Semantic Anomaly Detection with Vector Search

Presented to you by Mohamed Arbi Nsibi

Table of content

Mohamed Arbi Nsibi

- ML engineer
- Qdrant Star ⭐
- Former GDSC Lead 23/24

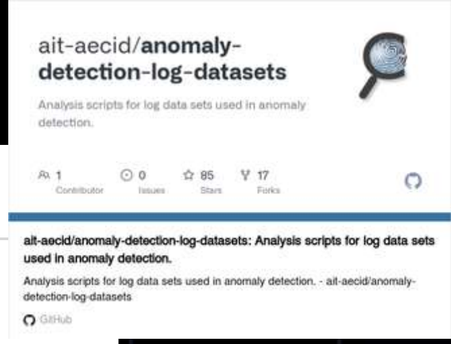
The Problem: The "Regex" Wall

- Traditional monitoring: use static rules (like finding the word ‘Error’)
- Flaw : Modern attacks use legitimate commands in illegitimate order, so that those rules can’t keep up with the volume or the nuance of the system

The solution : Logs as Geometry

- **From Text to Tokens:** Raw logs are turned into Template IDs (e.g., 5 = Receiving, 22 = Verified).
- **The Vector Shift**: using Embeddings to turn a sequence of IDs (like 5 5 22 11) into a high-dimensional vector.
- **The Concept:** "Normal" system behavior creates a "cloud" of points in space. Anomalies are points that land in the "empty" space far away.

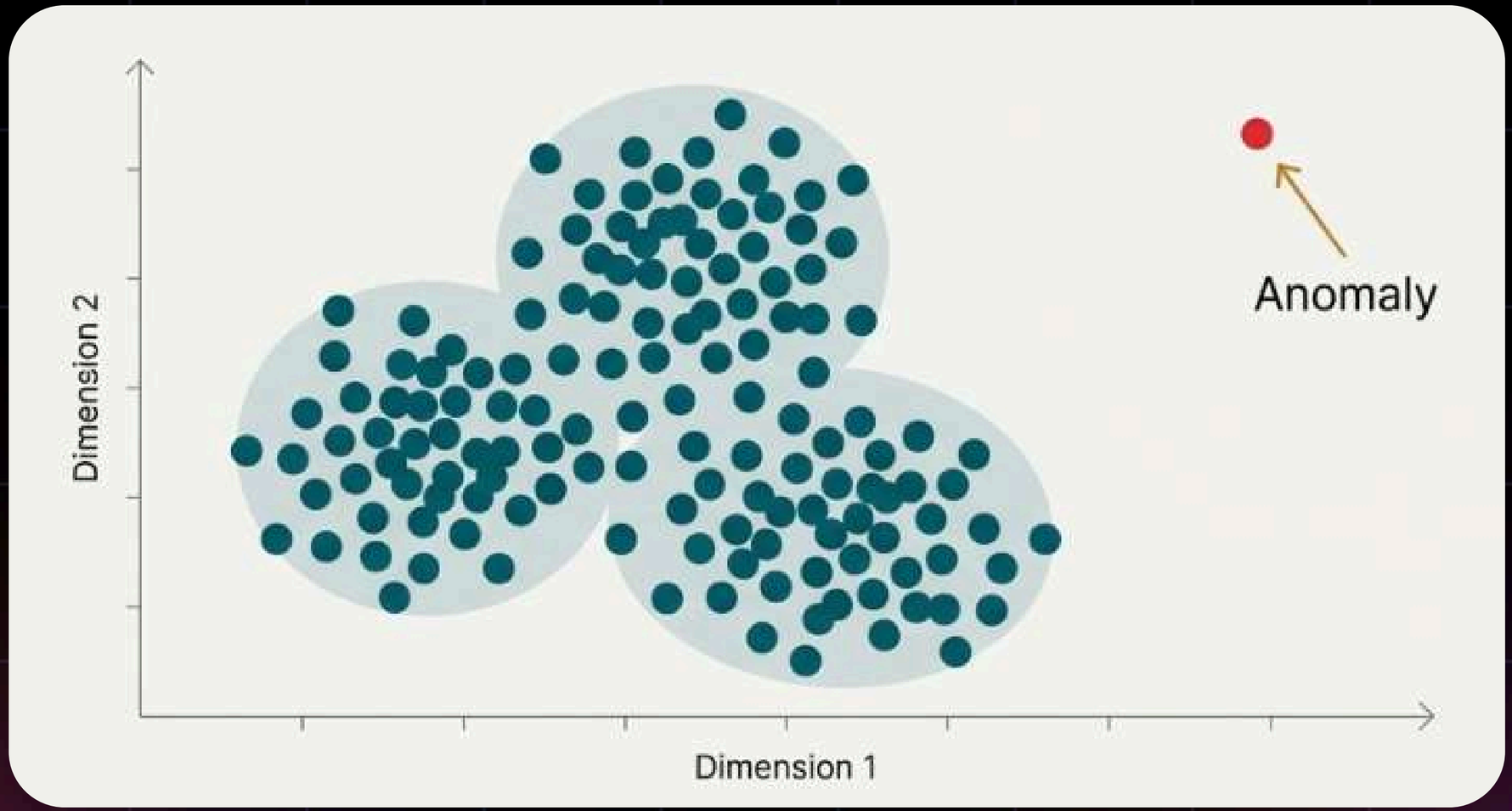
ID	Action Type	Log Message Template	Notes
5	Block Reception	`Receiving block [blk_ID] src: [IP] dest: [IP]`	Incoming block transfer
22	Verification	`Verification succeeded for [blk_ID]`	Block integrity confirmed
11	Termination	`PacketResponder [ID] for block [blk_ID] terminating`	End of packet responder lifecycle
9	Block Transfer	`Received block [blk_ID] of size [size] from [IP]`	Data payload received
3	Heartbeat	Heartbeats / Status Updates	High frequency system signals
26	Network Update	`Adding an internally managed node [IP] to the network`	Cluster topology change
23 / 21	Block Termination	Closing block or PacketResponder terminating	Block lifecycle completed



Finding the Rebels in Your Data

- The Goal: Automatically identify unusual data points, outliers that deviate significantly from the norm. These are often the most critical data points to investigate.

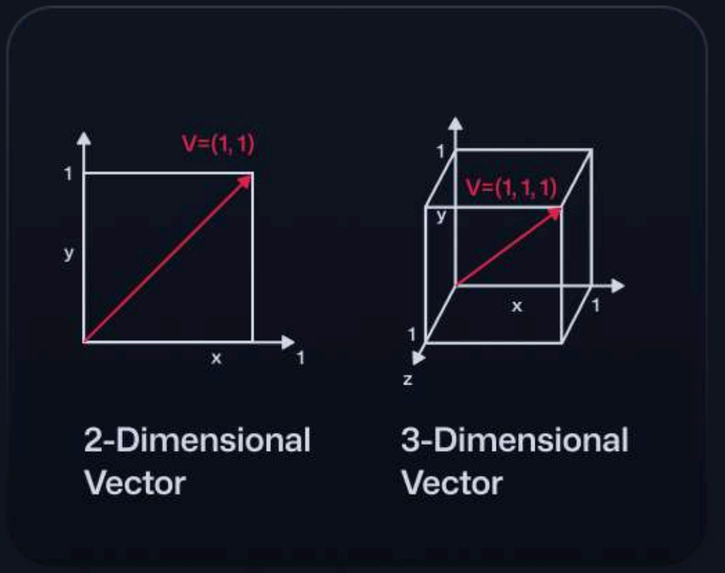
- Cybersecurity: Detecting fraudulent transactions or network intrusions.
- Finance: Identifying irregular trading patterns.
- User Behavior: Spotting bots or users with broken experiences.

