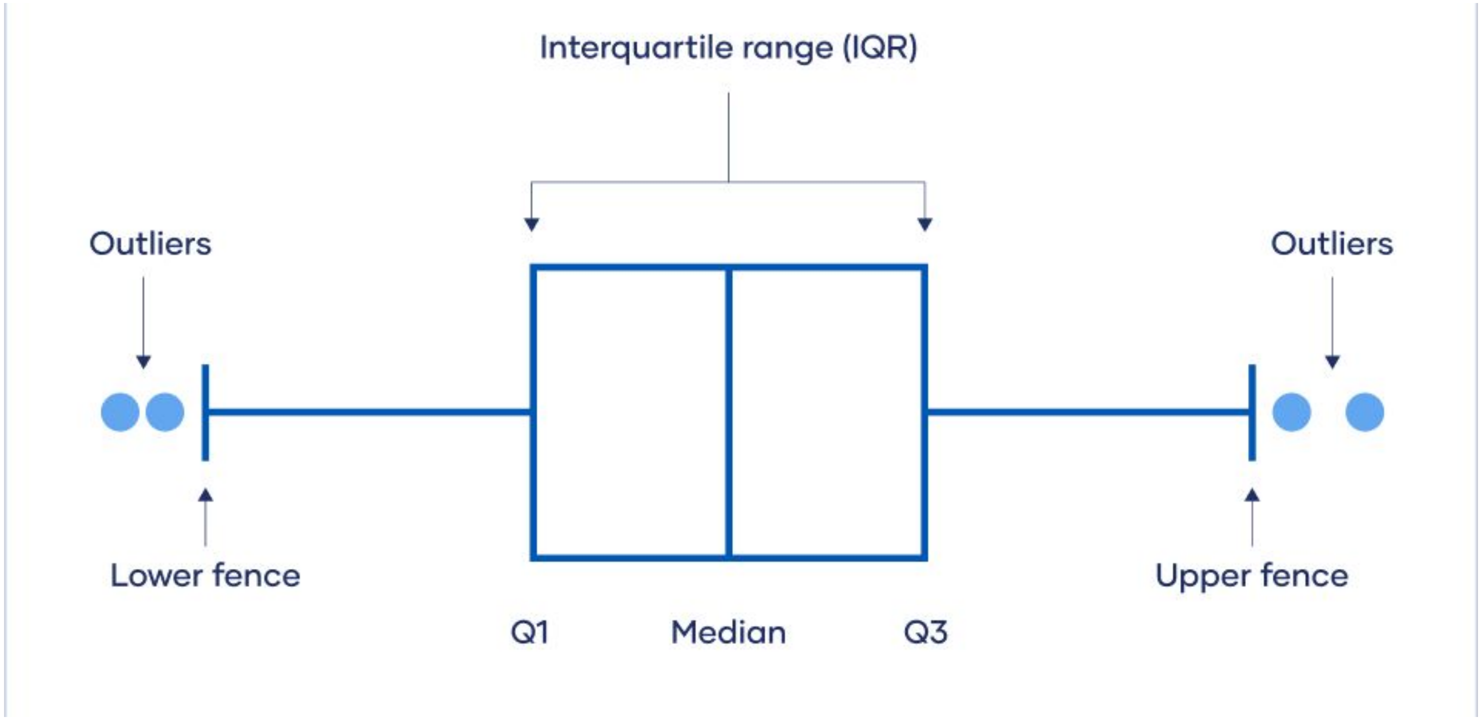
- Data is approximately normally distributed
- The underlying distribution of data is symmetric



