Data Pre-processing

Prof. (Dr.) Honey Sharma

Reference Books:

- Mitchell M., T., Machine Learning, McGraw Hill (1997) 1st Edition.
- https://www.geeksforgeeks.org/

Need of Data Pre-processing

Data has quality if it satisfies the requirements of its intended use.

Following are the factors comprising data quality:

- **Accuracy**
- **Completeness**
- **Consistency**
- **Timeliness**
- **Believability**
- **Interpretability**

Lets you have to analyze the some company’s data with respect to the sales at one branch. You will carefully inspect the company’s database identifying and selecting the attributes or dimensions to be included in your analysis, such as item name, price, and units sold.

Alas! You notice that several of the attributes for various tuples have no recorded value.

For your analysis, you would like to include information about the items sold during sale, yet you discover that this information has not been recorded.

Furthermore, users of your database system have reported errors, unusual values, and inconsistencies in the data recorded for some transactions.

In other words, the data you wish to analyze are **incomplete** (**lacking attribute values or certain attributes of interest were missing**), **inaccurate or noisy (containing errors, or values that deviate from the expected), and inconsistent (e.g., containing discrepancies in the department codes used to categorize items)**. Welcome to the real world

Data Pre-Processing is required

This scenario illustrates three of the elements defining data quality - **accuracy, completeness, and consistency.**

There are many possible reasons for inaccurate data :

- The data collection instruments used may be faulty.
- There may have been human errors occurring at data entry.
- Users may purposely submit incorrect data values for mandatory fields when they do not wish to submit personal information.
- Errors in data transmission can also occur.

Recall that data quality depends on the intended use of the data. Two different users may have very different assessments of the quality of a given database.

For example, a marketing analyst may need to access the database mentioned above for a list of customer addresses. Some of the addresses are outdated or incorrect, yet overall, 80% of the addresses are accurate and still is pleased with the accuracy of the database, although, as sales manager, you found the data inaccurate

Timeliness also affects data quality.

Suppose that you are overseeing the distribution of monthly sales bonuses to the top sales representatives. Several sales representatives, however, fail to submit their sales records on time at the end of the month. For a period of time following each month, the data stored in the database is incomplete. However, once all of the data is received, it is correct.

The fact that the month-end data is not updated in a timely fashion has a negative impact on the data quality.

Believability reflects how much the data are trusted by users, while **interpretability** reflects how easy the data are understood.

Suppose that a database, at one point, had several errors, all of which have since been corrected. The past errors, however, had caused many problems for users in the sales department, and so they no longer trust the data.

The data also use many accounting codes, which the sales department does not know how to interpret. Even though such a database is now accurate, complete, consistent, and timely, users from the sales department may regard it as of low quality due to poor **believability and interpretability**

https://docs.google.com/spreadsheets/d/1z0FlBpnu1VHayTi1XSpZw2pwWVkaIpngrwDt4i0-t-M/edit#gid=0

Data Pre-processing Methods: Data Cleaning

Real-world data tend to be incomplete, noisy, and inconsistent.

Data cleaning (or data cleansing) routines attempt to

- fill in missing values
- smooth out noise while identifying outliers
- correct inconsistencies in the data.

How to Handle Missing Data?

It is common to identify missing values in a dataset and replace them with a numeric value. T**his is called data imputing, or missing data imputation.**

Ignore the data row:

- This method is suggested for records where maximum amount of data is missing, rendering the record meaningless.
- This method is usually avoided where only less attribute values are missing.
- If all the rows with missing values are ignored i.e. removed, it will result in poor performance.

Fill the missing values manually:

This is a very time consuming method and hence infeasible for given a large data set with many missing values

Use a global constant to fill in for missing values:

A global constant like “NA” or 0 can be used to fill all the missing data. This method is used when missing values are difficult to be predicted. In this case the algorithm may mistakenly think that these constant are an interesting concept, since they all have a value in common.
Hence, although this method is simple, it is not foolproof.

Use a measure of central tendency for the attribute to fill in the missing value:

Mean or median of the attribute is used to fill the missing value. For symmetric data distributions, the mean can be used, while skewed data distribution should employ the median.

