Learning Outcome

## Data Analytics using Python (30 Hrs)

## 

# Data mining, wrangling, data manipulation techniques

## Data mining

Data mining is the process of sorting through large data sets to identify patterns and relationships that can help solve business problems through data analysis. Data mining techniques and tools enable enterprises to predict future trends and make more-informed business decisions.

Data mining is a key part of data analytics overall and one of the core disciplines in data science, which uses advanced analytics techniques to find useful information in data sets. At a more granular level, data mining is a step in the knowledge discovery in databases (KDD) process, a data science methodology for gathering, processing and analyzing data. Data mining and KDD are sometimes referred to interchangeably, but they're more commonly seen as distinct things.

### Why is data mining important?

Data mining is a crucial component of successful analytics initiatives in organizations. The information it generates can be used in business intelligence (BI) and advanced analytics applications that involve analysis of historical data, as well as real-time analytics applications that examine streaming data as it's created or collected.

Effective data mining aids in various aspects of planning business strategies and managing operations. That includes customer-facing functions such as marketing, advertising, sales and customer support, plus manufacturing, supply chain management, finance and HR. Data mining supports fraud detection, risk management, cybersecurity planning and many other critical business use cases. It also plays an important role in healthcare, government, scientific research, mathematics, sports and more.

### Data Mining Steps

When asking “what is data mining,” let’s break it down into the steps data scientists and analysts take when tackling a data mining project.

1. **Understand Business**

What is the company’s current situation, the project’s objectives, and what defines success?

1. **Understand the Data**

Figure out what kind of data is needed to solve the issue, and then collect it from the proper sources.

1. **Prepare the Data**

Resolve data quality problems like duplicate, missing, or corrupted data, then prepare the data in a format suitable to resolve the business problem.

1. **Model the Data**

Employ algorithms to ascertain data patterns. Data scientists create, test, and evaluate the model.

1. **Evaluate the Data**

Decide whether and how effective the results delivered by a particular model will help meet the business goal or remedy the problem. Sometimes there’s an iterative phase for finding the best algorithm, especially if the data scientists don’t get it quite right the first time. There may be some data mining algorithms shopping around.

1. **Deploy the Solution**

Give the results of the project to the people in charge of making decisions.

### Benefits of Data Mining

Data mining provides us with the means of resolving problems and issues in this challenging information age.

Data mining benefits include:

* It helps companies gather reliable information
* It’s an efficient, cost-effective solution compared to other data applications
* It helps businesses make profitable production and operational adjustments
* Data mining uses both new and legacy systems
* It helps businesses make informed decisions
* It helps detect credit risks and fraud
* It helps data scientists easily analyze enormous amounts of data quickly
* Data scientists can use the information to detect fraud, build risk models, and improve product safety
* It helps data scientists quickly initiate automated predictions of behaviors and trends and discover hidden patterns.

### Data Mining tools

**Artificial Intelligence**

AI systems perform analytical functions that mimic human intelligence, such as learning, planning, problem-solving, and reasoning.

**Association Rule Learning**

This toolset, also called market basket analysis, searches for relationships among dataset variables. For example, association rule learning can determine which products are frequently purchased together (e.g., a smartphone and a protective case).

**Clustering**

This process partitions datasets into a set of meaningful sub-classes, known as clusters. The process helps users understand the natural structure or grouping within the data.

**Classification**

This technique assigns particular items in a dataset to different target categories or classes. The goal is to develop accurate predictions within the target class for each case in the data.

**Data Analytics**

The data analytics process enables professionals to evaluate digital information and turn it into useful business intelligence.

**Data Cleansing and Preparation**

This technique transforms the data into a form optimal for further analysis and processing. Preparation includes activities such as identifying and removing errors and missing or duplicate data.

**Data Warehousing**

Data warehousing consists of an extensive collection of business data that businesses use to help them make decisions. Warehousing is a fundamental and necessary component of most large-scale data mining efforts.

**Machine Learning**

Related to the AI technique mentioned earlier, machine learning is a computer programming technique that employs statistical probabilities to provide computers with the ability to learn without human intervention or being manually programmed.

**Regression**

The regression technique predicts a range of numeric values in categories such as sales, stock prices, or even temperature. The ranges are based on the information found in a particular data set.

**Two specific tools need mentioning.**

**R.** This language is an open-source tool used for graphics and statistical computing. It provides analysts with a wide selection of statistical tests, classification and graphical techniques, and time-series analysis.

**Oracle Data Mining (ODM).** This tool is a module of the Oracle Advanced Analytics Database. It helps data analysts make predictions and generate detailed insights. Analysts use ODM to predict customer behavior, develop customer profiles, and identify cross-selling opportunities.

### Data Mining Applications

**Retail**

Online retailers mine customer data and internet clickstream records to help them target marketing campaigns, ads and promotional offers to individual shoppers. Data mining and predictive modeling also power the recommendation engines that suggest possible purchases to website visitors, as well as inventory and supply chain management activities.

**Financial services**

Banks and credit card companies use data mining tools to build financial risk models, detect fraudulent transactions and vet loan and credit applications. Data mining also plays a key role in marketing and in identifying potential upselling opportunities with existing customers.

**Insurance**

Insurers rely on data mining to aid in pricing insurance policies and deciding whether to approve policy applications, including risk modeling and management for prospective customers.

**Manufacturing**

Data mining applications for manufacturers include efforts to improve uptime and operational efficiency in production plants, supply chain performance and product safety.

**Entertainment**

Streaming services do data mining to analyze what users are watching or listening to and to make personalized recommendations based on people's viewing and listening habits.

**Healthcare**

Data mining helps doctors diagnose medical conditions, treat patients and analyze X-rays and other medical imaging results. Medical research also depends heavily on data mining, machine learning and other forms of analytics.

## Data Wrangling

Data Wrangling is the process of gathering, collecting, and transforming Raw data into another format for better understanding, decision-making, accessing, and analysis in less time. Data Wrangling is also known as Data Munging.

**Example**

Books selling Website want to show top-selling books of different domains, according to user preference. For example, a new user search for motivational books, then they want to show those motivational books which sell the most or having a high rating, etc.

But on their website, there are plenty of raw data from different users. Here the concept of Data Munging or Data Wrangling is used. As we know Data is not Wrangled by System. This process is done by Data Scientists. So, the data Scientist will wrangle data in such a way that they will sort that motivational book that are sold more or have high ratings or user buy this book with these packages of Books, etc. On the basis of that, the new user will make choice. This will explain the importance of Data wrangling.

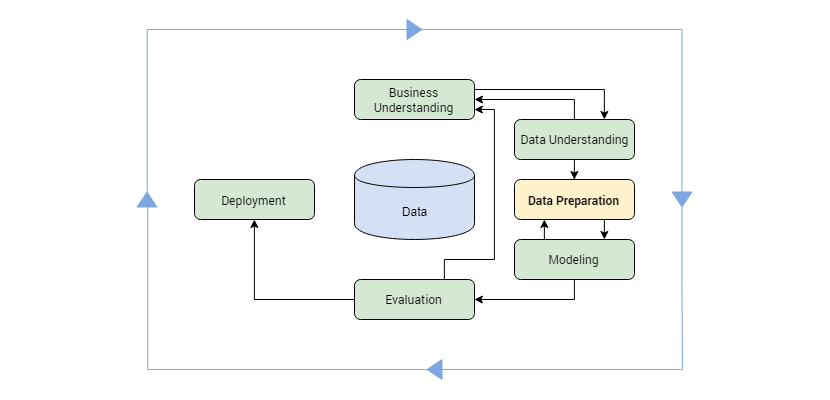


Image 1: Data Wrangling

Reference: <https://theappsolutions.com/blog/development/data-wrangling-guide-to-data-preparation/>

What is “raw data”?

It is any repository data (texts, images, database records) that is documented but yet to be processed and fully integrated into the system.

The process of wrangling can be described as “digesting” data (often referred to as “munging” thus the alternative term “data munging”) and making it useful (aka usable) for the system. It can be described as a preparation stage for every other data-related operation.

Data Wrangling is usually accompanied by Mapping. The term “Data Mapping” refers to the element of the wrangling process that involves identifying source data fields to their respective target data fields. While Wrangling is dedicated to transforming data, Mapping is about connecting the dots between different elements.

### What is the Purpose of Data Wrangling?

The primary purpose of data wrangling can be described as getting data in coherent shape. In other words, it is making raw data usable. It provides the substance for further proceedings.

As such, Data Wrangling acts as a preparation stage for the data-mining operation. Process-wise these two operations are coupled together as you can’t do one without another.

Overall, data wrangling covers the following processes:

* Getting data from the various source into one place
* Piecing the data together according to the determined setting
* Cleaning the data from the noise or erroneous, missing elements

### Data Wrangling Steps

**Data Discovery:**

This is an all-encompassing term that describes understanding what your data is all about. In this first step, you get familiar with your data

**Data Structuring:**

When you collect raw data, it initially is in all shapes and sizes, and has no definite structure. Such data needs to be restructured to suit the analytical model that your enterprise plans to deploy

**Data Cleaning:**

Raw data comes with some errors that need to be fixed before data is passed on to the next stage. Cleaning involves the tackling of outliers, making corrections, or deleting bad data completely

**Data Enriching:**

By this stage, you have kind of become familiar with the data in hand. Now is the time to ask yourself this question – do you need to embellish the raw data? Do you want to augment it with other data?

