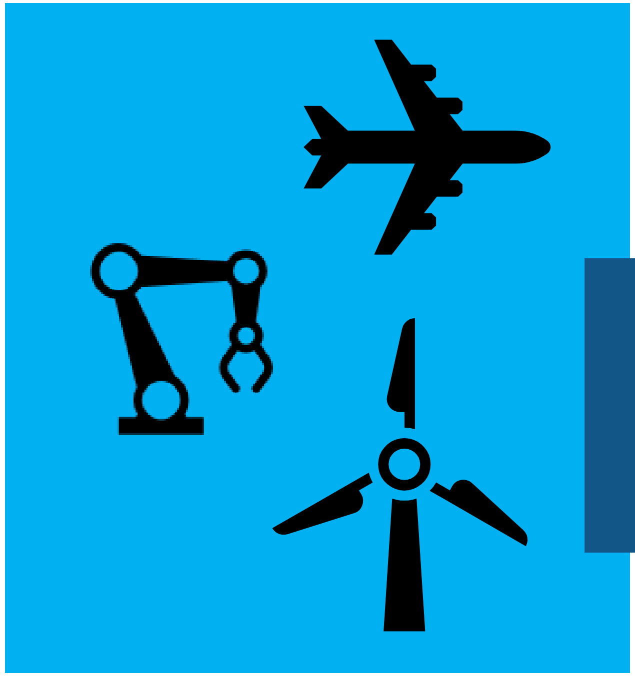
MATLAB EXPO 2019

제조 생산 현장에서 관리 시스템까지

빠른 인공지능 기반 시스템 구축

엄준상

The Need for Large-Scale Streaming



Jet engine: ~800TB per day
Turbine: ~ 2 TB per day

Predictive Maintenance

Increase Operational Efficiency

Reduce Unplanned Downtime

More applications require near real-time analytics

Medical Devices

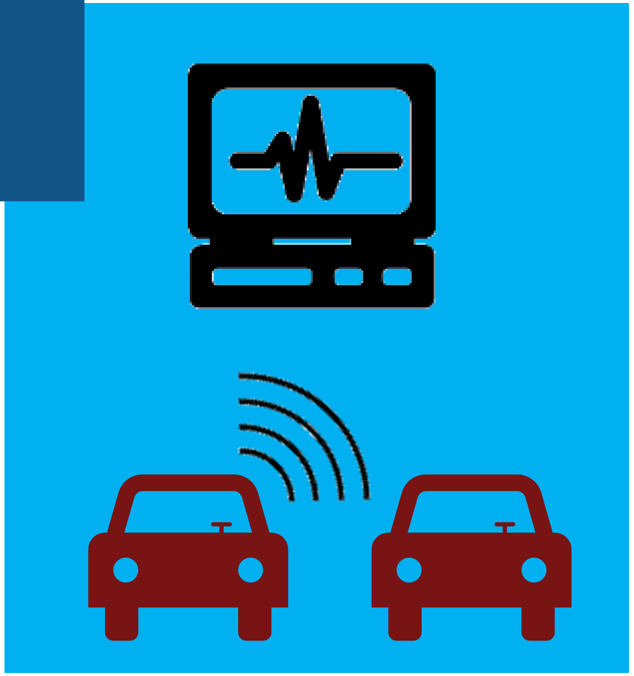
Patient Safety

Better Treatment Outcomes

Connected Cars

Safety, Maintenance

Advanced Driving Features



Car: ~25 GB per hour

Example Problem: Develop a machine learning model to predict failures in industrial pumps

- We did this for the customer
- We wanted to go further:
  - Create a streaming application based on this real customer request
  - Develop application in a 3-4 week sprint
- We believe this represents a realistic customer situation

Our Project: Develop and operationalize a machine learning model to predict failures in industrial pumps

Process Engineer

Develops models in MATLAB and Simulink

System Architect

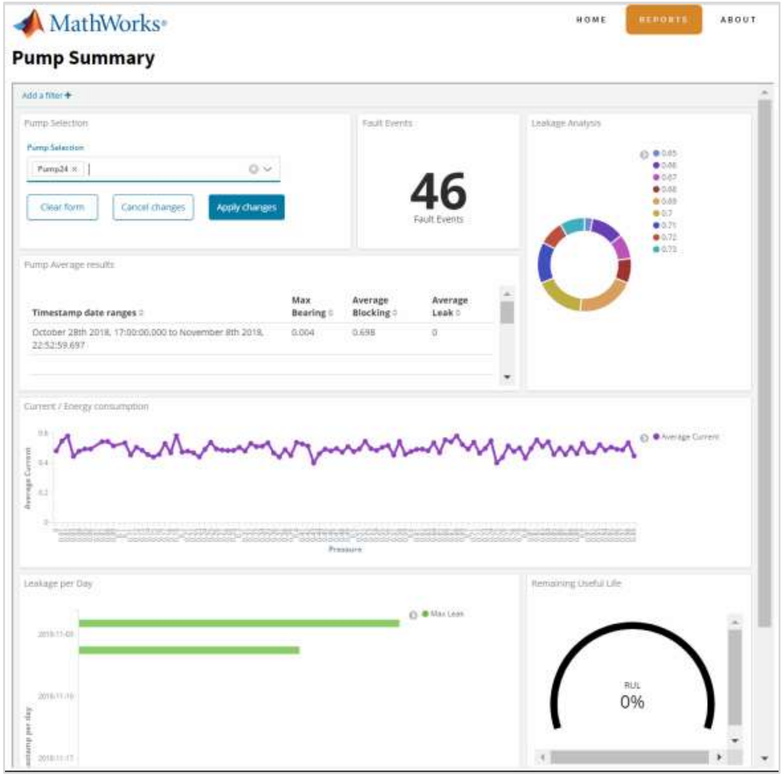
Deploys and operationalizes model on Azure cloud

Azure

Operator

Makes operational decisions based on model output

kibana



Current system requires Operator to manually monitor operational metrics for anomalies. Their expertise is required to detect and take preventative action

12,000 PSI

Project statement: Develop end-to-end predictive maintenance system and demo in one 3-4 week sprint

Plant Operator

1. Monitor ***flow***, ***pressure***, and ***current*** of each pump so I always know their ***operational state***
2. Need ***alert*** when fault parameters drift outside an acceptable range so I can take ***immediate corrective action***
3. Continuous estimate of each pump's ***remaining useful life (RUL)*** so I can ***schedule maintenance or replace*** the asset

Challenges of AI Deployment

Process Engineer

We don’t have a large set of failure data, and it’s too costly to generate real failures in our plant for this project

Solution: Use an accurate physics-based software model for the pump to develop synthetic training sets

MathWorks

Challenges of AI Deployment

We don’t have a large IT/hardware budget, and we need to see results before committing to a particular platform or technology

System Architect

Solution: Leverage cloud platform to quickly configure and provision the services needed to build the solution, while minimizing lock-in to a particular provider

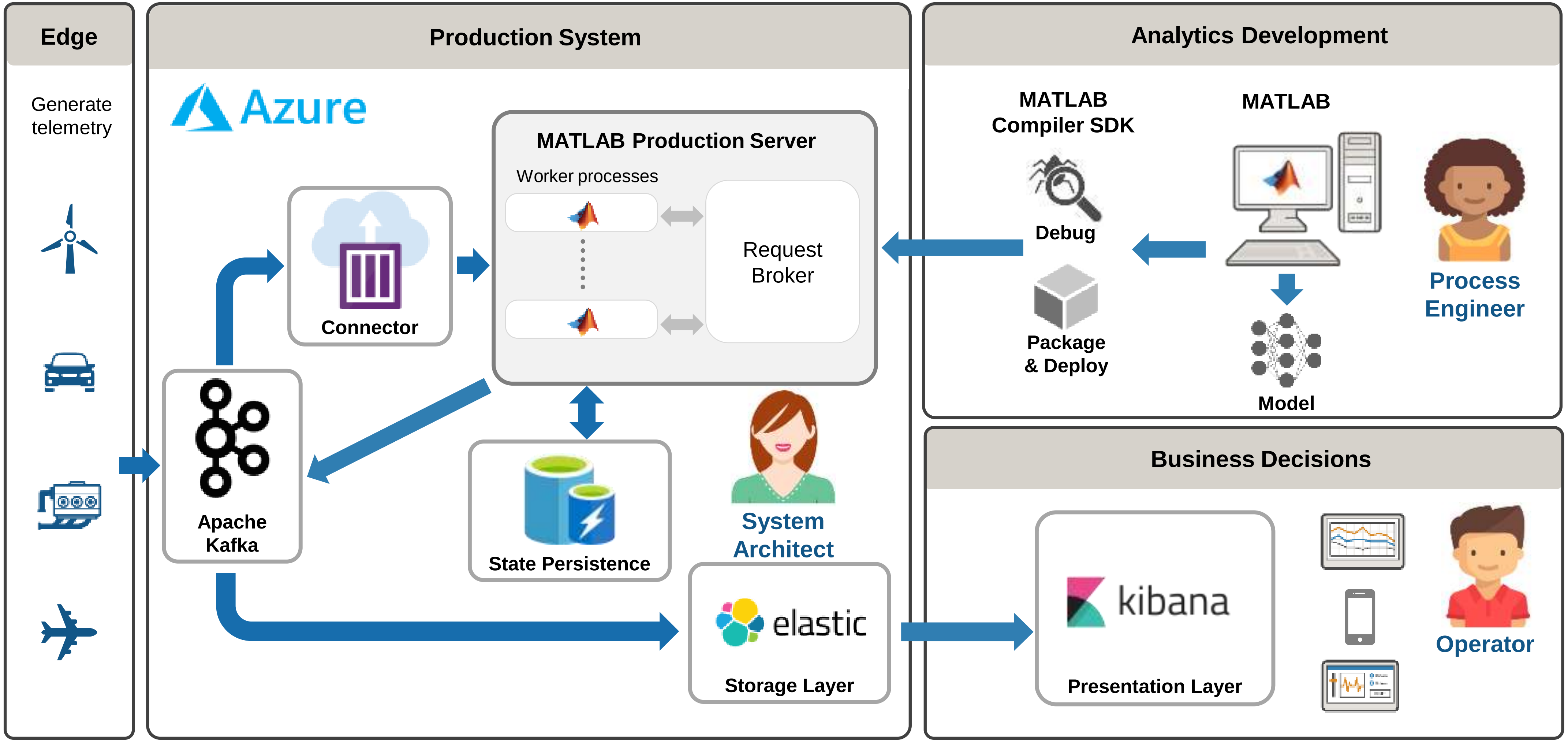
Challenges of AI Deployment

Process Engineer

Need software for multidisciplinary problem across teams, plus integration w/ IT

Solution: Use MATLAB and integrate with OSS

Predictive Maintenance Architecture on Azure



Modeling approach

Process Engineer

1. **Access and Explore Data**
   - Files
   - Databases
   - Sensors
2. **Preprocess Data**
   - Working with Messy Data
   - Data Reduction/ Transformation
   - Feature Extraction
3. **Develop Predictive Models**
   - Model Creation e.g. Machine Learning
   - Parameter Optimization
   - Model Validation
4. **Integrate with Production Systems**
   - Desktop Apps
   - Enterprise Scale Systems: kafka, AWS Kinesis, OSIsoft
   - Embedded Devices and Hardware
5. **Visualize Results**
   - 3rd party dashboards: Spotfire TIBCO Software, tableau, Qlik, Power BI
   - Web apps

Process Engineer

Review model requirements

Requirements From Operator

- Continuous predictions of type of fault
  - “Blocking”
  - “Leaking”
  - “Bearing”
  - Combination of above
- Continuous predictions of Remaining Useful Life [RUL]

