

30 Python Libraries

to (Hugely) Boost

Your Data Science

Productivity



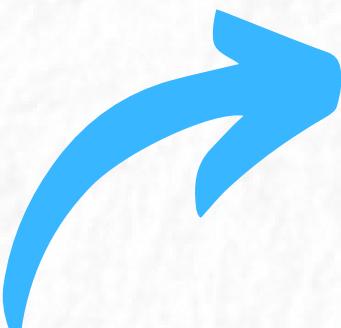
Avi Chawla



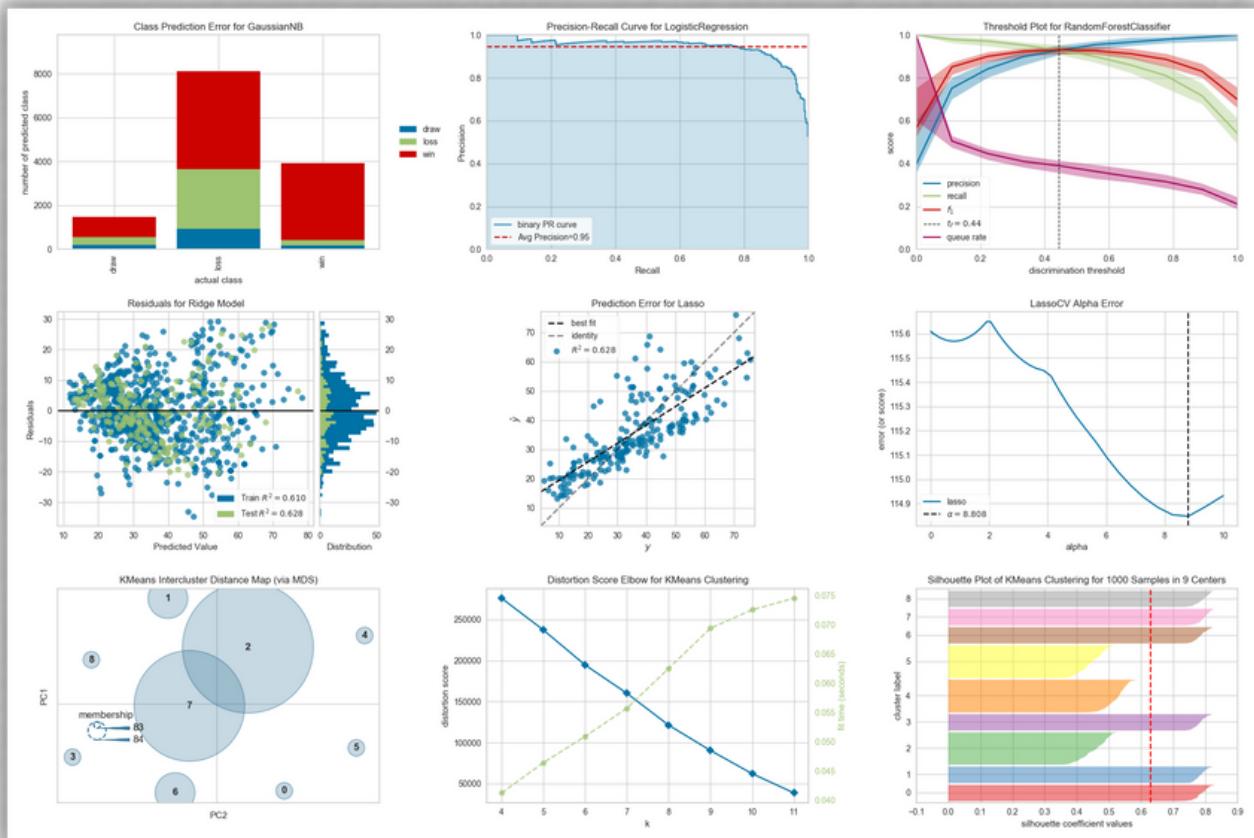
avichawla.substack.com

**Data Science is much
more than Pandas,
NumPy and
Sklearn.**

Here are **30 open-source
libraries** to upgrade
your data game.



