

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

**Eduardo Blancas**

Ploomber Co-Founder, CEO



**ploomber.io**



# Can we escape the mess? <sup>1</sup> <sup>2</sup>

---

<sup>1</sup> Image source: [Unsplash](#)

<sup>2</sup> Proof: [Evidation's use case \(ploomber.io/blog/evidation\)](https://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](#)

The screenshot displays the JupyterLab environment. On the left, a file browser shows the directory structure, including a file named `fit.py` which is selected. The main area on the right contains a code editor with the following Python code:

```
[9]: plot.confusion_matrix(y_test, y_pre  
[9]: <AxesSubplot:title={'center': 'Conf  
      usion matrix'}, xlabel='Predicted  
      label', ylabel='True label'>  
      Confusion matrix  
      True label  
      Class 0  
      Class 1  
      Class 2  
      Class 0  
      Class 1  
      Class 2  
      Predicted label  
      19 0 0  
      0 15 0  
      0 1 15  
      17.5  
      15.0  
      12.5  
      10.0  
      7.5  
      5.0  
      2.5  
      0.0  
[10]: with open(product['model'], 'wb') as  
      pickle.dump(clf, f)  
[ ]:
```

Below the code editor, a console output shows the results of the confusion matrix calculation:

```
0.98      50  
weighted avg      0.98      0.98  
0.98      50
```

The bottom status bar indicates the current state: Simple, 2, 0, No Kernel | Un..., Mem: 201.97 / 8192..., Mode: Com..., Ln 1, C... fit...

# Trick: Swap the format! <sup>3</sup>

1. Better git integration

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

2. Interoperability with other IDEs (VSCode, Spyder, PyCharm)

```
# %%  
# one cell  
df = pd.read_csv('my-data.csv')  
  
# %%  
# another cell  
df['new-column'] = df['column'] + 1
```

---

<sup>3</sup> Package: [jupyter](#)

