Deploying AI on the Plant Floor with FrameworX and ML.NET

Aim high, start simple, scale without limits.

What we'll cover today

Agenda

1. Tatsoft + FrameworX: the platform context
2. Why ML is becoming practical in industrial systems
3. Why ML.NET fits naturally inside a .NET-based platform
4. Demo: anomaly detection with ML.NET → operational data in FrameworX
5. Plus: a quick look at MCP (connecting to external AI services)
6. Q&A

Who will be presenting

Meet our team

Dave Hellyer

VP of Business Development

Eduardo Bogo

Automation Engineer

Isabela Taccolini

Marketing Director

Scott Gray

Senior Solutions Consultant

Tatsoft + FrameworX

Tatsoft

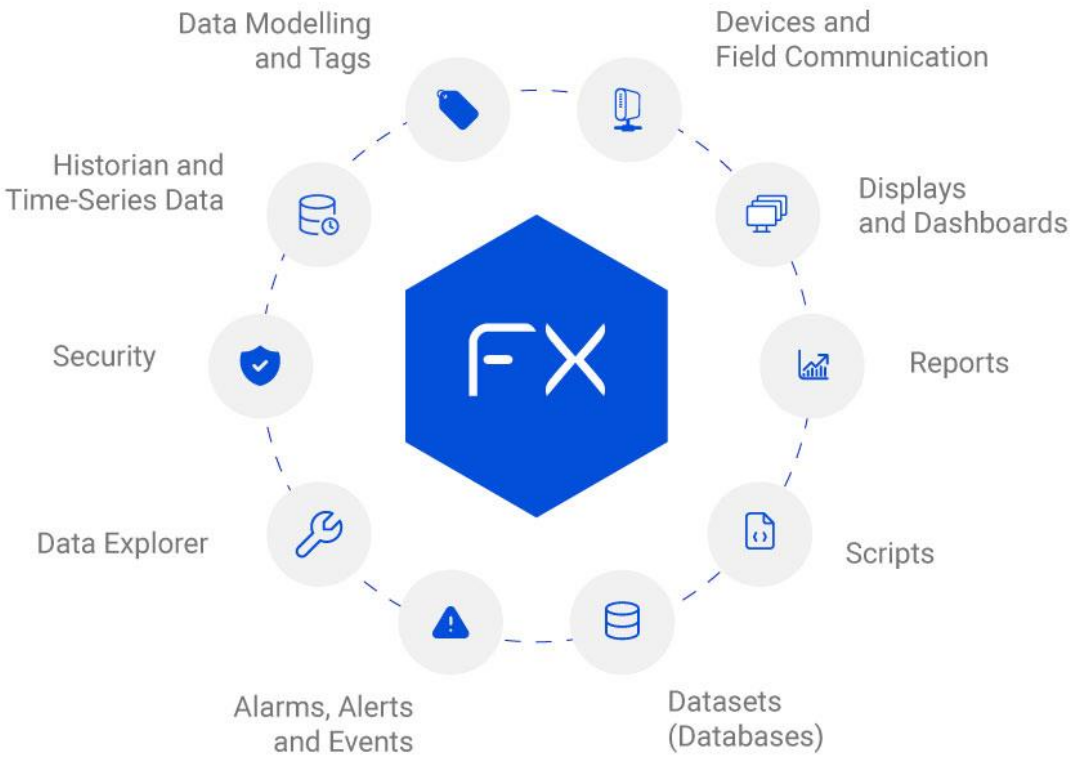
About our company

- 30+ years building industrial real-time software
- Founded by the original creators of InduSoft
- Global operations (US, Brazil, Europe, Asia)
- Deployed across manufacturing, energy, water, F&B, life sciences, transportation, O&G, data centers
- Focus: production industrial infrastructure (not an experimental analytics tool)

FrameworX

Unified SCADA + IIoT platform

- .NET 8 managed architecture + native Python 3 support
- Single unified platform (not a collection of add-ons)
- Alarms, historian, security, scripting, MQTT, connectivity, visualization — in one designer + runtime
- Deploy the same project to edge devices, servers, VMs, or containers
- This unified foundation makes ML practical (results become operational data)



