EUROPYTHON PRAGUE & REMOTE 17-23 July 2023

FROM JUPYTER NOTEBOOKS TO A PYTHON PACKAGE:

THE BEST OF BOTH WORLDS

Sin-seok SEO, Safran Tech, Safran SA

OUTLINE

1. Introduction
   - ✓ Jupyter Notebook
   - ✓ Full-fledged IDEs
2. Jupyter Notebook Data Science Workflow
   - ✓ Data loading
   - ✓ Preprocessing
   - ✓ Exploratory Data Analysis (EDA)
   - ✓ Prediction
3. To (Your Own) Package
   - ✓ *What is a Python package?*
   - ✓ *How to create a (minimal) package*
   - ✓ *How to import and use*
   - ✓ *Live refactoring examples*
4. Wrap-up / Some Tips

* All the materials are available here:
https://github.com/sesise0307/europython2023-package

INTRODUCTION

- **Jupyter Notebook**
  - ✓ Web-based interactive application for creating and sharing computational documents
  - ✓ Provides ideal workflows for data science, scientific computing and machine learning
  - ✓ Pros: REPL (Read-Eval-Print-Loop), interactivity, integration of code / output / documentation, visualization, rapid prototyping, result sharing, etc.
  - ✓ Cons: lacks of debugging, code sharing, refactoring, version control, advanced editing, etc.

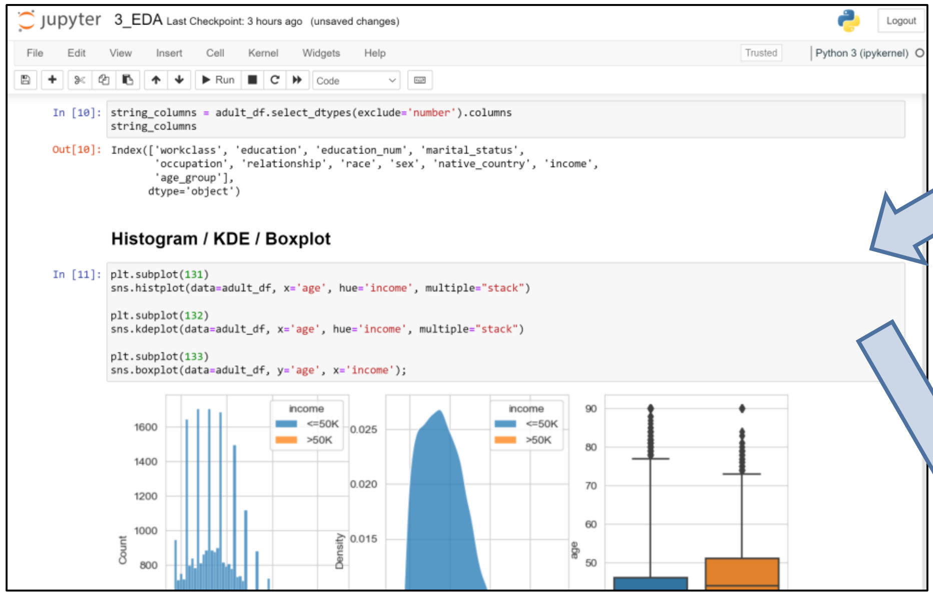
- **Full-fledged IDE** (Integrated Development Env.)
  - ✓ Such as VS Code, PyCharm, Eclipse, Spyder, …
  - ✓ An application for software development providing
    - Code editing / Syntax highlighting / Code completion
    - Debugging / Building / Testing
    - Version control / Packaging
  - ✓ Designed to maximize programmer productivity
  - ✓ One iteration might take a long journey

- Fortunately, they are not exclusive
  - ✓ We can benefit from the best of both worlds
  - ✓ By using a Python package

BEST OF BOTH WORLDS

Jupyter Notebook:
REPL, Prototyping, Visualizations, Experiments, Documentation, …

