0101-training-notebook

March 18, 2024

1 0101 - First Session With Python - Training Notebook

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

1.1 About

1.1.1 Using Jupyter

You have 3 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** : https://github.com/AlexandreGazagnes/Unilassalle-Public-Ressources/blob/main/4a-data-analysis/01-session/0101-training-notebook.ipynb
    - 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 on PCA

- About ACP
  - https://www.youtube.com/
  - https://www.youtube.com/
  - https://www.youtube.com/
  - https://www.youtube.com/watch?v=HMOI_lkzW08
  - https://www.youtube.com/watch?v=FgakZw6K1QQ
  - https://www.youtube.com/watch?v=0Jp4gsfOLMs&list=PLblh5JKOoLUJJpBNfk8_YadPwDTO2SC
  - https://www.youtube.com/watch?v=oRvgq966yZg
  - https://www.youtube.com/watch?v=FgakZw6K1QQ&list=PLblh5JKOoLUIcdlgu78MnlATeyx4cEVeR
  - https://www.youtube.com/watch?v=_UVHneBUBW0
  - https://www.youtube.com/watch?v=KrNbyM925wI&list=PLnZgp6epRBbRn3FeMdaQgVsFh9Kl0fjqX
  - https://www.youtube.com/watch?v=2UFiMvXvdZ4
  - THE BEST ONE : https://www.youtube.com/watch?v=VdpNEjStT5g

1.1.8 Teacher

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

1.2 Preliminaries

1.2.1 System

[ ]:
```
# pwd
```

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

[ ]:
```
# ls
```

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

[ ]:
```
# ls
```

[ ]:
```
# !pip install -r requirements.txt
```

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

[ ]:
```
# If you want to download the data from the web, please uncomment the following␣
↪lines

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

1.2.2 Imports

[ ]:
```
# Imports

import numpy as np
import pandas as pd
```

[ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

# from sklearn.datasets import load_iris
```

1.2.3 Data

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

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

```
df
```

[ ]:
```
# or

# data = load_iris()
# df = pd.DataFrame(data.data, columns=data.feature_names)
# df["Species"] = data.target
# df.head()
```

[ ]:
```
# or

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

1.3 Data Exploration

1.3.1 Display

[ ]:
```
# head
```

[ ]:
```
# tail
```

[ ]:
```
# sample 10
```

[ ]:
```
# sample frac
```

1.3.2 Structure

[ ]:
```
# shape
```

[ ]:
```
# dtypes
```

[ ]:
```
# count?
```

[ ]:
```
# select ?
```

[ ]:
```
# nunique int ?
```

[ ]:
```
# nunique float?
```

1.3.3 Select data

[ ]:
```
# columns ?
```

[ ]:
```
columns = [
    "Area Abbreviation",
    "Area Code",
    "Area",
```

```
    "Item Code",
    "Item",
    "Element Code",
    "Element",
    "Unit",
    "latitude",
    "longitude",
    "Y2010",
    "Y2011",
    "Y2012",
    "Y2013",
]
columns
```

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

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

[ ]:
```
# iloc ?
```

[ ]:
```
# head
```

[ ]:
```
# columns ?
```

[ ]:
```
# Creating a list of column with code

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

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

[ ]:
```
# Output columns
```

[ ]:
```
# If needed :
column_list = ["Area Code", "Item Code", "Element Code"]
column_list
```

[ ]:
```
# Drop columns
```

[ ]:
```

```

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

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

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

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

[ ]:
```
# Just a specific column
```

[ ]:
```
# Just a specific column
```

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

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

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

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

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

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

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

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

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

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

[ ]:
```
# Drop other useless columns

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

1.3.4 NaN

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

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

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

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

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

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

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

[ ]:
```
# Useless but fun
```

[ ]:
```
# Output 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


def corr_heatmap(df):
    tmp = df.select_dtypes(include="number")
    corr = tmp.corr()
    mask = np.triu(corr)
    sns.heatmap(
        corr, annot=True, cmap="coolwarm", fmt=".4f", vmin=-1, vmax=1, mask=mask
```

```
    )
```

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

1.3.6 Visualisation

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

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

[ ]:
```
# Distplot
```

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

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

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

[ ]:
```
# Log1p ?
```

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

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

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

[ ]:
```
# My favorite plot
```

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

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

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

[ ]:
```
# Boxplot
```

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

[ ]:
```
# Melt
```

1.4 Export

[ ]:
```
# Export Csv
```

[ ]:
```

```