Why .NET matters (in industrial systems)

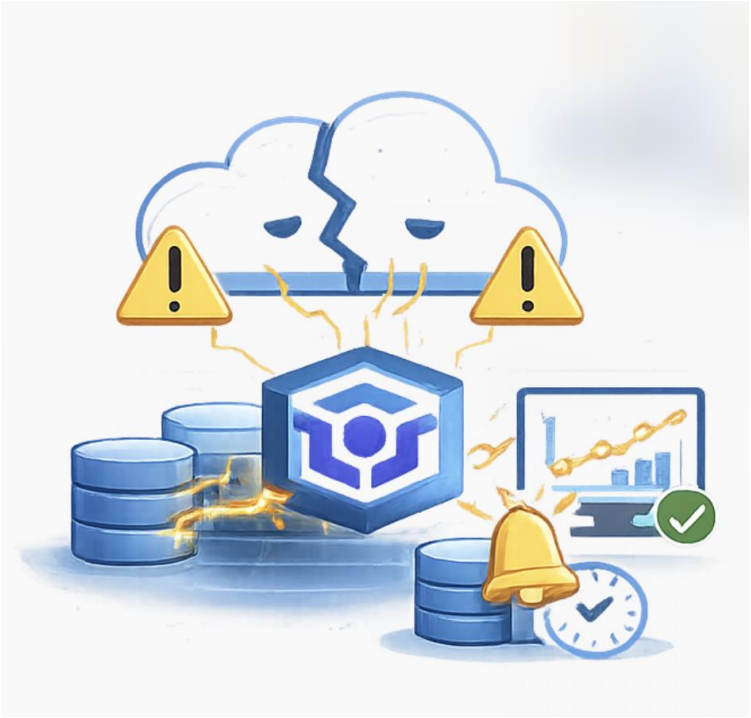
Practical Reasons

- Modern managed runtime: performance, debugging, versioning, long-term support
- One runtime for scripts, calculations, alarms, business logic — and machine learning
- Avoids “bolt-on” execution environments that make production integration fragile

Why ML matters (inside the platform)

Operationalizing ML

- External ML can create gaps: duplicated data, fragile integrations, limited visibility
- When ML runs inside FrameworX, results become first-class operational data
- ML outputs can be alarmed, historized, visualized, and acted on like any other signal

