30 Python Libraries to (Hugely) Boost Your Data Science Productivity

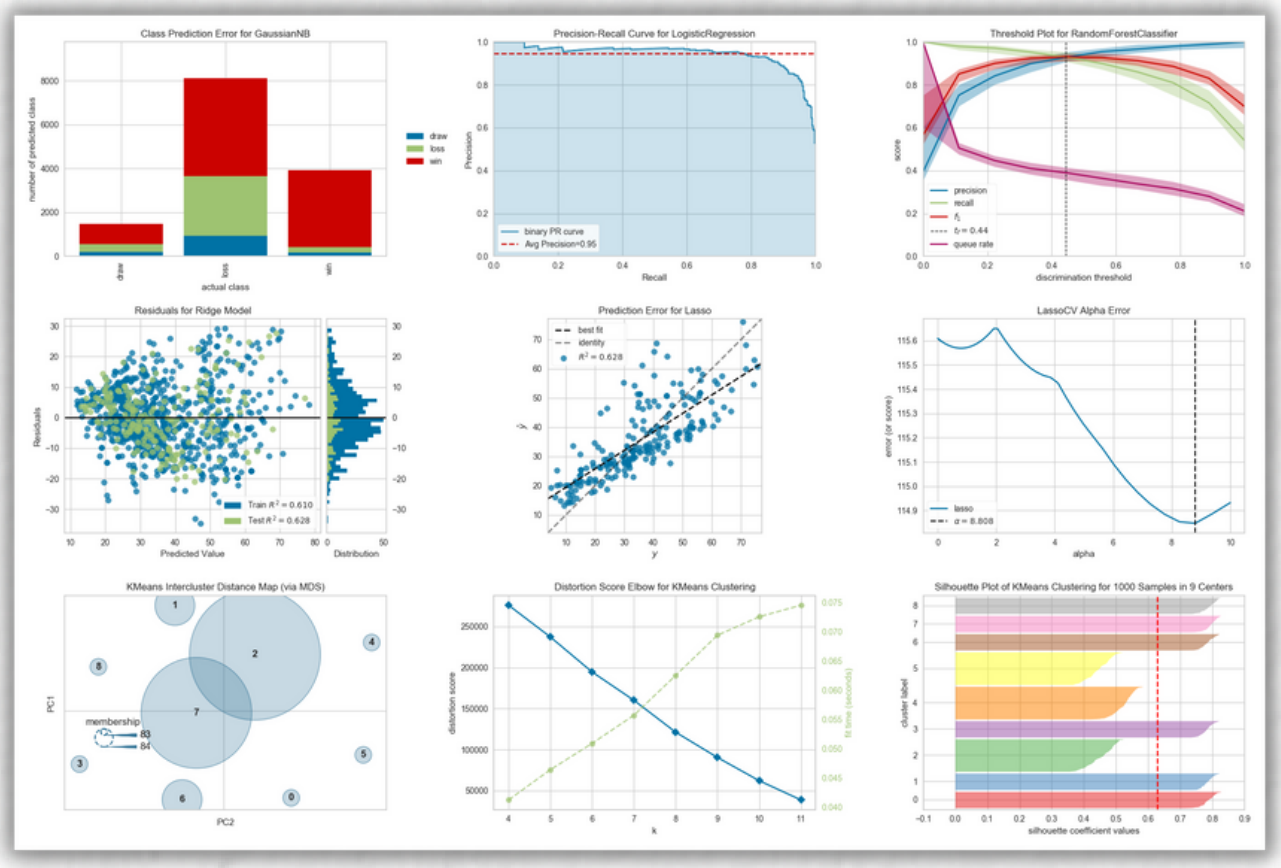
Avi Chawla

avichawla.substack.com

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

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

