Data Collection for Edge AI / Tiny ML with Sensors

Overview

Reality AI software from Renesas provides solution suites and tools for R&D engineers who build products and internal solutions using sensors. Working with accelerometers, vibration, sound, electrical (current/voltage/capacitance), radar, RF, proprietary sensors, and other types of sensor data, Reality AI software identifies signatures of events and conditions, correlates changes in signatures to target variables, and detects anomalies. Since data collection and preparation is both the most costly part of any machine learning project, and also the place where most failed projects go wrong, Reality AI software contains functionality to keep data collection on track, to assist with its pre-ML processing, and to get the most out of it using synthetic augmentation techniques. This white paper covers the approach we recommend for data collection planning, execution, and post-collection processing.

Data Collection Planning

Data collection is the most time consuming, most expensive part of any machine learning development effort — and it is also the part where the seeds of project failure are most easily set. Planning for data collection is the stage where problems can be foreseen and avoided without extra expense. Therefore we always advise customers to create a comprehensive data collection plan before beginning.

Avoid the biggest pitfalls in data collection with good planning

- Inadequate data coverage
- Insufficient ground truth
- Unreliable or inappropriate instrumentation
- Inconsistent collection protocol

So what are the elements of a good data collection plan?

DATA COVERAGE MATRIX

The Data Coverage Matrix lays out the breadth of data that would need to be collected if examples of every possible combination of target, environmental or background condition, and equipment-related variation were to be covered. It is rarely necessary to collect data from each possible combination, but understanding these sources of variation is key to creating an adequate plan.

INSTRUMENTATION PLAN

The Instrumentation Plan covers hardware and communication aspects of the data collection effort. It should also layout explicitly how ground truth will be collected. For machine learning development, it is always important to have some way of knowing — as definitively and precisely as possible — what the correct prediction result should be from the data collected (the “ground truth”).

DATA COLLECTION PROTOCOL

The Data Collection Protocol spells out the specific steps that will be followed during data collection. How the apparatus will be set up, how trials will be run, how results will be recorded, how data will be manipulated and stored. The protocol should be as detailed as possible and should contain checklists that can be followed by those doing the actual collection. Consistently following a comprehensive and carefully constructed protocol is the best way to ensure that the data is usable and that machine learning results are not "tricked" by artifacts in the data.

DATA COVERAGE MATRIX

Data coverage refers to the distribution of data across target classes or values, as well as across other variables that are significant contributors to variation. Target classes are just the things you are looking to detect or predict. For example, if you're building a model that will detect whether a machine is behaving normally or is exhibiting a particular kind of fault, then "Normal" and "Fault" are the target classes.

But other variables may also be important. If you have good reason to expect that "Normal" and "Fault" will look different when the machine is operating in different modes, those modes are contributors to variation and should be tracked as metadata variables. Different versions of equipment or different installation types, for example, are almost always significant contributors to variation.

The purpose of coverage planning is to make sure you collect enough data to capture and overcome the variation inherent in what you're trying to measure. To do that you'll need a statistically significant number of observations taken across a reasonable number of combinations of different target classes and metadata variables (contributors to variation).

So in our example above, we have two target classes and a couple of different modes:

	OPERATING MODE 1	OPERATING MODE 2	OPERATING MODE 3
Normal	# of actual or planned trials/observations		
Failure			

Example of a simple data coverage matrix

This is a data coverage matrix and allows us to plan for the data we need to collect. In the beginning, since we don't yet know whether it's possible to detect fault conditions in different types of equipment with the same machine learning model, we probably want a separate matrix for each equipment type – making this more of a data cube than a 2x3 matrix.

A fully articulated data coverage matrix would cover all of the material variations in target, background environment and equipment that we believe would make for differences in the data, or that we need to test to have confidence in the outcome. Usually, these have too many dimensions to list in table form, so instead, we just list each of the dimensions and the categories or ranges within each. These can then be used to construct a table or database for actual tracking.

