DATA PREPROCESSING PROCESS

Introduction

Data mining is a methodology in computer science for discovering meaningful patterns and knowledge from large amounts of data. However, before a data mining model can be applied, the raw data must be preprocessed to ensure that it is in a suitable format for analysis. Data preprocessing is an essential step in the data mining process and can greatly impact the accuracy and efficiency of the final results.

This article provides a hands-on guide to data preprocessing in data mining. We will cover the most common data preprocessing techniques, including data cleaning, data integration, data transformation, and feature selection. With practical examples and code snippets, this article will help you understand the key concepts and techniques involved in data preprocessing and equip you with the skills to apply them to your own data mining projects. Whether you are a beginner or an experienced data miner, this guide will be a valuable resource to help you achieve high-quality results from your data.

Learning Objectives

- Key concepts and techniques in data preprocessing.
- Importance of data preprocessing in data mining.
- Define and understand data cleaning, data integration, data transformation, and feature selection.
- Implement data preprocessing in machine learning.

What Is Data Preprocessing? Why Is It Important?

Data preprocessing is the process of transforming raw data into an understandable format. It is also an important step in data mining as we cannot work with raw data. The quality of the data should be checked before applying machine learning or data mining algorithms.

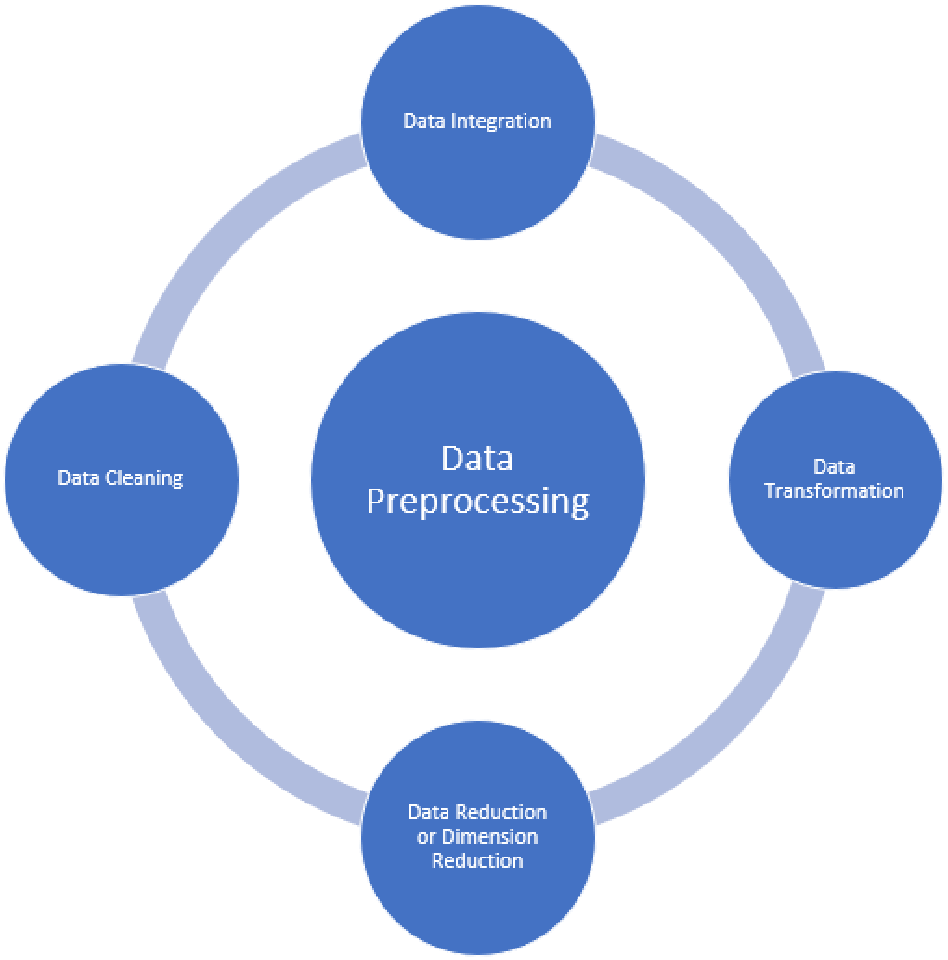
Why Is Data Preprocessing Important?

Preprocessing of data is mainly to check the data quality. The quality can be checked by the following:

- **Accuracy**: To check whether the data entered is correct or not.
- **Completeness**: To check whether the data is available or not recorded.
- **Consistency:** To check whether the same data is kept in all the places that do or do not match.
- **Timeliness**: The data should be updated correctly.
- **Believability**: The data should be trustable.
- **Interpretability**: The understandability of the data.

Major Tasks in Data Preprocessing

There are 4 major tasks in data preprocessing – Data cleaning, Data integration, Data reduction, and Data transformation.



Source: medium.com

1. Data Cleaning

Data cleaning is the process of removing incorrect data, incomplete data, and inaccurate data from the datasets, and it also replaces the missing values. Here are some techniques for data cleaning:

Handling missing values

- Standard values like “Not Available” or “NA” can be used to replace the missing values.
- Missing values can also be filled manually, but it is not recommended when that dataset is big.
- The attribute’s mean value can be used to replace the missing value when the data is normally distributed wherein in the case of non-normal distribution median value of the attribute can be used.
- While using regression or decision tree algorithms, the missing value can be replaced by the most probable value.

Handling noisy data

Noisy generally means random error or containing unnecessary data points. Handling noisy data is one of the most important steps as it leads to the optimization of the model we are using Here are some of the methods to handle noisy data.

- **Binning:** This method is to smooth or handle noisy data. First, the data is sorted then, and then the sorted values are separated and stored in the form of bins. There are three methods for smoothing data in the bin. **Smoothing by bin mean method**: In this method, the values in the bin are replaced by the mean value of the bin; **Smoothing by bin median**: In this method, the values in the bin are replaced by the median value; **Smoothing by bin boundary**: In this method, the using minimum and maximum values of the bin values are taken, and the closest boundary value replaces the values.