**Data Validating:**

This activity surfaces data quality issues, and they have to be addressed with the necessary transformations. The rules of validation rules require repetitive programming steps to check the authenticity and the quality of your data

**Data Publishing:**

Once all the above steps are completed, the final output of your data wrangling efforts are pushed downstream for your analytics needs

Data wrangling is a core iterative process that throws up the cleanest, most useful data possible before you start your actual analysis.



Image 2: Data Wrangling Steps

Reference: <https://theappsolutions.com/blog/development/data-wrangling-guide-to-data-preparation/>

### Data Wrangling Tools

* Excel Power Query / Spreadsheets: it is the basic manual structure wrangling tool.
* Open Refine: more sophisticated solutions, requires programming.
* Google Data Prep: for data exploration, cleaning, and feature engineering.
* Tabula: it's suitable for all kinds of data.
* Data Wrangler: used for data cleaning and transformation.
* CSVKit: used for converting data.

### Data Wrangling in Python

**Numpy (aka Numerical Python)**

the most basic package. Lots of features for operations on n-arrays and matrices in Python. The library provides vectorization of mathematical operations on the NumPy array type, which improves performance and accordingly speeds up the execution.

**Pandas**

designed for fast and easy data analysis operations. Useful for data structures with labeled axes. Explicit data alignment prevents common errors that result from misaligned data coming in from different sources.

**Matplotlib**

Python visualization module. Good for line graphs, pie charts, histograms, and other professional grade figures.

**Plotly**

for interactive, publication-quality graphs. Excellent for line plots, scatter plots, area charts, bar charts, error bars, box plots, histograms, heatmaps, subplots, multiple-axis, polar graphs, and bubble charts.

**Theano**

library for numerical computation similar to Numpy. This library is designed to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.

## Data manipulation

### What are Data Manipulations?

Data manipulation with python is defined as a process in the python programming language that enables users in data organization in order to make reading or interpreting the insights from the data more structured and comprises of having better design.

For example, arranging the employee’s names in alphabetical order will enable quicker searching of a particular employee by their name. The key feature of data manipulation is enabling faster business operations and also emphasize optimization in the process. Through proper manipulated data one can analyze trends, interpret insights from financial data, analyze consumer behavior or pattern, etc. Not only the analyzing, but it also enables users to neglect any unnecessary data in the set so that one can save space and only fill the limited space with important and necessary data.

### Pandas

module is basically an **open-source Python module**. It has a wide scope of use in the field of computing, data analysis, statistics, etc.

Pandas’ module uses the basic functionalities of the **NumPy module**.

There are various ways to install the Python Pandas module.

One of the easiest ways is to install using **Python package installer**i.e., **PIP**.

Type the following command in your Command-prompt:

**pip install pandas**

In order to add the Pandas and NumPy module to your code, we need to import these modules in our code.

|  |
| --- |
| import pandas  import numpy |

**Python Pandas Module – Data Structures**

Panda’s work around the following data structures:

* **Series**
* **Data Frame**
* **Panel**

These data structures are faster as compared to the NumPy arrays.

**1. Series**

Pandas Series is a 1-dimensional structure resembling arrays containing homogeneous data in it. It is a linear data structure and stores elements in a single dimension.

**Note**: The **size of the Series Data Structure in Pandas is immutable** i.e once set, it cannot be changed dynamically. While the **values/elements in the Series can be changed or manipulated**.

**Syntax:**

pandas.Series(input\_data, index, data\_type, copy)

**input\_data**: Takes input in vivid forms such as list, constants, NumPy arrays, Dict, etc.

**index**: Index values passed to the data.

**data\_type**: Recognizes the data type.

**copy**: Copies Data. The default value is False.

**Example:**

|  |
| --- |
| import pandas  import numpy  input = numpy.array(['John','Bran','Sam','Peter'])  series\_data = pandas.Series(input,index=[10,11,12,13])  print(series\_data) |

In the above code snippet, we have provided the input using NumPy arrays and set the index values to the input data.

**Output:**

10 John

11 Bran

12 Sam

13 Peter

dtype: object

**2. DataFrame**

Python Pandas module provides DataFrame that is a 2-dimensional structure, resembling the 2-D arrays. Here, the input data is framed in the form of rows and columns.

**Note**: The **size of the DataFrame Data Structure in Pandas is mutable**.

**Syntax**:

pandas.DataFrame(input\_data, index\_value, columns, data\_type, copy)

**input\_data**: Takes input in vivid forms such as list, series, NumPy arrays, Dict, another DataFrame, etc.

**index** **values**: Index values being passed to the data.

**data\_type**: Recognizes the data type of each column.

**copy**: Copy Data. The default value is False.

**columns:**Labels provided the data of the columns.

**Example:**

|  |
| --- |
| import pandas  input = [['John','Pune'],['Bran','Mumbai'],['Peter','Delhi']]  data\_frame = pandas.DataFrame(input,columns=['Name','City'],index=[1,2,3])  print(data\_frame) |

In the above code, we have provided the input using lists, have added labels: ‘Name’ and ‘City’ to the columns and have set the index values for the same.

**Output:**

Name City

1 John Pune

2 Bran Mumbai

3 Peter Delhi

**3. Panel**

Python Pandas module offers a Panel that is a 3-dimensional data structure and contains 3 axes to serve the following functions:

**items**: (axis 0) Every item of it corresponds to a DataFrame in it.

**major\_axis**: (axis 1) It corresponds to the rows of each DataFrame.

**minor\_axis**: (axis 2) It corresponds to the columns of each DataFrame.

**Syntax:**

pandas.Panel(input\_data, items, major\_axis, minor\_axis, data\_type, copy)

### 9 Effective Pandas Techniques in Python for Data Manipulation

Pandas is an open-source python library that implements easy, high-performance data structures and data analysis tools. The name comes from the term ‘panel data’, which relates to multidimensional data sets found in statistics and econometrics.

**Installation**

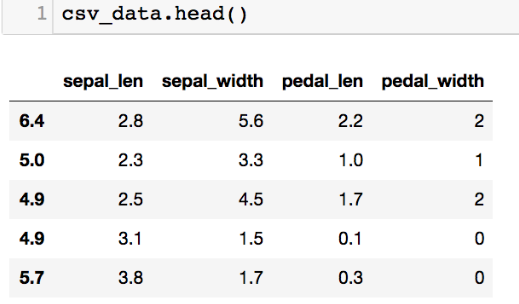
To install pandas, just run pip install pandas inside Python environment. Then we can import pandas as pd.

[](https://149695847.v2.pressablecdn.com/wp-content/uploads/2019/01/f1-1.png)

One of the most familiar things that pandas is used for is reading in CSV files, utilising pd.read\_csv. It is often the starting point for practising pandas.

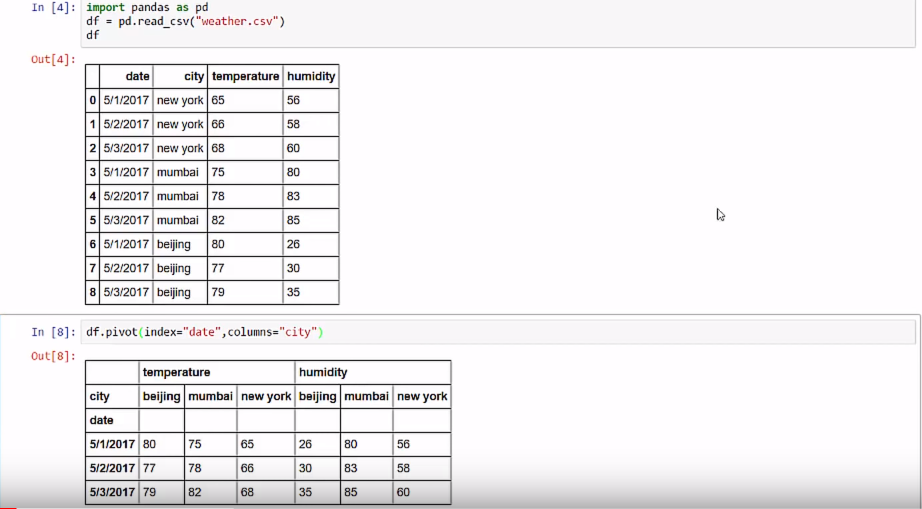
[](https://149695847.v2.pressablecdn.com/wp-content/uploads/2019/01/f2.png)

pd.read\_csv loads this data into a DataFrame. This can be considered as essentially a table or spreadsheet. Once loaded we can take a quick glimpse of the dataset by calling head() on the data frame.

