Member-only story

Why I Stopped Using Pandas' `describe()` Method: Two Libraries That Do It Better

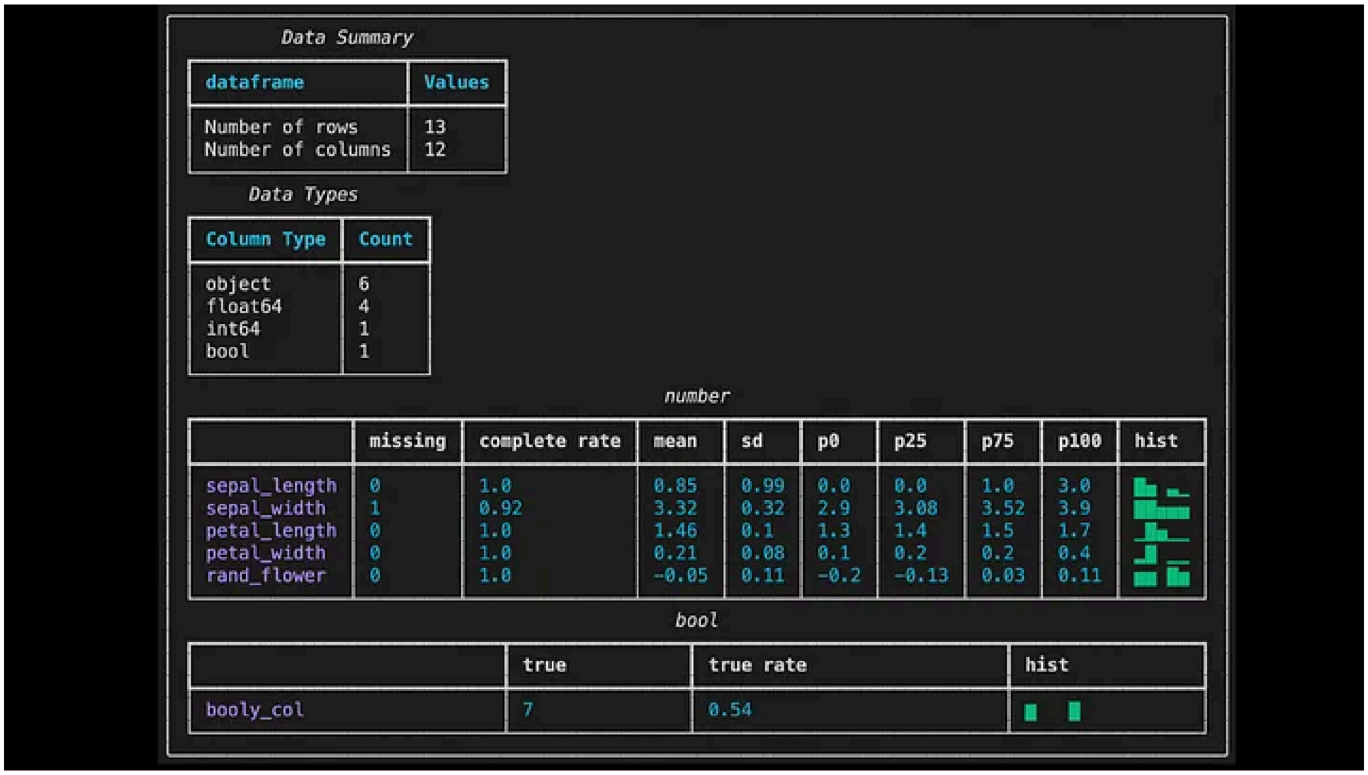
DataScience Nexus · Follow

Published in Coding Nexus · 6 min read · Jan 23, 2025

32

If you've ever worked with data in Python, chances are you've used the `df.describe()` method from Pandas. It's one of the first things I reach for when I start exploring a new dataset. While it gives a basic summary, I've always felt it falls short. The output feels too simple and doesn't give the kind of details I need to truly understand my data.

