# DATA SCIENCE & MACHINE LEARNING Introduction Course

(23 JAN - 01 March 2019)

**Coordination Team:**

Augusto Albuquerque (ISCTE-IUL)
Inês Cortez Esteves (Deloitte)
Pedro Sebastião (ISCTE-IUL/AUDAX-ISCTE)
Pedro Tavares (Deloitte)

**Team:**
António Raimundo (ISCTE-IUL)
Dária Baikova (ISCTE-IUL)
João Oliveira (ISCTE-IUL)
Ricardo Ribeiro (ISCTE-IUL)
Sérgio Moro (ISCTE-IUL)

ISCTE IUL Instituto Universitário de Lisboa AUDAX ISCTE Entrepreneurship & Innovation Deloitte.

# DATA SCIENCE & MACHINE LEARNING Introduction Course

## General Contents (23 JAN 2019)

- **#0 Introduction – Overview (23 JAN 2019)**
- **#1 IPython – Beyond Normal Python (23 JAN 2019)**
- #2 Introduction to NumPy
- #3 Data Manipulation and Visualization
- #4 Statistical Analysis
- #5 Time-Series and High Perfomance Pandas
- #6 Introduction to Machine Leaning
- #7 Machine Learning Techniques
- #8 Deep Learning
- #9 Network Analysis Social Networks
- #10 Recommendation Systems using NLP

# DATA SCIENCE & MACHINE LEARNING
# #0 Introduction – Overview

(23 JAN 2019)

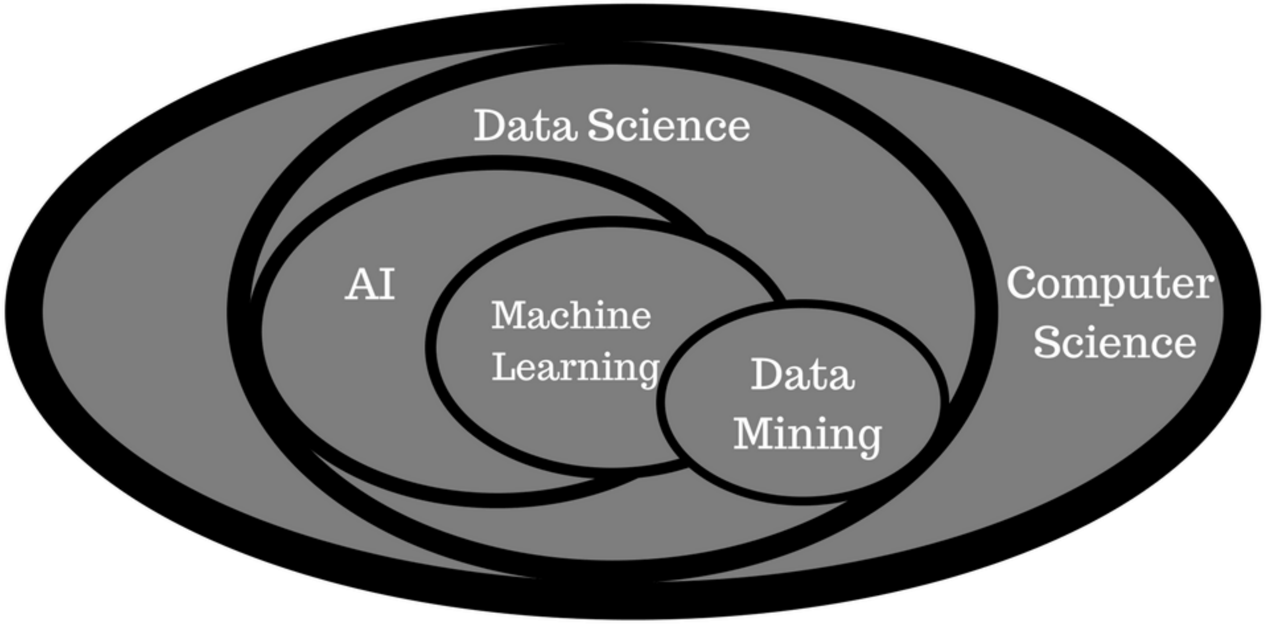
Lecturers: Sérgio Moro & António Raimundo

# #0 Introduction – Overview

## Contents (23 JAN 2019)

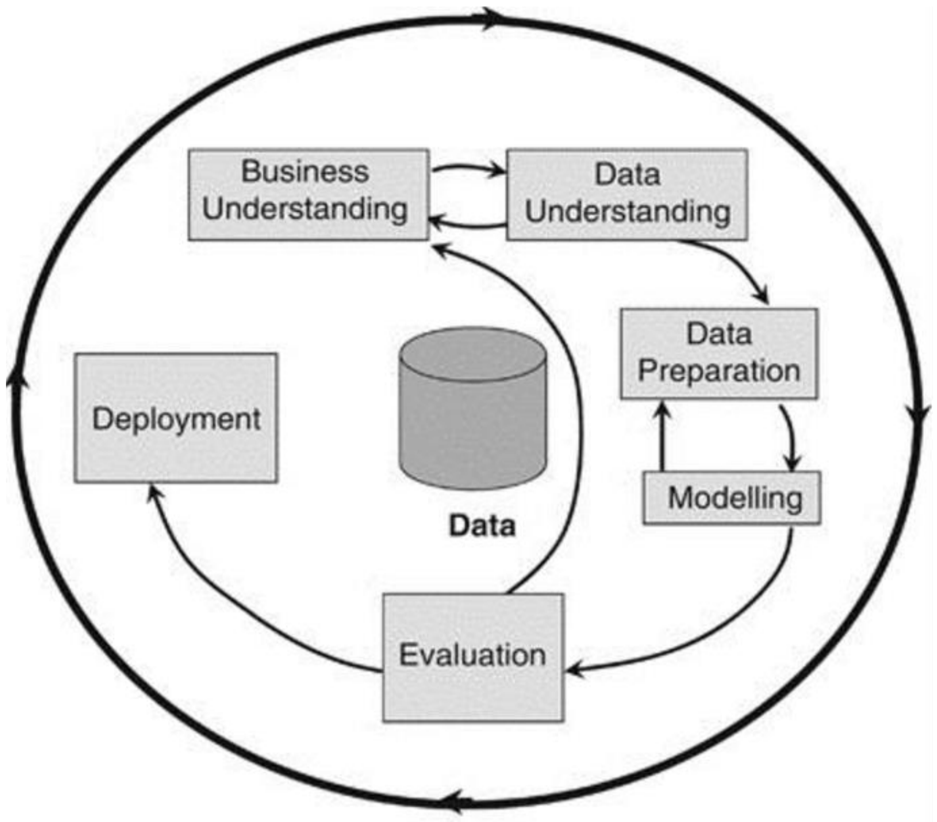
- Introduction to Data Science
- CRISP-DM
- Practical example: CRM
- Practical example: Service Desk System

## #0 Introduction – Overview



What is Data Science?
**Interdisciplinary data-driven** approach to address problems for which there is **available data**.

