Going from notebooks

to scalable systems

github.com/catherinenelson1/from_notebooks_to_scalable

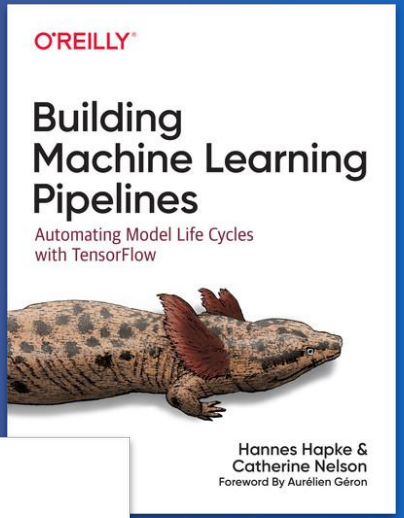
About me

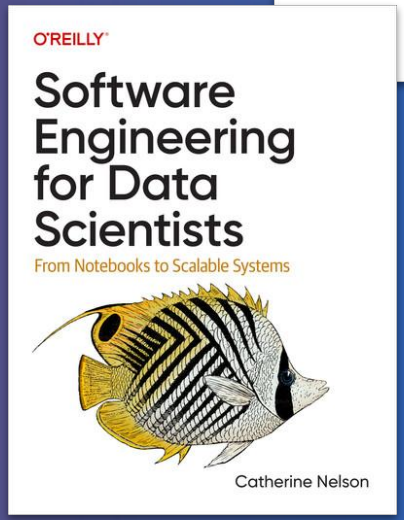
~10 years in data science with a strong focus on ML