- **Regression:** This is used to smooth the data and will help to handle data when unnecessary data is present. For the analysis, purpose regression helps to decide the variable which is suitable for our analysis.
- **Clustering:** This is used for finding the outliers and also in grouping the data. Clustering is generally used in unsupervised learning.

2. Data Integration

The process of combining multiple sources into a single dataset. The Data integration process is one of the main components of data management. There are some problems to be considered during data integration.

- **Schema integration**: Integrates metadata(a set of data that describes other data) from different sources.
- **Entity identification problem:** Identifying entities from multiple databases. For example, the system or the user should know the student *id of one database and student*name of another database belonging to the same entity.
- **Detecting and resolving data value concepts**: The data taken from different databases while merging may differ. The attribute values from one database may differ from another database. For example, the date format may differ, like “MM/DD/YYYY” or “DD/MM/YYYY”.

3. Data Reduction

This process helps in the reduction of the volume of the data, which makes the analysis easier yet produces the same or almost the same result. This reduction also helps to reduce storage space. Some of the data reduction techniques are dimensionality reduction, numerosity reduction, and data compression.

- **Dimensionality reduction:** This process is necessary for real-world applications as the data size is big. In this process, the reduction of random variables or attributes is done so that the dimensionality of the data set can be reduced. Combining and merging the attributes of the data without losing its original characteristics. This also helps in the reduction of storage space, and computation time is reduced. When the data is highly dimensional, a problem called the "Curse of Dimensionality" occurs.
- **Numerosity Reduction**: In this method, the representation of the data is made smaller by reducing the volume. There will not be any loss of data in this reduction.
- **Data compression**: The compressed form of data is called data compression. This compression can be lossless or lossy. When there is no loss of information during compression, it is called lossless compression. Whereas lossy compression reduces information, but it removes only the unnecessary information.

4. Data Transformation

The change made in the format or the structure of the data is called data transformation. This step can be simple or complex based on the requirements. There are some methods for data transformation.

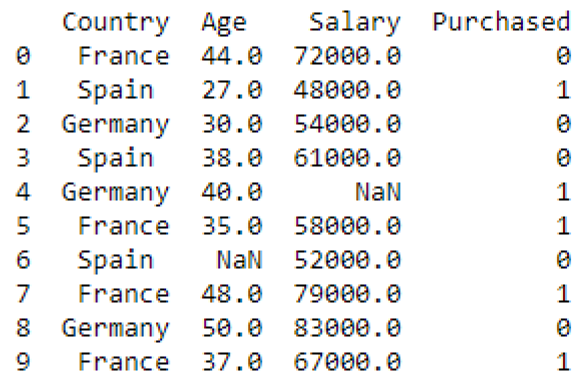
- **Smoothing:** With the help of algorithms, we can remove noise from the dataset, which helps in knowing the important features of the dataset. By smoothing, we can find even a simple change that helps in prediction.
- **Aggregation:** In this method, the data is stored and presented in the form of a summary. The data set, which is from multiple sources, is integrated into with data analysis description. This is an important step since the accuracy of the data depends on the quantity and quality of the data. When the quality and the quantity of the data are good, the results are more relevant.
- **Discretization:** The continuous data here is split into intervals. Discretization reduces the data size. For example, rather than specifying the class time, we can set an interval like (3 pm-5 pm, or 6 pm-8 pm).
- **Normalization:** It is the method of scaling the data so that it can be represented in a smaller range. Example ranging from -1.0 to 1.0.

Data Preprocessing Steps in Machine Learning

Step 1: Importing libraries and the dataset

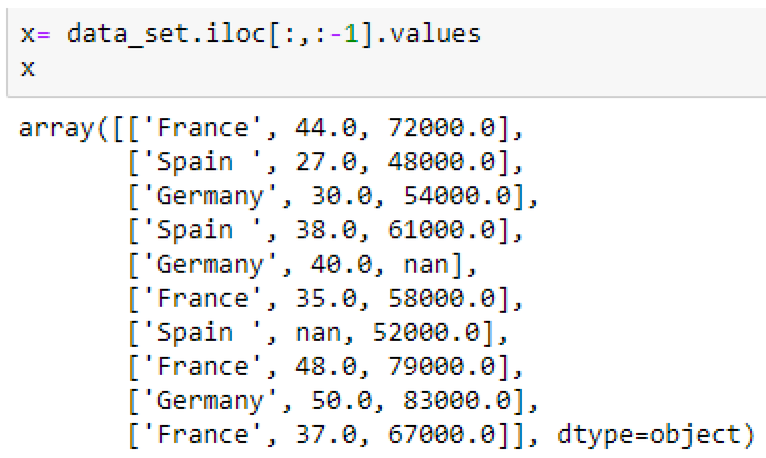
import pandas as pd
import numpy as np
dataset = pd.read_csv('Data.csv')
print (dataset)

Python Code:



```
   Country  Age   Salary  Purchased
0   France  44.0  72000.0          0
1   Spain   27.0  48000.0          1
2  Germany  30.0  54000.0          0
3   Spain   38.0  61000.0          0
4  Germany  40.0      NaN          1
5   France  35.0  58000.0          1
6   Spain    NaN  52000.0          0
7   France  48.0  79000.0          1
8  Germany  50.0  83000.0          0
9   France  37.0  67000.0          1
```

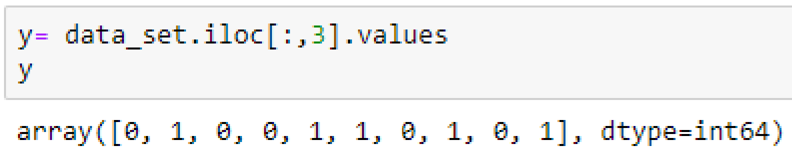
Step 2: Extracting the independent variable



```
x= data_set.iloc[:,:-1].values
x
```

```
array([['France', 44.0, 72000.0],
       ['Spain ', 27.0, 48000.0],
       ['Germany', 30.0, 54000.0],
       ['Spain ', 38.0, 61000.0],
       ['Germany', 40.0, nan],
       ['France', 35.0, 58000.0],
       ['Spain ', nan, 52000.0],
       ['France', 48.0, 79000.0],
       ['Germany', 50.0, 83000.0],
       ['France', 37.0, 67000.0]], dtype=object)
```