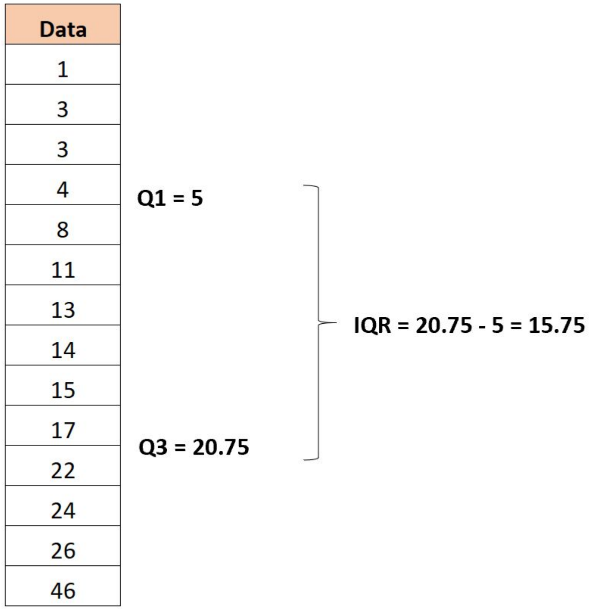
Handling Outliers - IQR Method

1. **Identify Quartiles:** The first quartile (Q1) is the middle number between the smallest number and the median of the dataset, representing the 25th percentile. The third quartile (Q3) is the middle value between the median and the highest value, representing the 75th percentile.
2. **Calculate IQR:** The Interquartile Range (IQR) is the difference between Q3 and Q1. It represents the middle 50% of the data.
   IQR = Q3 - Q1
3. **Define Outlier Bounds**: Outliers are values that fall outside the range defined by Q1 - 1.5 * IQR and Q3 + 1.5 * IQR.
   Lower bound = Q1 - 1.5 * IQR
   Upper bound = Q3 + 1.5 * IQR
4. **Identify Outliers**: Any data points falling below the lower bound or above the upper bound are considered outliers.

Handling Outliers - IQR Method



Handling Outliers - IQR Method



Data
1
3
3
4
8
11
13
14
15
17
22
24
26
46

Q1 = 5

Q3 = 20.75

IQR = 20.75 - 5 = 15.75

Lower Limit = 5 - 1.5*15.75 = -18.625

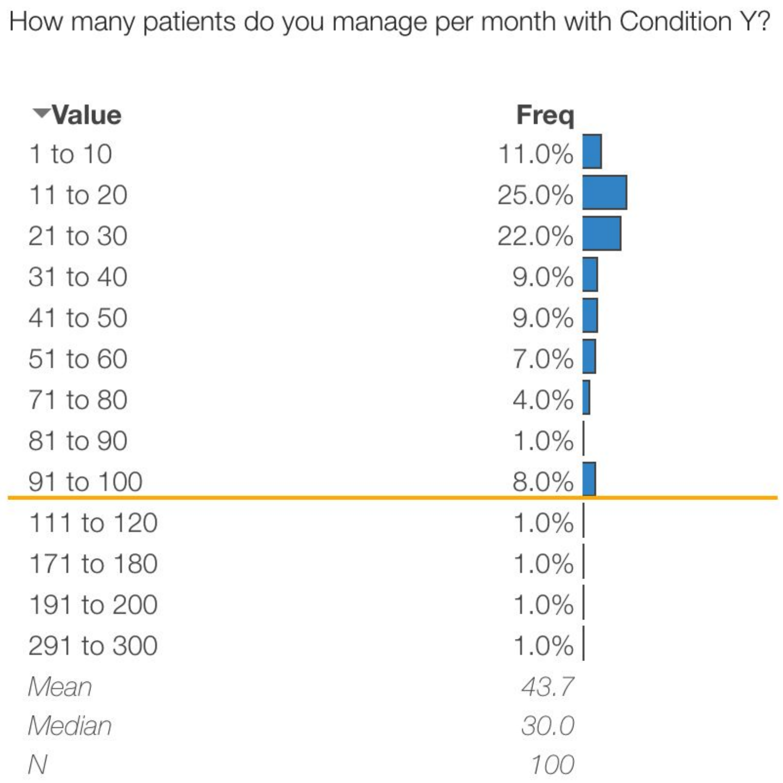
Upper Limit = 20.75 + 1.5*15.75 = 44.375

Pros:
- Robust to extreme values
- Doesn't assume normal distribution

Cons:
- Can be less effective for identifying outliers in small datasets
- Assumes data is unimodal

Use when:
- Data is not normally distributed
- Robustness to extreme values is important

Handling Outliers - Winsorization



How many patients do you manage per month with Condition Y?