Target: Failure Mode	**Equipment: Model**	**Size**	**Age**	**Environment: Temp**	**Humidity**
Normal Motor Rotor Capacitor Bearing Coil	QV22 QA22 QV23 SA22 SA23	15 20 22 25	New to 15yrs	-40°F to 110°F	20% to 80%
6	5	4	4 (5yr bucket)	15 (10°F bucket)	7 (10% bucket)

Example of a more complex data coverage matrix

If we bucket Equipment Age by five year increments, temperature by increments of 10°F, and humidity by 10%, this coverage matrix has 6 x 5 x 4 x 4 x 15 x 7 = 50,400 cells. Ideally, each of these cells would be populated with at least 25 independent observations or trials — say 25 machines in each combination of model, size and age (i.e. 25 x 5 x 4 x 4 = 2,000 machines) that could be observed a statistically significant number of times in each failure mode, in each environmental condition.

Why 25 independent observation sets per cell? It comes from the Central Limit Theorem and the Law of Large Numbers in statistics and is usually cited as a rule of thumb for achieving statistical significance. In efforts like those we are discussing here, what we really need are enough independent observations to be able to account for noise and variation in the data from sources that we cannot control (or may not even be aware of) and that the model will need to learn to ignore. You may need more, or fewer, depending on the amount of variation in target, background, and equipment.

But the real world is not ideal, and the ideal population of the Data Coverage Matrix is not always possible.

How Much Data Do You Really Need? Iterative Data Coverage

The purpose of the data coverage matrix is to help understand how much data you really need. How much is ideal, and how much will be the bare minimum? The real answer to “How much data do I need?” is “You need enough.”

There is no one-size-fits-all answer to the question. It very much depends on the specifics of your problem and its difficulty, and the amount of variation in targets, backgrounds, and equipment.

Therefore, we always recommend collecting data iteratively and using empirical results to guide cost-effective data collection. You typically do not need to cover every single cell in the coverage matrix, but a smart collection plan that fills one section of it comprehensively and then tests the degree to which more data from other sections is required can get you to effective, reliable results with a minimum of cost.

It works like this:

In our example above of a “More Complex Data Coverage Matrix”, we start with a “Proof-of-Concept” test. Perhaps we pick one model and size, get 4 or 5 examples at different ages into a test facility where we can induce the target failure modes. If we can control the environmental variables in our test facility, we collect data in 4-5 different environment types, but if we can’t we just record what the environmental variables are and do our best to keep them constant through the data collection.

Target: Failure Mode	**Equipment: Model**	**Size**	**Age**	**Environment: Temperature**	**Humidity**
Normal Motor Rotor Capacitor Bearing Coil	QV23	15	New to 15yrs	60 70 80	30% 60%
6	1	1	4	3	2

Example of the data coverage matrix for the "proof-of-concept"

We collect data to fully populate the 6 x 1 x 1 x 4 x 3 x 2 = 144 cells of this matrix and use the data to build a version 1.0 of our prediction model. In evaluating the results, we would look carefully at the prediction accuracy for each failure mode, and the false positive rate on Normal, but we would also look to see how that accuracy varies across the other metadata categories. Did it do equally well across age, temperature, and humidity? If not, those might be categories that require additional coverage in later stages.

We can also do some selective holdout testing to see which categories are important. Perhaps we train our model on data from 4 of the 5 machines, holding out data from the oldest. Does our model predict that holdout data well? If so, you have good reason to believe that you will generalize well across age and can deprioritize that dimension in the later collection. If not, the opposite.

Similarly, we could run a test of a sixth machine, either the same model in a different size or a different model in the same size. Collect data, and use our ML model to predict. How did it work? If accuracy was low, add some of the data to the training set, retrain, and test again (we recommend using K-fold validation for this kind of test). If accuracy improves, that tells you that additional data coverage will likely get you good results — you just need coverage of different models or sizes.

Steps for iterative data coverage

1. Pick a section of the Data Coverage Matrix for an initial focus
2. Collect data to cover that section as comprehensively as possible
3. Build an initial version of the model based on initial data
4. Examine variation in accuracy across dimensions of the matrix and prioritize additional collection accordingly
5. Test generalization to new sections of the matrix:
   a. Collect data representing a new category
   b. Test the previous version of the model against new data and evaluate accuracy.
   c. If accuracy is high, deprioritize additional data collection.
   d. If accuracy is disappointing, incorporate new data into the training set and test again. If accuracy improves, prioritize additional data collection for this region. If not, investigate differences in data more closely.
6. Continue until all regions of the Data Coverage Matrix have been explored and tested

So how much data will be enough? As a general rule of thumb, if you follow the Iterative Data Coverage method outlined above, you will likely wind up with full population (data from at least 25 independent trials) in around three of cells in the Data Coverage Matrix, with enough additional data collected for testing that no row or column is completely empty.