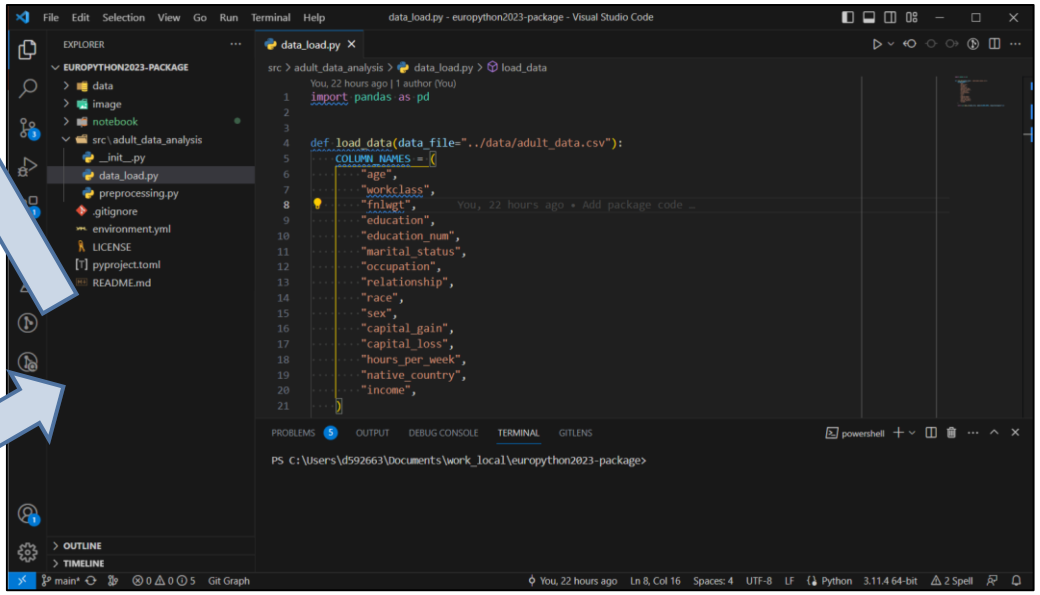
Python Package with IDE:
Common code (function / class / module / package), Refactoring, Unit tests, Version control, Debugging, …


jupyter 3_EDA Last Checkpoint: 3 hours ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
income
<=50K
>50K
Count
Density
age

```
In [10]: string_columns = adult_df.select_dtypes(exclude='number').columns
         string_columns

Out[10]: Index(['workclass', 'education', 'education_num', 'marital_status',
                'occupation', 'relationship', 'race', 'sex', 'native_country', 'income',
                'age_group'],
               dtype='object')
```