1. YellowBrick

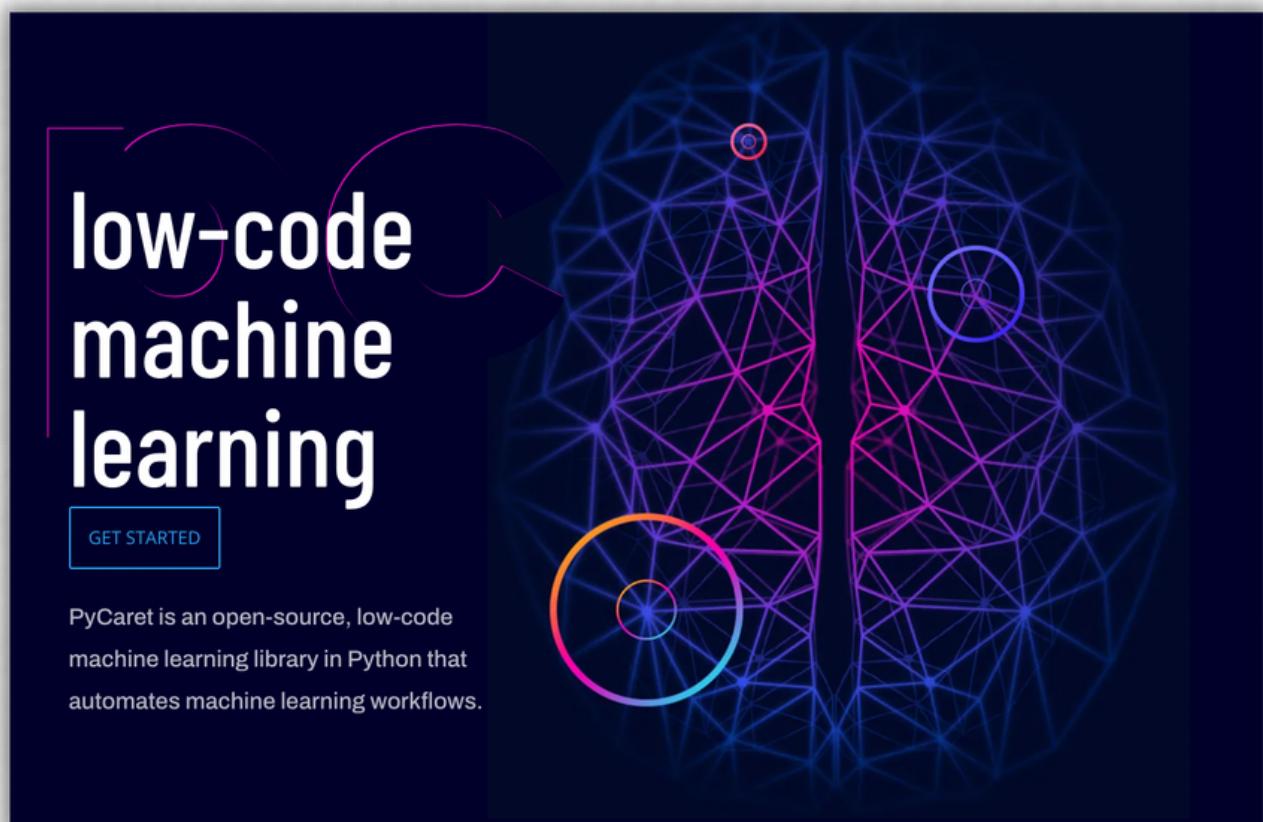


A suite of visualization
and diagnostic tools
for **faster model
selection.**

Matplotlib Sklearn



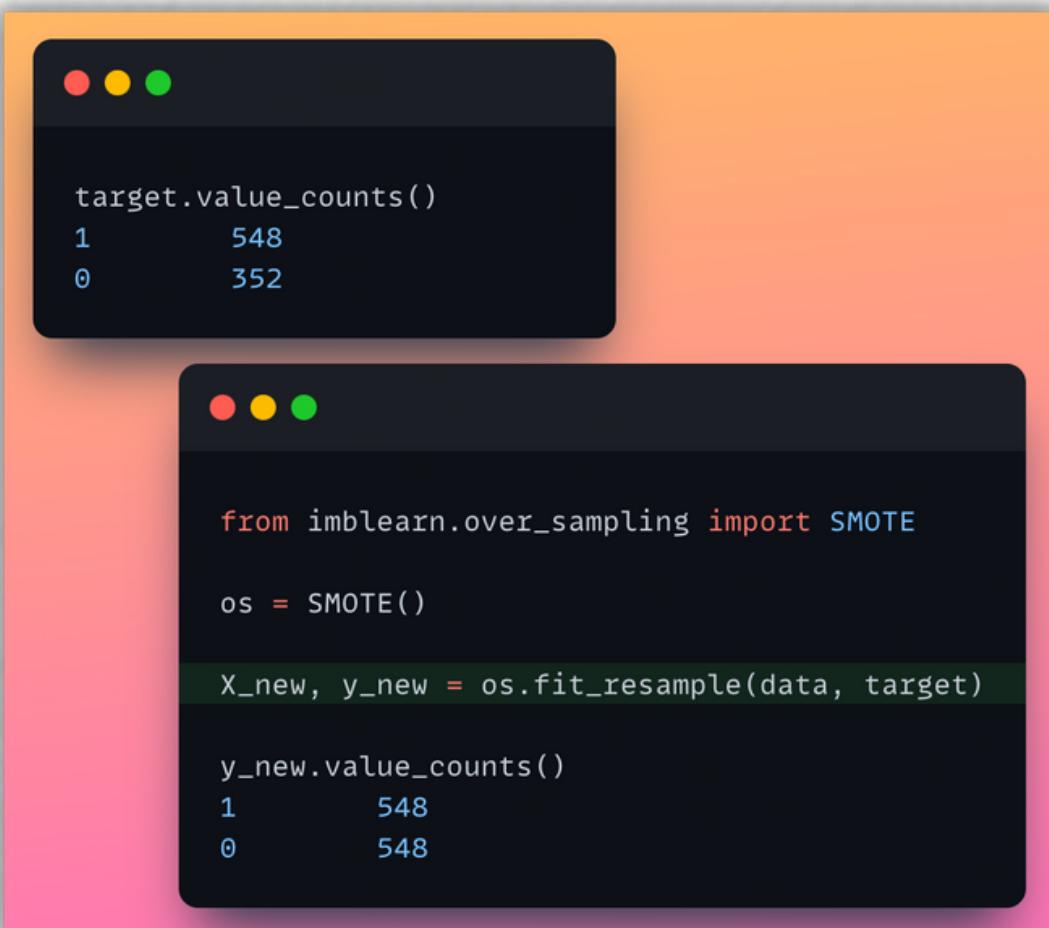
2. PyCaret



→ **Automate ML workflows
with this **low-code
library**.**



3. imbalanced-learn



```
target.value_counts()
1      548
0      352

from imblearn.over_sampling import SMOTE
os = SMOTE()

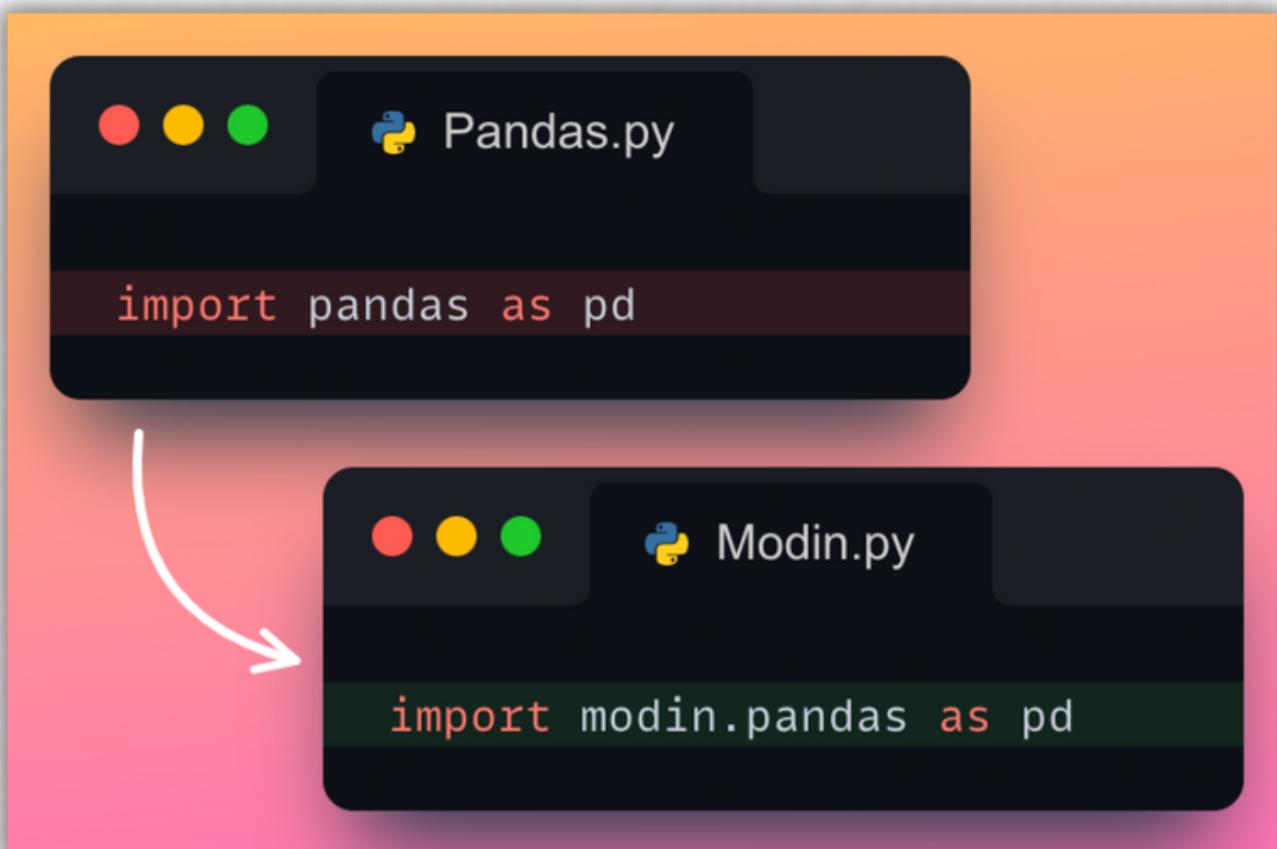
X_new, y_new = os.fit_resample(data, target)

y_new.value_counts()
1      548
0      548
```



A variety of methods
to handle **class
imbalance**.

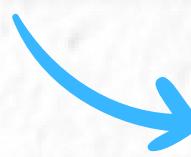
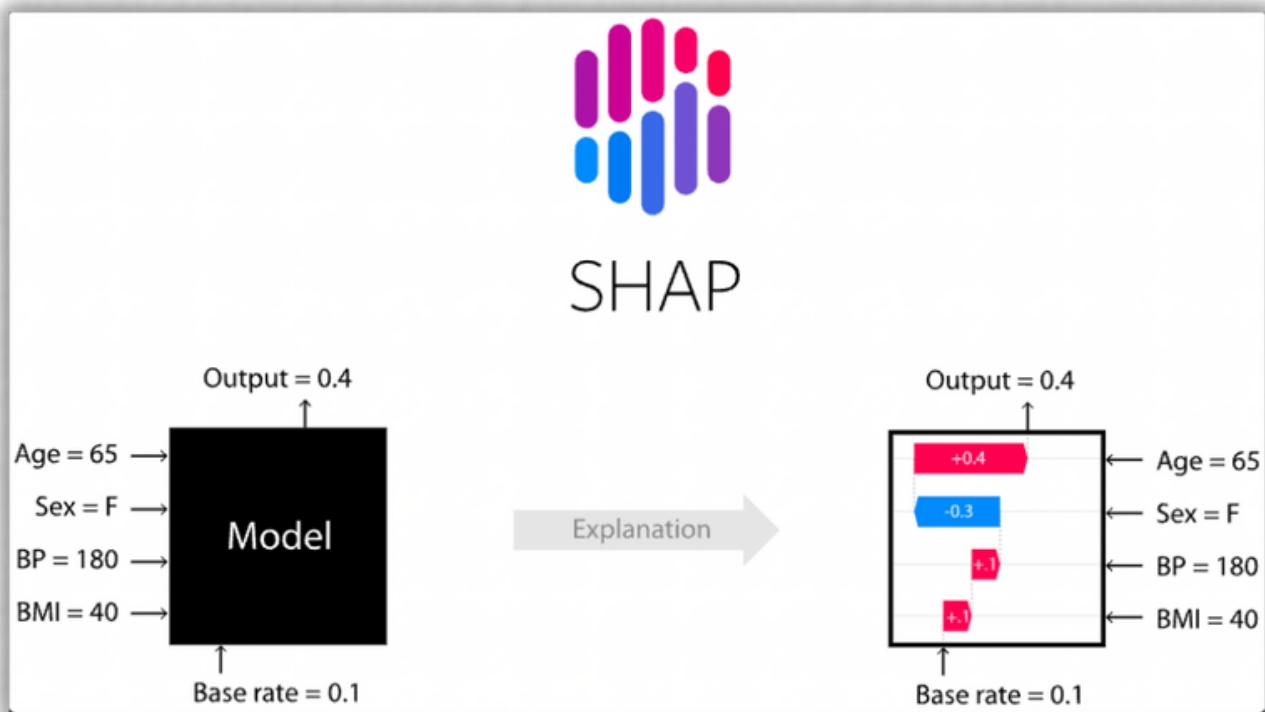
4. Modin



Boost Pandas' performance
up to 70x by modifying
the import.



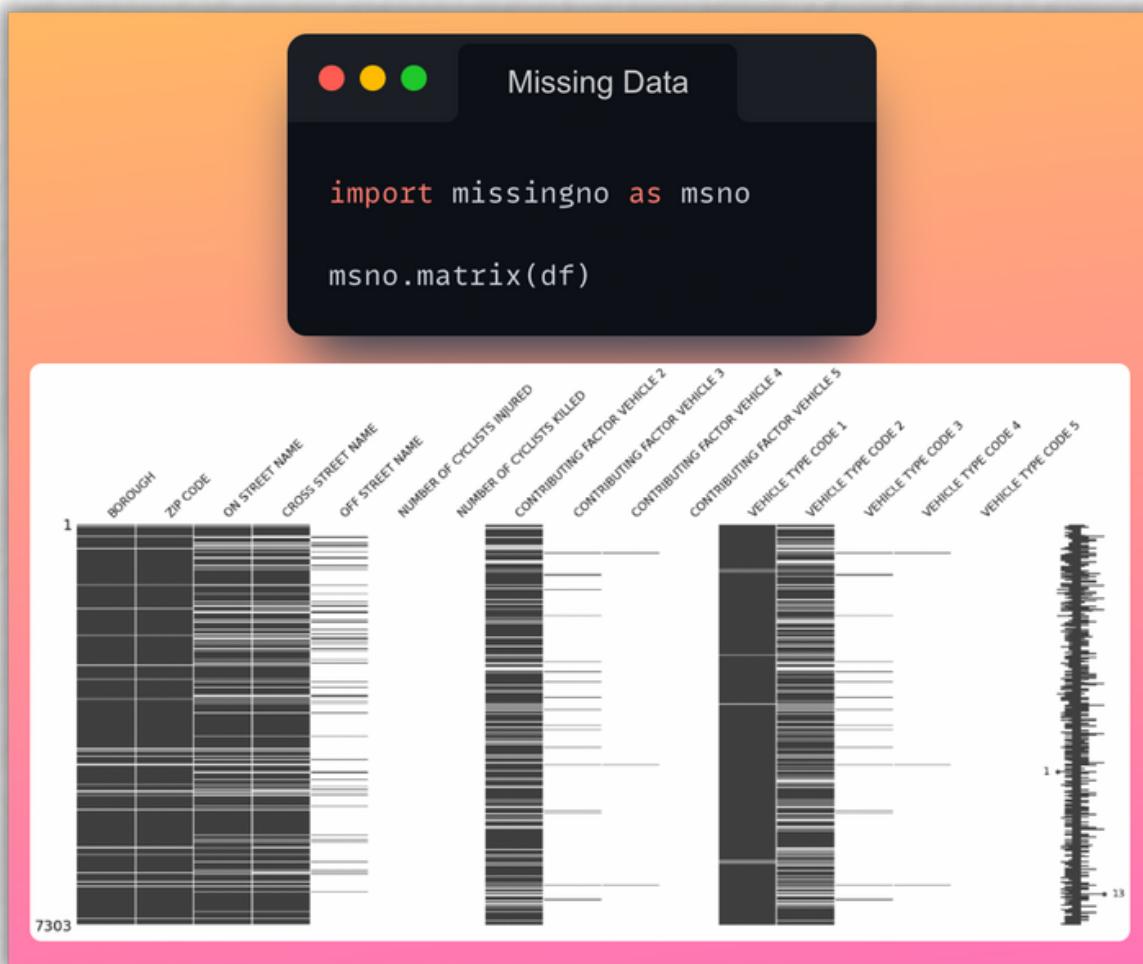
5. SHAP



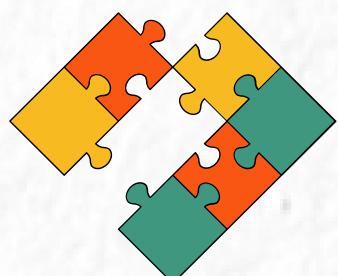
**Explain the output of
any ML model in few
lines of code.**