Use forward fill or backward fill method:

In this, either the previous value or the next value is used to fill the missing value. A mean of the previous and succession values may also be used.

Use a data-mining algorithm to predict the most probable value:

kNN Imputation for Missing Values

kNN Imputation for Missing Values

The k-nearest neighbor (KNN) algorithm (often referred to as “nearest neighbor imputation) has proven to be generally effective.

KNNimpute appears to provide a more robust and sensitive method for missing value estimation […] and KNNimpute surpass the commonly used row average method (as well as filling missing values with zeros).

The default distance measure is a Euclidean distance measure that is NaN aware.

The number of neighbors is set to five by default and can be configured by the “n_neighbors” argument.

Finally, the distance measure can be weighed proportional to the distance between instances (rows), although this is set to a uniform weighting by default, controlled via the “weights” argument.

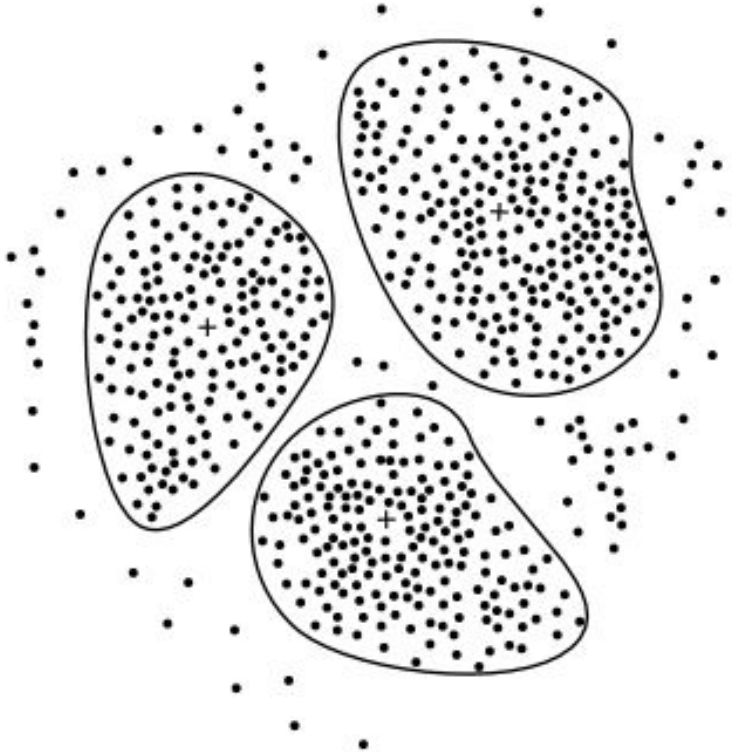
https://docs.google.com/spreadsheets/d/1EU8yfTuVPyvLv-1zgQE0uATYCTtx-adNfe_4q1hOPyI/edit#gid=0

Handle Noise and Outliers

Noise can be handled using **binning**. In this technique, sorted data is placed into bins or buckets. Bins can be created by equal-width (distance) or equal-depth (frequency) partitioning. On these bins, smoothing can be applied. Smoothing can be by bin mean, bin median or bin boundaries.

Regression: Here data can be made smooth by fitting it to a regression function.The regression used may be linear (having one independent variable) or multiple (having multiple independent variables).

Clustering: This approach groups the similar data in a cluster. The outliers may be undetected or it will fall outside the clusters.



Data Integration

We often requires data integration—the merging of data from multiple data stores. Careful integration can help reduce and avoid redundancies and inconsistencies in the resulting data set.

This can help improve the accuracy and speed of the subsequent process.

The Entity Identification Problem

There are a number of issues to consider during data integration.

Schema integration and object matching can be tricky.

How can equivalent real-world entities from multiple data sources be matched up?

This is referred to as the entity identification problem. For example, how can the data analyst or the computer be sure that customer id in one database and cust number in another refer to the same attribute?

When matching attributes from one database to another during integration, special attention must be paid to the structure of the data.