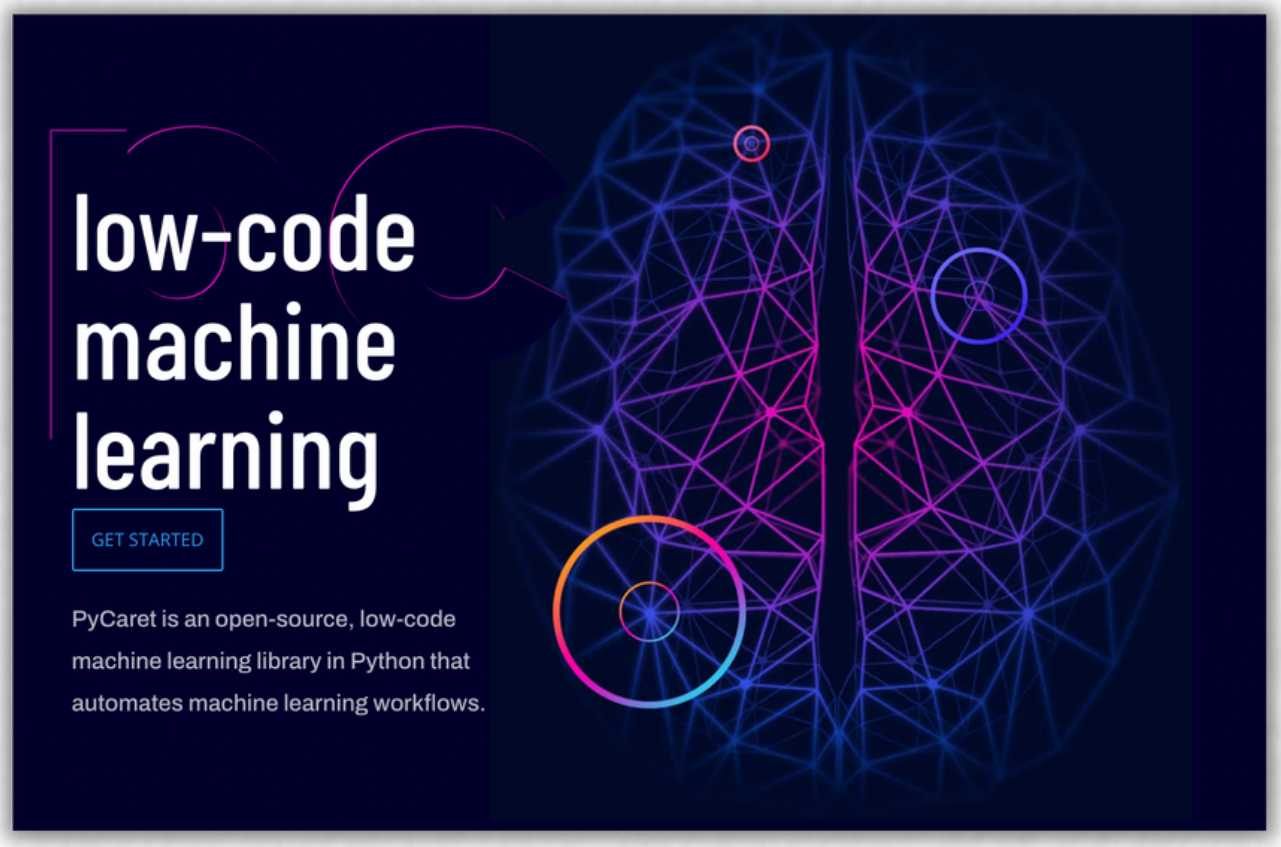
1. YellowBrick



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

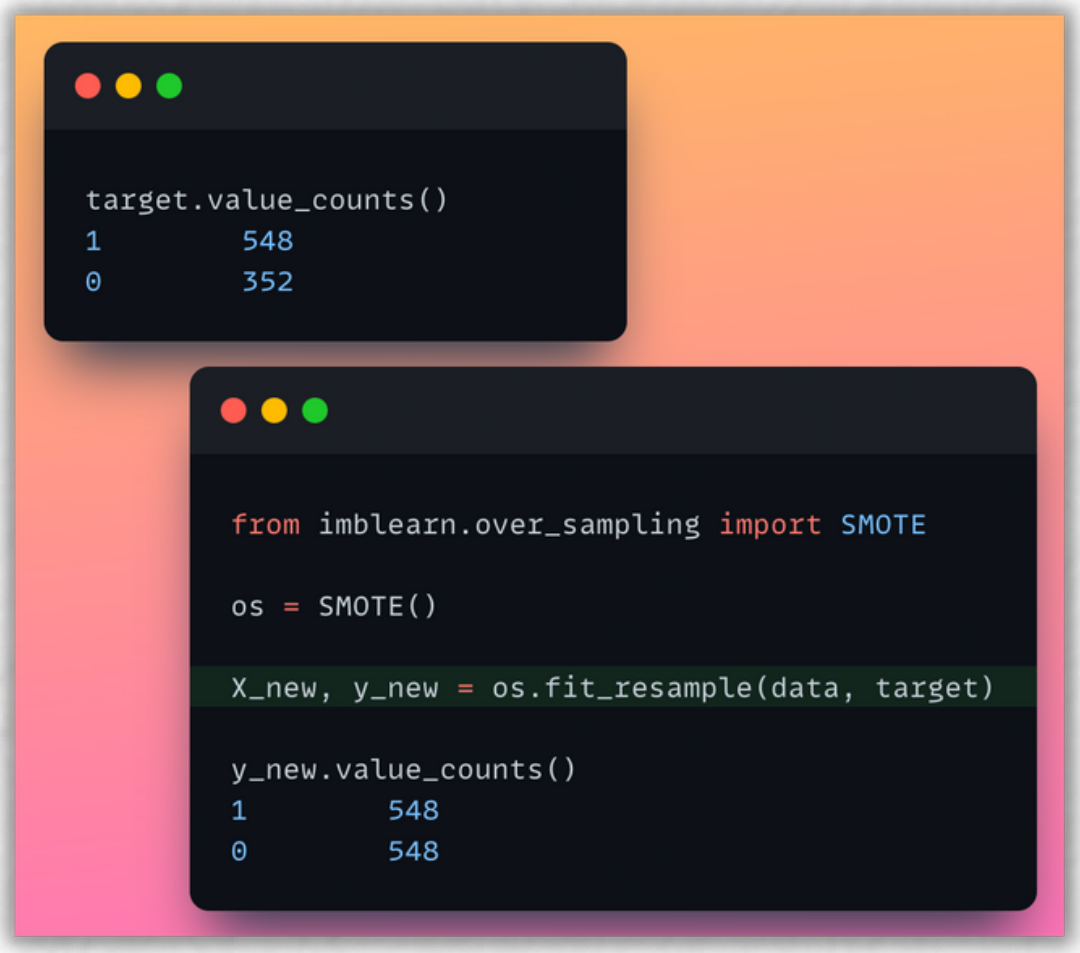
Matplotlib Sklearn

2. PyCaret



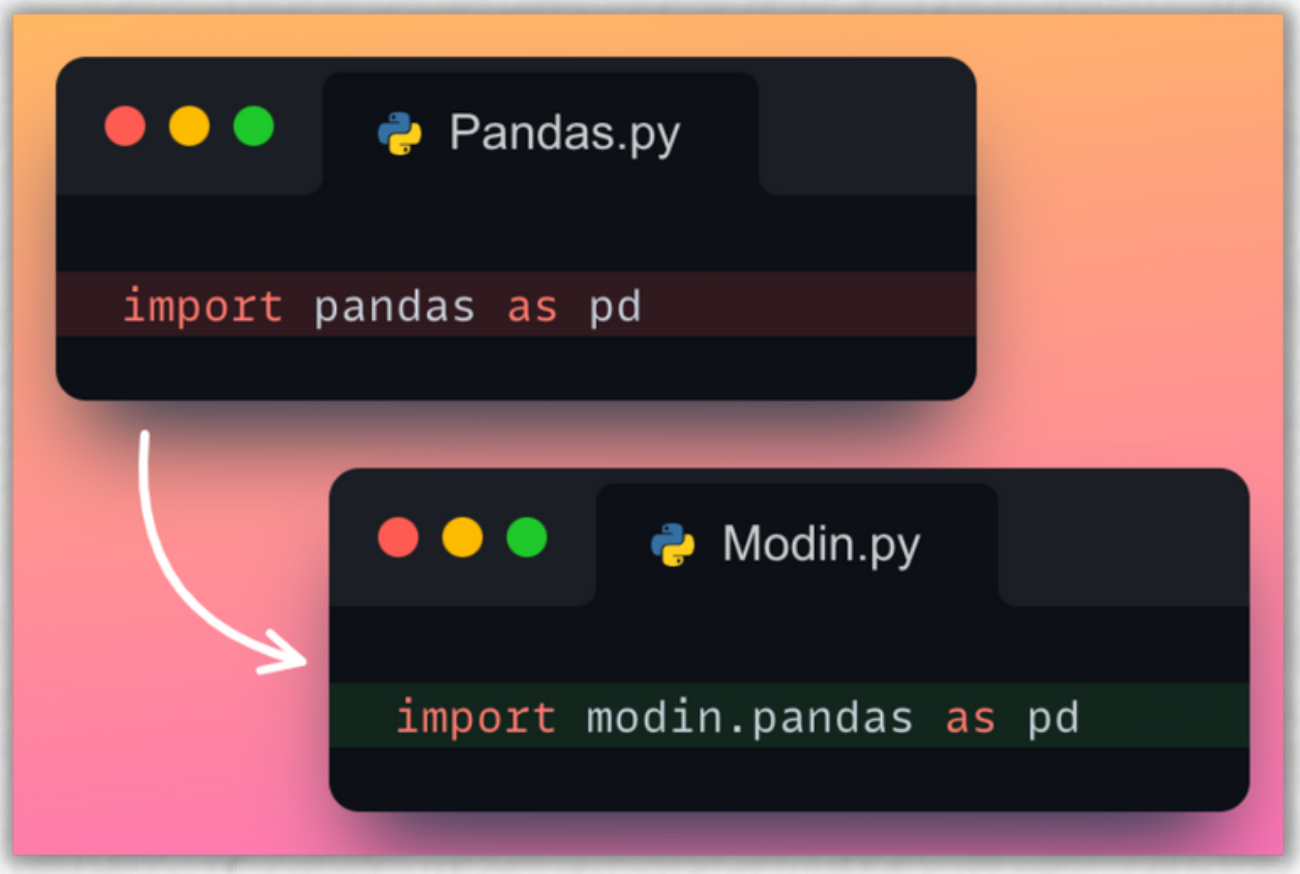
Automate ML workflows with this low-code library.