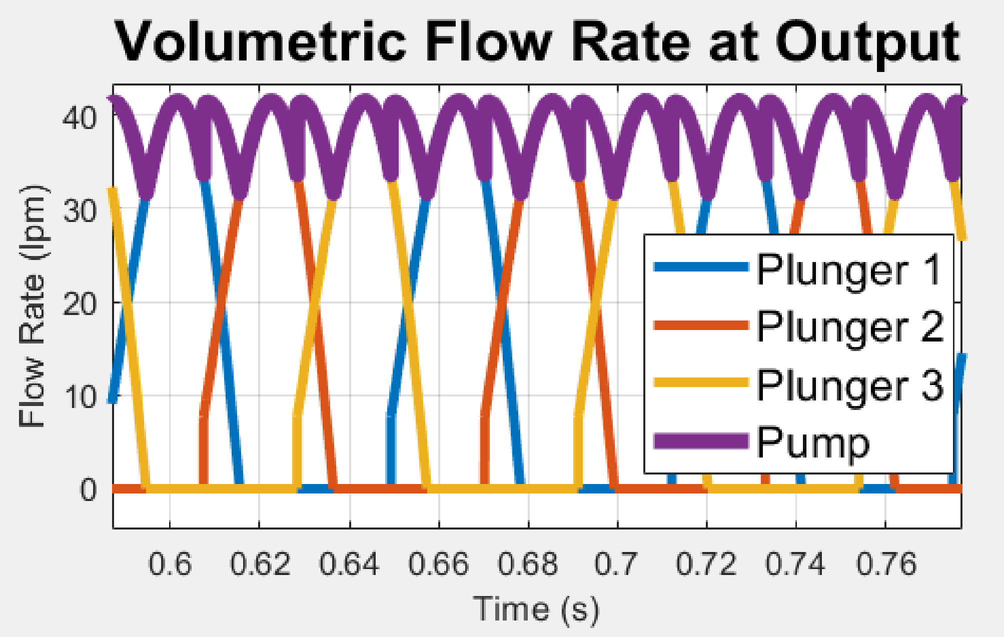
Requirements From System Architect

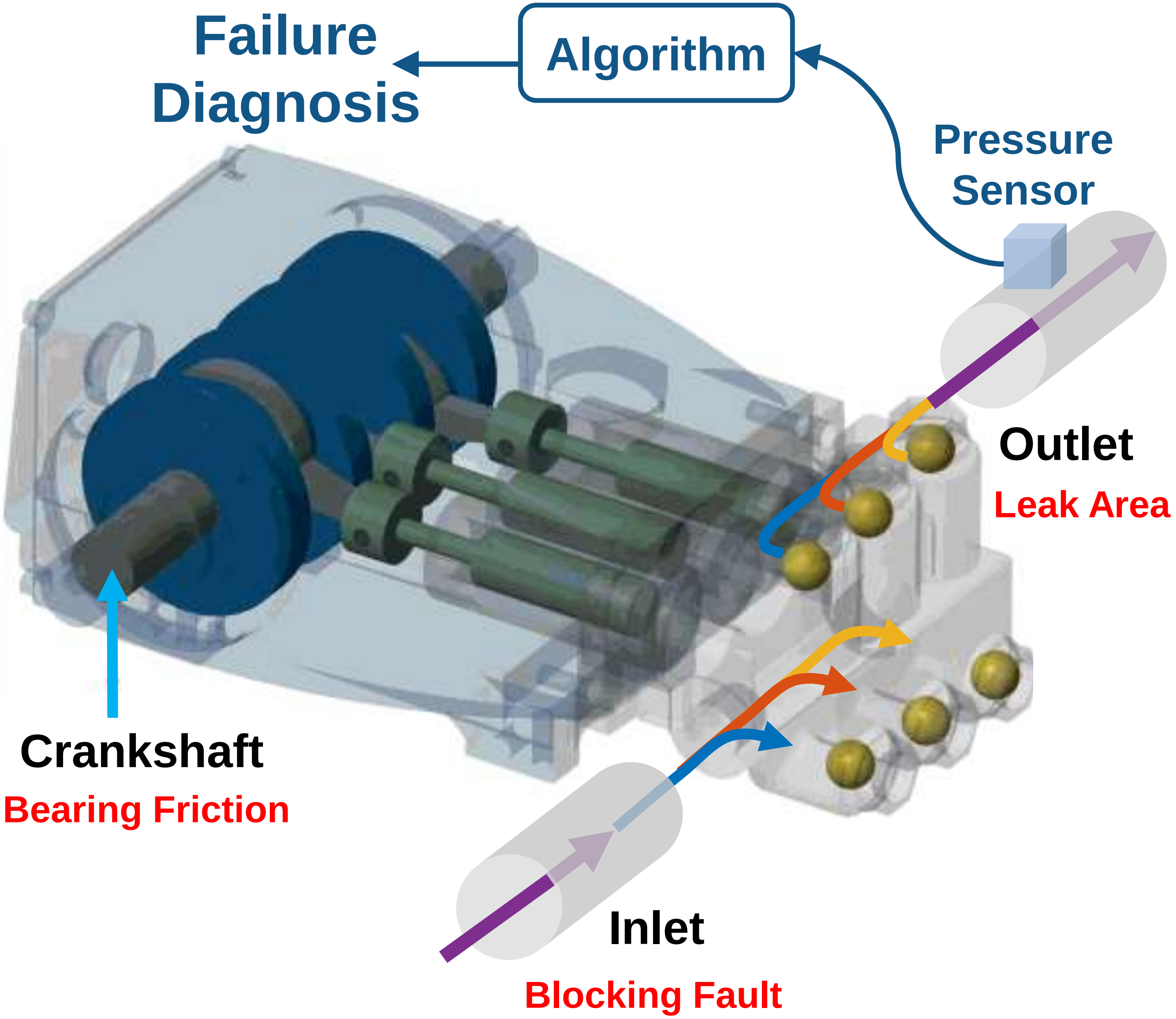
- Define window for streaming
- Define format of results, intermediate values
- Test code
- Scale code

Physics of Triplex Pump

Process Engineer

- Crankshaft drives three plungers
  - Each 120 degrees out of phase
  - One chamber always discharging
  - Three types of **failures**

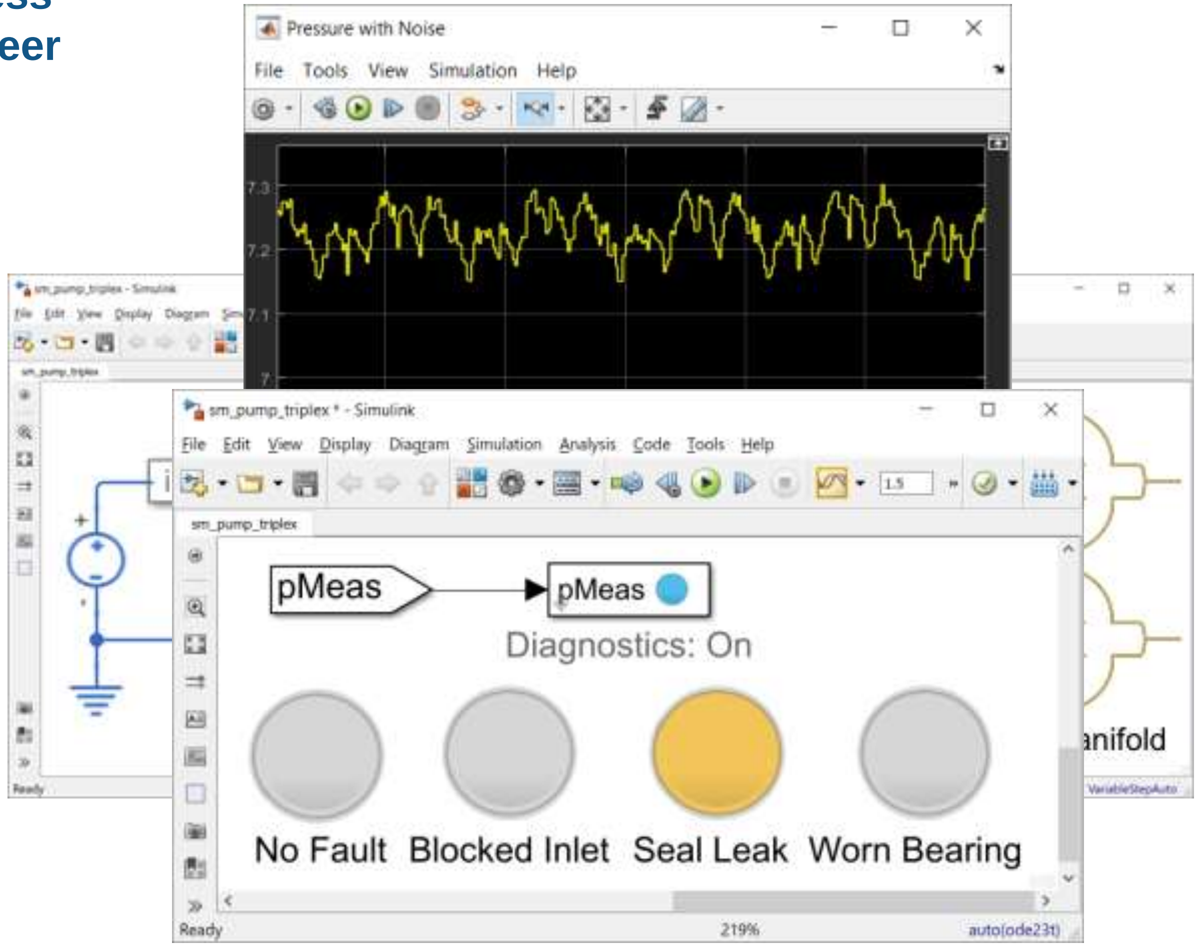




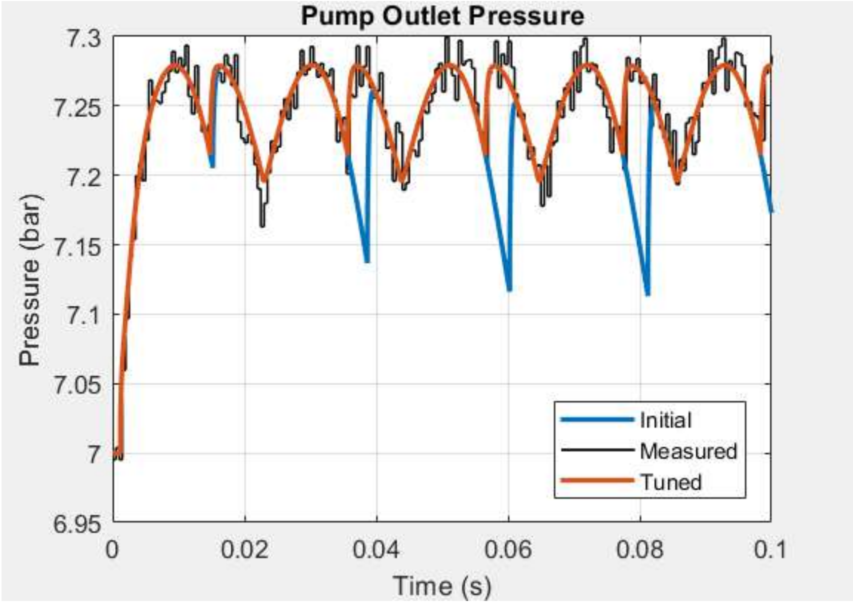
Process Engineer

1 Access and Explore Data

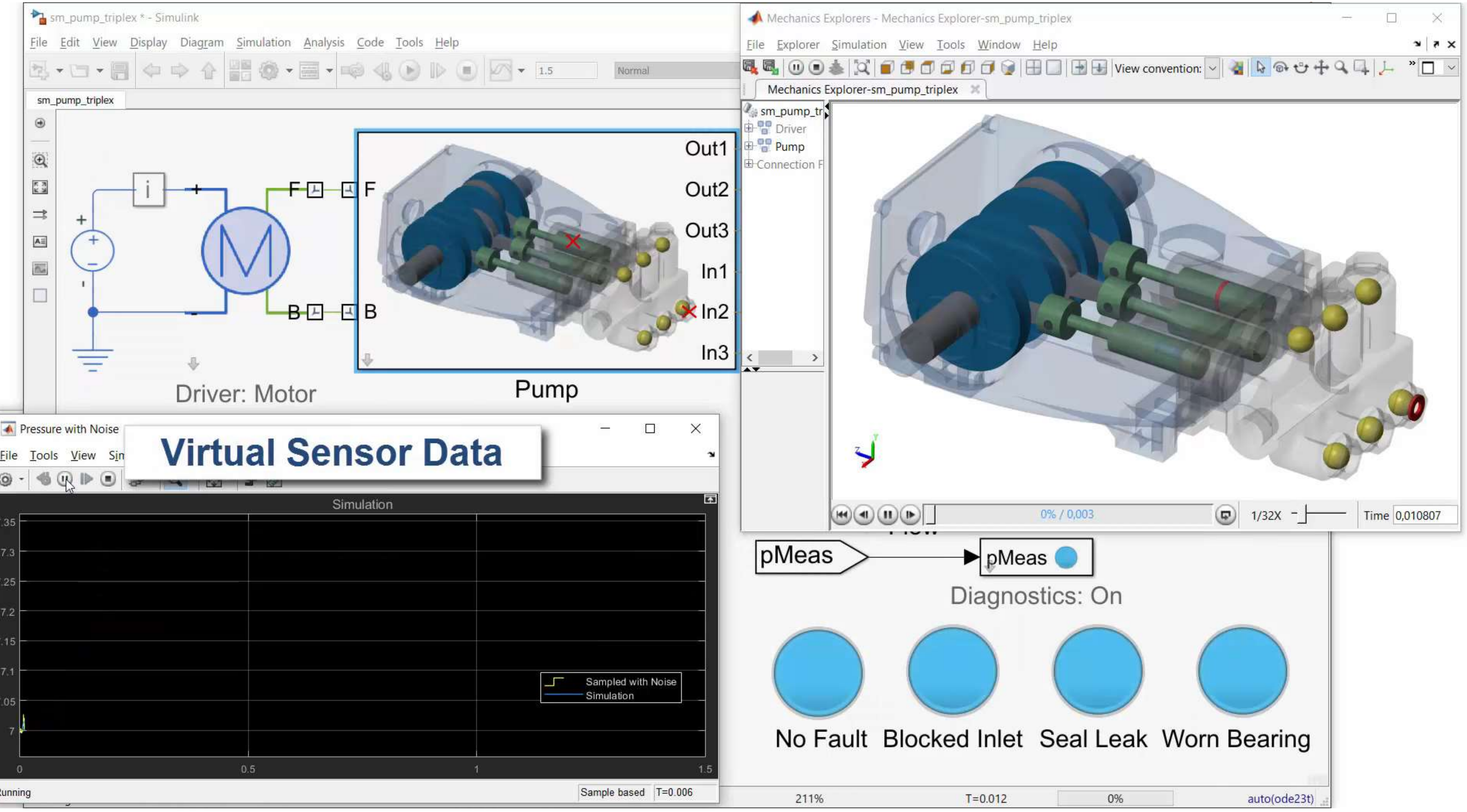
Use sensor data from pump to identify levels of failure