Author of two O'Reilly books

GenAI and DevRel consultant/contractor

Principal Data Scientist at SAP Concur





`github.com/catherinenelson1/from_notebooks_to_scalable`

There are no standard techniques

github.com/catherinenelson1/from_notebooks_to_scalable

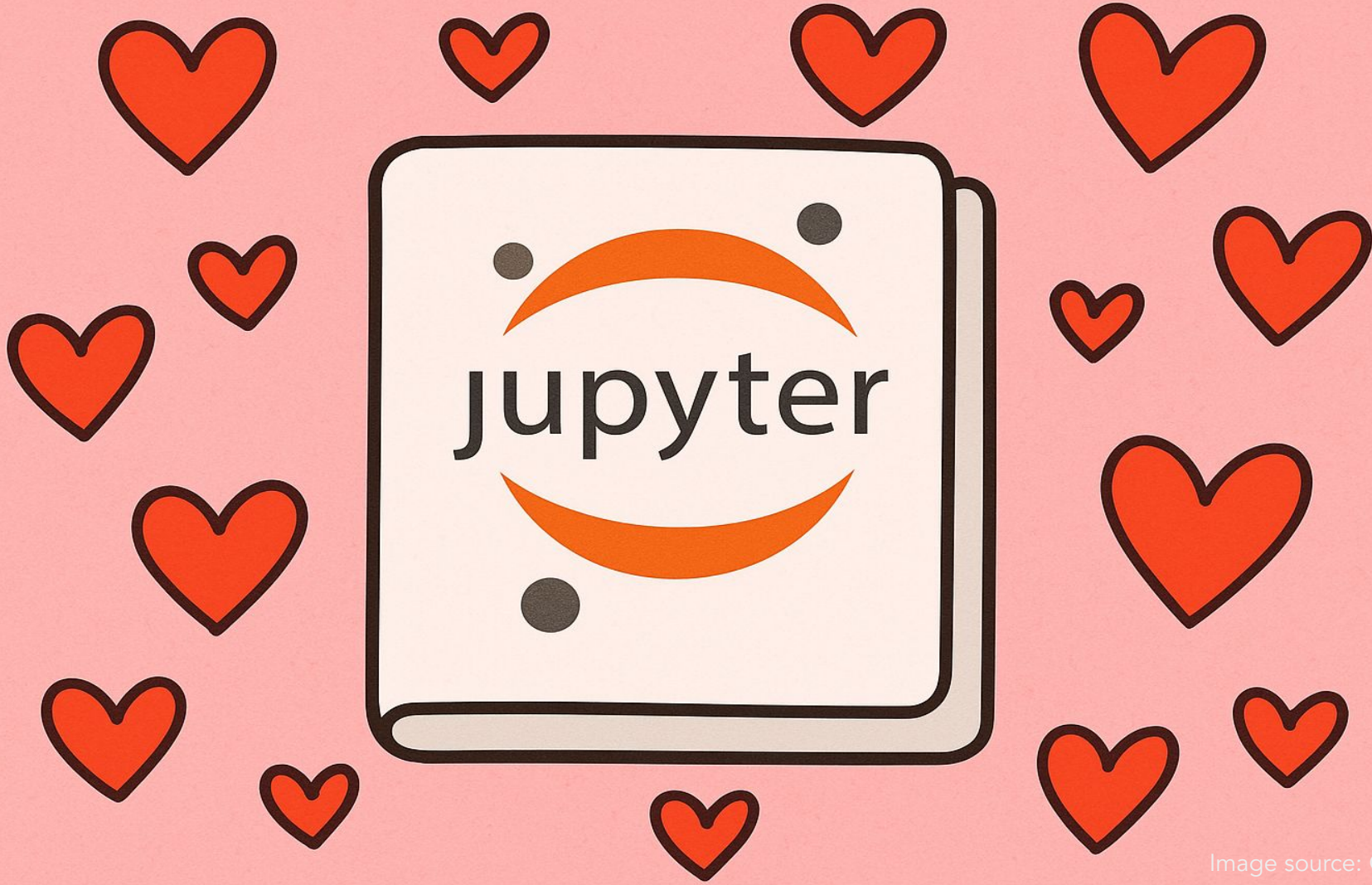
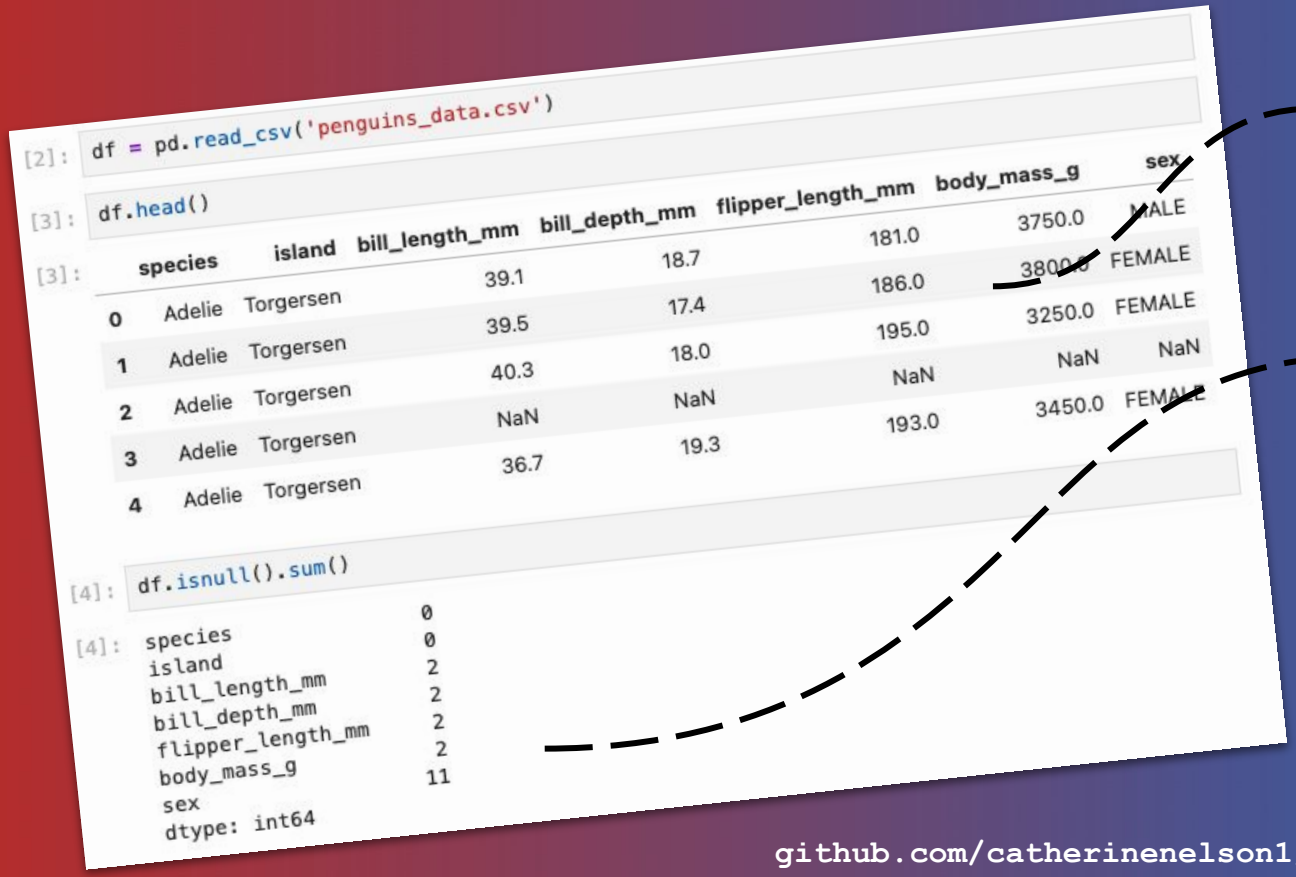