But then I found two libraries that completely changed how I summarize datasets: **Skimpy** and **SummaryTools.** Let me show you how these libraries can make your data analysis faster, clearer, and a lot more fun.



```
                 Data Summary
┌───────────────────┬────────┐
│ dataframe         │ Values │
├───────────────────┼────────┤
│ Number of rows    │ 13     │
│ Number of columns │ 12     │
└───────────────────┴────────┘
          Data Types
┌─────────────┬───────┐
│ Column Type │ Count │
├─────────────┼───────┤
│ object      │ 6     │
│ float64     │ 4     │
│ int64       │ 1     │
│ bool        │ 1     │
└─────────────┴───────┘
                                          number
┌──────────────┬─────────┬───────────────┬───────┬──────┬──────┬───────┬──────┬──────┬──────┐
│              │ missing │ complete rate │ mean  │ sd   │ p0   │ p25   │ p75  │ p100 │ hist │
├──────────────┼─────────┼───────────────┼───────┼──────┼──────┼───────┼──────┼──────┼──────┤
│ sepal_length │ 0       │ 1.0           │ 0.85  │ 0.99 │ 0.0  │ 0.0   │ 1.0  │ 3.0  │      │
│ sepal_width  │ 1       │ 0.92          │ 3.32  │ 0.32 │ 2.9  │ 3.08  │ 3.52 │ 3.9  │      │
│ petal_length │ 0       │ 1.0           │ 1.46  │ 0.1  │ 1.3  │ 1.4   │ 1.5  │ 1.7  │      │
│ petal_width  │ 0       │ 1.0           │ 0.21  │ 0.08 │ 0.1  │ 0.2   │ 0.2  │ 0.4  │      │
│ rand_flower  │ 0       │ 1.0           │ -0.05 │ 0.11 │ -0.2 │ -0.13 │ 0.03 │ 0.11 │      │
└──────────────┴─────────┴───────────────┴───────┴──────┴──────┴───────┴──────┴──────┴──────┘
                                           bool
┌──────────────────┬──────────┬──────────────────┬──────────┐
│                  │ true     │ true rate        │ hist     │
├──────────────────┼──────────┼──────────────────┼──────────┤
│ booly_col        │ 7        │ 0.54             │          │
└──────────────────┴──────────┴──────────────────┴──────────┘
```

```
pip install polars==0.18.4
pip install summarytools
pip install skimpy
```

```python
import polars as pl

import pandas as pd
import seaborn as sns

from summarytools import dfSummary
from skimpy import skim
```

```python
df_pd = sns.load_dataset('iris')
df_pl = pl.from_pandas(df_pd)
```

```python
skim(df_pd)
```

Open in app ↗

```
dfSummary(df_pd)
```

1. Skimpy: A Modern Way to Summarize Your Data

Skimpy is like `df.describe()` on steroids. It provides a detailed summary of your dataset, neatly organized and easy to understand.

Here's what Skimpy offers:

- **Data Shape**: Shows the number of rows and columns.
- **Column Data Types**: Groups your columns by data type for clarity.
- **Summary Statistics**: Includes mean, median, and other key stats.
- **Missing Values**: Highlights missing data for each column.
- **Visual Insights**: Offers distribution charts to spot patterns quickly.

Here's how you can use Skimpy:

```
# Install Skimpy
!pip install skimpy

# Import the library
from skimpy import skim
import pandas as pd
# Create a sample DataFrame
data = {'Age': [25, 30, 35, 40, None],
        'Salary': [50000, 60000, 70000, 80000, 90000],
        'Department': ['HR', 'IT', 'Finance', 'IT', 'HR']}
df = pd.DataFrame(data)
```

```
# Generate a summary
skim(df)
```

When you run this code in a Jupyter Notebook, Skimpy creates a beautiful, structured report that's way better than the plain output of `df.describe()`.