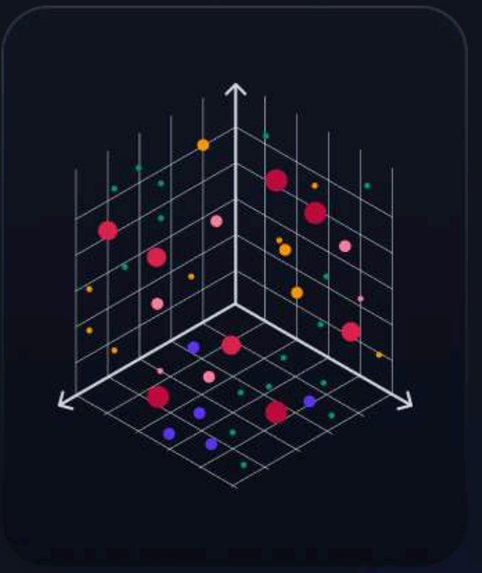

Dimension 2
Dimension 1
Anomaly

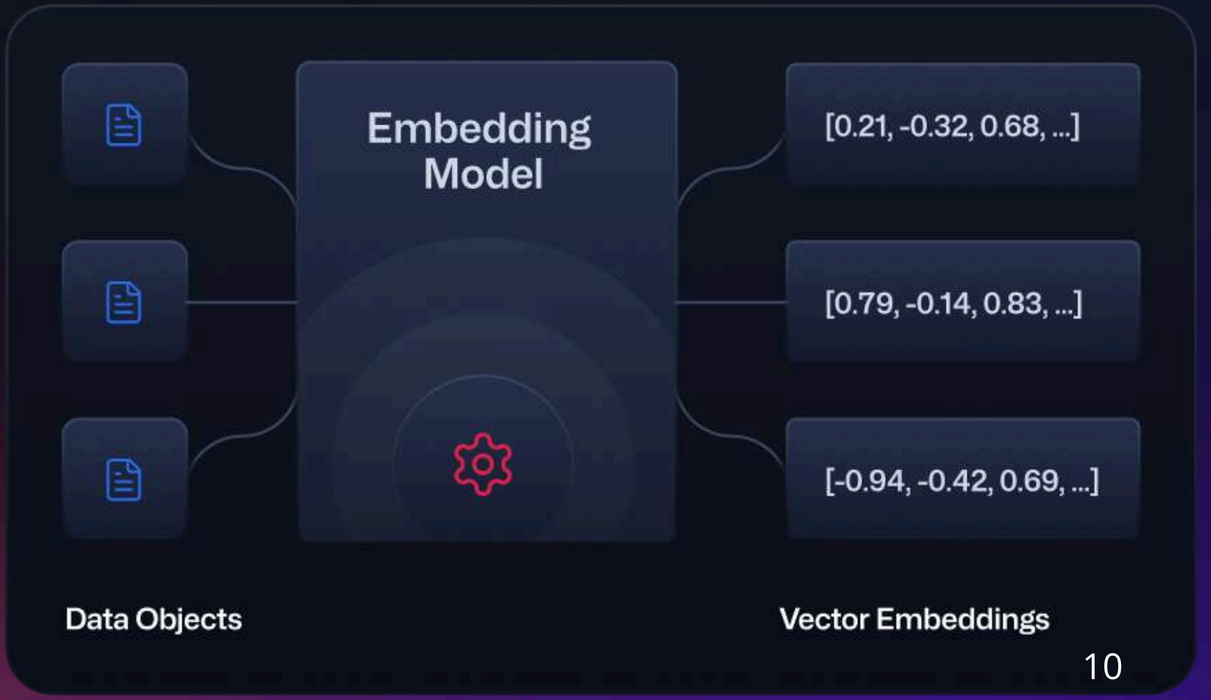
Vector Search Basics

Vector Search Basics

Two different vector embeddings should be close to each other if they represent a similar input object.
Embeddings are generated by neural networks and can represent thousands of dimensions.







Vector Search Basics

Although word counting produces embeddings, dense embeddings are needed to capture semantics

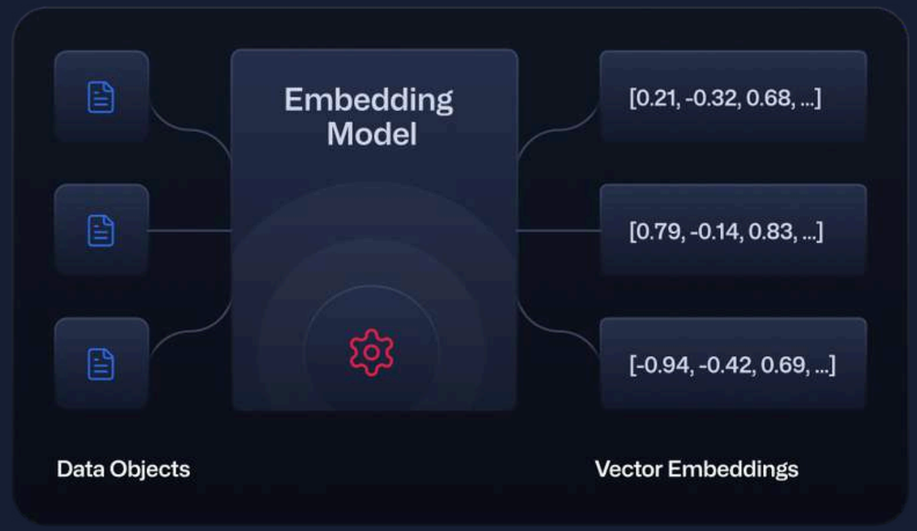
Sparse embedding:
e.g. One Hot Encoding

	an	another	embedding	is	this	Query Sim.
“this is an embedding”	[1,	0,	1,	1,	1]	3
“this is another embedding”	[0,	1,	1,	1,	1]	2

Query:

“What is an embedding?”

