0101-training-notebook

March 19, 2024

1 0101 - First Session With Python - Solution Notebook

- Written by Alexandre Gazagnes
- Last update: 2024-02-01

1.1 About

1.1.1 Using Jupyter

You have 2 options: - Locally:

```
- **Install Anaconda https://www.anaconda.com/ or Jupyter https://jupyter.org/install on your m

- Use Anaconda or Jupyter installed on the Unilasalle PC (**Warning **: some packages may be m
```

- Online:
  - **Use Google Colab https://colab.research.google.com/** (you have to be connected to your google account)
  - **Open this notebook on Google colab URL**
    - Badge
  - Use Jupyter online https://jupyter.org/try-jupyter (**Warning** : External packages cannot be installed)

1.1.2 Material

All the material for this course could be found here. - https://github.com/AlexandreGazagnes/Unilassalle-Public-Ressources/tree/main/4a-data-analysis

1.1.3 Python / Jupyter ?

Few Questions : - Why Python - Python vs R ? - What is Data Analysis ? - What are we talking about ? - What is Jupyter ?

1.1.4 Context

You are a new employee of the NPO named "NPO".

You are in charged of data analysis.

First project is about GHG emissions, more precisely regarding Bovine Meat.

1.1.5 Data

After a quick look on the internet, you find a very interesting dataset on the FAO website. It contains a list of various indicators. You decide to use this dataset to identify segments of countries.

- Find relevant data :
  - https://www.kaggle.com/datasets/unitednations/global-food-agriculture-statistics
  - https://www.kaggle.com/datasets/dorbicycle/world-foodfeed-production
  - https://www.fao.org/faostat/en/
  - https://fr-en.openfoodfacts.org/
  - https://fr-en.openfoodfacts.org/data

You can use a preprocessed version of the dataset here. (Best option)

1.1.6 Mission

Our job is to : * Prepare notebook environment * Load data * Explore data * Clean data ==> Select relevant data * Clean data ==> Handle missing values * Clean data ==> Handle duplicates ? * Clean data ==> Handle outliers ? * Perform some basic analysis and data inspection * Perform some basic visualisation * Export our data

1.1.7 Usefull Ressources about Google Colab

- On Youtube :
  - https://www.youtube.com/watch?v=8KeJZBZGtYo
  - https://www.youtube.com/watch?v=JJYZ3OE_lGo
  - https://www.youtube.com/watch?v=tCVXoTV12dE

1.1.8 Usefull Ressources about Anaconda and Jupyter

- On Youtube :
  - https://www.youtube.com/watch?v=ovlID7gefzE
  - https://www.youtube.com/watch?v=IMrxB8Mq5KU
  - https://www.youtube.com/watch?v=Ou-7G9VQugg
  - https://www.youtube.com/watch?v=5pf0_bpNbkw

1.1.9 Teacher

- More info :
  - https://www.linkedin.com/in/alexandregazagnes/
  - https://github.com/AlexandreGazagnes

1.2 Preliminaries

1.2.1 System

These commands will display the system information:

Uncomment theses lines if needed.

[ ]:
```
# pwd
```

[ ]:
```
# cd ..
```

[ ]:
```
# ls
```

[ ]:
```
# cd ..
```

[ ]:
```
# ls
```

These commands will install the required packages:

Please note that if you are using google colab, all you need is already installed

[ ]:
```
# !pip install pandas matplotlib seaborn plotly scikit-learn
```

This command will download the dataset:

Please note that we will download the dataset later, in this notebook

[ ]:
```
# !wget https://gist.githubusercontent.com/AlexandreGazagnes/
  ↪2000e5c0e9149ffdb8c682a751ac448a/raw/
  ↪35ad83320c26155415b7cccff8a4150ee80ee501/FAO_Unilassalle_raw.csv
```

1.2.2 Imports

Import data libraries:

[ ]:
```
import pandas as pd  # DataFrame
import numpy as np  # Matrix and advanced maths operations
```

Import Graphical libraries:

[ ]:
```
import matplotlib.pyplot as plt  # Visualisation
import seaborn as sns  # Visualisation
import plotly.express as px  # Visualisation (not used here)
```

:warning:**These imports must be done, it is not possible to use this notebook without pandas, matplotlib etc.**

1.2.3 Data

1st option : Download the dataset from the web

[ ]:
```
# url
url = "https://gist.githubusercontent.com/AlexandreGazagnes/
  ↪2000e5c0e9149ffdb8c682a751ac448a/raw/
  ↪35ad83320c26155415b7cccff8a4150ee80ee501/FAO_Unilassalle_raw.csv"
url
```

Read the data :

[ ]:
```
df = pd.read_csv(url, encoding="latin1")
df.head()
```

2nd Option : Read data from a file

[ ]:
```
# # or

# fn = "my/awsome/respository/my_awsome_file.csv"
# fn = "./data/source/FAO.csv"
# df = pd.read_csv(fn, encoding='latin1')
```

1.3 Data Exploration

1.3.1 Display

Display the first rows of the dataset:

[ ]:
```
# head

df.head()
```

Display the last rows of the dataset:

[ ]:
```
# tail

df.tail(10)
```

Display a sample of the dataset:

[ ]:
```
# sample 10

df.sample(10)
```

[ ]:
```
# Sample with just 10% of the dataset

df.sample(frac=0.1)
```

1.3.2 Structure

What is the shape of the dataset?

[ ]:
```
# shape
```

What data types are present in the dataset?

