Running Notebooks in Production? Blessing or Curse? Let's Find out!

Eduardo Blancas

Ploomber Co-Founder, CEO

ploomber.io

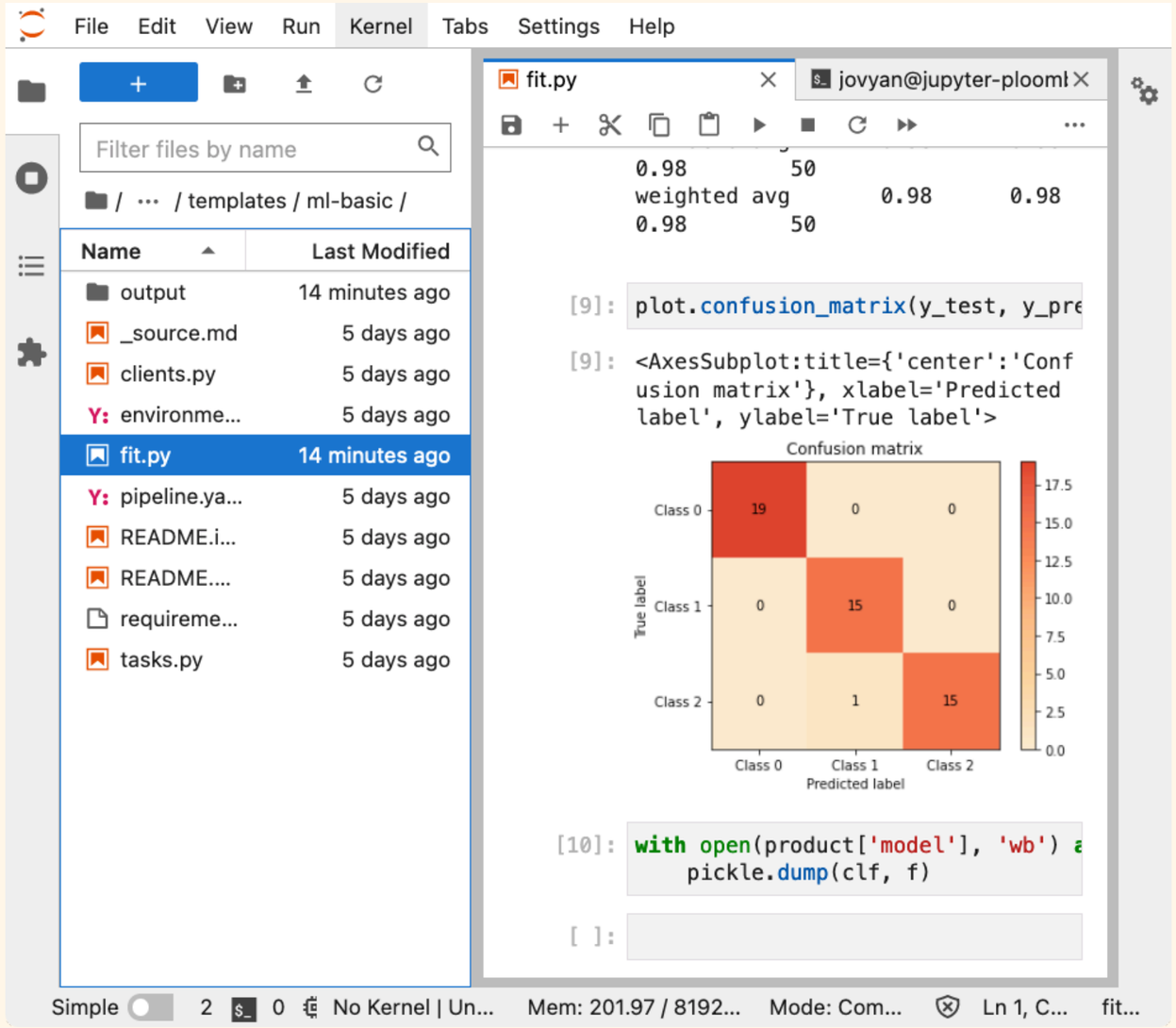
Can we escape the mess? [1] [2]

[1] Image source: Unsplash

[2] Proof: Evidation's use case (ploomber.io/blog/evidation)

Jupyter: a format and a platform

```
{
  "metadata" : {
    "kernel_info": {
        "name" : "name of the kernel"
    },
    "language_info": {
        "name" : "programming language",
        "version": "version of the language",
    }
  },
  "nbformat": 4,
  "nbformat_minor": 0,
  "cells" : [
  ],
}
```