Image source: ChatGPT

Why notebooks are great



```
[2]: df = pd.read_csv('penguins_data.csv')
```

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

[3]:

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE

```
[4]: df.isnull().sum()
```

```
[4]: species             0
     island              0
     bill_length_mm      2
     bill_depth_mm       2
     flipper_length_mm   2
     body_mass_g         2
     sex                11
     dtype: int64
```

Display your data

View results of running code

github.com/catherinenelson1/from_notebooks_to_scalable

Why notebooks are great

Mix text and code

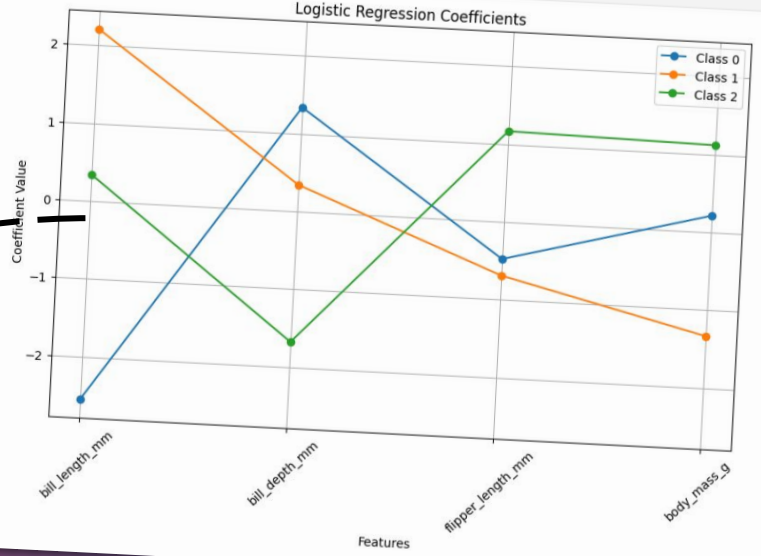
Inline plots

Explore the feature importance

Plot logistic regression coefficients

[32]:
```
plt.figure(figsize=(10, 6))
for i in range(coefficients.shape[0]):
    plt.plot(np.arange(coefficients.shape[1]), coefficients[i], marker='o', label=f'Class {i}')

plt.xticks(np.arange(coefficients.shape[1]), ['bill_length_mm', 'bill_depth_mm', 'flipper_length_mm', 'body_mass_g'], rotation=45)
plt.xlabel('Features')
plt.ylabel('Coefficient Value')
plt.title('Logistic Regression Coefficients')
plt.legend()
plt.grid(True)
plt.show()
```

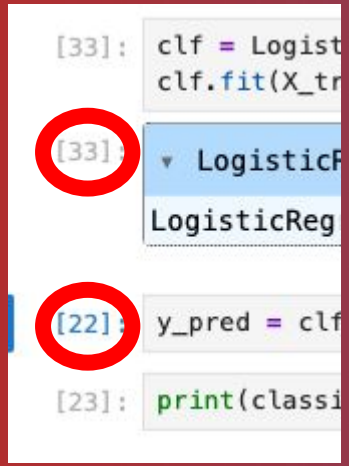


github.com/catherinenelson1/from_notebooks_to_scalable

Less great things about notebooks

Hard to automate

Easy to leave in code that you're not using



github.com/catherinenelson1/from_notebooks_to_scalable

It's tough to scale notebooks

github.com/catherinenelson1/from_notebooks_to_scalable

When to go from a notebook to standalone scripts?

github.com/catherinenelson1/from_notebooks_to_scalable

Change your mindset!

github.com/catherinenelson1/from_notebooks_to_scalable

An example project

github.com/catherinenelson1/from_notebooks_to_scalable

Let's classify penguins!