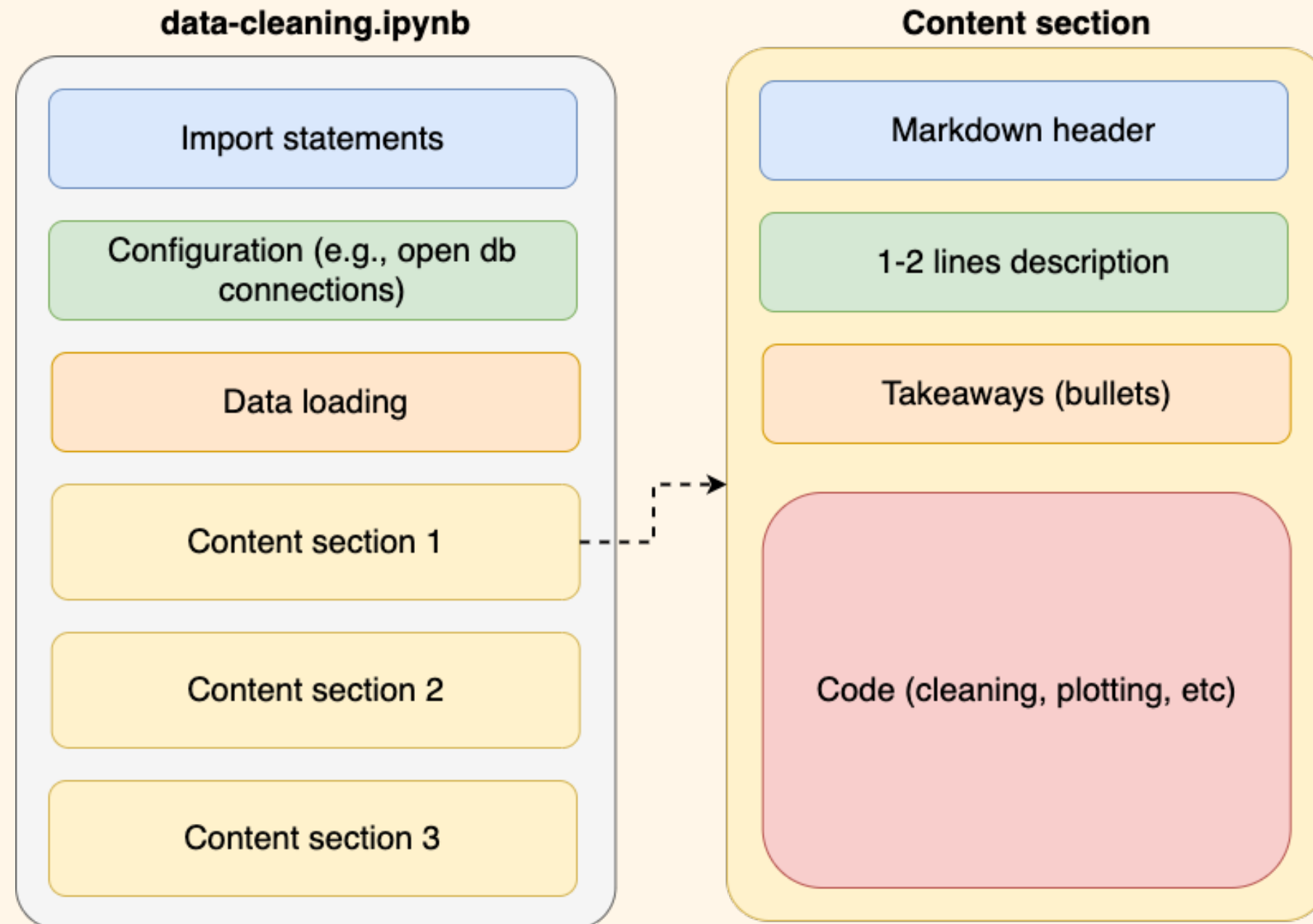


# Part I: Clean notebooks <sup>4</sup>

---

<sup>4</sup> Recommended read: [ploomber.io/blog/clean-nbs](https://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 <sup>5</sup>

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:** `sourgeon clean nb.ipynb`

---

<sup>5</sup> Package [Sourgeon](#) (by Ploomber)

# Clean notebooks: split logic from narrative

1. **Logic:** functions, classes

2. **Narrative:** plots, chain of transformations

Define logic in a file

```
# 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 <sup>6</sup>

---

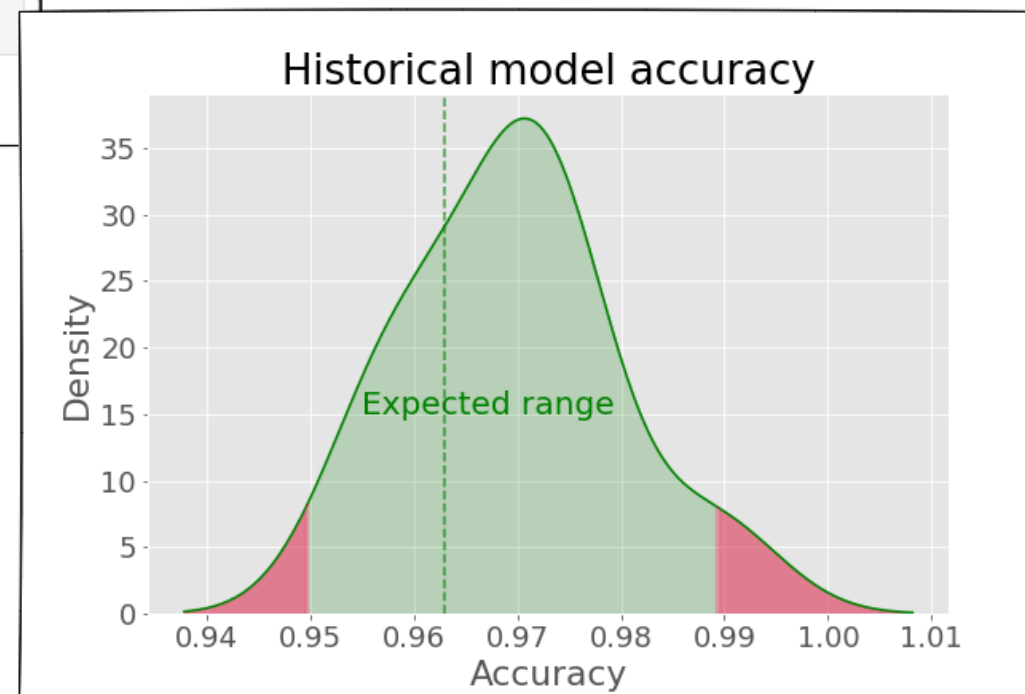
<sup>6</sup> Recommended read: [ploomber.io/blog/ci-for-ds](https://ploomber.io/blog/ci-for-ds) and [ploomber.io/blog/ml-testing-i](https://ploomber.io/blog/ml-testing-i)

# Testing notebooks: cell outputs <sup>7</sup>

```
[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.
```



<sup>7</sup> Package: [nbsnapshot](#) (by Ploomber)

# Testing notebooks: output artifacts <sup>8</sup>

## ◆ 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
```

---

<sup>8</sup> Package: [Ploomber](#)