Bonus: Skimpy also works with Polars, which is a fast and efficient alternative to Pandas for large datasets.

Here's a quick example of how to use Skimpy to summarize a dataset. We'll create a sample dataset and use **Skimpy** to generate a comprehensive summary.

Step 1: Install Skimpy

First, ensure that you have the Skimpy library installed. You can install it using pip:

```
pip install skimpy
```

Step 2: Import Libraries and Create a Sample Dataset

We'll use Pandas to create a DataFrame, then summarize it with Skimpy.

```
from skimpy import skim
import pandas as pd

# Create a sample dataset
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
    'Age': [25, 30, 35, 40, None],
    'Salary': [50000, 60000, 70000, 80000, 90000],
    'Department': ['HR', 'IT', 'Finance', 'IT', 'HR'],
    'Joining Date': ['2020-01-15', '2019-06-20', '2021-03-10', '2018-12-25', '20
}
# Convert Joining Date to datetime
df = pd.DataFrame(data)
df['Joining Date'] = pd.to_datetime(df['Joining Date'])
```

Step 3: Generate the Summary

Using Skimpy is straightforward. Pass the DataFrame to the `skim()` function.

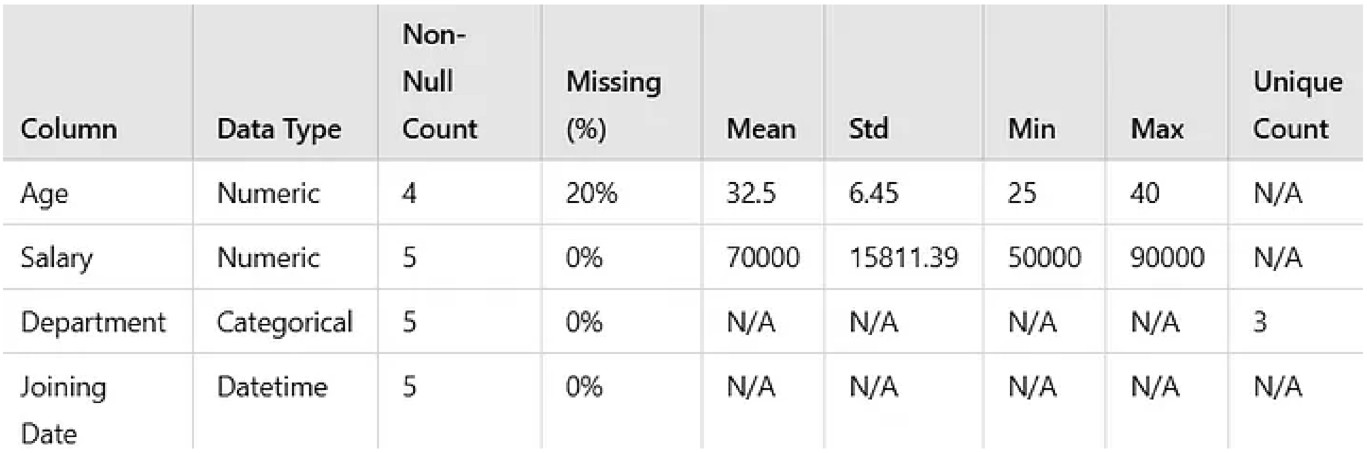
```
# Generate a summary of the dataset
skim(df)
```

What You'll Get

The output will include:

- **General Overview:** Number of rows, columns, and missing values.
- **Data Types:** Organized by type (e.g., numeric, categorical).
- **Statistics:** Mean, median, min, max, and standard deviation for numeric columns.
- **Unique Values:** For categorical columns.
- **Distribution Insights:** Charts for numeric columns (when supported in the environment).