Histogram / KDE / Boxplot

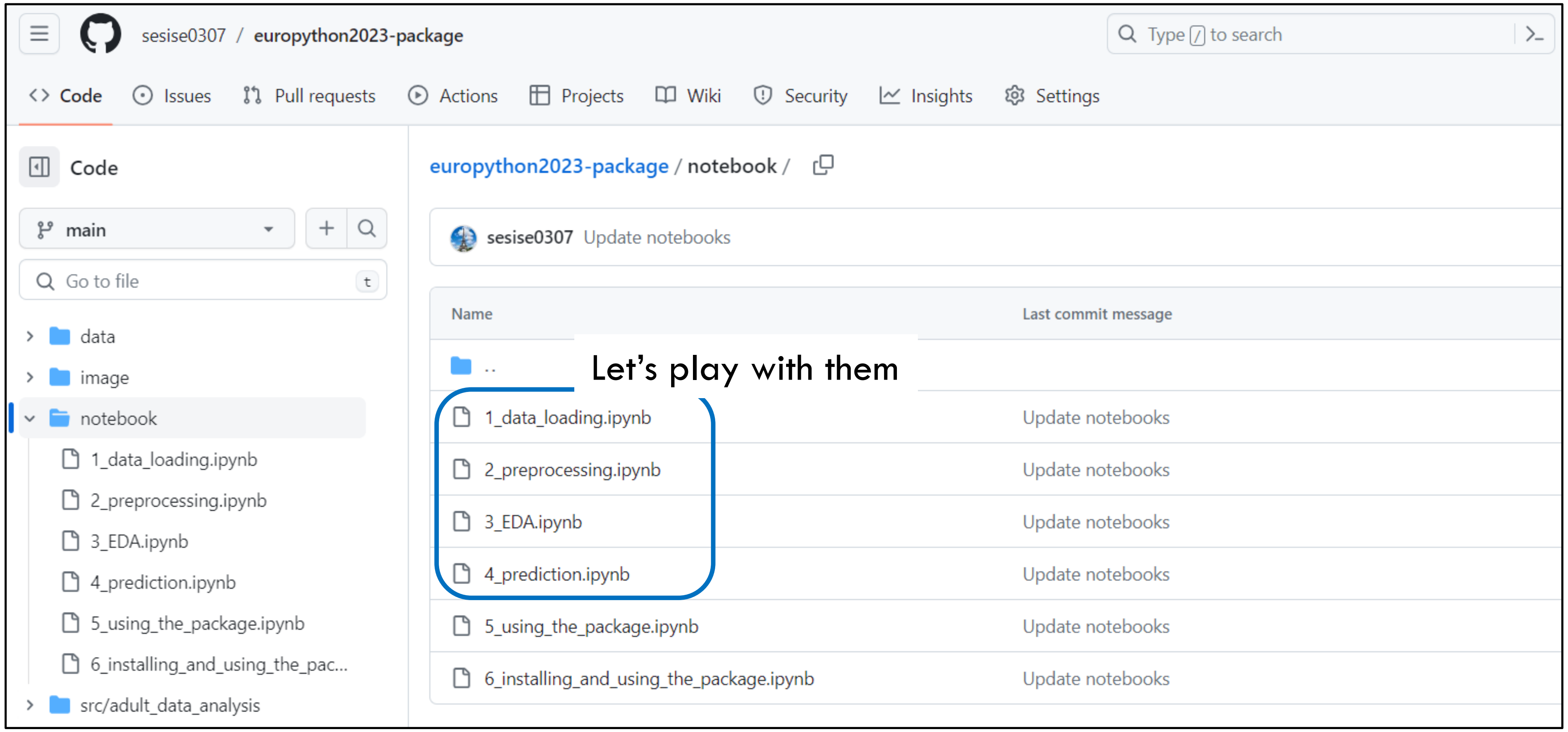
```
In [11]: plt.subplot(131)
         sns.histplot(data=adult_df, x='age', hue='income', multiple="stack")

         plt.subplot(132)
         sns.kdeplot(data=adult_df, x='age', hue='income', multiple="stack")

         plt.subplot(133)
         sns.boxplot(data=adult_df, y='age', x='income');
```



```
You, 22 hours ago | 1 author (You)
import pandas as pd


def load_data(data_file="../data/adult_data.csv"):
    COLUMN_NAMES = (
        "age",
        "workclass",
        "fnlwgt",        You, 22 hours ago • Add package code …
        "education",
        "education_num",
        "marital_status",
        "occupation",
        "relationship",
        "race",
        "sex",
        "capital_gain",
        "capital_loss",
        "hours_per_week",
        "native_country",
        "income",
    )
```

```
PS C:\Users\d592663\Documents\work_local\europython2023-package>
```

JUPYTER NOTEBOOK DATA SCIENCE WORKFLOW



sesise0307 / europython2023-package

Code · Issues · Pull requests · Actions · Projects · Wiki · Security · Insights · Settings