## Data Mining CRISP-DM Methodology



**<u>Data Mining</u>: knowledge discovery** process that aims to unveil **insightful patterns** from **raw data** (it is encompassed within Data Science)

**<u>Text Mining</u>**: Unstructured data

*Note: Machine Learning (ML) aims to learn from data.*
*Thus, ML may (and it is often) used for building models within a DM project.*

## Data Mining CRISP-DM Methodology

**Business Understanding**

- Hear from business experts:
  - How do they define the problem?
  - What are the factors that may influence the result?
  - How would they measure the success of a possible solution?
- Create a project
  - Goals
  - Risks
  - Stakeholders
  - Plan

## Data Mining CRISP-DM Methodology

### Data Understanding

- What are the variables that best characterize the problem (input variables)? What is their range of possible values and type (e.g., categorical vs. numeric)?
- From these, which best translate the business goal (for supervised learning)?
- Are there data quality issues?
    - Inconsistent data originated in more than one source
    - Missing values

## Data Mining CRISP-DM Methodology

### Data Preparation

- Data integration: consolidate data from distinct sources
- Feature selection: which input variables?
- Feature Engineering: computing new interesting variables
- Data cleaning: deal with missing values, outliers

## Data Mining CRISP-DM Methodology

**Modeling**

- Uses Statistics or Machine Learning algorithms to train a model
- It depends on the goal:
    - Unsupervised learning – find relations between input variables
    - Supervised learning – use a target (dependent) variable to build a model based on input variables
        - If the target is categorical (e.g., grant loan: Y/N), then it is a classification problem; else (for numeric variables, e.g., how many products sold), it is a regression problem

## Data Mining CRISP-DM Methodology

**Evaluation**

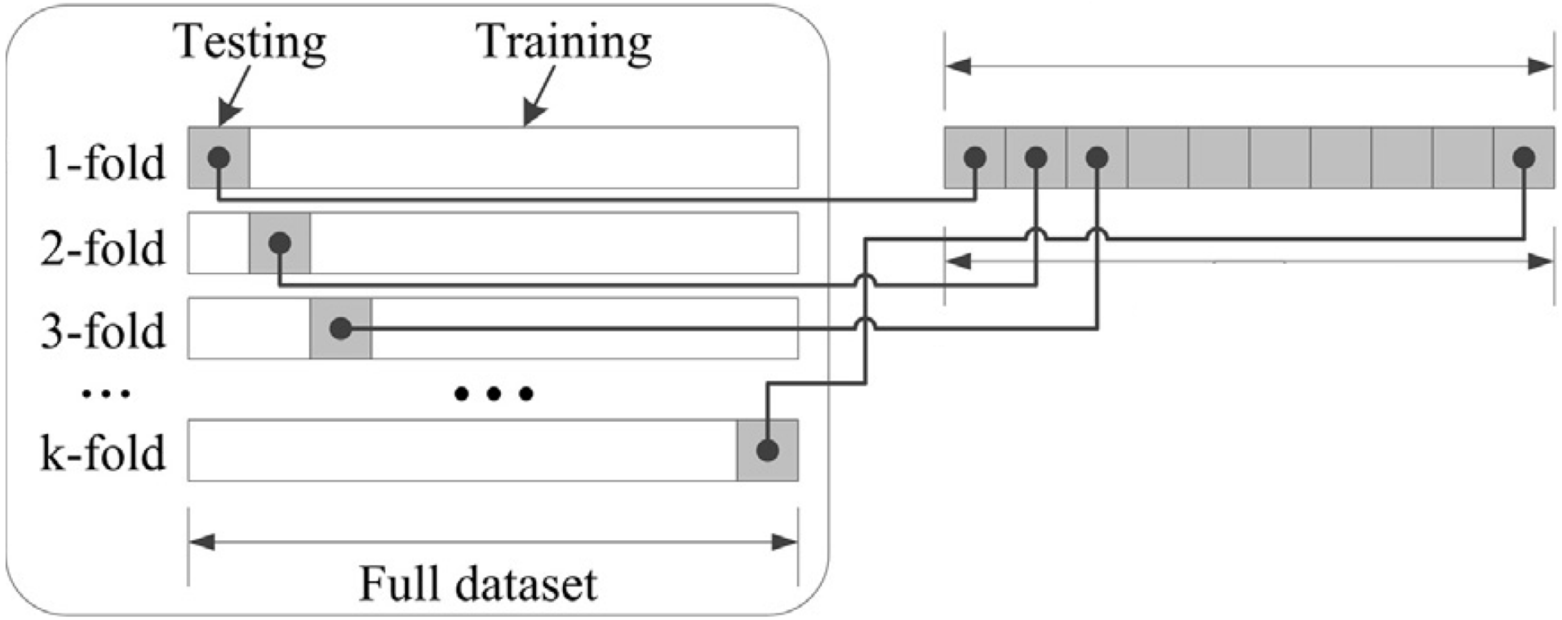
- Assess if the model is accurate
- For supervised learning models: how close are the predictions from the real values?
- For unsupervised learning: how consistent are the identified groups?

- In a production scenario: a model is trained using past data, and it is evaluated using future data
- During project (or for a quick model tuning): partition the dataset into train (to build the model )and test sets

## Data Mining CRISP-DM Methodology

### Evaluation

- K-fold cross validation: each fold is used once for testing and K-1 times for training



