TabPFN the new XGBoost, easy models for Efficient High Performance Machine Learning

Abish Pius · Follow
Published in Writing in the World of Artificial Intelligence
4 min read · Jun 15

Listen Share More

TabPFN the New King of Tabular Machine Learning

Full Publication Link: 2207.01848.pdf (arxiv.org)

Tabular data analysis is a fundamental task in machine learning and data science, with numerous applications across various domains. In recent years, researchers have made significant advancements in developing models specifically designed for tabular data. One such model is the TabPFN (Tabular Prior-Data Fitted Network), which introduces novel architectural enhancements and a unique prior for tabular data. In this article, we will delve into the details of the TabPFN and its contributions compared to existing methods.

The TabPFN Architecture:

The TabPFN is a modified version of the original PFN architecture, designed to handle tabular data efficiently. Two key modifications have been made to the architecture. Firstly, slight adjustments have been made to the attention masks, resulting in shorter inference times. Secondly, the TabPFN has been equipped to handle datasets with varying numbers of features through zero-padding. These architectural enhancements, combined with the main contributions of the TabPFN, are discussed in detail in the appendix.

Training the TabPFN

In the prior-fitting phase, the TabPFN is trained on samples generated from a novel prior specifically designed for tabular data. The training process involves using a 12-layer Transformer and synthetic datasets. The training is computationally expensive, requiring significant time and computational resources. However, it is a one-time offline step done during algorithm development. The trained TabPFN model is then used for all subsequent experiments and evaluations.

Inference with the TabPFN

During the inference phase, the TabPFN approximates the Posterior Predictive Distribution (PPD) for the dataset prior. It captures the marginal predictions across different spaces of Structural Causal Models (SCMs) and Bayesian Neural Networks (BNNs), with a focus on simplicity and causal explanations for the data. The predictions are obtained through a single forward pass of the TabPFN, as well as an ensemble of 32 forward passes on modified datasets. These modifications involve power transformations and index rotations of feature columns and class labels.

A Prior for Tabular Data

The performance of the TabPFN relies on a suitable prior designed for tabular data. The prior incorporates distributions instead of point estimates for most hyperparameters. The notion of simplicity is at the core of the prior, aligning with Occam's Razor and the Speed Prior. The prior leverages SCMs and BNNs as

fundamental mechanisms for generating diverse data. SCMs capture causal relationships among columns in tabular data, while BNNs offer flexibility in modeling complex patterns. The prior also accounts for peculiarities of tabular data, such as correlated and categorical features, exponentially scaled data, and missing values.

Fundamentally Probabilistic Models

Unlike traditional models that rely on point estimates for hyperparameters, the TabPFN allows for full Bayesian treatment of hyperparameters. By defining a probability distribution over the hyperparameter space, including BNN

Multi-Class Prediction

To generate multi-class prediction the scalar labels obtained from the described priors are converted into discrete class labels. This transformation is done by dividing the values of the scalar labels into intervals that correspond to different class labels. It will automatically account for potential class imbalances as well.

TabPFN vs XGBoost Head-to-Head

	LightGBM	CatBoost	XGBoost	ASKL2.0	AutoGluon	TabPFN$_{n.e.}$	TabPFN	TabPFN + AutoGluon
M. rank AUC OVO	6.9722	4.9444	6.1944	4.4722	4	3.8056	2.9444	**2.6667**
Mean rank Acc.	6.8889	4.9722	6.0556	5.1667	3.8889	3.8889	2.8889	**2.25**
Mean rank CE	5.7778	5.4444	6	6.4167	3.1111	4.1389	3.0278	**2.0833**
Mean AUC OVO	0.92±.013	0.924±.011	0.924±.01	0.929±.0096	0.93±.0091	0.932±.0088	**0.934**±.0086	**0.934**±.0084
Mean Acc.	0.862±.012	0.864±.011	0.866±.011	0.87±.014	0.881±.01	0.873±.0095	0.879±.0089	**0.886**±.0094
Mean CE	0.75±.039	0.747±.029	0.759±.04	0.813±.073	0.714±.014	0.727±.021	0.716±.019	**0.711**±.014
Mean time (s) (Tune + Train + Predict)	3280	3746	3364	3601	3077	**1.301** (CPU) **0.0519** (GPU)	37.59 (CPU) 0.6172 (GPU)	3109 (CPU) 3077 (GPU)