Code

main

Go to file

- data
- image
- notebook
  - 1_data_loading.ipynb
  - 2_preprocessing.ipynb
  - 3_EDA.ipynb
  - 4_prediction.ipynb
  - 5_using_the_package.ipynb
  - 6_installing_and_using_the_pac...
- src/adult_data_analysis

europython2023-package / notebook /

sesise0307 Update notebooks

Name	Last commit message
..	
1_data_loading.ipynb	Update notebooks
2_preprocessing.ipynb	Update notebooks
3_EDA.ipynb	Update notebooks
4_prediction.ipynb	Update notebooks
5_using_the_package.ipynb	Update notebooks
6_installing_and_using_the_package.ipynb	Update notebooks

Let's play with them

PYTHON PACKAGE

➢ **What is a Python Package?**

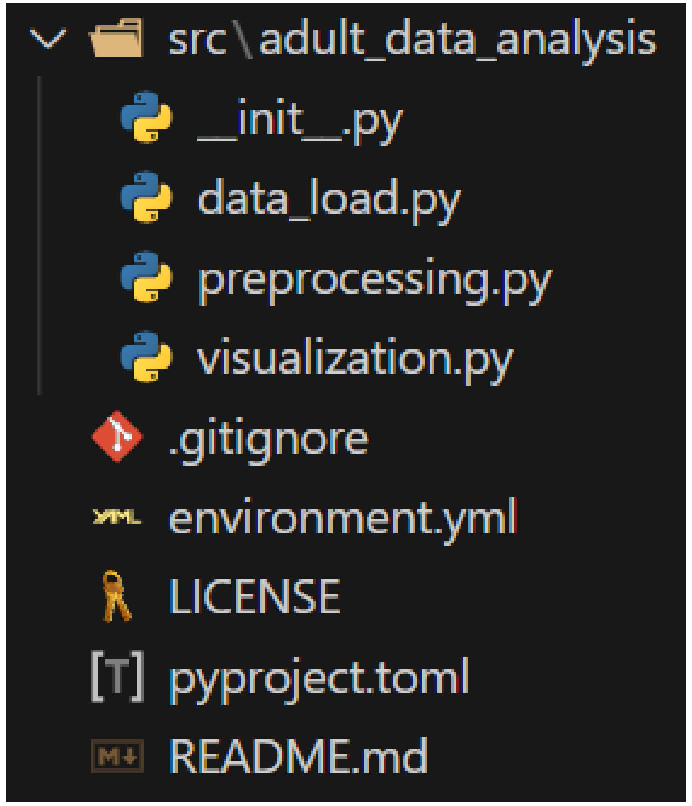
- ✓ Any directory with an *__init__.py* file
- ✓ A package can contain modules (python files) and sub-packages (sub-directories)
- ✓ We can say that it is a collection of modules

➢ **Packages Make It Easy**

- ✓ To reuse and share code
- ✓ To install with *pip* or *easy_install*
- ✓ To specify as a dependency for another package
- ✓ For other users to download and run tests
- ✓ To add and distribute documentation

HOW TO CREATE A (MINIMAL) PACKAGE

- Pick a package name
  - ✓All lowercase / Underscore-separated
  - ✓In this case: *adult_data_analysis*
- Create a package root directory
  - ✓Normally same to the package name
  - ✓You can put extra files here such as *README.md* or *LICENSE*
- Create a package source directory
  - ✓*src/package_name*[1]
  - ✓Create a *__init__.py* file
  - ✓Add python module files and put your code

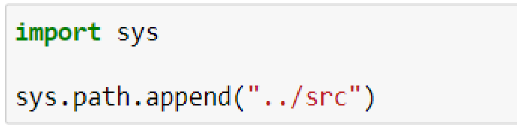


```
src\adult_data_analysis
    __init__.py
    data_load.py
    preprocessing.py
    visualization.py
.gitignore
environment.yml
LICENSE
pyproject.toml
README.md
```

