Applied Data Science - Notes

Dom Hutchinson

February 1, 2021

Contents

1 Introduction

Remark 1.1 - *Types of Data*
Data comes in many forms including, but not limited to, the following

- Dense & Sparse data.
- Structured/Relational Data.
- Numerical; Categorical; Ordinal; or Boolean.
- Test (Emails, Tweets, Articles).
- Records (User-Level Data, Timestamped Event Data, Log Files).
- Geo-Based Location Data.
- Data-Time Data.
- Network Data.
- Sensor Data.
- Images and Video.
- Audio and Music.

Remark 1.2 - *Big & Small Data*
Whether a dataset is big or small depends on the computational-resources available, and thus will vary over time. Here are some ways to evaluate this

	Big Data	**Small Data**
Data Condition	Always unstructured, not read for analysis, many relational database tables that need to be merged.	Ready for analysis, flat file, no need to merge tables.
Location	Cloud, offshore, SQL server etc.	Database, local PC.
Data Size	Over 50k variables, over 50k individuals, random samples, unstructured	File that is in a spreadsheet, that can be viewed on a few sheets of paper.
Data Purpose	No intended purpose.	Intended purpose for data collection.

Remark 1.3 - *What is Data Science?*
Data-Driven Science. An interdisciplinary field about scientific processes and systems to extract knowledge or insights from data in various forms.

Data science incorporates fields from: Mathematics, Computer Science; &, Domain Expertise.

Remark 1.4 - *Motivating Applications*
Data science is motivated by its applications. Here are some examples of such applications

- *Amazon* use recommender systems to suggest products to customers.
- *Energy Companies* use data science to try and predict future usage of customers, so that resources can be applied efficiently.

- *Agriculture* use sensors in fields to collect data in order to monitor crops and predict weather.
- *Healthcare* use sensors in homes to monitor the health of people over long periods of time (especially when the person cannot go to the hospital).

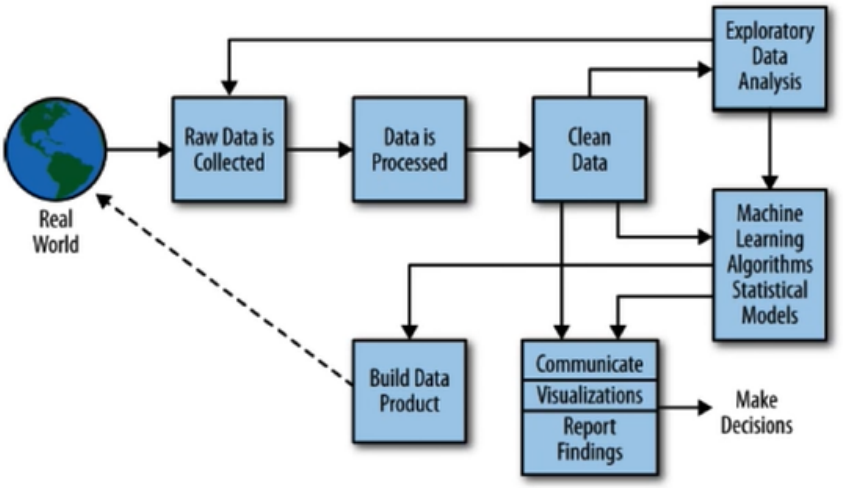

Figure 1: The pipeline for approaching problems in data science.

Proposition 1.1 - *Data Science Pipeline*
See `Figure 1` for a pipeline for approaching data science problems.

2 Data Ingress & Pre-Processing

2.1 Data Structures

Proposition 2.1 - *Native Python Data Structures*
Here are some data structures which are native to python and are popular in data science

- `list` - List of elements of varying types.
- `set` - List of unique elements of varying types.
- `dict` - Key-Value pairings.

Proposition 2.2 - *Non-Native Python Data Structures*
Here are some data structures which are not native to python but are popular in data science.

- `np.array`.
- `pandas.DataFrame`.

2.2 Data Formats

Definition 2.1 - *Object Persistence*
Object Persistence is the process of ensuring that the objects which are created are kept through multiple sessions. This comes in two stages

i). *Serialisation* - Translating data structures or objects from memory into a format which can be stored.

ii). *Deserialisation* - The inverse. Translating data structures which have been stored, into memory.

Remark 2.1 - *Bespoke Serialisation & Deserialisation*
Bespoke serialisation and deserialisation methods can be crafted manually. (e.g. Instantiating an output file; Writing each element of a list to a different line in the file; Closing the file.)

However, there are limitations to bespoke methods:

- Methods are specific to each use case (not standardised).
- Methods may not be robust.
- Methods require testing against many test cases.
- Object metadata is not encoded.

These limitations are rarely a problem in very controlled environments.

Definition 2.2 - *Comma-Separated Values (CSV)*
Comma-Separated Values (CSV) files are well suited to tabular data (inc. matrices). Each line of a *CSV File* stores one row of the table, and each element in a row is separated by a comma.

CSV Files are generally very readable, as well as time- and space-efficient for tabular data.

Remark 2.2 - *Using CSV Files*
As *CSV Files* are very popular, there are many library methods which can interact with them. Including reading & writing to and from memory (e.g. `pandas.read_csv`).