Dense embedding:
e.g. from BERT



Gemini TwelveLabs NOMIC

Named Vectors: Dual-Vector Approach

Allows us to analyze Behavior (the sequence) and Context (IP/User-Agent) as separate specialized "profiles" for a single event.

blk_-7229676369905620586,5 5 22 5 11 9 11 9 11 9 26 26 26 4 3 4 2 2 2 23 23 23 21 21 21

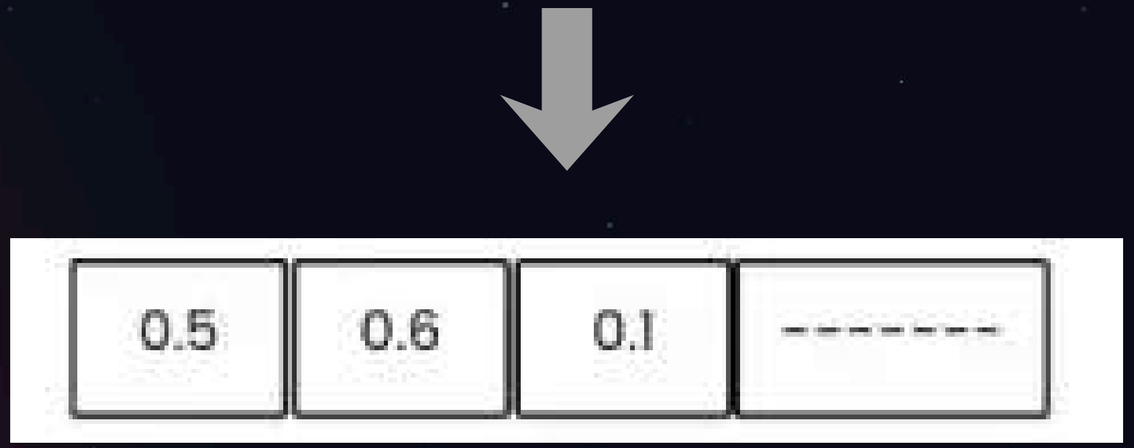
```python
### Configure coll
client.create_collection(
    collection_name="hdfs_logs",
    vectors_config={
        "behavior": models.VectorParams(size=1536, distance=models.Distance.COSINE),
        "context": models.VectorParams(size=1536, distance=models.Distance.COSINE)
    }
)

### store
client.upsert(
    collection_name="hdfs_logs",
    points=[
        models.PointStruct(
            id=1,
            vector={
                # Vector from embedding JUST the numbers "5 5 22..."
                "behavior": [0.11, 0.22, ...],
                # Vector from embedding JUST the ID "blk_1568..."
                "context": [0.01, -0.05, ...]
            },
            payload={
                "sequence": "blk_1568..., 5 5 22...",
                "label": "normal"
            }
        )
    ]
)
```

Single-vector approach

- behavioral patterns and contextual information in a single 1536-dimensional representation

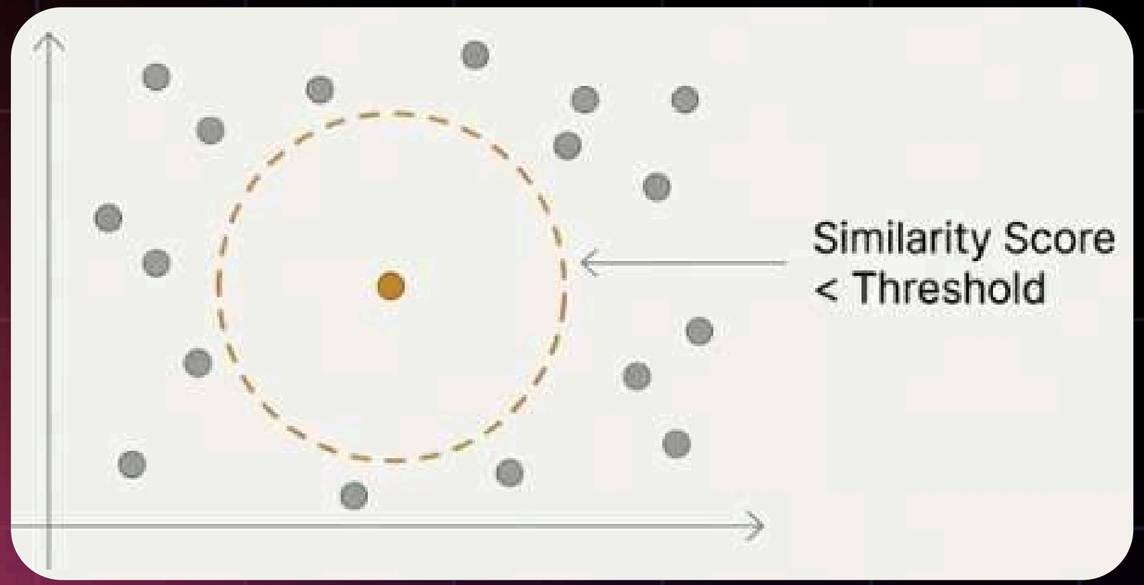
blk_-7229676369905620586 , 5 5 22 5 11 9 11 9 11 9 26 26 26 4 3 4 2 2 2 23 23 23 21 21 21



