**Data Preprocessing**

## Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.

## Why do we need Data Preprocessing?

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data pre-processing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

**Data cleaning**

**I. Assess Data quality**

The term “data quality” refers to the suitability of data to serve its intended purpose. So, measuring data quality involves performing data quality assessments to determine the degree to which your data adequately supports the business needs of the company.

A data quality assessment is done by measuring particular features of the data to see if they meet defined standards. Each such feature is called a “data quality dimension” and is rated according to a relevant metric that provides an objective assessment of quality.

## The industry hasn’t yet settled on a standard set of data quality dimensions, but the following is a representative group:

## Four metrics of data quality

Let’s take a brief look at each of these and at the metrics used in assessing them.

### 1. Completeness

Completeness relates to whether all required information is present in the dataset. For example, if the customer information in a database is required to include both first and last names, any record in which the first name or last name field is not populated is marked as incomplete. The metric used in assessing this dimension is the percentage of records that are complete.

### 2. Validity

Data is characterized as valid if it matches the rules specified for it. Those rules typically include specifications such as format (number of digits, etc.), allowable types (integer, floating-point, string, etc.), and range (minimum and maximum values). For example, a telephone number field that contains the string ‘1809 Oak Street’ is not valid. The metric for this dimension is the percentage of records in which all values are valid.

### 3. Timeliness

Timeliness relates to whether the information is up to date for the intended use. In other words, is the correct information available when needed?

For example, if a customer has notified the company of an address change, but the new address is not in the database at the time billing statements are processed, that entry fails the timeliness test. The metric used to measure timeliness is the time difference between when data is needed and when it is available.

### 4. Consistency

A data item is consistent if all representations of that item across data stores match.

If, for example, a birth date is entered in one system using the U.S. format (mm/dd/yyyy), but it is imported into another system where the date is entered using the [European standard](https://en.wikipedia.org/wiki/Date_and_time_notation_in_Europe) (dd/mm/yyyy), that data lacks consistency.

**Data Anomalies**

Generally speaking, an anomaly is something that differs from a norm: a deviation, an exception. In software engineering, by anomaly we understand a rare occurrence or event that doesn’t fit into the pattern, and, therefore, seems suspicious. Some examples are:

* sudden burst or decrease in activity
* error in the text;
* sudden rapid drop or increase in temperature.

Common reasons for outliers are:

* data preprocessing errors
* noise
* fraud
* attacks

**Handling Missing Values:**

Missing values are a fact of life. If you are a data scientist or a data engineer and receives data, then missing values abound. How you should deal with missing values is highly context-dependent:

* Maybe remove all the rows with missing values?
* Maybe drop an entire feature that has too many missing values?
* Maybe fill in the missing values in a clever way?

**Detect missing values with pandas dataframe functions: .info() and .isna() - Program**

**Diagnose type of missing values with visual methods - Missingno program**

**Outliers**

Outlier detection can often be an important part of any exploratory data analysis. This is because in the real world, data is often messy and many different things can affect the underlying data. It is thus important to be able to identify different methods to be able to identify these from the underlying data.

The first thing to ask, then, is “what is an outlier?” An outlier can be classed as a data point, or several data points, that don’t fit the pattern, data structure, or within the normal bounds to what we would expect for the data that we have. This is a very subjective definition as outliers depend heavily on the context in which you are examining as to whether a data point is an outlier or not.

**Reason for outlier occurrence:**

This outlier could be the result of many different issues:

* Human error
* Instrument error
* Experimental error
* Intentional creation
* Data processing error
* Sampling error
* Natural outlier

**Purpose of detecting outliers:**

The purpose for being able to identify this outlier of course can also be different. This could be because an outlier would indicate something has changed in the action that produces the data which is useful in the case of:

* Fraud detection
* Intrusion detection
* Fault diagnostics
* Time series monitoring
* Health monitoring

Where an outlier would indicate maybe something has gone wrong in the process or the nature of the process generating the data has changed. This would thus entail identifying outliers on the basis of underlying accepted normal data.