By Andrew Shiva / Wikipedia, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=46714803

Penguins can be hard to tell apart

Given data on flipper size, bill length, bill depth, body mass, build a model to predict penguin species

Features: flipper size, bill length, body mass

Labels: species

Put the model in production

github.com/catherinenelson1/from_notebooks_to_scalable

Download dataset

[2]:
```
# dataset from https://github.com/mwaskom/seaborn-data/blob/master/penguins.csv
import requests

url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/refs/heads/master/penguins.csv'
response = requests.get(url)

with open('penguins_data.csv', 'wb') as file:
    file.write(response.content)
```

Explore and clean the data

[2]:
```
import pandas as pd
```

[3]:
```
df = pd.read_csv('penguins_data.csv')
```

[4]:
```
df.head()
```

[4]:

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE

[5]:
```
df.isnull().sum()
```

[5]:
```
species               0
island                0
bill_length_mm        2
bill_depth_mm         2
flipper_length_mm     2
body_mass_g           2
sex                  11
dtype: int64
```

github.com/catherinenelson1/from_notebooks_to_scalable

[6]:
```
# drop rows with missing values
df = df.dropna(subset=['species', 'bill_length_mm', 'bill_depth_mm', 'flipper_length_mm', 'body_mass_g'])
```

[7]:
```
len(df)
```

[7]:
```
342
```

[8]:
```
# select only columns with relevant features
df = df[['species', 'bill_length_mm', 'bill_depth_mm', 'flipper_length_mm', 'body_mass_g']]
```

[9]:
```
df.head()
```

[9]:

	species	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g
0	Adelie	39.1	18.7	181.0	3750.0
1	Adelie	39.5	17.4	186.0	3800.0
2	Adelie	40.3	18.0	195.0	3250.0
4	Adelie	36.7	19.3	193.0	3450.0
5	Adelie	39.3	20.6	190.0	3650.0

[10]:
```
df['species'].value_counts()
```

[10]:
```
species
Adelie       151
Gentoo       123
Chinstrap     68
Name: count, dtype: int64
```

[11]:
```
features = df[['bill_length_mm', 'bill_depth_mm', 'flipper_length_mm', 'body_mass_g']].to_numpy()
values = df['species'].to_numpy()
```

github.com/catherinenelson1/from_notebooks_to_scalable

Scale and encode the data

[12]:
```
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
```

[13]:
```
features
```

[13]:
```
array([[  39.1,   18.7,  181. , 3750. ],
       [  39.5,   17.4,  186. , 3800. ],
       [  40.3,   18. ,  195. , 3250. ],
       ...,
       [  50.4,   15.7,  222. , 5750. ],
       [  45.2,   14.8,  212. , 5200. ],
       [  49.9,   16.1,  213. , 5400. ]], shape=(342, 4))
```

[14]:
```
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)
```

[15]:
```
# one-hot encode the species
encoder = LabelEncoder()
values_encoded = encoder.fit_transform(values)
```

[16]:
```
values_encoded
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
```

[17]:
```
# split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features_scaled, values_encoded, test_size=0.2, random_state=42)
```

github.com/catherinenelson1/from_notebooks_to_scalable

Try out some models

[18]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
```

[19]:
```
clf = LogisticRegression()
clf.fit(X_train, y_train)
```

[19]:
```
▾ LogisticRegression
LogisticRegression()
```

[20]:
```
y_pred = clf.predict(X_test)
```

[21]:
```
print(classification_report(y_test, y_pred, target_names=['Adelie', 'Gentoo', 'Chinstrap']))
```

```
              precision    recall  f1-score   support

      Adelie       1.00      0.97      0.99        35
      Gentoo       0.92      1.00      0.96        12
   Chinstrap       1.00      1.00      1.00        22

    accuracy                           0.99        69
   macro avg       0.97      0.99      0.98        69
weighted avg       0.99      0.99      0.99        69
```

Make a prediction on new data

[32]:
```
encoder.inverse_transform(clf.predict([[40, 17, 190, 3500]]))[0]
```

[32]:
```
'Gentoo'
```

github.com/catherinenelson1/from_notebooks_to_scalable

Tools to extract code

github.com/catherinenelson1/from_notebooks_to_scalable

Nbconvert

```
% pip install nbconvert

% jupyter nbconvert --to script penguins_notebook.ipynb
```

Nbconvert output

```
# ### Download dataset
# In[2]:

# dataset from https://github.com/mwaskom/seaborn-data/blob/master/penguins.csv
import requests

url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/refs/heads/master/penguins.csv'
response = requests.get(url)

with open('penguins_data.csv', 'wb') as file:
    file.write(response.content)