INSTRUMENTATION PLANNING

The second main section of the Data Collection plan covers instrumentation and hardware. The instrumentation plan should cover what sensors will be used, how they will be sampled, where and how they should be mounted, what control apparatus will be required, and how data will be stored and forwarded to the machine learning environment for processing and use. If there are environmental considerations (high temperatures, hazardous environments, etc.) these should be highlighted and special requirements they put on the equipment specified.

In early-stage Proofs-of-Concept, shortcuts make sense. Dev boards with sensors pre-mounted or pre-integrated can make a lot of sense and help you collect data quickly and inexpensively. But they have their limitations.

RECOMMENDATIONS FOR INSTRUMENTATION PLANNING

- Over-instrumentation
- Collect rich data
- Incorporate ground truth

Over-Instrumentation

Once past the initial PoC, we generally recommend over-instrumentation for the first serious product-development data collections, mainly to collect data that can help explore the interaction between component choices and machine learning accuracy.

During this stage, we suggest using more sensors than you actually intend to put in the final device solution. If there are different candidates for the accelerometer or mic placement, for example, place a sensor in each potential location and collect simultaneously from all of them. Similarly, use high-fidelity sensors for those early collections, sampled at the highest possible rate. Software like Reality AI Tools® can then explore the data to select the optimum set of data channels, and can examine the relationship between solution accuracy and component sample rate, bit depth, and sensitivity — and do it much more quickly and cheaply than repeating data collection with different sensor choices.

Rich Data

We always recommend collecting data at the most detailed level possible — particularly in the early stages of development. Many engineers working with vibration data, for example, will collect only traditional RMS energy or peak-to-peak measures. Perhaps they'll collect FFT results for a set of frequency bands they expect to be important. But these data, if that's all that is collected and retained, discards much of the information necessary to make the judgments that you most want your machine learning algorithm to make.

RMS can certainly help identify some conditions, and an FFT run with relatively high frequency and time resolution can be quite valuable. But high-level descriptive statistics lose almost all the essential signature information you need to find granular events and conditions. And as this excellent example from another domain shows, descriptive statistics often don't let you see the most interesting things:

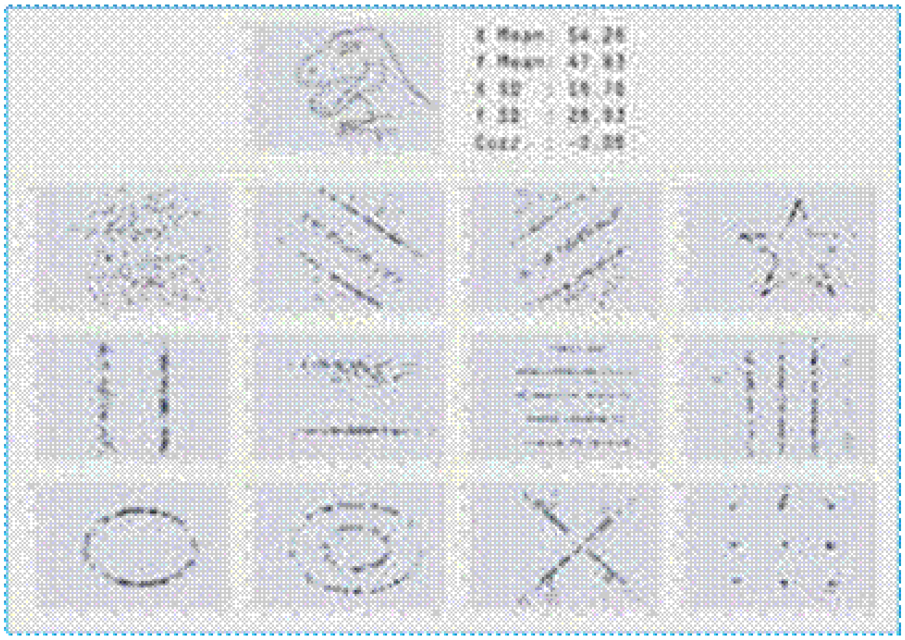
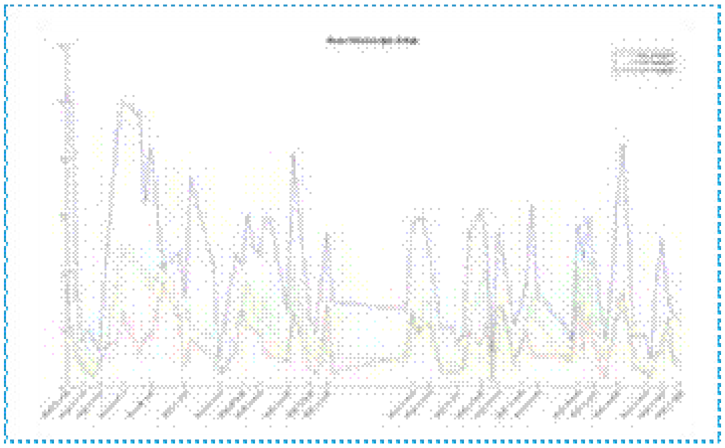