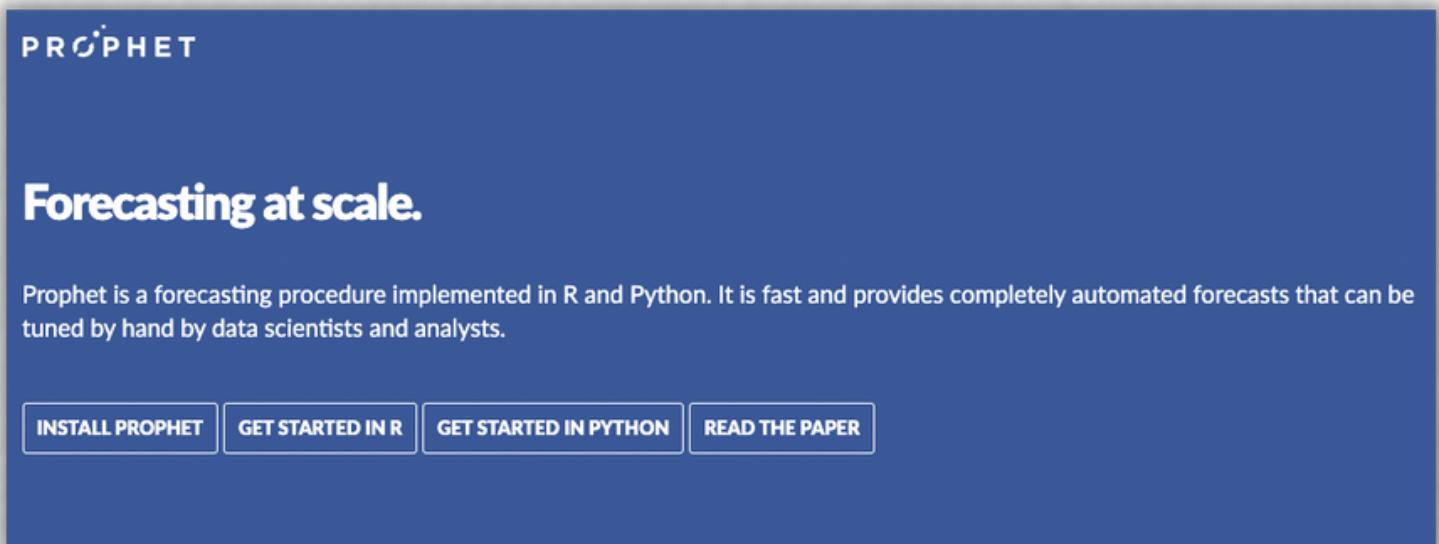
6. Missingno



**Visualize missing values
in your dataset
with ease.**



7. Prophet

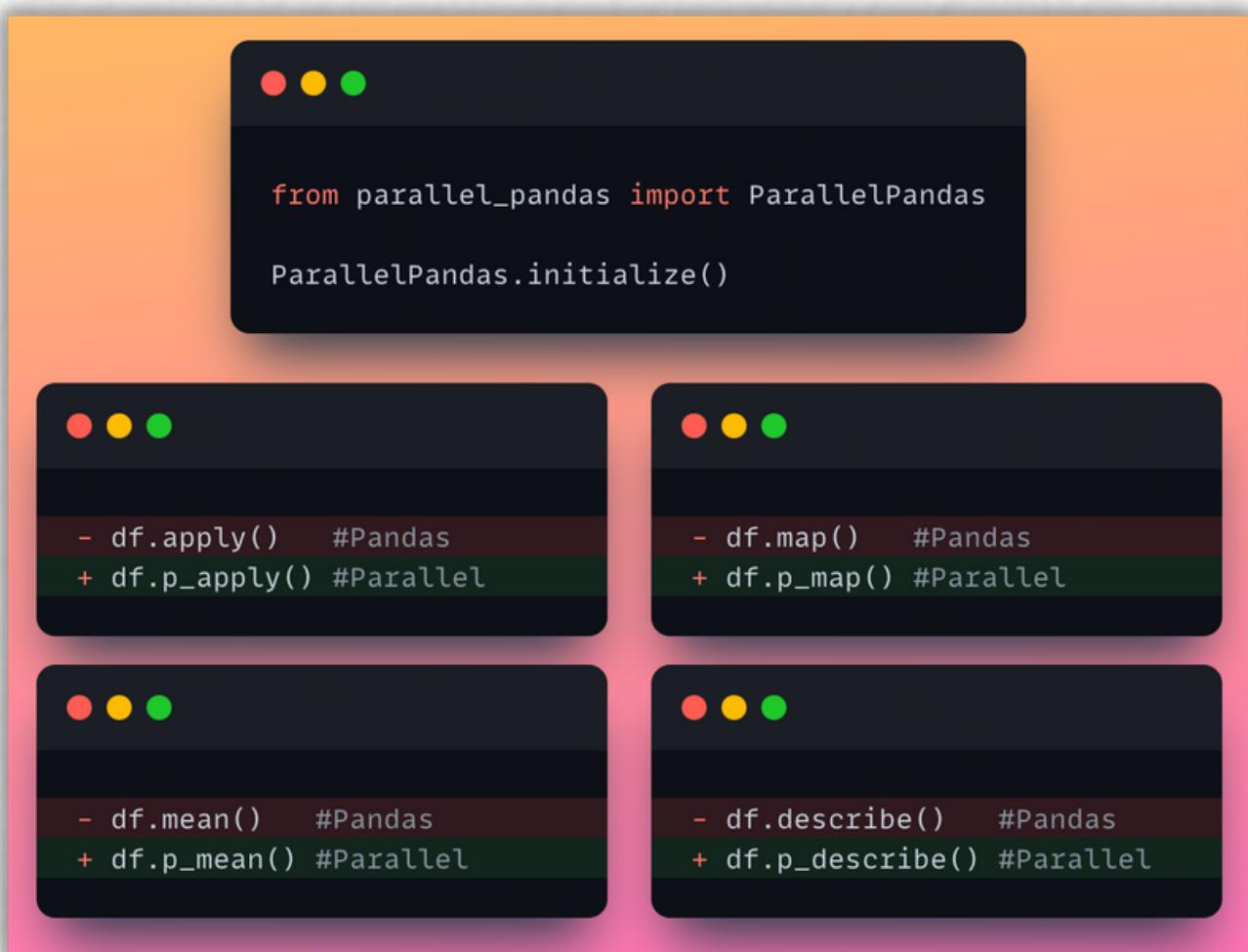


The screenshot shows the official Prophet project landing page. At the top left is the word "PROPHET". Below it is a large white banner with the text "Forecasting at scale." in bold. A paragraph explains that Prophet is a forecasting procedure implemented in R and Python, providing automated forecasts tunable by hand. At the bottom are four buttons: "INSTALL PROPHET", "GET STARTED IN R", "GET STARTED IN PYTHON", and "READ THE PAPER".



Produce **high-quality forecasts** on time-series data.

8. Parallel-Pandas



```
from parallel_pandas import ParallelPandas
ParallelPandas.initialize()
```

```
- df.apply()    #Pandas
+ df.p_apply() #Parallel
```

```
- df.map()     #Pandas
+ df.p_map()  #Parallel
```

```
- df.mean()    #Pandas
+ df.p_mean() #Parallel
```

```
- df.describe()   #Pandas
+ df.p_describe() #Parallel
```



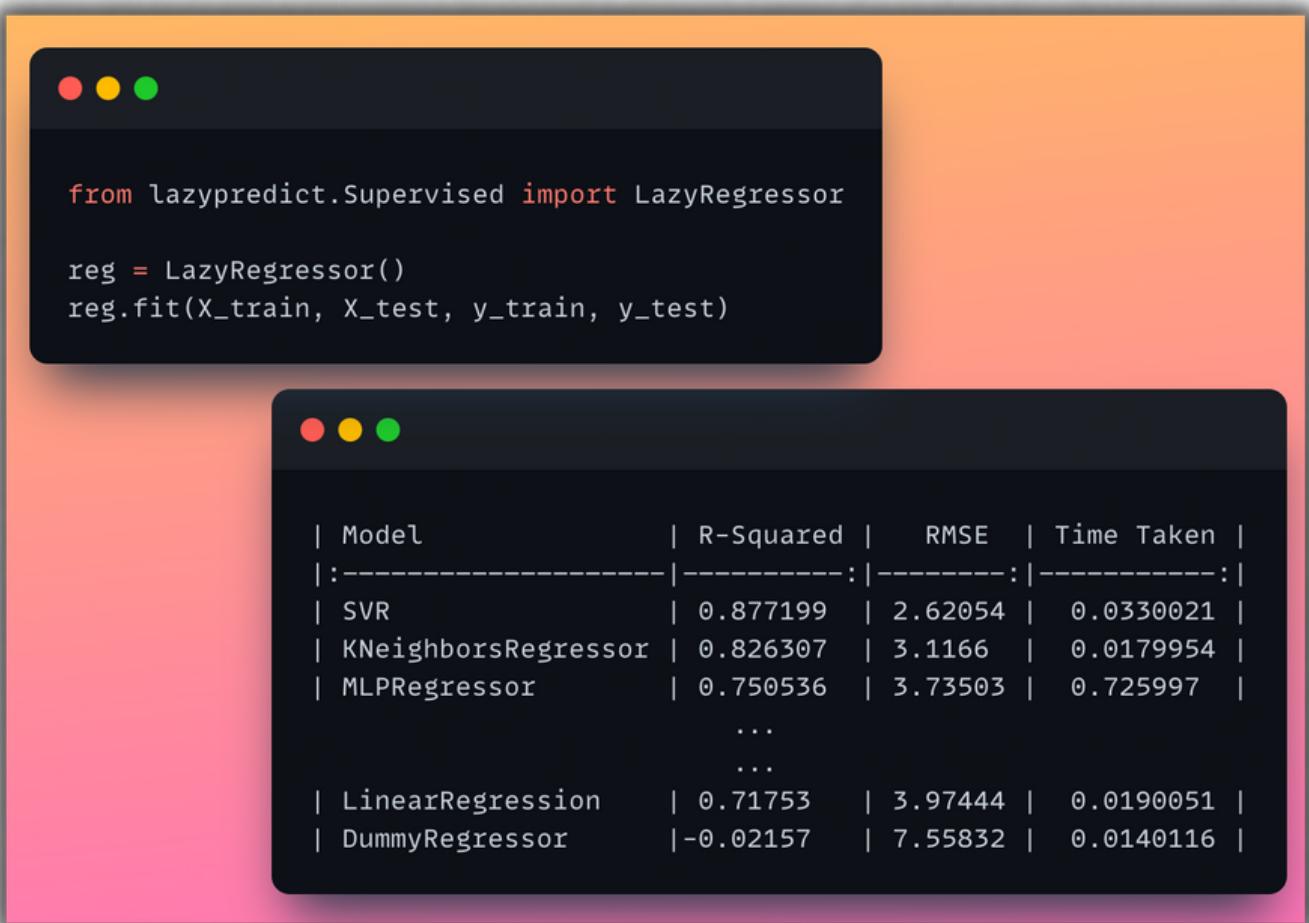
Parallelize Pandas across all
CPU cores for **faster
computation.**

9. Featuretools



**Automated feature
engineering for
ML models.**

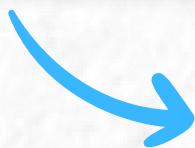
10. Lazy Predict



```
from lazypredict.Supervised import LazyRegressor

reg = LazyRegressor()
reg.fit(X_train, X_test, y_train, y_test)
```

Model	R-Squared	RMSE	Time Taken
SVR	0.877199	2.62054	0.0330021
KNeighborsRegressor	0.826307	3.1166	0.0179954
MLPRegressor	0.750536	3.73503	0.725997
...
LinearRegression	0.71753	3.97444	0.0190051
DummyRegressor	-0.02157	7.55832	0.0140116



Train 30 machine learning models in one line of code.