This is to ensure that any attribute functional dependencies and referential constraints in the source system match those in the target system.

For example, in one system, a discount may be applied to the order, whereas in another system it is applied to each individual line item within the order. If this is not caught before integration, items in the target system may be improperly discounted.

Redundancy and Correlation Analysis

Redundancy is another important issue in data integration. An attribute (such as annual revenue, for instance) may be redundant if it can be “derived” from another attribute or set of attributes.

Some redundancies can be detected by correlation analysis. Given two attributes, such analysis can measure how strongly one attribute implies the other, based on the available data

Data Transformation

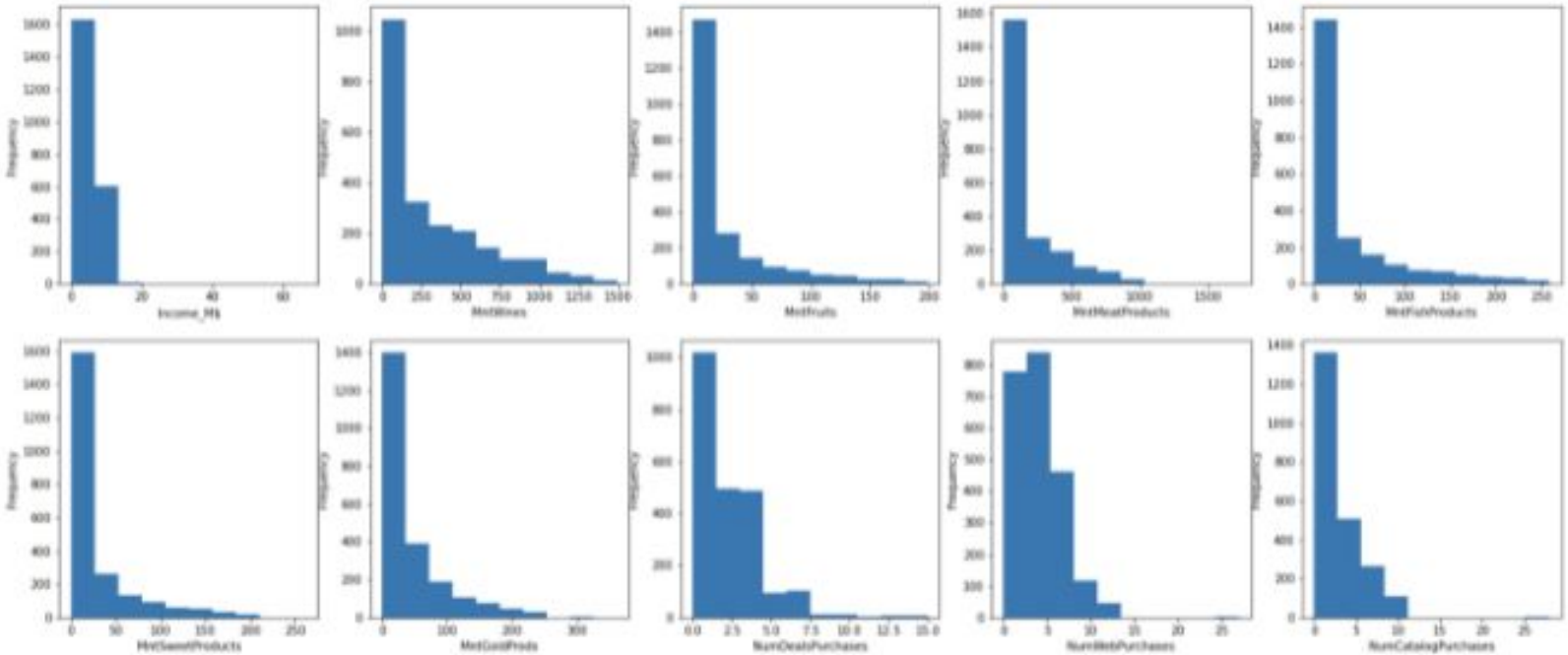
Data transformation is the process of converting raw data into a format or structure that would be more suitable for model building and also data discovery in general. It is an imperative step in feature engineering that facilitates discovering insights.