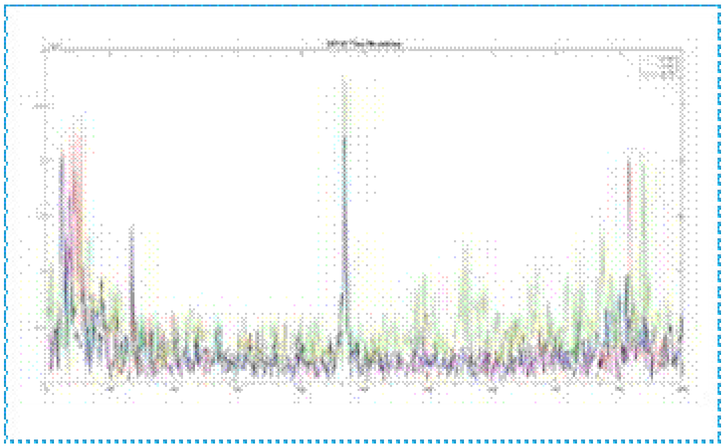
Illustration of why aggregate statistics are never enough. All of these plots have the same X and Y means, the same X and Y standard deviations, and the same X:Y correlation. Without the ability to consider all of the source detail, your algorithm would never see any of these patterns in your data.

Instead of relying on aggregate features defined by the traditional signal processing tools, Reality AI discovers features directly from the raw sensor signal data in all of its inherent noise and complexity. This approach allows for consideration of information in both time and frequency domains and allows for detection of conditions with much more subtle signatures.

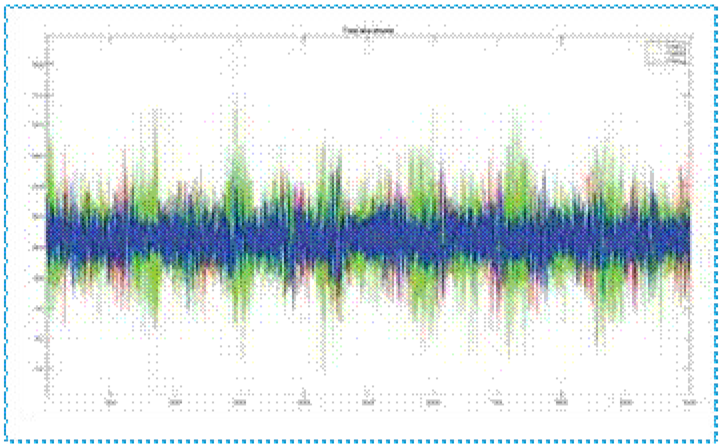
Example showing a time series of data from an accelerometer attached to a machine in a manufacturing facility. X, Y, and Z components of the acceleration vector are averaged over one second. There is very little information in this data. This data was provided from an actual customer implementation in a manufacturing facility and is basically useless for anomaly detection, condition monitoring, or predictive maintenance.



Example showing vibration data pre-processed using FFT. The X-axis is frequency and the Y-axis is intensity. Peaks at multiples of the equipment's base rotation frequency give important information. FFT data is most useful for rotating equipment and can be good for many applications, but it discards the time-domain. It only shows a single snapshot in time – this entire chart is an expansion of a single data point from the previous example.