11. mlxtend



A collection of utility functions
for **processing, evaluating,**
visualizing models.

12. Vaex

```
%%time
df
```

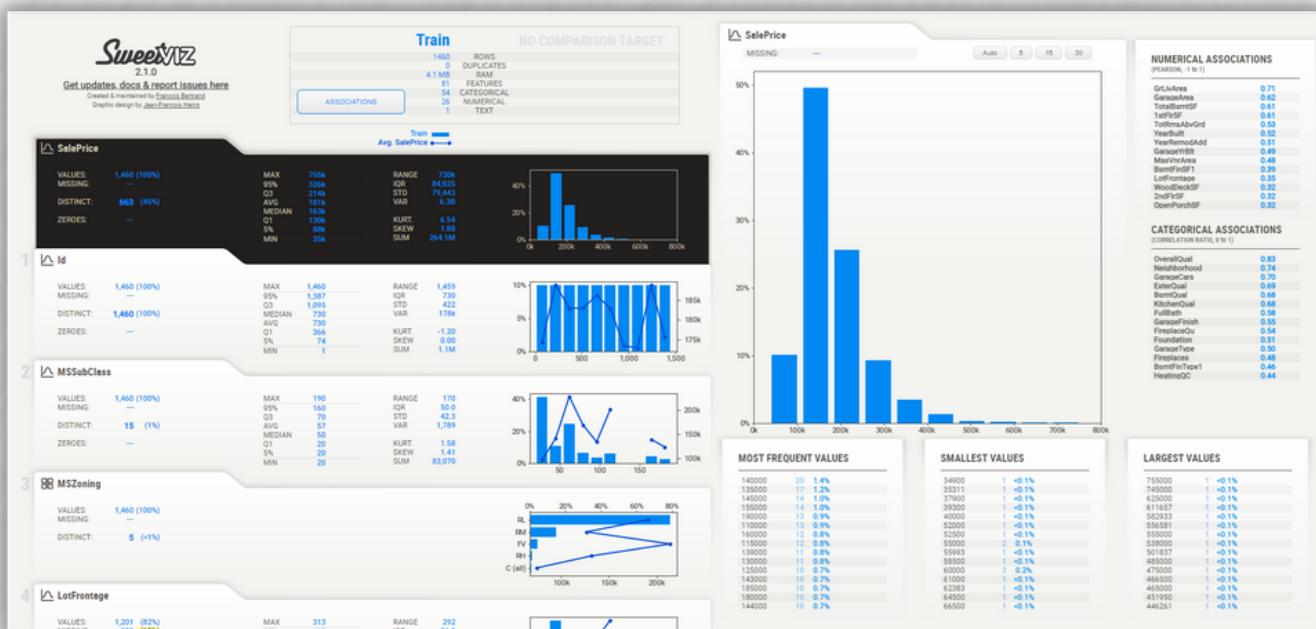
CPU times: user 8 µs, sys: 0 ns, total: 8 µs
Wall time: 16.7 µs

#	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	payment_type	trip_distance	pickup_longitude	pickup_latitude
0	VTS	2009-01-04 02:52:00.000000000	2009-01-04 03:02:00.000000000	1	CASH	2.630000114440918	-73.99195861816406	40.72156524658203
1	VTS	2009-01-04 03:31:00.000000000	2009-01-04 03:38:00.000000000	3	Credit	4.550000190734863	-73.98210144042969	40.736289978027344
2	VTS	2009-01-03 15:43:00.000000000	2009-01-03 15:57:00.000000000	5	Credit	10.350000381469727	-74.0025863647461	40.73974609375
3	DDS	2009-01-01 20:52:58.000000000	2009-01-01 21:14:00.000000000	1	CREDIT	5.0	-73.9742660522461	40.79095458984375
4	DDS	2009-01-24 16:18:23.000000000	2009-01-24 16:24:56.000000000	1	CASH	0.4000000059604645	-74.00157928466797	40.719383239746094
...
1,173,057,922	VTS	2015-12-31 23:59:56.000000000	2016-01-01 00:08:18.000000000	5	1	1.2000000476837158	-73.99381256103516	40.72087097167969
1,173,057,923	CMT	2015-12-31 23:59:58.000000000	2016-01-01 00:05:19.000000000	2	2	2.0	-73.96527099609375	40.76028060913086
1,173,057,924	CMT	2015-12-31 23:59:59.000000000	2016-01-01 00:12:55.000000000	2	2	3.799999952316284	-73.98729705810547	40.739078521728516
1,173,057,925	VTS	2015-12-31 23:59:59.000000000	2016-01-01 00:10:26.000000000	1	2	1.9600000381469727	-73.99755859375	40.72569274902344
1,173,057,926	VTS	2015-12-31 23:59:59.000000000	2016-01-01 00:21:30.000000000	1	1	1.059999942779541	-73.9843978881836	40.76725769042969



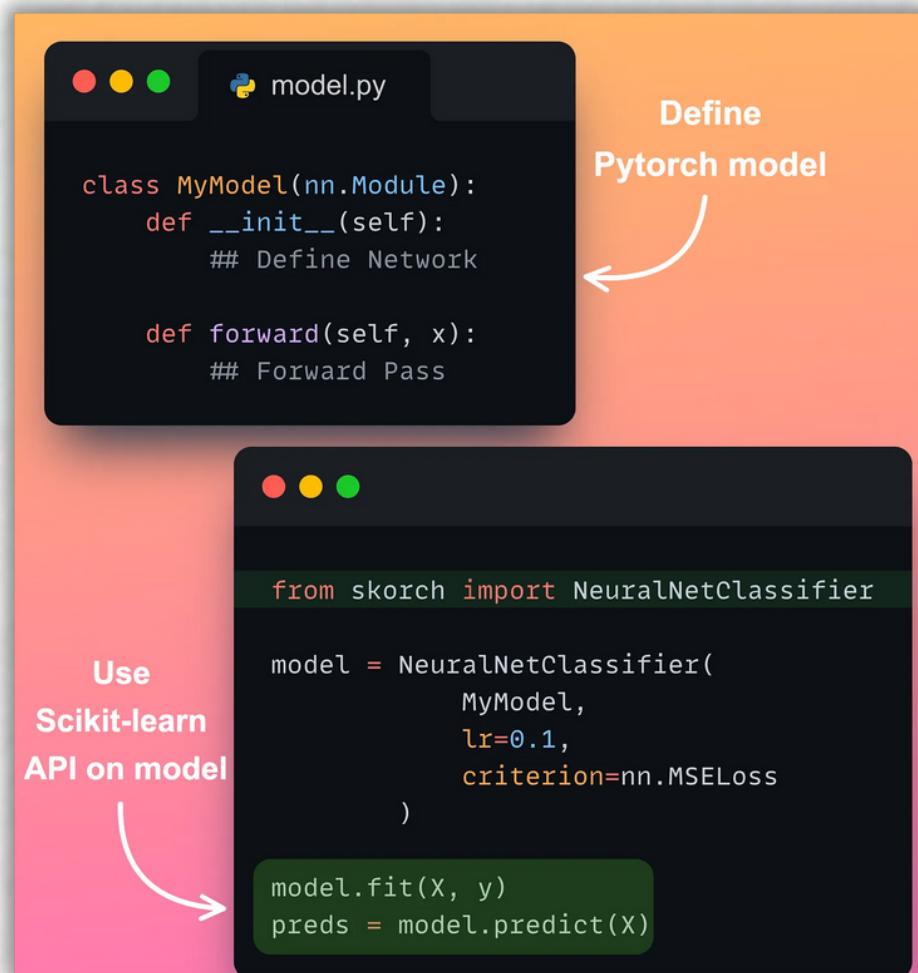
High performance package
for **lazy Out-of-Core**
DataFrames.

13. SweetViz



In-depth EDA report
in two lines
of code.

14. Skorch



Define Pytorch model

```
class MyModel(nn.Module):
    def __init__(self):
        ## Define Network

    def forward(self, x):
        ## Forward Pass
```

Use Scikit-learn API on model

```
from skorch import NeuralNetClassifier

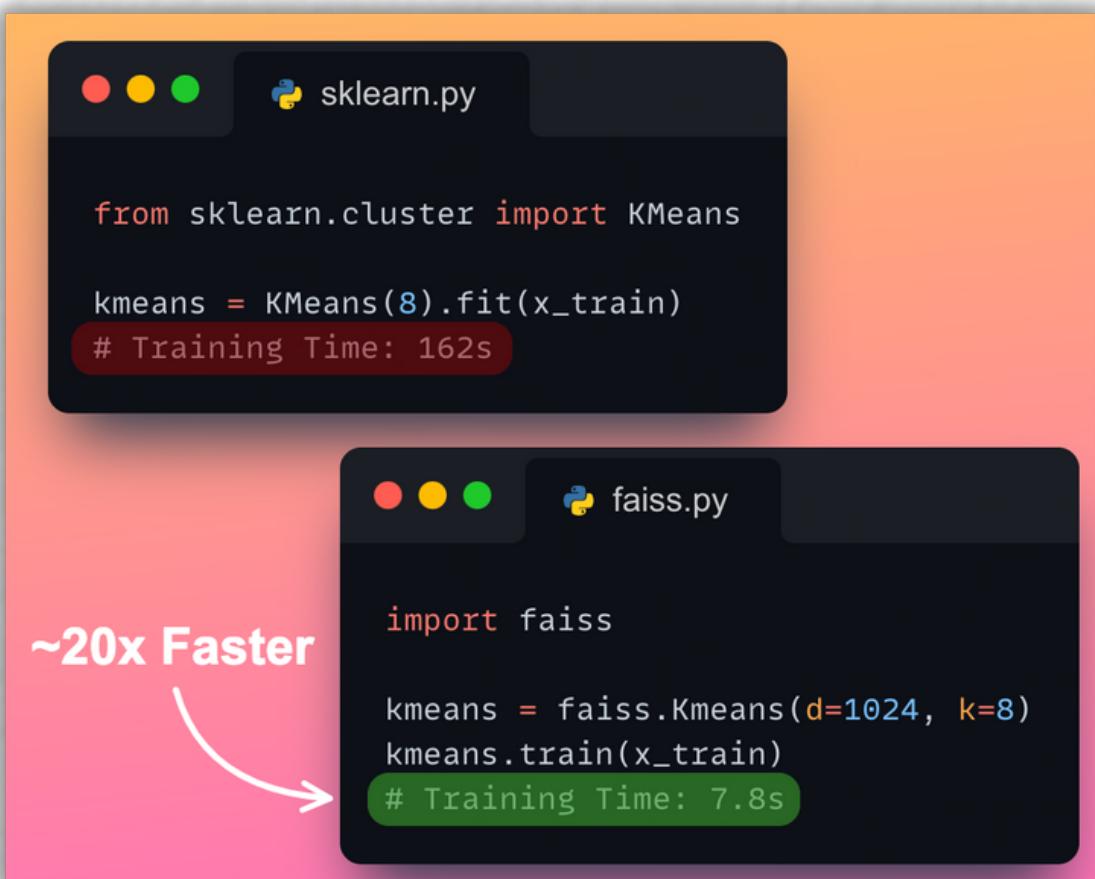
model = NeuralNetClassifier(
    MyModel,
    lr=0.1,
    criterion=nn.MSELoss
)

model.fit(X, y)
preds = model.predict(X)
```



Leverage the **power of PyTorch** with the **elegance of sklearn**.