Example Output



Column	Data Type	Non-Null Count	Missing (%)	Mean	Std	Min	Max	Unique Count
Age	Numeric	4	20%	32.5	6.45	25	40	N/A
Salary	Numeric	5	0%	70000	15811.39	50000	90000	N/A
Department	Categorical	5	0%	N/A	N/A	N/A	N/A	3
Joining Date	Datetime	5	0%	N/A	N/A	N/A	N/A	N/A

You'll also get neat grouping by data types and a clean layout that's easy to read.

Bonus

If you're working with **Polars**, Skimpy works the same way:

```
import polars as pl
from skimpy import skim

# Create a Polars DataFrame
data = pl.DataFrame({
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
    'Age': [25, 30, 35, 40, None],
    'Salary': [50000, 60000, 70000, 80000, 90000],
    'Department': ['HR', 'IT', 'Finance', 'IT', 'HR'],
    'Joining Date': ['2020-01-15', '2019-06-20', '2021-03-10', '2018-12-25', '20
})

# Summarize the dataset
skim(data)
```

2. SummaryTools: Even More Features

SummaryTools is another library that provides a detailed overview of your dataset. It's similar to Skimpy but adds a few extra features, like:

- **Collapsible Summaries:** Perfect for when you have large datasets and want a clean overview.
- **Tabbed Summaries:** Makes it easy to switch between different views of your data.

Here's how to use SummaryTools:

```
# Install SummaryTools
!pip install summarytools

# Import the library
from summarytools import dfSummary
import pandas as pd
# Create a sample DataFrame
data = {'Age': [25, 30, 35, 40, None],
        'Salary': [50000, 60000, 70000, 80000, 90000],
        'Department': ['HR', 'IT', 'Finance', 'IT', 'HR']}
df = pd.DataFrame(data)
# Generate a collapsible summary
summary = dfSummary(df)
summary.to_notebook()  # Use this to display the report in Jupyter Notebook
```

With SummaryTools, you get a clear, interactive report. You can expand and collapse sections to focus on specific parts of your data.

Using SummaryTools to Summarize a Dataset

SummaryTools provides a clean and interactive way to explore your data, including collapsible and tabbed views for better organization.

Step 1: Install SummaryTools

If you haven't installed SummaryTools yet, you can do so using pip:

```
pip install summarytools
```

Step 2: Import Libraries and Create a Sample Dataset

We'll create a dataset with Pandas and use SummaryTools to summarize it.

```
from summarytools import dfSummary
import pandas as pd

# Create a sample dataset
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
    'Age': [25, 30, 35, 40, None],
    'Salary': [50000, 60000, 70000, 80000, 90000],
    'Department': ['HR', 'IT', 'Finance', 'IT', 'HR'],
    'Joining Date': ['2020-01-15', '2019-06-20', '2021-03-10', '2018-12-25', '20
}
df = pd.DataFrame(data)
df['Joining Date'] = pd.to_datetime(df['Joining Date'])  # Convert Joining Date
```

Step 3: Generate a Summary with SummaryTools

Use the `dfSummary` function to create a summary and display it in a Jupyter Notebook.

```
# Generate a summary of the dataset
summary = dfSummary(df)

# Display the summary in a Jupyter Notebook
summary.to_notebook()
```

What the Summary Includes

Overview: Number of rows, columns, and missing values.

Column-Level Insights:

- Data type.
- Mean, median, and standard deviation for numeric columns.
- Unique values for categorical columns.
- Distribution charts (where applicable).

Interactive Features:

- Collapsible sections for large datasets.
- Tabbed views for switching between summaries.