Schema definition: nbformat

Trick: Swap the format! [3]

1. Better `git` integration
2. Interoperability with other IDEs (VSCode, Spyder, PyCharm)

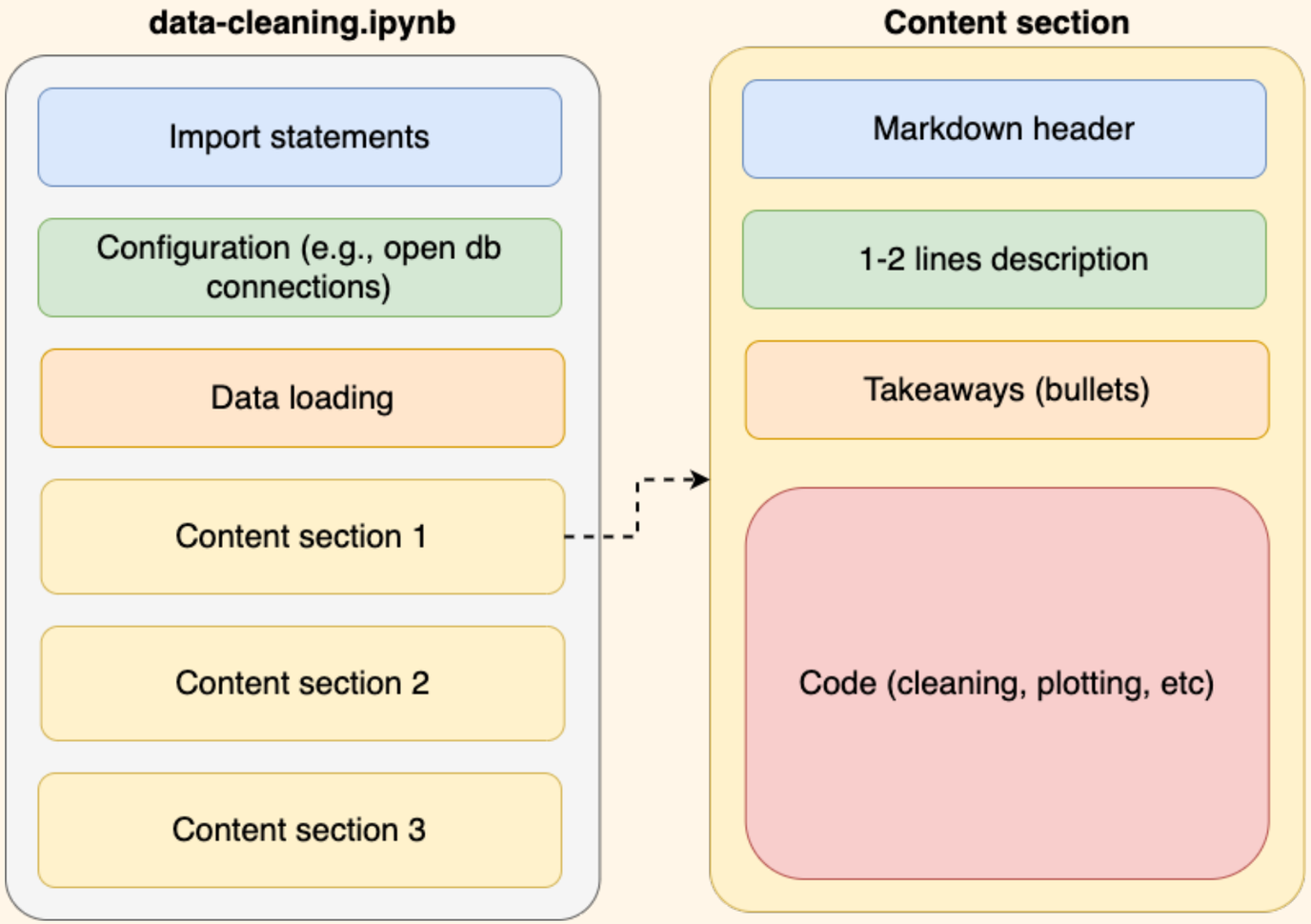
```python
# data-cleaning.py
import pandas as pd

# %%
# one cell
df = pd.read_csv('my-data.csv')

# %%
# another cell
df['new-column'] = df['column'] + 1
```