Why need data transformation?

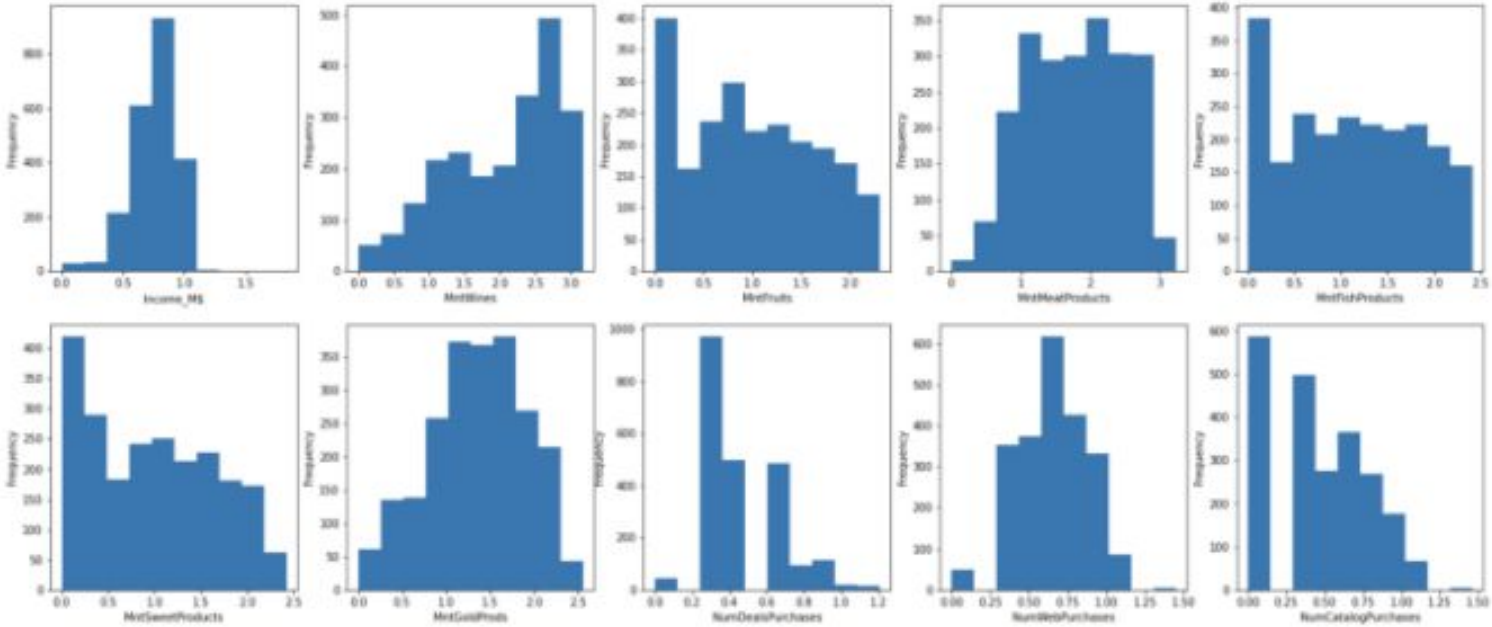
- the algorithm is more likely to be biased when the data distribution is skewed
- transforming data into the same scale allows the algorithm to compare the relative relationship between data points better

Log Transformation — right skewed data

When the data sample follows the power law distribution, we can use log scaling to transform the right skewed distribution into normal distribution. To achieve this, simply use the np.log() function.



before transformation (image by author)



after transformation (image by author)

Clipping — handle outliers

This approach is more suitable when there are outliers in the dataset. Clipping method sets up the upper and lower bound, and all data points will be contained within the range.

We can use quantile() to find out what is the range of the majority amount of data (between 0.05 percentile and 0.95 percentile). **Any numbers below the lower bound (defined by 0.05 percentile) will be rounded up to the lower bound. Similarly, the numbers above upper bound (defined by 0.95 percentile) will be rounded down to upper bound.**

Scaling Transformation

After log transformation and addressing the outliers, we can use the scikit-learn preprocessing library to convert the data into the same scale. This library contains some useful functions: min-max scaler, standard scaler and robust scaler. Each scaler serves different purpose.