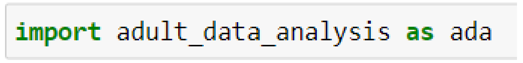
[1]src-layout vs. flat-layout: https://setuptools.pypa.io/en/latest/userguide/package_discovery.html#automatic-discovery

HOW TO IMPORT AND USE YOUR PACKAGE [1/2]

- Adding your local path



```
import sys

sys.path.append("../src")
```



```
import adult_data_analysis as ada
```

✓ Simple
✓ Path is relative to the notebook path
✓ Have to add the two lines of code in each notebook
✓ Not an usual way of importing packages

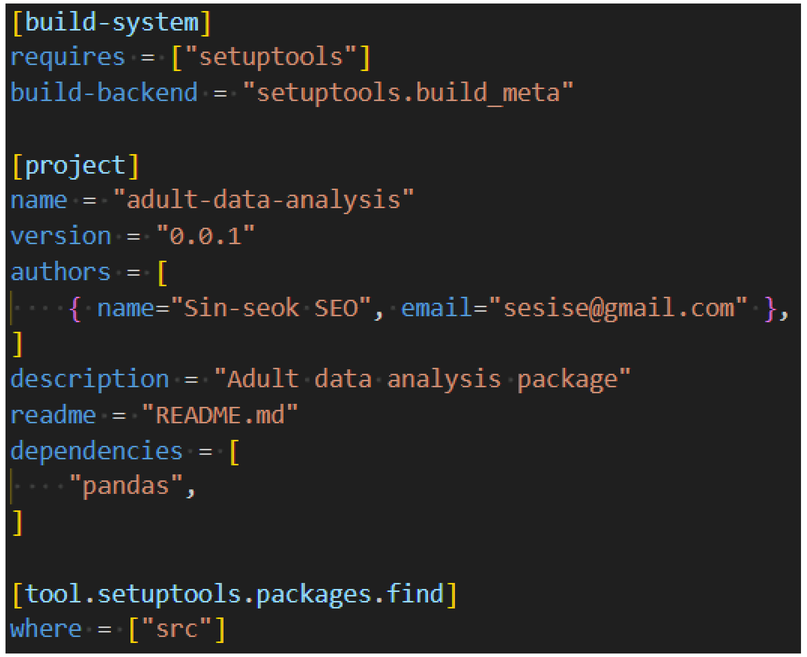
- Updating the package and making it effective
  - ✓Option 1: Restarting the jupyter kernel and run cells again
  - ✓Option 2: Using jupyter *autoreload* magic command (recommended)

- Let's check it out with the notebook *5_using_the_package.ipynb*

Reference: https://setuptools.pypa.io/en/latest/userguide/quickstart.html

HOW TO IMPORT AND USE YOUR PACKAGE [2/2]

- Instead, let's create an installable package
  - ✓ Option 1: Using *setup.py*
  - ✓ Option 2: Using *setup.cfg* + *setup.py*
  - ✓ Option 3: Using *pyproject.toml*
    - Starting with PEP 621, the Python community selected *pyproject.toml* as a standard way of specifying project metadata



```toml
[build-system]
requires = ["setuptools"]
build-backend = "setuptools.build_meta"

[project]
name = "adult-data-analysis"
version = "0.0.1"
authors = [
    { name="Sin-seok SEO", email="sesise@gmail.com" },
]
description = "Adult data analysis package"
readme = "README.md"
dependencies = [
    "pandas",
]

[tool.setuptools.packages.find]
where = ["src"]
```

- Then, install the package using `*pip*`
  - `*pip install <path>*` → static installation
  - `*pip install --editable <path>*` → editable or develop mode

- Let's check it out with the notebook *6_installing_and_using_the_package.ipynb*