3. imbalanced-learn



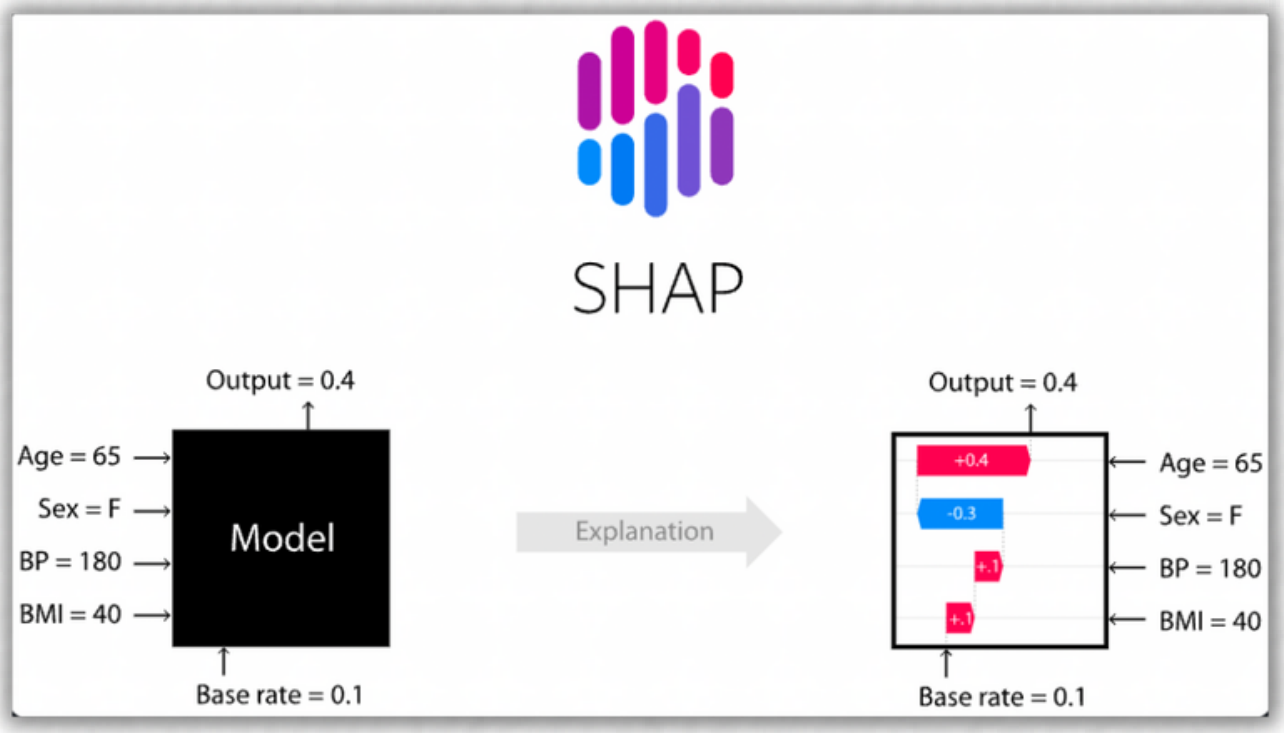
A variety of methods to handle class imbalance.

4. Modin



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

5. SHAP

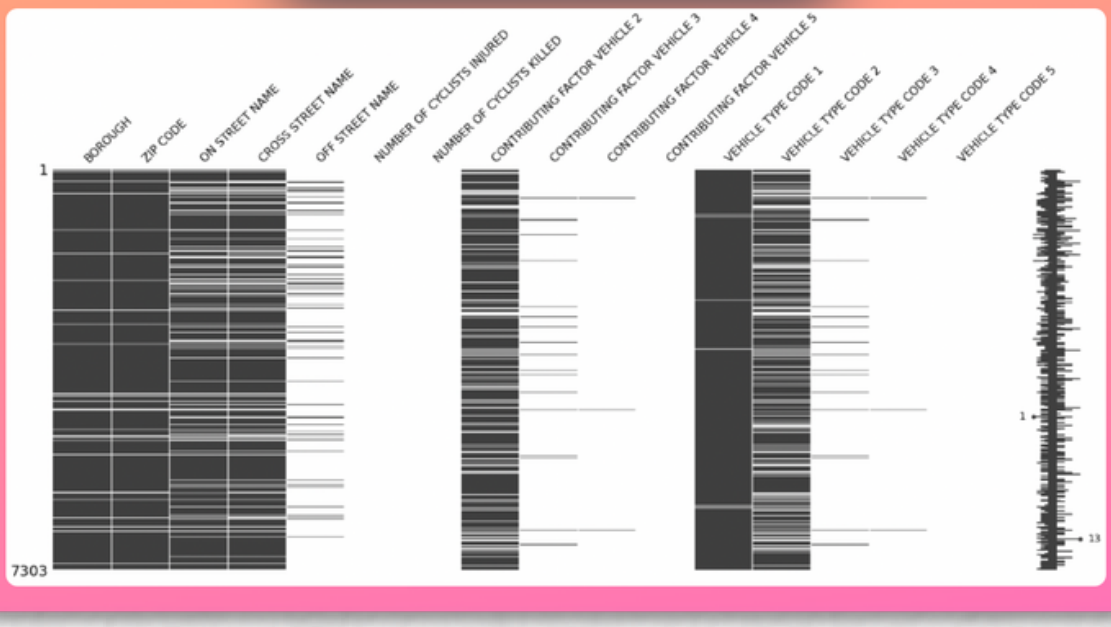


Explain the output of any ML model in few lines of code.

