VAAS-X SDK · SPORTS & HUMAN PERFORMANCE

Athlete Monitoring Without the Per-Sport Integration Project

How Bootstrap gives wearable sensor streams from any sport the same zero-configuration operational memory already validated on industrial telemetry — now proven on real human-movement IMU data across 8 independent subjects.

8/8	45	154×	0
SUBJECTS, CONSISTENT RESULT	IMU CHANNELS, ZERO CONFIG	VARIANCE, SPORT VS. SEDENTARY	GPU REQUIRED

VAAS-X Ltd (SC873624) · July 2026 · vaasx.com

The problem: every sport, every device, its own integration project

A wearable IMU on an athlete produces the same kind of multi-channel telemetry stream as an industrial sensor array — accelerometer, gyroscope, and magnetometer readings across several body-worn units, sampled tens of times a second. But today, turning that stream into something a coach, physio, or performance analyst can actually query means building a bespoke feature pipeline per sport, per device placement, and often per athlete — because a threshold or model tuned for a sprinter's wearable doesn't transfer to a rower's or a basketball player's without re-engineering.

That's the same integration bottleneck predictive maintenance teams hit with industrial sensors, just wearing different clothes: a real project per equipment type, or a generic model nobody trusts near their specific athletes.

What Bootstrap actually does

Bootstrap is the ingestion layer of the `vaas-x` SDK. Point it at a sensor stream and it profiles each channel statistically, classifies the data's structure, and starts accumulating state-action-outcome episodes — what the system observed, what happened, and what the outcome was — without a schema or a single labelled reading supplied first. The same ingestion path that works on a turbofan's telemetry works unchanged on a wearable's.

```
from vaasx import Bootstrap

brain = Bootstrap(api_key="YOUR_API_KEY", device_id="athlete_042_wearable")

# Any http(s), ws(s), mqtt, local file (.jsonl/.csv/.json), pandas
# DataFrame, or plain Python generator.
brain.connect("mqtt://wearable-hub.local/athlete_042/imu")

hits = brain.query("elevated load pattern matching prior soft-tissue flag", k=5)
for hit in hits:
    print(hit["score"], hit["episode"])

# Tell the system whether a past episode's flagged pattern actually
# preceded something that mattered -- future queries weight toward
# what turned out to be real, not just what looks statistically similar.
brain.outcome(episode_id=hits[0]["id"], success=True)
```

Every stored episode is retrievable by similarity in milliseconds, weighted toward episodes with a recorded outcome by default. The retrieval and outcome-grounding mechanics are identical to the industrial deployment — what changes is only the sensor stream you point it at.

Validated: real wearable IMU data, zero configuration

Unlike the industrial vertical, sports performance has no equivalent of the CMAPSS benchmark most teams would recognise. So before making any claim here, we ran the same blind test discipline against a real public human-movement dataset: the UCI Machine Learning Repository's *Daily and Sports Activities* set (Altun, Barshan & Tunçel, 2010) — 8 subjects, each wearing five Xsens IMU units (torso, both arms, both legs), 9 axes per unit, 45 channels total, sampled at 25Hz. No feature engineering, no per-activity tuning: Bootstrap's profiler and classifier saw raw channel values only, exactly as they'd arrive from a real wearable.

For each of the 8 subjects independently, we ran two single-device streams through Bootstrap unmodified: a sedentary baseline (sitting) and a dynamic sport activity (basketball or rowing), each as its own continuous per-subject session — matching how one wearable on one athlete actually streams in production.

> **RESULT**
>
> Across all 8 subjects, sitting left an average of 14.8 of 45 channels classified stable with 22.4 flagged significant; basketball and rowing collapsed to zero stable channels every time, with 40–44 of 45 flagged significant in every single subject — a consistent, zero-config separation between a sedentary and a dynamic activity on real human-movement data, with no channel labels or activity hints ever supplied.

We cross-checked that result independently of Bootstrap's own code: raw per-channel variance, computed separately with plain statistics libraries, was roughly 154× higher during basketball and 27× higher during rowing than during sitting, averaged across all 8 subjects. Bootstrap's zero-config classification is tracking a real, independently measurable physical signal — not noise.

> **SCOPE, HONESTLY**
>
> What this validates: Bootstrap's zero-config channel profiling and significant/stable classification transfers correctly from industrial telemetry to real human-movement wearable data, with no reconfiguration. What this does **not** validate: injury prediction, movement-quality scoring, or any claim that flagged episodes correspond to injury risk. That would require labelled injury-outcome data specific to a sport and population, which we have not tested. Anyone evaluating Bootstrap for athlete monitoring should treat the ingestion and retrieval layer as proven, and any injury-risk model built on top of it as a separate claim requiring its own validation.

Deployment model

Profiling, classification, and episode accumulation all run locally — on a team's own hardware, at the venue, or in the cloud. Only the resulting episodes cross the network to reach retrieval, so an athlete's raw sensor stream never has to leave the building it was recorded in unless a team

chooses to send it. Fleet-wide queries across every wearable a squad owns run through the same cross-device retrieval used for a multi-machine industrial fleet — one athlete's flagged pattern can be checked for similarity against the whole squad's history without any of them sharing a schema.

Pricing is infrastructure-based, not per-query: a fixed monthly cost per tier regardless of how many athletes are connected or how many queries a coaching staff makes.

Get started

Free tier: instant API key, no credit card — `pip install vaas-x`, connect a source, query your first episode in under ten minutes.

Full guide: vaasx.com/docs/getting-started.html | Enterprise: hello@vaasx.com