**Outlier detection and removal programs**

**Data Integration**

Whenever the data we need for our analytic goals are from different sources, before we can perform the data analytics, we need to integrate the data sources into one dataset that we need for our analytic goals. There are many challenges that you need to overcome before integration is possible. These challenges could be due to organizational privacy and security challenges that restrict our data accessibility. But even assuming that these challenges are not in the way when different data sources need to be integrated, they arise because each data source is collected and structured based on the needs, standards, technology, and opinions of the people who have collected them. Regardless of correctness, there are always differences in the ways that the data is structured and because of that, data integration becomes challenging.

**Data integration Approaches**

Data integration may happen in two different directions.

* The first is by **adding attributes**; we might want to supplement a dataset with more describing attributes. In this direction, we have all the data objects that we need, but other sources might be able to enrich our dataset.
* The second is by **adding data objects**; we might have multiple sources of data with distinct data objects, and integrating them will lead to a population with more data objects that represent the population we want to analyze.

**Challenges in Data Integration**

**Challenge 1 – entity identification**

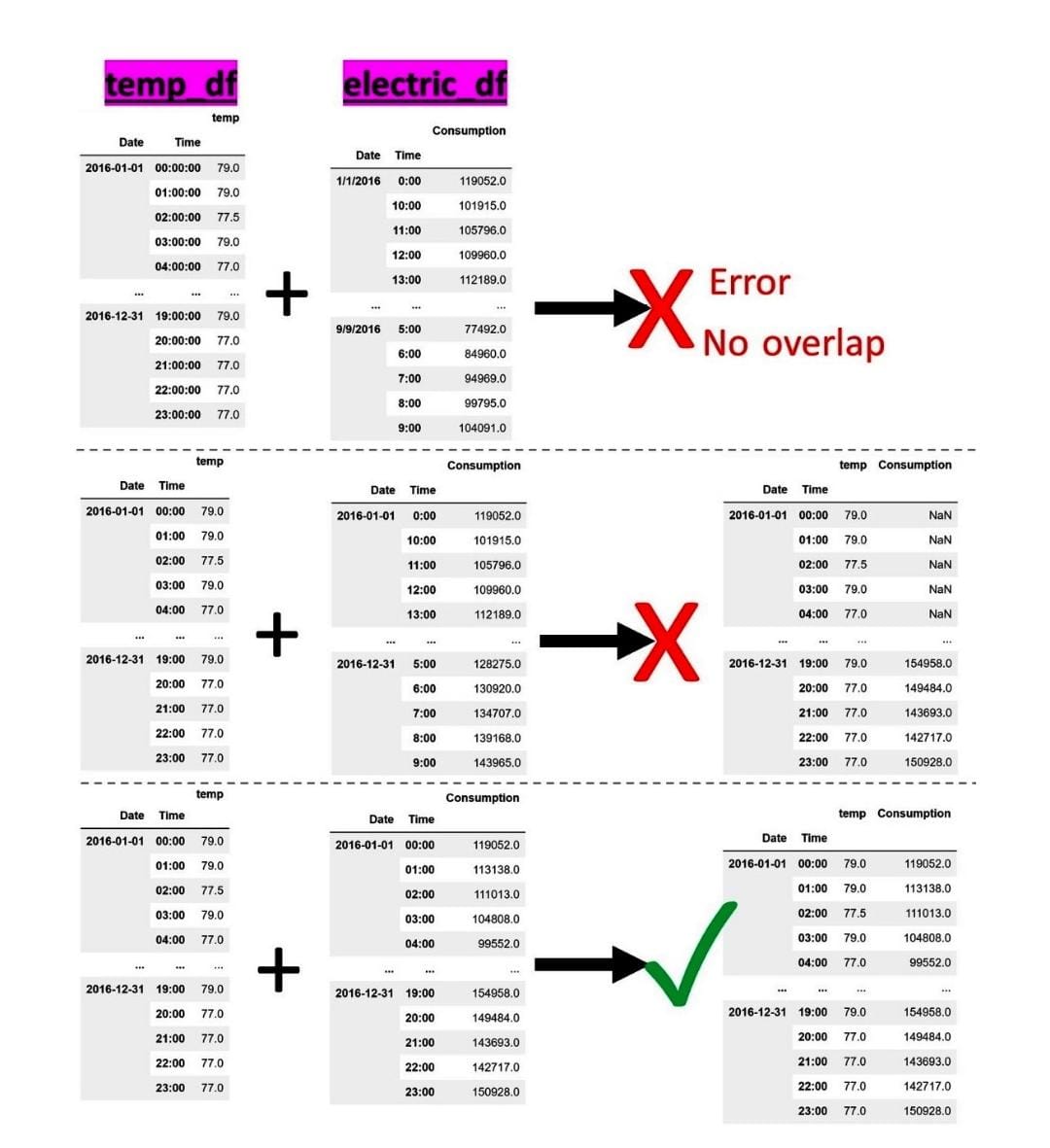
* The entity identification challenge may occur when the data sources are being integrated by adding attributes.
* The challenge is that the data objects in all the data sources are the same real-world entities with the same definitions of data objects, but they are not easy to connect due to the unique identifiers in the data sources.
* For instance, in the data integration example section, the sales department and the marketing department did not use a central customer unique identifier for all their customers.
* Due to this lack of data management, when they want to integrate the data, they will have to figure out which customer is which in the data sources.