[](https://149695847.v2.pressablecdn.com/wp-content/uploads/2019/01/f3-1.png)

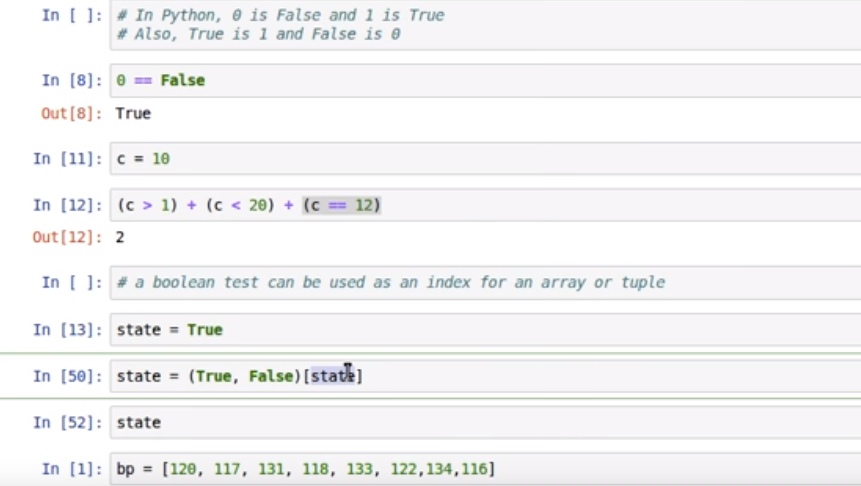
**1.Pivot Table**

Pandas can be practised to produce MS Excel style pivot tables. For example, in a table, a key [column](https://analyticsindiamag.com/most-common-activation-functions-in-neural-networks-and-rationale-behind-it/) which has missing values. We can impute it using mean amount of other groups.

[](https://149695847.v2.pressablecdn.com/wp-content/uploads/2019/01/pivot_1.png)

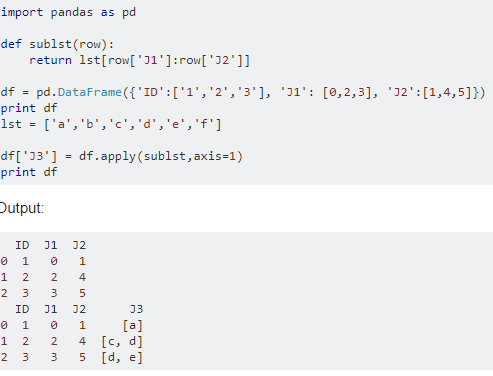
**2.Boolean Indexing**

Boolean Indexing is used if user wants to filter the values of a column based on conditions from another set of columns. For instance, we want a list of all students who are not scholars and got a loan. Boolean indexing can support here.

[](https://149695847.v2.pressablecdn.com/wp-content/uploads/2019/01/boolean_1.png)

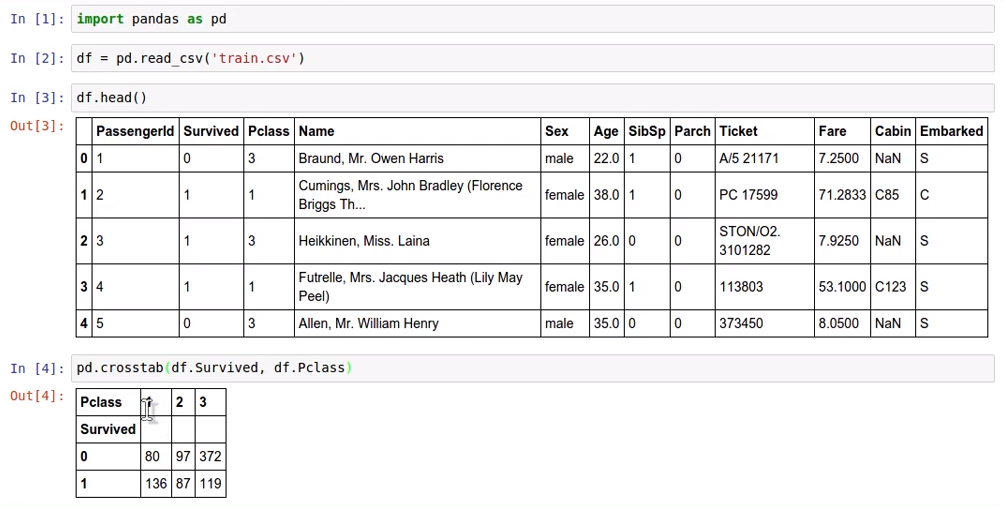
**3.Apply Function**

It is one of the regularly used functions for working with data and building new variables. Apply gains some value after passing each row/column of a data frame with some function. The function can be either default or user-defined.

[](https://149695847.v2.pressablecdn.com/wp-content/uploads/2019/01/apply_1.png)

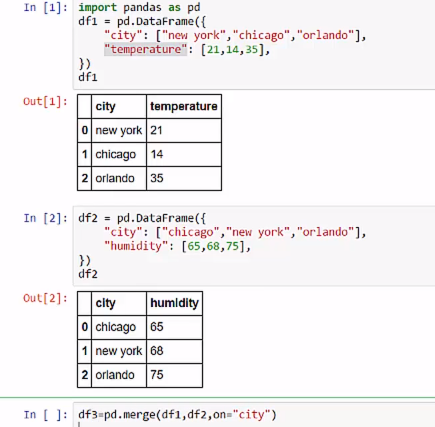
**4.Crosstab**

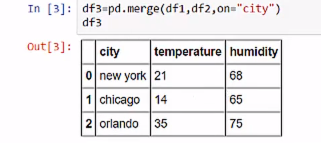
This function is used to get an original view of the data. The function provides scope to validate some fundamental hypothesis. For instance, one column is expected to affect the other column.

[](https://149695847.v2.pressablecdn.com/wp-content/uploads/2019/01/crosstab_1.png)

**5.Merge DataFrames**

Merging data frames is vital when a user has data coming from various sources to be related.

[](https://149695847.v2.pressablecdn.com/wp-content/uploads/2019/01/merge_1.png)

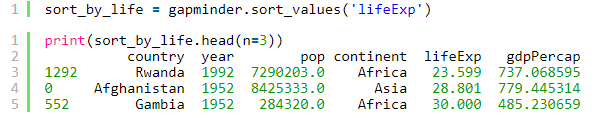
[](https://149695847.v2.pressablecdn.com/wp-content/uploads/2019/01/merge_2.png)

**6.Sorting DataFrames**

When we want to sort Pandas data frame in a particular way. When a user wants to sort pandas data frame based on the values of one or more columns or sort based on the contents of row index or row names of the panda’s data frame. Pandas data frame has two useful functions

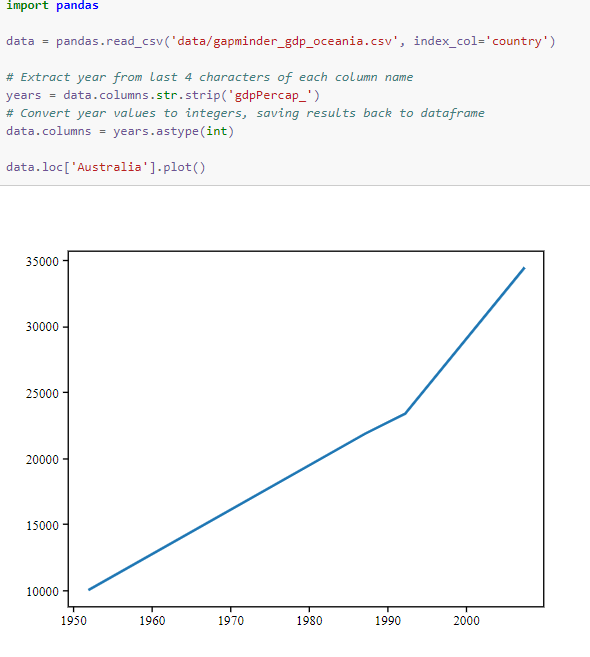
* sort\_values(): this command is used to sort pandas data frame by one or more columns
* sort\_index(): this command is used to sort pandas data frame by row index

The above functions come with various options, like sorting the data frame in a specific order, place, sorting with missing values, sorting by a specific algorithm and many more.

[](https://149695847.v2.pressablecdn.com/wp-content/uploads/2019/01/soprting_1.png)

**7.Plotting**

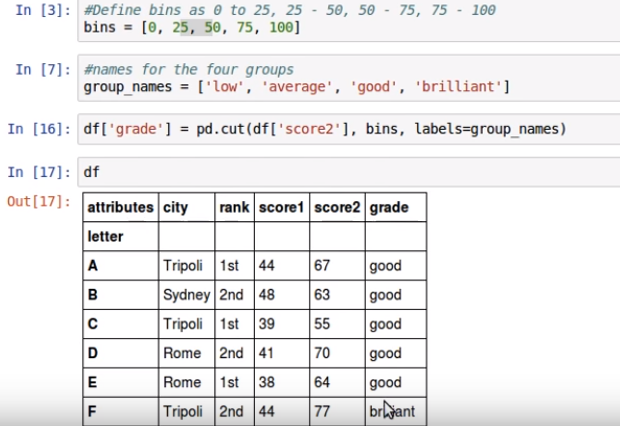
Analyzing data from different columns can be very illuminating. Pandas make doing so simple with multi-column DataFrames. By default, calling df.plot() will make pandas to over-plot all column data, with each column as a single line.

[](https://149695847.v2.pressablecdn.com/wp-content/uploads/2019/01/plotting_1.png)

**8.Cut function**

Seldom numerical values make more sense if grouped together. For illustration, if we’re examining to model aeroplanes (#planes flying) with the time of the day (minutes). The specific minute of an hour might not be that appropriate for predicting air traffic as analysed to the actual period of the day like “Morning”, “Afternoon”, “Evening”, “Night”, “Late Night”. Modelling air traffic this way will be more intuitive and will bypass overfitting.