Raw time-waveform as sampled directly from the accelerometer. This is information-dense and is the raw data from which both previous examples were computed. Here we have both frequency and time information, including transients and phase. This kind of data is very difficult for human analysts to use directly, but is the best for algorithmic tools like Reality AI, who's feature discovery process zeros in on exactly those parts of the signal – and only those parts of the signal – that are relevant for the specific problem.



When deciding what data to collect and retain, always opt for the most information-rich data available, reducing only as a final tuning and optimizing step in the final stages of development.

Ground Truth

Perhaps the most frequently overlooked aspect of data collection — yet also the most important — is the question of how to collect ground truth. Ground truth refers to the real outcome that you are trying to predict: for example, was the machine actually normal, or was there really an error condition?

Ground truth is important both as a basis for training and as a basis for judging accuracy. If you don't know what any of the data really was, how will you ever know whether your model is predicting accurately?

Your instrumentation plan should include a means of determining and collecting ground truth with as much accuracy and as much time precision as possible. In some cases, this may mean deploying a second set of sensors that enable the determination. In other cases, it may mean capturing maintenance logs, or recording observer perceptions. But whatever it is, make sure it's covered by your plan.

Approaches for Collecting Ground Truth

EXPERIMENTAL DESIGN
set up conditions to create specific ground truth values in advance and collect data in blocks

SIMULTANEOUS MEASUREMENT
add additional sensors that allow complete determination of ground truth state so that signals can be correlated

DIRECT OBSERVATION
operators or data collectors record events or states

SERVICE LOGS
events determined after observation by correlating to other documentation

AFTER-THE-FACT INVESTIGATION
Most often used in anomaly detection when other sources of ground truth are unavailable. Anomalous readings are investigated to determine what actually happened.

Even if you plan to use an anomaly detection approach, where data will not (or can not) be systematically labeled with ground truth values, you will need a plan for determining ground truth. When an anomaly is detected, you will need some way to determine whether or not it is valid — without that kind of feedback you will never be able to determine the model's accuracy, and there will be no way for it to improve.

Data Collection Protocol

The Data Collection Protocol is the step-by-step instructions for performing the data collection. These steps should be laid out as specifically as possible to minimize inconsistencies in the way data is collected.

Things to Cover in Data Collection Protocol

- Installing instrumentation
- Site preparation and recording of metadata
- Initiation of data collection
  - How to initiate or induce conditions
  - Stability or equilibrium requirements for reliable data

- Recording data and ground truth
- Ending data collection
  - Minimum length of observation or time-in-class
  - How to terminate conditions
- Data annotation - how observations should be logged and files should be named
- Data storage and transmission - how and how often data should be recorded, harvested, and loaded into staging area for further processing

Collection Monitoring And Data Readiness

Another common issue we see in data collection efforts is failure to monitor data collection and spot problems early. Issues with instrumentation, file capture, and other technical snags happen often, and if left undetected until the data analysis step once the collection is complete, risk needing to discard entire data sets. More than once, we have seen projects need to start over from scratch due to minor instrumentation or processing issues that, though easily correctable, rendered data unusable.

To address these issues in Reality AI Tools, we have added functionality for automated Data Readiness Assessment.

Automated Data Readiness Assessment

Every time a new source data file is added to a project in Reality AI Tools, the system reads the file, parses it, and looks for issues. If any are found, they generate immediate user alerts. The system also generates statistics that can be used to identify trends and report on progress.

Reality AI Tools data readiness reports on:

FILE CONSISTENCY

File size, file type, number of data columns, sample rate, and other file-level demographics. Inconsistencies in these categories can indicate systemic problems in data collection or in post-collection processing.