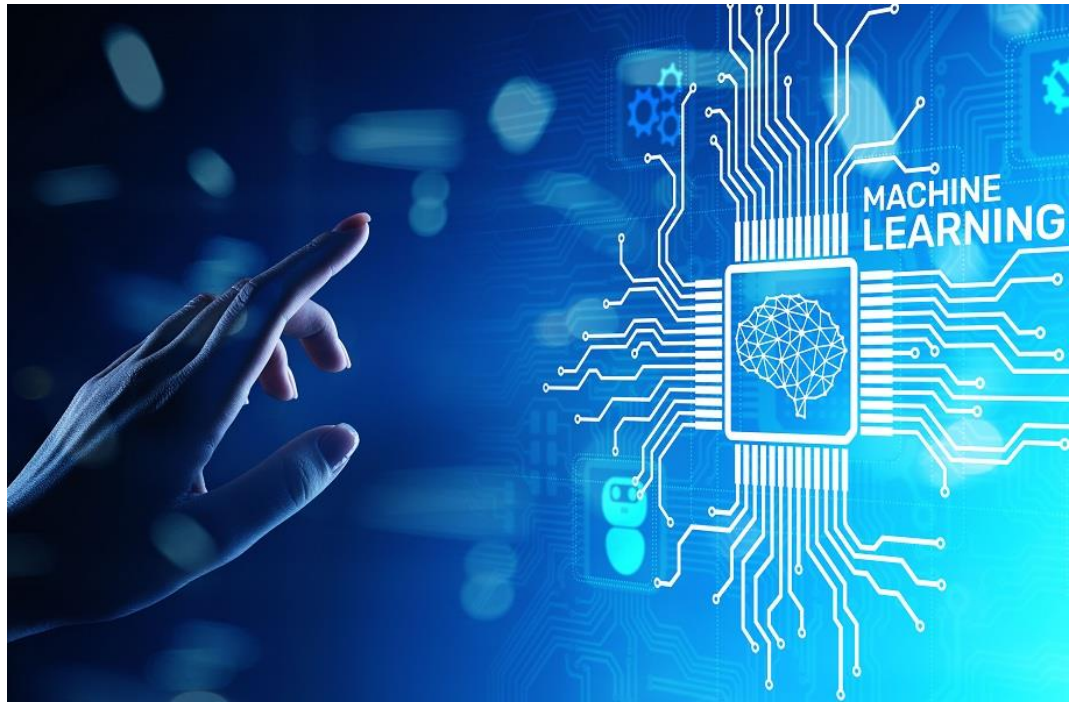


Why Machine Learning in Industrial Applications?

Why use machine learning?

Practical Value

- Optimize processes and improve outcomes
- Retain expertise (capture “tribal knowledge” before it walks out the door)
- Scale insights across the whole plant — not just one machine at a time



Ways ML Can Improve a Process

Practical Value

- Increase production output
- Reduce scrap and rework
- Shorter cycle times
- Reduce unplanned downtime
- Improve product quality
- Energy savings and better scheduling
- Root cause analysis support

Where ML delivers the best ROI

A practical starting shortlist

Predictive maintenance

Detect issues early and avoid downtime

Defect detection

Flag quality problems before they ship

Process optimization

Tune setpoints, reduce scrap, improve throughput

Production forecasting

Plan scheduling and materials with better accuracy

ML Models and Practical Examples

Real-life applications

Binary classification	Fail in next 24 hours?
Multiclass classification	Which defect type?
Regression	Estimate cycle time or energy consumption
Time-series forecasting	Forecast hourly production output
Anomaly detection	Abnormal vibration/temperature patterns
Clustering	Normal vs stressed vs inefficient operating states
Decision trees / random forests	Which variables drive failures?
Gradient-boosted trees	Complex relationships to improve yield

Key Takeaway

One thing to remember

Machine learning complements traditional control logic — it doesn’t replace it.