[](https://149695847.v2.pressablecdn.com/wp-content/uploads/2019/01/cut_1.png)

[](https://149695847.v2.pressablecdn.com/wp-content/uploads/2019/01/cut_2.png)

**9.Impute Missing Values**

Imputing relates to applying a model to restore missing values.

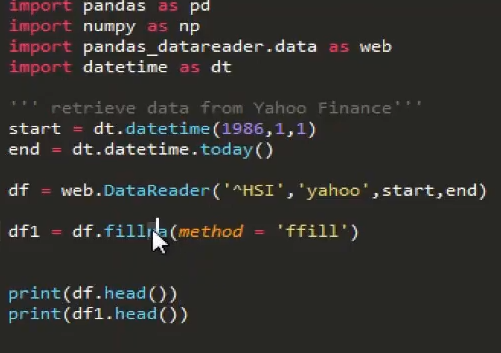
There are several options users can consider while replacing a missing value, for example:

* A fixed value that has meaning within the domain, such as 0, distinct from all other values.
* A value from another randomly chosen from the record.
* A mean, median or mode value replaced for the column.
* A value determined by another predictive model.

Any imputing conducted on the training dataset will have to be performed on new data in the future when predictions are required from the finalized model. This needs to be taken into factor when choosing how to impute the missing values.

For example, if one chooses to impute with mean column values, the mean column values will need to be stored to file for later exercise new data that has missing values.

Pandas provide the fillna() function for returning values with a specific value.

[](https://149695847.v2.pressablecdn.com/wp-content/uploads/2019/01/missing_1.png)

# Data cleaning and pre-processing techniques

## Data Cleaning

Data Cleaning means the process of identifying the incorrect, incomplete, inaccurate, irrelevant or missing part of the data and then modifying, replacing or deleting them according to the necessity. Data cleaning is considered a foundational element of the basic data science.

For example, if you conduct a survey and ask people for their phone numbers, people may enter their numbers in different formats.

When working with multiple data sources, there are many chances for data to be incorrect, duplicated, or mislabeled. If data is wrong, outcomes and algorithms are unreliable, even though they may look correct. Data cleaning is the process of changing or eliminating garbage, incorrect, duplicate, corrupted, or incomplete data in a dataset. There’s no such absolute way to describe the precise steps in the data cleaning process because the processes may vary from dataset to dataset. Data cleansing, data cleansing, or data scrub is that the initiative among the general data preparation process. Data cleaning plays an important part in developing reliable answers and within the analytical process and is observed to be a basic feature of the info science basics. The motive of data cleaning services is to construct uniform and standardized data sets that enable data analytical tools and business intelligence easy access and perceive accurate data for each problem.

### Why data cleaning is essential?

Data cleaning is the most important task that should be done as a data science professional. Having wrong or bad quality data can be detrimental to processes and analysis. Having clean data will ultimately increase overall productivity and permit the very best quality information in your decision-making. Following are some reasons why data cleaning is essential:



Image 3: Data Cleaning

Reference: <https://www.analyticsvidhya.com/blog/2021/06/data-cleaning-using-pandas/>

**1. Error-Free Data**

When multiple sources of data are combined there may be chances of so much error. Through Data Cleaning, errors can be removed from data. Having clean data which is free from wrong and garbage values can help in performing analysis faster as well as efficiently. By doing this task our considerable amount of time is saved. If we use data containing garbage values, the results won’t be accurate. When we don’t use accurate data, surely, we will make mistakes. Monitoring errors and good reporting helps to find where errors are coming from, and also makes it easier to fix incorrect or corrupt data for future applications.

**2. Data Quality:**

The quality of the data is the degree to which it follows the rules of particular requirements. For example, if we have imported phone numbers data of different customers, and in some places, we have added email addresses of customers in the data. But because our needs were straightforward for phone numbers, then the email addresses would be invalid data. Here some pieces of data follow a specific format. Some types of numbers have to be in a specific range. Some data cells might require a selected quite data like numeric, Boolean, etc. In every scenario, there are some mandatory constraints our data should follow. Certain conditions affect multiple fields of data in a particular form. Particular types of data have unique restrictions. If the data isn’t in the required format, it would always be invalid. Data cleaning will help us simplify this process and avoid useless data values.

**3. Accurate and Efficient:**

Ensuring the data is close to the correct values. We know that most of the data in a dataset are valid, and we should focus on establishing its accuracy. Even if the data is authentic and correct, it doesn’t mean the data is accurate. Determining accuracy helps to figure out the data entered is accurate or not. For example, the address of a customer is stored in the specified format, maybe it doesn’t need to be in the right one. The email has an additional character or value that makes it incorrect or invalid. Another example is the phone number of a customer. This means that we have to rely on data sources, to cross-check the data to figure out if it’s accurate or not. Depending on the kind of data we are using, we might be able to find various resources that could help us in this regard for cleaning.

**4. Complete Data:**

Completeness is the degree to which we should know all the required values. Completeness is a little more challenging to achieve than accuracy or quality. Because it’s nearly impossible to have all the info we need. Only known facts can be entered. We can try to complete data by redoing the data gathering activities like approaching the clients again, re-interviewing people, etc. For example, we might need to enter every customer’s contact information. But a number of them might not have email addresses. In this case, we have to leave those columns empty. If we have a system that requires us to fill all columns, we can try to enter missing or unknown there. But entering such values does not mean that the data is complete. It would be still being referred to as incomplete.

**5. Maintains Data Consistency:**

To ensure the data is consistent within the same dataset or across multiple datasets, we can measure consistency by comparing two similar systems. We can also check the data values within the same dataset to see if they are consistent or not. Consistency can be relational. For example, a customer’s age might be 25, which is a valid value and also accurate, but it is also stated as a senior citizen in the same system. In such cases, we have to cross-check the data, similar to measuring accuracy, and see which value is true. Is the client a 25-year-old? Or the client is a senior citizen? Only one of these values can be true. There are multiple ways to for your data consistent.

* By checking in different systems.
* By checking the source.
* By checking the latest data.

### 8 effective data cleaning techniques

* Remove duplicates
* Remove irrelevant data
* Standardize capitalization
* Convert data type
* Clear formatting
* Fix errors
* Language translation
* Handle missing values

**1. Remove Duplicates**

When you collect your data from a range of different places, or scrape your data, it’s likely that you will have duplicated entries. These duplicates could originate from human error where the person inputting the data or filling out a form made a mistake.

Duplicates will inevitably skew your data and/or confuse your results. They can also just make the data hard to read when you want to visualize it, so it’s best to remove them right away.

**2. Remove Irrelevant Data**

Irrelevant data will slow down and confuse any analysis that you want to do. So, deciphering what is relevant and what is not is necessary before you begin your data cleaning. For instance, if you are analyzing the age range of your customers, you don’t need to include their email addresses.

Other elements you’ll need to remove as they add nothing to your data include:

* Personal identifiable (PII) data
* URLs
* HTML tags
* Boilerplate text (for ex. in emails)
* Tracking codes
* Excessive blank space between text

**3. Standardize Capitalization**

Within your data, you need to make sure that the text is consistent. If you have a mixture of capitalization, this could lead to different erroneous categories being created.

It could also cause problems when you need to translate before processing as capitalization can change the meaning. For instance, Bill is a person's name whereas a bill or to bill is something else entirely.

If, in addition to data cleaning, you are text cleaning in order to process your data with a computer model, it’s much simpler to put everything in lowercase.

**4. Convert Data Types**

Numbers are the most common data type that you will need to convert when cleaning your data. Often numbers are imputed as text, however, in order to be processed, they need to appear as numerals.

If they are appearing as text, they are classed as a string and your analysis algorithms cannot perform mathematical equations on them.

The same is true for dates that are stored as text. These should all be changed to numerals. For example, if you have an entry that reads September 24th 2021, you’ll need to change that to read 09/24/2021.

**5. Clear Formatting**

Machine learning models can’t process your information if it is heavily formatted. If you are taking data from a range of sources, it’s likely that there are a number of different document formats. This can make your data confusing and incorrect.

You should remove any kind of formatting that has been applied to your documents, so you can start from zero. This is normally not a difficult process, both excel and google sheets, for example, have a simple standardization function to do this.

**6. Fix Errors**

It probably goes without saying that you’ll need to carefully remove any errors from your data. Errors as avoidable as typos could lead to you missing out on key findings from your data. Some of these can be avoided with something as simple as a quick spell-check.

Spelling mistakes or extra punctuation in data like an email address could mean you miss out on communicating with your customers. It could also lead to you sending unwanted emails to people who didn’t sign up for them.

Other errors can include inconsistencies in formatting. For example, if you have a column of US dollar amounts, you’ll have to convert any other currency type into US dollars so as to preserve a consistent standard currency. The same is true of any other form of measurement such as grams, ounces, etc.

**7. Language Translation**

To have consistent data, you’ll want everything in the same language.

The Natural Language Processing (NLP) models behind software used to analyze data are also predominantly monolingual, meaning they are not capable of processing multiple languages. So, you’ll need to translate everything into one language.