Simulate faults



Pump sensor data

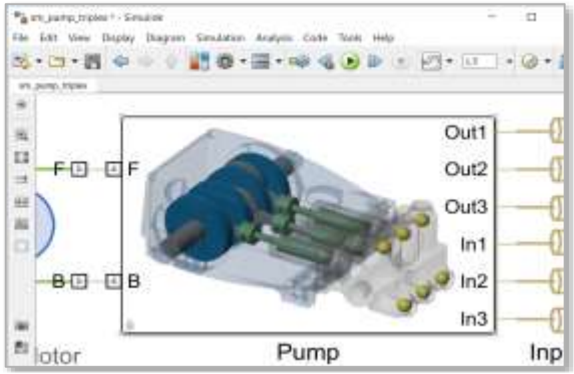


sm_pump_triplex * - Simulink
File Edit View Display Diagram Simulation Analysis Code Tools Help
sm_pump_triplex
Out1
Out2
Out3
In1
In2
In3
F
B
Driver: Motor
Pump
Mechanics Explorers - Mechanics Explorer-sm_pump_triplex
File Explorer Simulation View Tools Window Help
View convention:
Mechanics Explorer-sm_pump_triplex
Driver
Pump
0% / 0,003
1/32X
Time 0,010807
Pressure with Noise
Virtual Sensor Data
Simulation
Sampled with Noise
Simulation
Running
Sample based T=0.006
pMeas
pMeas
Diagnostics: On
No Fault
Blocked Inlet
Seal Leak
Worn Bearing
211%
T=0.012
0%
auto(ode23t)

Process Engineer

1 Access and Explore Data

Simulate data with many failure conditions



Leak Area = [1e-9 0.036]

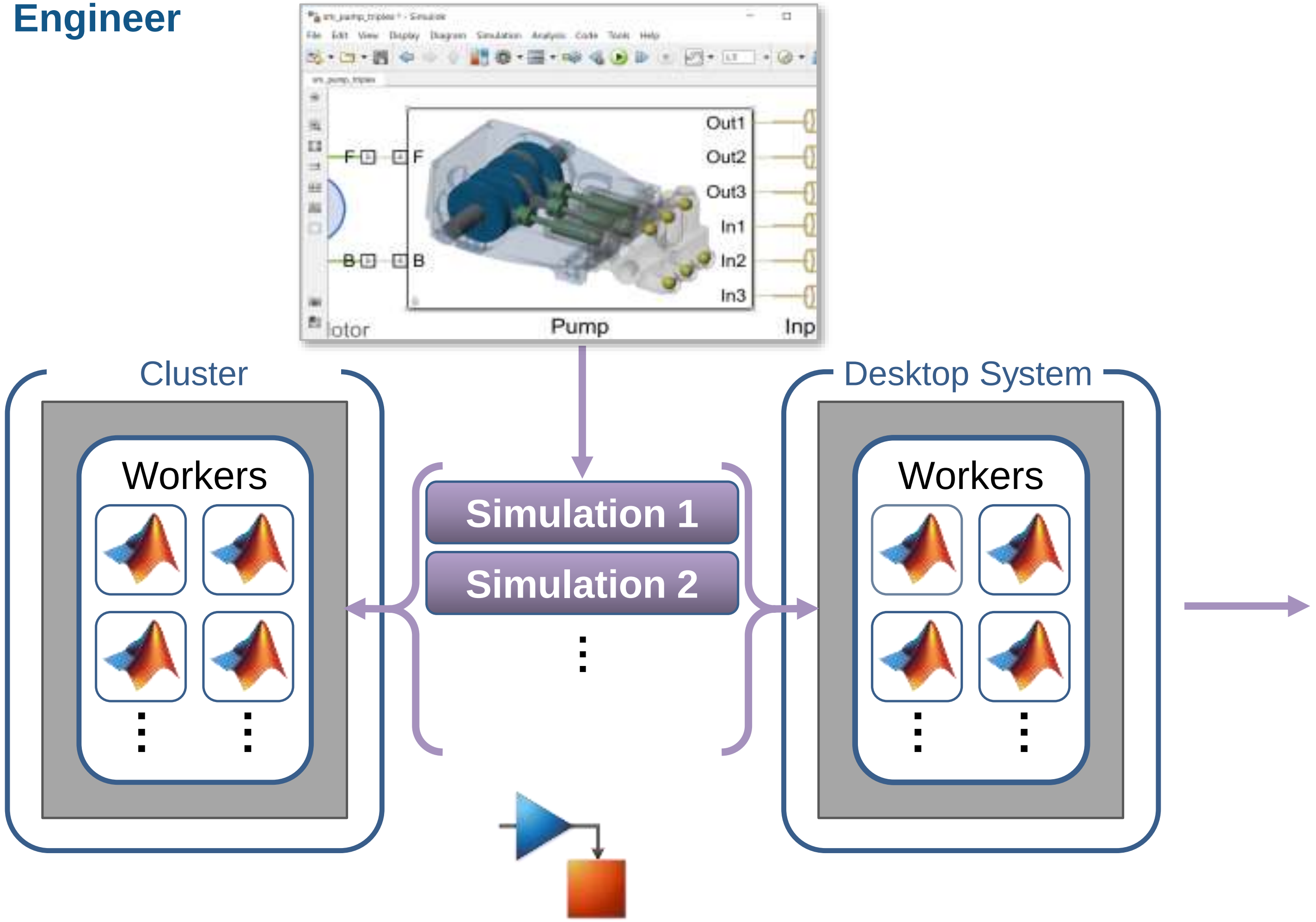
Bearing Friction = [0 6e-4]

Blocking Fault = [0.5 0.8]

Process Engineer

1 Access and Explore Data

Simulate data with many failure conditions



Run parallel simulations

Access Data

```
ens = simulationEnsembleDatastore(location)
```

```
ens = 

  simulationEnsembleDatastore with properties:

           DataVariables: [25×1 string]
    IndependentVariables: [0×0 string]
      ConditionVariables: [0×0 string]
       SelectedVariables: [25×1 string]
                ReadSize: 1
              NumMembers: 702
          LastMemberRead: [0×0 string]
                   Files: [702×1 string]
```

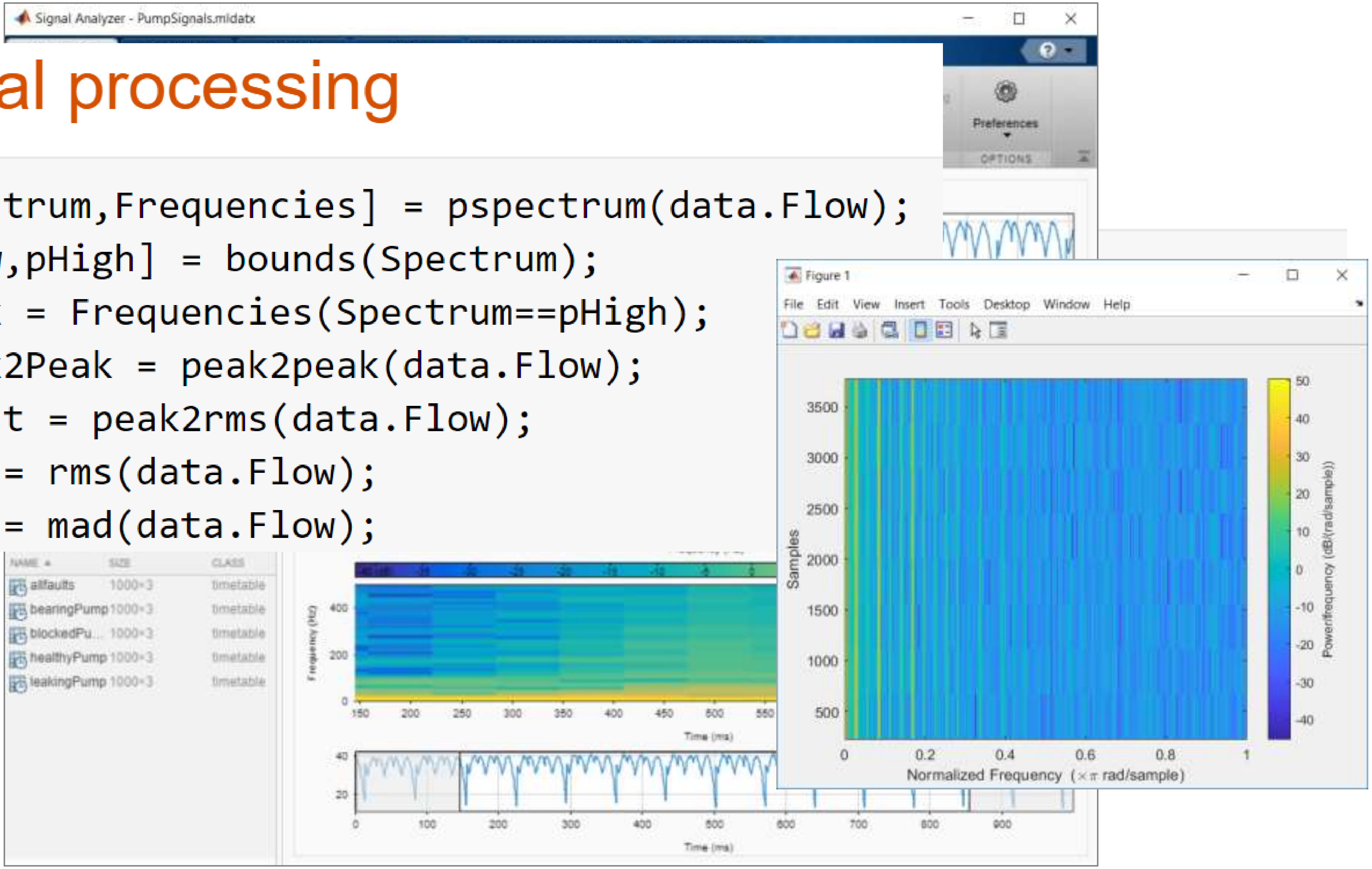
Store data on HDFS

Process Engineer

Represent signal information

Signal processing

```
[Spectrum,Frequencies] = pspectrum(data.Flow);
[pLow,pHigh] = bounds(Spectrum);
fPeak = Frequencies(Spectrum==pHigh);
qPeak2Peak = peak2peak(data.Flow);
qCrest = peak2rms(data.Flow);
qRMS = rms(data.Flow);
qMAD = mad(data.Flow);
```



3

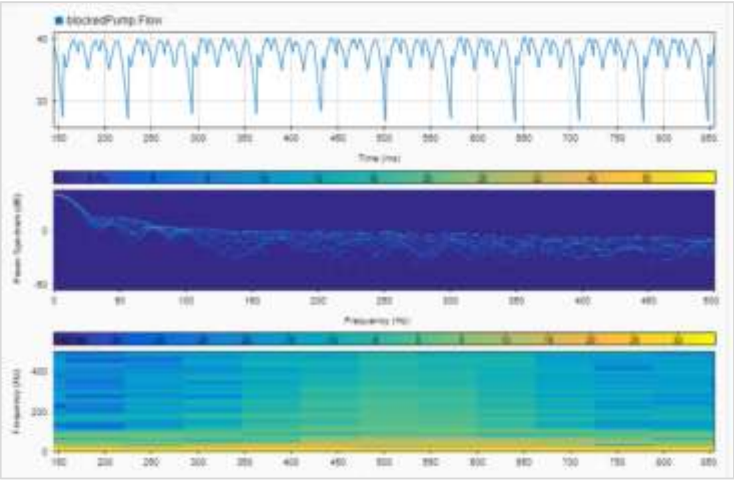
Develop Predictive Models

Develop Predictive Models in MATLAB

Process Engineer

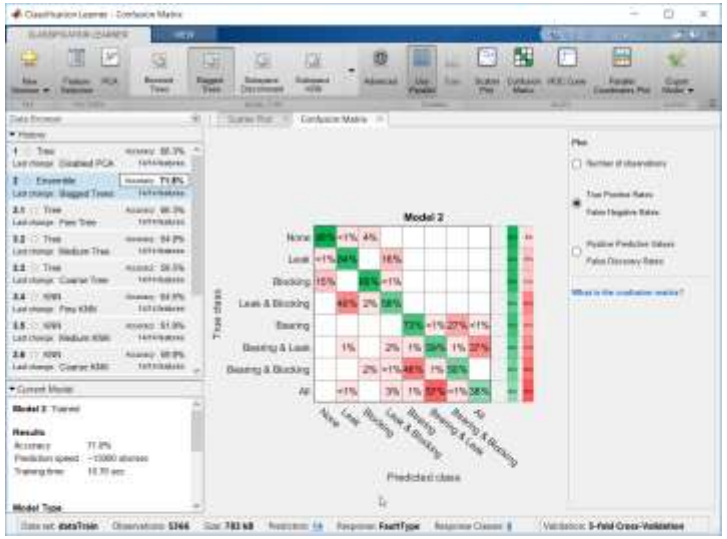
	Time	1 LeakFault	2 BlockingFault	3 BearingFault	4 FaultType
1	0 sec	2.8472	-0.1477	1.8000	All
2	0.001 sec	-0.1498	-0.4207	1.3103	Bearing & Blocking
3	0.002 sec	0.6511	1.6521	-0.5557	Leak
4	0.003 sec	0.1469	-0.2775	1.0074	All
5	0.004 sec	-0.6480	0.7065	-0.8878	Blocking
6	0.005 sec	-0.8165	-0.5434	-0.3079	Blocking
7	0.006 sec	-1.0061	1.2083	0.0661	Bearing
8	0.007 sec	1.0125	-1.9098	-0.7027	Leak & Blocking