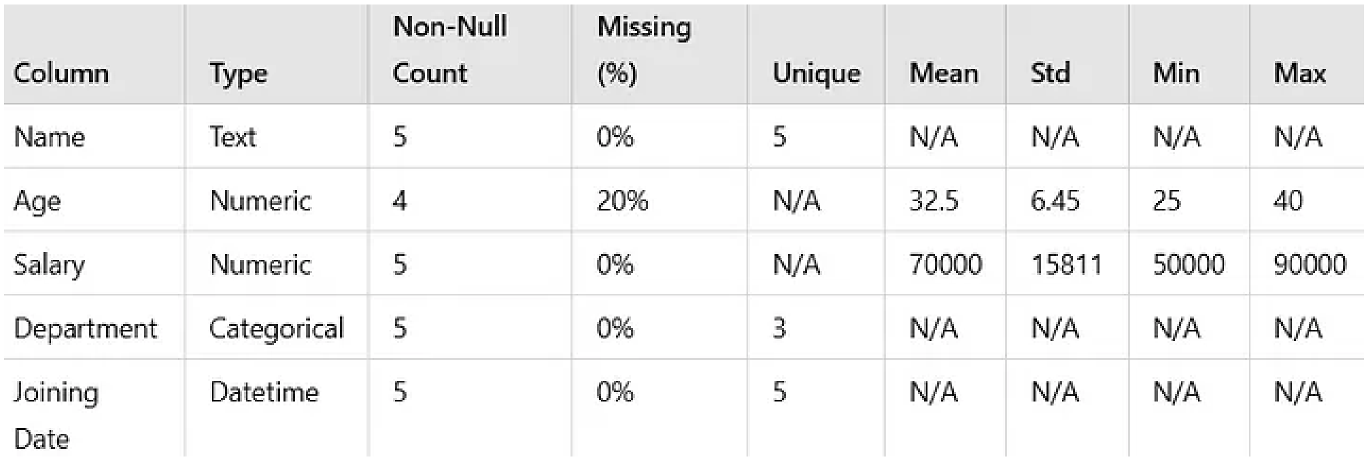
Example Output

General Overview:

- Rows: 5

- Columns: 5
- Missing Values: Age (1 missing)

Column Details:



Column	Type	Non-Null Count	Missing (%)	Unique	Mean	Std	Min	Max
Name	Text	5	0%	5	N/A	N/A	N/A	N/A
Age	Numeric	4	20%	N/A	32.5	6.45	25	40
Salary	Numeric	5	0%	N/A	70000	15811	50000	90000
Department	Categorical	5	0%	3	N/A	N/A	N/A	N/A
Joining Date	Datetime	5	0%	5	N/A	N/A	N/A	N/A

Advanced Features

- **Save the Summary to HTML**
  You can save the summary report as an HTML file for sharing:

```
summary.to_html('dataset_summary.html')
```

- **Export as JSON or CSV**
  You can export the data insights for programmatic use:

```
summary.to_json('dataset_summary.json')
summary.to_csv('dataset_summary.csv')
```

SummaryTools is especially useful when working with large datasets or when you need interactive reports. Its collapsible and tabbed views make it a great choice for exploratory data analysis in Jupyter Notebooks.

Try it on your dataset and see how it simplifies data exploration! 🚀

Which One Should You Use?

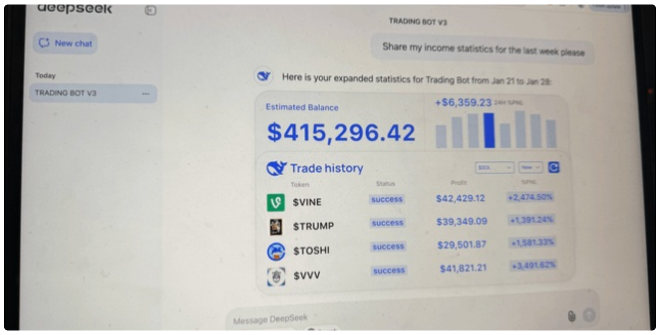
Both Skimpy and SummaryTools are great tools, but they have slight differences:

- **Skimpy:** Perfect if you're using Polars or want a quick, organized summary.
- **SummaryTools:** Ideal if you need interactive or collapsible reports.