[3] Package: jupytext

Part I: Clean notebooks 🧼 [4]

[4] Recommended read: ploomber.io/blog/clean-nbs (On Writing Clean Notebooks)

Clean notebooks are short



Clean notebooks have immutable operations

Mutable: avoid! ⚠️

```
import pandas as pd

def add_one(df):
    df['zeros'] = df['zeros'] + 1

df = pd.DataFrame({'zeros': [0, 0, 0]})

# more code...

# a few dozen cells below...
add_one(df)

df
#    zeros
# 0      1
# 1      1
# 2      1
```

Immutable ✅

```
def add_one(df):
    # copying the data frame!
    another = df.copy()
    another['zeros'] = another['zeros'] + 1
    return another

df = pd.DataFrame({'zeros': [0, 0, 0]})

ones = add_one(df)

df
#    zeros
# 0      0
# 1      0
# 2      0
```

Clean notebooks lock dependencies 🔒

1. Package updates break code
2. It'll make it challenging to re-run the notebook

```
# requirements.lock.txt
package-a==1.1
package-b==2.4
...
package-z==0.4
```

After installing dependencies:

```
pip freeze > requirements.lock.txt
```

Re-create environment:

```
pip install -r requirements.lock.txt
```

Clean notebooks are formatted [5]

Before 👀

```
[ ]: df = pd.DataFrame({"x": np.random.rand(10),
     "y": np.random.rand(10),
             "x": np.random.rand(10),})
```

```
[ ]: something = True
     another = False

     if something
     \or another:
         print("at least one is True!")
```

After 🧼

```
[ ]: df = pd.DataFrame(
         {
             "x": np.random.rand(10),
             "y": np.random.rand(10),
             "x": np.random.rand(10),
         }
     )
```

```
[ ]: something = True
     another = False

     if something or another:
         print("at least one is True!")
```

Command: `soorgeon clean nb.ipynb`

---

[5] Package Soorgeon (by Ploomber)

Clean notebooks: split **logic** from **narrative**

1. **Logic:** functions, classes
2. **Narrative:** plots, chain of transformations

Define logic in a file

```python
# transformations.py
def add_one(series):
    return series + 1
```

```
[1]: import pandas as pd
     import numpy as np

[2]: # import from transformations.py
     from transformations import add_one

[3]: df = pd.DataFrame({"x": np.random.rand(10)})

[4]: df["y"] = add_one(df.x)

[5]: df.head(3)
```

[5]:

	x	y
0	0.010910	1.010910
1	0.201756	1.201756
2	0.404869	1.404869

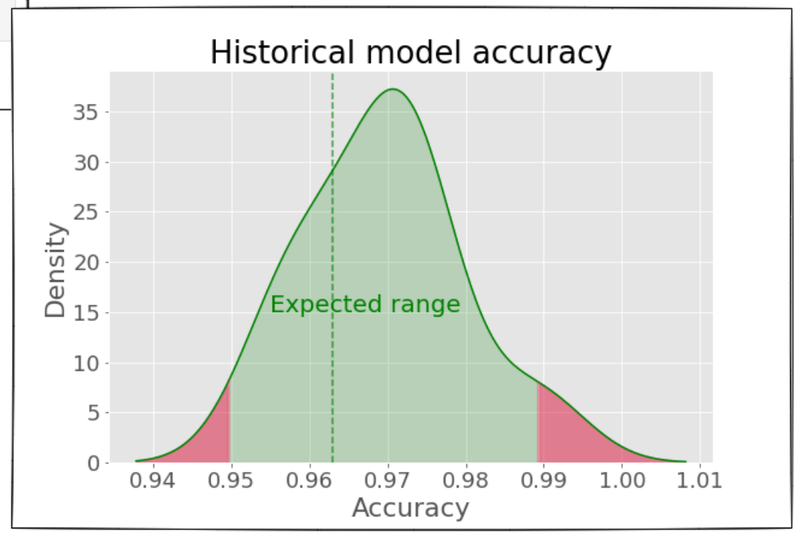
Part II: Testing notebooks 🧪 [6]

[6] Recommended read: ploomber.io/blog/ci-for-ds and ploomber.io/blog/ml-testing-i