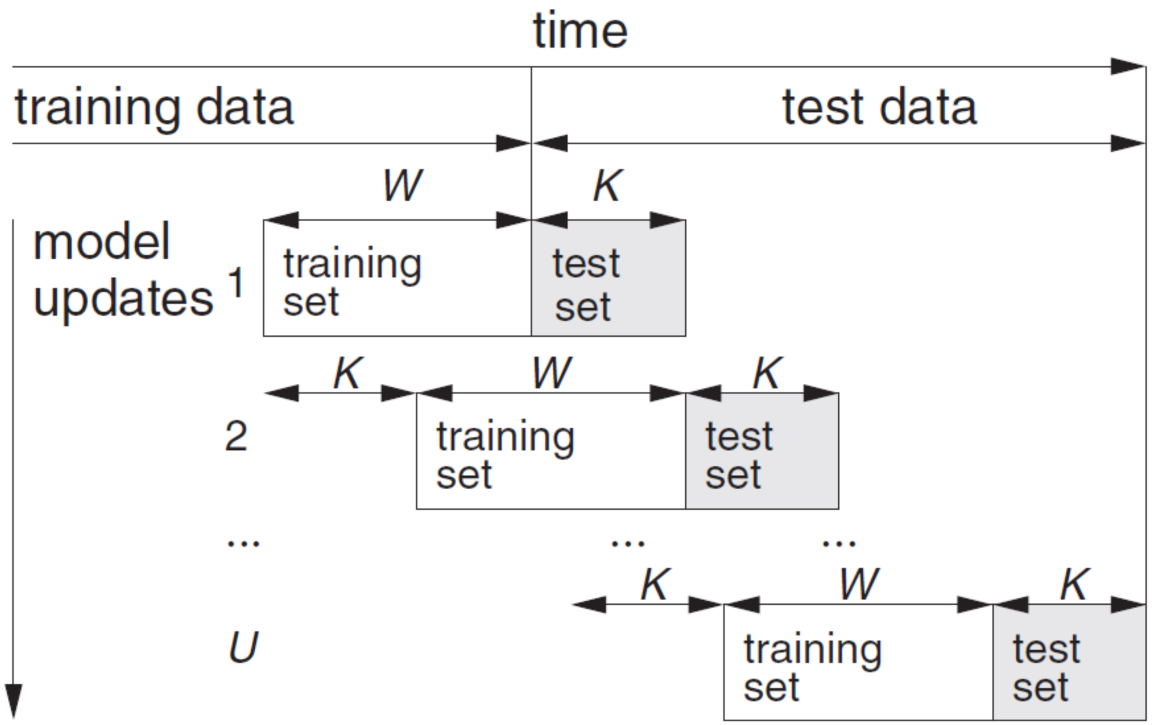
## Data Mining CRISP-DM Methodology

### Evaluation

- Rolling Windows: to simulate a periodic train/test scenario

All occurrences of the problem (also called instances) must be sorted by date

This is what will happen once the model is deployed into production

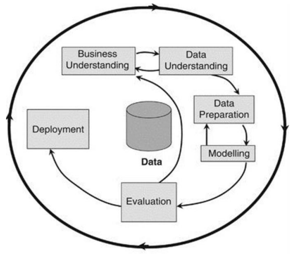


# Data Mining CRISP-DM Methodology

## Deployment

- Install the model into a production environment

- Yet, the work isn't over:
  - Real problems are dynamic and the model's performance can quickly deteriorate
  - New variables may need to be incorporated, or some removed
  - Out-of-the-ordinary events may induce unexpected behavior
- **Thus, tuning should be done periodically**
- CRISP-DM is a cyclic methodology that advocates revision iterations

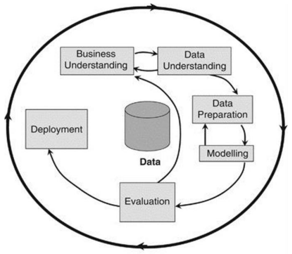


# Data Mining CRISP-DM Methodology

## Deployment

- Install the model into a production environment

- Yet, the work isn't over:
    - Real problems are dynamic and the model's performance can quickly deteriorate
    - New variables may need to be incorporated, or some removed
    - Out-of-the-ordinary events may induce unexpected behavior
- **Thus, tuning should be done periodically**
- CRISP-DM is a cyclic methodology that advocates revision iterations

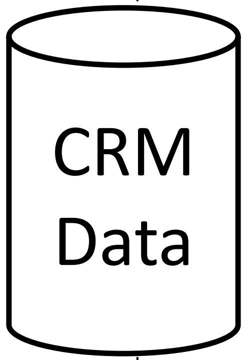


## Customer Relationship Management (CRM)

Back Office
***Analytical CRM***

Front Office
***Operational CRM***

Usual questions:
- Which customers to contact?
- Which products to sell?



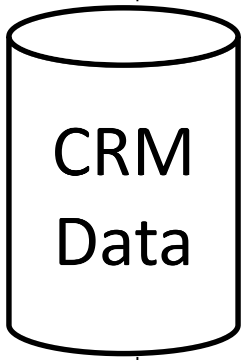
Usual operations:
- Contact customer
- Sell product