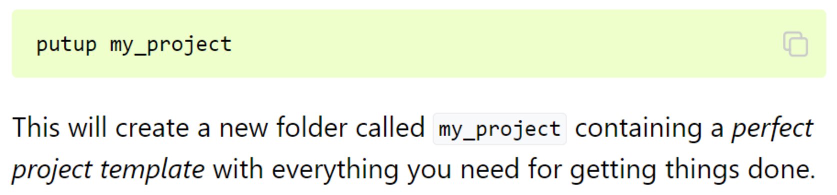
WRAP UP / SOME TIPS

➢ We Can Take Full Advantages of

- ✓ Jupyter Notebooks and full-fledged IDEs
- ✓ By using a Python package
- ✓ And some simple tricks (*autoreload*, editable install)
- ✓ This will help you boost your productivity

➢ PyScaffold

- ✓ Check this really nice package generator for bootstrapping high-quality Python packages



```
putup my_project
```

This will create a new folder called `my_project` containing a *perfect project template* with everything you need for getting things done.

➢ Next Step

- ✓ Publish your awesome package to the Python Package Index (PyPI) and share with the world
- ✓ Refer to this page as a starting point:
  - https://packaging.python.org/en/latest/tutorials/packaging-projects/

➢ VS Code is Great and Free

- ✓ Developer friendly and very customizable IDE
- ✓ Lots of features / extensions
- ✓ Very responsive
- ✓ Monthly updates

Thank you for your attention!