15. Faiss



```
sklearn.py
```

```
from sklearn.cluster import KMeans
kmeans = KMeans(8).fit(x_train)
# Training Time: 162s
```

```
faiss.py
```

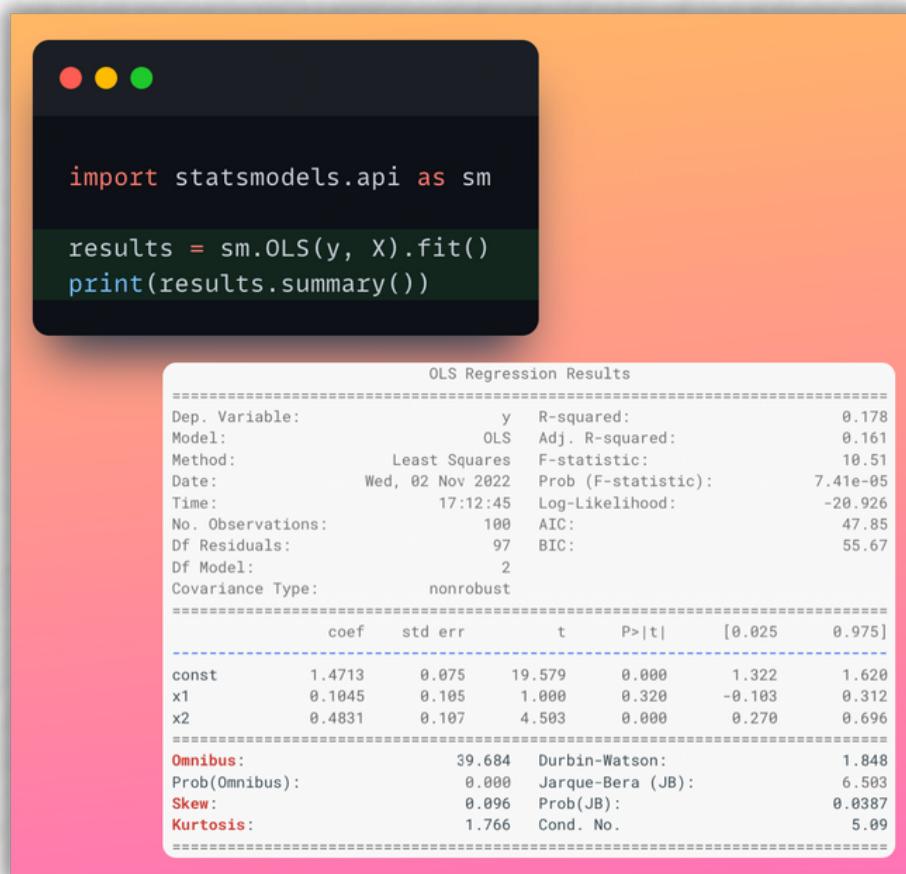
```
import faiss
kmeans = faiss.Kmeans(d=1024, k=8)
kmeans.train(x_train)
# Training Time: 7.8s
```

~20x Faster

Efficient algorithms for
similarity search and
clustering dense
vectors.



16. statsmodel



```
import statsmodels.api as sm

results = sm.OLS(y, X).fit()
print(results.summary())
```

OLS Regression Results

	coef	std err	t	P> t	[0.025	0.975]
const	1.4713	0.075	19.579	0.000	1.322	1.620
x1	0.1045	0.105	1.000	0.320	-0.103	0.312
x2	0.4831	0.107	4.503	0.000	0.270	0.696

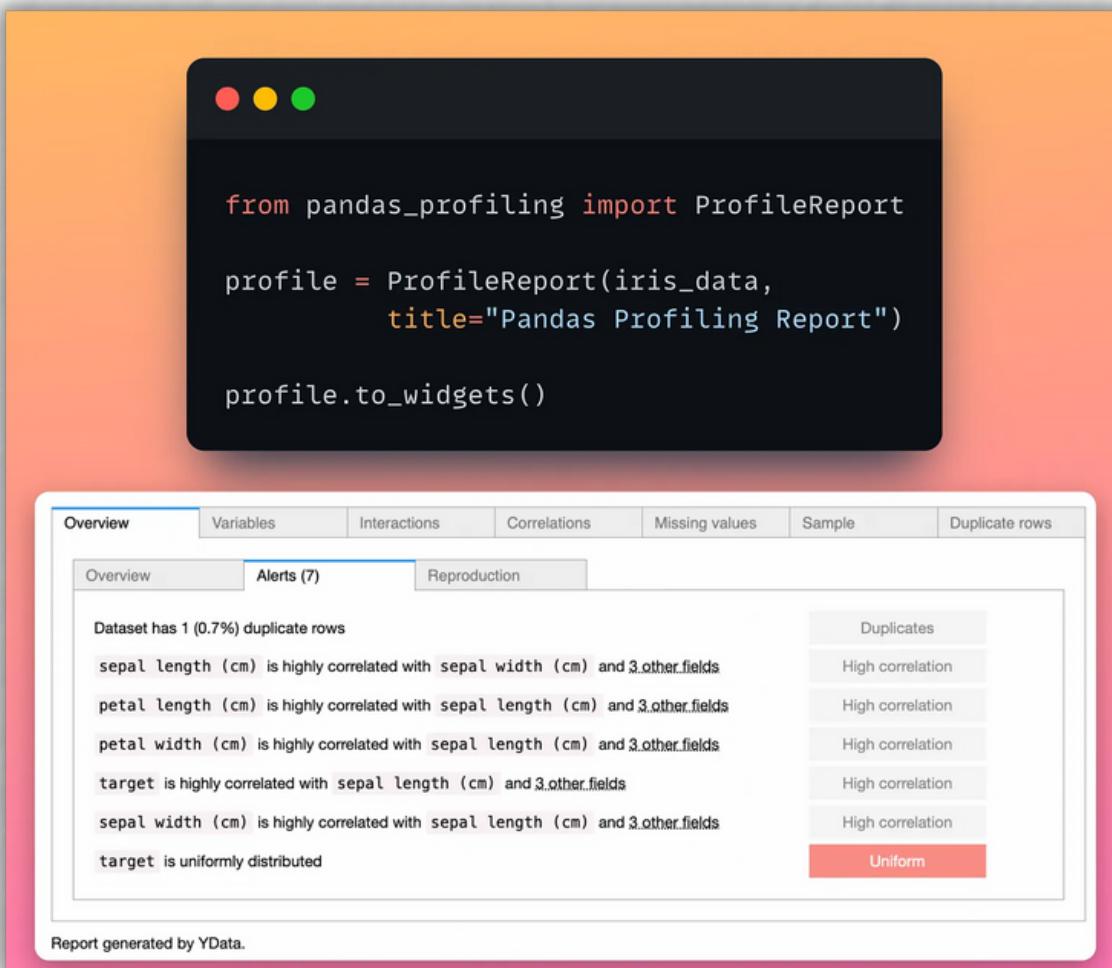
Omnibus: 39.684 Durbin-Watson: 1.848
Prob(Omnibus): 0.000 Jarque-Bera (JB): 6.503
Skew: 0.096 Prob(JB): 0.0387
Kurtosis: 1.766 Cond. No. 5.09



Statistical testing and
data exploration
at fingertips.



17. Pandas-Profiling



A screenshot of a Jupyter Notebook cell containing Python code for generating an EDA report:

```
from pandas_profiling import ProfileReport

profile = ProfileReport(iris_data,
                       title="Pandas Profiling Report")

profile.to_widgets()
```

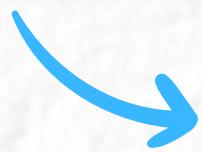
The resulting EDA report is displayed below the code cell. It has a tabular navigation bar at the top:

Overview	Variables	Interactions	Correlations	Missing values	Sample	Duplicate rows
----------	-----------	--------------	--------------	----------------	--------	----------------

The "Alerts (7)" tab is selected, showing the following findings:

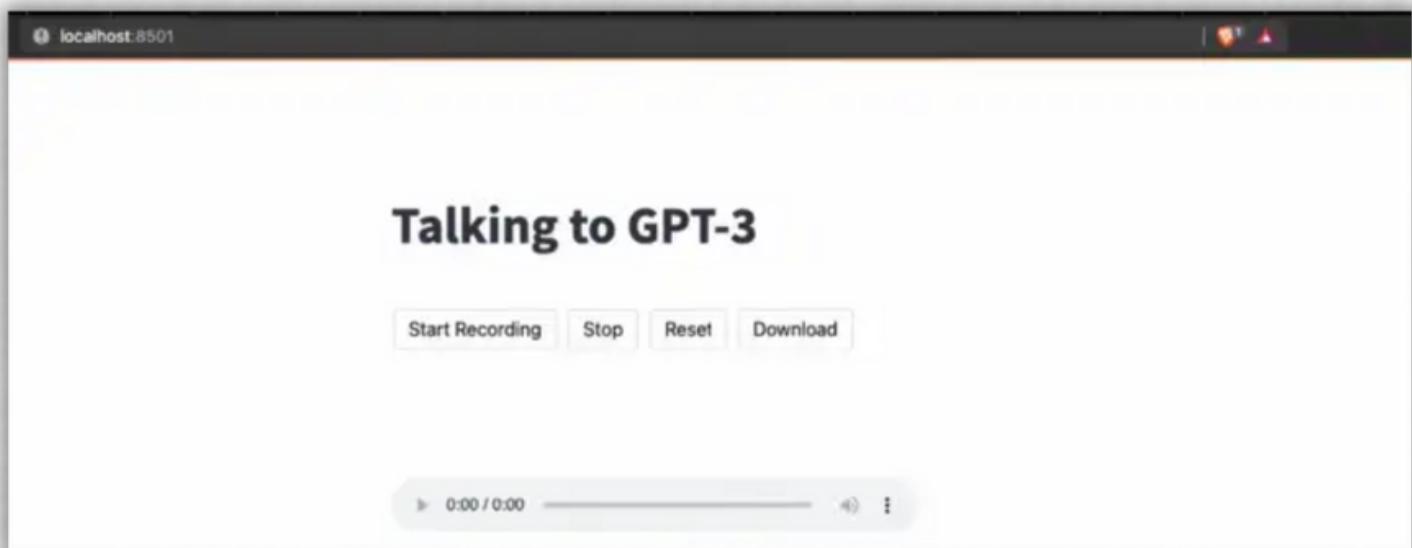
Alert Type	Details
Duplicates	Dataset has 1 (0.7%) duplicate rows
High correlation	sepal length (cm) is highly correlated with sepal width (cm) and 3_other.fields
High correlation	petal length (cm) is highly correlated with sepal length (cm) and 3_other.fields
High correlation	petal width (cm) is highly correlated with sepal length (cm) and 3_other.fields
High correlation	target is highly correlated with sepal length (cm) and 3_other.fields
High correlation	sepal width (cm) is highly correlated with sepal length (cm) and 3_other.fields
Uniform	target is uniformly distributed

At the bottom of the report, it says "Report generated by YData."



Generate a high-level
EDA report of your
data in no time.

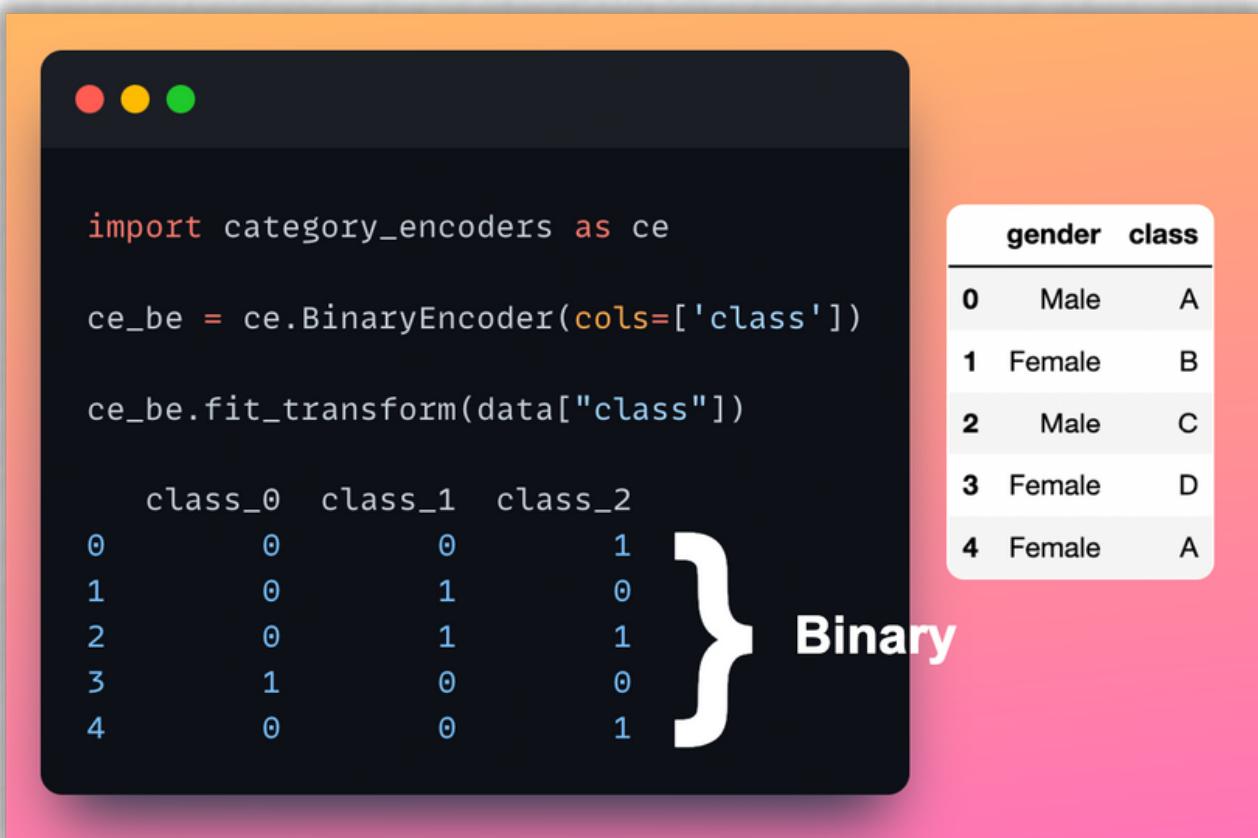
18. Streamlit



→ Create and host data-based
Python web apps
in few lines
of code.



19. Category-encoders



```
import category_encoders as ce

ce_be = ce.BinaryEncoder(cols=['class'])

ce_be.fit_transform(data["class"])

    class_0  class_1  class_2
0          0        0        1
1          0        1        0
2          0        1        1
3          1        0        0
4          0        0        1
```

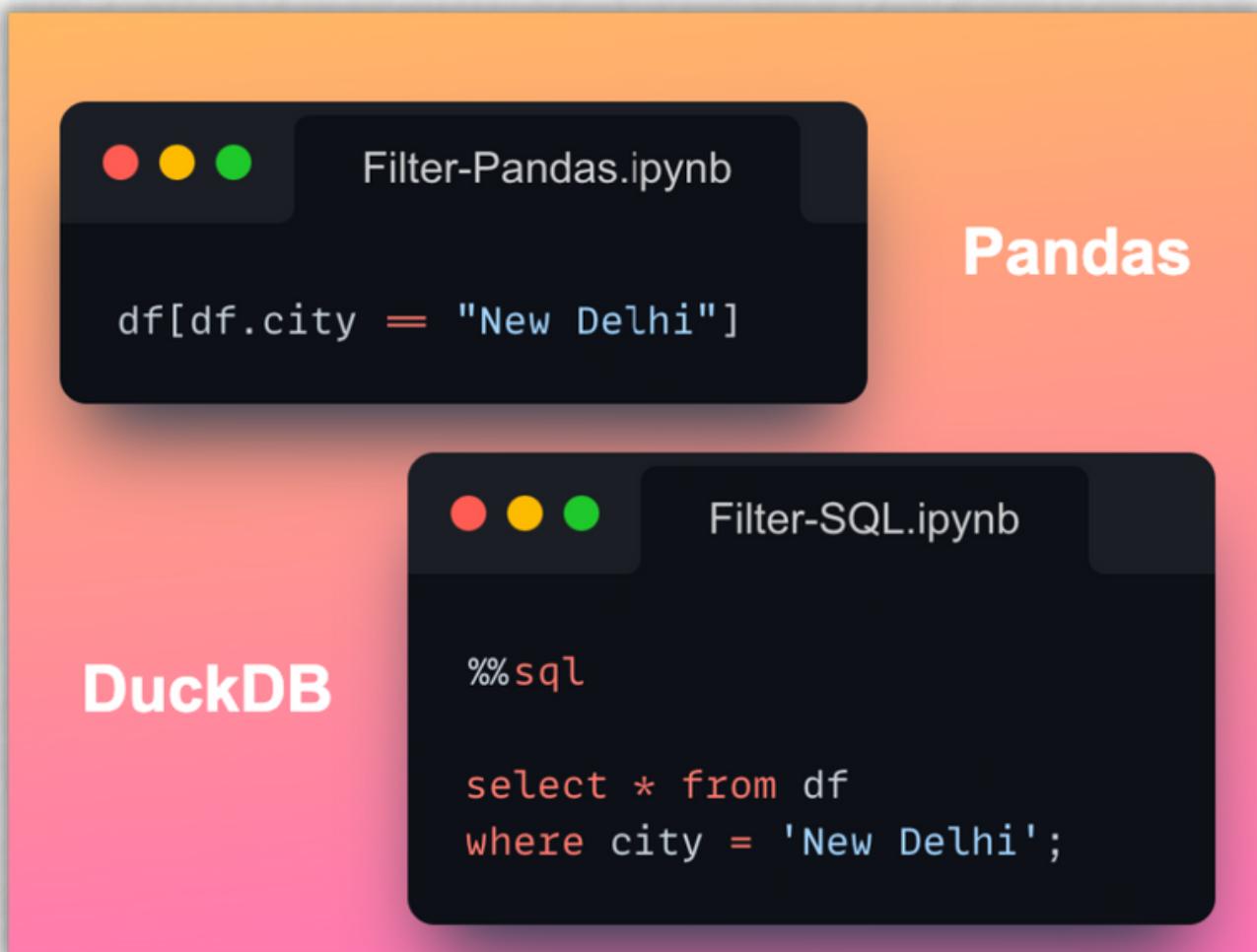
The code imports the category_encoders module and creates a BinaryEncoder object for the 'class' column. It then fits and transforms the data, resulting in a binary matrix where each row corresponds to a data point and each column corresponds to a class category (0, 1, or 2). A large white brace on the right side of the matrix is labeled 'Binary'.

	gender	class
0	Male	A
1	Female	B
2	Male	C
3	Female	D
4	Female	A



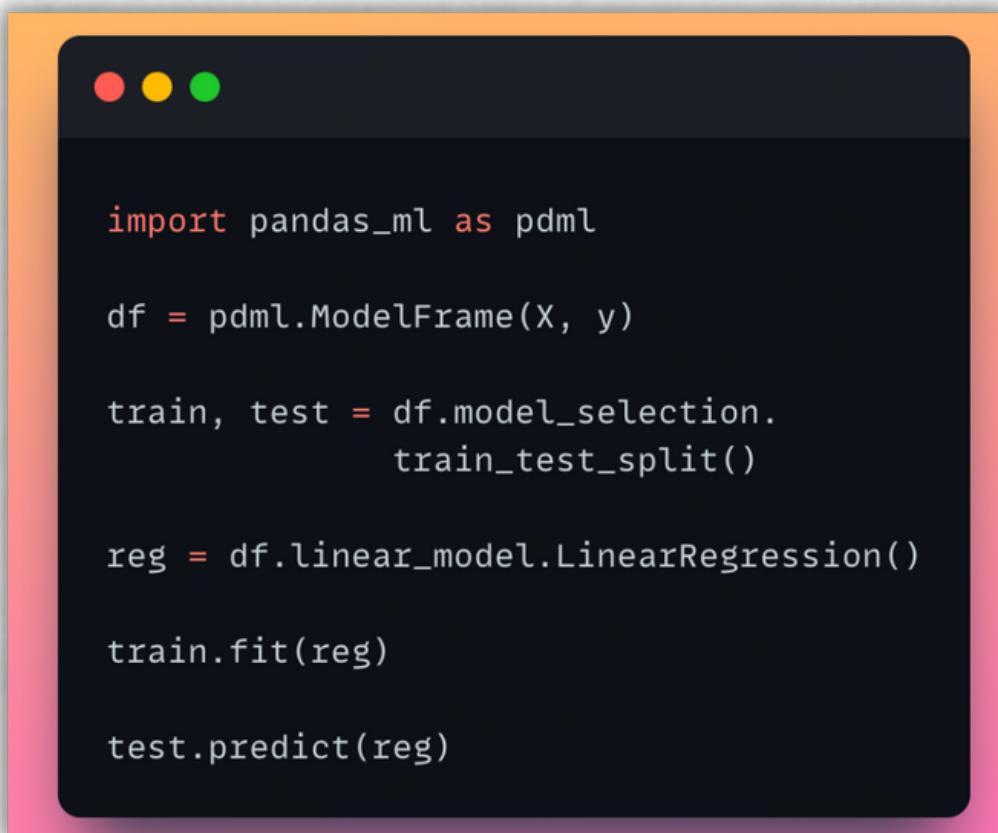
Over 15 categorical data encoders.