**8. Handle Missing Values**

When it comes to missing values, you have two options:

* Remove the observations that have this missing value
* Input the missing data

What you choose to do will depend on your analysis goals and what you want to do next with your data.

Removing the missing value completely might remove useful insights from your data. After all, there was a reason that you wanted to pull this information in the first place.

Therefore, it might be better to input the missing data by researching what should go in that field. If you don’t know what it is, you could replace it with the word missing. If it is numerical, you can place a zero in the missing field.

However, if there are so many missing values that there isn’t enough data to use, then you should remove the whole section.

### 6 Steps to Manipulate and Cleanse Data with Python

**1: Implementing missing values imputation** – This is a standard statistical imputing constant, using KNN imputation.

Outlier/Anomaly Detection is carried out using: Isolation Forest, One Class SVM, Local Outlier Factor, and/or outlier detection algorithms.

**2: Carrying out outlier/anomaly detection**- You can accomplish this by using Isolation Forest, One-Class SVM, and/or Local Outlier Factor outlier detection algorithms.

**3: Utilizing cleaning techniques from the X-Variable family**- In this instance, you want to apply custom functions, remove duplicates, as well as replace crucial values.

**4: Using Cleaning Techniques of the Y-Variable sort** – Here it is important to do label encoding, one-hot encoding, as well as dictionary mapping.

**5:’DataFrames’ need to be merged**- This step includes concatenating, merging, and joining.

**6: The last step consists of ‘parsing dates’**- Here you need to use auto-format detecting strings to accomplish ‘DateTime’ converting, including changing ‘DateTime’ objects to numbers.

Let’s go into detail for each step:

**ONE: Imputing Missing Values**

One of the most common issues you may come across in raw extracted data sets is missing values. As long as there are not too many of them, they can easily be imputed at this stage.

* Simple Imputation Methods, like mean, median, mode can be used to fill in missing values (NaN) with the statistical measure of each column. The parameter can be replaced with ‘mean’, ‘median’, ‘most\_frequent’ or mode, or ‘constant’ which is a manual value.
* KNN Imputing is a more complex method for imputing missing values. The KNN algorithm is used to find different data points like the ones missing values within the datasets.

It is important to note that in order to use KNN imputation, the data needs to be normalized to remove differences in scale. To use KNN imputation you will need to:

* Normalize the data
* KNN impute to fill in missing values
* Inverse scale/normalize the data again

**TWO: Outlier and Anomaly Detection**

* Isolation Forest is an algorithm used to return the anomaly score of the datasets. The algorithm selects a feature and isolates observations by randomly choosing a split value. Paths are then created by representing the value’s normality. The shorter the paths reveal anomalies. ‘Trees’ of shorter paths for these samples make up a ‘forest’ likely to reveal the anomalies.
* One Class SVM is another method for finding outliers. This is suited for instances where Isolation Forest cannot be applied due to excessive variance.
* Local Outlier Factor is the third method used to detect anomalies. The Local Outlier Factor measures the deviation of density in each dataset in comparison to the other. Samples that display lower density than their neighbors are likely to be outliers. This algorithm is distance based, meaning you will need to normalize the data before you can use it. This method is a high variance alternative to Isolation Forest.

It is important when using any of these three methods to be sure that the anomalies are not simply data clusters. You can use PCA visualization to double-check.

**THREE: X-Variable Cleaning Methods**

* Applying customs functions is necessary when cleaning cannot be done via the built in functions. In this case, you may need to write your own functions but you can try to use an external built in function first.
* Removing duplicates is an important part of data cleansing. This can be done with data.drop\_duplicates(), which removes rows of identical value. You must be careful to check that the duplicate rows are not errors, especially in smaller datasets.
* Sampling data points is important for large datasets. This allows you to sample random data points and can be done with data.sample(number\_of\_samples).
* Renaming columns is done with .rename, where the key is the original column name and the value is the renamed value.
* Replacing values can be done with data,replace(), which takes two values from the dataframe that you will replace with other values. This is useful for imputing missing values so that imputing algorithms can work effectively.

**FOUR: Y-Variable Cleaning Methods**

* Label Encoding is necessary for categorical y-variables. If your data has two classes, they need to be converted into 0 and 1, because machine learning algorithms can only operate with mathematical characters. You can do this by using the .map() function, which converts a dictionary or original names and replaces the values with numbers. If there are too many classes to manually map, you can use sklearn’s automated method. This method is beneficial because the data can easily be reverted to the original format by using encoder.inverse\_transform(array).
* One-Hot Encoding may be preferred in specific cases when you have many classes and you do not want to place quantitative measures on the data. With one-hot encoding, every y-value is a vector the length of the number of each class, with a ‘1’ marking an index within the vector and the rest of the values are marked with ‘0’s. Pandas has a built in function called get\_dummies, that can automatically take forms and output the one-hot encoded dataframe.

**FIVE: Merging DataFrames**

* Concatenation is the top-down method of joining DataFrames
* Merging is the left to right process of merging two DataFrames
* Joining is for other types of merging. Merging only combines rows where there is a common keyword in both data frames. Joining includes left outer joining, where all keywords in the left DataFrame are included, while rows in the right DataFrame are only included if their keywords exist in the left one

**SIX: Parsing Dates**

* Dates can be very difficult sets of data, but are also some of the most important. This is why it is so important for you to understand how to correctly work with this type of data
* Auto-format detecting string to datetime converting is a crucial skill, as datasets rarely come with datetime objects that can be readily accessed. You can use dateutil to automatically determine the location of days, months and years.
* Converting dates to numbers is necessary for models to be able to understand the concept of time. Datetime objects are converted to numbers. In other words, each date represents the number of days passed since the earliest date in your dataset. The function is applied to the date column using .apply()

## Data Pre-processing

Data preprocessing is a step in the data mining and data analysis process that takes raw data and transforms it into a format that can be understood and analyzed by computers and machine learning.

Raw, real-world data in the form of text, images, video, etc., is messy. Not only may it contain errors and inconsistencies, but it is often incomplete, and doesn’t have a regular, uniform design.

Machines like to process nice and tidy information – they read data as 1s and 0s. So calculating structured data, like whole numbers and percentages is easy. However, unstructured data, in the form of text and images must first be cleaned and formatted before analysis.

**Data Preprocessing Importance**

When using data sets to train machine learning models, you’ll often hear the phrase **“garbage in, garbage out”** This means that if you use bad or “dirty” data to train your model, you’ll end up with a bad, improperly trained model that won’t actually be relevant to your analysis.

Good, preprocessed data is even more important than the most powerful algorithms, to the point that machine learning models trained with bad data could actually be harmful to the analysis you’re trying to do – giving you “garbage” results.



Image 4: Data Pre-processing

Reference: <https://monkeylearn.com/blog/data-preprocessing/>

Depending on your data gathering techniques and sources, you may end up with data that’s out of range or includes an incorrect feature, like household income below zero or an image from a set of “zoo animals” that is actually a tree. Your set could have missing values or fields. Or text data, for example, will often have misspelled words and irrelevant symbols, URLs, etc.

When you properly preprocess and clean your data, you’ll set yourself up for much more accurate downstream processes. We often hear about the importance of “data-driven decision making,” but if these decisions are driven by bad data, they’re simply bad decisions.

### Data Preprocessing Steps

* Data quality assessment
* Data cleaning
* Data transformation
* Data reduction

**1.Data quality assessment**



Image 5: Data Sources

Reference: <https://medium.com/easyread/basics-of-data-preprocessing-71c314bc7188>

Take a good look at your data and get an idea of its overall quality, relevance to your project, and consistency. There are a number of data anomalies and inherent problems to look out for in almost any data set, for example:

* **Mismatched data types**: When you collect data from many different sources, it may come to you in different formats. While the ultimate goal of this entire process is to reformat your data for machines, you still need to begin with similarly formatted data. For example, if part of your analysis involves family income from multiple countries, you’ll have to convert each income amount into a single currency.
* **Mixed data values**: Perhaps different sources use different descriptors for features – for example, man or male. These value descriptors should all be made uniform.
* **Data outliers**: Outliers can have a huge impact on data analysis results. For example if you're averaging test scores for a class, and one student didn’t respond to any of the questions, their 0% could greatly skew the results.
* **Missing data**: Take a look for missing data fields, blank spaces in text, or unanswered survey questions. This could be due to human error or incomplete data. To take care of missing data, you’ll have to perform data cleaning.

**2. Data cleaning**

Data cleaning is the process of adding missing data and correcting, repairing, or removing incorrect or irrelevant data from a data set. Dating cleaning is the most important step of preprocessing because it will ensure that your data is ready to go for your downstream needs.



Image 6: Data Cleaning

Reference: <https://medium.com/easyread/basics-of-data-preprocessing-71c314bc7188>

Data cleaning will correct all of the inconsistent data you uncovered in your data quality assessment. Depending on the kind of data you’re working with, there are a number of possible cleaners you’ll need to run your data through.

**Missing data**

There are a number of ways to correct for missing data, but the two most common are:

* **Ignore the tuples**: A tuple is an ordered list or sequence of numbers or entities. If multiple values are missing within tuples, you may simply discard the tuples with that missing information. This is only recommended for large data sets, when a few ignored tuples won’t harm further analysis.
* **Manually** **fill in missing data**: This can be tedious, but is definitely necessary when working with smaller data sets.