6. Missingno

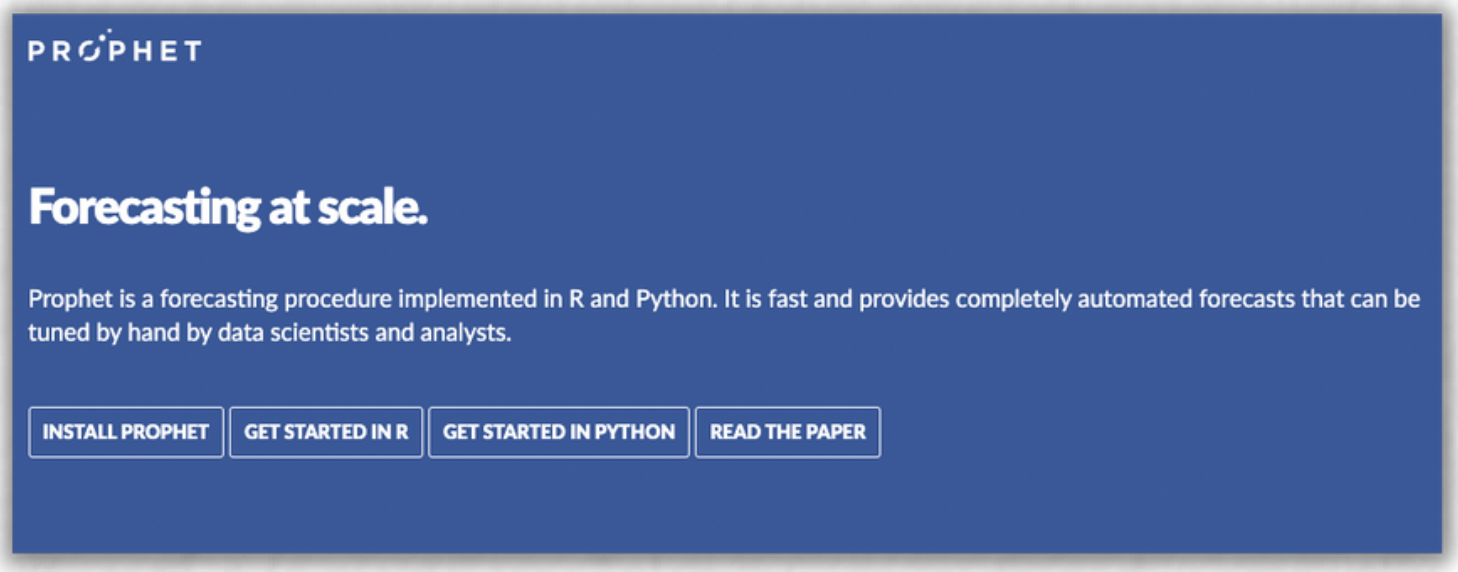
Missing Data

```
import missingno as msno

msno.matrix(df)
```



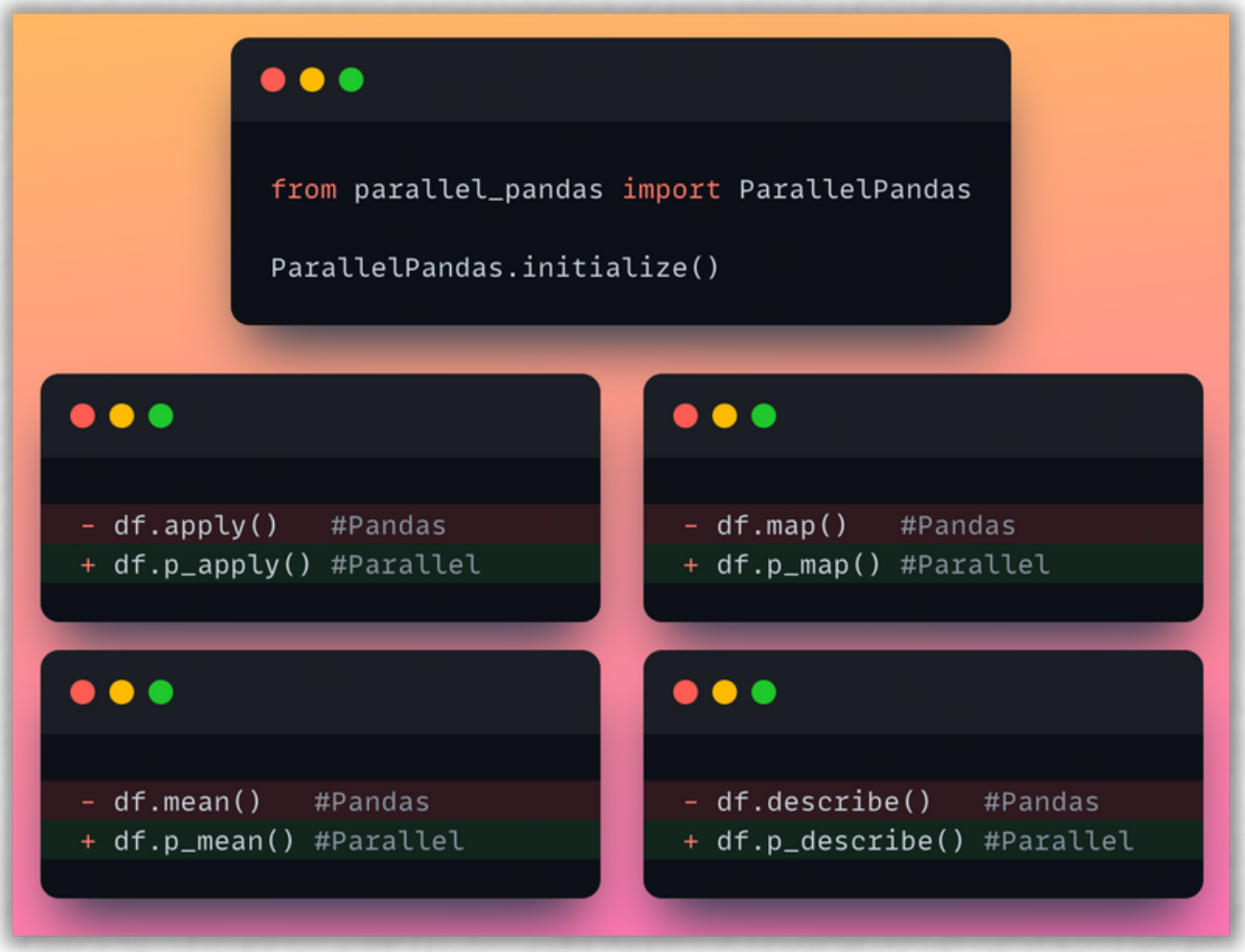
Visualize missing values in your dataset with ease.

7. Prophet



Produce high-quality forecasts on time-series data.

8. Parallel-Pandas



Parallelize Pandas across all CPU cores for faster computation.

9. Featuretools

Automated feature engineering for ML models.

