MEME19403 Data Analytics and Visualisation: Basic Programming

Dr Liew How Hui

June 2021

Outline

Background

Data science

- becomes very popular lately because FAANG (Facebook, Amazon, Apple, Neflix, Google) and BAT? (Baidu, Alibaba, Tencent) are the richest companies because of their powerful ability in *big data* processing.
- because cloud computing companies are trying to trick the public into believing storing their data on the Internet is safe and purchasing the computing services is “not costly”.
- has a long history — related to the development of statistics.
- is not a panacea to an organisation.

Background (cont)

	Data Mining	Data Science
Data Sources	Structured Data from RDBMS	Unstructured Data (log files, audio, images, emails, tweets, raw text, documents)
Tools	Data Visualisation, Statistics	Machine Learning
Goals	Knowledge discovery, Decision making	New business value with new functionality (e.g. MS Teams for Online learning, etc.)

Background (cont)

What data engineers are saying:

- Data Mining: Data + Rules $\Rightarrow$ Output
- Data Science: Data + Output $\Rightarrow$ Rules

Actual history:

- Rules is the 'key': In the old days, rules are developed by experts ($$$); Today, IT companies say they have machine learning algorithms ($$$) to learn the 'rules' from the vast data.
- Reality: 'rules' are only useful for things that can be 'predicted' such as speech & image recognition but not things that cannot be 'predicted' such as stock price.

Various "Data" Programming Languages

- Java: Use in cloud computing
- Go
- R (1992, Ross Ihaka and Robert Gentleman) and S-PLUS (1988, commercial S)
- COBOL? : designed in 1959 by CODASYL
- SQL: by IBM in early 1970s
- JavaScript?
- Julia?
- Python: Everyone is telling me that the industry wants this.

Software

Python Software Environments:

- Windows: Anaconda Python 3.8 (is 3.9 ready?)
- Linux / MacOS: Python $\geq$ 3.6 + Add-on packages
- Python Shell + Text Editor: VS Code, Atom, Vim, etc.
- Spyder: IDE
- Jupyter Notebook: I don't recommend

Outline

(Data) Programming (with Python)

The ‘steps’ in data programming:

1. Representing “data” in computer.
   - Basic data types: Integers (Not use), Floating-point numbers, Booleans, Strings
   - Containers: Tuple, List, Dictionary (Hash), Set
   - Numpy array, Pandas Data Frame
2. Exploratory Data Analysis & Dashboard
3. Predictive Modelling
4. Reporting and Presentation

Programming (cont)

Basic data type — Integer

- ..., -3, -2, -1, 0, 1, 2, 3, ... (can be arbitralily large)
- Real data has limited range: e.g. 64-bit:
  $-2^{63} = -9223372036854775808$ to
  $2^{63} - 1 = 9223372036854775808$
- +, -, *, /, //, **, %
- >, >=, ==, ! =, <=, <

Programming (cont)

Basic data type — Floating-point Number

- Scientific notation: $\pm$ X.DDDDD $\times 10^{y}$
- Precision: 32-bit (float), 64-bit (double)
- Special purpose: 16-bit (used by Google in Machine Learning), 128-bit (???)
- +, -, *, /, **
- >, >=, ==, ! =, <=, <
- math library

Programming (cont)

Basic data type — Boolean

- True, False
- Not really used in data science???
- and, or, not
- if statement, while loop

Programming (cont)

Basic data type — String

- Text strings: ”a string”, ’a string’
- For IO and “categorical data”
- +, len()
- Beware of encoding problems: UTF-8, ISO 8859-1, ..., ISO 8859-9, GBK, etc.
- Byte strings

Programming (cont)

Non-basic data types: Form from basic data types

- Dates and Time: A mixed of “integers” from different fields.
- Records: Entries containing a mixed of the basic data types (and non-basic data types). For a university, it consists of the fields Full-time/Part-time (Boolean), Student ID (integer), Name (string), results (Floating point numbers), etc.
- Table: A collection of records stuctured properly. E.g. Excel Table

Containers

Tuple: (a,b,c)

Can put any basic data type and containers.

