10 Little-Known Python Libraries That Will Make You Feel Like a Data Wizard

In this article, I will introduce you to 10 little-known Python libraries every data scientist should know.

Josep Ferrer

Image by Author

Python remains a key resource in any data scientist's toolset. One of its main advantages is the limitless amount of available libraries to enhance your workflows: Pandas, Numpy, Matplotlib, Scikit-learn... the list goes on and on.

But beyond these well-known libraries are a wealth of lesser-knowns that

supercharge your workflow. In this article, I will introduce you to 10 little-known Python libraries every data scientist should know.

1. Altair: Declarative Visualization Made Simple

Overview:

Altair is a declarative statistical visualization library focusing on simplicity and expressiveness. Unlike Matplotlib, it minimizes boilerplate code and emphasizes interactive charts.

Strengths:

- Intuitive syntax with minimal code.
- Interactive visualizations by default.
- Built-in support for statistical transformations.

Code example:

```
import altair as alt
import pandas as pd

data = pd.DataFrame({
    'x': range(10),
    'y': [val**2 for val in range(10)]
})

chart = alt.Chart(data).mark_circle(size=60).encode(
    x='x',
    y='y',
    color=alt.value('blue')
)
chart.show()
```

2. DuckDB: High-Performance SQL OLAP

Overview:

DuckDB is an in-process SQL OLAP database optimized for analytical workloads, allowing seamless integration with Python tools like Pandas and Jupyter.

Strengths:

- Incredibly fast for large datasets.
- Requires no separate server, running in-process.
- Simple integration with existing workflows.

Code example:

```
import duckdb

# Query a Pandas DataFrame
import pandas as pd
data = pd.DataFrame({'a': [1, 2, 3], 'b': [4, 5, 6]})
result = duckdb.query("SELECT a, b, a + b AS total FROM data").to_df()
print(result)
```

H3: Uber's Hexagonal Grid System

Overview:

H3 is an open-source library for spatial indexing that partitions the globe into consistent hexagonal cells, simplifying geospatial analysis.

Strengths:

- Uniform cell sizes for consistent metrics.
- Ideal for proximity searches, clustering, and spatial queries.
- Simplifies handling complex geospatial datasets.

Code example:

```
import h3
lat, lon = 37.7749, -122.4194
resolution = 9
hex_id = h3.geo_to_h3(lat, lon, resolution)
print(hex_id)
```

---

4. Ydata Profiling: Automated Data Insights

Overview:

Ydata Profiling automates dataset exploration by generating detailed HTML reports that highlight distributions, correlations, and data quality.

Strengths:

- Comprehensive insights without manual analysis.
- Visualizes missing values and distributions.
- Interactive and shareable reports.

Code example:

```
from ydata_profiling import ProfileReport

import pandas as pd
```

```
data = pd.read_csv('data.csv')
report = ProfileReport(data)
report.to_file('report.html')
```

---

5. Poetry: Streamlined Dependency Management

Overview:

Poetry simplifies dependency management and packaging, replacing the cumbersome requirements.txt approach with a single pyproject.toml file.

Strengths:

- Manages dependencies consistently.
- Simplifies virtual environment creation.
- Easy-to-use interface for project setup and deployment.

Code example:

```
# Create a new project
poetry new my_project

# Add a dependency
poetry add numpy
```

---

6. NetworkX: Analyzing Graph Data

Overview:

NetworkX is a versatile library for analyzing and visualizing graph structures, from social networks to transportation systems.

Strengths:

- Supports directed and undirected graphs.
- Extensive library of graph algorithms.
- Integrates with visualization tools.

Code example:

```
import networkx as nx
import matplotlib.pyplot as plt

G = nx.Graph()
G.add_edges_from([(1, 2), (2, 3), (3, 4), (1, 4)])

nx.draw(G, with_labels=True, node_color='skyblue', node_size=1500)
plt.show()
```

---

7. H2O.ai: Scalable Machine Learning

Overview:

H2O.ai provides tools for distributed machine learning, AutoML, and advanced algorithms, excelling in handling massive datasets.

Strengths:

- Distributed computing for scalability.
- Powerful AutoML capabilities.
- Supports advanced algorithms like GBM.

Code example:

```
import h2o
from h2o.automl import H2OAutoML

h2o.init()
data = h2o.import_file('data.csv')
train, test = data.split_frame(ratios=[.8])

aml = H2OAutoML(max_models=10, seed=1)
aml.train(y='target_column', training_frame=train)
print(aml.leaderboard.head())
```

---

8. PyCaret: Simplifying ML Pipelines

Overview:

PyCaret is an all-in-one library that streamlines model development, from preprocessing to evaluation, with minimal code.

Strengths:

- Unified API for over 25 ML algorithms.
- Fast and efficient baseline model creation.
- Integrated deployment tools.

Code example:

```
from pycaret.classification import setup, compare_models

from pycaret.datasets import get_data
data = get_data('iris')

clf = setup(data, target='species')
```

```
best_model = compare_models()
print(best_model)
```

9. Missingno: Visualizing Missing Data

Overview:

Missingno provides quick and intuitive visualizations for missing data, helping identify patterns and correlations.

Strengths:

- Visualizes missing data matrices and heatmaps.
- Easy integration with Pandas.
- Highlights relationships in missing data.

Code example:

```
import missingno as msno

collisions = pd.read_csv("https://raw.githubusercontent.com/ResidentMario/missin
msno.matrix(collisions.sample(250))
```

10. FlashText: Efficient Text Search and Replacement

Overview:

FlashText is a lightweight library for keyword extraction and replacement, outperforming regex in speed and simplicity for many use cases.

Strengths:

- Faster and more intuitive than regex.
- Ideal for NLP tasks like standardization.

Code example:

```
from flashtext import KeywordProcessor

keyword_processor = KeywordProcessor()
keyword_processor.add_keyword('Python', 'R')

text = "I love programming in Python"
new_text = keyword_processor.replace_keywords(text)
print(new_text)
```

By exploring these hidden gems, you'll unlock new efficiencies and capabilities, becoming a true data wizard. Try them out and supercharge your Python workflows!

Josep Ferrer is an analytics engineer from Barcelona. He graduated in physics engineering and is currently working in the data science field applied to human mobility. He is a part-time content creator focused on data science and technology. Josep writes on all things AI, covering the application of the ongoing explosion in the field.

More On This Topic

- Testing Like a Pro: A Step-by-Step Guide to Python's Mock Library
- Fake It Till You Make It: Generating Realistic Synthetic Customer Datasets

- Machine Learning Is Not Like Your Brain Part Two: Perceptrons vs Neurons
- SQL LIKE Operator Examples
- Build a ChatGPT-like Chatbot with These Courses
- How to MLOps like a Boss: A Guide to Machine Learning without Tears