Customer

## Customer Relationship Management (CRM)

Back Office
***Analytical CRM***

How to answer usual questions:
- Through implemented Data Mining models



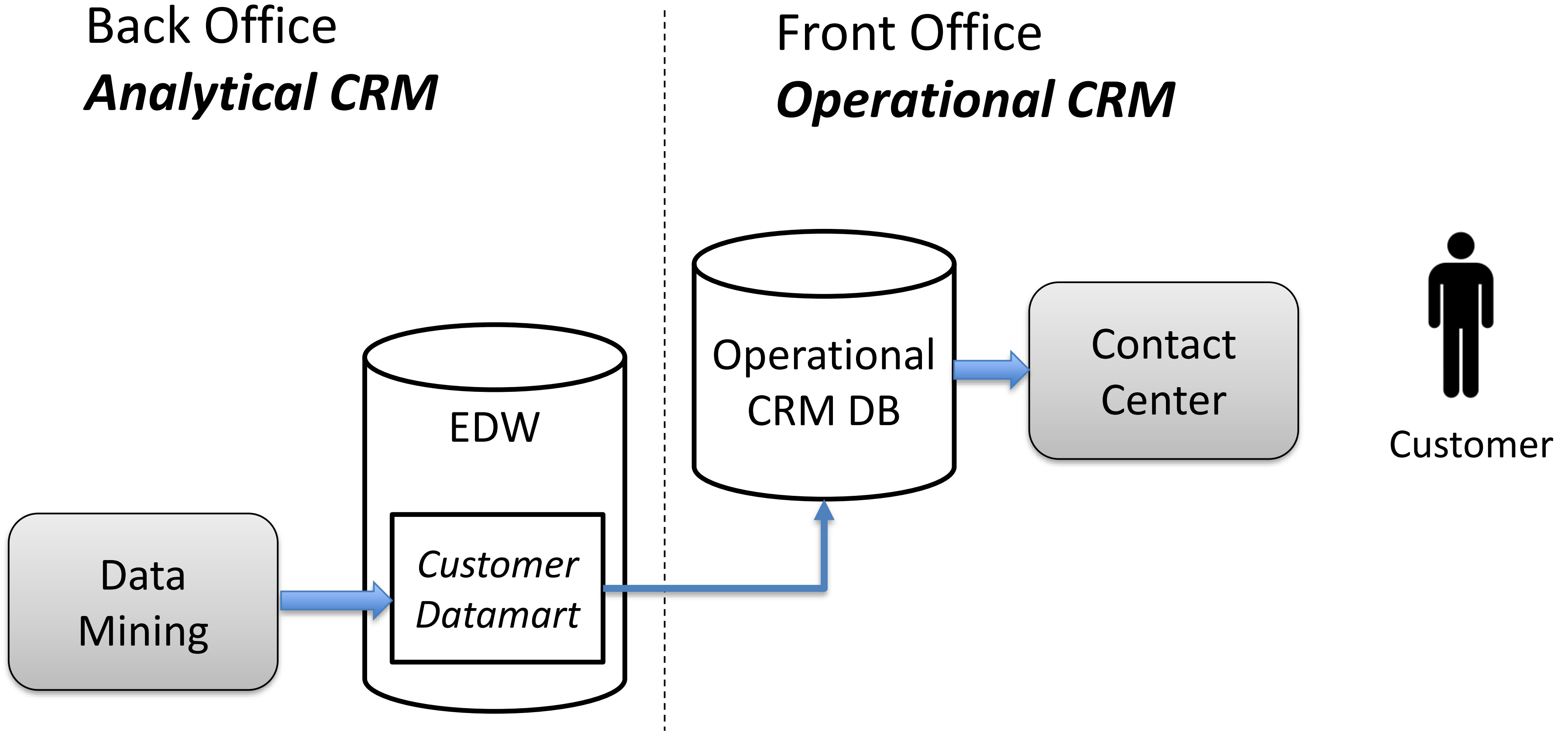
Front Office
***Operational CRM***

How to make usual operations:
- Through operational applications (e.g., contact center)

Customer

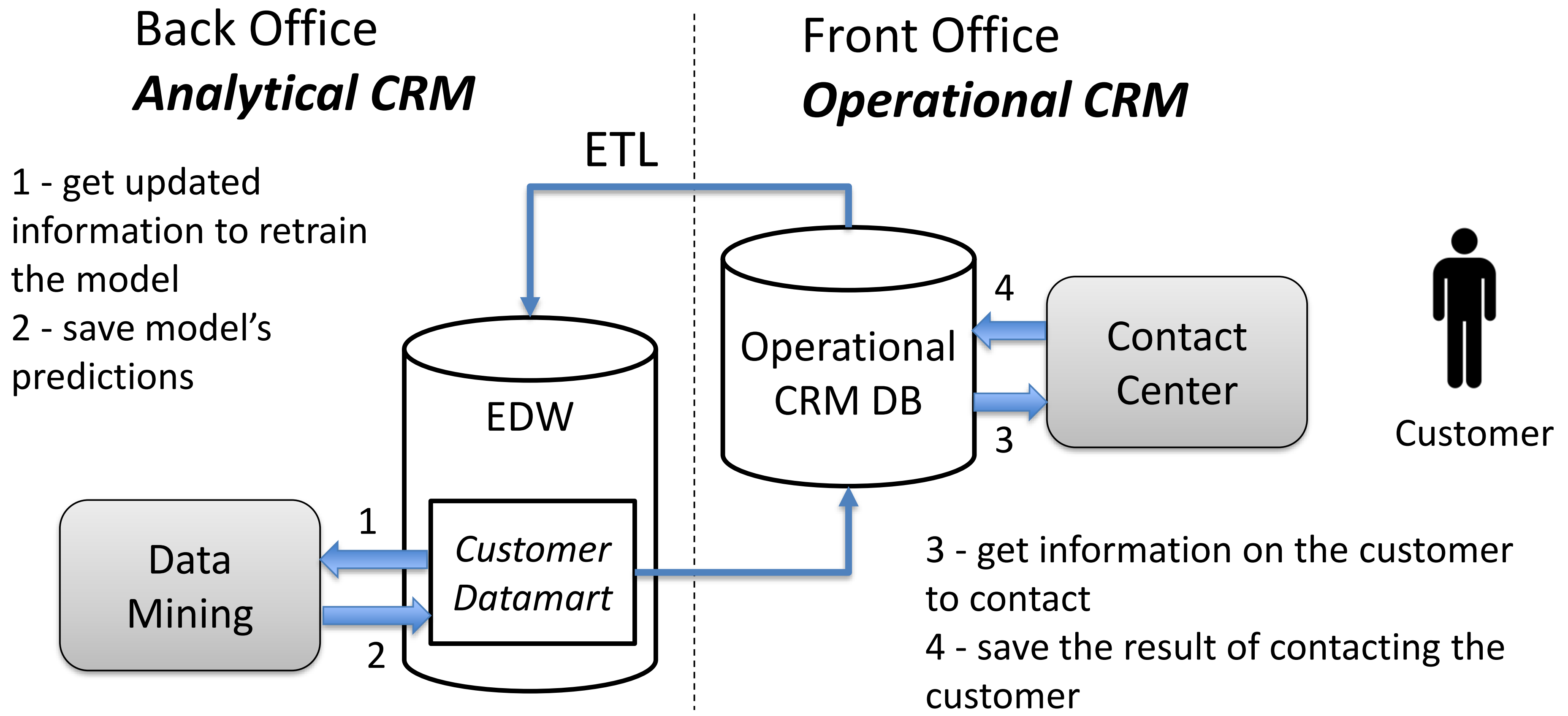
## Customer Relationship Management (CRM)