Step 3: Extracting the dependent variable

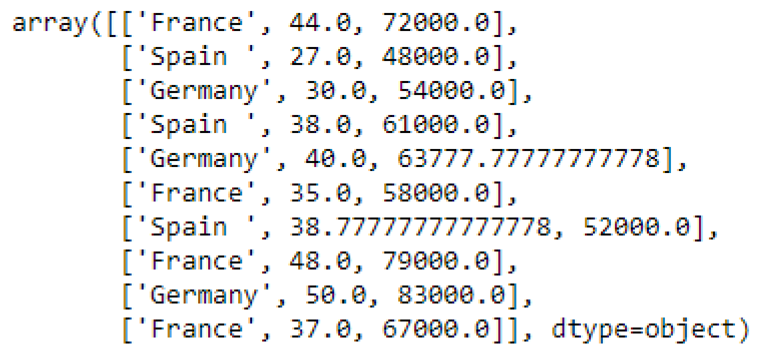


```
y= data_set.iloc[:,3].values
y
```

```
array([0, 1, 0, 0, 1, 1, 0, 1, 0, 1], dtype=int64)
```

Step 4: Filling the dataset with the mean value of the attribute

```
from sklearn.preprocessing import Imputer
imputer= Imputer(missing_values ='NaN', strategy='mean', axis = 0)
imputerimputer= imputer.fit(x[:, 1:3])
x[:, 1:3]= imputer.transform(x[:, 1:3])
x
```

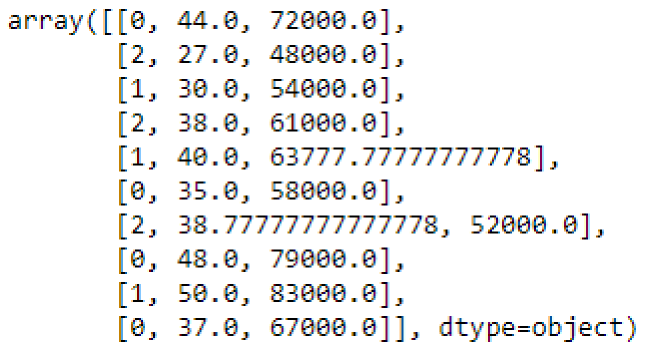


```
array([['France', 44.0, 72000.0],
       ['Spain ', 27.0, 48000.0],
       ['Germany', 30.0, 54000.0],
       ['Spain ', 38.0, 61000.0],
       ['Germany', 40.0, 63777.77777777778],
       ['France', 35.0, 58000.0],
       ['Spain ', 38.77777777777778, 52000.0],
       ['France', 48.0, 79000.0],
       ['Germany', 50.0, 83000.0],
       ['France', 37.0, 67000.0]], dtype=object)
```

Step 5: Encoding the country variable

The machine learning models use mathematical equations. So categorical data is not accepted, so we convert it into numerical form.

```
from sklearn.preprocessing import LabelEncoder
label_encoder_x= LabelEncoder()
x[:, 0]= label_encoder_x.fit_transform(x[:, 0])
```



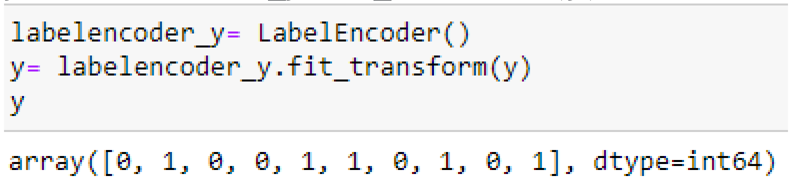
```
array([[0, 44.0, 72000.0],
       [2, 27.0, 48000.0],
       [1, 30.0, 54000.0],
       [2, 38.0, 61000.0],
       [1, 40.0, 63777.77777777778],
       [0, 35.0, 58000.0],
       [2, 38.77777777777778, 52000.0],
       [0, 48.0, 79000.0],
       [1, 50.0, 83000.0],
       [0, 37.0, 67000.0]], dtype=object)
```

Step 6: Dummy encoding

These dummy variables replace the categorical data as 0 and 1 in the absence or the presence of the specific categorical data.

Encoding for purchased variable

```
labelencoder_y= LabelEncoder()
y= labelencoder_y.fit_transform(y)
```



```
labelencoder_y= LabelEncoder()
y= labelencoder_y.fit_transform(y)
y
```

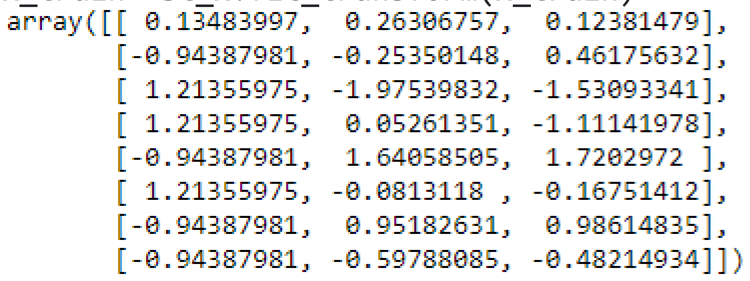
```
array([0, 1, 0, 0, 1, 1, 0, 1, 0, 1], dtype=int64)
```

Step 7: Splitting the dataset into training and test sets

```
 from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size=
0.2, random_state=0)
```