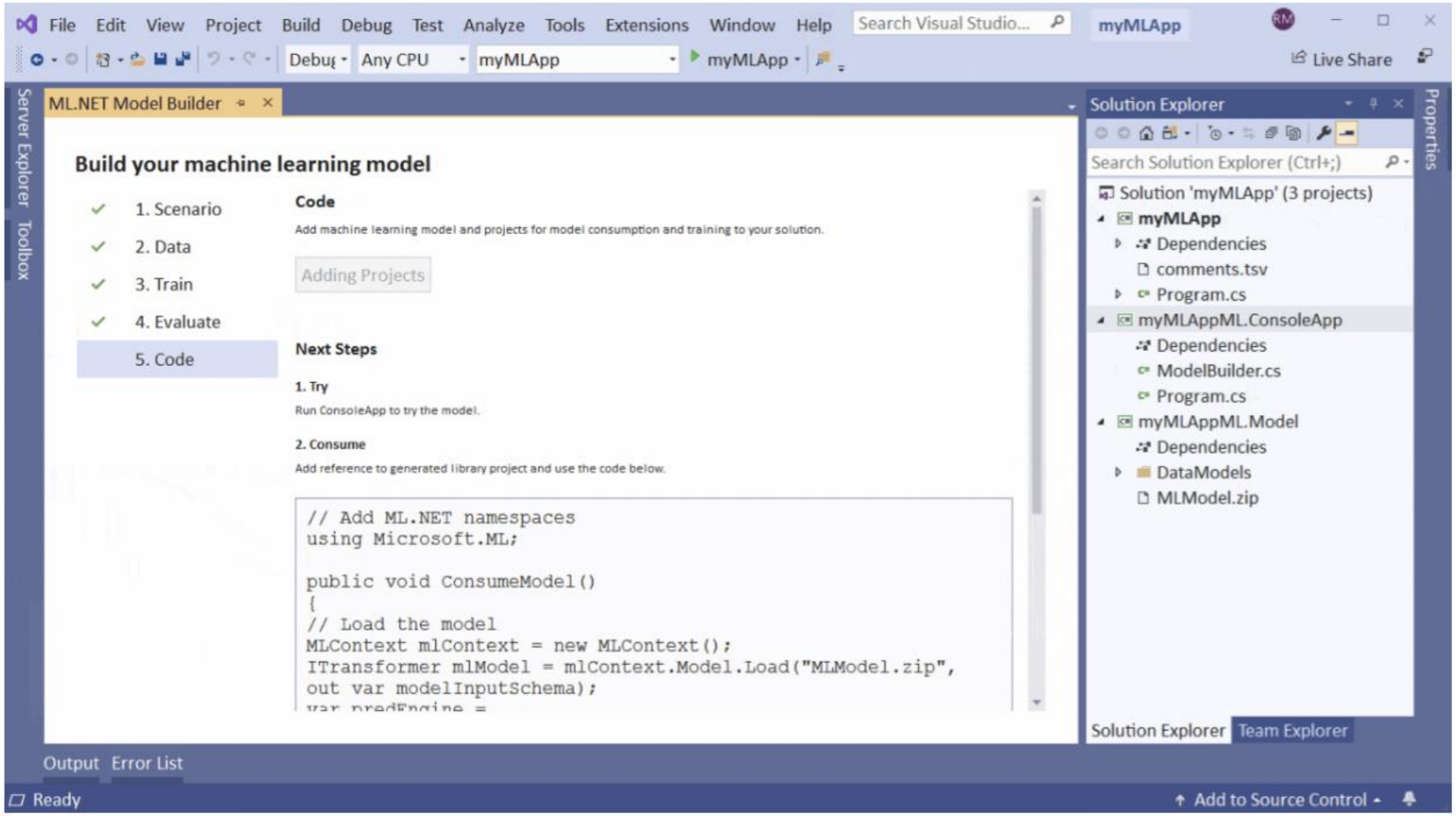
ML.NET inside FrameworX

Complete Solution

- Typical Data type: structured/tabular sensor data (temp, vibration, pressure)
- Speed: fast, compiled
- Integration: full .NET/C# compatibility; deploy inside FrameworX without external runtimes
- Model Building: Visual Studio Model Builder extension to create/train/evaluate to create code or manually written code which calls methods

ML.NET Model Builder

Visual Studio



Python-based libraries

Complete Solution

- Typical Data type: complex or time-series sensor data
- Speed: Slower, interpreted
- Integration: Run Python code in FrameworX
- Model Building: Write code to access open-source machine learning libraries. Typically create a pipeline of scripts to build required functionality.

Choose the right tool for the job

ML.NET vs Python (both are valid)

ML.NET/C# + Python support → deploy the right approach for each application.

When ML.NET is a great fit

- Speed matters (fast reactions, real-time setpoint/decisions)
- Tight .NET integration
- Easy, menu-driven code creation with Model Builder