Label Faults



Represent Signals

Train Model



Validate Model

Scale

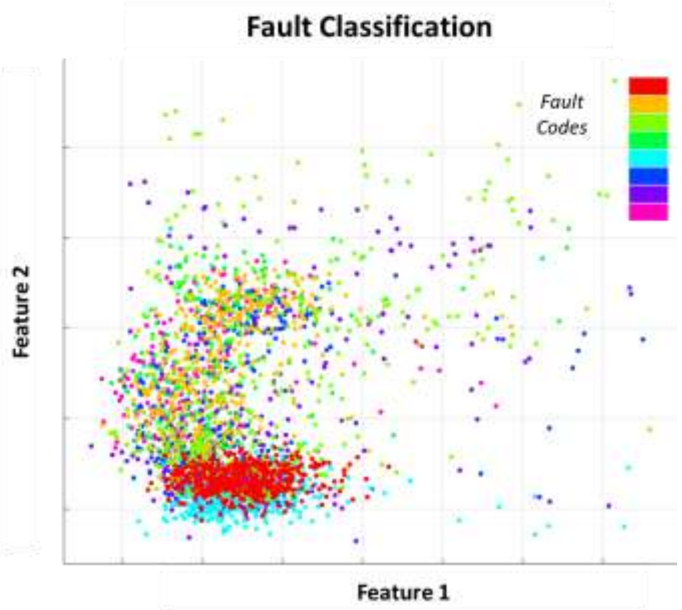
```
tt = tall(ds);
tt = preprocessData(tt);
model = TreeBagger(50,tt,'Event');
```

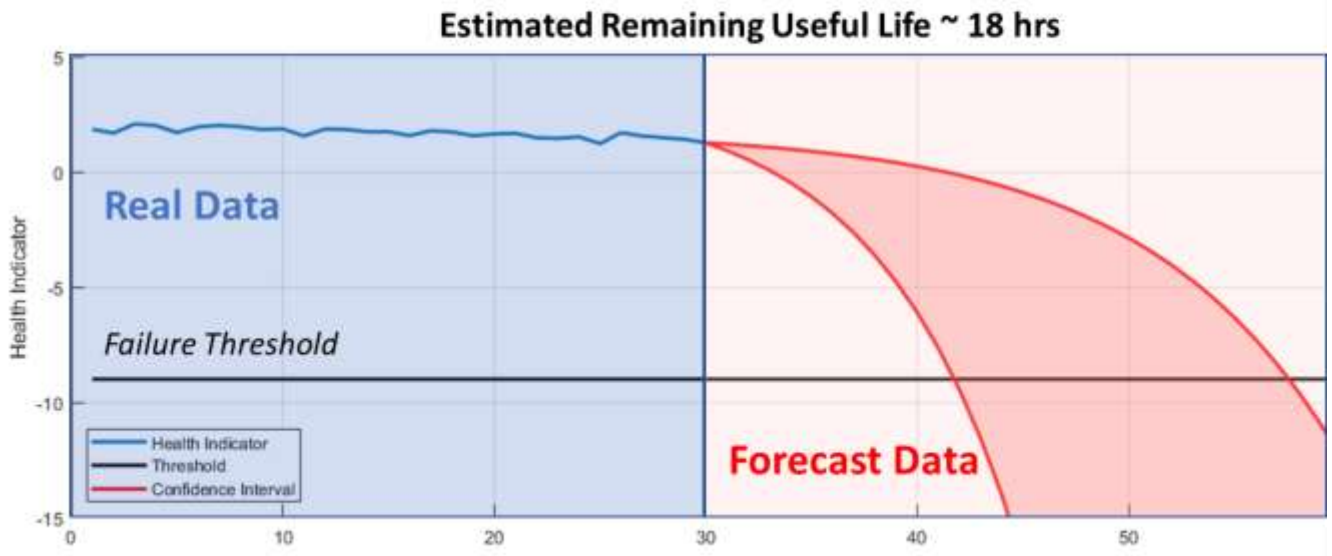
```
Evaluating tall expression using the Spark Cluster:
- Pass 1 of 2: Completed in 11 sec
- Pass 2 of 2: Completed in 2.3333 min
Evaluation completed in 2.6167 min
```

Process Engineer

3 Develop Predictive Models

Develop Predictive Models in MATLAB

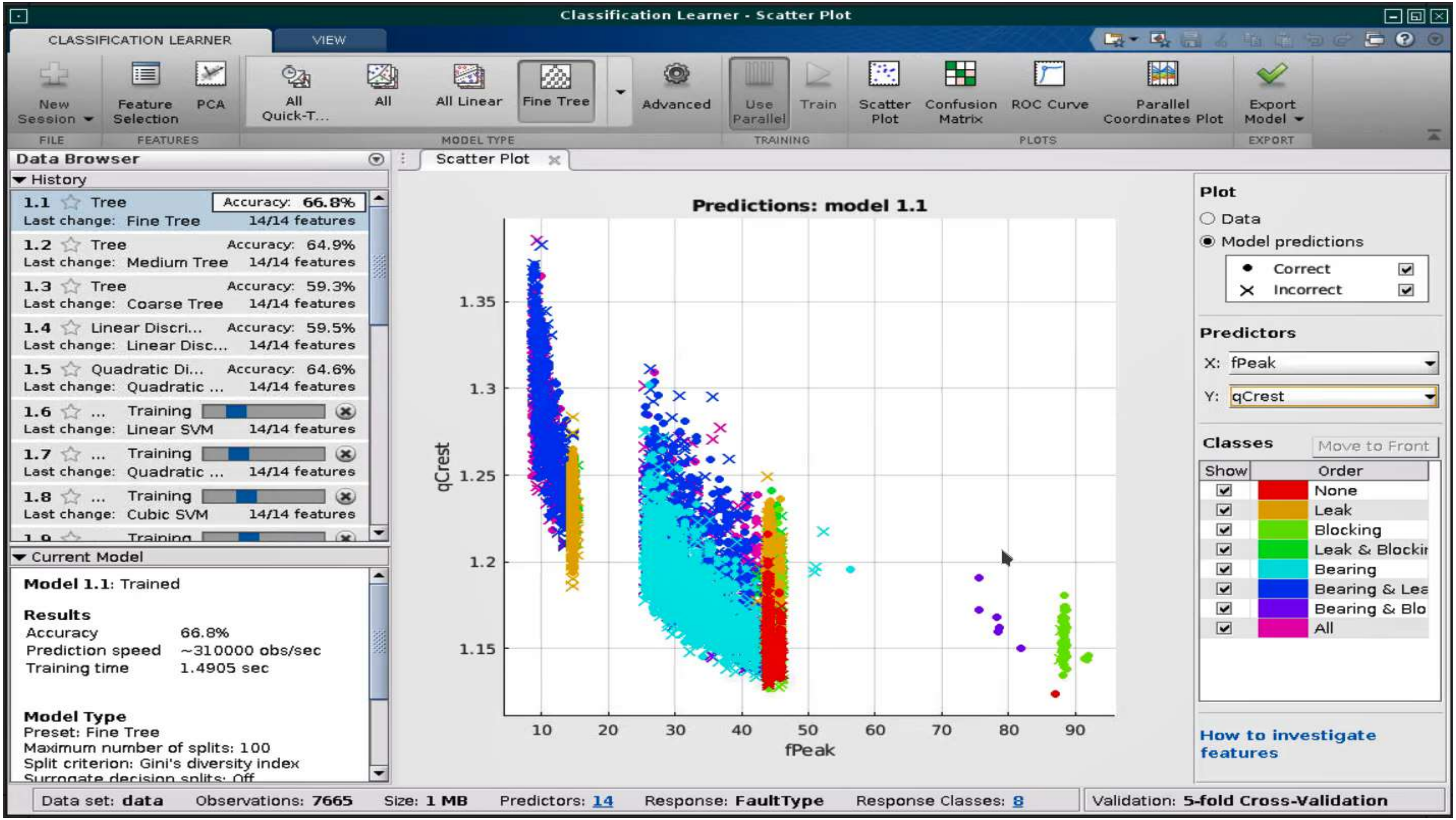




Type of Fault (Classification)

Remaining Useful Life (Regression)

Plant Operator

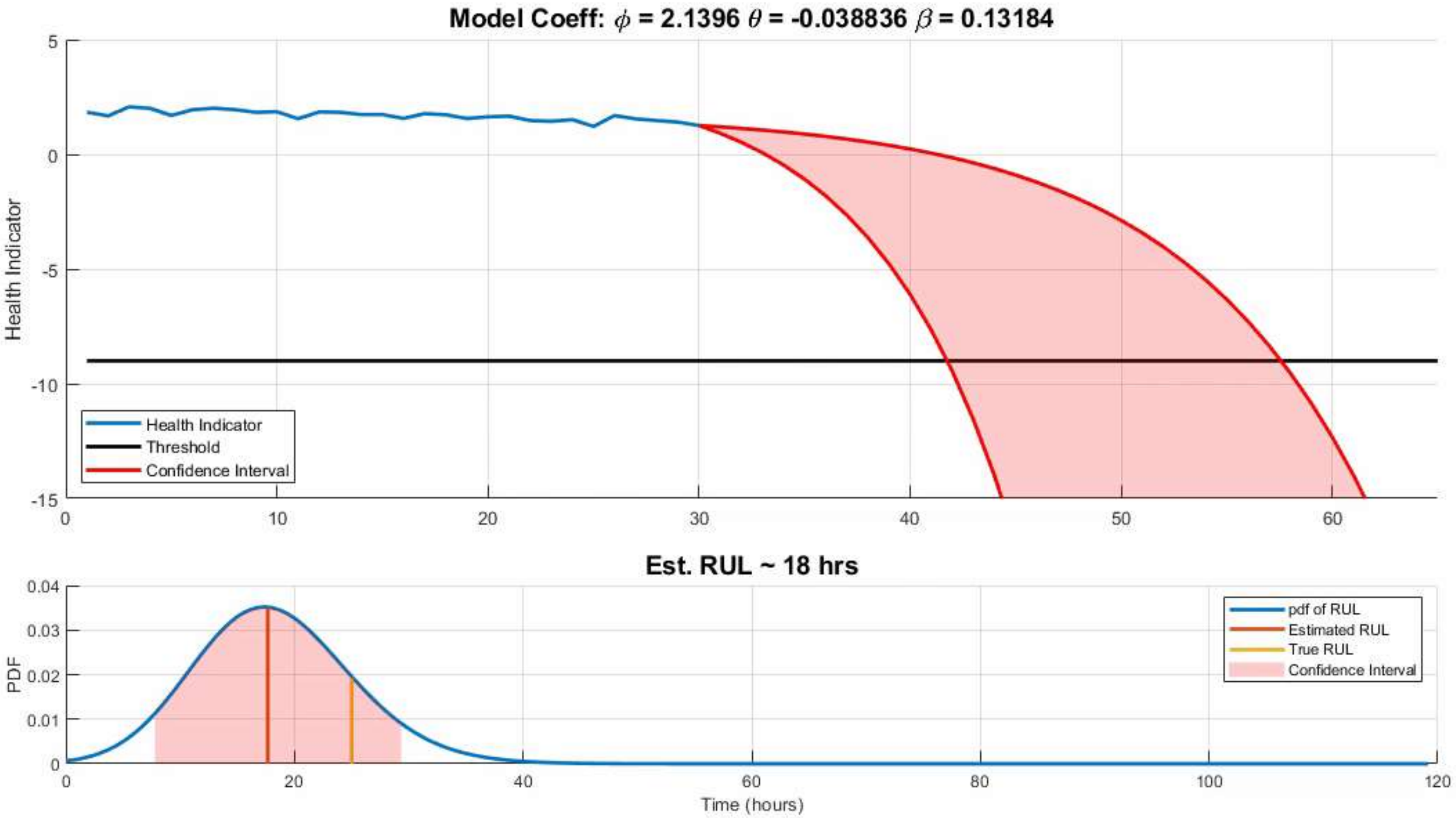


Classification Learner - Scatter Plot
CLASSIFICATION LEARNER
VIEW
New Session
Feature Selection
PCA
All Quick-T...
All
All Linear
Fine Tree
Advanced
Use Parallel
Train
Scatter Plot
Confusion Matrix
ROC Curve
Parallel Coordinates Plot
Export Model
FILE
FEATURES
MODEL TYPE
TRAINING
PLOTS
EXPORT
Data Browser
History
1.1 Tree Accuracy: 66.8%
Last change: Fine Tree 14/14 features
1.2 Tree Accuracy: 64.9%
Last change: Medium Tree 14/14 features
1.3 Tree Accuracy: 59.3%
Last change: Coarse Tree 14/14 features
1.4 Linear Discri... Accuracy: 59.5%
Last change: Linear Disc... 14/14 features
1.5 Quadratic Di... Accuracy: 64.6%
Last change: Quadratic ... 14/14 features
1.6 ... Training
Last change: Linear SVM 14/14 features
1.7 ... Training
Last change: Quadratic ... 14/14 features
1.8 ... Training
Last change: Cubic SVM 14/14 features
Current Model
Model 1.1: Trained
Results
Accuracy 66.8%
Prediction speed ~310000 obs/sec
Training time 1.4905 sec
Model Type
Preset: Fine Tree
Maximum number of splits: 100
Split criterion: Gini's diversity index
Scatter Plot
Predictions: model 1.1
qCrest
1.35
1.3
1.25
1.2
1.15
10
20
30
40
50
60
70
80
90
fPeak
Plot
Data
Model predictions
Correct
Incorrect
Predictors
X: fPeak
Y: qCrest
Classes
Move to Front
Show
Order
None
Leak
Blocking
Leak & Blockir
Bearing
Bearing & Lea
Bearing & Blo
All
How to investigate features
Data set: data
Observations: 7665
Size: 1 MB
Predictors: 14
Response: FaultType
Response Classes: 8
Validation: 5-fold Cross-Validation