# ### Explore the data
# In[3]:

import pandas as pd
```

github.com/catherinenelson1/from_notebooks_to_scalable

Using the nbconvert output

```
import requests
import pandas as pd

if __name__ == "__main__":
    # dataset from https://github.com/mwaskom/seaborn-data/blob/master/penguins.csv
    url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/refs/heads/master/penguins.csv'
    response = requests.get(url)

    with open('penguins_data.csv', 'wb') as file:
        file.write(response.content)

    # ### Explore the data
    df = pd.read_csv('penguins_data.csv')

    df.head()

    df.isnull().sum()
```

github.com/catherinenelson1/from_notebooks_to_scalable

Jupytext

```
% pip install jupytext

% jupytext --to py:percent penguins_notebook.ipynb
```

Jupytext output

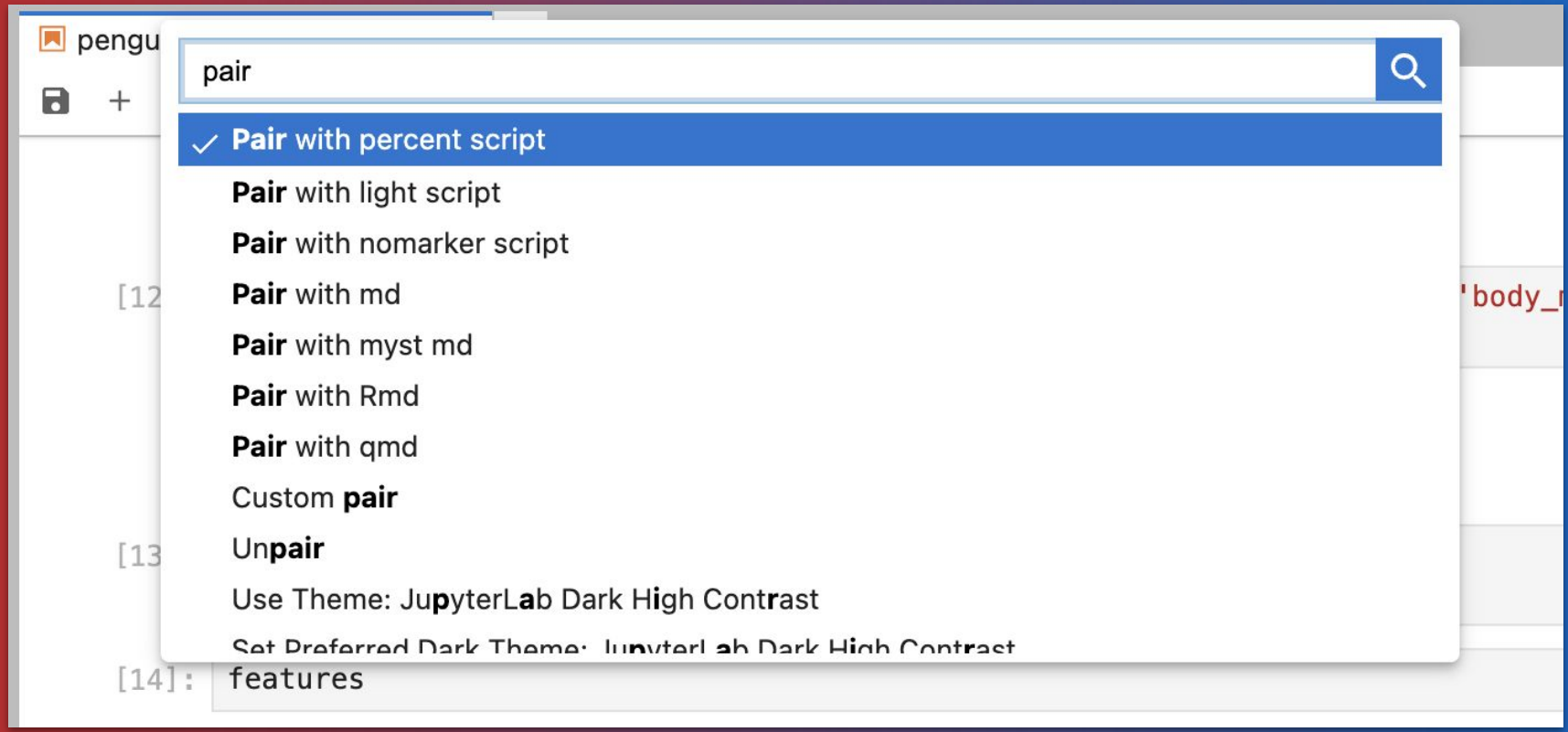
```
# %% [markdown]
# ## Download dataset