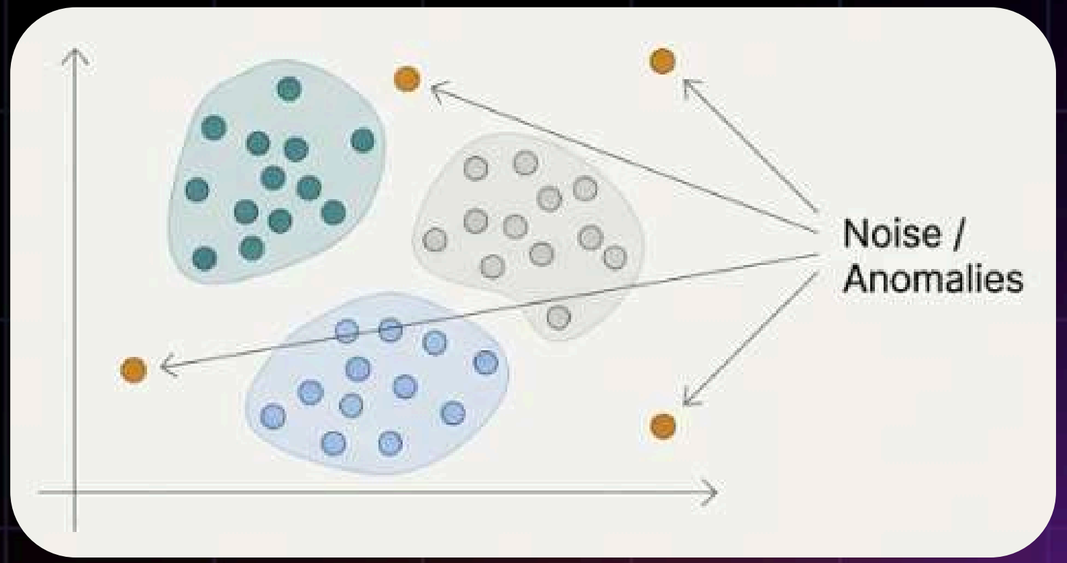
Detection

- **Distance-Based Detection:**

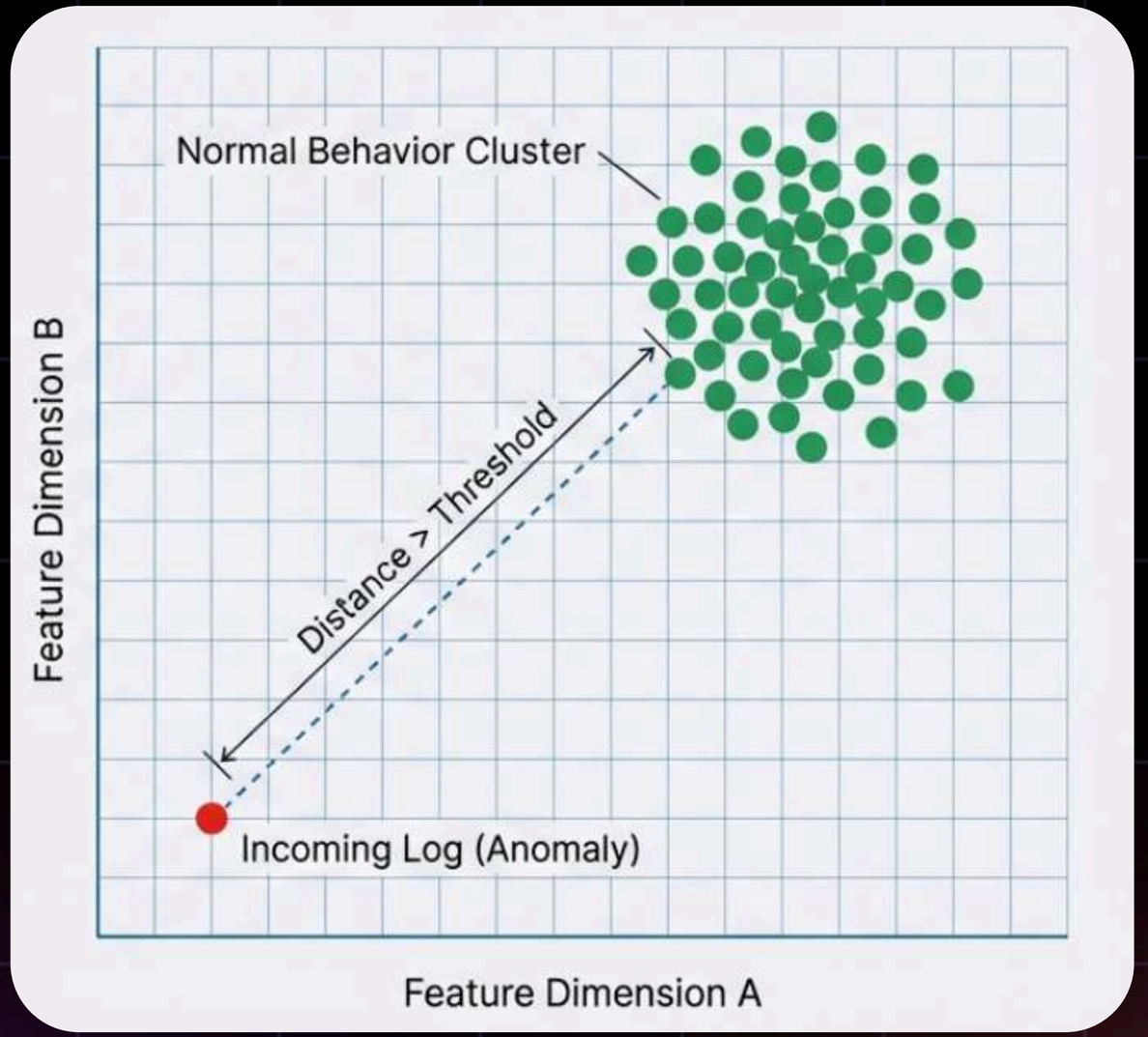
Identifies anomalous points that have no close neighbors within a predefined threshold.



- **Clustering-Based Detection (DBSCAN): run density based clustering algorithm :**Points that do not belong to any identified cluster are labeled as "noise" or anomalies.



The detection logic :: NN & thresholding



The clustering logic :: DBSCAN

DBSCAN automatically identifies dense clusters of normal behavior while flagging isolated points as anomalies. Unlike K-means, it doesn't require pre-defining cluster numbers and naturally detects outliers.

input

- eps=0.08 (search neighborhood radius)
- min_samples (minPts) =10 (minimum cluster size)
- metric='cosine' (for high-dimensional text embeddings)

The clustering logic :: DBSCAN

output

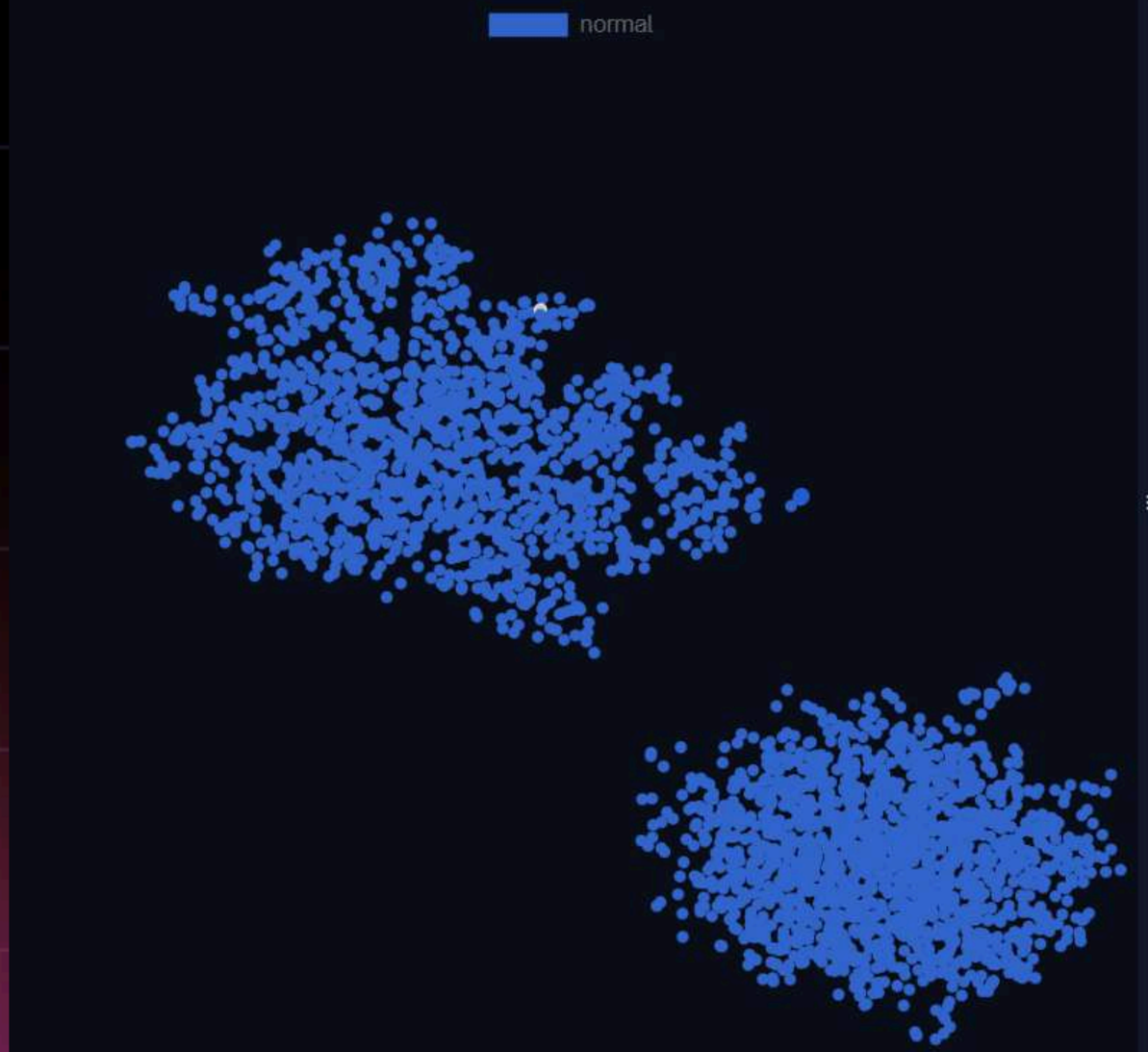
- Core Points: "Popular Kids." They have at least minPts within their epsilon radius. [standard, high-volume logs]
- Border Points: The "Tag-alongs." They are within epsilon distance of a Core Point but don't have enough neighbors of their own. [considered "Normal."]
- Noise : These points have no Core Points within their epsilon radius and don't have enough neighbors to form their own group