Estimate Remaining Useful Life

3 Develop Predictive Models

Process Engineer



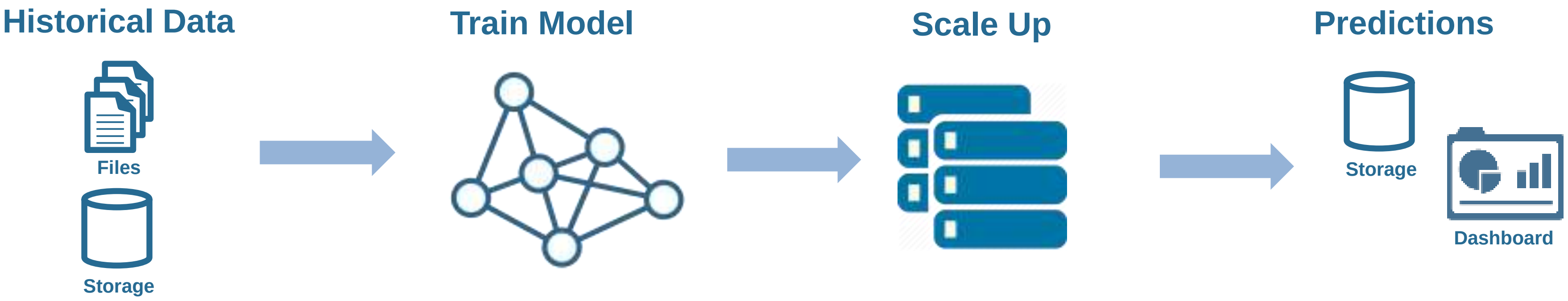
$$S(t) = \phi + \theta(t)\, e^{\left(\beta(t)t+\epsilon(t) - \frac{\sigma}{2}\right)}$$

Process Engineer

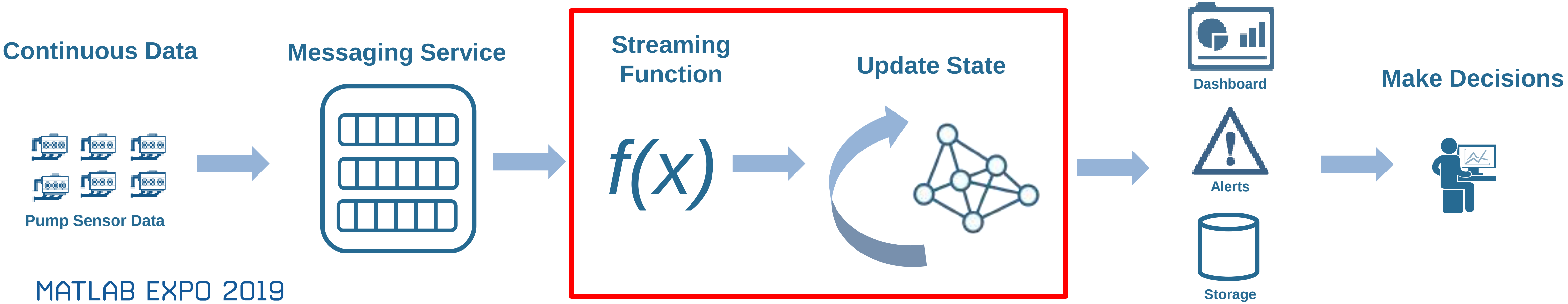
4 Integrate with Production Systems

Develop a Stream Processing Function

- ***Batch Processing:*** Build and test model on simulated data



- ***Stream Processing:*** Apply model to sensor data in near real-time



Process Engineer

4 Integrate with Production Systems

Develop a Stream Processing Function

Streaming Function

```
function new_state = streamingFunction(data,old_state)
```

Preprocess signals

```
[data,features] = preprocessData(data);
```

Predict faults

```
[Leak,Blocking,Bearing] = predictFaultValues(features);
FaultType = predictFault(features);
[RUL,Model] = predictUpdateRUL(data.Timestamp,data.Flow,500);
```

Update state

```
new_state = updateState(data,old_state);
```

Write results

```
writeResults(Leak,Blocking,Bearing,FaultType,RUL,Model)
end
```

Process each window of data as it arrives

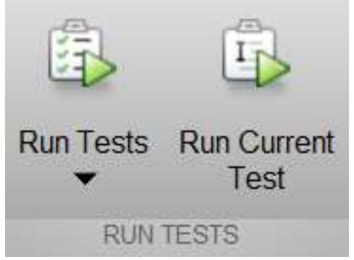
Previous state

Current window of data to be processed

Process Engineer

4 Integrate with Production Systems

Test Stream Processing Function

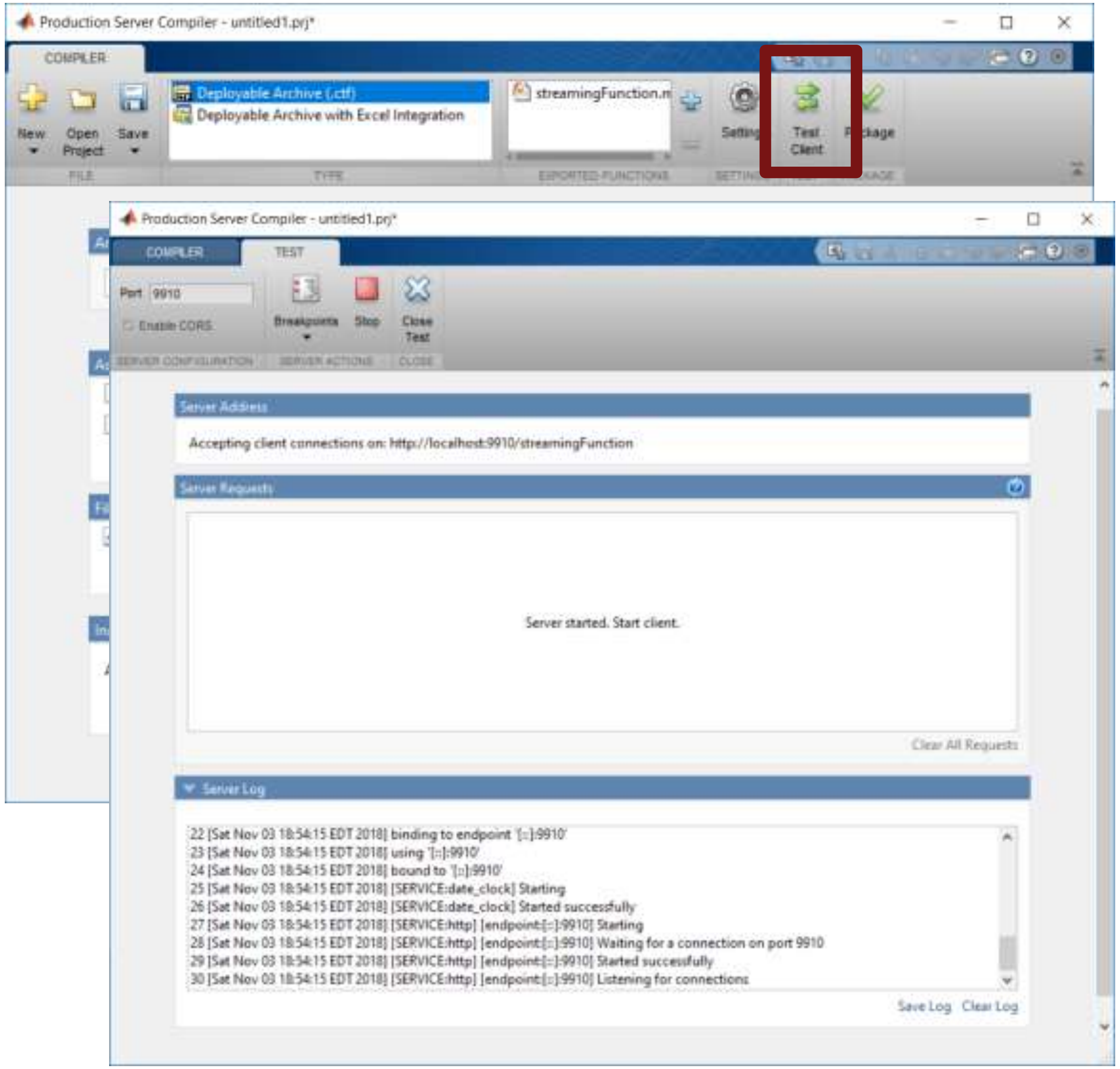



```
results = runtests('predictFaults_tests')
```

```
Running predictFaults_tests
....
Done predictFaults_tests
__________

results =
  1×4 TestResult array with properties:

    Name
    Passed
    Failed
    Incomplete
    Duration
    Details
Totals:
   4 Passed, 0 Failed, 0 Incomplete.
   0.01614 seconds testing time.
```