Testing notebooks: cell outputs [7]

```
[7]: y_pred = clf.predict(X_test)

[8]: acc = accuracy_score(y_test,
                          y_pred)
     print(f'{acc:.3f}')

     0.963
```

```
nbsnapshot test nb.ipynb

Testing nb.ipynb...
Checking "accuracy"...
Value within expected range!
No issues found.
```



[7] Package: nbsnapshot (by Ploomber)

Testing notebooks: output artifacts [8]

- Example: data quality tests

```
# assume your notebook produces df
def check_data_quality(df):
    # no nas
    assert df['column'].isna().sum() == 0
    # no negative numbers
    assert (df['column'] < 0).sum() == 0
```

[8] Package: Ploomber

Testing notebooks: definitions

Split narrative from logic [9]

```python
from transformations import add_one

def test_add_one():
    assert add_one(1, 2) == 3
```

Embedded logic [10]

```python
from testbook import testbook


@testbook('path/to/nb.ipynb')
def test_func(tb):
    add_one = tb.get("add_one")

    assert add_one(1, 2) == 3
```

---

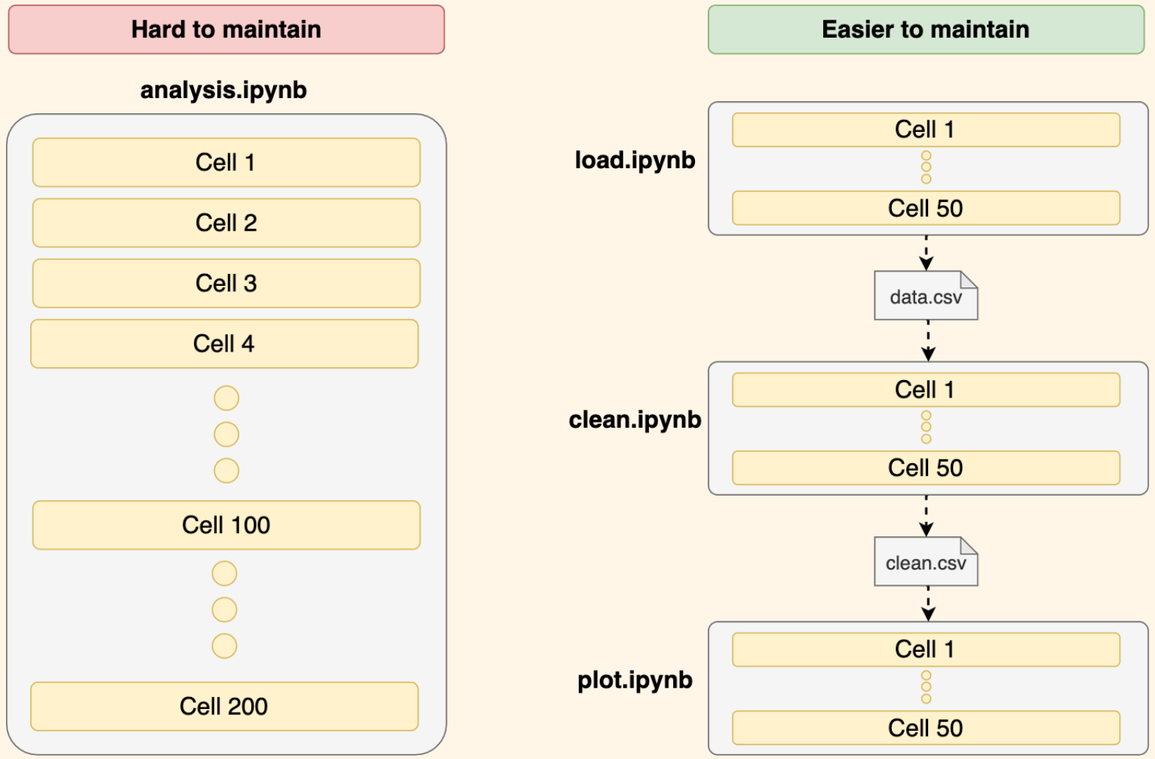
[9] Package: pytest

[10] Package: testbook

Part III: Building data pipelines 🛠️ [11]

[11] Recommended read ploomber.io/blog/clean-pipelines

Building data pipelines [12]



[12] Build pipelines: Ploomber. Refactoring existing notebooks: Soorgeon (by Ploomber)

Learn more

- Free online course:
  notebooks.academy
- Slides (with links):
  blancas.io/talks/pydata-chi-22.pdf

Thanks! 🎉

Reach out: eduardo@ploomber.io