**Challenge 2 – unwise data collection**

* This data integration challenge happens, as its name suggests, due to unwise data collection.
* For instance, instead of using a centralized database, the data of different data objects is stored in multiple files.
* Regardless, in these situations, our goal is to make sure that the data is integrated into one standard data structure. This type of data integration challenge happens when data objects are being added.

**Challenge 3 – index mismatched formatting**

* When we start integrating data sources by adding attributes, we will use the pandas DataFrame .join() function to connect the rows of two DataFrames that have the same indices.
* To use this valuable function, the integrating DataFrames needs to have the same index formatting; otherwise, the function will not connect the rows.
* For example, the following figure shows three attempts of combining two DataFrames: temp\_df and electric\_df.

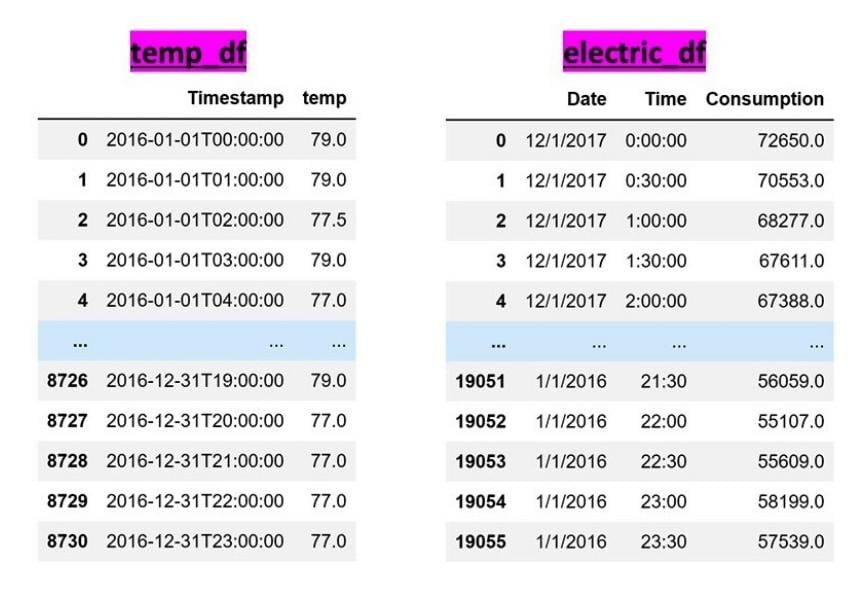


* temp\_df contains the hourly temperature (temp) of 2016, while electric\_df carries the hourly electricity consumption (consumption) for the same year.
* The first two attempts (the top one and the middle one) are unsuccessful due to the index mismatched formatting challenge.
* For instance, consider the attempt at the top; while both DataFrames are indexed with Date and Time and both show the same Date and Time, attempting the .join() function will produce a "cannot join with no overlapping index names" error.
* The attempt to integrate was unsuccessful because the index formatting from the two DataFrames is not the same:

In the preceding diagram, while the attempt in the middle is better than the one at the top, it is still unsuccessful. Pay close attention and see if you can figure out why there are so many NaNs in the output of the integration.