Principal Component Analysis (PCA)

PCA projects our high-dimensional data onto the most informative 2D plane, preserving 80%+ of variance while making patterns visually interpretable."

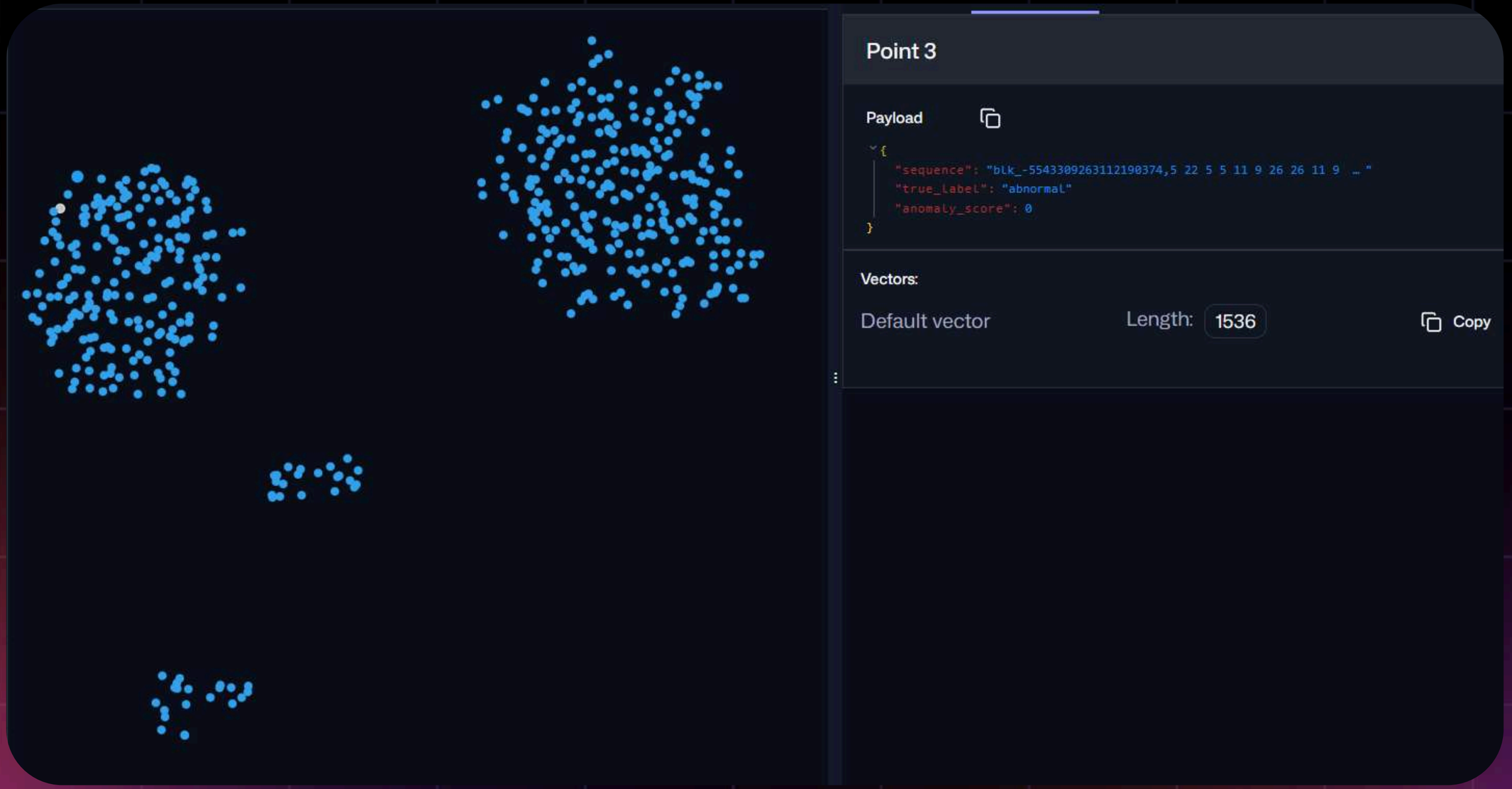
- Algorithm: Linear dimensionality reduction
- Purpose: Reduces 1536D embeddings to 2D for visualization
- Params: n_components=2