**Noisy data**

Data cleaning also includes fixing “noisy” data. This is data that includes unnecessary data points, irrelevant data, and data that’s more difficult to group together.

**Binning:** Binning sorts data of a wide data set into smaller groups of more similar data. It’s often used when analyzing demographics. Income, for example, could be grouped: $35,000-$50,000, $50,000-$75,000, etc.

**Regression:** Regression is used to decide which variables will actually apply to your analysis. Regression analysis is used to smooth large amounts of data. This will help you get a handle on your data, so you’re not overburdened with unnecessary data.

**Clustering:** Clustering algorithms are used to properly group data, so that it can be analyzed with like data. They’re generally used in unsupervised learning, when not a lot is known about the relationships within your data.

If you’re working with text data, for example, some things you should consider when cleaning your data are:

* Remove URLs, symbols, emojis, etc., that aren’t relevant to your analysis
* Translate all text into the language you’ll be working in
* Remove HTML tags
* Remove boilerplate email text
* Remove unnecessary blank text between words
* Remove duplicate data

After data cleaning, you may realize you have insufficient data for the task at hand. At this point you can also perform data wrangling or data enrichment to add new data sets and run them through quality assessment and cleaning again before adding them to your original data.

**3. Data transformation**

With data cleaning, we’ve already begun to modify our data, but data transformation will begin the process of turning the data into the proper format(s) you’ll need for analysis and other downstream processes.

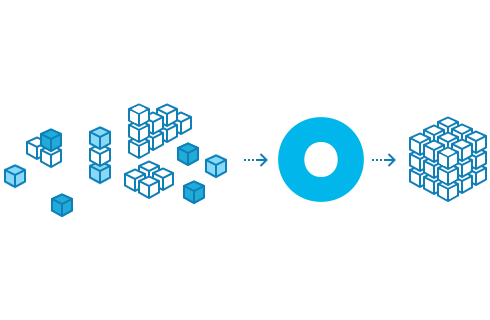


Image 7: Data Transformation

Reference: <https://medium.com/easyread/basics-of-data-preprocessing-71c314bc7188>

This generally happens in one or more of the below:

1. Aggregation
2. Normalization
3. Feature selection
4. Discreditization
5. Concept hierarchy generation

* **Aggregation:** Data aggregation combines all of your data together in a uniform format.
* **Normalization**: Normalization scales your data into a regularized range so that you can compare it more accurately. For example, if you’re comparing employee loss or gain within a number of companies (some with just a dozen employees and some with 200+), you’ll have to scale them within a specified range, like -1.0 to 1.0 or 0.0 to 1.0.
* **Feature selection:** Feature selection is the process of deciding which variables (features, characteristics, categories, etc.) are most important to your analysis. These features will be used to train ML models. It’s important to remember, that the more features you choose to use, the longer the training process and, sometimes, the less accurate your results, because some feature characteristics may overlap or be less present in the data.

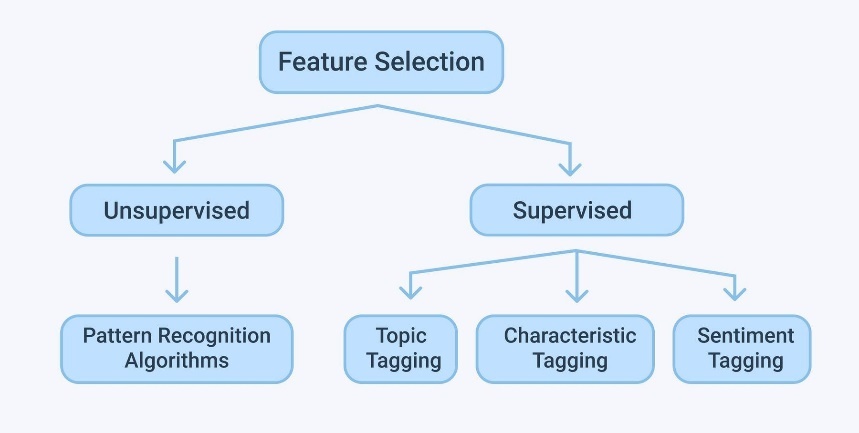


Image 8: Feature Selction

Reference: <https://medium.com/easyread/basics-of-data-preprocessing-71c314bc7188>

* **Discreditization:** Discreditiization pools data into smaller intervals. It’s somewhat similar to binning, but usually happens after data has been cleaned. For example, when calculating average daily exercise, rather than using the exact minutes and seconds, you could join together data to fall into 0-15 minutes, 15-30, etc.
* **Concept hierarchy generation:** Concept hierarchy generation can add a hierarchy within and between your features that wasn’t present in the original data. If your analysis contains wolves and coyotes, for example, you could add the hierarchy for their genus: canis.

**4. Data reduction**

The more data you’re working with, the harder it will be to analyze, even after cleaning and transforming it. Depending on your task at hand, you may actually have more data than you need. Especially when working with text analysis, much of regular human speech is superfluous or irrelevant to the needs of the researcher. Data reduction not only makes the analysis easier and more accurate, but cuts down on data storage.

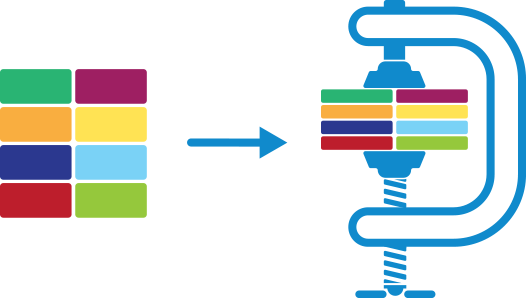


Image 9: Data reduction

Reference: <https://medium.com/easyread/basics-of-data-preprocessing-71c314bc7188>

It will also help identify the most important features to the process at hand.

* **Attribute selection:** Similar to discreditization, attribute selection can fit your data into smaller pools. It, essentially, combines tags or features, so that tags like male/female and professor could be combined into male professor/female professor.
* **Numerosity reduction:** This will help with data storage and transmission. You can use a regression model, for example, to use only the data and variables that are relevant to your analysis.
* **Dimensionality reduction:** This, again, reduces the amount of data used to help facilitate analysis and downstream processes. Algorithms like K-nearest neighbors use pattern recognition to combine similar data and make it more manageable.

### Steps involved in data preprocessing:

* Importing the required Libraries
* Importing the data set
* Handling the Missing Data.
* Encoding Categorical Data.
* Splitting the data set into test set and training set.
* Feature Scaling.

**Step 1: Importing the required Libraries**

To follow along you will need to download this dataset: [**Data.csv**](https://github.com/afrozchakure/Internity-Summer-Internship-Work/tree/master/Blogs/Preprocessing)

Every time we make a new model, we will require to import Numpy and Pandas. Numpy is a Library which contains Mathematical functions and is used for scientific computing while Pandas is used to import and manage the data sets.

**import pandas as pd  
import numpy as np**

Here we are importing the pandas and Numpy library and assigning a shortcut “pd” and “np” respectively.

**Step 2: Importing the Dataset**

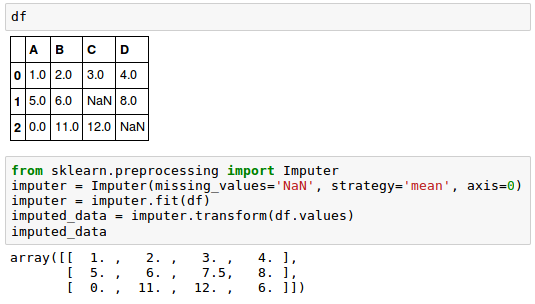
Data sets are available in .csv format. A CSV file stores tabular data in plain text. Each line of the file is a data record. We use the read\_csv method of the pandas library to read a local CSV file as a **dataframe**.

**dataset = pd.read\_csv('Data.csv')**

After carefully inspecting our dataset, we are going to create a matrix of features in our dataset (X) and create a dependent vector (Y) with their respective observations. To read the columns, we will use iloc of pandas (used to fix the indexes for selection) which takes two parameters — [row selection, column selection].

**X = dataset.iloc[:, :-1].values  
y = dataset.iloc[:, 3].values**

**Step 3: Handling the Missing Data**



The data we get is rarely homogenous. Sometimes data can be missing and it needs to be handled so that it does not reduce the performance of our machine learning model.

To do this we need to replace the missing data by the Mean or Median of the entire column. For this we will be using the sklearn.preprocessing Library which contains a class called Imputer which will help us in taking care of our missing data.

**from sklearn.preprocessing import Imputer  
imputer = Imputer(missing\_values = "NaN", strategy = "mean", axis = 0)**

Our object name is **imputer.**The Imputer class can take parameters like:

1. **missing\_values** : It is the placeholder for the missing values. All occurrences of missing\_values will be imputed. We can give it an integer or “NaN” for it to find missing values.
2. **strategy** : It is the imputation strategy — If “mean”, then replace missing values using the mean along the axis (Column). Other strategies include “median” and “most\_frequent”.
3. **axis** : It can be assigned 0 or 1, 0 to impute along columns and 1 to impute along rows.

Now we fit the imputer object to our data.

**imputer = imputer.fit(X[:, 1:3])**

Now replacing the missing values with the mean of the column by using transform method.