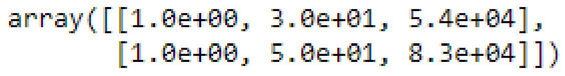
Step 8: Feature scaling

```
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
```



```
array([[ 0.13483997,  0.26306757,  0.12381479],
       [-0.94387981, -0.25350148,  0.46175632],
       [ 1.21355975, -1.97539832, -1.53093341],
       [ 1.21355975,  0.05261351, -1.11141978],
       [-0.94387981,  1.64058505,  1.7202972 ],
       [ 1.21355975, -0.0813118 , -0.16751412],
       [-0.94387981,  0.95182631,  0.98614835],
       [-0.94387981, -0.59788085, -0.48214934]])
```

```
x_test= st_x.transform(x_test)
```



```
array([[1.0e+00, 3.0e+01, 5.4e+04],
       [1.0e+00, 5.0e+01, 8.3e+04]])
```

Conclusion

In conclusion, data preprocessing is an essential step in the data mining process and plays a crucial role in ensuring that the data is in a suitable format for analysis. This article provides a comprehensive guide to data preprocessing techniques, including data cleaning, integration, reduction, and transformation. Through practical examples and code snippets, the article helps readers understand the key concepts and techniques involved in data preprocessing and gives them the skills to apply these techniques to their own data mining projects. Whether you are a beginner or an experienced data miner, this article will provide valuable information and resources to help you achieve high-quality results from your data.

Key Takeaways

- The quality of data is checked based on its accuracy, completeness, consistency, timeliness, believability, and interpretability.
- The 4 major tasks in data preprocessing are data cleaning, data integration, data reduction, and data transformation.
- The practical examples and code snippets mentioned in this article have helped us better understand the application of data preprocessing in data mining.

Frequently Asked Question

Q1. What is the meaning of data cleansing?

A. Data cleansing is the process of identifying and removing errors, inconsistencies and duplicate records from a dataset. The goal is to improve the accuracy, completeness, and consistency of data. Data cleansing can involve tasks such as correcting inaccuracies, removing duplicates, and standardizing data formats. This process helps to ensure that data is reliable and trustworthy for business intelligence, analytics, and decision-making purposes.

Q2. What are the data preprocessing steps in order?

A. The steps involved in data preprocessing are: Data collection, Data cleaning, Data integration, Data transformation, Data reduction, Data discretization, Data normalization or Data standardization, Feature selection, and Data representation.

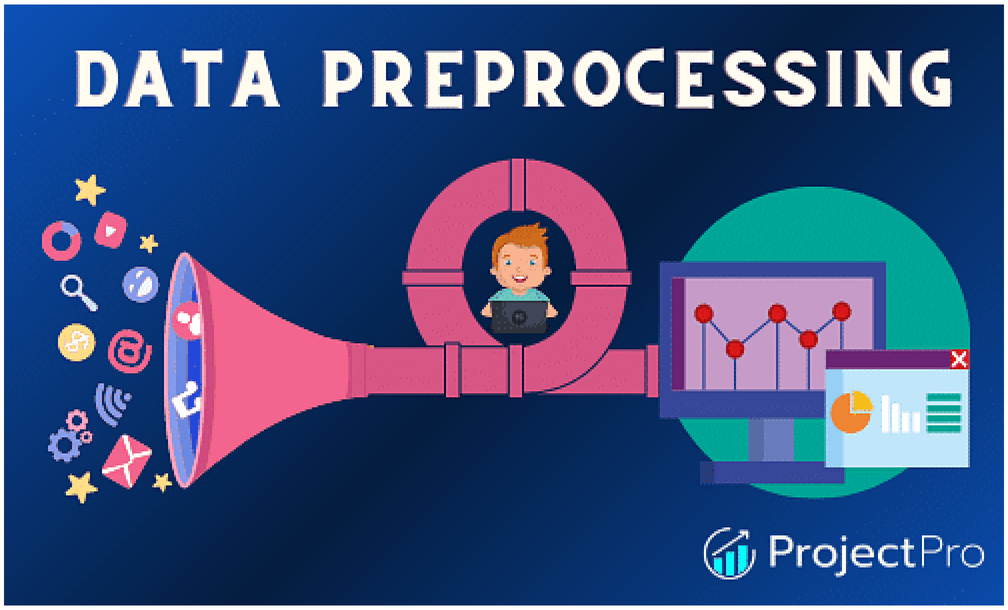
Q3. What is the difference between data mining and data preprocessing?

A. Data mining is the process of discovering patterns and insights from large amounts of data, while data preprocessing is the initial step in data mining which involves preparing the data for analysis. Data preprocessing involves cleaning and transforming the data to make it suitable for analysis. The goal of data preprocessing is to make the data accurate, consistent, and suitable for analysis. It helps to improve the quality and efficiency of the data mining process.

MORE MATERIAL ON DATA PREPROCESSING

What is Data Preprocessing?

Data preprocessing is a step that involves transforming raw data so that issues owing to the incompleteness, inconsistency, and/or lack of appropriate representation of trends are resolved so as to arrive at a dataset that is in an understandable format.



Data Preprocessing in Data Mining

Data mining is the process of uncovering patterns, drawing insights, and other valuable information by sifting through large volumes of data. Business intelligence and data analytics teams widely use data mining techniques in domains like sales and marketing and operational optimization.

With the size of the datasets used for data mining, the data preprocessing step is such a vital part of data mining that it has come to be known as a data mining technique. When employed before mining, data processing techniques can significantly improve the quality of the patterns mined and/or the time required for the actual mining.

It is no secret that the data mining techniques like clustering, classification, regression, and even deep learning models perform better when the data is well prepared. This is why we will get back to the über important topic of improving data quality by preprocessing in the later section... But before we get there, let’s take a short detour to understand what data quality is all about fully.

Understanding the Multi-Dimensionality of Data Quality

Data is said to be of good quality as long as it can satisfy the requirements of its intended use factors that makeup data quality include accuracy, completeness, consistency, timeliness, believability, and interpretability. Before moving on to the steps to improve data quality, let us spend a moment in this section to understand just what it is we seek to change.