**Challenge 4 – aggregation mismatch**

* This challenge occurs when integrating data sources by adding attributes.
* When integrating time series data sources whose time intervals are not identical, this challenge arises.
* For example, if the two DataFrames presented in the following figure are to be integrated, not only do we have to address the challenge of index mismatch formatting, but we will also need to face the aggregation mismatch challenge.
* This is because temp\_df carries the hourly temperature data but electric\_df carries the electricity consumption of every half an hour.
* To deal with this challenge, we will have to restructure one source or both sources to get them to have the same level of data aggregation.



**Challenge 5 – duplicate data objects**

* This challenge occurs when we're integrating data sources by adding data objects.
* When the sources contain data objects that are also in the other sources, when the data sources are integrated, there will be duplicates of the same data objects in the integrated dataset.
* For example, imagine a hospital that provides different kinds of healthcare services. For a project, we need to gather the socioeconomic data of all of the patients in the hospital.
* The imaginary hospital does not have a centralized database, so all of the departments are tasked with returning a dataset containing all the patients they have provided services for.
* After integrating all of the datasets from different departments, you should expect that there are multiple rows for the patients that had to receive care from different departments in the hospital.

**Challenge 6 – data redundancy**

* This challenge's name seems to be appropriate for the previous challenge as well, but in the literature, the term data redundancy is used for a unique situation.
* Unlike the previous challenge, this challenge may be faced when you're integrating data sources by adding attributes.
* As the name suggests, after data integration, some of the attributes may be redundant.
* This redundancy could be shallow as there are two attributes with different titles but the same data.
* Or, it could be deeper. In deeper data redundancy cases, the redundant attribute does not have the same title, nor is its data the same as one of the other attributes, but the values of the redundant attribute can be derived from the other attributes.
* For example, after integrating data sources into a dataset of customers, we have the following seven attributes: age, average order $, days from the last visit, weekly visit frequency, weekly $ purchase, and satisfaction score. If we use all seven attributes to cluster customers, we have made a mistake regarding data redundancy. Here, the weekly visit frequency, weekly $ purchase, and average order $ attributes are distinct but the value of weekly $ purchase can be derived from weekly visit frequency and average order $. By doing so, inadvertently, we will have given the information regarding the customer's visit and their purchase amount more weight in the clustering analysis.

**Data Redundancy**

Another important step of data preprocessing that is not concerned with data cleaning; this is known as data reduction. To successfully perform analytics, we need to be able to recognize situations where data reduction is necessary and know the best techniques and the how-to of their implementation.

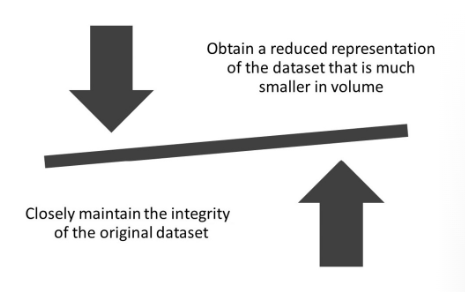
**The distinction between data reduction and data redundancy**

* While data redundancy and data reduction have very similar names and their terms use words that have connected meanings, the concepts are very different.
* Data redundancy is about having the same information presented under more than one attribute. As we saw, this can happen when we integrate data sources.
* However, data reduction is about reducing the size of data due to one of the following three reasons:
  + **High-Dimensional Visualizations**: When we have to pack more than three to five dimensions into one visual, we will reach the human limitation of comprehension.
  + **Computational Cost:** Datasets that are too large may require too much computation. This might be the case for algorithmic approaches.
  + **Curse of Dimensionality:** Some of the statistical approaches become incapable of finding meaningful patterns in the data because there are too many attributes
* In other words, data redundancy is a characteristic that a dataset may have. This characteristic is about having redundant data in the dataset, so we may have to take some actions.
* On the other hand, data reduction is a set of actions that we can take to reduce the size of data due to the aforementioned reasons.

**The objectives of data reduction**

Successful data reduction seeks to achieve the following two objectives at the same time.

* First, data reduction seeks to obtain a reduced representation of the dataset that is much smaller in volume.
* Second, it tries to closely maintain the integrity of the original data, which means making sure that data reduction will not lead to including bias and critical information being lost in the data.