[ ]:
```
# dtypes
```

:warning: **Please note that we have here main python dtypes** Data types : - int : *Integer* : 1,2,12332, 1_000_000 - float : *Float* : 1.243453, 198776.8789, 1.9776 - object : In this example object stands for *String* : “Paris”, “Rouen”, “Lea”

Count the number of columns with specific data types:

[ ]:
```
# value_counts
```

Select only string columns:

[ ]:
```
# select_dtypes
```

Counting unique values for string columns :

[ ]:
```
# nunique
```

1.3.3 Select data

Display all the columns :

[ ]:
```
# columns
```

Just use a small number of columns :

[ ]:
```
columns = [
    "Area Abbreviation",
    "Area Code",
    "Area",
    "Item Code",
    "Item",
    "Element Code",
    "Element",
    "Unit",
    "latitude",
    "longitude",
    "Y2010",
    "Y2011",
    "Y2012",
    "Y2013",
]
columns
```

Make your column selection and display the output :

[ ]:
```
# loc ? => JUST THE OUTPUT
```

If this Transformation is OK, you can re-write your `df` variable :

[ ]:
```
# loc ? => REWRITE the DF
```

Use `iloc` to select the nth line and the mth column :

[ ]:
```
# iloc
```

Use `iloc` to select data from 1st to the nth line and from first to the mth column :

[ ]:
```
# iloc
```

Just keep in mind the global shape of our dataset :

[ ]:

And the names of our columns :

[ ]:

Columns with the *code* key word are not relevant :

[ ]:
```
columns = ["Area Code", "Item Code", "Element Code"]
columns
```

Suppose we have 1_000 columns ...

Let's find a more *pythonic* way to extract the *code* columns :

[ ]:

:clap: We have used : - a `list` : `columns = []` - a `for` loop - an `if` statement

What is the value of the `columns` variable ?

[ ]:

Let's drop these columns :

[ ]:
```
# drop columns
```

Rewrite our dataframe

[ ]:

[ ]:
```
# drop indexes
```

[ ]:
```
# Drop with errors="ignore"
```

Another usage of iloc :

[ ]:
```
# Implenting iloc
```

So far so good, we can rewrite our `df`

[ ]:
```
# Saving our df
```

Selecting a specific column :

[ ]:
```
# 1st implementation
```

[ ]:
```
# 2nd implementation
```

Can we have a good representation of each unique value for the `Item` column ?

[ ]:
```
# Item unique ?
```

Is `meat` in our Item column ?

[ ]:
```
# Meat in Item unique ?
```

Use a list, a for loop and an if statement to be sure to have all items with `Meat` :

[ ]:
```
# Select meat items
```

Build a boolean selector :

[ ]:
```
# Creating a selector True / False
```

Select relevant data with the `loc` method :

[ ]:
```
# .loc
```

Try a more advanced selection :

[ ]:
```
# More advanced selection
```

What about Area ?

[ ]:
```
# Area?
```

And area number of unique values ?

[ ]:
```
# Area nunique ?
```

Same for Item :

[ ]:
```
# Item nunique ?
```

Same for Unit :

[ ]:
```
# Unit unique ?
```

Drop uselss columns :

[ ]:
```
# Drop other useless columns

columns = [
    "Item",
    "Element",
    "Unit",
    "latitude",
    "longitude",
]
```

1.3.4 NaN Values

Lets have a look to NaN (Not a Number) aka missing values :

[ ]:
```
# Nan Values
```

Compute the sum of missing values for each line :

[ ]:
```
# Sum of Nan Values
```

Try to focus on a specifc column:

[ ]:
```
# Select Nan Values
```

Try to focus on a specific Country :

[ ]:
```
# Other selection
```

Drop Sudan from our DataFrame :

[ ]:
```
# Drop a specific row
```

[ ]:
```
# Drop a specific row
```

Are we done ?

[ ]:
```

```

Useless but fun :

[ ]:
```

```

Final output of `df` :

[ ]:
```

```

1.3.5 Data Inspection

[ ]:
```
# Describe
```

[ ]:
```
# Better describe ?
```

[ ]:
```
# Recast as int
```

[ ]:
```
# Sort by values
```

[ ]:
```
# Select small values
```

[ ]:
```
# Select small values and sort
```

[ ]:
```
# select 'big' values ==> drop lower values
```

[ ]:
```
# sort by values top :
```

[ ]:
```
# Are we good ?
```

[ ]:
```
# Just to be sure :
```

[ ]:
```
# Creating tmp variable, just with numeric values
```

[ ]:
```
# Correlation matrix is non sens here
# (sorry for that  )
```

[ ]:
```
# Heatmap ?
```

[ ]:
```
# Better heatmap ?
```

[ ]:
```
# Best heatmap ever done ?
```

[ ]:
```
# Build your first function
```

[ ]:
```
# Use this function
```

1.4 Visualisation

1.4.1 Distplot

[ ]:
```
# Just to be sure
```

[ ]:
```
# Just to be sure
```

[ ]:
```
# Distplot
```

[ ]:
```
# Distplot normal
```

[ ]:
```
# What about skewness ?
```

[ ]:
```
# What about kurtosis ?
```

[ ]:
```
# Log1p => log(x+1) ?
```

[ ]:
```
# Top 5
```

1.4.2 Barplot

[ ]:
```
# Bar plot
```

[ ]:
```
# Same but better
```

1.4.3 Boxplot

[ ]:
```
# My favorite plot EVER ;)
```

[ ]:
```
# Ok, this one
```

[ ]:
```
# Just another df output
```

1.4.4 Lineplot

[ ]:
```
# Melt ?
```

[ ]:
```
# Boxplot
```

[ ]:
```
# Line plot
```

[ ]:
```
# Melt only top 5
```

1.5 Export

Export the csv file :

[ ]:
```

```