1. Accuracy

Accuracy refers to how well the information recorded reflects a real event or object. Data inaccuracy can be caused by faulty data collection instruments or computer errors, purposeful submission of incorrect data values by users, errors in data transmission, inconsistencies in naming conventions and input formats, etc.

2. Completeness

Data is considered “complete” when all the mandatory or necessary features are present. The incompleteness of data can occur due to unavailability of requisite information, equipment malfunctions during data collection, unintended deletion, or failure to record history or modifications.

3. Consistency

If the same information stored and used at multiple instances matches, with or without formatting inconsistencies between the various sources or datastores, then the data is consistent. It is quantitatively expressed as the percentage of values that match across the different stored instances.

4. Timeliness

Timeliness also affects data quality as data is of value only when it is available when needed. If the data is outdated or the corrections are incorporated post evaluations or analysis of the dataset, the data quality is affected.

5. Believability

The believability describes the trust the users have in the data. If the data was at any point found to be rife with errors and inconsistencies, its users will likely harbor reservations when it comes to using this data in the future. As this hinders the effective use of data for its intended purpose, believability is also a factor in deciding data quality.

6. Interpretability

Interpretability of data defines how easy it is to understand the information present in the dataset and derive meaning from it. The availability of statistical data collection and processing methodologies for the users can affect the interpretability of the dataset.

While the six dimensions described for data quality assessment in this section are by no means a comprehensive list, they should serve to introduce you to the multi-dimensionality of data quality. The objective of defining the data quality dimensions remains the same- to ensure that the data can satisfy the requirements of its intended use! Therefore, fret not if you find widely different sets of data quality dimensions when you refer to various sources.

Data Preprocessing Steps in Machine Learning

While there are several varied data preprocessing techniques, the entire task can be divided into a few general, significant steps: data cleaning, data integration, data reduction, and data transformation.

1. Data Cleaning

The tasks involved in data cleaning can be further subdivided as:

- **Data and Metadata Acquisition:** Data stored in a DBMS will be retrieved using ODBC or JDBC protocols. You must also retrieve metadata regarding field types, roles, and descriptions. If the data is contained in a flat file, the columns will need to be separated by using the delimiter to verify the consistency i the number of fields.
- **Handling Missing Values:** There are various ways of handling missing values, namely:
    - Ignoring the tuple- Simply discard the tuples with that missing information. This is feasible for large data sets, where ignoring a small proportion of tuples wouldn’t significantly affect the further analysis.
    - Manually filling in missing values can be an extremely tiresome process but might be necessary, especially when the dataset is small.
    - Assigning a constant to all missing values
    - Imputation- Done by replacing the mean of all samples, the mean of samples with similar classification or resultant values, or any other logical manner.

    - Replacing the missing value by the most probable value can be done using the Bayesian formula, decision tree, etc.

- **Reformatting:** This involves making data format changes into a standard format to ensure that the attributes such as date have a similar format throughout, performing binning of numerical values, and detecting and handling errors.
- **Attribute Conversions:** Since some methods require only numerical inputs, different strategies need to be employed to handle binary, ordered, multi-valued nominal fields. These may include using 0 and 1 for binary fields, numbers preserving order for ordered nominal attributes, and integer or one-hot encoding for unordered attributes.

- **Outlier Identification and Smoothening Out Noisy Data:** Noise is a random error in a quantifiable variable. Some of the commonly used to remove the noise or smoothen the data are as follows:
    - Binning- This involves sorting numerical values into some bins. This can be done using equal-width bins, resulting in a uniform grid, or equal-depth bins, which handle skewed data better. Besides smoothing, binning is also used as a discretization technique.
    - Regression- In this method, smoothing is achieved by fitting the data into regression functions.
    - Outlier analysis- One technique for detecting outliers is clustering, wherein values that fall well outside the set of clusters encompassing the data points are considered outliers and subsequently removed.

2. Data Integration

Data integration is the process of combining data from multiple sources into a single dataset. This involves schema integration, i.e., the integration of metadata from the different sources and resolving data value conflicts that may arise from differences in units of measurement, representation, etc. Further, there is a need to handle redundant data through methods such as correlational analysis to ensure good data quality after data integration.

3. Data Reduction

Data reduction techniques aim to derive a reduced representation of the data in terms of volume while closely maintaining the integrity of the original data.

The various data reduction strategies include:

- **Dimensionality Reduction:** Dimensionality reduction is done by reducing the number of attributes to be considered. Some dimensionality reduction methods are:

    - Wavelet transforms- This involves converting the data vector into a vector of wavelet coefficients. This wavelet transformed data can then be truncated to obtain a compressed approximation of the original data by storing only a fraction of the strongest of the wavelet coefficients.
    - Principal component analysis - Principal component analysis or PCA works by searching for a set of orthogonal vectors, which is smaller than the original attribute vectors, that can best represent the data, thus resulting in dimensionality reduction. PCA compounds the original attributes into an alternative, smaller set.

    - Attribute subset selection- Involves selecting a set of features such that the weakly relevant or redundant features are removed. You can use heuristic methods such as stepwise forward selection, stepwise backward elimination, or a combination of the two, and decision tree induction to arrive at the subset of attributes.
- **Numerosity Reduction:** This involves replacing the original data with smaller forms of data representation to achieve volume reduction. The two types of methods used for this purpose are:
    - Parametric: These methods involve regression and log-linear models, whose parameters need to be stored instead of the actual data.
    - Nonparametric. Nonparametric methods involve storing the data in representations like histograms, clusters, a smaller sample of the original dataset, or data cube aggregation.
- **Data Compression:** This involves applying transformations to obtain a compressed representation of the original data. Depending on whether the reconstruction can be done with or without the loss of information the technique is called lossless or lossy compression. Dimensionality reduction and numerosity reduction techniques are also considered forms of data compression.

4. Data Transformation and discretization

This step involves converting data into a form appropriate for mining. While data smoothing, feature construction, and aggregation are also part of the data transformation step. However, they are likely to have been performed in the data cleaning or data reduction phase due to the wide overlap between the data preprocessing steps. Besides this, normalization, discretization, and concept hierarchy generation (the latter two of which are also part of data reduction) are considered parts of this step.