* As shown in the following diagram, these two objectives can be contradictory and when performing data reduction actions, the two objectives must be taken into consideration at the same time so that one is not overshadowed by the other:

**Types of data reduction**

There are two types of data reduction methods.

* **Numerosity data reduction:** performs data reduction by reducing the number of data objects or rows in a dataset
* **Dimensionality data reduction:** performs data reduction by reducing the number of dimensions or attributes in a dataset

**Numerosity reduction methods**

* **Random Sampling:** Randomly selecting some of the data objects to avoid unaffordable computational costs.
* **Stratified Sampling:** Randomly selecting some of the data objects to avoid the unaffordable computational costs, all the while maintaining the ratio representation of the sub-populations in the sample.
* **Random Over/Under Sampling:** Randomly selecting some of the data objects to avoid the unaffordable computational costs, all the while creating a prescribed representation of the sub-populations in the sample.

**Dimensionality reduction methods**

* **Linear Regression:** Using regression analysis to investigate the predictive power of independent attributes to predict a specific dependent attribute.
* **Decision Tree:** Using the decision tree algorithm to investigate the predictive power of the independent attributes to predict a specific dependent attribute.
* **Random Forest:** Using the random forest algorithm to investigate the predictive power of the independent attributes to predict a specific dependent attribute.
* **Brute-force Computational Dimension Reduction:** Computational experimentations to figure out the best subset of independent attributes that leads to the most successful prediction of the dependent attribute
* **Principal Component Analysis (PCA):** Representing the data by transforming the axes in such ways that most of the variation in the data is explained by the first attributes and the attributes are orthogonal to one another.
* **Functional Data Analysis (FDA):** Representing the data using fewer points using functional representation.

**Data transformation**

Data transformation normally is the last data preprocessing that is applied to our datasets. The dataset may need to be transformed to be ready for a prescribed analysis, or a specific transformation might help a certain analytics tool to perform better, or simply without a correct data transformation, the results of our analysis might be misleading.

**Need for data transformation**

The following are the reasons for performing data transformation:

* **Necessity**: The analytic method cannot work with the current state of the data. For instance, many data-mining algorithms, such as Multi-Layered Perceptron (MLP) and K-means, only work with numbers; when there are categorical attributes, those attributes need to be transformed before the analysis is possible.
* **Correctness**: Without the proper data transformation, the resulting analytic will be misleading and wrong. For instance, if we use K-means clustering without normalizing the data, we think that all the attributes have equal weights in the clustering result, but that's incorrect; the attributes that happen to have a larger scale will have more weight.
* **Effectiveness**: If the data goes through some prescribed changes, the analytics will be more effective.

**Normalization and standardization**

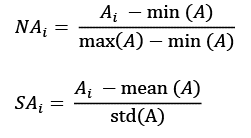
* We need **normalization** when we need the range of all the attributes in a dataset to be equal. This will be needed especially for algorithmic data analytics that uses the distance between the data objects. Examples of such algorithms are K-means and KNN.
* On the other hand, we need **standardization** when we need the variance and/or the standard deviation of all the attributes to be equal.

The following two equations show the formula we need to use to apply normalization and standardization.

The following list defines the variables used in the equations:

A: The attribute i: The index for the data objects Ai: The value of data object i in attribute A

NA: The normalized version of attribute A SA: The standardized version of attribute A



**Difference between Normalization and Standardization**

| **Normalization** | **Standardization** |
| --- | --- |
| Minimum and maximum value of features are used for scaling | Mean and standard deviation is used for scaling. |
| It is used when features are of different scales. | It is used when we want to ensure zero mean and unit standard deviation. |
| Scales values between [0, 1] or [-1, 1]. | It is not bounded to a certain range. |
| It is really affected by outliers. | It is much less affected by outliers. |
| Scikit-Learn provides a transformer called MinMaxScaler for Normalization. | Scikit-Learn provides a transformer called StandardScaler for standardization. |
| This transformation squishes the n-dimensional data into an n-dimensional unit hypercube. | It translates the data to the mean vector of original data to the origin and squishes or expands. |
| It is useful when we don’t know about the distribution | It is useful when the feature distribution is Normal or Gaussian. |
| It is a often called as Scaling Normalization | It is a often called as Z-Score Normalization. |

**Data transformation with Binary coding, ranking transformation, and discretization**

In our analytics journey, there will be many instances in which we want to transform our data from numerical representation to categorical representation, or vice versa. To do these transformations, we will have to use one of three tools: binary coding, ranking transformation, and discretization.