**X[:, 1:3] = imputer.transform(X[:, 1:3])**

**Step 4: Encoding categorical data**



Converting Categorical data into dummy variables

Any variable that is not quantitative is categorical. Examples include Hair color, gender, field of study, college attended, political affiliation, status of disease infection.

But why encoding ?

We cannot use values like “Male” and “Female” in mathematical equations of the model so we need to encode these variables into numbers.

To do this we import “LabelEncoder” class from “sklearn.preprocessing” library and create an object labelencoder\_X of the LabelEncoder class. After that we use the fit\_transform method on the categorical features.

After Encoding it is necessary to distinguish between between the variables in the same column, for this we will use OneHotEncoder class from sklearn.preprocessing library.

**One-Hot Encoding**

One hot encoding transforms categorical features to a format that works better with classification and regression algorithms.

**from sklearn.preprocessing import LabelEncoder, OneHotEncoder  
labelencoder\_X = LabelEncoder()  
X[:, 0] = labelencoder\_X.fit\_transform(X[:, 0])onehotencoder = OneHotEncoder(categorical\_features = [0])  
X = onehotencoder.fit\_transform(X).toarray()labelencoder\_y = LabelEncoder()  
y = labelencoder\_y.fit\_transform(y)**

**Step 5: Splitting the Data set into Training set and Test Set**

Now we divide our data into two sets, one for training our model called the**training set** and the other for testing the performance of our model called the **test set**. The split is generally 80/20. To do this we import the “train\_test\_split” method of “sklearn.model\_selection” library.

**from sklearn.model\_selection import train\_test\_split**

Now to build our training and test sets, we will create 4 sets —

1. **X\_train** (training part of the matrix of features),
2. **X\_test** (test part of the matrix of features),
3. **Y\_train** (training part of the dependent variables associated with the X train sets, and therefore also the same indices) ,
4. **Y\_test** (test part of the dependent variables associated with the X test sets, and therefore also the same indices).

We will assign to them the test\_train\_split, which takes the parameters — arrays (X and Y), test\_size (Specifies the ratio in which to split the data set).

**X\_train, X\_test, Y\_train, Y\_test = train\_test\_split( X , Y , test\_size = 0.2, random\_state = 0)**

**Step 6: Feature Scaling**

Most of the machine learning algorithms use the **Euclidean distance** between two data points in their computations. Because of this, **high magnitudes features will weigh more** in the distance calculations **than features with low magnitudes**. To avoid this Feature standardization or Z-score normalization is used. This is done by using “StandardScaler” class of “sklearn.preprocessing”.

**from sklearn.preprocessing import StandardScaler  
sc\_X = StandardScaler()**

Further we will transform our X\_test set while we will need to fit as well as transform our X\_train set.The transform function will transform all the data to a same standardized scale.

**X\_train = sc\_X.fit\_transform(X\_train)  
X\_test = sc\_X.transform(X\_test)**

# Data analytics project lifecycle

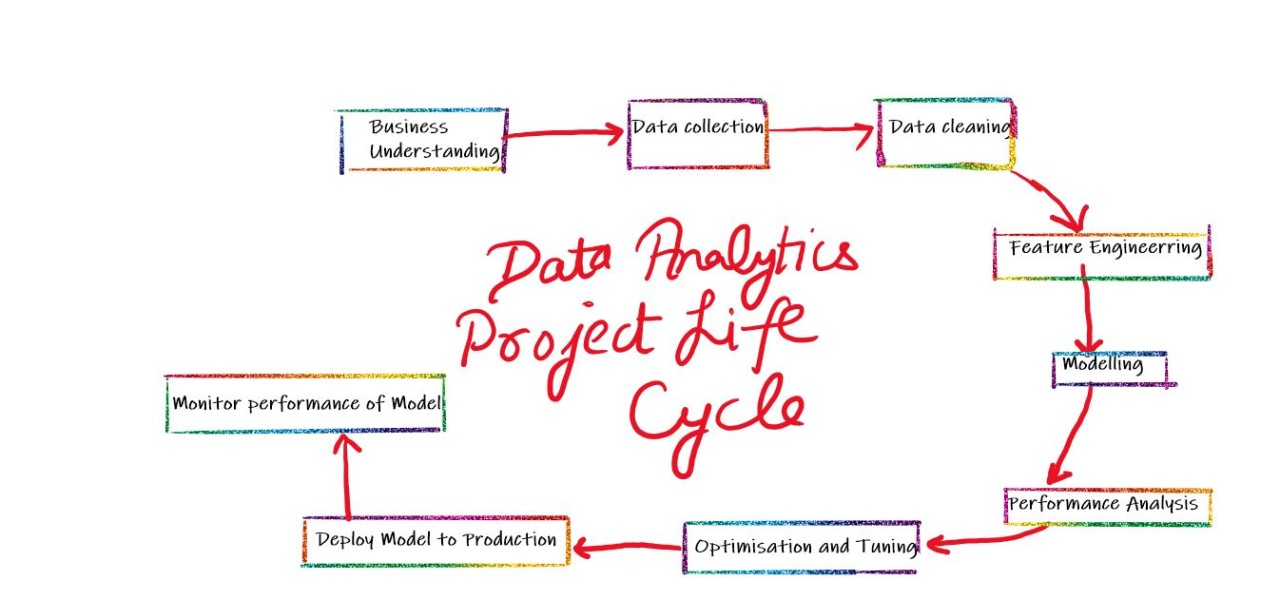


Image 10: Data Analytics Project Life Cycle

Reference: <https://medium.com/easyread/basics-of-data-preprocessing-71c314bc7188>

A Data Analytics Project has the below steps to be followed from start to end, these steps are performed in sequence but sometime after analyzing the situation there can be need to getting back to the previous step followed instead of moving ahead.

**Domain/Business Understanding:**

This initial phase focuses on understanding the project objectives and requirements from a business perspective, and then converting this knowledge into a data mining.

Understand the Business Process, beliefs, data generation process and storage.

It’s very important to spend a good amount of time in this step so that the future once become more relevant to perform and there are less chances of errors.

**Data collection/Data Exploration:**

The data understanding phase starts with an initial data collection and proceeds with activities in order to get familiar with the data, to identify data quality problems, to discover first insights into the data, or to detect interesting subsets to form hypotheses for hidden information.

**Data Cleaning:**

It basically includes filling the missing values, dropping irrelevant and duplicate data. Handling outliers and unnatural values.

**Feature Engineering:**

Feature engineering is about creating new input features from your existing ones.

In general, you can think of data cleaning as a process of subtraction and feature engineering as a process of addition.

Getting Qualitative and quantitative information from the data, Feature selection using Data Visualization.

**Modelling:**

In this phase, various modelling techniques are selected and applied and their parameters are calibrated to optimal values. Typically, there are several techniques for the same data mining problem type. Some techniques have specific requirements on the form of data. Therefore, it is often required to step back to the data preparation phase.

**Evaluation:**

At this stage in the project, you have built a model (or models) that appears to have high quality, from a data analysis perspective. Before proceeding to final deployment of the model, it is important to evaluate the model thoroughly and review the steps executed to construct the model, to be certain it properly achieves the business objectives.

**Optimization and Tuning:**

Even after evaluation of Model there can be certain requirements which can be raised at time of evaluation, Analyzing the same and integrating in the model.

**Deploy the Model to production:**

Finally, after all the UAT and implementing changes during UAT, the model is finally put to production. i.e the same goes live.

**Monitor the performance during Production:**

It takes some time to settle the model in live environment and familiarize the audience with the new features, so a continuous monitor is required so that any discrepancies can be solved straight away.

# Numerical Computing using NumPy Library

## What is Numerical Computing?

Numerical computing is an approach for solving complex mathematical problems using only simple arithmetic operations

There is a frequent need for processing large amounts of data in computational science applications. Storing data in lists and traversing lists with plain Python for loops leads to slow code, especially when compared with similar code in compiled languages such as Fortran, C, or C++. Fortunately, there is an extension of Python, commonly called Numerical Python, or abbreviated NumPy, which offers efficient array computations. **Numerical Python** has a fixed-size, homogeneous (fixed-type), multi-dimensional array type and lots of functions for various array operations. The result is a dynamically typed environment for array computing similar to basic Matlab. Usually, the speed of NumPy operations is quite close to what is obtained in pure Fortran, C, or C++.

### What is NumPy in Python?

NumPy is an open-source library available in Python, which helps in mathematical, scientific, engineering, and data science programming. It is a very useful library to perform mathematical and statistical operations in Python. It works perfectly for multi-dimensional arrays and matrix multiplication. It is easy to integrate with C/C++ and Fortran.

For any scientific project, NumPy is the tool to know. It has been built to work with the N-dimensional array, linear algebra, random number, Fourier transform, etc.

NumPy is a programming language that deals with multi-dimensional arrays and matrices. On top of the arrays and matrices, NumPy supports a large number of mathematical operations.