10. Lazy Predict

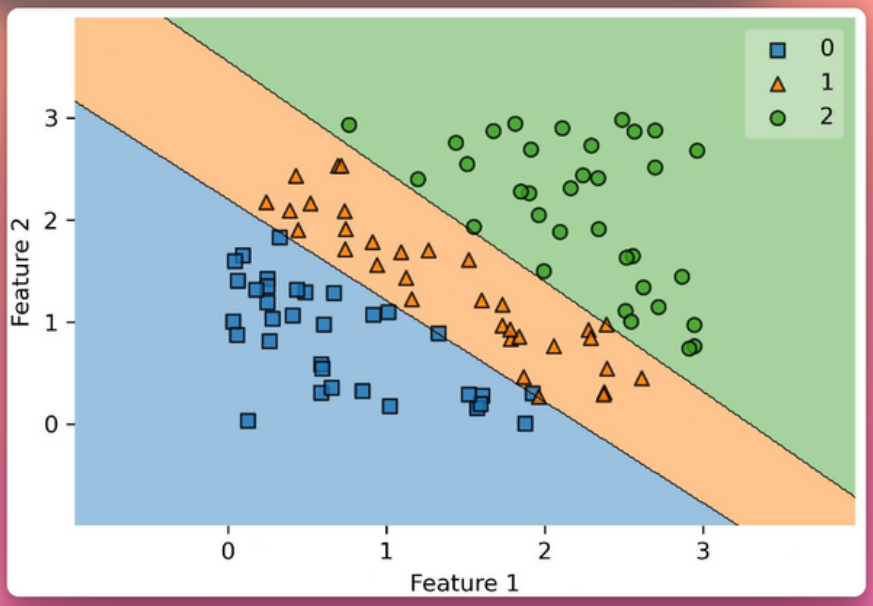
```python
from lazypredict.Supervised import LazyRegressor

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

```
from mlxtend.plotting import plot_decision_regions

model = LogisticRegression().fit(X, y)

plot_decision_regions(X, y, model)
```



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

model.py

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

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

Define Pytorch model

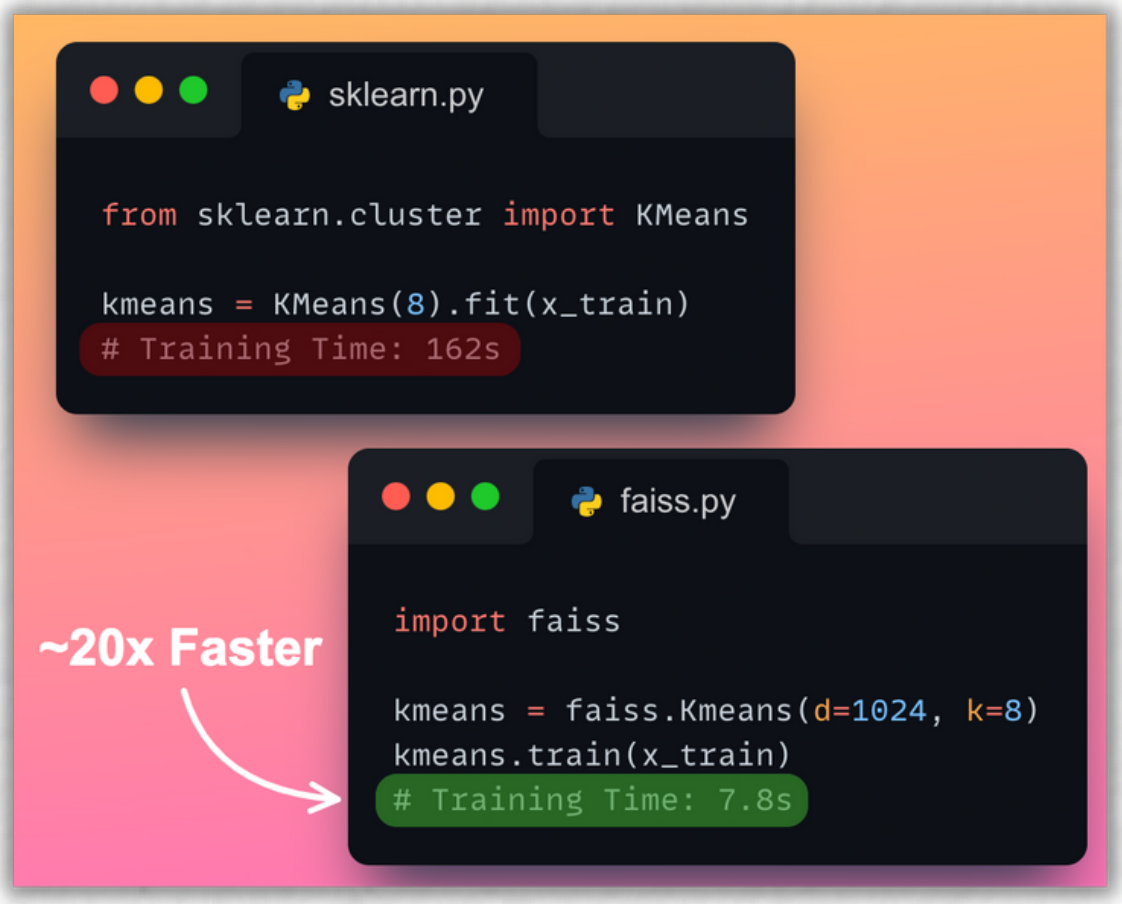
Use Scikit-learn API on model

```
from skorch import NeuralNetClassifier

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

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

PyTorch Sklearn

Leverage the power of PyTorch with the elegance of sklearn.

15. Faiss



Efficient algorithms for **similarity search** and **clustering** dense vectors.