20. DuckDB



Run SQL queries
on DataFrame.

21. PandasML



```
import pandas_ml as pdml

df = pdml.ModelFrame(X, y)

train, test = df.model_selection.
    train_test_split()

reg = df.linear_model.LinearRegression()

train.fit(reg)

test.predict(reg)
```



Pandas data wrangling +
Sklearn algorithms +
Matplotlib visualization.

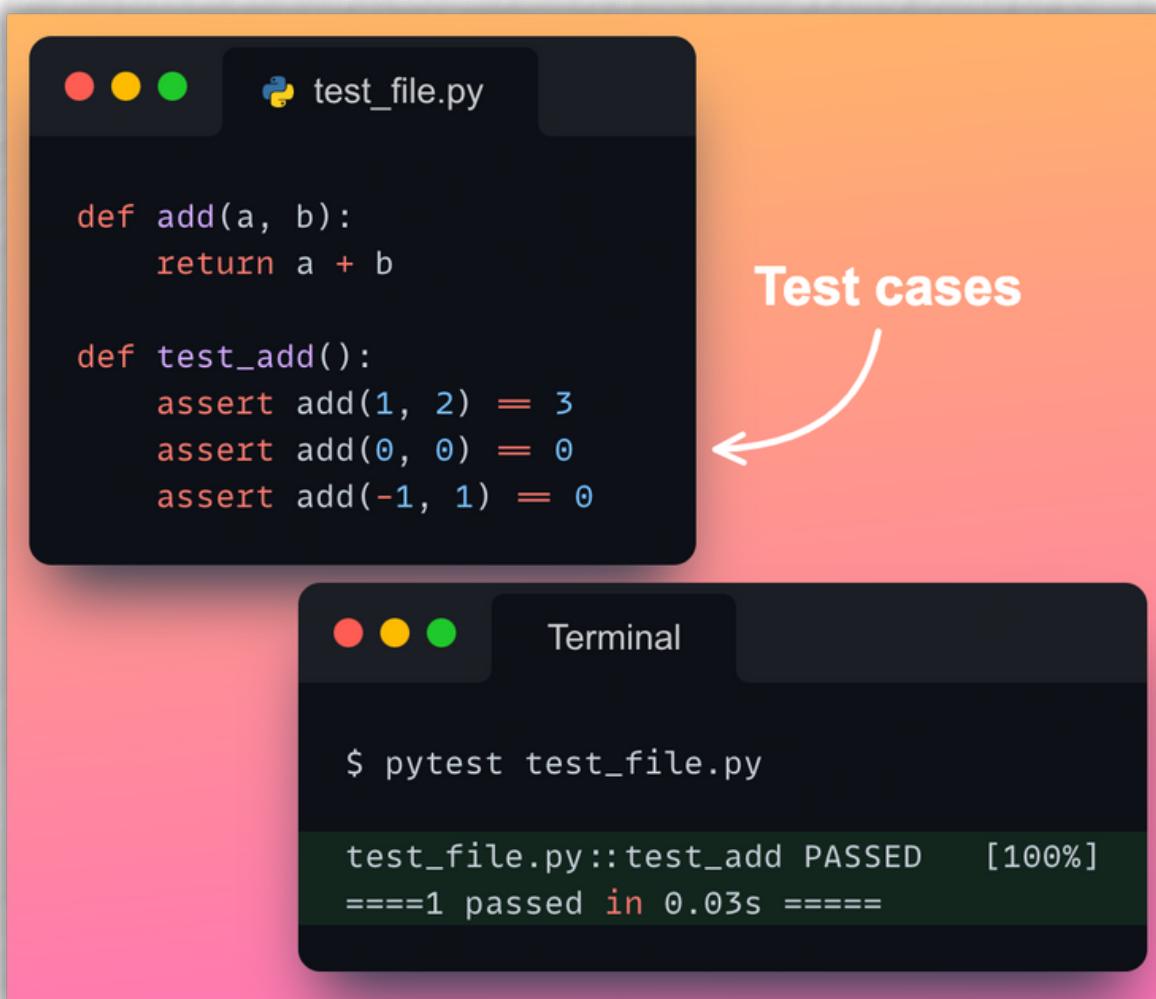
Sklearn

Matplotlib



Pandas

22. Pytest



```
test_file.py
```

```
def add(a, b):
    return a + b

def test_add():
    assert add(1, 2) == 3
    assert add(0, 0) == 0
    assert add(-1, 1) == 0
```

Test cases

```
Terminal
```

```
$ pytest test_file.py

test_file.py::test_add PASSED [100%]
=====1 passed in 0.03s=====
```



An **elegant testing framework** to test your code.



23. Numexpr

```
import numpy as np
import numexpr as ne

a = np.random.random(10**7)
b = np.random.random(10**7)

%timeit np.cos(a) + np.sin(b)
142 ms ± 257 µs per loop

%timeit ne.evaluate("cos(a) + sin(b)")
32.5 ms ± 229 µs per loop ~5x Faster
```



Parallelize NumPy to
all CPU cores for
20x speedup.



24. CSV-Kit

The screenshot shows the CSV-Kit application interface. At the top left is a table titled "data.csv" with columns Name, Marks, and Grade. The data is:

Name	Marks	Grade
Joe	95	A
Hanna	89	B
Chris	92	A
Julie	94	A

Below the table is a section titled "Column Names" showing the output of the command:

```
$ csvcut -n data.csv
1: Name
2: Marks
3: Grade
```

To the right is a "Column Stats" window showing the output of the command:

```
$ csvstat data.csv
2. "Marks"
Type of data: Number
Contains null values: False
Unique values: 4
Smallest value: 89
Largest value: 95
Sum: 370
Mean: 92.5
Median: 93
StDev: 2.646
```

At the bottom is a "Query" window showing the output of the command:

```
$ csvsql --query "select * from data where Marks>90" data.csv
| Name | Marks | Grade |
| ---- | ----- | ----- |
| Joe | 95 | A |
| Chris | 92 | A |
| Julie | 94 | A |
```



Explore, query and
describe CSV files
from terminal.

>_

25. PivotTableJS

The screenshot shows a Jupyter Notebook cell titled "notebook.ipynb" containing the following Python code:

```
from pivottablejs import pivot_ui
pivot_ui(df)
```

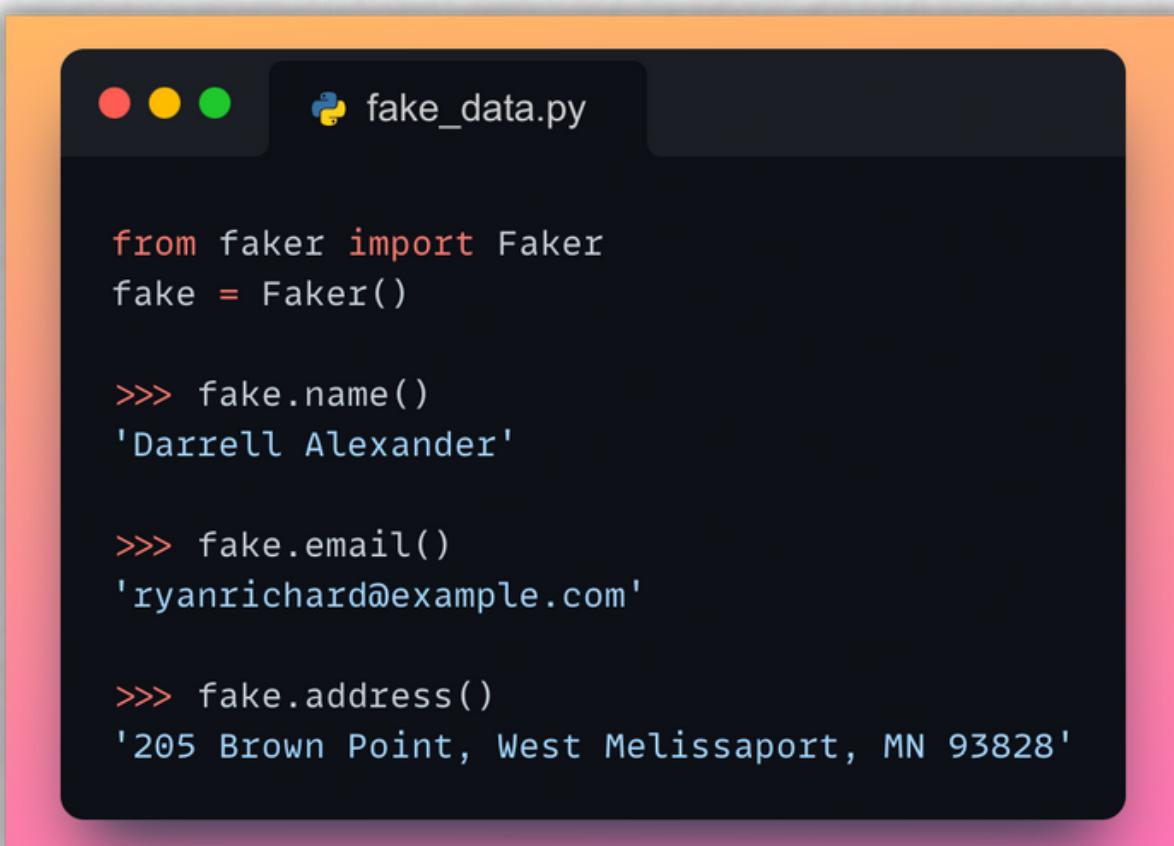
A pink "pop_out" button is visible above the visualization. The visualization itself is a PivotTableJS interface. It has three main sections: a left sidebar with dropdown menus for "Table", "Name", "Company_Name", "Employee_Job_Title", "Employee_Country", "Employee_Salary", and "Employee_Rating"; a middle section with dropdowns for "Count" and "Employee_City"; and a right section showing a pivot table with columns for "Employment_Status" (Full Time, Intern, Totals) and rows for various cities like Aliciafort, Kristaburgh, etc. The total count is 1,000.