Table 1: Results on the 18 numerical datasets in OpenML-CC18 for 60 minutes requested time per data-split. ± values indicate a metric's standard deviation. If available, all baselines optimize ROC AUC. TabPFN n.e. is a faster version of our method, without ensembling, while TabPFN + AutoGluon is an ensemble of TabPFN and AutoGluon.

- TabPFN achieves a better tradeoff between accuracy and training speed compared to all other methods, including XGBoost and other gradient boosting decision tree methods.
- TabPFN makes predictions within less than a second on one GPU, which is comparable to the performance of the best competitors (AutoML systems) after training for one hour.

- TabPFN outperforms tuned GBDT methods and performs better than XGBoost, CatBoost, and LightGBM.
- The simple baselines (LogReg, KNN) have lower performance overall.
- TabPFN is significantly faster than methods with comparable performance. It requires an average of 1.30s on a CPU and 0.05s on a GPU for prediction, resulting in a substantial speedup compared to baselines.
- TabPFN performs exceptionally well on purely numerical datasets, as confirmed by results on OpenML-CC18 Benchmark and additional validation datasets.
- In the OpenML-AutoML Benchmark, TabPFN outperforms all baselines in terms of mean cross-entropy, accuracy, and the OpenML Metric.

Code — Very Very Easy to Implement

It very much follows the basic Sklearn architecture.

```
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split

from tabpfn import TabPFNClassifier

X, y = load_breast_cancer(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, randc

# N_ensemble_configurations controls the number of model predictions that are e
# When N_ensemble_configurations > #features * #classes, no further averaging

classifier = TabPFNClassifier(device='cpu', N_ensemble_configurations=32)

classifier.fit(X_train, y_train)
y_eval, p_eval = classifier.predict(X_test, return_winning_probability=True)

print('Accuracy', accuracy_score(y_test, y_eval))
```

Improve your Gen AI App Productivity: https://medium.com/@abishpius/enhance-all-your-prompts-for-free-increase-productivity-and-gen-ai-app-effectiveness-10-fold-in-fa1d0bf9d91

My Gen AI App S.P.O.C.K: https://medium.com/@abishpius/introducing-s-p-o-c-k-my-first-stab-at-a-gen-ai-app-that-you-can-try-for-free-98bda5fc5a94

Plug: *Please purchase my book ONLY if you have the means to do so. Imagination Unleashed: Canvas and Color, Visions from the Artificial: Compendium of Digital Art Volume 1 (Artificial Intelligence Draws Art) — Kindle edition by P, Shaxib, A, Bixjesh. Arts & Photography Kindle eBooks @ Amazon.com.*

Donate by scanning QR

Machine Learning | Xgboost | Upcoming | Advanced

Follow

Written by Abish Pius

70 Followers · Editor for Writing in the World of Artificial Intelligence

Data Science Professional Python Enthusiast.

