**Data Preprocessing techniques and usage**

Improves Data Quality

Data preprocessing is the fast track to improving data quality since many of its steps mirror activities you’ll find in any data quality management process, such as data cleansing, data profiling, data integration, and more.

Handles Missing Data

There are several reasons why a data collection may be missing values (fields of data). Data practitioners must determine if it’s best to reject records with missing values, ignore them, or fill them in with an estimated value.

Normalizes and Scales Data

Dependent and independent variables change on separate scales, or one changes linearly while another changes exponentially. Salary, for example, might be a multiple-figure digit, whereas age is expressed in double digits. Normalizing and scaling help to modify data in a way that allows computers to extract a meaningful link between these variables.

Eliminates Duplicate Records

When two records appear to repeat, an algorithm must identify whether the same metric was captured twice or whether the data reflects separate occurrences. In rare circumstances, a record may have minor discrepancies due to an erroneously reported field. Techniques for finding, deleting, or connecting duplicates help to address such data quality issues automatically.

Handles Outliers

Data practitioners sometimes need to merge many data sources to construct a new machine learning model. Principal component analysis, for example, is an important technique for lowering the number of dimensions in the training data set and producing a more efficient representation.

Helps in Enhancing Model Performance

Preprocessing often entails developing new features or modifying existing ones to better capture the underlying problem and enhance model performance. This might include encoding category variables, developing interaction terms, and retrieving pertinent data from text or timestamps.

Data Preprocessing Steps

Acquire the Dataset

Naturally, data collection is the first step in any machine learning project and the first among the data preprocessing steps. Gathering data might seem like a straightforward process, but it’s far from that.

Most companies end up with data kept in silos and divide it across many departments, teams, and digital solutions. For example, the marketing team might have access to a CRM system, but that system may operate in isolation from the web analytics solution. Combining all data streams into consolidated storage will be challenging.

Import Libraries

Next, it’s time to import the libraries you’ll need for your machine learning project. A library is a collection of functions that an algorithm can call and utilize.

You can streamline data preprocessing procedures using tools and frameworks that make the process easier to organize and execute. Without certain libraries, one-liner solutions might take hours to code and optimize.

Import Datasets

The next key step is to load the data that will be utilized in the machine learning algorithm. This is the most critical machine learning preprocessing step.

Many companies start by storing data in warehouses that require data to pass through an ETL. The problem with this method is that you never know which data will be useful for an ML project. As a result, warehouses are commonly used to access data through business intelligence interfaces to observe metrics that we know we need to monitor.

Data lakes are used for both structured and unstructured data, including photos, videos, voice recordings, and PDF files. However, even when data is structured, it’s not transformed prior to storage. You load the data in its present condition and then decide how to use and alter it later.

Check for Missing Values

Evaluate the data and look for missing values. Missing values can break actual data trends and potentially result in additional data loss when entire rows and columns are deleted due to a few missing cells in the dataset.

If you discover any, you can choose from two methods to deal with this issue:

* Remove the whole row with a missing value. However, eliminating the full row increases the likelihood of losing some critical data. This strategy is beneficial if the dataset is massive.
* Estimate the value using the mean, median, or mode.

Encode the Data

Non-numerical data is incomprehensible to machine learning modules. To avoid issues later, the data should be arranged numerically. The answer to this problem is to convert all text values to numerical form.

Scaling

Scaling is unnecessary for non-distance-based algorithms (such as the decision tree). Distance-based models, on the other hand, require all features to be scaled.

These are some of the more common scaling approaches:

7. Split Dataset Into Training, Evaluation and Validation Sets

This is the final step among the data preprocessing steps. It’s time to divide your dataset into training, evaluation, and validation sets. The training set is the data you’ll use to train your machine learning model. The evaluation set will assess the data and the model, while the validation set will validate it.

Data Preprocessing Examples and Techniques

Data Transformation

One of the most important stages in the preparation phase is data transformation, which changes data from one format to another. Some algorithms require that the input data be changed – if you fail to finish this process, you may receive poor model performance or even introduce bias.

For example, the KNN model uses distance measurements to determine which neighbours are closest to a particular record. If you have a feature with a particularly high scale relative to the other features in your model, your model will likely employ this feature more than the others, resulting in a bias.

Feature Engineering

The feature engineering strategy is used to produce better features for your dataset, which will improve the model’s performance. We mostly employ domain knowledge to produce those features, which we manually generate from existing features after applying a transformation to them.

Here are some simple examples to help you understand this:

Imagine that you have a hair color feature in your data with values of brown, black, or unknown. In this scenario, you may add a new column named “has color” and assign 1 if there is a color and 0 if the value is unknown.

Another example is deconstructing a date/time feature, which provides significant information but is difficult for a model to use in its original format. So, if you believe your problem involves temporal dependencies and you discover a link between the date/time and the output variable, spend some time trying to turn that date/time column into a more intelligible feature for your model, such as “period of the day,” “day of the week,” or so on.

Imbalanced Data

One of the most prevalent issues you may encounter while working with real-world data categorization is that the classes are unbalanced (one contains more samples than the other), resulting in a significant bias for the model.

Imagine that you’d like to forecast if a transaction is fraudulent. Based on your training data, 95% of your dataset consists of legitimate transaction records, whereas just 5% consists of fraudulent transactions. Based on this, your model will most likely forecast the majority class, identifying fraudulent transactions as usual.