log_anomalies



Code | Data Panel

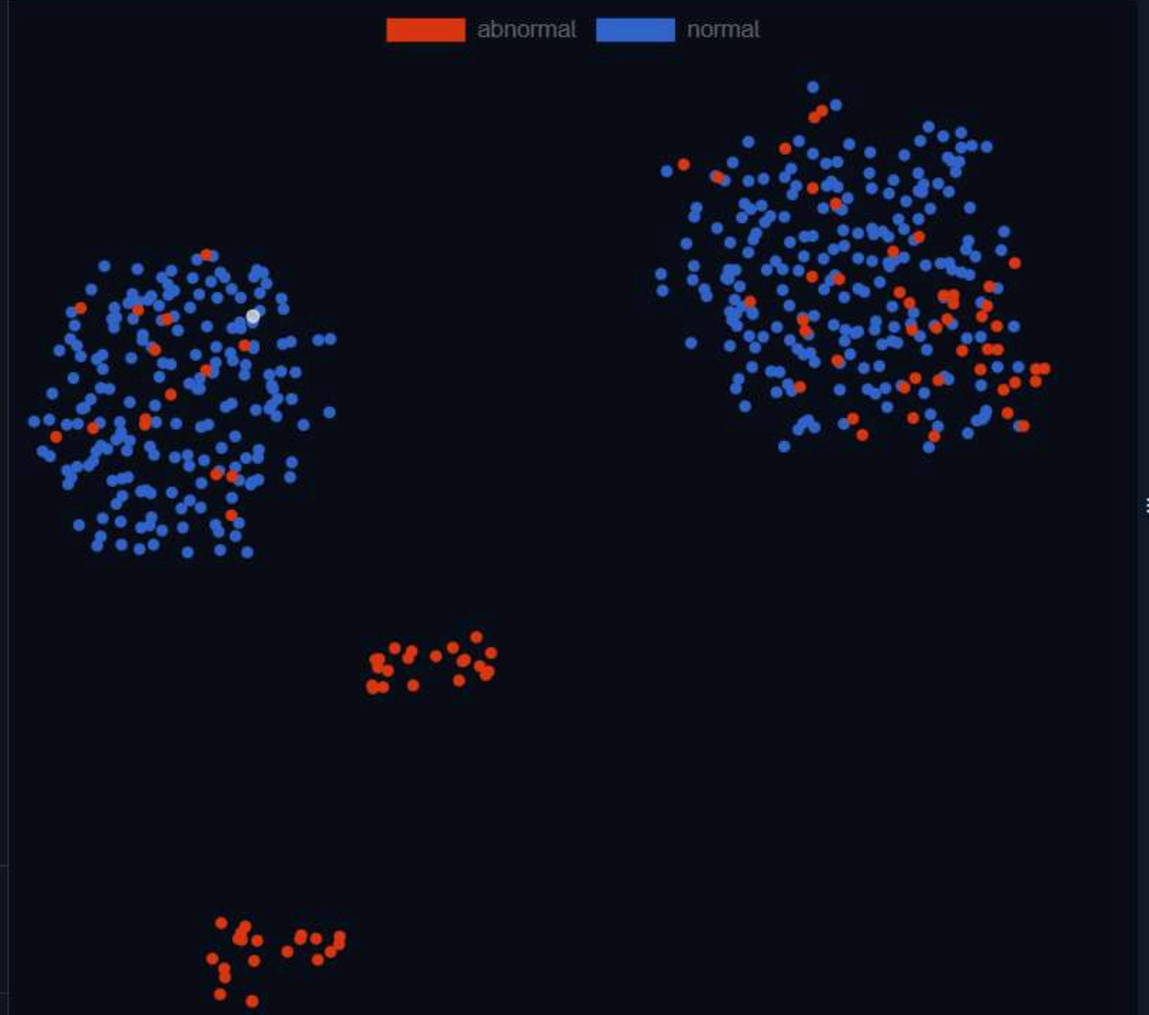
```
1
2
3   // Try me!
4
    RUN
5   {
6     "limit": 3000,
7     "color_by": "label"
8   }
9
10  // Specify request parameters to select data
11  //
12  // Available parameters:
13  //
14  // - 'limit': maximum number of vectors to v
15  //            *Warning*: large values may ca
16  //
17  // - 'filter': filter expression to select v
18  //             See https://qdrant.tech/docum
19  //
20  // - 'color_by': specify score or payload fi
    points.
21  //               How to use:
22  //
23  //                "color_by": {
24  //                  "payload": "field_name"
25  //                }
26  //
27  // - 'using': specify which vector to use fo
28  //                  if there are multiple.
29  //
30  // - 'algorithm': specify algorithm to use f
    options: 'TSNE', 'UMAP', 'PCA'.
31
32
```


Point 3
Payload
{
"sequence": "blk_-5543309263112190374,5 22 5 5 11 9 26 26 11 9 … "
"true_label": "abnormal"
"anomaly_score": 0
}
Vectors:
Default vector
Length:
1536
Copy

cluster / TheGreatCluster

Upgrade Cluster

balanced_log_sequences



Code | Data Panel

Point 443

Payload


```
{
  "sequence": "blk_-3574896762231209882,5 22 5 5 9 11 11 9 11 9 2 …"
  "true_label": "normal"
  "anomaly_score": 0
}
```


Vectors:

Default vector Length: 1536 Copy Ope

What is  Qdrant?

Qdrant is ...

a search engine

Qdrant

Qdrant is ...

engineered for vectors

Qdrant

Qdrant is ...

Fully Open-source
Self-hosting : run on your own infra
Super fast: Latency ~0.024s (~24ms)
Hybrid Search
UI support
Free Tier ~1M(vectors) 768-dim

setup and ingestion

```python
from qdrant_client import QdrantClient, models

# Initialize client in-memory for testing
client = QdrantClient(":memory:")

# Create collection
client.create_collection(
    collection_name="logs_and_behavior",
    vectors_config=models.VectorParams(
        size=128,
        distance=models.Distance.COSINE
    )
)

# Ingest Data using PointStruct
client.upload_points(
    collection_name="logs_and_behavior",
    points=[
        models.PointStruct(
            id=1,
            vector=[0.1, 0.9, 0.2, ...], # 128-dim vector
            payload={"type": "click_stream"}
        )
    ]
)
```