Employee_City	Employment_Status	Full Time	Intern	Totals
Aliciafort		89	24	113
Kristaburgh		77	20	97
New Cindychester		82	24	106
New Russellton		73	20	93
North Melissafurt		62	16	78
Ricardomouth		81	25	106
Wardfort		82	14	96
West Jamesview		94	26	120
Whitakerbury		66	21	87
Whiteside		85	19	104
Totals		791	209	1,000



Drap-n-drop tools to
group, pivot, plot
dataframe.

26. Faker



A terminal window with a dark theme and a blue gradient border. The title bar says 'fake_data.py'. The window contains Python code demonstrating the Faker library:

```
from faker import Faker
fake = Faker()

>>> fake.name()
'Darrell Alexander'

>>> fake.email()
'ryanrichard@example.com'

>>> fake.address()
'205 Brown Point, West Melissaport, MN 93828'
```



Generate **fake** yet
meaningful data
in seconds.

27. Icecream



```
Print
```

```
def func(arr, n):
    print("arr =", arr, "n =", n)

func([1,2,3], 2)
## arr = [1, 2, 3] n = 2
```



```
Icecream
```

```
from icecream import ic

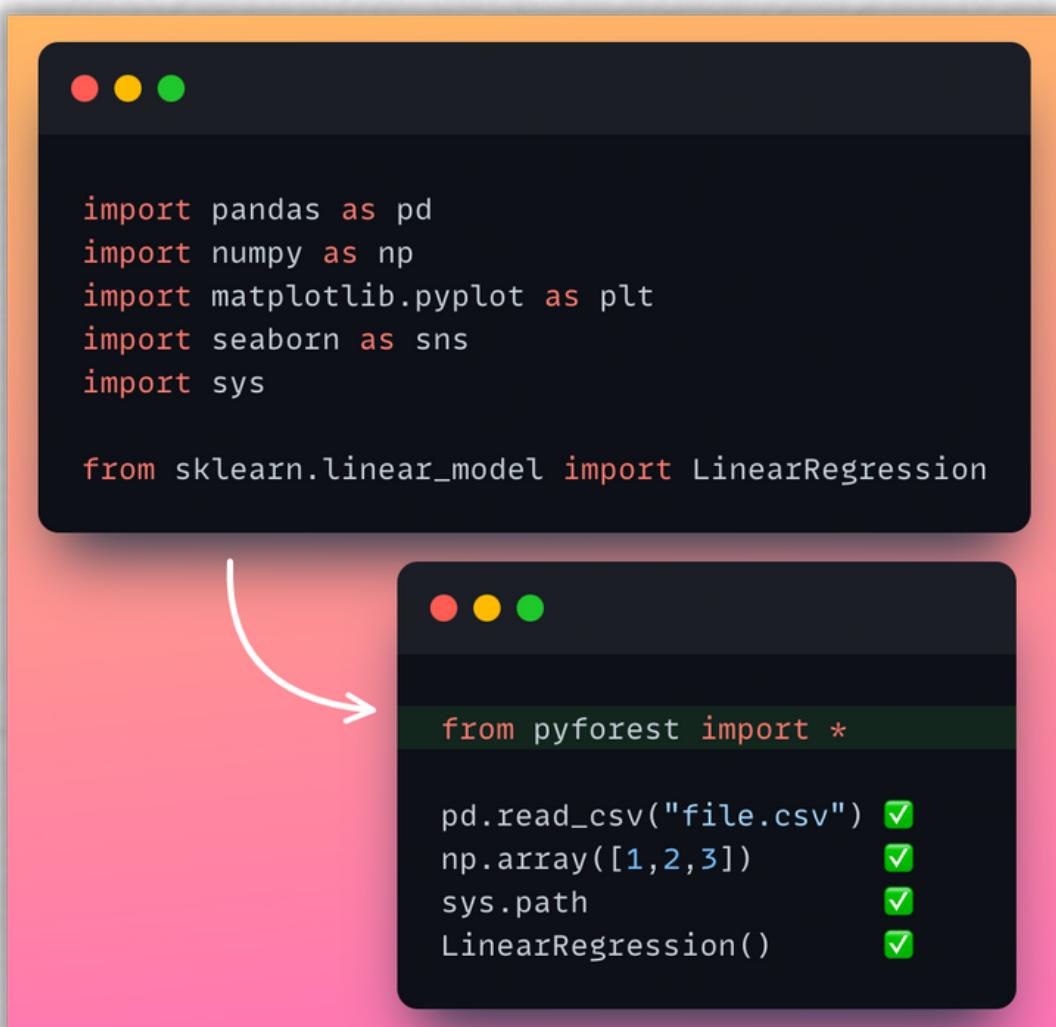
def func(arr, n):
    ic(arr, n)

func([1,2,3], 2)
## ic| arr: [1, 2, 3], n: 2
```



Don't debug with
print(). Use
icecream.

28. Pyforest



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sys

from sklearn.linear_model import LinearRegression
```

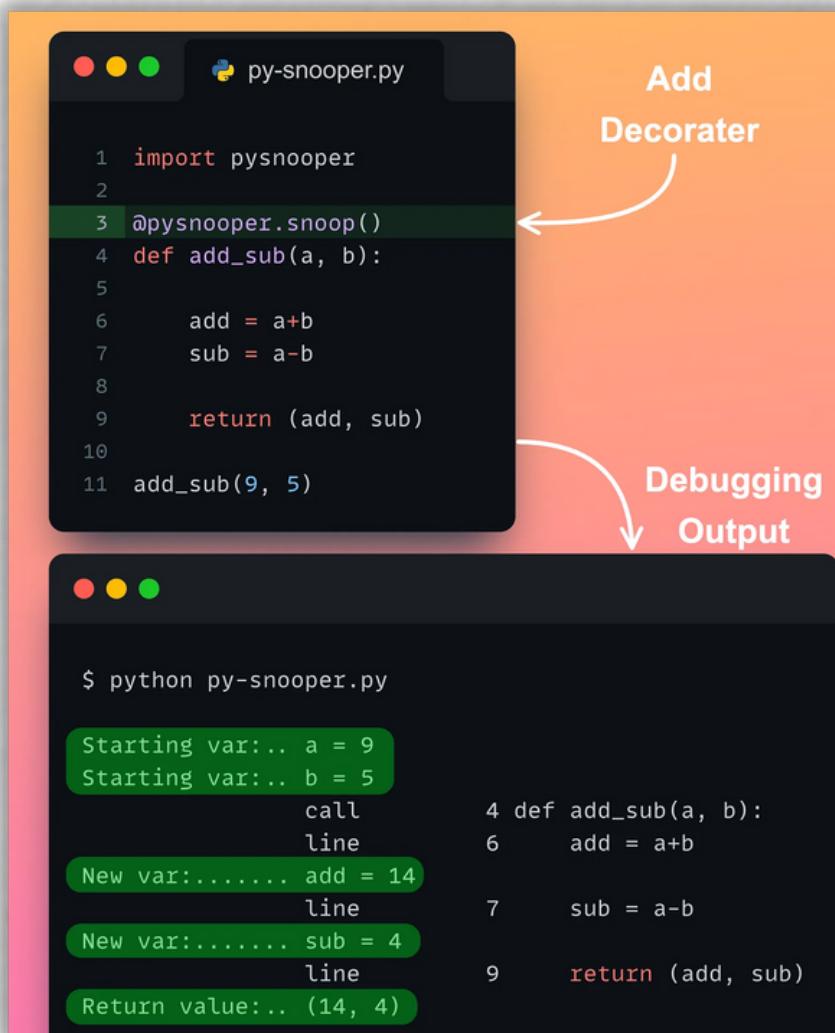
```
from pyforest import *

pd.read_csv("file.csv") ✓
np.array([1,2,3]) ✓
sys.path ✓
LinearRegression() ✓
```

No need to write imports.

**Automatic package
import.**

29. PySnooper



The screenshot shows two terminal windows. The top window displays the Python script `py-snooper.py` with the following code:

```
1 import pysnooper
2
3 @pysnooper.snoop()
4 def add_sub(a, b):
5
6     add = a+b
7     sub = a-b
8
9     return (add, sub)
10
11 add_sub(9, 5)
```

The bottom window shows the output of running the script with `$ python py-snooper.py`. It includes variable values and line numbers from the original code:

```
$ python py-snooper.py
Starting var:.. a = 9
Starting var:.. b = 5
call
line
New var:..... add = 14
line
New var:..... sub = 4
line
Return value:.. (14, 4)
```

Annotations on the right side of the top window explain the features:

- An arrow points to the `@pysnooper.snoop()` line with the text "Add Decorator".
- An arrow points to the output window with the text "Debugging Output".



Profile your code. **Track new variables, and their updates.**

30. Sidetable

The screenshot shows a Jupyter Notebook cell with the following code:

```
import sidetable  
  
df.stb.freq(["City"],  
            style=True)
```

To the right of the code, there are two tables:

value_counts()

City	Count	Percent
West Jamesview	120	12.00%
Aliciafort	113	11.30%
New Cindychester	106	10.60%
Ricardomouth	106	10.60%
Whiteside	104	10.40%
Kristaburgh	97	9.70%
Wardfort	96	9.60%
New Russellton	93	9.30%
Whitakerbury	87	8.70%
North Melissafurt	78	7.80%

Cumulative Frequency Table

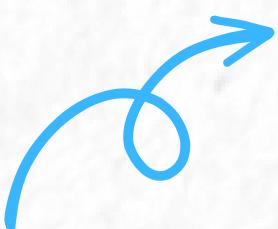
	City	Count	Percent	Cumulative Count	Cumulative Percent
0	West Jamesview	120	12.00%	120	12.00%
1	Aliciafort	113	11.30%	233	23.30%
2	Ricardomouth	106	10.60%	339	33.90%
3	New Cindychester	106	10.60%	445	44.50%
4	Whiteside	104	10.40%	549	54.90%
5	Kristaburgh	97	9.70%	646	64.60%
6	Wardfort	96	9.60%	742	74.20%
7	New Russellton	93	9.30%	835	83.50%
8	Whitakerbury	87	8.70%	922	92.20%
9	North Melissafurt	78	7.80%	1,000	100.00%

Supercharge Pandas'
value_counts()
method.

Hope that helped.



Checkout my daily newsletter to learn something new about **Python** and **Data Science** everyday.

 avichawla.substack.com

Connect with me on [LinkedIn](#).

<https://www.linkedin.com/in/avi-chawla>