**To switch from Categories to Numbers:**

We generally transform categorical attributes to numerical ones when our analytics tool of choice can only work with numbers. For instance, if we would like to use MLP for prediction and some of the independent attributes are categorical, MLP will not be able to handle the prediction task unless the categorical attributes are transformed into numerical attributes.

Methods of transforming categories to numbers:

* Binary Coding
* Ranking Transformation

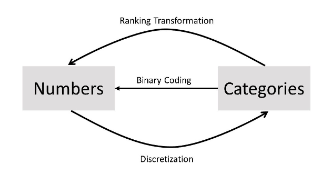
If the categories are nominal, we can only use binary coding; if they are ordinal, both may be used.

**To switch from numbers to categories:**

Most often, transforming numerical attributes into categorical ones is done because the resulting analytics output will become more intuitive for our consumption. For instance, instead of having to deal with a number that shows the GPA, we may be more comfortable dealing with categories such as excellent, good, acceptable, and unacceptable. This will become the case, especially if we want to use our attention to understand the interactions between attributes.

Methods of transforming numbers to categories:

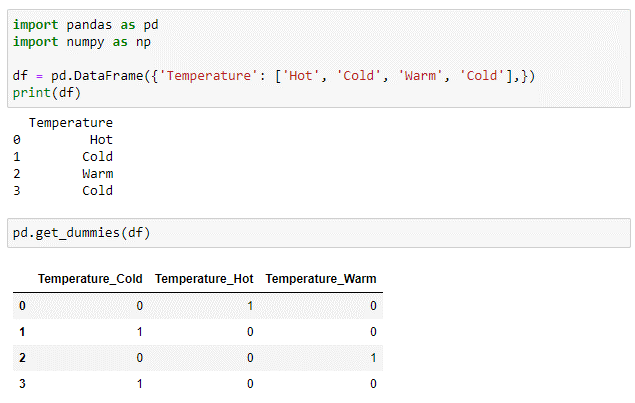
* Discretization



**Direction of application for binary coding, ranking transformation, and discretization**

**Binary coding of nominal attribute**

* When the attribute continent is nominal, we only have one choice and that is to use binary coding.
* pd.get\_dummies() pandas function to binary-code the nominal categorical attribute.