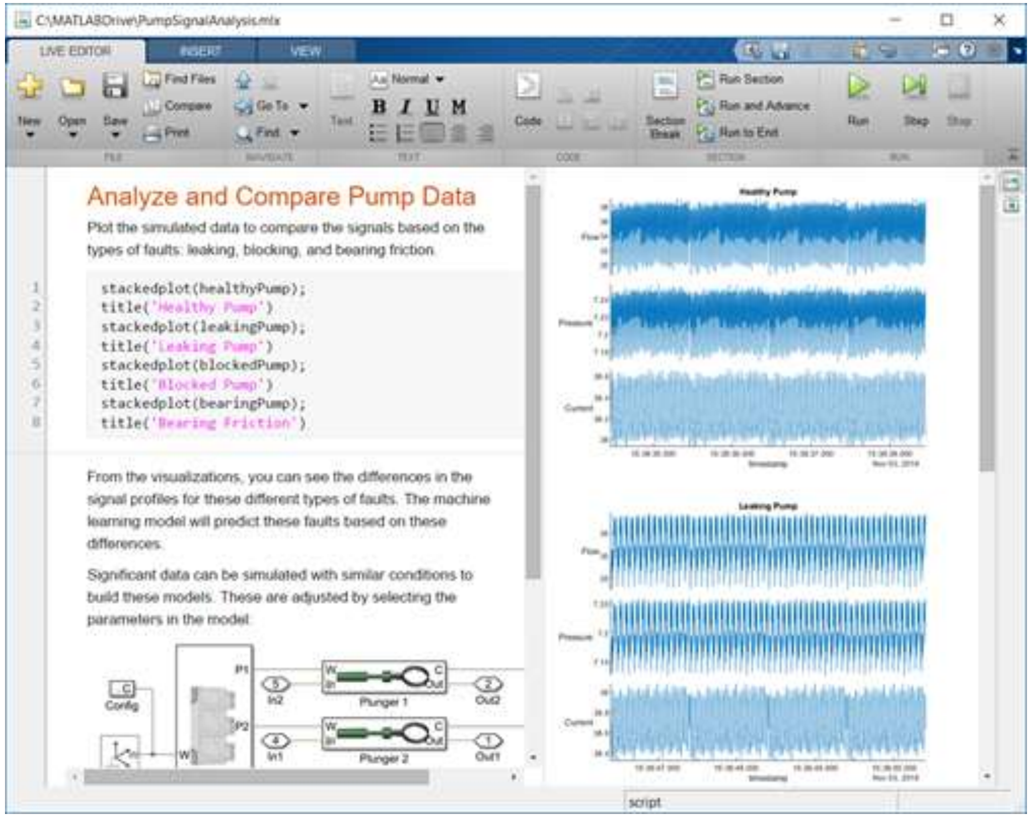


4 Integrate with Production Systems

Process Engineer

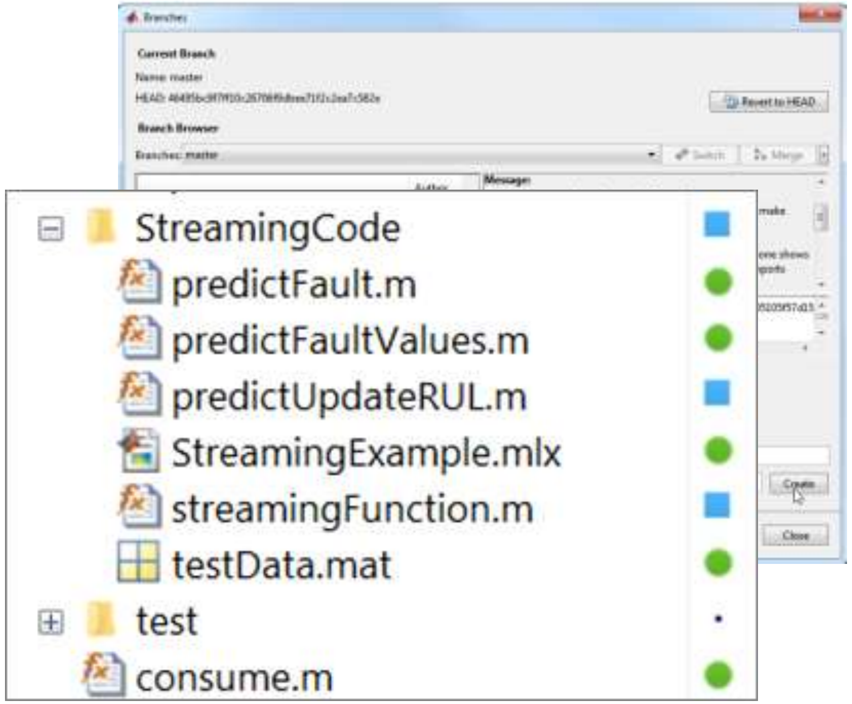
Share with the team

Review results with Operator



.pdf, html, LaTeX

Share code with System Architect

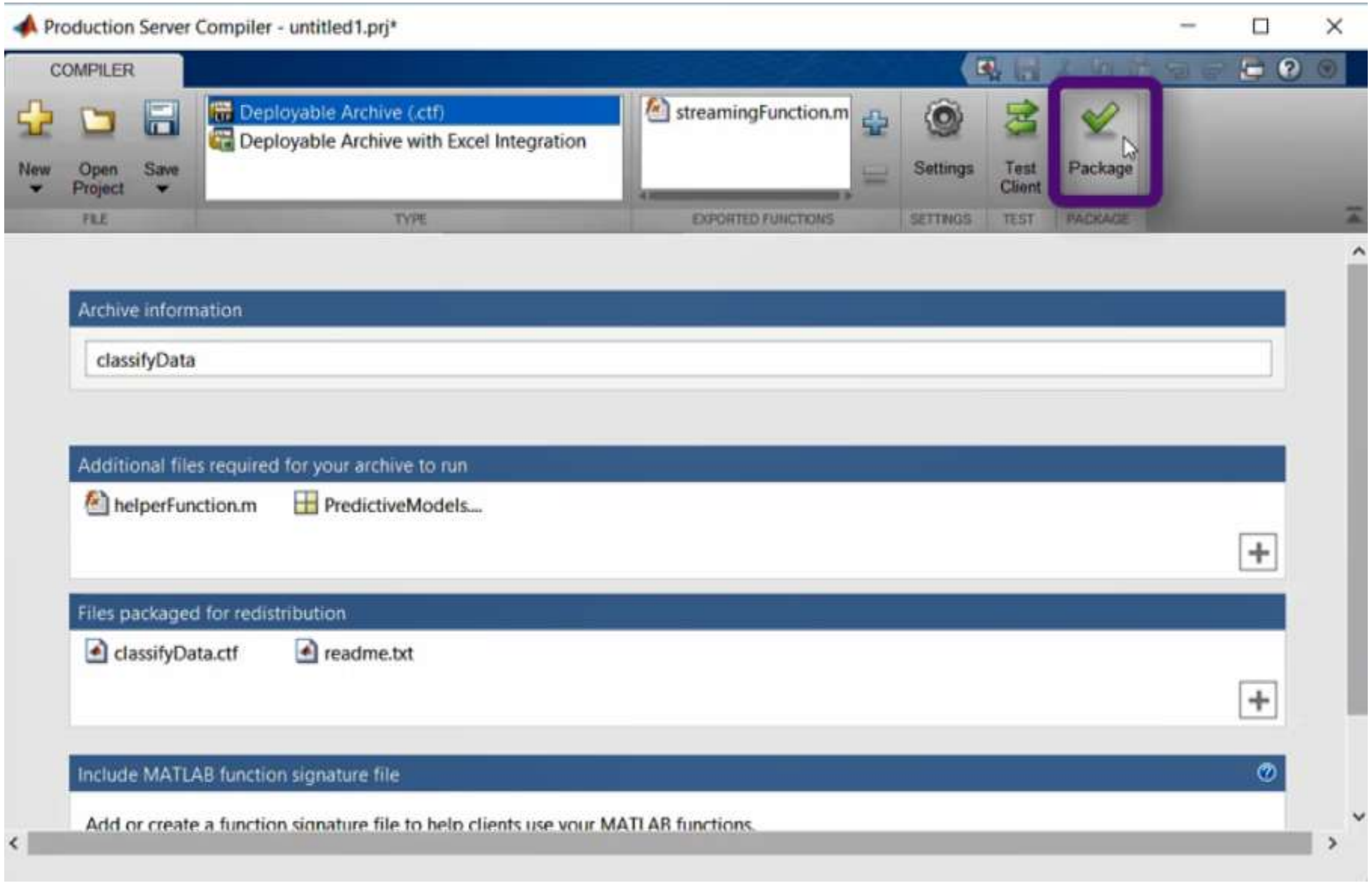


Source Control

Process Engineer

4 Integrate with Production Systems

Package Stream Processing Function



System Architect

4 Integrate with Production Systems

Review System Requirements

- Requirements from the Process Engineer
  - Every millisecond, each pump generates a time-stamped record of flow, pressure, and current
  - Model expects 1 sec. window of data per pump
  - Initially, 1’s – 10’s of devices, but quickly scale to 100’s

Process Engineer

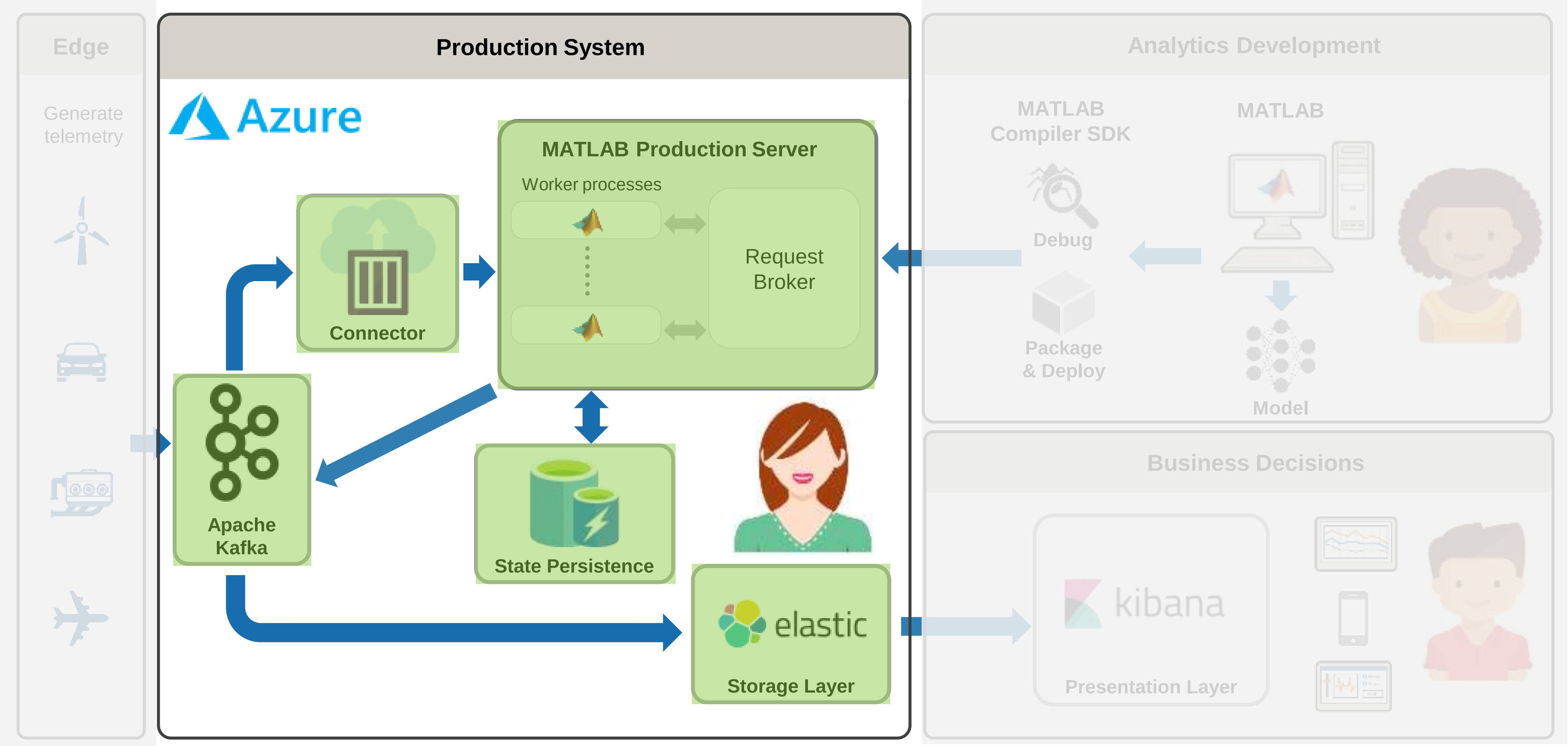
- Requirements from the Operator
  - Alerts when parameters drift outside the expected ranges
  - Continuous estimating of RUL for each pump