DATA QUALITY

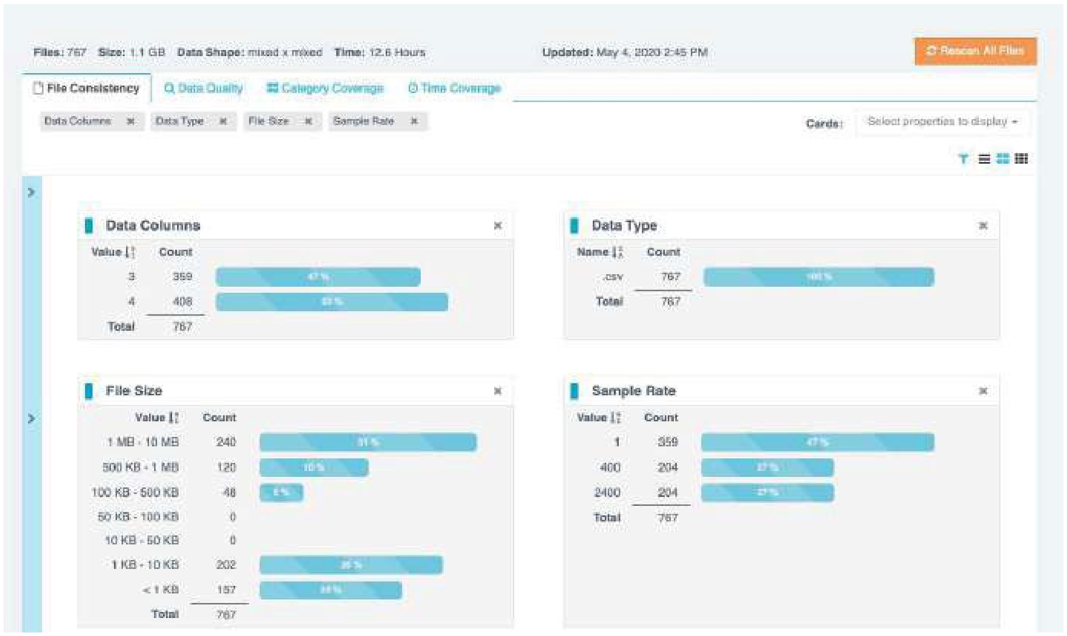
Issues that prevent a file from being read, such as corrupted files, invalid characters, blank rows, and missing data elements. It also looks for suspicious data, such as blocks of data with repeating patterns or zeros.

DATA COVERAGE

Cumulative statistics showing how much data has been collected for key target and metadata categories - can be used to track progress vs the Data Coverage Matrix in a Data Collection Plan.

TIME COVERAGE

Cumulative statistics describing data in terms of time-in-class for key target and metadata categories. This is particularly useful for looking at data coverage for real-time streaming applications when the length of the decision window or smoothing methods are under evaluation.



Files: 767 **Size:** 1.1 GB **Data Shape:** mixed x mixed **Time:** 12.6 Hours **Updated:** May 4, 2020 2:45 PM

Data Columns

Value	Count
3	359
4	408
Total	767

Data Type

Name	Count
.csv	767
Total	767

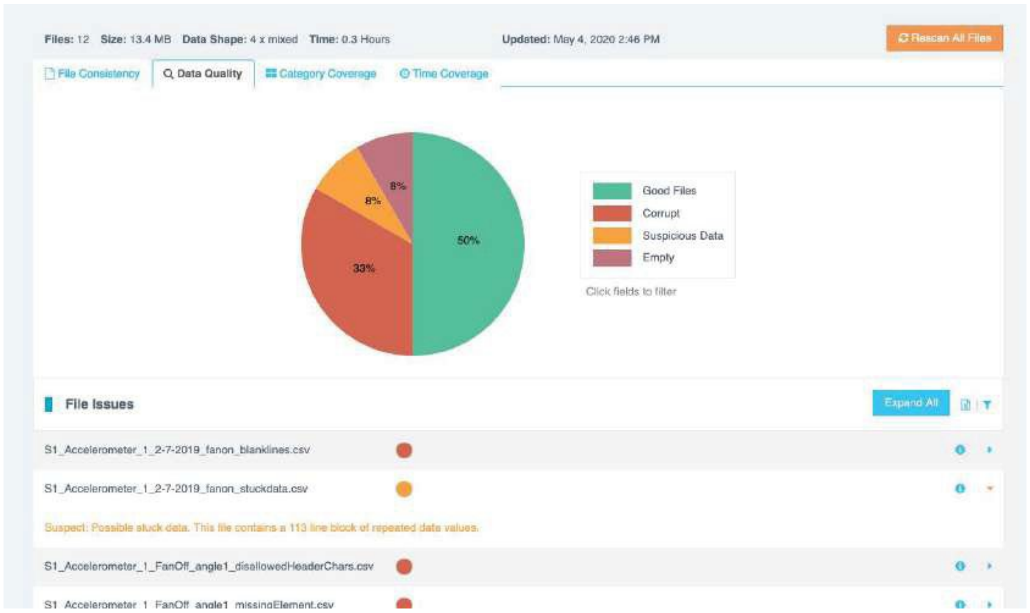
File Size

Value	Count
1 MB - 10 MB	240
500 KB - 1 MB	120
100 KB - 500 KB	48
50 KB - 100 KB	0
10 KB - 50 KB	0
1 KB - 10 KB	202
< 1 KB	157
Total	767