Min Max Scaler — normalization

MinMaxScaler() is usually applied when the dataset is not distorted. It normalizes the data into a range between 0 and 1 based on the formula:
x’ = (x — min(x)) / (max(x) — min(x))

Scaling Transformation

Standard Scaler — standardization

We use standardization when the dataset conforms to normal distribution. StandardScaler() converts the numbers into the standard form of mean = 0 and variance = 1 based on z-score formula:

x’ = (x — mean) / standard deviation.

Robust Scaler

RobustScaler() is more suitable for dataset with skewed distributions and outliers because it transforms the data based on median and quantile:

x = (x — median) / inter-quartile range.

The scalers don’t change the shape of the data distribution but instead changing the spread of data point.

MinMaxScaler() converts the values to be strictly between 0 and 1, StandardScaler() transforms dataset into mean = 0 whereas RobustScaler() transforms dataset into median = 0

Data Set for exercise
https://www.kaggle.com/jackdaoud/marketing-data

Data Reduction

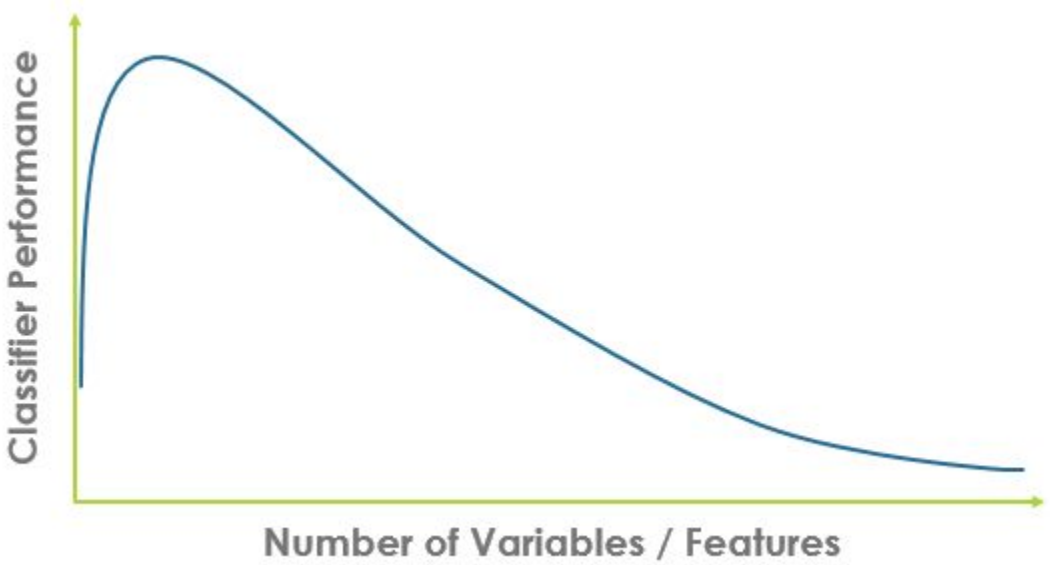
An universal problem of intelligent (learning) agents is where to focus their attention.

It is very critical to understand “What are the aspects of the problem at hand are important/necessary to solve it?” i.e. discriminate between the relevant and irrelevant parts of experience.

Data **reduction** techniques can be applied to obtain a reduced representation of the data set that is much smaller in volume, yet closely maintains the integrity of the original data.

Dimensionality reduction can be discussed through a simple e-mail classification problem, where we need to classify whether the e-mail is spam or not. This can involve a large number of features, such as whether or not the e-mail has a generic title, the content of the e-mail, whether the e-mail uses a template, etc. However, some of these features may overlap.

Dimensionality reduction is the process of reducing the number of random variables under consideration, by obtaining a set of principal variables. It can be divided into feature selection and feature extraction.



In practice: the number of training examples is fixed. the classifier's performance usually will degrade for a large number of features

Feature selection

What is Feature selection (or Variable Selection)?