To minimize redundancy, let us go over the details of only the last three steps in detail:

- **Normalization:** This involves scaling of attributes to ensure that they fall within a uniform range. In the absence of normalization, the attributes with larger magnitudes will have a greater weight innately.
- **Discretization** involves reducing the number of values for a continuous attribute by partitioning the attribute range into intervals to replace actual data values.

Discretization can be done by binning, histogram analysis, clustering, decision tree analysis, and correlation analysis.

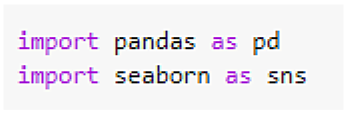
- **Concept hierarchy generation:** This involves reducing the data by changing the granularity level of the nominal attributes. For instance, replacing a location attribute describing the location in states with a variable describing the location by country reduces the number of unique values.

Implementation Examples of Various Data Preprocessing Techniques

Now that we have an overview of the steps to achieve data preprocessing let's get to the fun part- Actual Implementation!

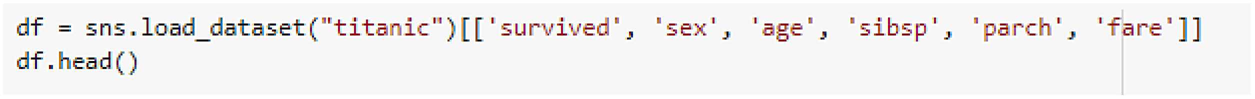
Machine Learning Data Preprocessing in Python

1. Let's start by importing the necessary libraries. For this example, we will use only pandas and seaborn.

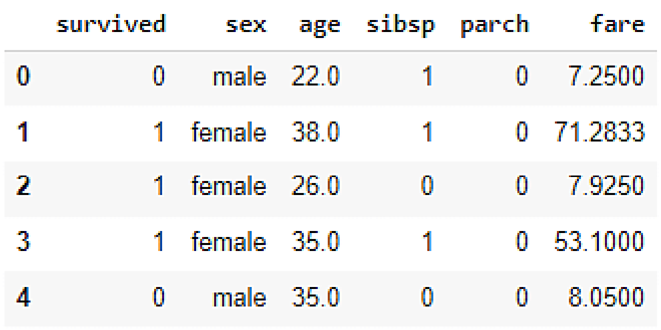


```
import pandas as pd
import seaborn as sns
```

2. Import the required columns from the seaborn 'titanic' dataset as follows,



```
df = sns.load_dataset("titanic")[['survived', 'sex', 'age', 'sibsp', 'parch', 'fare']]
df.head()
```



	survived	sex	age	sibsp	parch	fare
0	0	male	22.0	1	0	7.2500
1	1	female	38.0	1	0	71.2833
2	1	female	26.0	0	0	7.9250
3	1	female	35.0	1	0	53.1000
4	0	male	35.0	0	0	8.0500

The first five rows of the dataset will be displayed as shown above.

3. To achieve basic data cleaning, start by eliminating the na values using dropna() function.

```
print("Original length of dataframe: ", len(df))
df.dropna(inplace= True)
print("Length of dataframe after dropping na values: ", len(df))
```

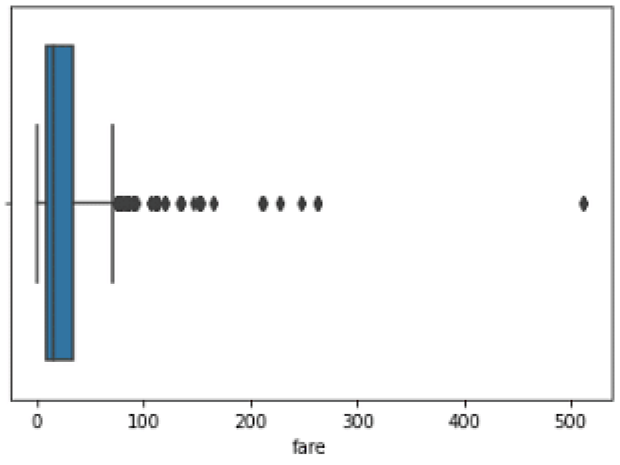
```
Original length of dataframe:  891
Length of dataframe after dropping na values:  714
```

You can observe the change in the length of the dataframe following this operation.

4. To eliminate the outliers, first, check for the presence of outliers using a boxplot. This can be done as follows,

```
sns.boxplot(x=df["fare"])
quantile1 = df["fare"].quantile(0.25)
quantile3 = df["fare"].quantile(0.75)
IQR = quantile3-quantile1

print("Q1: ", quantile1)
print("Q3: ", quantile3)
print("IQR: ", IQR)

lower_inner_fence = quantile1 - 1.5*IQR
upper_inner_fence = quantile3 + 1.5*IQR
```

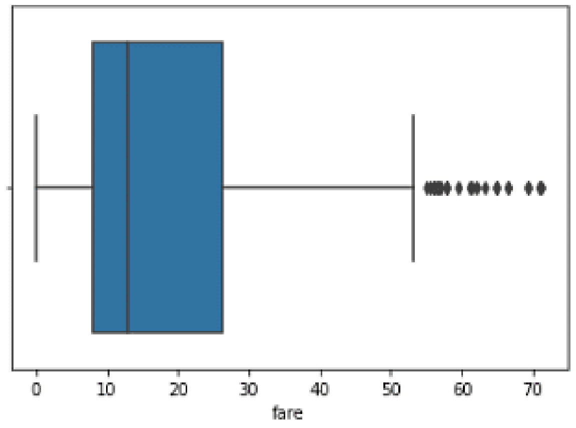
```
Q1:  8.05
Q3:  33.375
IQR:  25.325
```