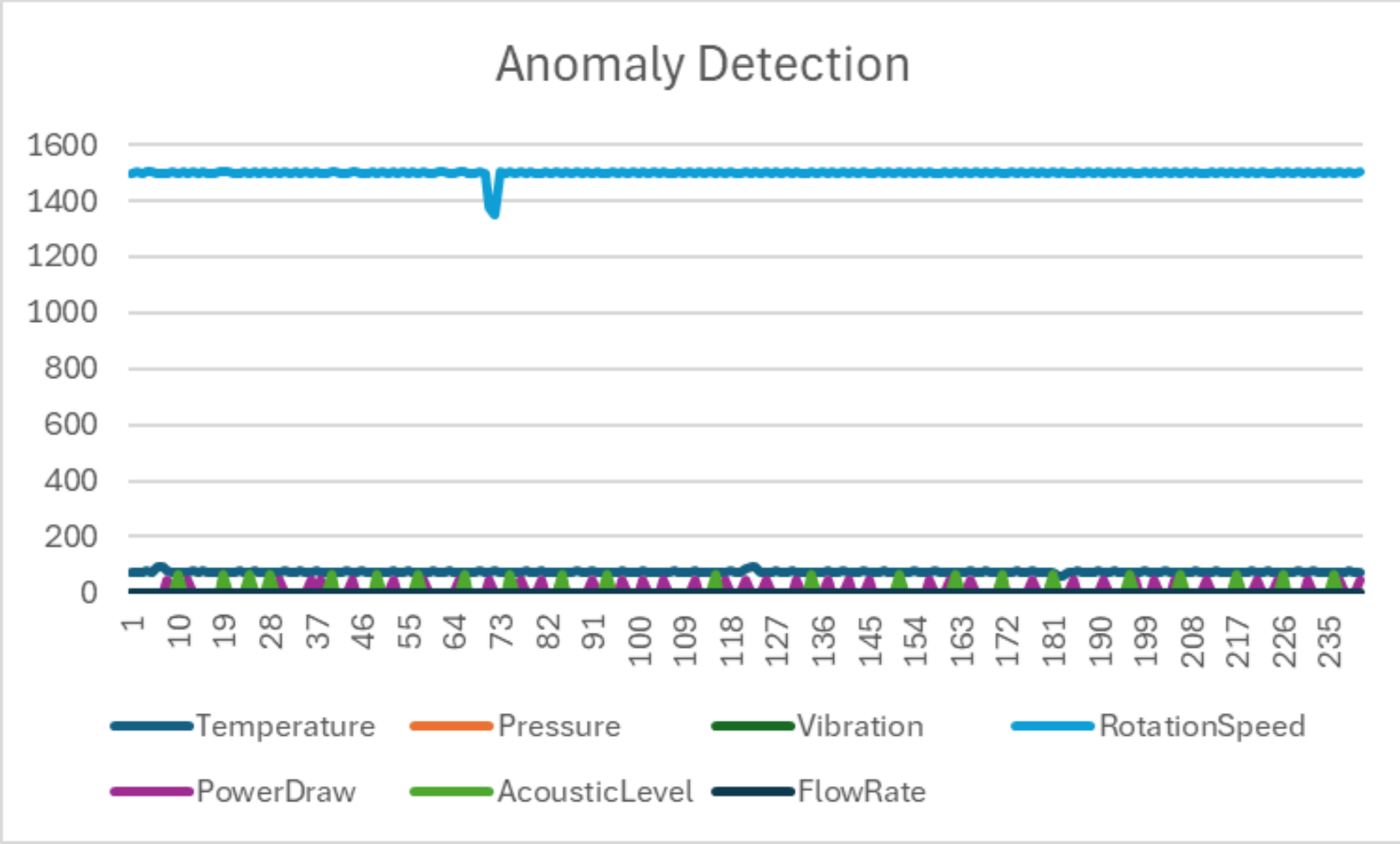
When Python is a great fit

- Richer ML ecosystem and libraries
- Deep learning / neural networks
- Heavier, offline analysis and planning

Demo: ML inside FrameworX

Anomaly Detection

Machine Learning



Name	Value
AccousticLevel	76.7466956907482
FlowRate	26.4813852177961
IsAbnormal	Abnormal
PowerDraw	53.8901434565185
Pressure	12.4006607336845
RotationSpeed	1500.1589601594
Score	0.60468852519989
Temperature	99.8498622188511
Vibration	3.14479811491097

Name	Value
AccousticLevel	69.3046045778247
FlowRate	26.6561734396889
IsAbnormal	Normal
PowerDraw	47.4823470761861
Pressure	13.5059260954551
RotationSpeed	1498.67228179664
Score	0.339477300643921
Temperature	78.6996222053799
Vibration	3.23727443229141

ML.NET Workflow

A repeatable 7-step pattern

1. Load data
2. Create input/output classes
3. Build a pipeline
4. Train model
5. Evaluate model
6. Save model
7. Use model with new data

Why predictive maintenance?