More from Abish Pius and Writing in the World of Artificial Intelligence

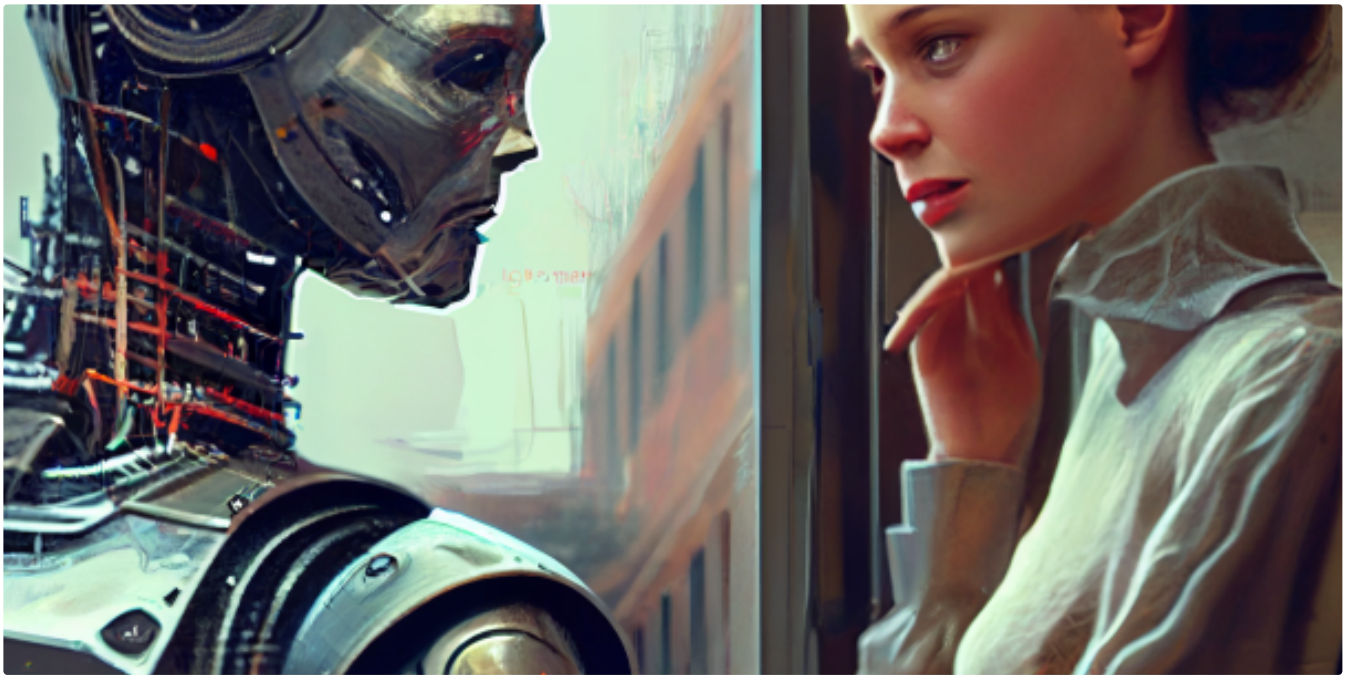
Abish Pius in Writing in the World of Artificial Intelligence

Adding Interpretability to PyTorch Models with Captum

Introduction:

7 min read · Jun 27

51

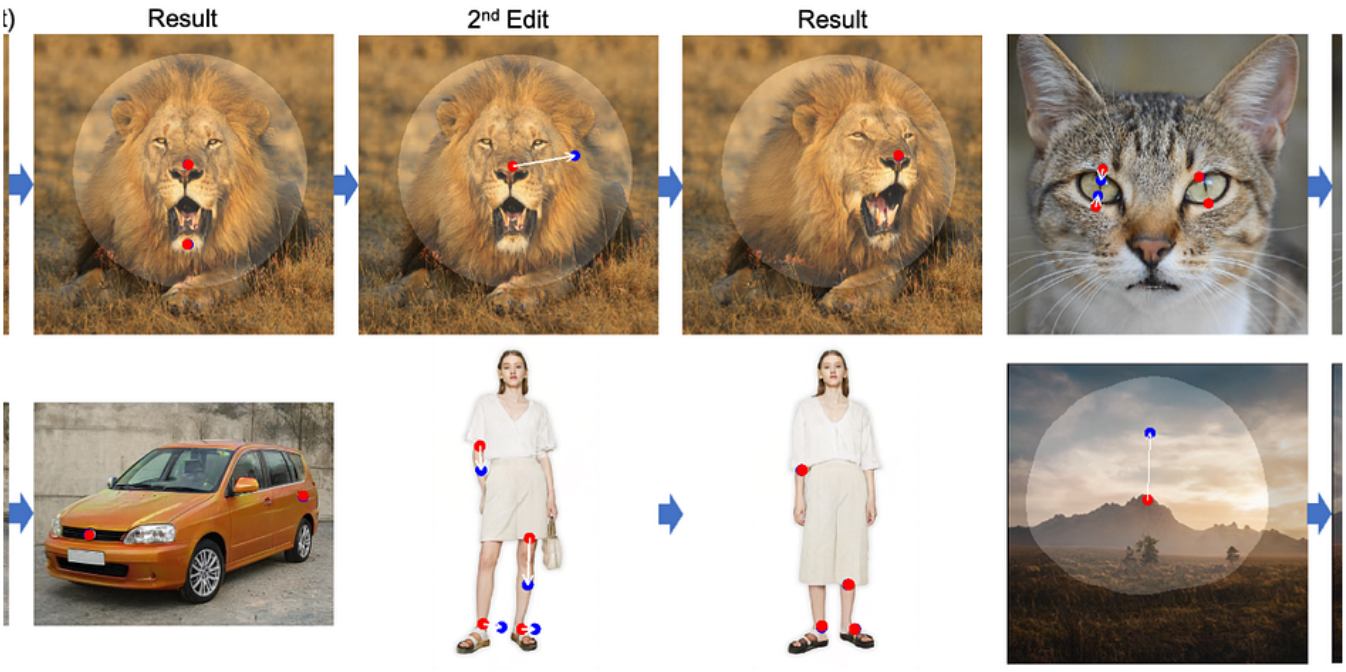


Abish Pius in Writing in the World of Artificial Intelligence

Advanced AI—Reinforcement Q-Learning & Nim

Introduction

9 min read · Jun 10



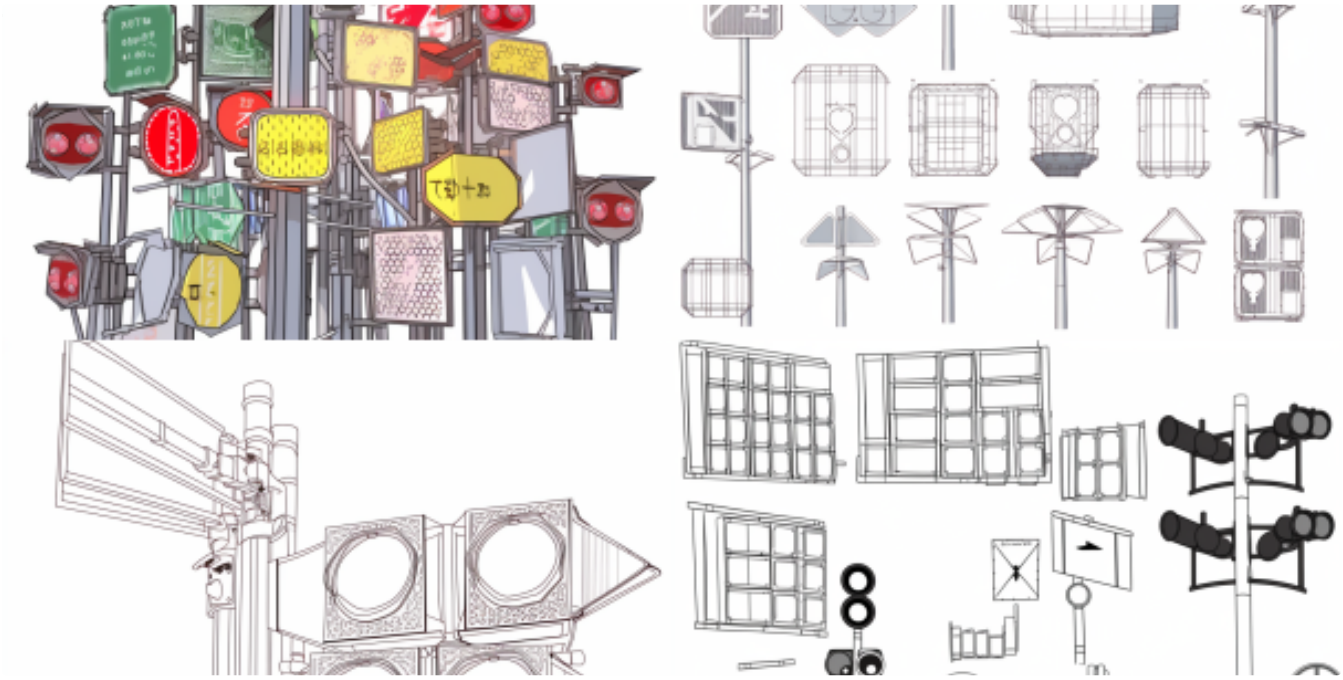
Abish Pius in Writing in the World of Artificial Intelligence