5. Using the boundaries lower_inner_fence and upper_inner_fence defined above, eliminate the outliers.

```
print("Original length of dataframe: ", len(df))
outliers_index= df[(df['fare']<lower_inner_fence) | (df['fare']>upper_inner_fence)].index
df.drop(outliers_index, inplace = True)
sns.boxplot(x=df["fare"])
print("Number of outliers: ",len(outliers_index))
print("Length of dataframe after dropping outliers: ", len(df))
```

```
Original length of dataframe:  714
Number of outliers:  94
Length of dataframe after dropping outliers:  620
```



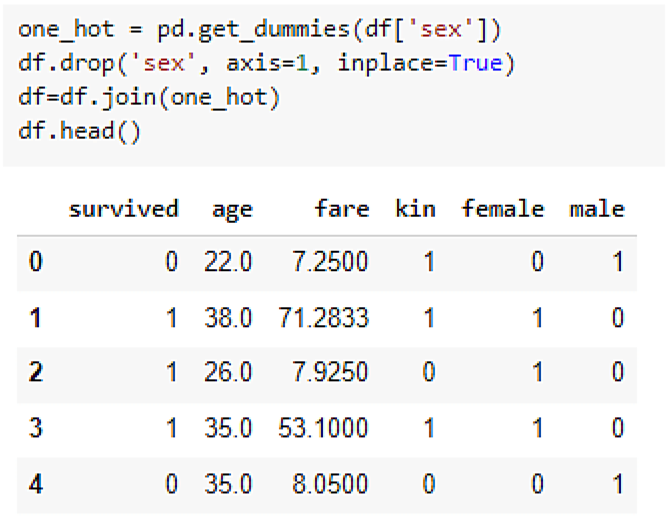
Observe that the boxplot, as well as the length of the dataframe, has now changed.

6. Achieve data reduction by combining two closely related columns into one, as shown below

```
df['kin']=df['sibsp']+df['parch']
df.drop(['sibsp', 'parch'], axis=1, inplace=True)
df.head()
```

	survived	sex	age	fare	kin
0	0	male	22.0	7.2500	1
1	1	female	38.0	71.2833	1
2	1	female	26.0	7.9250	0
3	1	female	35.0	53.1000	1
4	0	male	35.0	8.0500	0

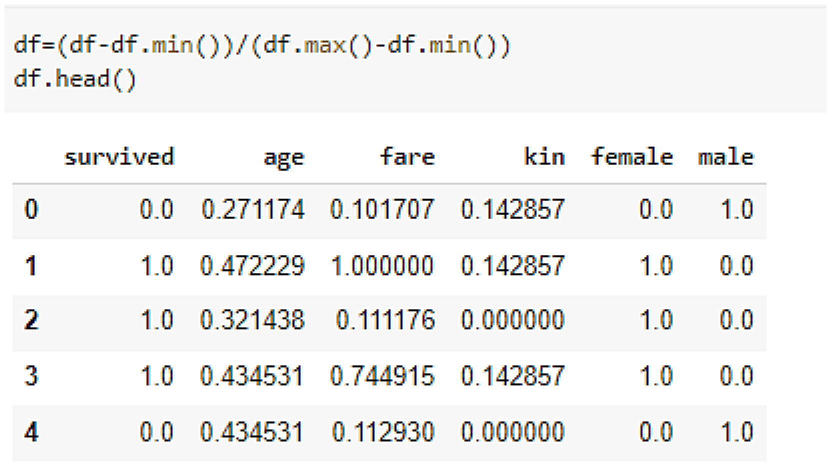
7. Convert the nominal variable gender into a numerical variable using one-hot encoding. (It would perhaps be advisable to use a binary variable instead. However, one-hot encoding is used here for demonstration purposes.)



```
one_hot = pd.get_dummies(df['sex'])
df.drop('sex', axis=1, inplace=True)
df=df.join(one_hot)
df.head()
```

	survived	age	fare	kin	female	male
0	0	22.0	7.2500	1	0	1
1	1	38.0	71.2833	1	1	0
2	1	26.0	7.9250	0	1	0
3	1	35.0	53.1000	1	1	0
4	0	35.0	8.0500	0	0	1

8. Finally, min-max normalization is done as follows,



```
df=(df-df.min())/(df.max()-df.min())
df.head()
```

	survived	age	fare	kin	female	male
0	0.0	0.271174	0.101707	0.142857	0.0	1.0
1	1.0	0.472229	1.000000	0.142857	1.0	0.0
2	1.0	0.321438	0.111176	0.000000	1.0	0.0
3	1.0	0.434531	0.744915	0.142857	1.0	0.0
4	0.0	0.434531	0.112930	0.000000	0.0	1.0

Data Preprocessing in R

1. Read the required columns of the ‘titanic’ dataset using the CSV file. This CSV file has been taken from the publicly available Kaggle dataset: Titanic (https://www.kaggle.com/heptapod/titanic).

```
df <- read.csv('t/train_and_test2.csv')[,c("Age", "Sex", "X2urvived", "sibsp", "Parch", "Fare")]
head(df)
```

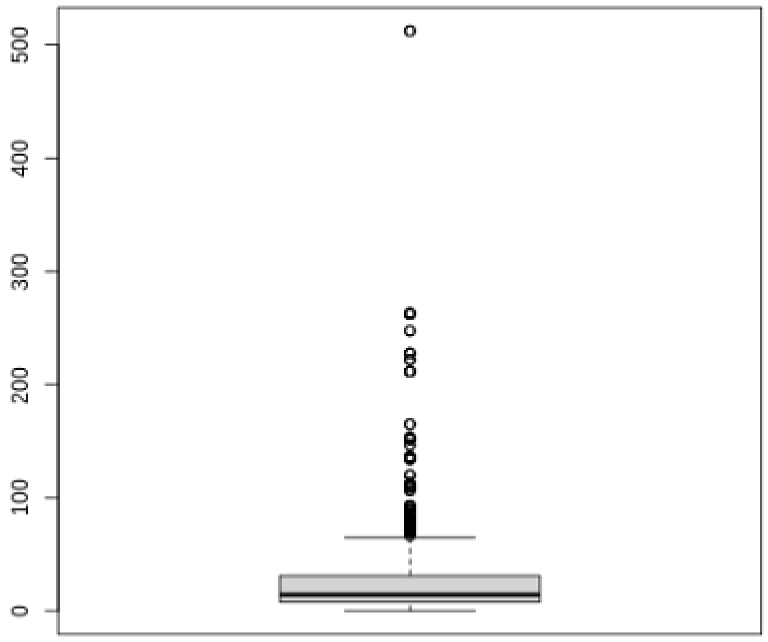
```
  Age Sex X2urvived sibsp Parch    Fare
1  22   0         0     1     0  7.2500
2  38   1         1     1     0 71.2833
3  26   1         1     0     0  7.9250
4  35   1         1     1     0 53.1000
5  35   0         0     0     0  8.0500
6  28   0         0     0     0  8.4583
```

The first six rows can be displayed as shown above using the head() function.