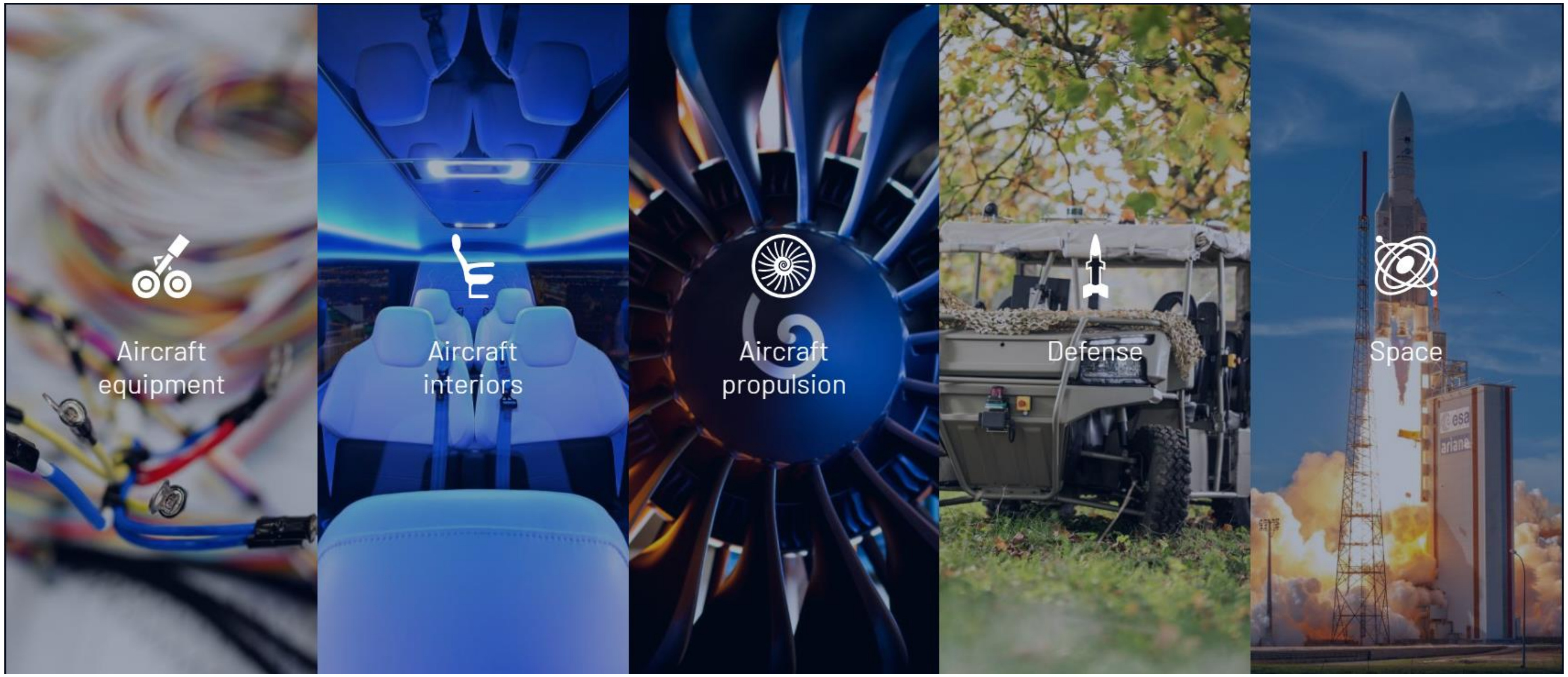
Q&A

REFERENCES

- https://setuptools.pypa.io/en/latest/userguide/quickstart.html
- https://python-packaging.readthedocs.io/
- https://setuptools.pypa.io/en/latest/userguide/pyproject_config.html
- *https://setuptools.pypa.io/en/latest/userguide/package_discovery.html*
- *https://packaging.python.org/en/latest/guides/single-sourcing-package-version/*

ABOUT SAFRAN GROUP

More than 83000 employees in 276 locations across 27 countries



SAFRAN'S AIRCRAFT ENGINES

Through CFM International (the 50/50 joint company between Safran Aircraft Engines and GE) we produce the LEAP® turbofan, successor to the best-selling CFM56®. The LEAP powers new-generation single-aisle commercial jets: the Airbus A320neo, Boeing 737 MAX and COMAC C919. We're also a leading military aircraft engine manufacturer, supplying the M88 for the Rafale fighter, and as part of a consortium making the TP400 turboprop engine for the Airbus A400M transport aircraft

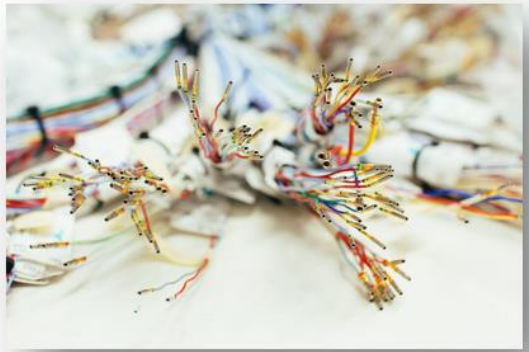
SAFRAN TECH

- About 500 employees / 63 doctoral theses
- More than 23 nationalities
- 300 invention disclosures
- 5 sites : Paris-Saclay, Itteville, Gennevilliers, Le Haillan and Toulouse

MATERIALS & PROCESSES

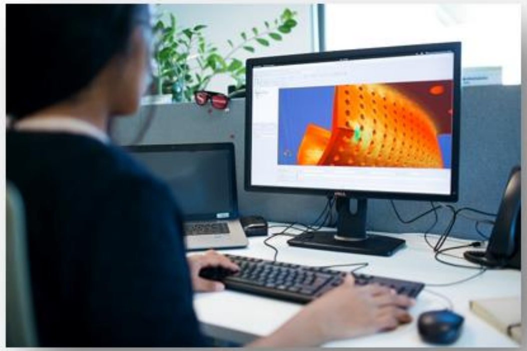
- **Identify** and **apply innovative materials and processes** to make products that offer **high performance, lighter** weight and **simpler** production and maintenance

ENERGETICS



- **Develop** innovative **energy and propulsion systems** and technologies to **address environmental challenges** and new applications

INFORMATION



- **Capture, process and model information** to improve our productivity and **increase** product **competitiveness and performance**

Our ambition

Bring differentiating technologies to the Group businesses' products and services of the future.