# %%
# dataset from https://github.com/mwaskom/seaborn-data/blob/master/penguins.csv
import requests

url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/refs/heads/master/penguins.csv'
response = requests.get(url)

with open('penguins_data.csv', 'wb') as file:
    file.write(response.content)
```

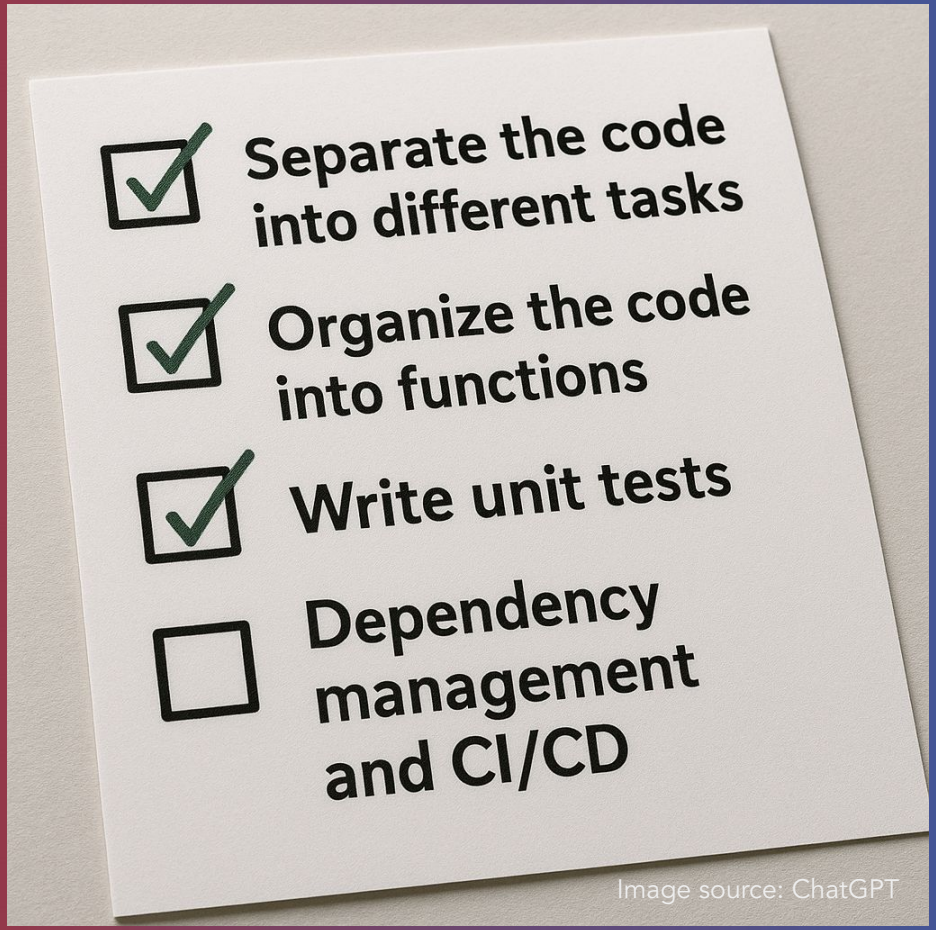
Jupytext



github.com/catherinenelson1/from_notebooks_to_scalable

A notebook refactoring journey

github.com/catherinenelson1/from_notebooks_to_scalable



github.com/catherinenelson1/from_notebooks_to_scalable

Identify the steps in your notebook

What are the steps you need to keep?

Write a function stub for each step

For each function, what are the inputs and outputs?

Add in the code

Think about how it could go wrong and write a test

github.com/catherinenelson1/from_notebooks_to_scalable

Steps in the notebook

Load penguin data

Clean missing values

Scale/encode features and labels

Train model

Predict on new penguins

github.com/catherinenelson1/from_notebooks_to_scalable

Steps in the notebook

```python
def download_data():
    pass

def clean_data():
    pass

def preprocess_data():
    pass

def train_model():
    pass

def predict_new_data():
    pass
```

github.com/catherinenelson1/from_notebooks_to_scalable

Add inputs and returns

```python
def download_data(file_path):
    # download the dataset if it doesn't exist already
    # save it to file_path
    pass

def clean_data(file_path):
    # load the dataset
    # drop rows with missing values
    # return numpy arrays for features and labels
    pass
```

github.com/catherinenelson1/from_notebooks_to_scalable

Add inputs and returns