### Why use NumPy?

NumPy is memory efficiency, meaning it can handle the vast amount of data more accessible than any other library. Besides, NumPy is very convenient to work with, especially for matrix multiplication and reshaping. On top of that, NumPy is fast. In fact, TensorFlow and Scikit learn to use NumPy array to compute the matrix multiplication in the back end.

pip install numpy

import it in your applications by adding the import keyword:

import numpy

import numpy as np  
arr = np.array([1, 2, 3, 4, 5])  
print(arr)

**Checking NumPy Version**

The version string is stored under \_\_version\_\_ attribute.

import numpy as np  
print(np.\_\_version\_\_)

### Creating Arrays

NumPy is used to work with arrays. The array object in NumPy is called ndarray. We can create a NumPy ndarray object by using the array() function.

import numpy as np  
arr = np.array([1, 2, 3, 4, 5])  
print(arr)  
print(type(arr))

**Dimensions in Arrays**

A dimension in arrays is one level of array depth (nested arrays).

**0-D Arrays**

0-D arrays, or Scalars, are the elements in an array. Each value in an array is a 0-D array.

**arr = np.array(42)**

**1-D Arrays**

An array that has 0-D arrays as its elements is called uni-dimensional or 1-D array.

import numpy as np  
arr = np.array([1, 2, 3, 4, 5])  
print(arr)

**2-D Arrays**

An array that has 1-D arrays as its elements is called a 2-D array.

These are often used to represent matrix or 2nd order tensors.

**arr = np.array([[1, 2, 3], [4, 5, 6]])**

**3-D arrays**

An array that has 2-D arrays (matrices) as its elements is called 3-D array.

These are often used to represent a 3rd order tensor.

**arr = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])**

### Random Numbers using NumPy

To generate random numbers for Gaussian distribution, use:

**numpy.random.normal(loc, scale, size)**

Here,

* **Loc**: the mean. The center of distribution
* **Scale**: standard deviation.
* **Size**: number of returns

**Example:**

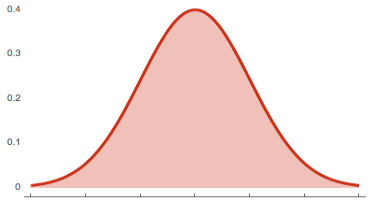
## Generate random nmber from normal distribution

normal\_array = np.random.normal(5, 0.5, 10)

print(normal\_array)

[5.56171852 4.84233558 4.65392767 4.946659 4.85165567 5.61211317 4.46704244 5.22675736 4.49888936 4.68731125]

If plotted the distribution will be similar to following plot



### Indexing and Slicing in Python

Slicing data is trivial with numpy. We will slice the matrice “e”. Note that, in Python, you need to use the brackets to return the rows or columns  
**Example:**

## Slice

import numpy as np

e = np.array([(1,2,3), (4,5,6)])

print(e)

[[1 2 3]

[4 5 6]]

Remember with numpy the first array/column starts at 0.

## First column

print('First row:', e[0])

## Second col

print('Second row:', e[1])

**Output:**

First row: [1 2 3]

Second row: [4 5 6]

In Python, like many other languages,

* The values before the comma stand for the rows
* The value on the rights stands for the columns.
* If you want to select a column, you need to add : before the column index.
* : means you want all the rows from the selected column.

print('Second column:', e[:,1])

Second column: [2 5]

To return the first two values of the second row. You use : to select all columns up to the second

## Second Row, two values

print(e[1, :2])

[4 5]

**Shape of an Array**

The shape of an array is the number of elements in each dimension.

import numpy as np

arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])

print(arr.shape)

**Output**

(2,4)

**What is numpy.zeros()?**

**numpy.zeros()** or np.zeros Python function is used to create a matrix full of zeroes. numpy.zeros() in Python can be used when you initialize the weights during the first iteration in TensorFlow and other statistic tasks.

**Syntax**

numpy.zeros(shape, dtype=float, order='C')

**Python numpy.zeros() Parameters**

Here,

* **Shape**: is the shape of the numpy zero array
* **Dtype**: is the datatype in numpy zeros. It is optional. The default value is float64
* **Order**: Default is C which is an essential row style for numpy.zeros() in Python.

**Example**

import numpy as np

np.zeros((2,2))

**Output:**

array([[0., 0.],

[0., 0.]])

**Example of numpy zero with Datatype**

import numpy as np

np.zeros((2,2), dtype=np.int16)

**Output:**

array([[0, 0],

[0, 0]], dtype=int16)

What is numpy.ones()?

**np.ones() function** is used to create a matrix full of ones. numpy.ones() in Python can be used when you initialize the weights during the first iteration in TensorFlow and other statistic tasks.

**Syntax**

numpy.ones(shape, dtype=float, order='C')

**Python numpy.ones() Parameters**

Here,

* **Shape**: is the shape of the np.ones [Python Array](https://www.guru99.com/python-arrays.html)
* **Dtype**: is the datatype in numpy ones. It is optional. The default value is float64
* **Order**: Default is C which is an essential row style.

**Python numpy.ones() 2D Array with Datatype Example**

import numpy as np

np.ones((1,2,3), dtype=np.int16)

**Output:**

array([[[1, 1, 1],

[1, 1, 1]]], dtype=int16)

**numpy.reshape() function in Python**

**Python NumPy Reshape** function is used to shape an array without changing its data. In some occasions, you may need to reshape the data from wide to long.

**Syntax**

numpy.reshape(a, newShape, order='C')

Here,

* **a**: Array that you want to reshape
* **newShape**: The new desires shape
* **Order**: Default is C which is an essential row style.

**Example**

import numpy as np

e = np.array([(1,2,3), (4,5,6)])

print(e)

e.reshape(3,2)

**Output:**

// Before reshape

[[1 2 3]

[4 5 6]]

//After Reshape

array([[1, 2],

[3, 4],

[5, 6]])

**numpy.flatten() in Python**

**Python NumPy Flatten** function is used to return a copy of the array in one-dimension. When you deal with some neural network like convnet, you need to flatten the array.

**Syntax**

numpy.flatten(order='C')

Here,  
**Order**: Default is C which is an essential row style.

**Example**

e.flatten()

**Output:**

array([1, 2, 3, 4, 5, 6])

### Statistical Functions in Python

NumPy has quite a few useful statistical functions for finding minimum, maximum, percentile standard deviation and variance, etc from the given elements in the array. The functions are explained as follows

Numpy is equipped with the robust statistical function as listed below

Consider the following Array:

**Example:**

import numpy as np

normal\_array = np.random.normal(5, 0.5, 10)

print(normal\_array)

**Output:**

[5.56171852 4.84233558 4.65392767 4.946659 4.85165567 5.61211317 4.46704244 5.22675736 4.49888936 4.68731125]

**Example of NumPy Statistical function**

### Min

print(np.min(normal\_array))

### Max

print(np.max(normal\_array))

### Mean

print(np.mean(normal\_array))

### Median

print(np.median(normal\_array))

### Sd

print(np.std(normal\_array))

**Output:**

4.467042435266913

5.612113171990201

4.934841002270593

4.846995625786663

0.3875019367395316

**What is numpy dot product?**

**Numpy.dot product**is a powerful library for matrix computation. For instance, you can compute the dot product with np.dot. Numpy.dot product is the dot product of a and b. numpy.dot () in Python handles the 2D arrays and perform matrix multiplications.

**Syntax:**

numpy.dot (x, y, out=None)

**Parameters**

Here,

* **x,y**: Input arrays. x and y both should be 1-D or 2-D for the np.dot() function to work
* **out**: This is the output argument for 1-D array scalar to be returned. Otherwise ndarray should be returned.

**Returns**

The function numpy.dot() in Python returns a Dot product of two arrays x and y. The dot() function returns a scalar if both x and y are 1-D; otherwise, it returns an array. If ‘out’ is given then it is returned.

**Raises**

Dot product in Python raises a ValueError exception if the last dimension of x does not have the same size as the second last dimension of y.

**Example:**

## Linear algebra

### Dot product: product of two arrays

f = np.array([1,2])

g = np.array([4,5])

### 1\*4+2\*5

np.dot(f, g)

**Output:**

14

### Matrix Multiplication in Python

The Numpy matmul() function is used to return the matrix product of 2 arrays. Here is how it works

1. 2-D arrays, it returns normal product
2. Dimensions > 2, the product is treated as a stack of matrix
3. 1-D array is first promoted to a matrix, and then the product is calculated

**Syntax:**

**numpy.matmul(x, y, out=None)**

Here,

x,y: Input arrays. scalars not allowed

out: This is optional parameter. Usually output is stored in ndarray

**Example:**

In the same way, you can compute matrices multiplication with np.matmul

### Matmul: matruc product of two arrays

h = [[1,2],[3,4]]

i = [[5,6],[7,8]]

### 1\*5+2\*7 = 19

np.matmul(h, i)

**Output:**

array([[19, 22],

[43, 50]])

**Determinant**

Last but not least, if you need to compute the determinant, you can use np.linalg.det(). Note that numpy takes care of the dimension.

**Example:**

## Determinant 2\*2 matrix

### 5\*8-7\*6np.linalg.det(i)

**Output:**

-2.000000000000005

References

1. <https://www.simplilearn.com/what-is-data-mining-article>
2. <https://expressanalytics.com/blog/what-is-data-wrangling-what-are-the-steps-in-data-wrangling/>
3. <https://analyticsindiamag.com/9-effective-pandas-techniques-in-python-for-data-manipulation/>
4. <https://monkeylearn.com/blog/data-preprocessing/>
5. <https://www.guru99.com/numpy-tutorial.html>