Problem of selecting some subset of a learning algorithm's input variables upon which it should focus attention, while ignoring the rest. In other words, Dimensionality Reduction.

Mathematically speaking,

- Given a set of features F = { f1 ,…, fi ,…, fn } the Feature Selection problem is to find a subset that “maximizes the learner’s ability to classify patterns”
- Formally F’ should maximize some scoring function
- **This general definition subsumes feature selection (i.e. a feature selection algorithm also performs a mapping but can only map to subsets of the input variables)**

Why even think about Feature Selection in Machine Learning?

- The information about the target class is inherent in the variables
- Naive theoretical view:
  More features means:
  More information
  More discrimination power.
- In practice: many reasons why this is not the case!
- Also: Optimization is (usually) good, so why not try to optimize the input-coding?

Feature Selection in Machine Learning? YES!

- Many explored domains have hundreds to tens of thousands of variables/features with many irrelevant and redundant ones!
- Irrelevant and redundant features can confuse "learners"
- Limited training data
- Limited computational resources
- Curse of Dimensionality

Feature Selection (Summary)

- Feature selection can significantly increase the performance of a learning algorithm (both accuracy and computation time) — but it is not easy!
- One can work on problems with very high- dimensional feature-spaces

Feature Extraction

Feature Extraction aims to reduce the number of features in a dataset by **creating new features from the existing ones** (and then discarding the original features).

These new reduced set of features should then be able to summarize most of the information contained in the original set of features.

In this way, a summarised version of the original features can be created from a combination of the original set.

The difference between Feature Selection and Feature Extraction is that feature selection aims instead to rank the importance of the existing features in the dataset and discard less important ones (no new features are created).

Benefits of Data Reduction

- A lower number of dimensions in data means less training time and less computational resources and increases the overall performance of machine learning algorithms
- Dimensionality reduction avoids the problem of overfitting.
- Dimensionality reduction is extremely useful for data visualization.
- Dimensionality reduction removes noise in the data — By keeping only the most important features and removing the redundant features, dimensionality reduction removes noise in the data.
- Dimensionality reduction can be used to transform non-linear data into a linearly-separable form

- Dimensionality reduction takes care of multicollinearity — In regression, multicollinearity occurs when an independent variable is highly correlated with one or more of the other independent variables. Dimensionality reduction takes advantage of this and combines those highly correlated variables into a set of uncorrelated variables.

Splitting dataset into Training and Testing set

A very common issue when training a model is overfitting. This phenomenon occurs when a model performs really well on the data that we used to train it but it fails to generalise well to new, unseen data points.

There are numerous reasons why this can happen —

- it could be due to the noise in data or it could be that the model learned to predict specific inputs rather than the predictive parameters that could help it make correct predictions.
- Typically, the higher the complexity of a model the higher the chance that it will be overfitted.

On the other hand, underfitting occurs when the model has poor performance even on the data that was used to train it.

In most cases, underfitting occurs because the model is not suitable for the problem you are trying to solve.

Usually, this means that the model is less complex than required in order to learn those parameters that can be proven to be predictive..

Creating different data samples for training and testing the model is the most common approach that can be used to identify these sort of issues.

In this way, we can use the training set for training our model and then treat the testing set as a collection of data points that will help us evaluate whether the model can generalise well to new, unseen data.

The simplest way to split the modelling dataset into training and testing sets is to assign 2/3 data points to the Training Data and the remaining one-third to the Testing Data.

Therefore, we train the model using the training set and then apply the model to the test set. In this way, we can evaluate the performance of our model.

For instance, if the training accuracy is extremely high while the testing accuracy is poor then this is a good indicator that the model is probably overfitted.

For instance, if both the training and testing sets contain patterns that do not exist in real world data then the model would still have poor performance even though we wouldn’t be able to observe it from the performance evaluation.

On a second note, you should be aware that there are certain situations you should consider creating an extra set called t**he validation set**.

The validation set is usually required when apart from model performance we also need to choose among many models and evaluate which model performs better.