DragGAN: The New Future Gen AI Method to Photoshop … AND you can TRY IT HERE FOR FREE!

Introduction:

3 min read · Jul 2

50



Abish Pius in Writing in the World of Artificial Intelligence

Building a Traffic Sign Recognition Model using TensorFlow

Traffic sign recognition is an essential task for autonomous vehicles, advanced driver-assistance systems, and traffic management. In this…

5 min read · Jun 17

See all from Abish Pius

See all from Writing in the World of Artificial Intelligence

Recommended from Medium

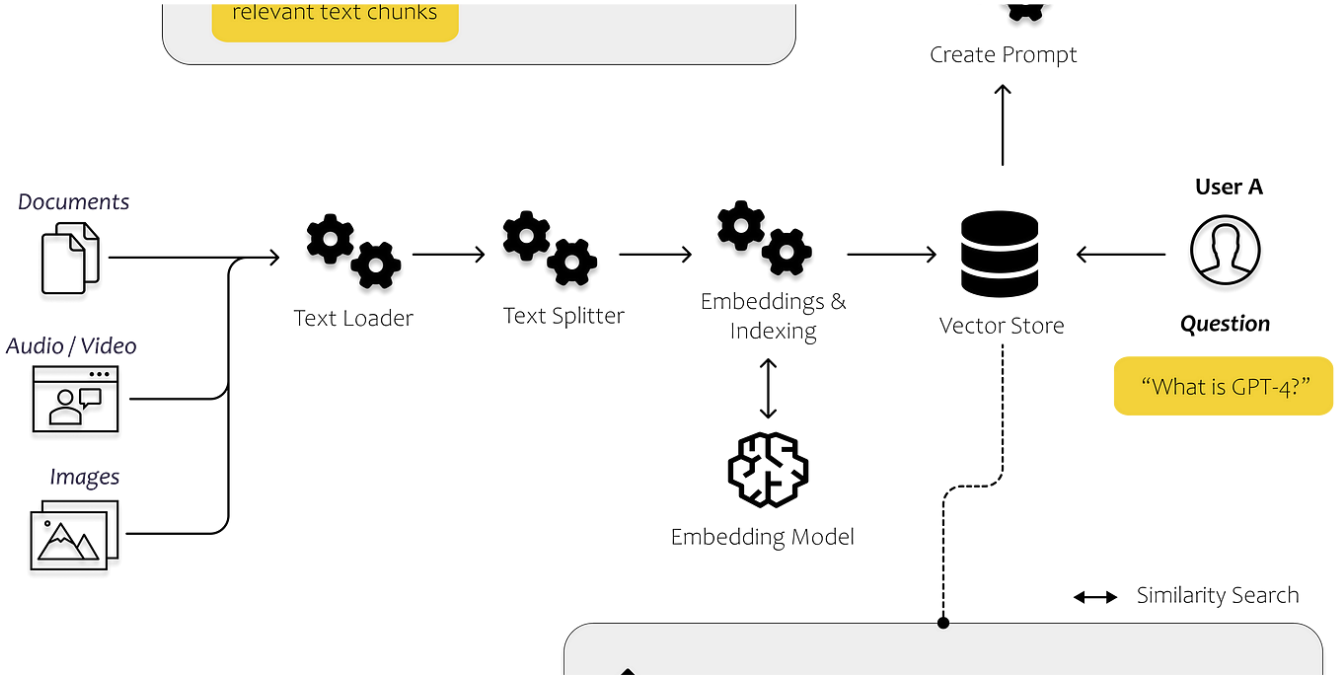
Marco Peixeiro in Towards Data Science

PatchTST: A Breakthrough in Time Series Forecasting

From theory to practice, understand the PatchTST algorithm and apply it in Python alongside N-BEATS and N-HiTS