Application:

- Simple assignment: a,b,c = 1,2,3
- Encoding error: (result, error)
- Construct array of certain shape: np.zeros((2,2,3))

Containers (cont)

List : super-powerful but slow

Can put any basic data type and containers.

Application:

- Store arbitrary items: [”string”, 1, 3.14, True]
- Programming: [], append, len
- Construct Numpy array
- Construct tree?

Containers (cont)

Dictionary / Associative Map: For lookup a value based on a key.

Application:

- Key-value pair: {key : value}, mydict[”word”] = 1
- Some NoSQL database are based on the generalisation of “dictionary” — key-value pair based Database.
- Count items, e.g. mydict[”word”] += 1
- Construct data frame

Containers (cont)

Set: works like finite set in maths

Application:

- Find $A \cap B$, e.g. A=Customers buying A, B=Customers buying B

Outline

Unstructured Data

Examples:

- text (variable length)
- images
- documents: HTML, XML, SGML, Word, PDF, Excel Reports, Powerpoints, etc.

Computer Representations:

- Document-oriented database
- Key-Value pair database
- Graph database

Outline

Structured Data

Examples of semi-structured data:

- Company CSVs, Excels Files (.xls, .xlsx, .xlsb, .xlsm, .xltx, .xltm)
- SQL database

Computer Representations:

- Numpy Array: Homegeneous rectangular data.
- Data Frame: Python's pd.DataFrame (11 Jan 2008), R's data.frame (1991)
- SQL Database: Data definition (schema)

Structured Data (cont)

Python libraries that work with (semi-)structured data:

```
import numpy as np
import openpyxl # read Excel 2010 files
import xlsxwriter # write Excel 2010 files
import xlrd, xlwt # read/write Excel .xls files
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels as sm
```

Structured Data (cont)

Python libraries to read CSV file

```
1 fn="world-population-by-world-regions-post-1820"
2 fn="data/" + fn + ".csv"
3 pop_data = pd.read_csv(fn)
4 print(pop_data.head(4))
5 print(pop_data.tail(4))
6 print(pop_data.index)
7 print(pop_data.columns)
```

Structured Data (cont)

HTML with Tables:

```
<html>
<body>
Table 1
<table>
  <tr><th>Firstname</th><th>Lastname</th><th>Age</th><th>Salary</th></tr>
  <tr><td>Jill</td><td>Smith</td><td>35</td><td>3,000--6,000</td></tr>
  <tr><td>Eve</td><td>Jackson</td><td>?</td><td>6,000--9,000</td></tr>
</table>

Table 2
<table>
  <tr><th>Name</th><th colspan="2">Telephone</th></tr>
  <tr><td>Bill Gates</td><td>55577854</td><td>55577855</td></tr>
</table>
</body>
</html>
```

Structured Data (cont)

Python libraries to read HTML Tables:

```
import pandas as pd
tbls = pd.read_html("tableeg.html")   # requires lxml
for i,tbl in enumerate(tbls):
    print("="*8 + f"Table {i+1}" + "="*8)
    print(tbl)
```

Structured Data (cont)

JSON = JavaScript Object Notation

Sample 1:

```
{
    "Firstname" :    ["Jill", "Eve"],
    "Lastname" :     ["Smith", "Jackson"],
    "Age" :          [35, "?"],
    "Salary" :       ["3,000--6,000",
                      "6,000--9,000"]
}
```

Structured Data (cont)

Sample 2:

```
[
{
    "Firstname" :    "Jill",
    "Lastname" :     "Smith, II",
    "Age" :          35,
    "Salary" :       "3,000--6,000"
},
{
    "Firstname" :    "Eve",
    "Lastname" :     "Jackson",
    "Age" :          "?",
    "Salary" :       "6,000--9,000"
}
]
```

Structured Data (cont)