Reactive vs preventive vs predictive

Strategy	Description	Pros	Cons
Reactive maintenance	Fix when it breaks	Low upfront cost	High downtime, unpredictable
Preventive maintenance	Fix on a fixed schedule	Structured planning	Can cause over- or under-maintenance
Predictive maintenance	Fix based on actual condition & predictions	Optimized cost, minimal downtime	Requires data, sensors, analytics

Tablet Compression Machine

5-step approach

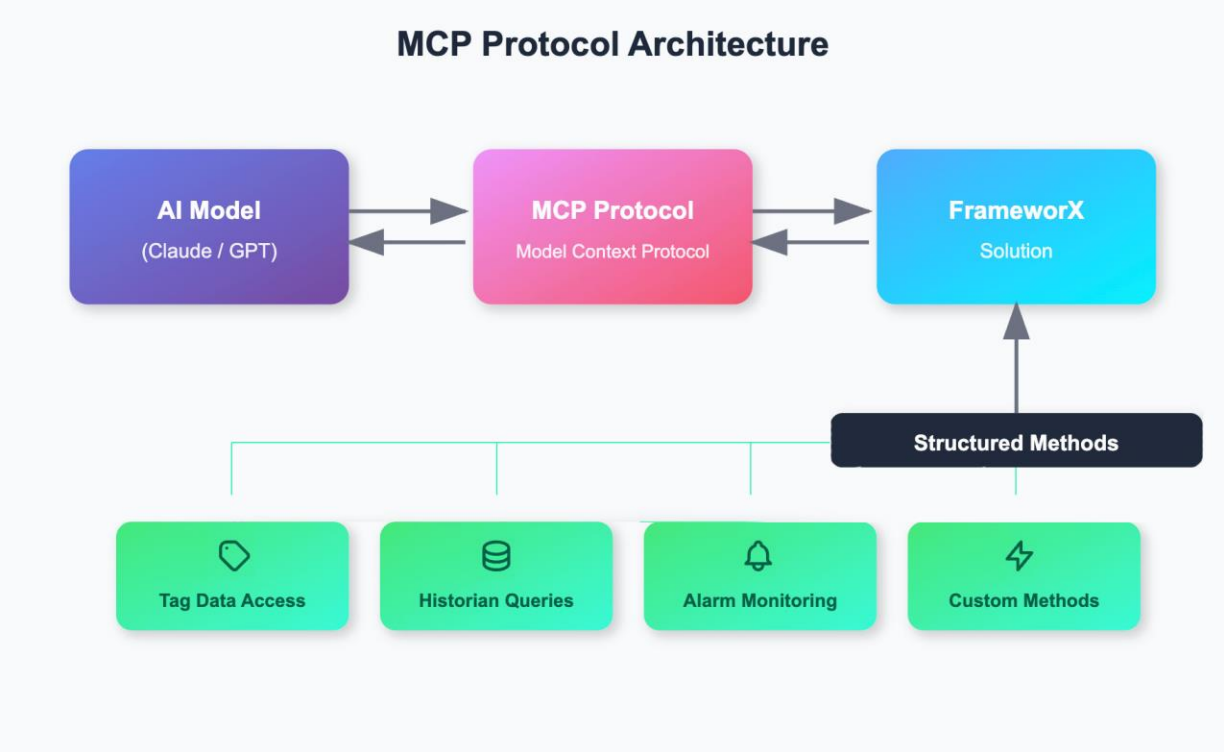
A practical way to start:

1. Collect healthy data (normal operation only)
2. Train an anomaly detection model (learn “normal”)
3. Monitor live machine data (current readings)
4. Detect & notify (normal vs anomaly → alarm)
5. Act on the alert (assess severity; schedule maintenance)

MCP (Model Context Protocol) — quick look

Connecting FrameworX to external AI services

- FrameworX can connect to external AI services / cloud models via MCP
- This is most useful when you want natural-language access to operational context
- Example: query historian + ML outputs (“Which motors look likely to fail next month?”)
- We’ll keep this grounded: operational data first, then AI on top



Wrap-Up + Q&A

Practical guidance

How to succeed with ML in operations

Start small: pick one asset/process and one clear question

Anomaly detection is often the best first step (no labeled data required)

Expand gradually: more signals, tuning, thresholds, forecasting/regression, external models

Re-train: ML improves through feedback and re-training

The hardest part isn't the code — it's the data (quality, consistency, context)

Key Takeaway

One thing to remember

Machine learning works best when it is part of the system — not a separate experiment.

Start small • Re-train • Let the platform do the heavy lifting

Q&A

Thanks for joining!

- Schedule a call with our team
- Download the software and start today

tatsoft.com/ml