✨ · 10 min read · Jun 20

725 9



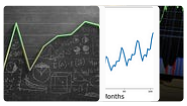
Dominik Polzer in Towards Data Science

All You Need to Know to Build Your First LLM App

A step-by-step tutorial to document loaders, embeddings, vector stores and prompt templates

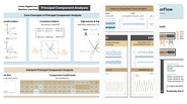
· 25 min read · Jun 22

Lists



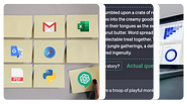
Predictive Modeling w/ Python

18 stories · 88 saves



Practical Guides to Machine Learning

10 stories · 100 saves

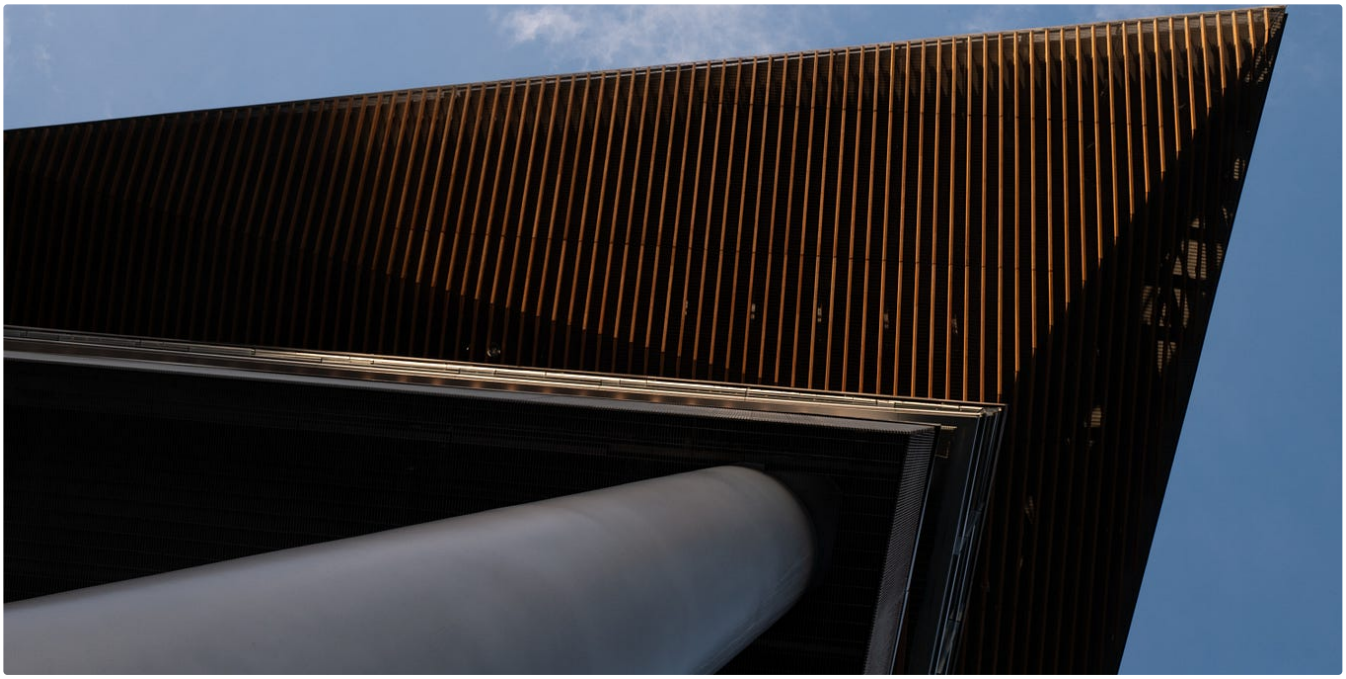


Natural Language Processing

379 stories · 36 saves

The New Chatbots: ChatGPT, Bard, and Beyond

13 stories · 35 saves

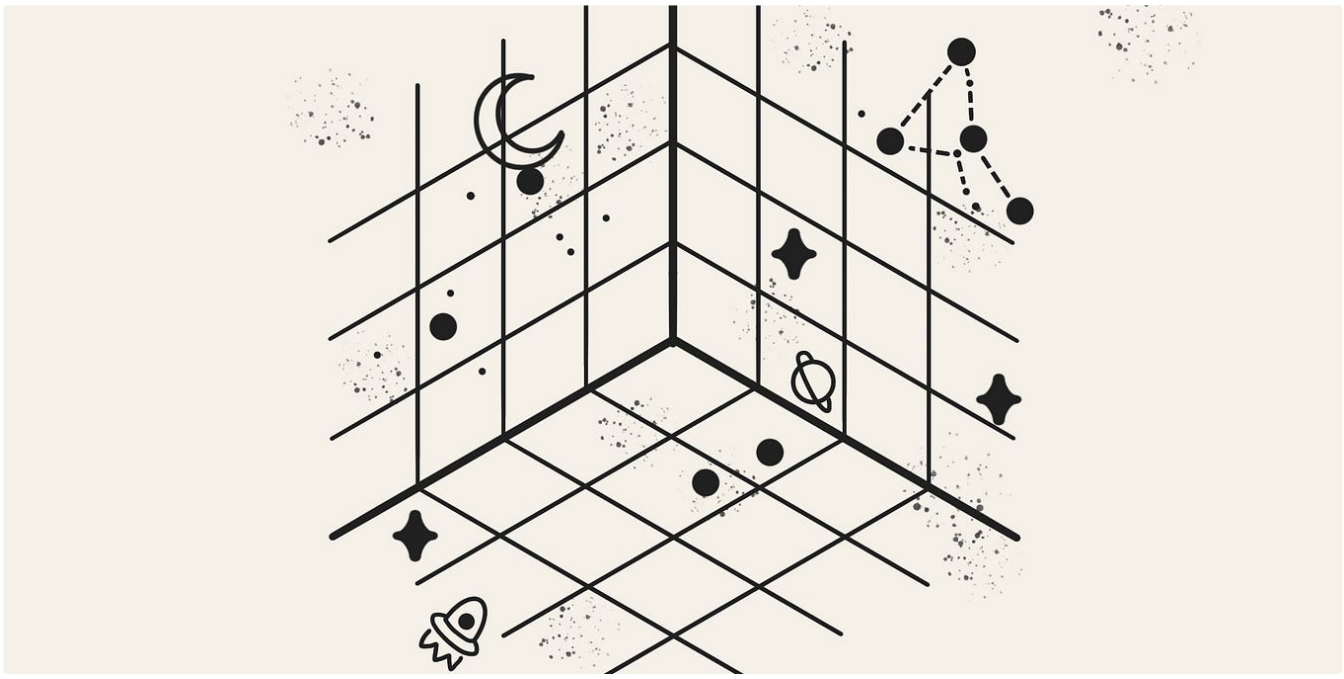


Carlos Costa, Ph.D. in Towards Data Science

Developing Scientific Software

Part 2: Practical Aspects with Python

· 14 min read · Jul 1



Leonie Monigatti in Towards Data Science

Explaining Vector Databases in 3 Levels of Difficulty

From noob to expert: Demystifying vector databases across different backgrounds

· 8 min read · 5 days ago

685 7

John Adeojo in Towards Data Science

How to Avoid Being Fooled by Model Accuracy

A visual guide to binary classification model metrics and their proper usage

· 8 min read · 3 days ago

65 1

Bex T. in Towards AI

Bentoml vs. Fastapi: The Best ML Model Deployment Framework and Why It's Bentoml

Detailed comparison between BentoML and FastAPI for model deployment.

· 9 min read · Jan 19