Pandas library supports JSON data:

```
import pandas as pd
df1 = pd.read_json("sample1.json")
print(df1)
df2 = pd.read_json("sample2.json")
print(df2)
```

Structured Data (cont)

SQLite database: Use sqlite & SQL language

or

Python sqlite3 library to read sqlite database

```
from pandas.io import sql
import sqlite3
conn = sqlite3.connect('data.db')
query = 'SELECT * FROM tablename'
tbl = sql.read_sql(query, con=conn,
                   parse_dates={'date': '%d/%m/%Y'})
print(tbl.head())
```

Structured Data (cont)

SQLite Database only supports 5 basic data types:

- NULL. The value is a NULL value.
- BLOB. The value is a blob of data, stored exactly as it was input.
- TEXT. The value is a text string, stored using the database encoding (UTF-8, UTF-16BE or UTF-16LE).
- INTEGER. The value is a signed integer, stored in 1, 2, 3, 4, 6, or 8 bytes depending on the magnitude of the value.
- REAL. The value is a floating point value, stored as an 8-byte IEEE floating point number.

Structured Data (cont)

Other formats:

- SPSS (requires special API from IBM),
- SAS (?)
- Advanced / Commercial SQL Database (requires authentication)
- NoSQL Database (Datalog Query? SPARQL?)
- SAP (?)

Structured Data (cont)

Advanced / Commercial SQL Database supports more data types. E.g. Postgresql (see `http://www.postgresqltutorial.com/postgresql-data-types/`) has richer data structures:

- Boolean;
- Character types such as char, varchar, and text;
- Numeric types such as integer and floating-point number;
- Temporal types such as date, time, timestamp, and interval;
- UUID for storing Universally Unique Identifiers;
- Array for storing array strings, numbers, etc.;
- JSON stores JSON data;
- hstore stores key-value pair;
- Special types such as network address and geometric data.

Structured Data (cont)

Numpy and Panda datas:

- dtype / pd.DataFrame().dtypes
  - b boolean
  - i signed integer
  - u unsigned integer
  - f floating-point
  - c complex floating-point
  - m timedelta
  - M datetime
  - O object
  - S (byte-)string
  - U Unicode
  - V void
- Array integer (32 bits, 64 bits) is not the same as Python integer
- Floating point numbers 32 bits and 64 bits

Structured Data (cont)

```
1  df = pd.DataFrame({
2     'a' : [1,2,3],
3     'b' : [4,5,6],
4     'c' : [7,8,9],
5  })   # default index = 0,1,2,...
6
7  df = df.rename(columns={'a': 'A', 'b' : 'B'})
8
9  # Reshaping: df.pivot, pd.concat([df1,df2]) # axi
```

Outline

Assignment

Submit report at the end of Week 11

Malaysia is embracing the Industrial Revolution 4.0 to upgrade our industrial production to meet international competition, to adopt Green Technology and to make sure that environments are protected and the production is sustainable.

Predictive maintenance is an essential part of Industrial Revolution 4.0.

Assignment

In this assignment, we will analysis data from the UCI Machine Learning repository — AI4I 2020 Predictive Maintenance Dataset (`https://archive.ics.uci.edu/ml/datasets/AI4I+2020+Predictive+Maintenance+Dataset`).

- Find out and write down the background and usefulness of the data
- Write down the descriptive statistics you will carry out for the data
- Write down the application of the analysis
- Develop a predictive model (or predictive models) for the data and study the performance of the predictive model on the data.

Assignment (cont)

The assignment report (20%) should consist of **four** sections:

1. Objective of Study . . . . . (2 marks)
2. Exploratory Data Analytics . . . . . (5 marks)
3. Feature Engineering . . . . . (4 marks)
4. Predictive modelling . . . . . (9 marks)

Oral presentation (5%) in Week 12.

Week 10: Test (25%). Covering Week 7 to Week 9 (PCA) — Python Programming with Pandas & Matplotlib