*Which customers to contact for selling a product?*



# Customer Relationship Management (CRM)

*Closing the loop…*

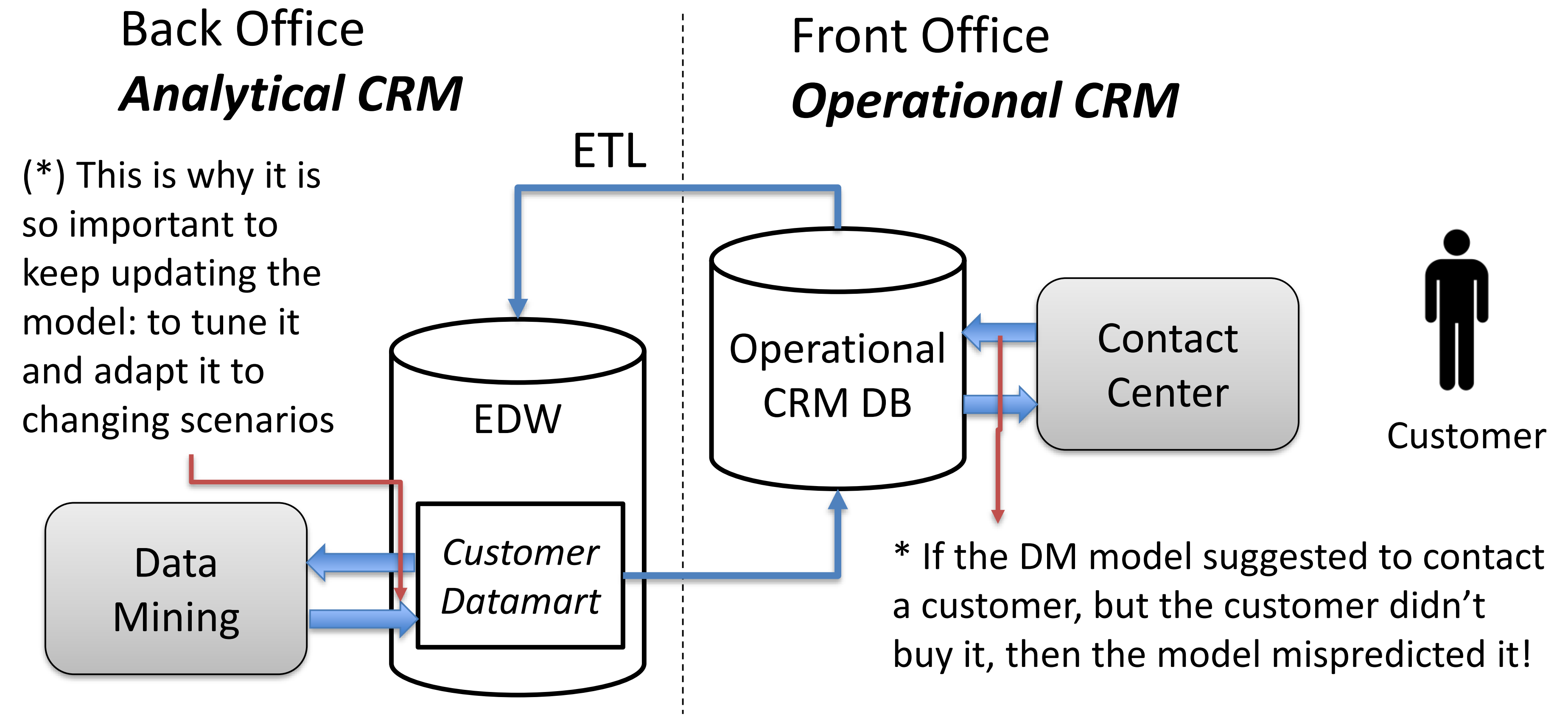


1 - get updated information to retrain the model
2 - save model's predictions

3 - get information on the customer to contact
4 - save the result of contacting the customer

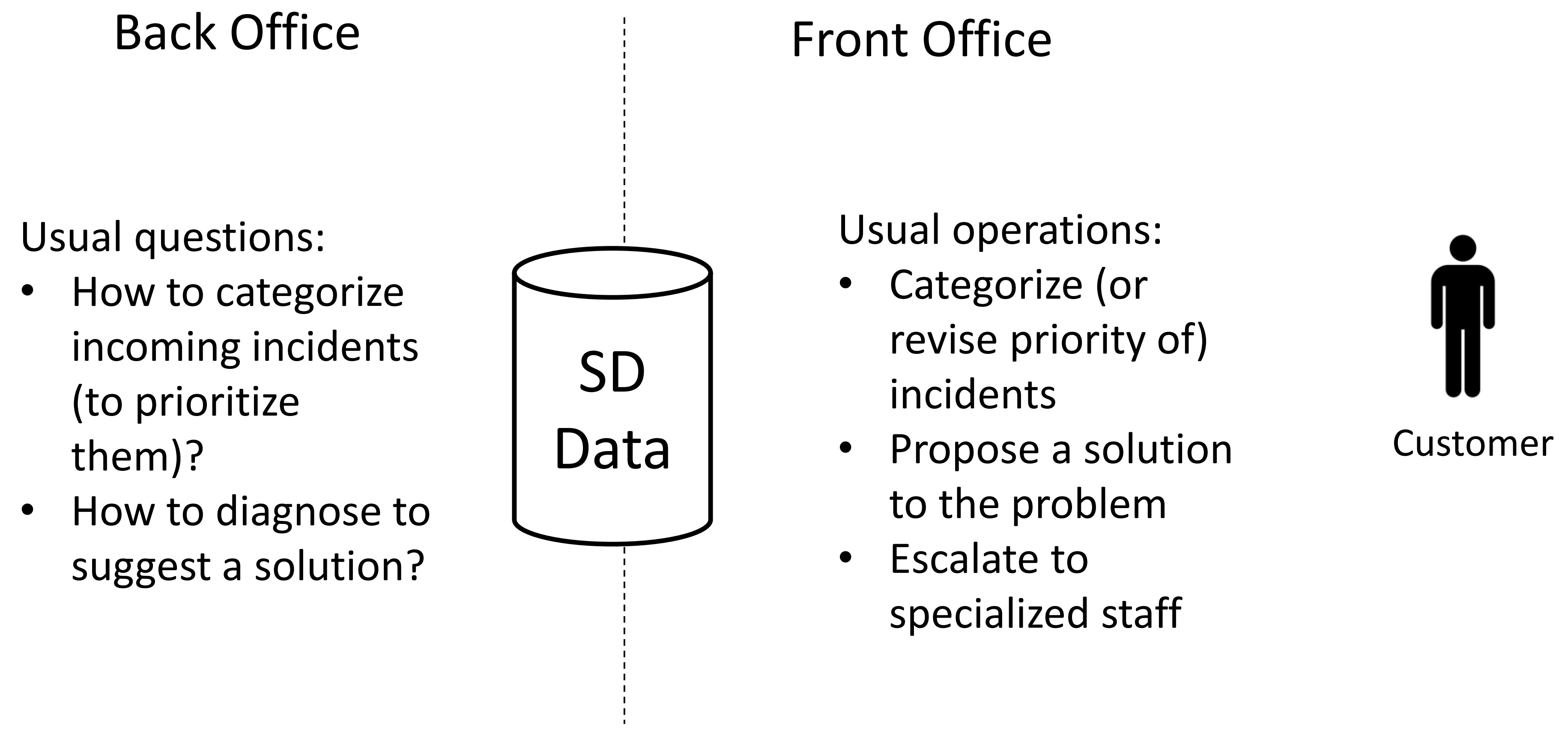
# Customer Relationship Management (CRM)

*Which customers to contact for selling a product?*


Back Office
*Analytical CRM*
Front Office
*Operational CRM*
ETL
(*) This is why it is so important to keep updating the model: to tune it and adapt it to changing scenarios
EDW
*Customer Datamart*
Data Mining
Operational CRM DB
Contact Center
Customer
* If the DM model suggested to contact a customer, but the customer didn't buy it, then the model mispredicted it!