```python
def preprocess_data(features, labels):
    # accepts numpy arrays
    # encode categorical variables
    # scale numerical features
    # split the data into train/test features and labels
    # return numpy arrays for X_train, X_test, y_train, y_test
    pass

def train_model(X_train, y_train):
    # train a model and save it
    pass

def predict_new_data(X_new):
    # load the model and make predictions
    # return a string of the predicted class
    pass
```

github.com/catherinenelson1/from_notebooks_to_scalable

Then add the code

```python
import pandas as pd

def clean_data(file_path):
    # drop rows with missing values
    # return numpy arrays for features and labels
    df = pd.read_csv(file_path)
    df = df.dropna(subset=['species', 'bill_length_mm', 'bill_depth_mm',
                           'flipper_length_mm', 'body_mass_g'])
    features = df[['bill_length_mm', 'bill_depth_mm',
                   'flipper_length_mm', 'body_mass_g']].to_numpy()
    labels = df['species'].to_numpy()
    return features, labels
```

github.com/catherinenelson1/from_notebooks_to_scalable

Test the code

```
def create_test_data():
    test_data = pd.DataFrame({
        'species': ['Adelie', 'Gentoo', 'Chinstrap'],
        'bill_length_mm': [39.1, 46.5, 49.7, None],
        'bill_depth_mm': [18.7, 14.3, 16.0, None],
        'flipper_length_mm': [181, 217, 193, None],
        'body_mass_g': [3750, 5200, 3800, None],
    })
    test_data.to_csv('test_penguins.csv', index=False)
```

github.com/catherinenelson1/from_notebooks_to_scalable

Test the code

```python
def test_clean_data():
    # Arrange - create the data
    create_test_data()

    # Act - test the function
    features, labels = clean_data('test_penguins.csv')

    # Assert - check results
    assert features.shape == (3, 4)
    assert labels.shape == (3,)
    assert 'Adelie' in labels
    assert 'Gentoo' in labels
    assert 'Chinstrap' in labels

    # Clean up
    os.remove('test_penguins.csv')
```

github.com/catherinenelson1/from_notebooks_to_scalable

Add the code

```python
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split

def preprocess_data(features, labels):
    # encode categorical variables
    # scale numerical features
    # split the data into train/test features and labels
    scaler = StandardScaler()
    features_scaled = scaler.fit_transform(features)
    encoder = LabelEncoder()
    labels_encoded = encoder.fit_transform(labels)
    X_train, X_test, y_train, y_test = train_test_split(features_scaled,
                                                        labels_encoded,
                                                        test_size=0.2,
                                                        random_state=42)

    return X_train, X_test, y_train, y_test
```

github.com/catherinenelson1/from_notebooks_to_scalable

Test the code

```python
import numpy as np
from penguins_refactored_step2 import preprocess_data


def test_preprocess_data():
    features = np.array([[1.0, 2.0, 3.0, 4.0],
                         [5.0, 6.0, 7.0, 8.0],
                         [9.0, 10.0, 11.0, 12.0]])
    labels = np.array(['Adelie', 'Gentoo', 'Chinstrap'])

    X_train, X_test, y_train, y_test = preprocess_data(features, labels)

    assert X_train.shape == (2, 4)
    assert X_test.shape == (1, 4)
    assert y_train.shape == (2,)
    assert y_test.shape == (1,)

    assert y_train[0] in [0, 1, 2]
    assert y_test[0] in [0, 1, 2]
```

github.com/catherinenelson1/from_notebooks_to_scalable

Steps in the notebook

```python
def load_data():
    pass

def clean_data():
    pass

def preprocess_data():
    pass

def train_model():
    pass

def predict_new_data():
    pass
```

github.com/catherinenelson1/from_notebooks_to_scalable

All the rest are in the repo

main from_notebooks_to_scalable / penguins_refactored.py ↑ Top

Code Blame 93 lines (69 loc) · 2.95 KB Raw