Value	Freq
1 to 10	11.0%
11 to 20	25.0%
21 to 30	22.0%
31 to 40	9.0%
41 to 50	9.0%
51 to 60	7.0%
71 to 80	4.0%
81 to 90	1.0%
91 to 100	8.0%
111 to 120	1.0%
171 to 180	1.0%
191 to 200	1.0%
291 to 300	1.0%
Mean	*43.7*
Median	*30.0*
N	*100*

Winsorization involves replacing outliers with the nearest non-outlier value, typically using percentiles as a threshold (e.g., 1% and 99%).

Handling Outliers - Winsorization

Pros:

- Reduces the impact of extreme values without completely removing them
- Doesn't require assumptions about the data distribution

Cons:

- Can introduce bias if the threshold is set incorrectly
- Alters the original data, which might not be desirable

Use when:

- Retaining the overall structure of the data is important
- Reducing the impact of outliers without removing them is desired

Duplicate Data Detection and Removal

- **Duplicated data are identical entries in your dataset.** They can mislead the analysis by over-representing certain information, biasing the results. Let's explore how to detect and remove duplicate data.
- Duplicate data can be detected using programming languages like Python or software tools for data handling. In Python, the pandas library provides the duplicated() function, which can help identify duplicate rows in your DataFrame.

Duplicate Data Removal

Pros:

- Easy to implement
- Improves data accuracy

Cons:

- Requires caution; duplicates could be valid data repetitions

Use when:

- There are certain data points repeated without any variations
- Duplication is not a natural aspect of your data collection

Inconsistent Data Entry Correction

Inconsistencies in data entries can occur due to various reasons like

- human error,
- different data entry conventions, or
- system glitches.

These inconsistencies can lead to errors in data analysis

Techniques for Correcting Inconsistent Data Entries

1. **Case Standardization**: Convert all characters to lower or upper case to ensure uniformity.
2. **Formatting Dates**: Ensure dates are in a consistent format across the dataset.
3. **Merging Categories**: If the same category is represented in multiple ways, consolidate them into a single representation.
4. **Strip White Spaces**: Extra spaces at the beginning or end of entries can cause unique values to be seen as different.
5. **Use of a Master Dictionary**: For known inconsistencies, a master dictionary can be used for mapping and replacing values.

Techniques for Correcting Inconsistent Data Entries

1. **Case Standardization**: Transform "apple", "BANANA", "Cherry" to "apple", "banana", "cherry".
2. **Formatting Dates**: Convert "12/31/2020", "2020-12-31", "31 Dec 2020" to "2020-12-31".
3. **Merging Categories:** Change all "NYC" entries to "New York City".
4. **Strip White Spaces:** Clean "John ", " Jane", " Mary " to "John", "Jane", "Mary".
5. **Master Dictionary:** Replace "grey" and "colour" with "gray" and "color" respectively.

Techniques for Correcting Inconsistent Data Entries

Pros:

- Enhances data uniformity
- Improves accuracy of the analysis

Cons:

- Can be time-consuming
- Requires thorough understanding of the data and domain

Use when:

- There are irregularities in data representation
- The dataset comes from different sources or conventions

Introduction to data cleaning and preprocessing techniques

Data Preprocessing

1. **Data Transformation:** Applying techniques to make data meet machine learning assumptions.
2. **Feature Scaling and Normalization:** Implementing methods to balance numerical input variables.
3. **Encoding Categorical Variables:** Using techniques to convert categorical variables into numerical.
4. **Feature Selection:** Selecting the most pertinent features using methods such as filter, wrapper, or embedded methods.
5. **Feature Engineering:** Creating new features from existing ones to enhance model performance, based on domain knowledge or automation.