## Service Desk System (SD)

*Incident management*



Usual questions:

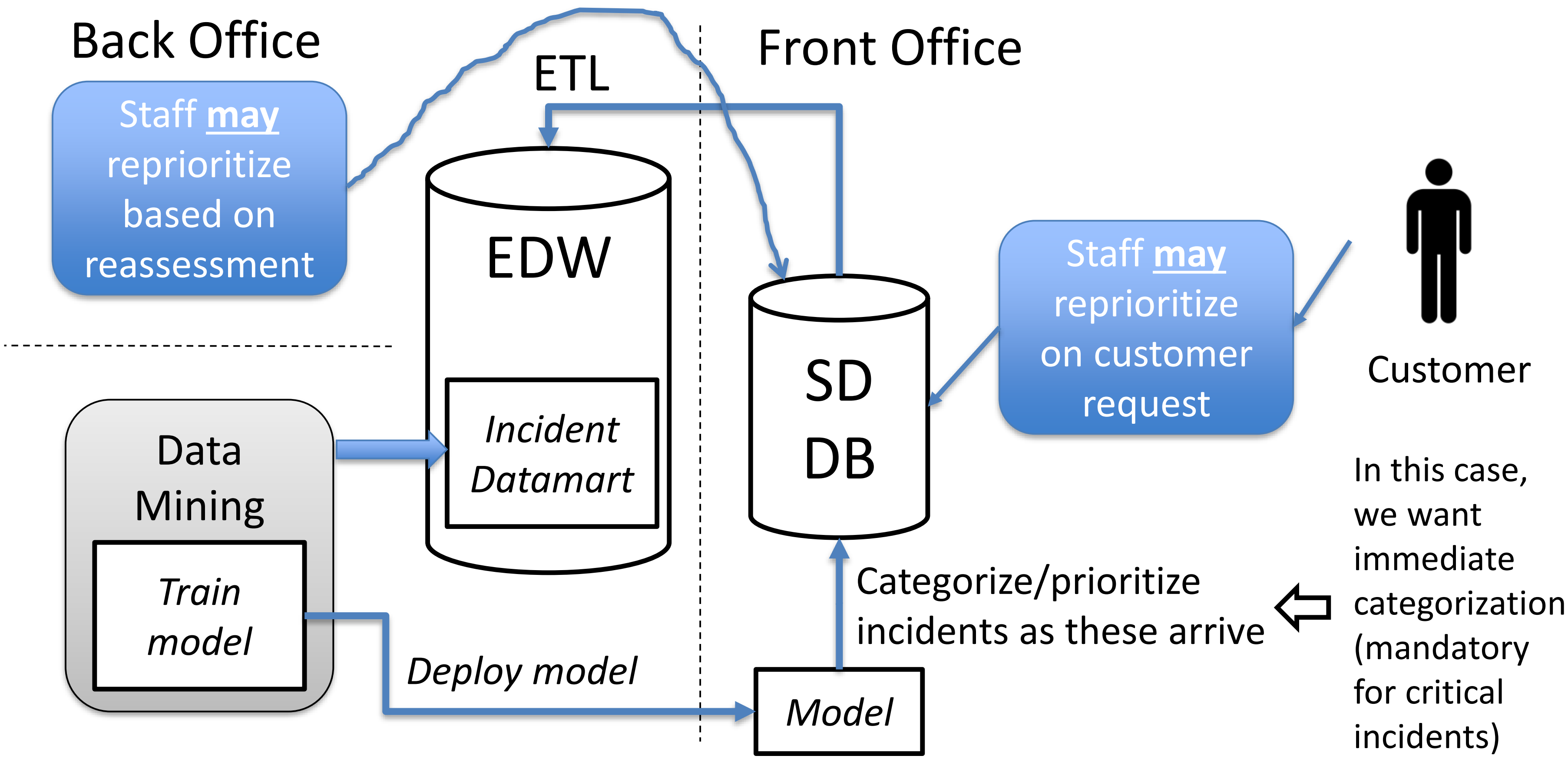
- How to categorize incoming incidents (to prioritize them)?
- How to diagnose to suggest a solution?

Usual operations:

- Categorize (or revise priority of) incidents
- Propose a solution to the problem
- Escalate to specialized staff

# Service Desk System (SD)

*Incident management – **Automatic categorization***


Back Office
Staff may reprioritize based on reassessment
ETL
Front Office
EDW
Incident Datamart
SD DB
Staff may reprioritize on customer request
Customer
Data Mining
Train model
Deploy model
Model
Categorize/prioritize incidents as these arrive
In this case, we want immediate categorization (mandatory for critical incidents)

## THANK YOU FOR YOUR PARTICIPATION

## #0 Introduction – Overview

(23 JAN 2019)

## Next module is:

**#1 IPython – Beyond Normal Python**

(23 JAN 2019)

# DATA SCIENCE & MACHINE LEARNING
# #1 IPython – Beyond Normal Python

(23 JAN 2019)

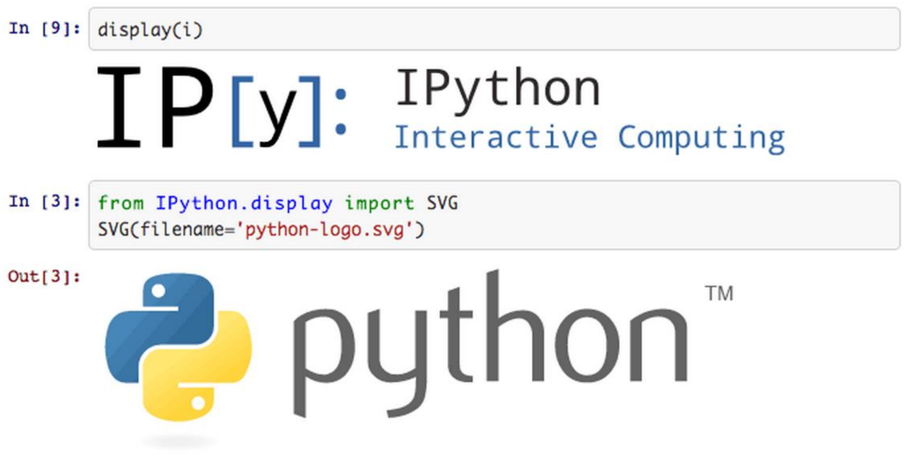
Lecturers: Sérgio Moro & António Raimundo

## #1 IPython – Beyond Normal Python

### Contents (23 JAN 2019)

- Shell or Notebook
- Help and Documentation in IPython
- Magic Commands
- Input and Output History
- Notebook and Shell Commands
- Errors and Debugging
- Profiling and Timing Code

- **What is IPython?**
  - Features
- **Ties with Jupyter project**
  - Notebook
- **Shell or Notebook?**



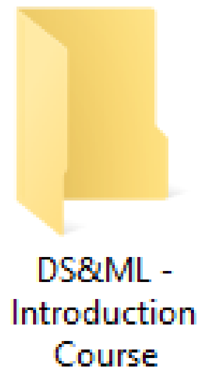
```
In [9]: display(i)
```

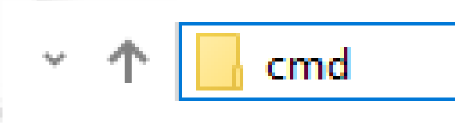
```
In [3]: from IPython.display import SVG
        SVG(filename='python-logo.svg')
Out[3]:
```

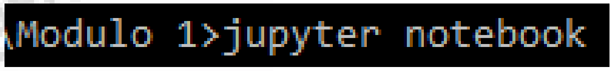
We will work all the code examples on Jupyter Notebook!