Operator

Integrate Analytics with Production Systems

System Architect

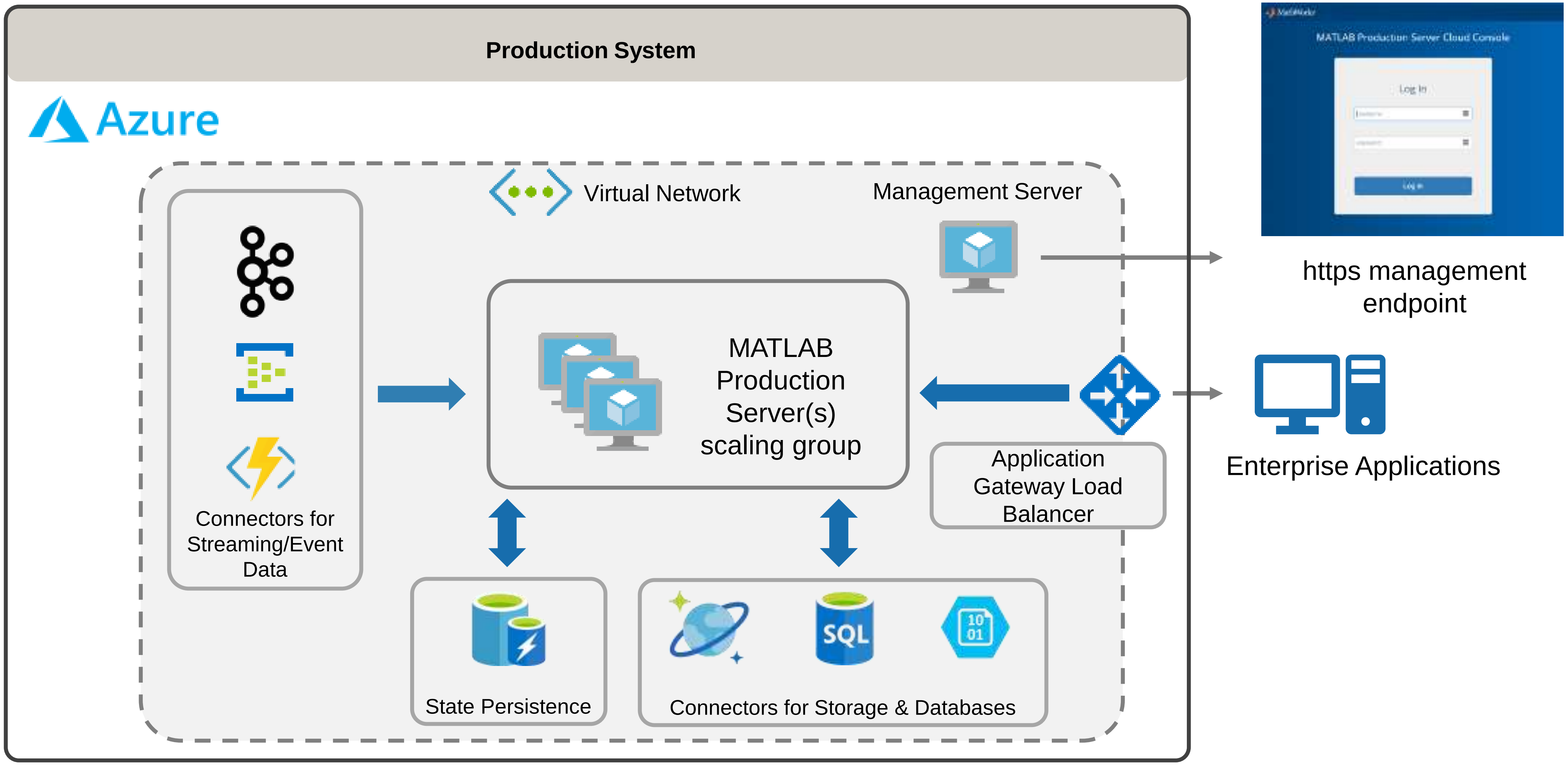
4 Integrate with Production Systems



System Architect

4 Integrate with Production Systems

MATLAB Production Server on Azure

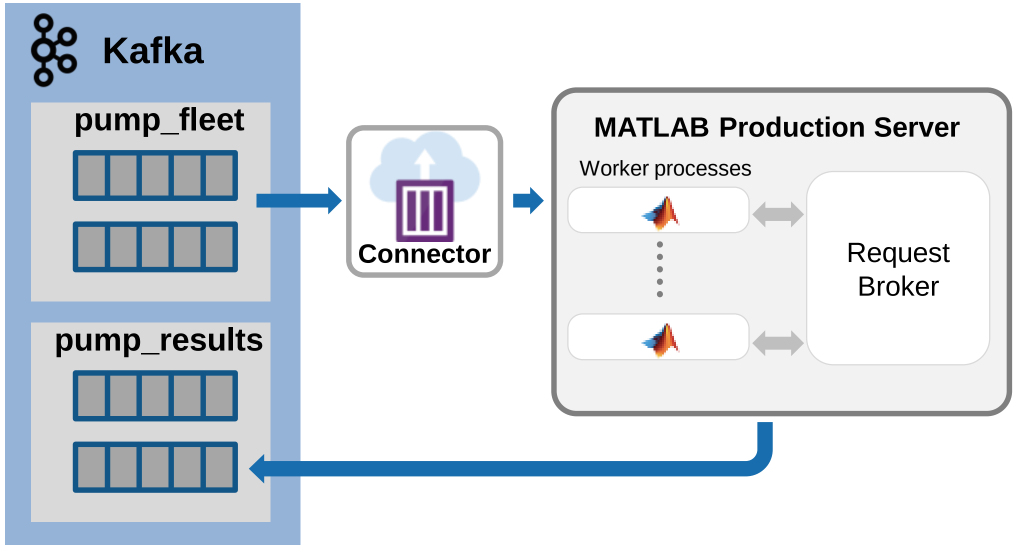


System Architect

4 Integrate with Production Systems

Connecting MATLAB Production Server to Kafka

- Connector feeds single Kafka topic to a MATLAB function
- Publisher library for MATLAB for writing to a results stream
- Connector Features:
  - Deploy as a micro-service with Docker
  - Drive everything through config
  - Group data into time windows and pass to MATLAB as a timetable
  - Use Kafka's check-pointing (i.e. at-least-once)



System Architect

4 Integrate with Production Systems

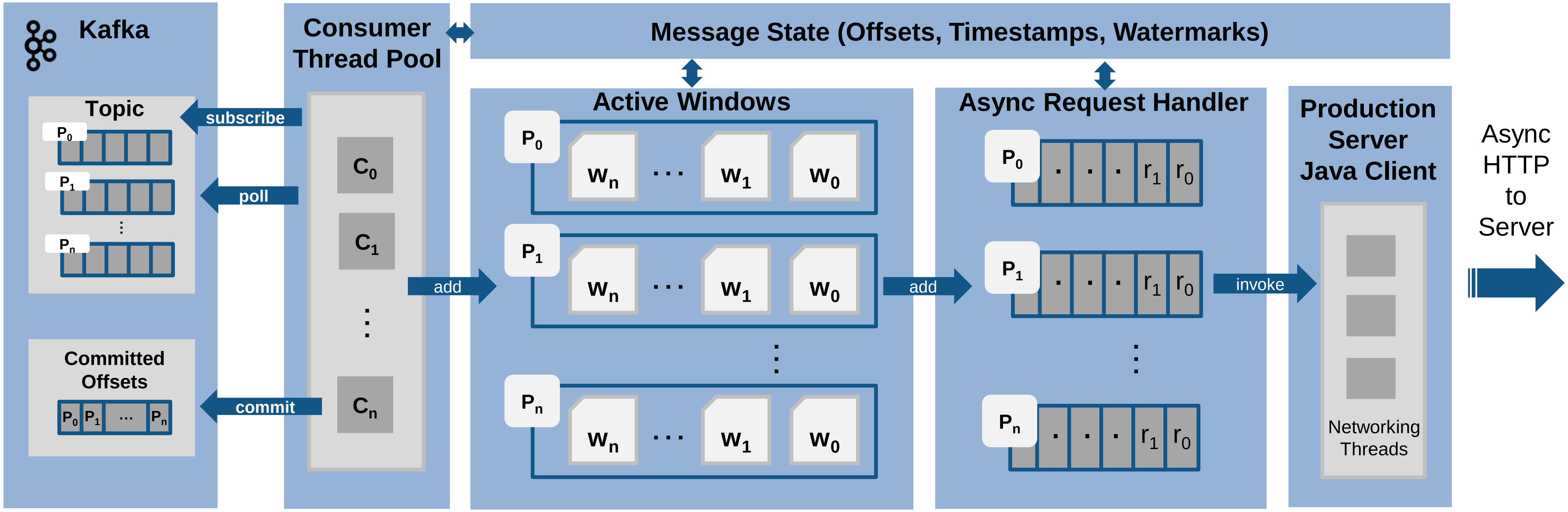
Messaging adapter for Production Server

- Bridges streaming data and Production Server Async Java Client
- Batches incoming messages and sends them via HTTP request/response
  - Time windows, event time processing, and out-of-order data
- Uses Asynchronous pipeline model with back-pressure
  - Kafka consumers are automatically paused when server is busy
- Supports sequential (stateful) and unordered (stateless) processing
  - Provide unique stream ID/topic/partition info for persistence layer
- Pass data as MATLAB timetables
- Partition aware – enables full exploitation of partition-based parallelism

System Architect

4 Integrate with Production Systems

Kafka connector architecture



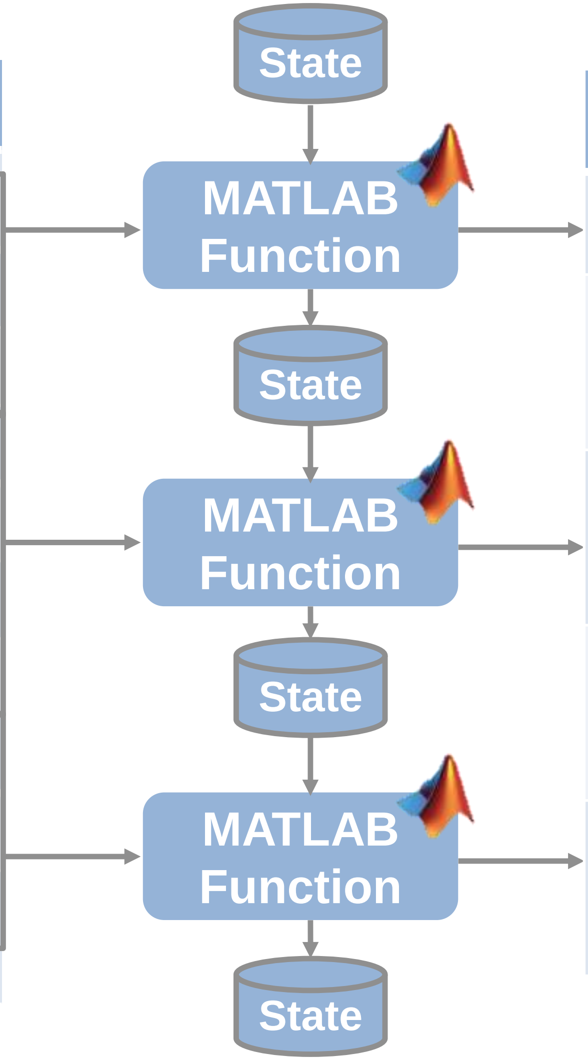
System Architect

4 Integrate with Production Systems

Streaming data is treated as an unbounded Timetable

Input Stream

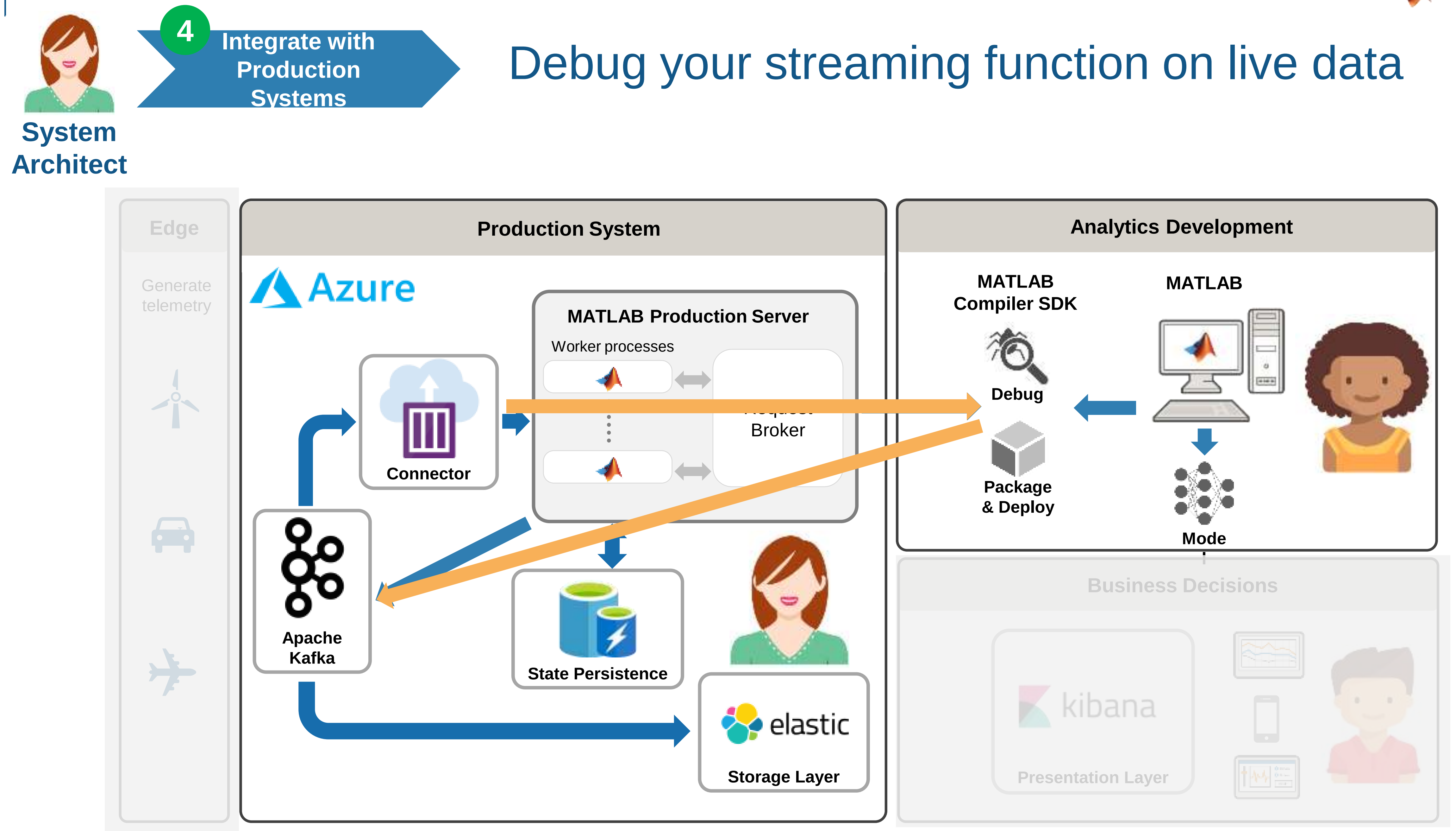
Event Time	Pump Id	Flow	Pressure	Current
18:01:10	Pump1	1975	100	110
18:10:30	Pump3	2000	109	115
18:05:20	Pump1	1980	105	105
18:10:45	Pump2	2100	110	100
18:30:10	Pump4	2000	100	110
18:35:20	Pump4	1960	103	105
18:20:40	Pump3	1970	112	104
18:39:30	Pump4	2100	105	110
18:30:00	Pump3	1980	110	113
18:30:50	Pump3	2000	100	110
…	…	…	…	…