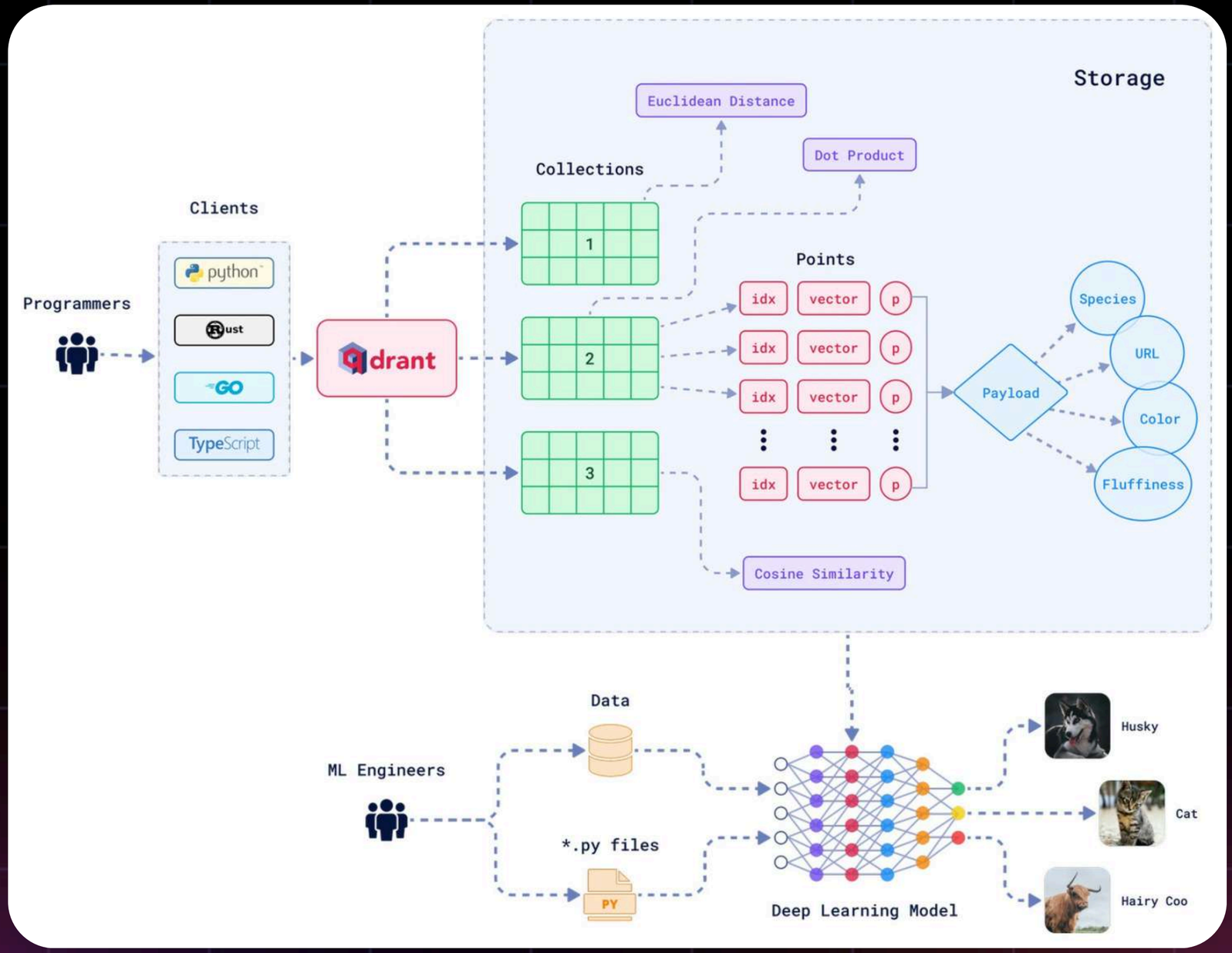
The search query

```python
# Define the threshold for "normality"
SIMILARITY_THRESHOLD = 0.5

# Perform the search
results = client.search(
    collection_name="logs_and_behavior",
    query_vector=[0.1, 0.8, 0.3, ...], # New incoming log
    limit=1
)

# The Logic: If the best match is too far away, it's an outlier
if not results or results[0].score < SIMILARITY_THRESHOLD:
    print(f"Anomaly Detected! Nearest neighbor score: {results[0].score}")
else:
    print("Log is within normal behavior parameters.")
```

Qdrant


Storage
Euclidean Distance
Dot Product
Collections
Clients
Programmers
python
Rust
GO
TypeScript
qdrant
1
2
3
Points
idx
vector
p
Payload
Species
URL
Color
Fluffiness
Cosine Similarity
Data
ML Engineers
*.py files
PY
Deep Learning Model
Husky
Cat
Hairy Coo

Optimization & Best Practices

- Data Normalization: Essential for Euclidean distance to ensure vector magnitude (e.g., 1000 clicks vs. 10 clicks) doesn't skew metrics and cause false positives.
- Contextual Filtering: Utilize payloads to ensure "apples-to-apples" comparisons (e.g., comparing a 'login attempt' against another 'login' rather than a 'page scroll'). [use "Payloads" to tell the computer: "Only compare this login attempt to other login attempts".]
- Metric Selection: Your distance metric (Cosine vs. Euclidean)[direction vs exact distance] must align with how the vectors were originally trained and generated.

How Qdrant Achieves Search

Core Capabilities

Vector Search
Scalable similarity and discovery search (billions of vectors)

Hybrid Search
Combine dense + sparse embeddings, filters, and metadata

Filtering
Numeric, categorical, geo, temporal filters out-of-the-box

Distributed & Resilient
Replication, sharding, multi-tenancy

Advanced Features

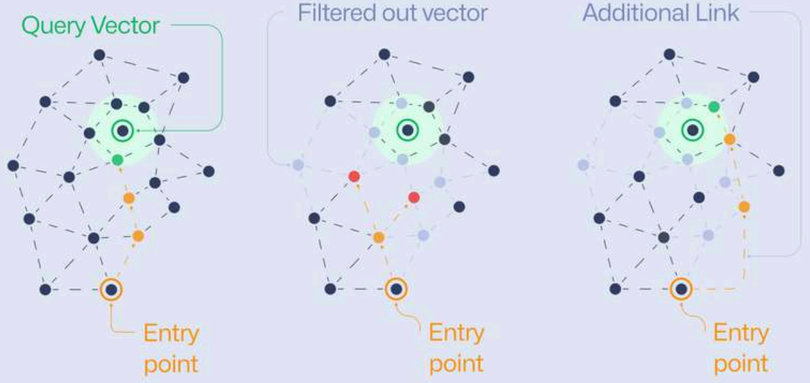
Re-ranking
Maximum Marginal Relevance (MMR), score boosting

Quantization
Binary, scalar & product; lower cost without major recall loss

Multi-vectors: Late interaction for retrieval models (e.g. ColBERT)

Performance Optimizations
HNSW tuning, payload indexing, prefetching

Filterable HNSW



Qdrant Innovations to make development easier

FastEmbed

Generate high-quality embeddings fast. A small Python library for embedding generation, built in and integrated with Qdrant.

- Works out of the box in Qdrant.
- Few dependencies: runs on CPU; skips multi-GB PyTorch downloads.
- Made for speed: uses ONNX Runtime and data parallelism.

Key features

- Use Qdrant models (miniCOIL, BM42).
- Support for late-interaction (ColPali, ColBERT) and sparse-neural methods (SPLADE, BM42, miniCOIL), MUVERA embeddings and more.
- Run inference and upsert/search in one call.

Import:

```
from qdrant_client.models import
Document, Image
```

MCP Servers

Build custom retrieval-based AI apps fast. Start from these servers and add tools/commands for your data and workflows.

- mcp-server-qdrant: official MCP server for storing and retrieving data in Qdrant.
- mcp-for-docs: open-source API reference for AI coding assistants using semantic code retrieval.

Key features

- Automate codebase documentation.
- Personalize your coding assistant to your project's conventions and rules.
- `Do inline RAG.`
- Speaks stdio, sse, and streamable-http protocols.

Run:

```
docker run  mcp-server-qdrant
```

Qdrant Edge

Bring vector search to the edge: an embeddable, high-performance engine that runs directly on mobile and other edge devices.

- Run on low-CPU devices.
- Use one API to manage and synchronize data on-device and in your cloud cluster.
- Fit common on-device cases: phones and laptops, smart-home/IoT, robotics.

Key features

- Use local storage to avoid network latency.
- Support multi-tenant setups; treat each device as its own tenant.
- Embed as a library; runs in-process with no background daemons.