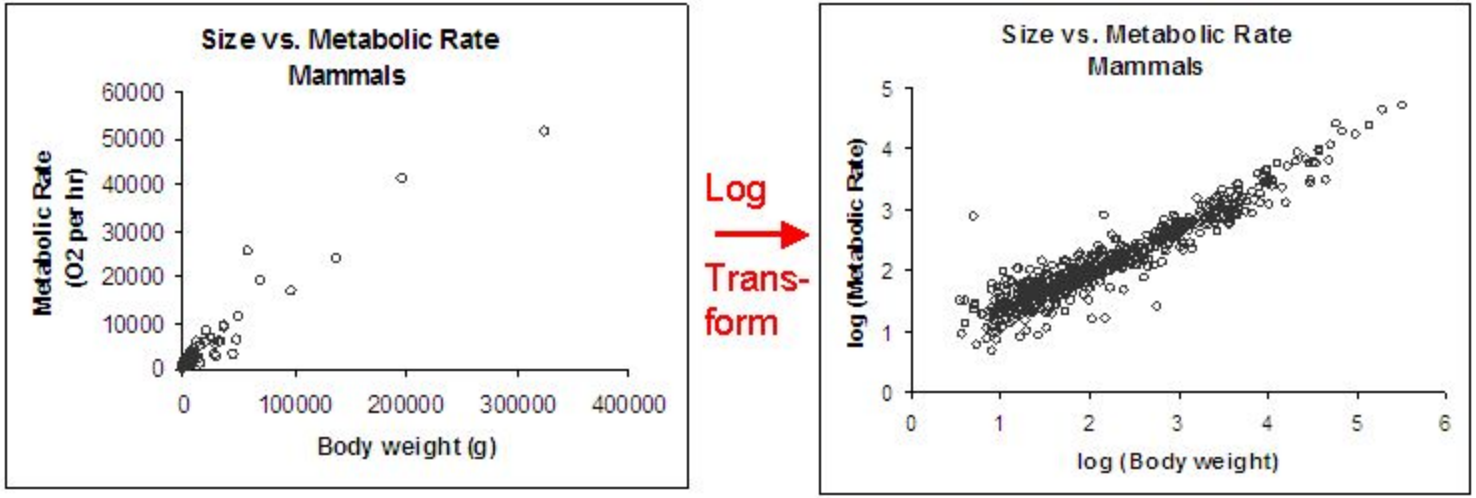
Data Preprocessing - Data Transformation

Data transformation involves changing the scale or distribution of variables to better suit the assumptions of certain algorithms or improve model performance. We'll be discussing three popular methods:

- Log Transformation,
- Box-Cox Transformation, and
- Yeo-Johnson Transformation.

Log Transformation

Log transformation is a commonly used technique to reduce skewness in a dataset, and it works by applying the logarithmic function to each data point.



Log Transformation

Pros:

- Helps manage skewed data
- Compresses the scale of the data, making it easier to handle

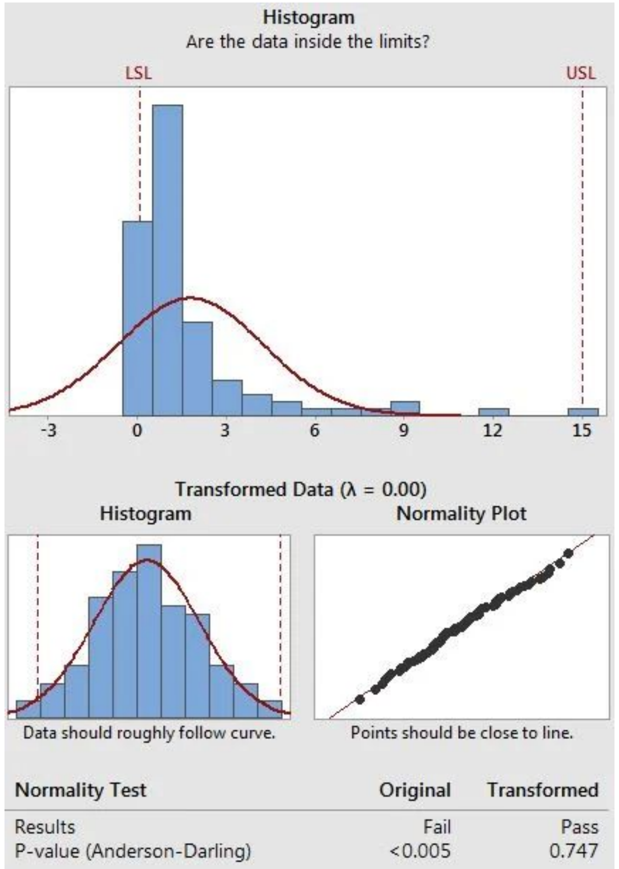
Cons:

- Cannot be applied to zero or negative values
- May not normalize the distribution if data is heavily skewed

Use when:

- Data is right-skewed (tail to the right)
- All data points are positive

Box-Cox Transformation



Normality Test	Original	Transformed
Results	Fail	Pass
P-value (Anderson-Darling)	<0.005	0.747

Box-Cox transformation is a flexible method for transforming data to approximate normality. It involves finding an optimal lambda parameter that best normalizes your data.

Box-Cox Transformation

Pros:

- Dynamically determines the power transformation
- Works well for positive non-zero data

Cons:

- Cannot handle zero or negative values
- Complexity of determining the right parameter

Use when:

- Data is skewed and all data points are positive
- You want a dynamic method to normalize data

Yeo-Johnson Transformation

Pros:

- Can handle negative, zero, and positive values
- Works well for a variety of distributions

Cons:

- Complexity of determining the right parameters
- May not be needed if simpler transformations are sufficient

Use when:

- Data contains negative values
- You need a more flexible method to normalize data

Data Preprocessing - Feature Scaling and Normalization

Feature scaling and normalization are techniques used to standardize the range of features in a dataset. This is crucial in machine learning, as models often perform better when numerical input variables are on a similar scale. We will discuss

- Min-Max Scaling,
- Standard Scaling (Z-score normalization), and
- Robust Scaling.

Feature Scaling and Normalization - Min-Max Scaling

Min-Max scaling, also known as normalization, rescales the features to a fixed range, usually 0 to 1.

X' = (X - X_min) / (X_max - X_min)

Where:

- **X** is the original value
- **X'** is the scaled value
- **X_min** is the smallest value in the feature column
- **X_max** is the largest value in the feature column

Pros:

- Simple and intuitive
- Preserves the original distribution

Cons:

- Sensitive to outliers
- Not suitable for data with large standard deviations

Use when:

- You want to preserve zero values in the dataset
- Your data doesn't contain extreme outliers

Feature Scaling and Normalization - Standard Scaling (Z-score normalization)

Standard scaling, or Z-score normalization, standardizes features by subtracting the mean and dividing by the standard deviation. The result has a mean of 0 and a standard deviation of 1.

$$Z = (X - \mu) / \sigma$$

Where:

- **X** is the original feature vector
- **μ** is the mean of the feature vector
- **σ** is the standard deviation of the feature vector
- **Z** is the standard score or Z-score.

Feature Scaling and Normalization - Standard Scaling (Z-score normalization)

Pros:

- Centers the distribution around 0
- Handles outliers better than Min-Max Scaling

Cons:

- Doesn't normalize the distribution to a specific range
- Can be influenced by outliers if extreme

Use when:

- Your data follows a Gaussian distribution
- Your algorithm assumes data is centered (e.g., Principal Component Analysis)

Feature Scaling and Normalization - Robust Scaling

Robust scaling uses the interquartile range, instead of the mean and standard deviation, making it robust to outliers.

$$X' = (X - Q1) / (Q3 - Q1)$$

Where:

- **X** is the original value
- **X'** is the scaled value
- **Q1** is the first quartile (25th percentile) of the feature column
- **Q3** is the third quartile (75th percentile) of the feature column

Feature Scaling and Normalization - Robust Scaling

Pros:

- Reduces the effects of outliers
- Useful for data with heavy-tailed distributions

Cons:

- Doesn't scale the data to a specific range
- May not perform well on non-Gaussian distributions

Use when:

- Your data contains many outliers
- You want to reduce the impact of outliers

Data Preprocessing - Encoding Categorical Variables

Categorical variables represent types of data which may be divided into groups. Since machine learning algorithms require numerical inputs, we need techniques to convert these categorical variables into a suitable numerical form. We'll cover three common methods:

- Label Encoding,
- One-Hot Encoding, and
- Target Encoding

Encoding Categorical Variables - Label Encoding

Label Encoding involves converting each value in a categorical column into a number. For example, if we have a feature called "Color" that has three categories: "Red", "Green", and "Blue". We could map these categories to numbers: "Red" = 1, "Green" = 2, "Blue" = 3.

Pros:

- Simple and easy to implement
- Does not increase the dimensionality of the data

Cons:

- Implies an ordered relationship between categories
- May lead to poor performance if the categorical variable is nominal

Use when:

- The categorical variable is ordinal (i.e., there's a logical order to the categories)

Encoding Categorical Variables - One-Hot Encoding

One-Hot Encoding involves creating new columns indicating the presence (or absence) of each unique value in the original data.

For example, if we have a feature called "Color" with three categories: "Red", "Green", and "Blue". One-Hot Encoding would create three new features: "Color_Red", "Color_Green", and "Color_Blue". If the original color was "Red", then "Color_Red" would be 1, and "Color_Green" and "Color_Blue" would both be 0.

Color	**Color_Red**	**Color_Green**	**Color_Blue**
Red	1	0	0
Green	0	1	0
Blue	0	0	1

Encoding Categorical Variables - One-Hot Encoding

Pros:

- Creates binary vectors, eliminating any implied order
- Suitable for nominal categories (no inherent order)

Cons:

- Can significantly increase data dimensionality
- Not suitable for categories with many unique values (high cardinality)

Use when:

- The categorical variable is nominal
- The number of unique categories is small

Encoding Categorical Variables - Target Encoding

Target Encoding, also known as mean encoding, is a method of encoding categorical variables based on the mean value of the target variable. This method can be particularly useful for high cardinality features where one-hot encoding might lead to high memory consumption.

Here's how Target Encoding works:

1. For each category in the feature, calculate the average value of the target variable.
2. Replace the category with the calculated average value.

For example, if we have a feature "City" and we want to predict the average house price (the target variable), we would replace each city name with the average house price in that city.

Encoding Categorical Variables - Target Encoding

Pros:

- Can capture information within the category, improving model performance
- Prevents high dimensionality

Cons:

- Risk of overfitting due to information leakage
- Requires careful validation strategies

Use when:

- Categorical feature is high cardinality
- There's a correlation between the category and the target variable"

Data Preprocessing - Feature Selection

Feature selection is the process of reducing the number of input variables when developing a predictive model. It is crucial to remove irrelevant or partially relevant features that can negatively impact model performance. We'll cover three common methods:

- Filter Methods,
- Wrapper Methods, and
- Embedded Methods

Feature Selection - Filter Methods

Filter methods for feature selection apply a statistical measure to assign a scoring to each feature. The features are ranked by the score and either selected to be kept or removed from the dataset. The methods are often univariate and consider the feature independently, or with regard to the dependent variable.

Here are some examples of filter methods:

1. **Pearson's Correlation:** It is used as a measure for quantifying linear dependence between two continuous variables.
2. **Chi-Squared Test:** It is used for testing the relationship between categorical variables.
3. **Mutual Information:** Mutual information measures the dependency between the features.
4. **ANOVA F-Value:** It is used to calculate the degree of linear dependency between the feature and the target variable.

Feature Selection - Filter Methods

Pros:

- Simple and easy to implement
- Fast and computationally efficient

Cons:

- Does not consider interactions between features
- Based only on the intrinsic properties of the data

Use when:

- You have a large number of features
- You want to quickly filter out irrelevant features

Feature Selection - Wrapper Methods

Wrapper methods for feature selection consider the selection of a set of features as a search problem, where different combinations are prepared, evaluated and compared to other combinations. A predictive model is used to evaluate a combination of features and assign a score based on model accuracy.