# Testing notebooks: definitions

## Split narrative from logic <sup>9</sup>

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

## Embedded logic <sup>10</sup>

```
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
```

---

<sup>9</sup> Package: [pytest](#)

<sup>10</sup> Package: [testbook](#)

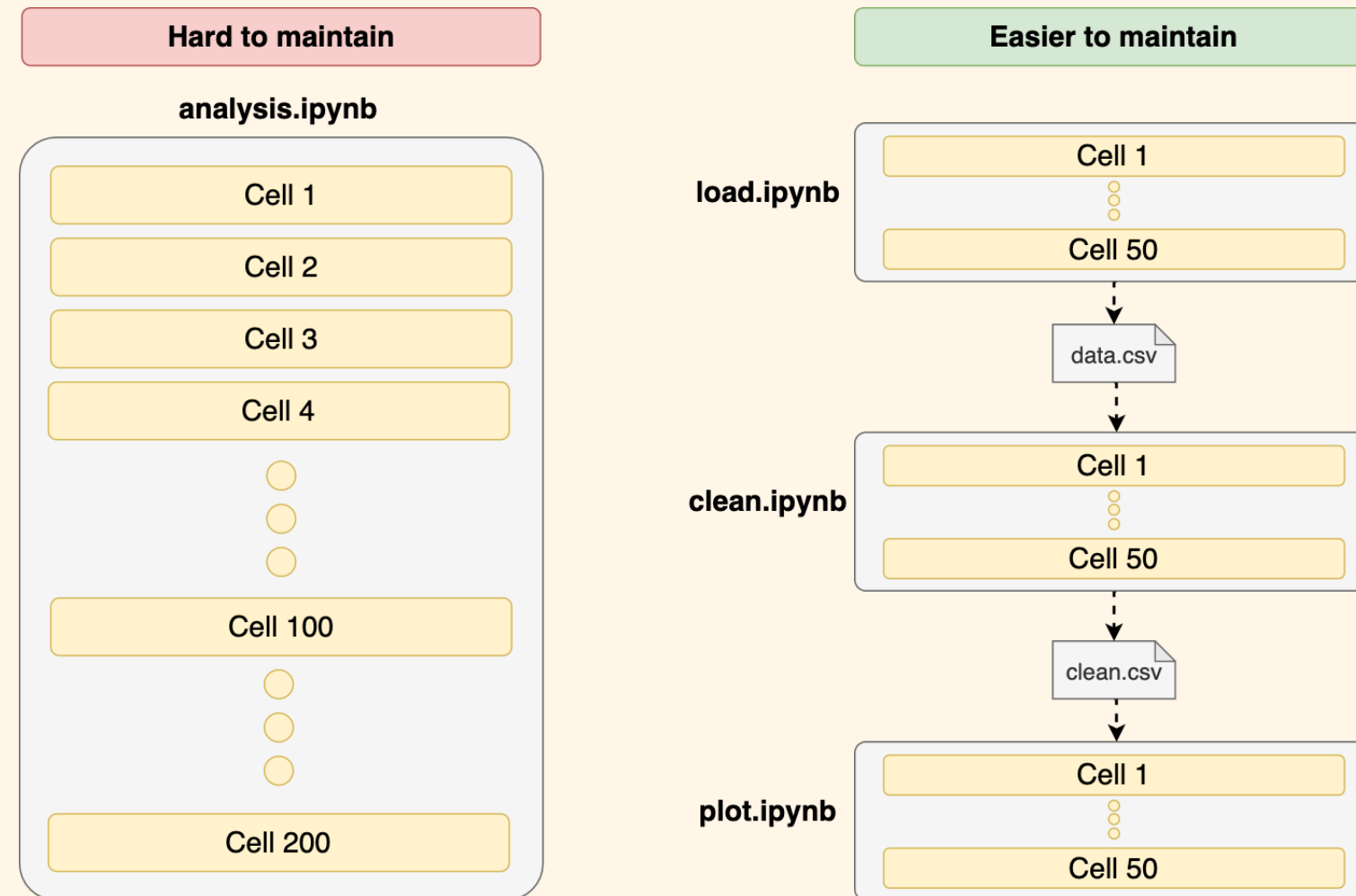
# Part III: Building data pipelines <sup>11</sup>

---

<sup>11</sup> Recommended read [ploomber.io/blog/clean-pipelines](https://ploomber.io/blog/clean-pipelines)



# Building data pipelines <sup>12</sup>



<sup>12</sup> Build pipelines: [Ploomber](#). Refactoring existing notebooks: [Soorgeon](#) (by Ploomber)

# Learn more

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



Thanks! 🎉

Reach out: [eduardo@ploomber.io](mailto:eduardo@ploomber.io)