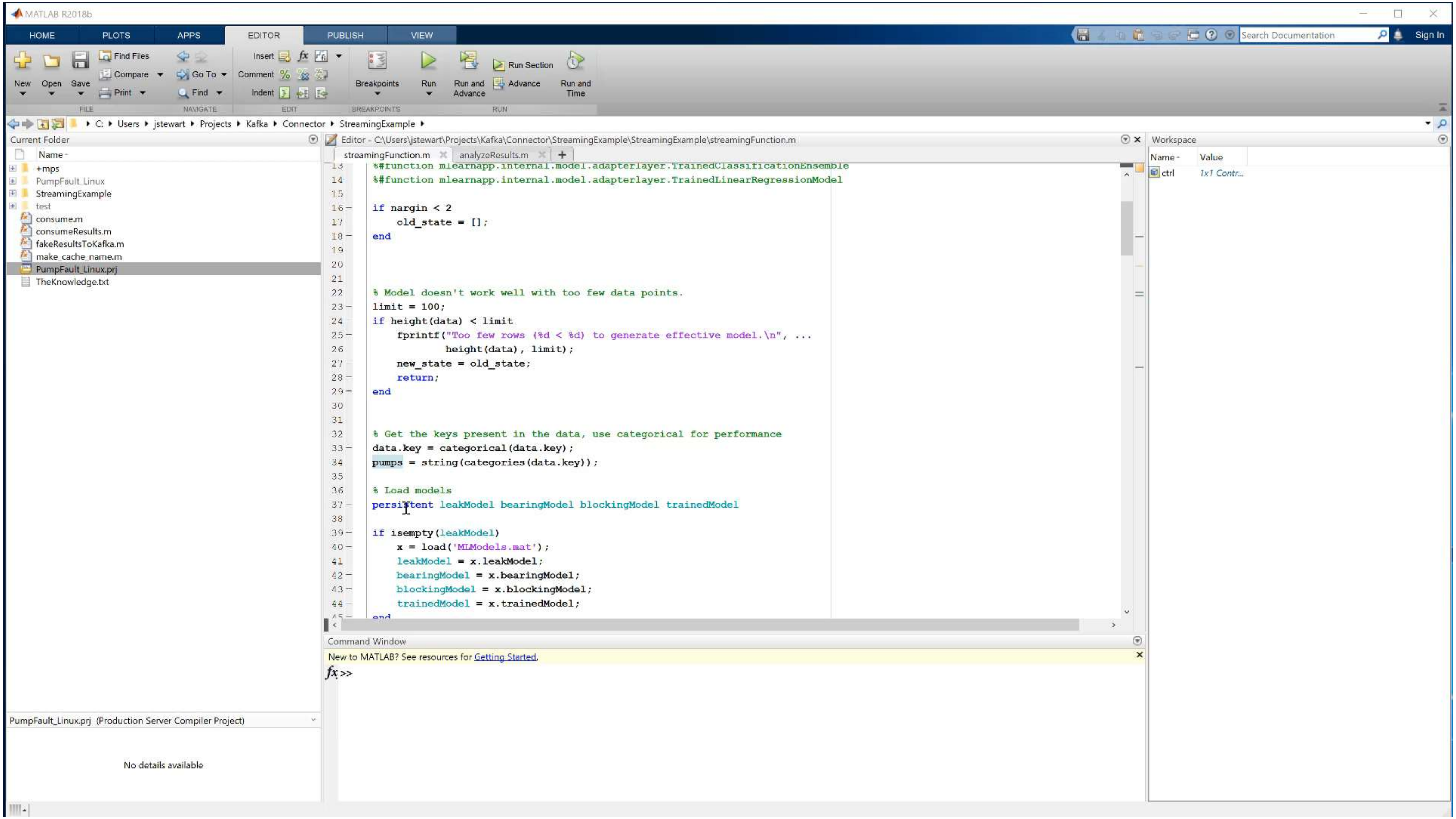


Output Stream

Time window		Pump Id	Bearing Friction
…		…	…
18:00:00	18:10:00	Pump1	5
		Pump3	…
		Pump4	…
18:10:00	18:20:00	Pump2	7
		Pump3	3
		Pump4	…
18:20:00	18:30:00	Pump1	…
		Pump3	4
		Pump4	…
18:30:00	18:40:00	Pump5	…
		Pump3	5
		Pump4	8

Debug your streaming function on live data



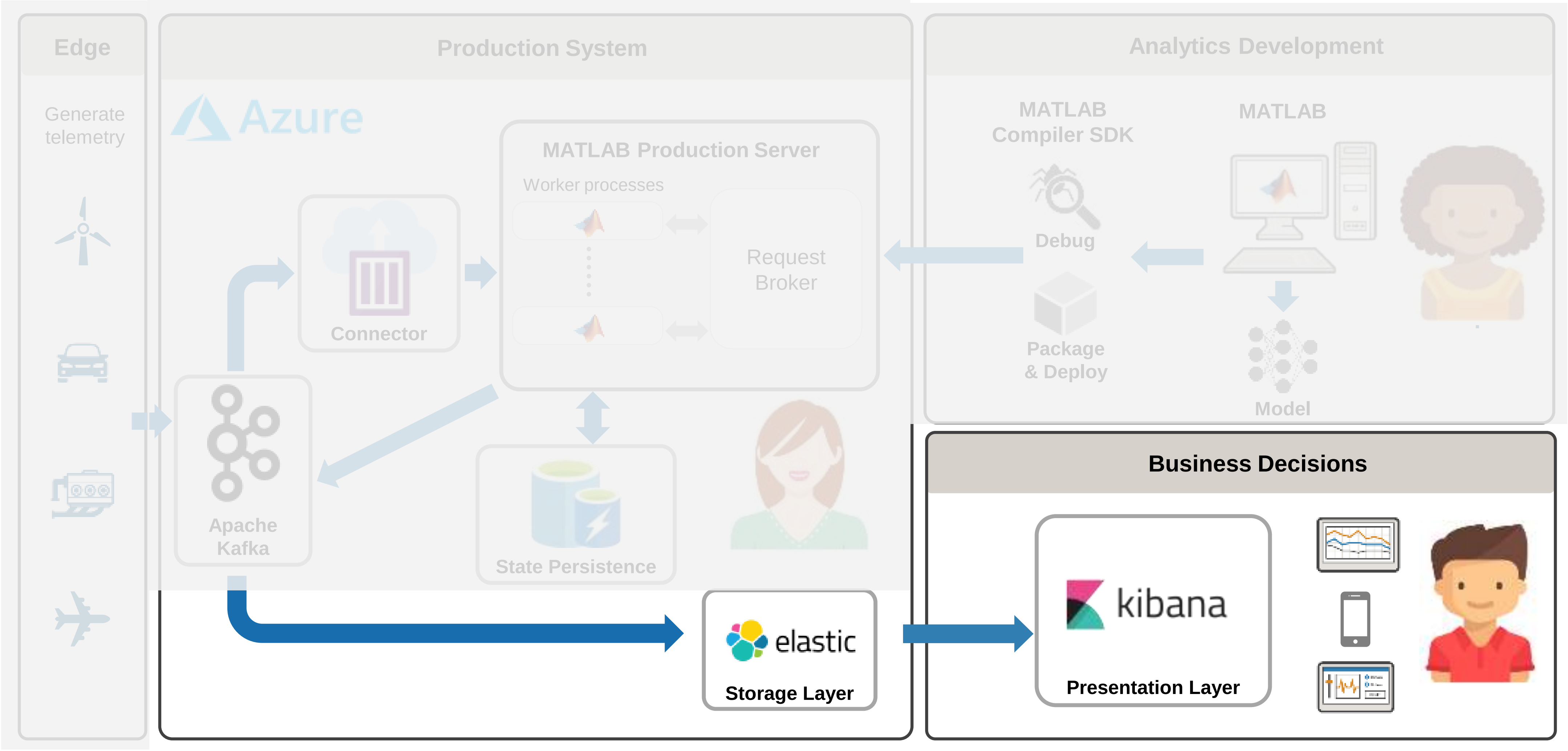


MATLAB R2018b
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
Sign In
New
Open
Save
Find Files
Compare
Print
FILE
Go To
Find
NAVIGATE
Insert
Comment
Indent
EDIT
Breakpoints
BREAKPOINTS
Run
Run and Advance
Run Section
Advance
Run and Time
RUN
C: Users jstewart Projects Kafka Connector StreamingExample
Current Folder
Name
+mps
PumpFault_Linux
StreamingExample
test
consume.m
consumeResults.m
fakeResultsToKafka.m
make_cache_name.m
PumpFault_Linux.prj
TheKnowledge.txt
Editor - C:\Users\jstewart\Projects\Kafka\Connector\StreamingExample\StreamingExample\streamingFunction.m
streamingFunction.m
analyzeResults.m
%#function mlearnapp.internal.model.adapterlayer.TrainedClassificationEnsemble
%#function mlearnapp.internal.model.adapterlayer.TrainedLinearRegressionModel
if nargin < 2
old_state = [];
end
% Model doesn't work well with too few data points.
limit = 100;
if height(data) < limit
fprintf("Too few rows (%d < %d) to generate effective model.\n", ...
height(data), limit);
new_state = old_state;
return;
end
% Get the keys present in the data, use categorical for performance
data.key = categorical(data.key);
pumps = string(categories(data.key));
% Load models
persistent leakModel bearingModel blockingModel trainedModel
if isempty(leakModel)
x = load('MLModels.mat');
leakModel = x.leakModel;
bearingModel = x.bearingModel;
blockingModel = x.blockingModel;
trainedModel = x.trainedModel;
end
Command Window
New to MATLAB? See resources for Getting Started.
fx >>
Workspace
Name
Value
ctrl
1x1 Contr...
PumpFault_Linux.prj (Production Server Compiler Project)
No details available

System Architect

4 Integrate with Production Systems

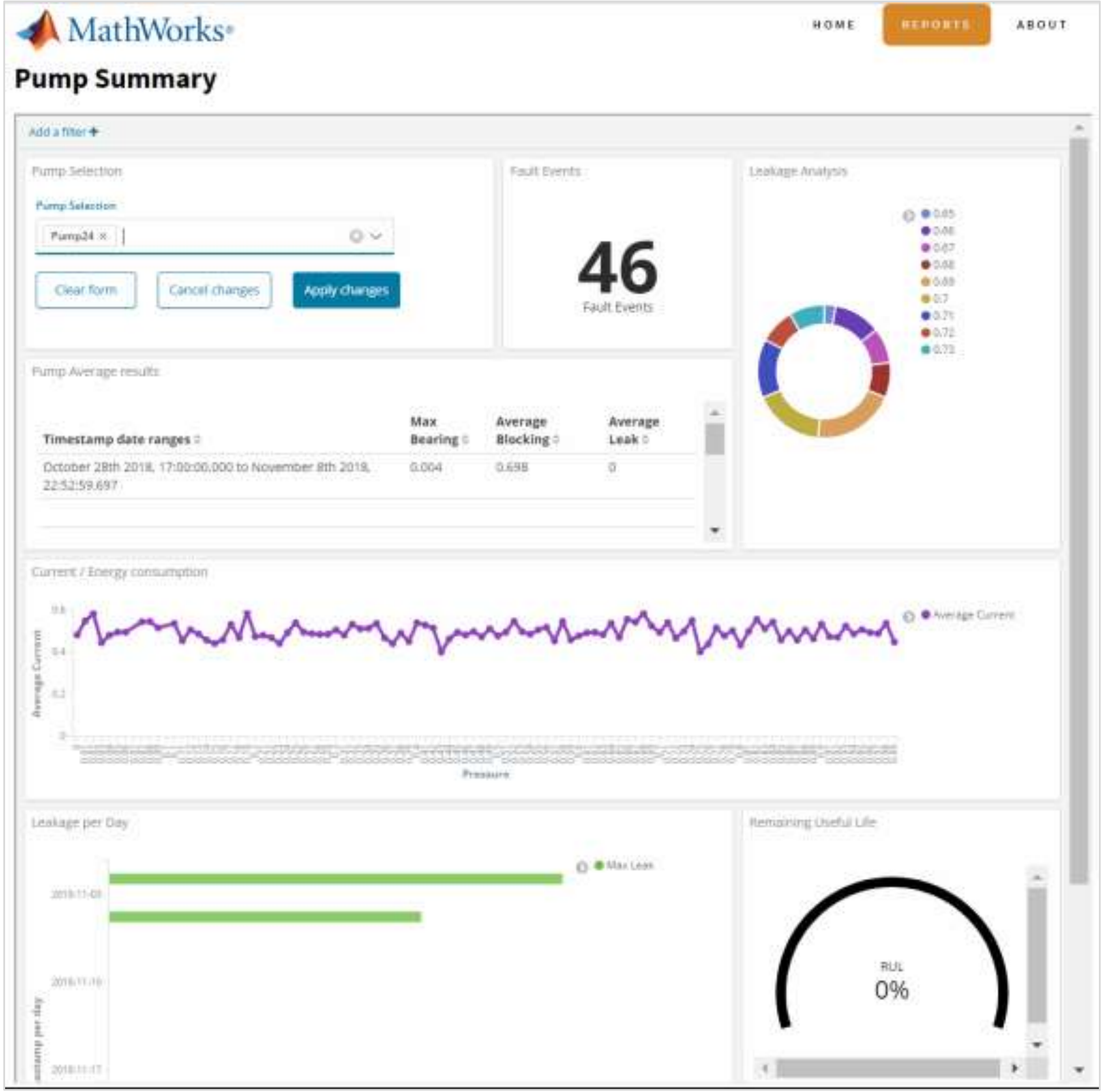
Complete your application



5 Visualize Results

Plant Operator

Complete Your Application



Team Retrospective

- Completed demo of full system in 3 week sprint
- Successfully used digital twin to generate faults and train models
- Fast prototyping of physical and AI models with MATLAB and Simulink. Easy integration with OSS
- Cloud platform enabled faster IT setup
- Next steps:
  - Make model adjustments
  - Test against real pump
  - Customize dashboard for Operator’s needs

Resources to learn and get started

- [GitHub: MathWorks Reference Architectures]
- [Working with Enterprise IT Systems]
- [Data Analytics with MATLAB]
- [Simulink]