Use:

```
client =
QdrantClient(path="qdrant_edge.db")
```

Vector Search in Production

- Written in **Rust** and offers **great performance**
- Allows to interact by **HTTP** or **gRPC protocols**.
- Runs both in **single and multiple node** setup.
- Incorporates **category, geo-coordinates** and **full-text filters**
- Supports **hybrid**, **multimodal**, **multivector** and **multi-staged** search
- Official **Python**, **Javascript/Typescript**, **Rust** and **Go SDKs**.
- Makes vector search **affordable**.

For Managed Cloud solutions, check out Cloud Embeddings Inference.

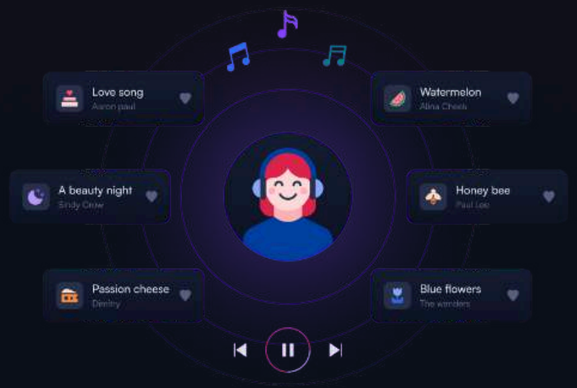
Vector Search

An essential part of the AI Transformation

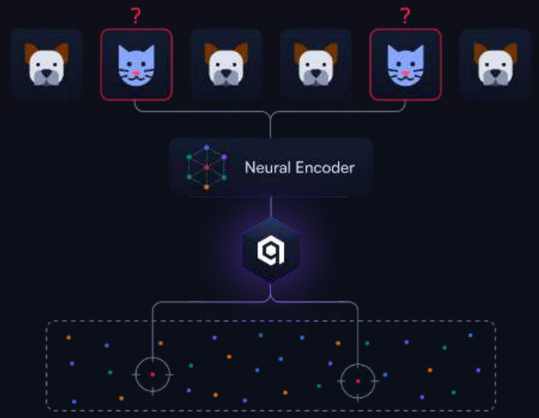
HR-Tech | Ad-Tech | Online Dating | Gig Economy | E-Commerce | Legal-Tech | Fashion | Med-Tech

Ed-Tech | Media & News | Agriculture | Manufacturing | Streaming Services | Marketplaces | Anti-fraud | Biometrics

Search Systems



Recommendations



Anomaly Detection

RAG / Information Assistants

60K

Community Members

250M+

OSS Downloads

26K+

Github Stars

>140

Contributors

Getting Started with Qdrant

Qdrant Open Source
Usually deployed with Docker containers. Lightweight, offers all the functionalities of Qdrant.

Qdrant Managed Cloud
Run on one of the three major cloud providers: AWS, Azure, or GCP. Provides a management UI and API. For US regions, we offer Cloud Inference that processes raw data into vectors.

Qdrant Hybrid Cloud
All the benefits of cloud deployment, but keeping the data on your premises. Requires a Kubernetes cluster and might be managed from Qdrant Cloud UI, but no data leaves your environment.

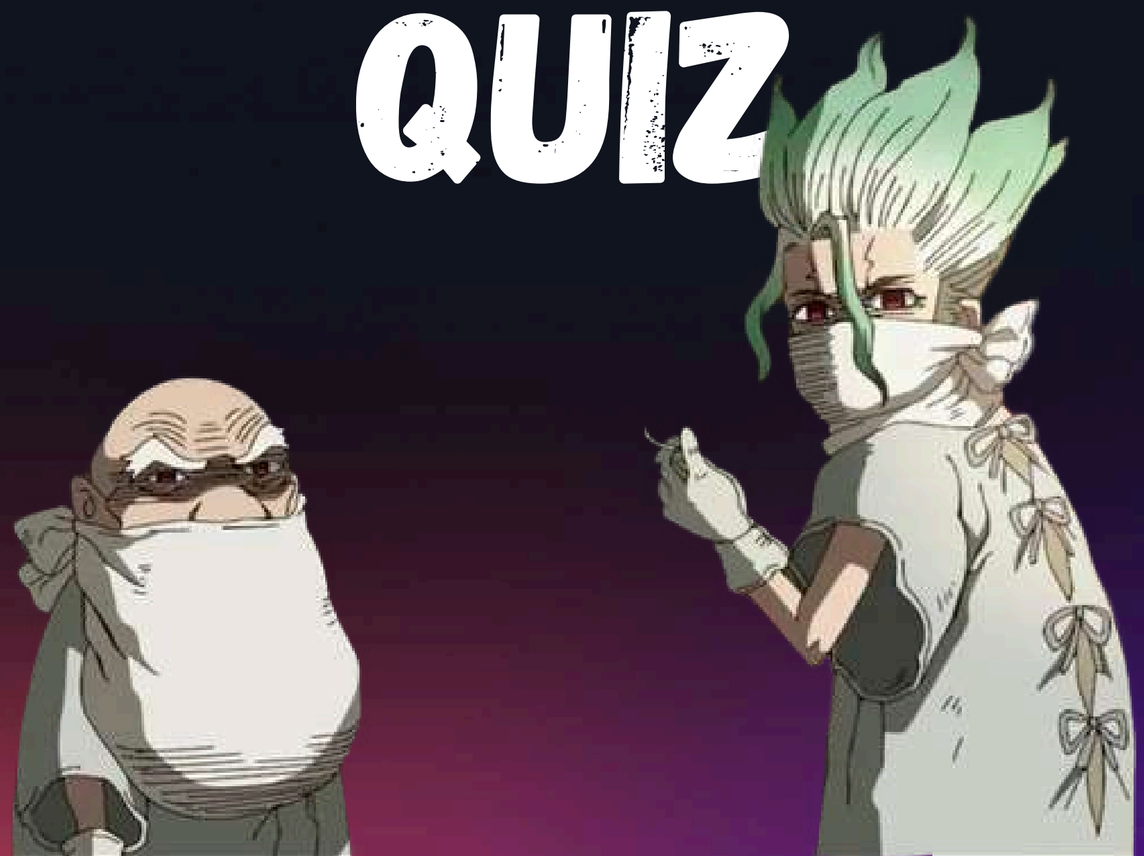
Qdrant Private Cloud
A dedicated, on-premise solution that guarantees supreme data privacy and sovereignty.

Python SDK Local Mode
Suitable mostly for quick experiments, but not intended to be running in production.

Ecosystem

Gemini | n8n | MISTRAL AI_
aws | UNSTRUCTURED | A
Google Cloud Platform | OpenAI | jina
Haystack by deepset | crewai | LlamaIndex
LangChain | cohere | *and more...*

QUIZ



THANK YOU FOR YOUR ATTENTION!!

Where to find me?

Goodnight

Linkedin Profile

MedArbiNsibi

GODS 5.0
AS NEVER BEFORE