MY WORK@SAFRAN TECH: RESEARCH ENGINEER (SINCE 2017)

Data Analysis

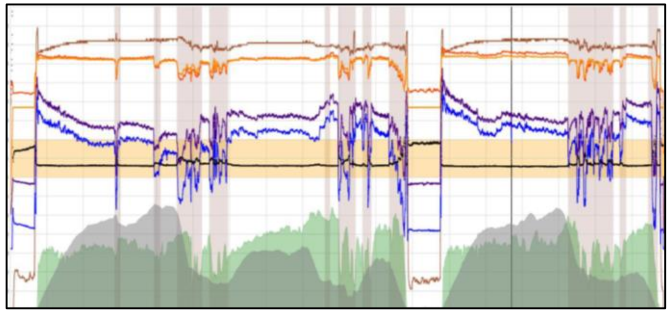
-Data Preprocessing
- EDA (Exploratory Data Analysis)
- Data Visualization
- Outlier and anomaly detection
- Multi-variate time series analysis

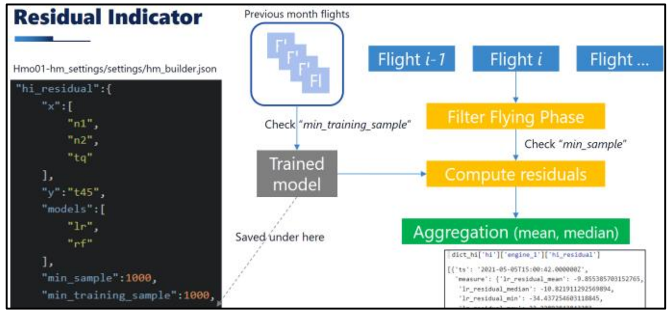
Machine Learning

- Regression
- Classification
- Deep Neural Network
- Health indicator development

Maintenance Policy

-Prognostics and Health Monitoring (PHM)
- Maintenance policy evaluations with Discrete Event Simulation (DES)
- Maintenance scheduling optimization

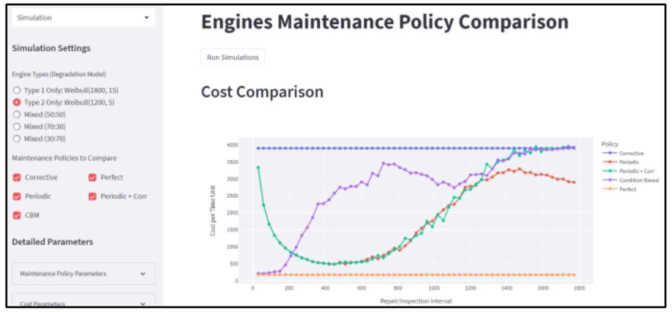




Hmo01-hm_settings/settings/hm_builder.json

```
"hi_residual":{
    "x":[
        "n1",
        "n2",
        "tq"
    ],
    "y":"t45",
    "models":[
        "lr",
        "rf"
    ],
    "min_sample":1000,
    "min_training_sample":1000,
```

```
dict_hi['hi']['engine_1']['hi_residual']

[{'ts': '2021-05-05T15:00:42.000000Z',
  'measure': {'lr_residual_mean': -9.855385703152765,
   'lr_residual_median': -10.821911292569894,
   'lr_residual_min': -34.437254603118845,
```



Engines Maintenance Policy Comparison

Simulation Settings

Engine Types (Degradation Model)
- ○ Type 1 Only: Weibull(1800, 15)
- ● Type 2 Only: Weibull(1200, 5)
- ○ Mixed (50:50)
- ○ Mixed (70:30)
- ○ Mixed (30:70)

Maintenance Policies to Compare
- ✓ Corrective
- ✓ Perfect
- ✓ Periodic
- ✓ Periodic + Corr
- ✓ CBM

Detailed Parameters

Cost Comparison