1. **Forward Selection:** Forward selection begins with an empty model and adds variables one by one. At each step, the variable that gives the highest improvement to the model is added, until no improvement is observed.
2. **Backward Elimination:** Backward elimination begins with a model including all the features. It removes variables one by one. At each step, the variable that causes the least decrease in performance is removed, until no improvement is observed.
3. **Recursive Feature Elimination (RFE):** This is a more sophisticated version of backward elimination. At each step, it removes either a fixed number or a fraction of variables. It uses model accuracy to identify which variables (and combination of variables) contribute the most to predict the target variable.

Feature Selection - Wrapper Methods

Pros:

- Searches for the best feature subset
- Considers interactions between features

Cons:

- Computationally expensive
- Prone to overfitting

Use when:

- You have a moderate number of features
- Computational resources and time are not primary constraints

Feature Selection - Embedded Methods

Embedded methods are a catch-all group of techniques which perform feature selection as part of the model construction process. They are constructed by algorithms that have their own built-in feature selection methods.

1. **LASSO Regression:** LASSO (Least Absolute Shrinkage and Selection Operator) adds a penalty equivalent to the absolute value of the magnitude of coefficients. This can result in some coefficients to become zero, hence eliminating them from the model.
2. **RIDGE Regression:** RIDGE adds a penalty equivalent to square of the magnitude of coefficients. All coefficients are shrunk by the same factor (so none are eliminated), but this can help multicollinearity issues.
3. **ElasticNet:** It is a combination of both LASSO and RIDGE. It adds both L1 (absolute) and L2 (squared) penalties on the coefficients.
4. **Tree-based methods:** Decision Trees, Random Forests, and Gradient Boosting algorithms can also provide feature importance which can be used for feature selection.

Feature Selection - Embedded Methods

Pros:

- More accurate than filter methods
- Faster than wrapper methods

Cons:

- Can be complex to implement
- Prone to overfitting if not handled correctly

Use when:

- You want to optimize the performance of the model
- You need a balance between performance and speed

Data Preprocessing - Feature Engineering

Feature engineering is the process of creating new features or modifying existing ones to improve the performance of machine learning models. We will discuss two approaches:

- Domain Knowledge-Based Feature Engineering and
- Automated Feature Engineering.

Feature Engineering - Domain Knowledge-Based Feature Engineering

- Utilizes specialized, field-specific knowledge to enhance datasets.
- Unveils key relationships or critical details that may not be immediately evident.
- Engineers new features that can improve the performance of machine learning models.
- Can adjust existing features based on expert insights.

Example:

Suppose we have data on houses, with features like **total price** and **size in square meters**. A real estate expert might know that price per square meter is crucial in determining house prices.

- Recognize the importance of price per square meter.
- Create a new feature: 'price_per_meter'.
- Calculate it by dividing total price by total square meter.

This new feature, 'price_per_meter', might now be a stronger predictor of house prices than the original features.

Feature Engineering - Domain Knowledge-Based Feature Engineering

Pros:

- Can significantly improve model performance
- Can provide interpretable features that make sense in the given context

Cons:

- Requires significant domain knowledge
- Can be time-consuming

Use when:

- You have a strong understanding of the domain
- Interpretability of the model is important

Feature Engineering - Automated Feature Engineering

Automated Feature Engineering is a process that involves automatically creating new features from the existing ones in a dataset. It can help to quickly create and test a large number of features, which can be particularly beneficial when dealing with high-dimensional data or when there's a lack of domain knowledge.

Automated Feature Engineering can involve a variety of techniques, such as:

- **Polynomial Features:** This involves creating new features by taking the polynomial of existing features. For example, if you have a feature **x**, you could create new features $\mathbf{x}^2$, $\mathbf{x}^3$, etc.
- **Interaction Features:** This involves creating new features by taking the **product of pairs** of features. For example, if you have features x and y, you could create a new feature xy.
- **Aggregation:** This involves creating new features by **aggregating data**. For example, you might create a feature that represents the **mean or maximum** value of a group of features.