2. We will be removing the outliers in the ‘Fare’ column of the dataset in the subsequent steps, analyze the boxplot for the column and the dimensions of the dataframe.

```
boxplot(df$Fare)
dim(df)
```

```
[1] 1309    6
```



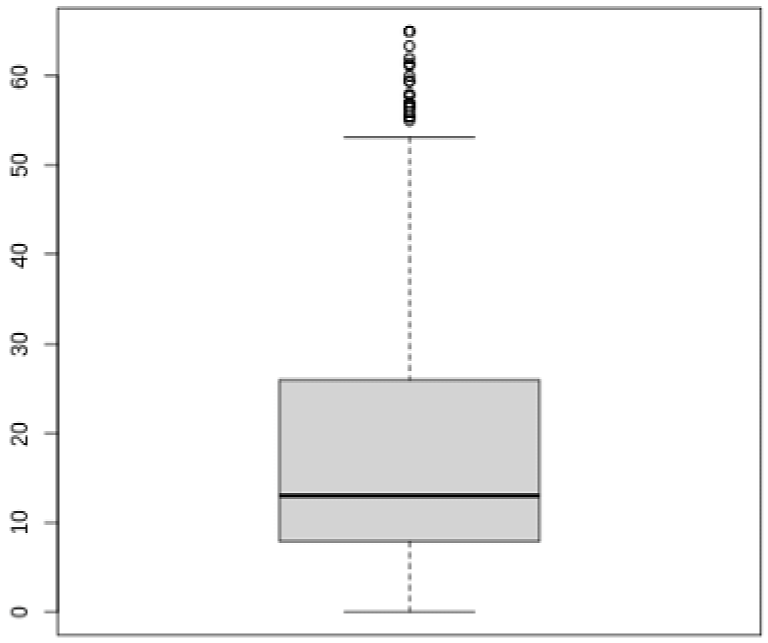
3. Define the limits beyond which the values will be removed, and then eliminate the outliers using subset(). Observe the change dimensions of the modified dataframe.

```
Q1 <- quantile(df$Fare, .25)
Q3 <- quantile(df$Fare, .75)
IQR <- IQR(df$Fare)

df <- subset(df, df$Fare> (Q1 - 1.5*IQR) & df$Fare< (Q3 + 1.5*IQR))

dim(df)
```

```
[1] 1138    6
```

You can observe the resultant boxplot (after the outliers are removed) as shown below.

```
boxplot(no_outliers$Fare)
```



4. Sum the values of the ‘sibsp’ and ‘Parch’ columns to achieve data reduction.

After having created the new column, exclude the two redundant columns using subset().

NOTE: The as .numeric() function was not required here as you can sum the two columns without this additional step. This has, however, been done for demonstration purposes.

```
df$sibsp <- as.numeric(df$sibsp)
df$Parch <- as.numeric(df$Parch)
df$kin <- df$sibsp + df$Parch
df <- subset(df, select = -c(sibsp,Parch))
head(df)
```

```
  Age Sex X2urvived    Fare kin
1  22   0         0  7.2500   1
2  38   1         1 71.2833   1
3  26   1         1  7.9250   0
4  35   1         1 53.1000   1
5  35   0         0  8.0500   0
6  28   0         0  8.4583   0
```

5. The next step would be to normalize the dataset. The normalize() function can be defined as shown below. As the ‘Sex’ and ‘X2urvived’ columns are already in the range 0 to 1, this function needs to be applied only to the remaining columns as shown below.

```
normalize <- function(x) {
return ((x - min(x)) / (max(x) - min(x)))
}

df$Age <- normalize(df$Age)
df$Fare <- normalize(df$Fare)
df$kin <- normalize(df$kin)
head(df)
```

```
        Age Sex X2urvived      Fare       kin
1 0.2734561   0         0 0.1115385 0.1428571
3 0.3235626   1         1 0.1219231 0.0000000
4 0.4363021   1         1 0.8169231 0.1428571
5 0.4363021   0         0 0.1238462 0.0000000
6 0.3486158   0         0 0.1301277 0.0000000
7 0.6743079   0         0 0.7978846 0.0000000
```

As with the data quality dimensions we went over earlier, it is perhaps important to mention, that the basic Python and R data preprocessing implementations demonstrated in this section are by no means the comprehensive set of preprocessing operations that can be performed on a given dataset (Actually far from it!). Even the set of steps or sequence in which they are to be performed during preprocessing aren’t quite set in stone. Further, the data preprocessing steps and techniques used will vary widely depending on the dataset provided and its intended use. Implementing or following along with the above examples, however, should help you get started. And who knows, maybe you’ll even come across some other interesting techniques you want to try out while you are at it.

The process of data preprocessing can sometimes be tedious in real-world scenarios as you will come across various data, both structured and unstructured. And while the easy way out may seem to collect more data and make do with it simply, you will soon find that there is no way to justify skimping on data preprocessing. For quoting from the words of the American computer scientist and director of research at Google, LLC, Peter Norvig, “More data beats clever algorithms, but better data beats more data.” No matter how much data or how many extraordinary models you have at hand, data preprocessing can make it better! So why not get good at it