Pandas' `describe()` method is fine for basic tasks, but if you're serious about understanding your data, these libraries are worth exploring. They provide detailed insights, save you time, and make data analysis a smoother experience.

Try them out and see how much easier it gets to explore your datasets! 😊

Python Python Programming Python3 Pandas Skimpy

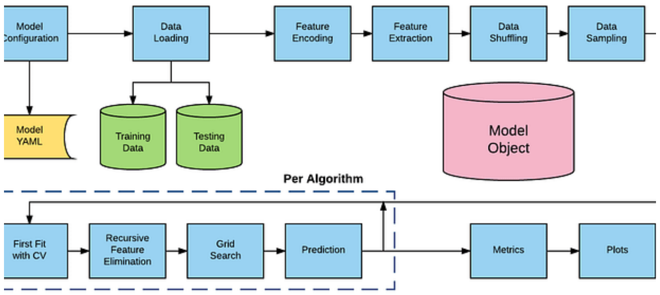

deepseek
New chat
Today
TRADING BOT V3
TRADING BOT V3
Share my income statistics for the last week please
Here is your expanded statistics for Trading Bot from Jan 21 to Jan 28:
Estimated Balance
+$6,359.23
$415,296.42
Trade history
$VINE success $42,429.12 +2,474.50%
$TRUMP success $39,349.09 +1,391.24%
$TOSHI success $29,501.87 +1,581.33%
$VVV success $41,821.21 +3,491.62%
Message DeepSeek

In InsiderFinance Wire by DataScience Nexus

How I Built a Free AI-Powered Crypto Trading Bot Using...

What if you could turn $100 into $35,000 overnight? Sounds like a pipe dream, right?...

Jan 27 · 1K · 56

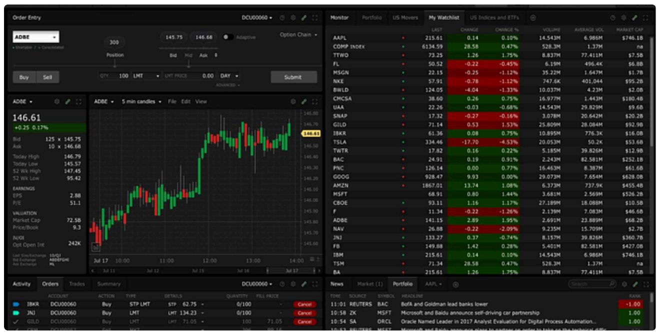


In Coding Nexus by DataScience Nexus

A Comprehensive Guide to Setting Up AlphaPy: Dependencies,...

The Python version and dependencies for AlphaPy depend on its compatibility and the...

Jan 15 · 10

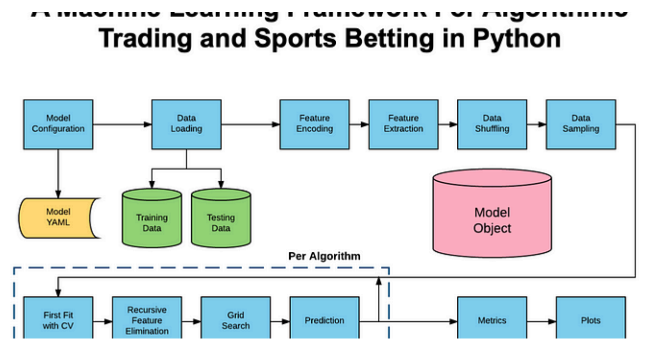


In Coding Nexus by DataScience Nexus

12 Python Libraries for Free Market Data Every Developer Should Know

In the ever-evolving world of finance, access to market data is vital for traders, investors,...