Feature Engineering - Automated Feature Engineering

Pros:

- Saves time and effort
- Can discover complex patterns across multiple tables of data

Cons:

- May generate a large number of features, leading to dimensionality problems
- Features generated may not be as interpretable

Use when:

- You have a large dataset with complex relationships
- You lack domain knowledge or time for manual feature engineering"

Evaluating the Impact of Data Cleaning and Preprocessing on Model Performance

The ultimate goal of data cleaning and preprocessing is to improve model performance. In this section, we'll discuss how to evaluate the impact of these steps on model performance using

- Cross-Validation,
- Model Performance Metrics, and by
- Comparing Model Performance With and Without Preprocessing.

Cross-Validation

Cross-Validation is a resampling procedure used to evaluate machine learning models on a limited data sample. It helps to assess how well your model will generalize to an independent data set.
The most common type is k-fold cross-validation, where the original sample is randomly partitioned into k equal-sized subsamples. The model is trained on k-1 folds, and the resulting model is validated on the remaining part.

Pros:

- Reduces overfitting by using different portions of data
- Provides a more accurate estimate of model performance

Cons:

- Can be computationally intensive, especially for large datasets
- Might not be suitable for time-series data

Model Performance Metrics

Model performance metrics help quantify the quality of your machine learning model.

For **classification** problems, metrics such as **Accuracy, Precision, Recall, F1 score**, and **AUC-ROC** are commonly used.

For **regression** problems, we use metrics like **Mean Absolute Error (MAE), Mean Squared Error (MSE)**, and **Root Mean Squared Error (RMSE)**.

It's important to choose the right metric that aligns with your objective and model goals.

Comparing Model Performance With and Without Preprocessing

A direct way to understand the impact of data cleaning and preprocessing is by comparing model performance with and without these steps.

You can train your model on the raw data and then on the preprocessed data, and compare the performance using the chosen metrics.

Keep in mind:

- Significant improvements suggest your preprocessing steps are beneficial.
- Little or no improvement might suggest you revisit your preprocessing steps.
- The goal is not just improvement, but also robustness and interpretability of your model.

Case Study - Impact of Data Cleaning and Preprocessing Techniques on a Real-World Dataset

Titanic Dataset

https://t.ly/ZZSV

Understanding the Titanic Dataset

The Titanic dataset is a classic, widely used dataset in data science and machine learning.

It contains demographic and travel information for **1,309 Titanic passengers**.

The dataset includes a **mix of textual, Boolean, continuous, and categorical variables**.

The **goal** is to **predict the survival** of these passengers based on these variables.

The dataset provides a rich resource for demonstrating data transformations, dealing with missing values, and other data preprocessing steps.

Preparing the Titanic Data for Machine Learning

- The dataset undergoes several preprocessing steps to make it suitable for machine learning.
- Missing values in the 'Age' attribute are filled with the median age.
- The 'Cabin' attribute is converted into a binary variable indicating whether a cabin was assigned or not.
- 'Sex' is label-encoded, and 'Embarked' is one-hot encoded.
- Two new features 'FamilySize' and 'IsAlone' are created from 'SibSp' and 'Parch'.
- 'Age' and 'Fare' are standardized to have zero mean and unit variance.

Predicting Survival with Logistic Regression

- The initial model is a Logistic Regression model, a simple yet powerful classification algorithm.
- The model is trained using the processed features, with 'Survived' as the target variable.
- The performance of the model is evaluated using various metrics, including accuracy, precision, recall, and F1 score.
- The model is also cross-validated to ensure its performance is robust and not due to overfitting on the training data.
- The results provide insights into the factors that influenced survival on the Titanic and serve as a basis for further model refinement and feature engineering.

Q&A Session

Thank you for your attention!

Please feel free to ask your questions.