```
52  def train_model(X_train, y_train, X_test, y_test, model_filename):
53      # train a model and save it
54      clf = LogisticRegression()
55      clf.fit(X_train, y_train)
56      # print the model's accuracy
57      accuracy = clf.score(X_test, y_test)
58      print(f'Model accuracy: {accuracy:.2f}')
59      joblib.dump(clf, model_filename)
60
```

github.com/catherinenelson1/from_notebooks_to_scalable

Run the pipeline

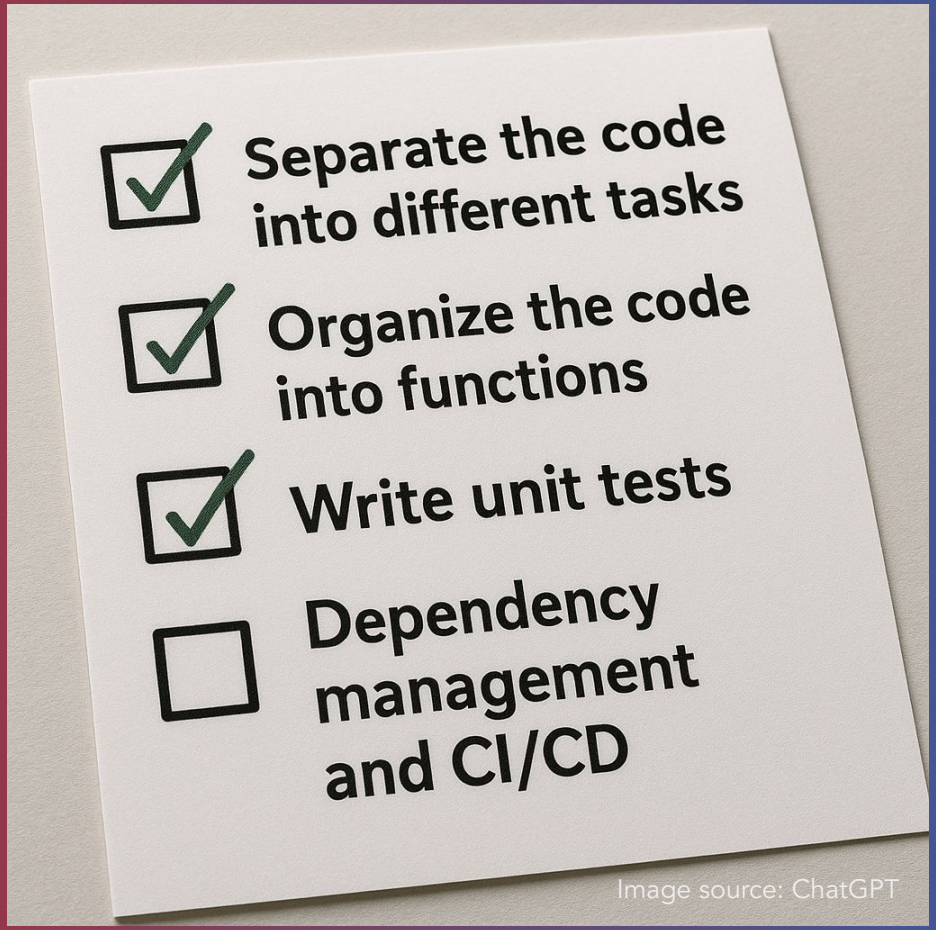
```python
def run_training_pipeline(data_file_path, encoder_filename, model_filename):
    # load data
    download_data(data_file_path)

    # clean data
    features, labels = clean_data(data_file_path)

    # preprocess data
    X_train, X_test, y_train, y_test = preprocess_data(features, labels, encoder_filename)

    # train model
    train_model(X_train, y_train, X_test, y_test, model_filename)
```

github.com/catherinenelson1/from_notebooks_to_scalable



github.com/catherinenelson1/from_notebooks_to_scalable

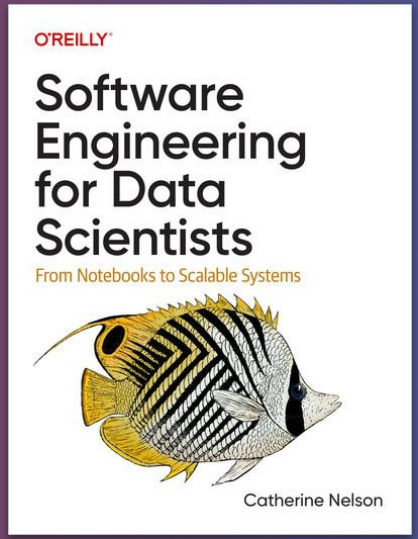
You'll need some key skills

github.com/catherinenelson1/from_notebooks_to_scalable

Pick the right mindset and tools for your code goals

github.com/catherinenelson1/from_notebooks_to_scalable

Thank you!



Book signing:
4pm today @ PSF Lounge

github.com/catherinenelson1/from_notebooks_to_scalable