16. statsmodel

```python
import statsmodels.api as sm

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

OLS Regression Results

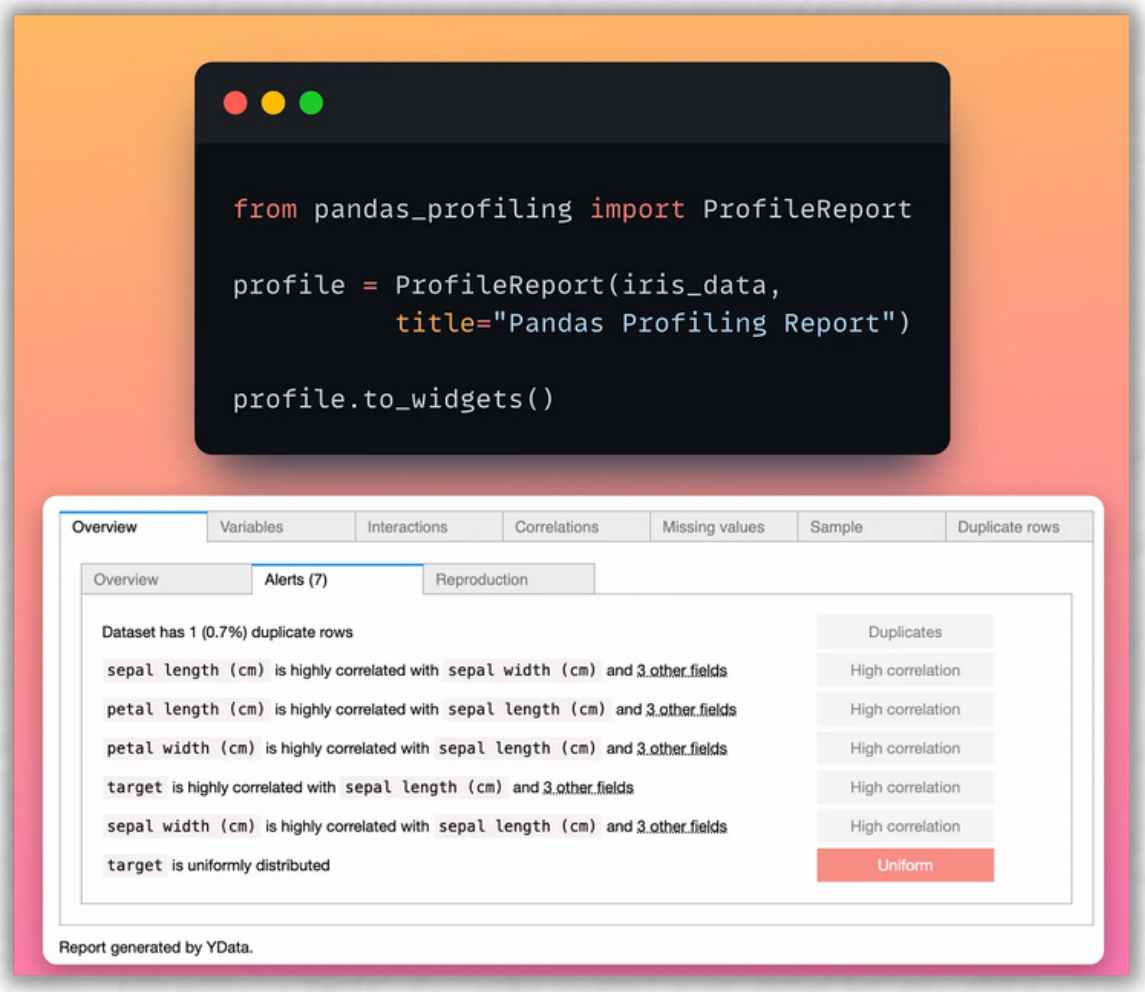
Dep. Variable:	y	R-squared:	0.178
Model:	OLS	Adj. R-squared:	0.161
Method:	Least Squares	F-statistic:	10.51
Date:	Wed, 02 Nov 2022	Prob (F-statistic):	7.41e-05
Time:	17:12:45	Log-Likelihood:	-20.926
No. Observations:	100	AIC:	47.85
Df Residuals:	97	BIC:	55.67
Df Model:	2		
Covariance Type:	nonrobust		

	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

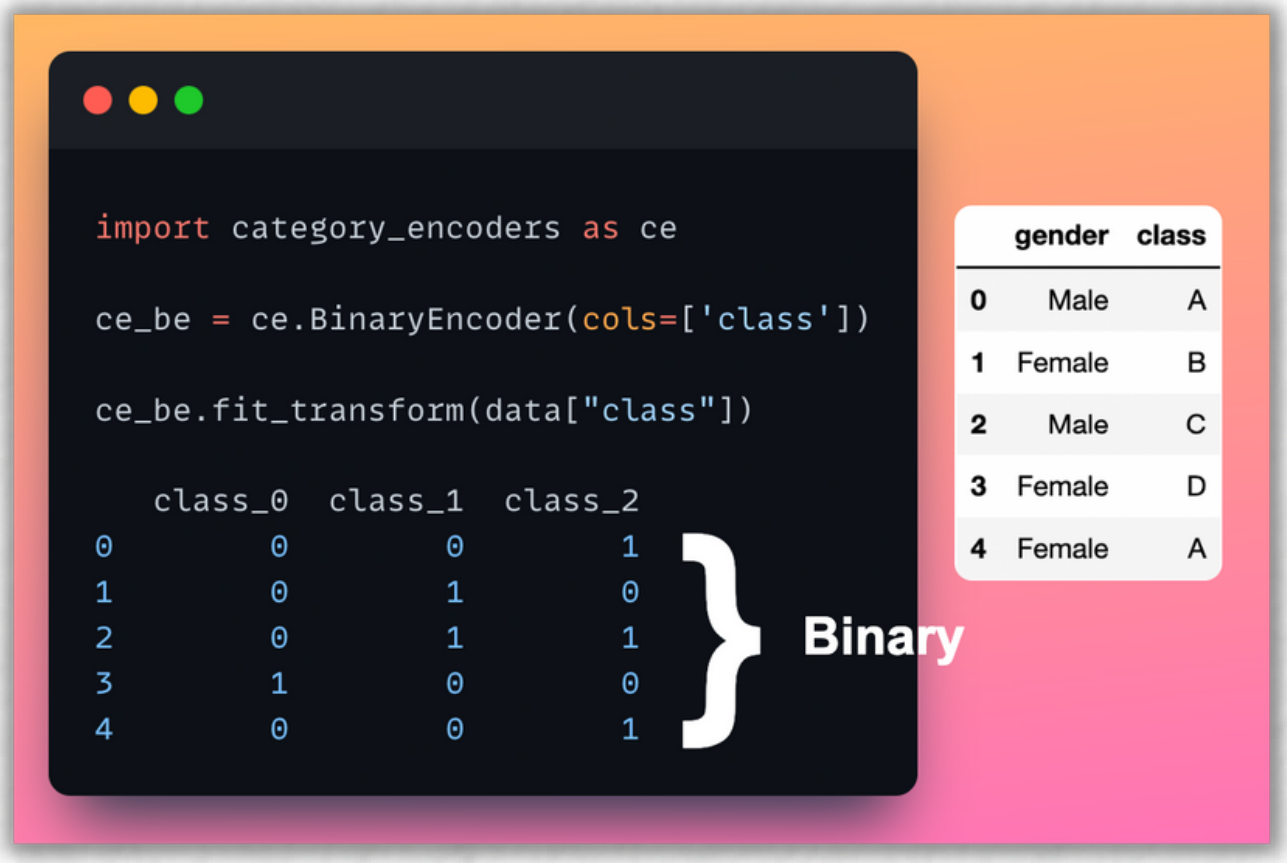


Generate a high-level **EDA report** of your data in no time.

18. Streamlit

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

19. Category-encoders



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.

22. Pytest

An elegant testing framework to test your code.

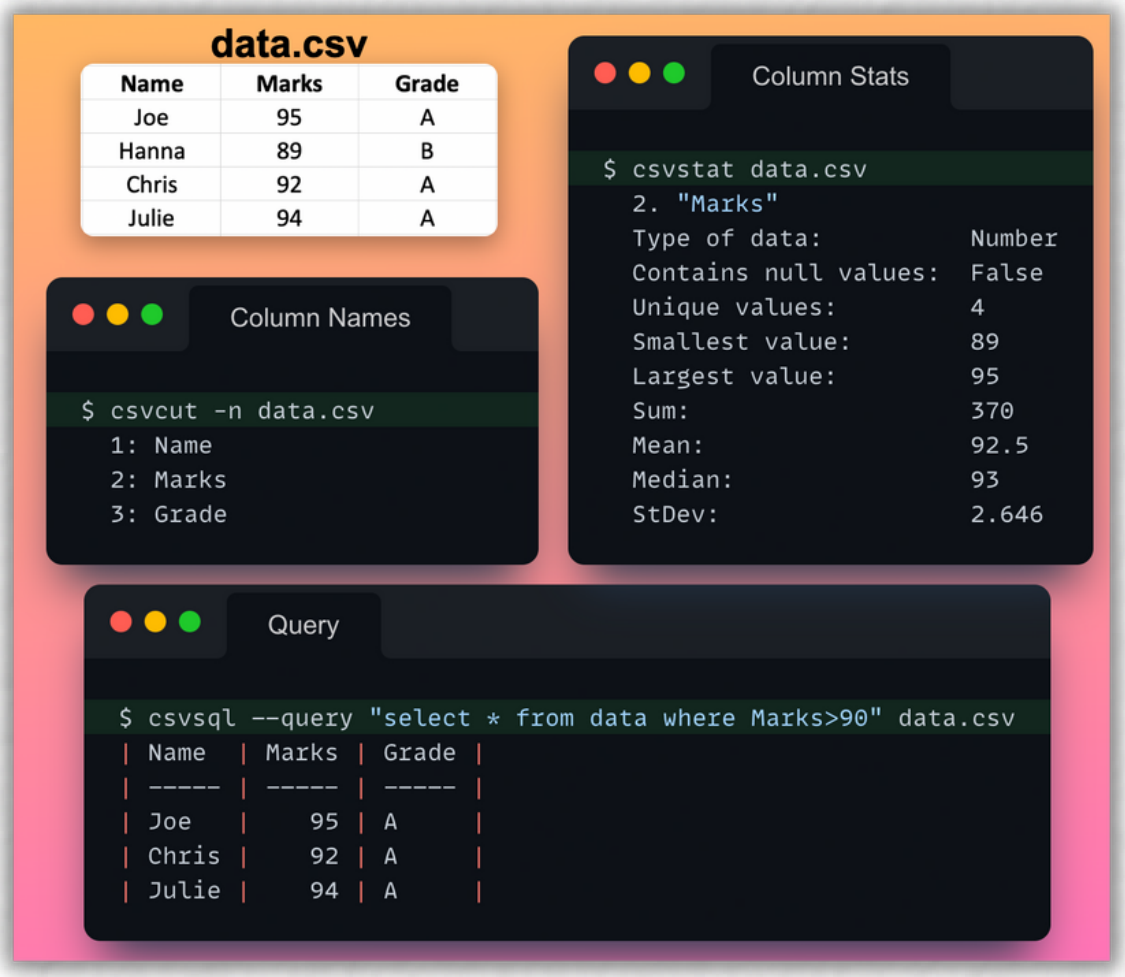
23. Numexpr