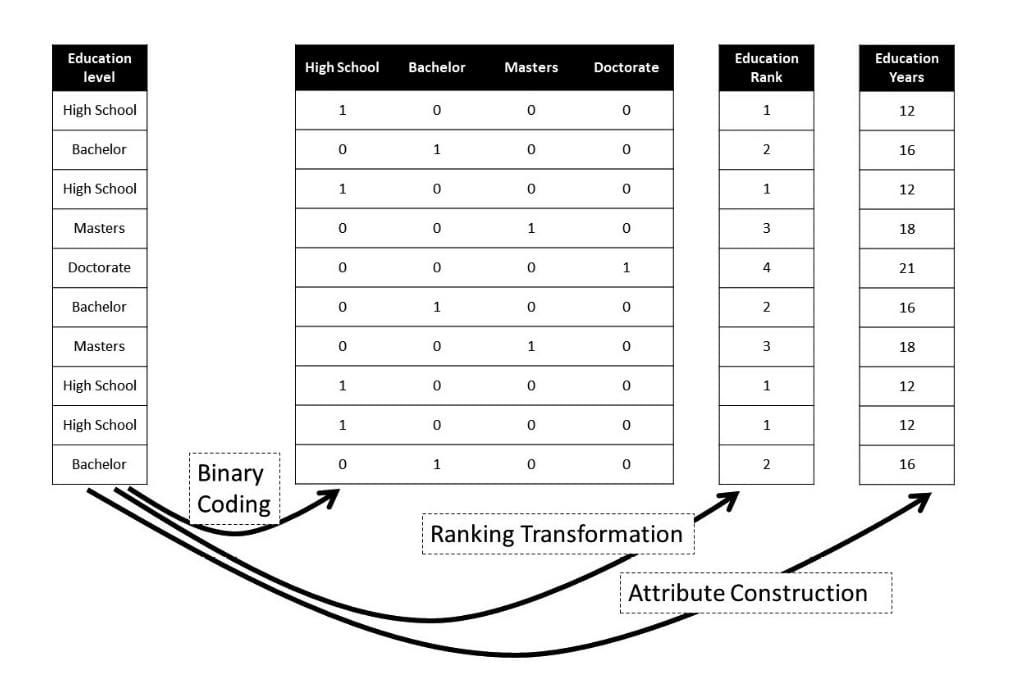


**Binary coding or ranking transformation of ordinal attributes**

* Transforming ordinal attributes into numbers is a bit tricky.
* There is no perfect solution; we either have to let go of the ordinal information in the attribute, or assume some information into the data.

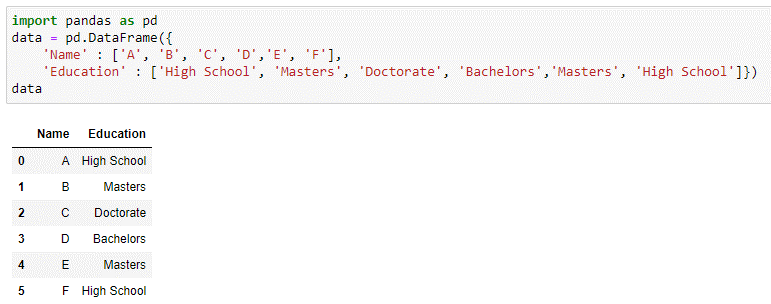
**Let's see what that means in an example**

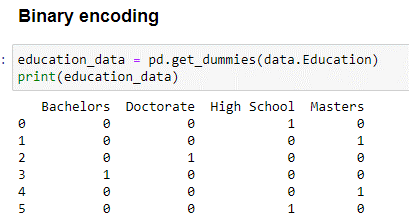
* The following figure shows the transformation of an example ordinal attribute into numbers by three methods: Binary Coding, Ranking Transformation, and Attribute Construction.
* In the case of **Binary Coding**, the transformation has not assumed any information into the result, but the transformation has stripped the attribute from its ordinal information.
* If we were to use the binary-coded values instead of the original attribute in our analysis, the data does not show the order of the possible values of the attribute.
* For example, while the binary-coded values make a distinction between High School and Bachelor, the data does not show that Bachelor comes after High School, as we know it does.
* The next transformation, **Ranking Transformation**, does not have this shortcoming; however, it has other cons. By trying to make sure that the order of the possible values is maintained, we had to engage numbers by ranking transformation.
* By engaging numbers, not only have we successfully included order in between the possible values of the attribute but we have also collaterally assumed information that does not exist in the original attribute.
* For example, with the ranking transformed attribute, we are assuming there is one unit difference between Bachelors and High School.

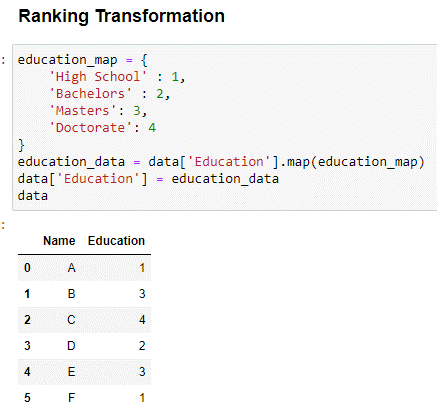


* The figure has another transformation, **Attribute Construction**, which is only possible if we have a good understanding of the attribute. What Attribute Construction tries to fix is the gross assumptions that are added by Ranking Transformation; instead,
* Attribute Construction uses the knowledge about the original attribute to assume more accurate information into the transformed data.
* Here, for example, as we know, achieving any of the degrees in the Education Level attribute takes a different number of years of education.
* So, instead, Attribute Construction uses that knowledge to assume more accurate assumptions into the transformed data.

**Ordinal attribute Transformation Example**







**Discretization of numerical attributes**

Discretizing is transforming numeric attributes to nominal. You might want to do that in order to use a classification method that can’t handle numeric attributes (unlikely), or to produce better results (likely), or to produce a more comprehensible model such as a simpler decision tree (very likely).