## Jupyter Notebook

- **Launch Jupyter Notebook**
  - Create a folder on a desired location on your PC named: “**DS&ML – Introduction Course**”
  - Create a subfolder called “**Module 1**”
    - You can create subfolders later for each module, according to the ongoing lectured module (2,3,4 and so on)
  - (*Windows users*)
    - Enter “**Module 1**” folder, and type “**cmd**” on ‘*current location*’ bar
    - On the command line terminal, type: ‘**jupyter notebook**’
  - (*Linux/MacOS users*)
    - Open Terminal and type: ‘**jupyter notebook**’
    - Navigate to “**Module 1**” folder using Notebook’s explorer interface.





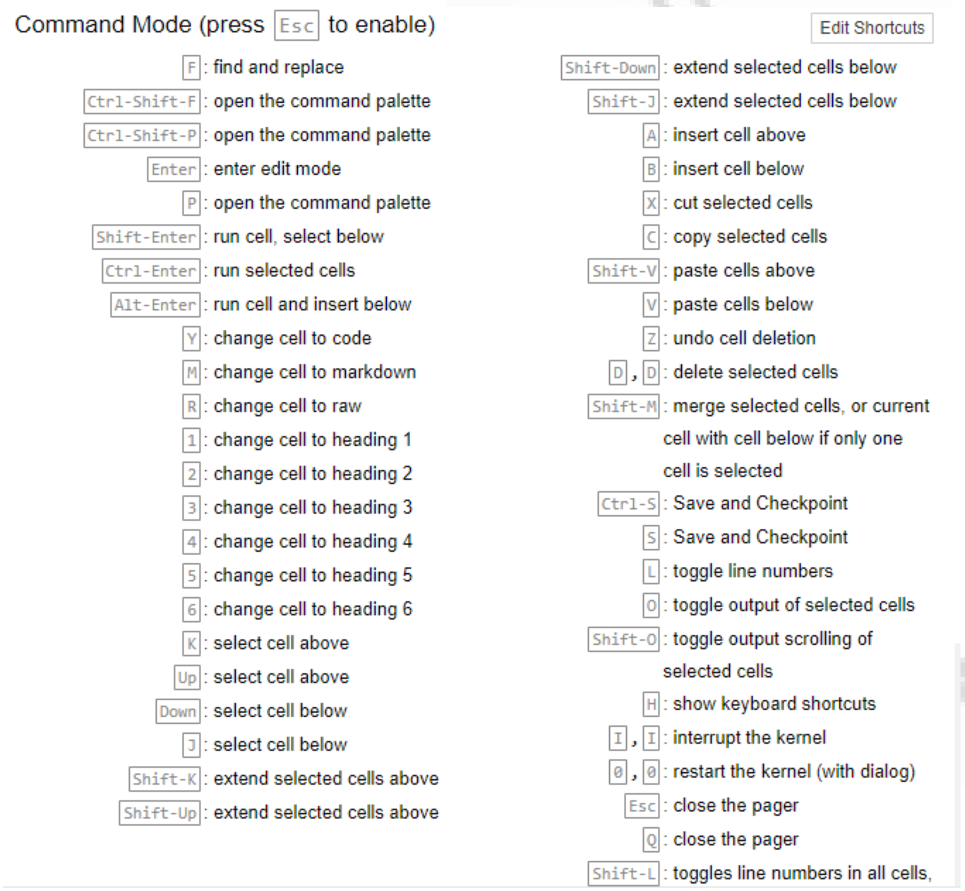


```
\Modulo 1>jupyter notebook
```

- **#1.1 Help and Documentation in IPython**
  - Accessing Documentation with ‘**?**’
  - Accessing Source Code with ‘**??**’
  - Exploring Modules with **Tab-Completion**
    - Tab-completion of object contents
    - Tab completion when importing

## #1.2 Keyboard Shortcuts

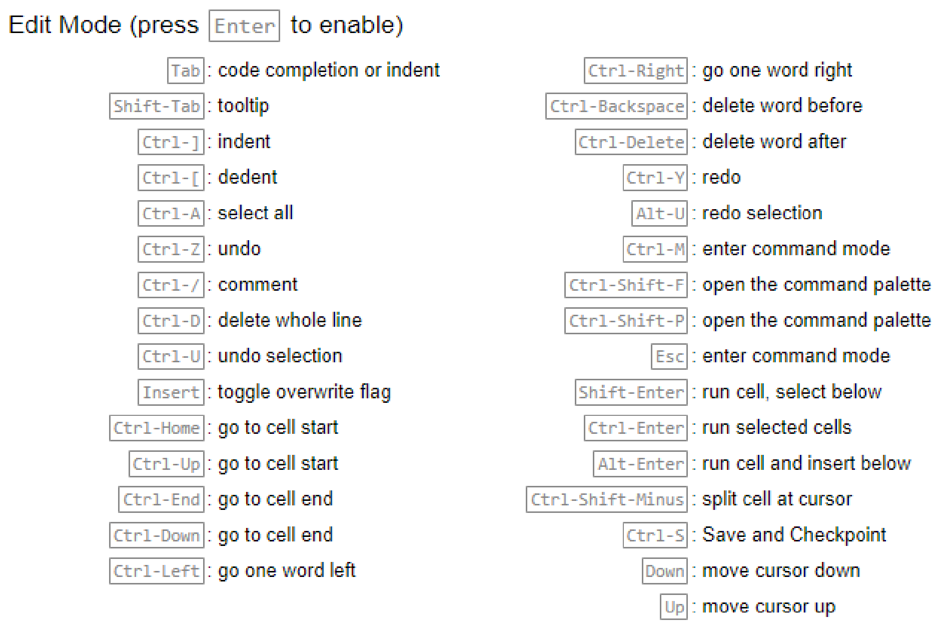
- Interactive Demonstration



Command Mode (press Esc to enable)

Edit Shortcuts

| | |
|---|---|
| F | find and replace |
| Ctrl-Shift-F | open the command palette |
| Ctrl-Shift-P | open the command palette |
| Enter | enter edit mode |
| P | open the command palette |
| Shift-Enter | run cell, select below |
| Ctrl-Enter | run selected cells |
| Alt-Enter | run cell and insert below |
| Y | change cell to code |
| M | change cell to markdown |
| R | change cell to raw |
| 1 | change cell to heading 1 |
| 2 | change cell to heading 2 |
| 3 | change cell to heading 3 |
| 4 | change cell to heading 4 |
| 5 | change cell to heading 5 |
| 6 | change cell to heading 6 |
| K | select cell above |
| Up | select cell above |
| Down | select cell below |
| J | select cell below |
| Shift-K | extend selected cells above |
| Shift-Up | extend selected cells above |
| Shift-Down | extend selected cells below |
| Shift-J | extend selected cells below |
| A | insert cell above |
| B | insert cell below |
| X | cut selected cells |
| C | copy selected cells |
| Shift-V | paste cells above |
| V | paste cells below |
| Z | undo cell deletion |
| D , D | delete selected cells |
| Shift-M | merge selected cells, or current cell with cell below if only one cell is selected |
| Ctrl-S | Save and Checkpoint |
| S | Save and Checkpoint |
| L | toggle line numbers |
| O | toggle output of selected cells |
| Shift-O | toggle output scrolling of selected cells |
| H | show keyboard shortcuts |
| I , I | interrupt the kernel |
| 0 , 0 | restart the kernel (with dialog) |
| Esc | close the pager |
| Q | close the pager |
| Shift-L | toggles line numbers in all cells, |