```
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

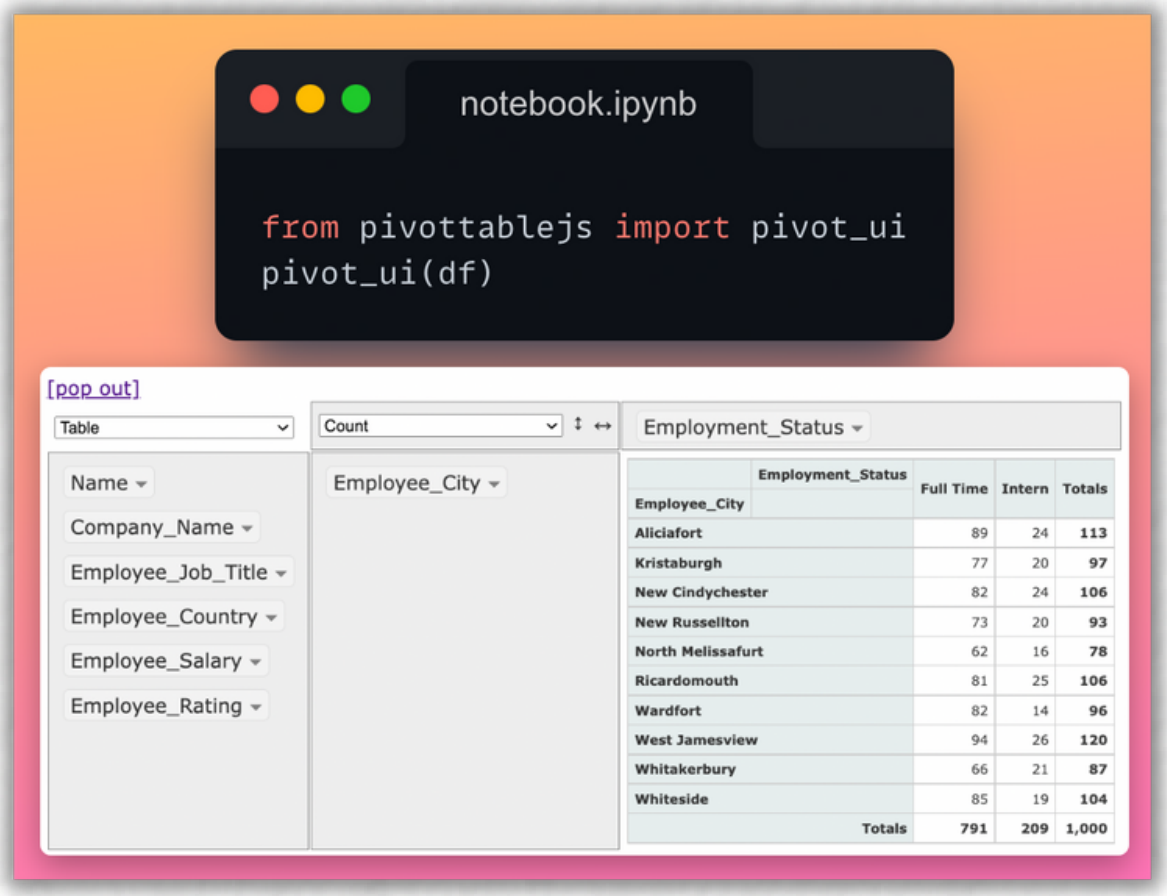
Parallelize NumPy to all CPU cores for 20x speedup.