Definition 2.3 - *JavaScript Object Notation (JSON)*
JavaScript Object Notation[1] (JSON) is a standardised syntax for storing & exchanging data, used for serialisation. JSON uses text files which are human-readable and `dict`-like (i.e. have value-key pairs). JSON is designed to be simple so is a very robust language & suits many purposes.[2]

Limitations of JSON are that a specific conversion process may be required to convert a JSON file into objects in memory, and JSON files can become large due to key repetition.

Definition 2.4 - *Hierarchical Data Format (HDF5)*
Hierarchical Data Format (HDF5) is a standardised format for serialisation. HDF5 is a binary format and tries to mimic file system-like access. HDF5 files have the following three components

i). *Datasets* - Array-like collections of data. Thousands of datasets can be stored in a single file, and can be categorised and tagged however you want.

ii). *Groups* - Folder-like structures which groups datasets & other groups.

iii). *Metadata* - Information which pertains to all the datasets. (e.g. author, edit data & version).

[1] JSON is not just compatible with JavaScript and is common for many languages & APIs
[2] `https://jsonlint.com/` is a useful site for JSON validation.

HDF5 files are ideal for large numerical data sets, and can easily be manipulated by `numpy`. HDF5 files support a variety of transparent storage features (inc. compression, error-detection and chunked I/O), which `numpy.array` do not.[3]

Proposition 2.3 - *HDF5 files vs Traditional File Systems*
There are a few key differences between *HDF5 Files* and *Traditional File Systems*.

i). *HDF5 Files* are portable, as the entire structure is contained in the file independent of the underlying file system.[4]

ii). Datasets in *HDF5 Files* are all homogeneous hyper-rectangular numerical arrays, whereas files in traditional file system can be anything.

iii). Metadata can be added to groups in *HDF5 Files*. This is not possible in traditional file systems.

Remark 2.3 - *Other Standardised Serialisation Method*
XML, *Protocol Buffers* & YAML are other popular standardised serialisation methods.

2.3 Web-Scraping & APIs

Remark 2.4 - *Terms of Use*
Web Scraping should be done within the website's terms of use.

Definition 2.5 - *Web Scraping*
Web Scraping is the practice of collecting data from websites. This can take many forms, but typically involves taking a raw webpage and parsing the desired data.

Proposition 2.4 - *Approaching Web Scraping*
To perform *Web Scraping* successfully, you need a good idea of how a webpage is structure. Typically webpages are based around a `html` file, which are well structured[5]. Identifying combinations of tags, classes & ids in the `html` file can help locate the desired data.

It is harder, sometimes impossible, to navigate poorly designed websites as they are less structured and inconsistent.

Remark 2.5 - *Tools for Web Scraping*
Web Scraping technologies need to be tolerant to several artefacts of real-world data (known as *"wrangling"*) as-well-as errors in the website.

Some popular tools for *Web Scraping* are

- `BeautifulSoup` - A python library for parsing `XML` & `HTML`.
- `scrapy` - A python library. Generally faster than `BeautifulSoup`.
- `Selenium` - A web-browser plugin, generally used to test web services.

[3] `http://docs.h5py.org/en/latest/quick.html` provides a quick-start guide to using HDF5 files in python.
[4] However, it does depend on the HDF5 library.
[5] Webpages are often interpreted to have a tree structure, with each tag being a node

Definition 2.6 - *Web APIs*
Web APIs greatly simplify *Web Scraping* by provide a portal for explicit data acquisition, and are generally less prone to the issues which arise when *Web Scraping.*[6]

- The code running *Web APIs* is optimised for data requesting & retrieval. It does not waste time on visualisation or aesthetics. This means the bandwidth required for an *API* request is much lower than for a similiar *Web Scraping* process (As images etc. do not need to be loaded).
- *Web API* querying is robust, reliable, well maintained and documented with a static schema (`HTML`-based *Web Scraping* is not).
- *Web APIs* use standardised *Serialisation Tools* (e.g. JSON).
- *Web APIs* have already extracted and organised the desired data, however this does mean the user can only access what the operator will allow. This is much better than `HTML`-based *Web Scraping* where you rely upon fickle naming conventions of tags.

Definition 2.7 - *RESTful APIs*
Representational State Transfer APIs (REST/RESTful) are a popular form of *Web API* and generally require an *API Key* to access data.

Request to *RESTful APIs* generally involves constructing a URL which contains your keys and the parameters of your query.

Remark 2.6 - *Regular Expression (RegEx)*
Regular Expression (ReGex) queries are useful for extracting data, either while web scraping or from API requests.[7] Python has the `re` library for *RegEx* queries, two popular methods from this library are

i). `re.match` - Attempts to match a *RegEx* pattern to the whole string.

ii). `re.search` - Searches for the first occurrence of a *RegEx* pattern in a string.

3 Storage & Management

4 Transformation & Integration

5 Exploration & Visualisation

6 Deployment

7 Sharing & Privacy

[6] See `https://github.com/public-apis/public-apis` for a categorised list of public APIs.
[7] `http://pythex.org/` is a website which can perform *RegEx*.