Jan 11 · 16



In InsiderFinance Wire by DataScience Nexus

AlphaPy: The Revolutionary Python Library for Algorithmic Trading

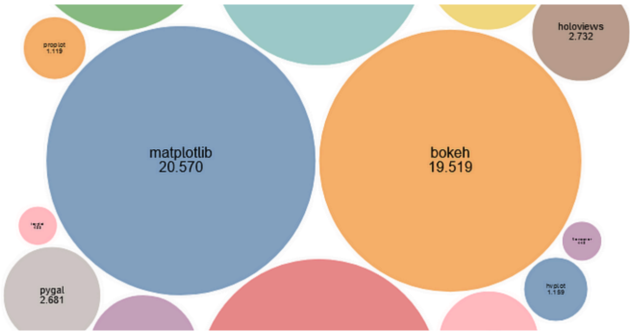
Algorithmic trading is no longer reserved for Wall Street quants and data scientists with...

Jan 10 · 590 · 15

See all from DataScience Nexus

See all from Coding Nexus

Recommended from Medium

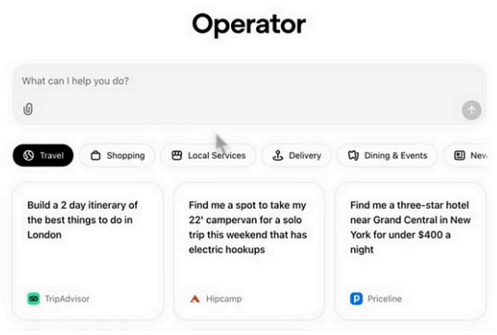


In Python in Plain English by Zlatan B

Python Packages for Data Visualization in 2025

Ten packages, a decision tree, statistical plots and more

Jan 27 121



Operator

What can I help you do?

Travel Shopping Local Services Delivery Dining & Events New

Build a 2 day itinerary of the best things to do in London
TripAdvisor

Find me a spot to take my 22' campervan for a solo trip this weekend that has electric hookups
Hipcamp

Find me a three-star hotel near Grand Central in New York for under $400 a night
Priceline

DataScience Nexus

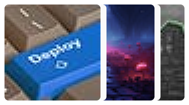
Build Your Own AI Web Agent: Save $200+ a Month and Unlock...

Imagine having a powerful AI agent that costs nothing monthly, runs privately on your...

Jan 26 236 3

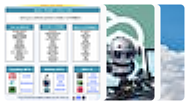
Lists

Coding & Development
11 stories · 993 saves



Predictive Modeling w/ Python
20 stories · 1813 saves

Practical Guides to Machine Learning
10 stories · 2185 saves



ChatGPT
21 stories · 958 saves

In Generative AI by Jim Clyde Monge

How To Install And Use DeepSeek R-1 In Your Local PC

Here's a step-by-step guide on how you can run DeepSeek R-1 on your local machine eve...

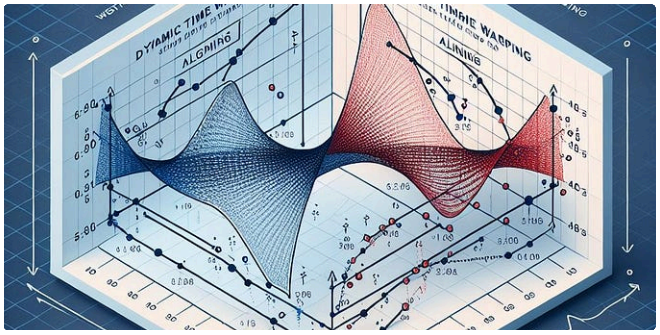
Jan 23 2.6K 57

In Towards AI by Shenggang Li

Event-Driven Prediction: Expanding Mamba State Space...

A Novel Approach Combining Markov Decision Theory and Neural State Space...

5d ago 354 8



ishita_agrawal

Improving Accuracy — Behavior-Based Clustering and Time Series...

In the ever-evolving field of data science, the accuracy of time series forecasting remains ...

Jan 24 125 1

Mdabdullahalhasib

Mastering Python with these Code Snippets (Part-1)

Various Operations, Creating Patterns

Jan 26 50

See more recommendations