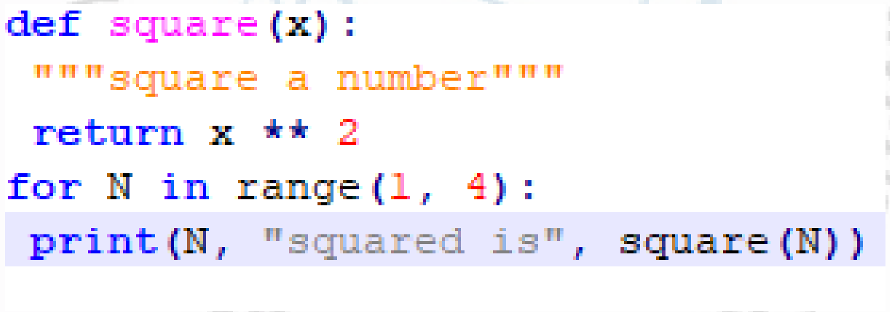


Edit Mode (press Enter to enable)

| | |
|---|---|
| Tab | code completion or indent |
| Shift-Tab | tooltip |
| Ctrl-] | indent |
| Ctrl-[ | dedent |
| Ctrl-A | select all |
| Ctrl-Z | undo |
| Ctrl-/ | comment |
| Ctrl-D | delete whole line |
| Ctrl-U | undo selection |
| Insert | toggle overwrite flag |
| Ctrl-Home | go to cell start |
| Ctrl-Up | go to cell start |
| Ctrl-End | go to cell end |
| Ctrl-Down | go to cell end |
| Ctrl-Left | go one word left |
| Ctrl-Right | go one word right |
| Ctrl-Backspace | delete word before |
| Ctrl-Delete | delete word after |
| Ctrl-Y | redo |
| Alt-U | redo selection |
| Ctrl-M | enter command mode |
| Ctrl-Shift-F | open the command palette |
| Ctrl-Shift-P | open the command palette |
| Esc | enter command mode |
| Shift-Enter | run cell, select below |
| Ctrl-Enter | run selected cells |
| Alt-Enter | run cell and insert below |
| Ctrl-Shift-Minus | split cell at cursor |
| Ctrl-S | Save and Checkpoint |
| Down | move cursor down |
| Up | move cursor up |

## #1.3 Notebook's Magic Commands

- Run external code: **%run**
  - Create a file named '**script1.py**' with the following content (continue on Notebook)



```
def square(x):
 """square a number"""
 return x ** 2
for N in range(1, 4):
 print(N, "squared is", square(N))
```

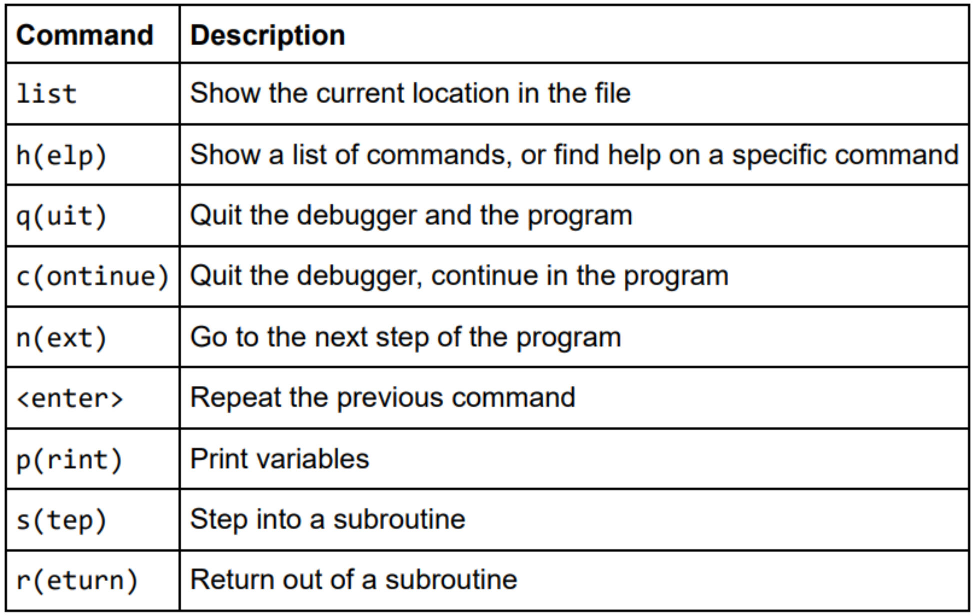
  - General description of magic commands: **%magic**
  - List all available magic commands: **%lsmagic**

## #1.4 Input and Output History

- Notebook's *In* and *Out* objects
- Underscore Shortcuts and Previous Outputs
- Suppressing Output
- Related Magic Commands

- **#1.5 Notebook and Shell Commands**
  - Shell Commands
  - Shell Commands in Jupyter Notebook
  - Passing values to and from Shell

- **#1.6 Errors and Debugging**
  - Controlling Exceptions: **%xmode**
  - Debug mode
  - Partial list of debugging commands



| Command | Description |
|---|---|
| `list` | Show the current location in the file |
| `h(elp)` | Show a list of commands, or find help on a specific command |
| `q(uit)` | Quit the debugger and the program |
| `c(ontinue)` | Quit the debugger, continue in the program |
| `n(ext)` | Go to the next step of the program |
| `<enter>` | Repeat the previous command |
| `p(rint)` | Print variables |
| `s(tep)` | Step into a subroutine |
| `r(eturn)` | Return out of a subroutine |

- **#1.7 Profiling and Timing Code**
  - Timing Code Snippets: **%timeit** and **%time**
  - Profiling Full Scripts: **%prun**
  - Line-by-Line Profiling: **%lprun**
    - Install Line Profiler using '*pip*':

      **'pip install line_profiler'**
  - Profiling Memory Use: **%memit** and **%mprun**
    - Install Memory Profiler using '*pip*':

      **'pip install memory_profiler'**

# THANK YOU FOR YOUR PARTICIPATION

## #1 IPython – Beyond Normal Python

(23 JAN 2019)

## Next module is:

**#2 Introduction to NumPy**

(25 JAN 2019)