24. CSV-Kit



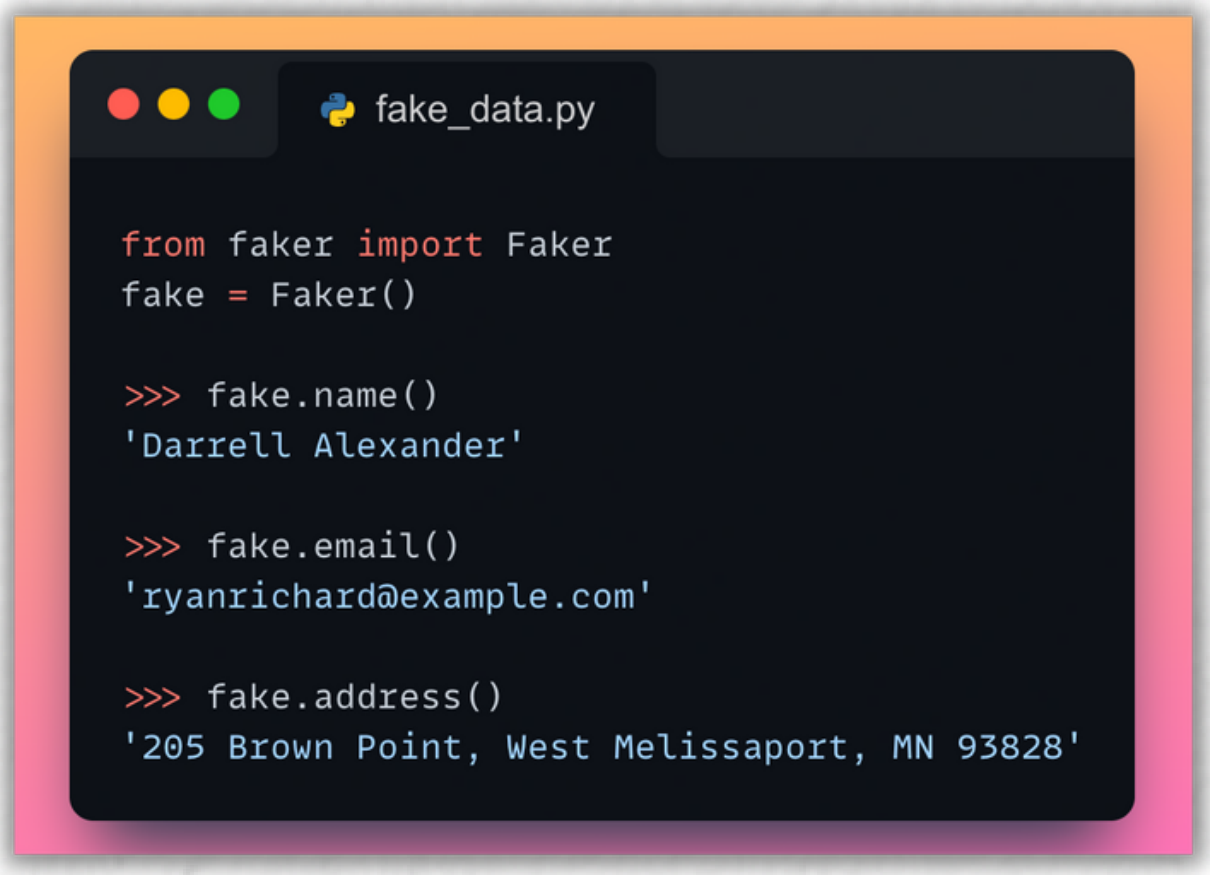
Explore, query and describe CSV files from terminal.

25. PivotTableJS



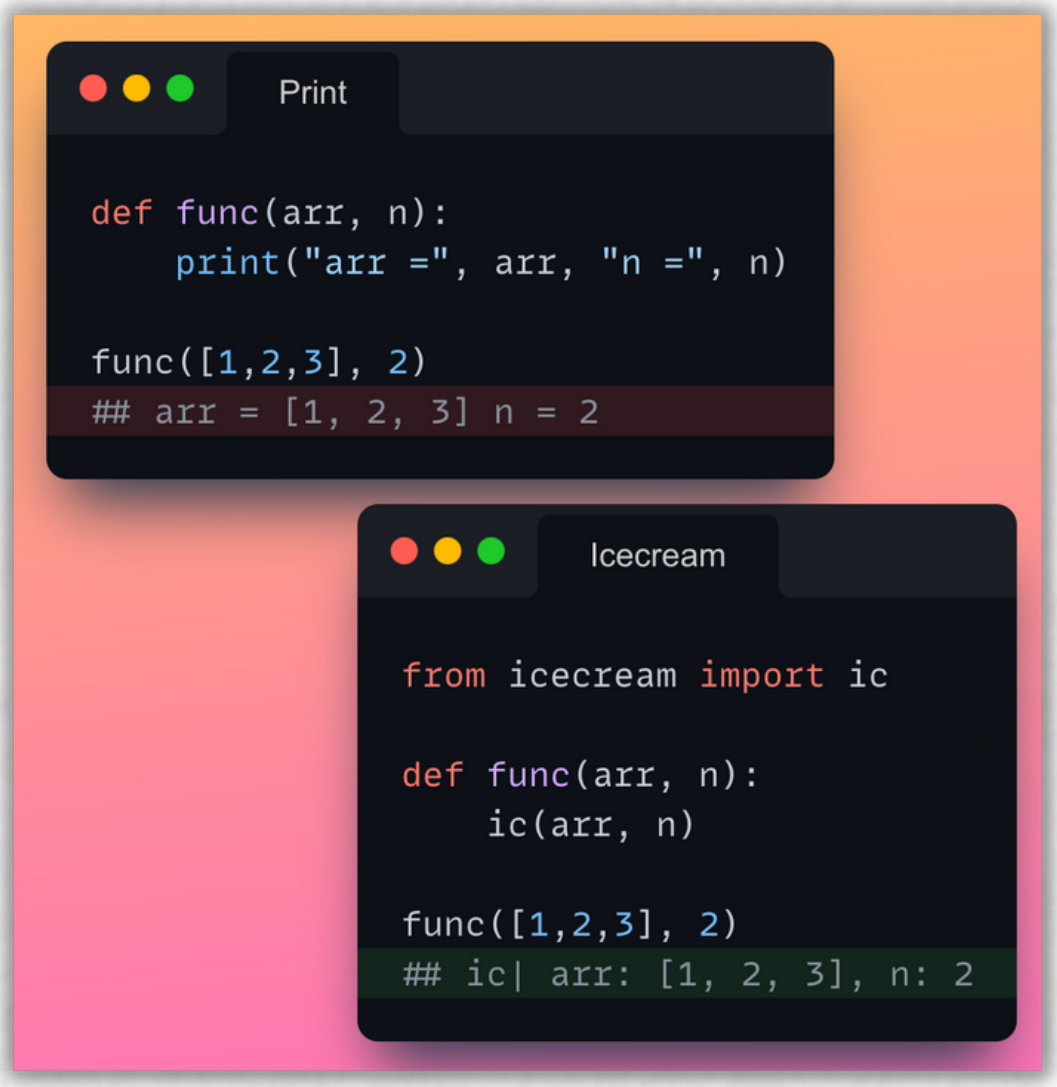
Drap-n-drop tools to group, pivot, plot dataframe.

26. Faker



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

27. Icecream



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

28. Pyforest

No need to write imports. Automatic package import.

29. PySnooper

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

30. Sidetable

sidetable.py

```
import sidetable

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

value_counts()

```
City
West Jamesview          120
Aliciafort              113
New Cindychester        106
Ricardomouth            106
Whiteside               104
Kristaburgh              97
Wardfort                 96
New Russellton           93
Whitakerbury             87
North Melissafurt        78
```

	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