Sample Rate

Value	Count
1	359
400	204
2400	204
Total	767

Screenshot of a File Consistency report from Reality AI Tools, run automatically whenever a file is loaded. Shows file-level demographics. Inconsistencies or unexpected values should be investigated immediately - they may indicate problems in data collection or post-collection processing.



Files: 12 **Size:** 13.4 MB **Data Shape:** 4 x mixed **Time:** 0.3 Hours **Updated:** May 4, 2020 2:46 PM

File Issues

S1_Accelerometer_1_2-7-2019_fanon_blanklines.csv ●

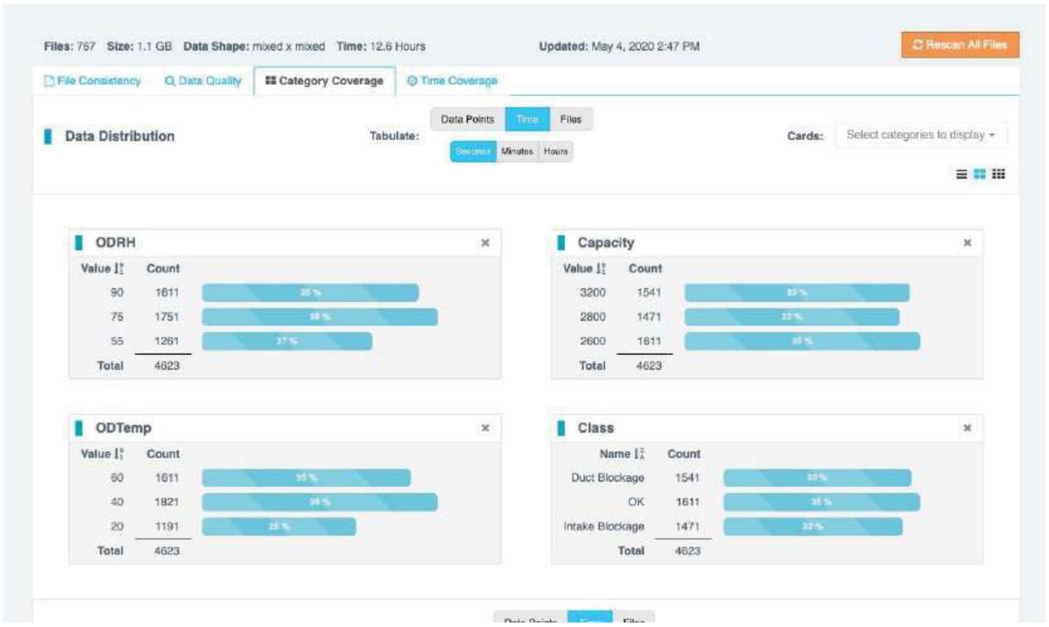
S1_Accelerometer_1_2-7-2019_fanon_stuckdata.csv ●

Suspect: Possible stuck data. This file contains a 113 line block of repeated data values.

S1_Accelerometer_1_FanOff_angle1_disallowedHeaderChars.csv ●

S1_Accelerometer_1_FanOff_angle1_missingElement.csv ●

Screenshot of a Data Quality report in Reality AI Tools, run automatically whenever a file is loaded. Shows issues spotted when the file is read that either prevent use entirely (Corrupt or Empty), or that indicate a potential problem with instrumentation (repeated or frozen data, zeros, etc.)



Files: 767 **Size:** 1.1 GB **Data Shape:** mixed x mixed **Time:** 12.6 Hours **Updated:** May 4, 2020 2:47 PM

Data Distribution

Tabulate: Time, Seconds

Cards:

ODRH

Value	Count	
90	1611	35 %
75	1751	38 %
55	1261	27 %
Total	4623	

Capacity

Value	Count	
3200	1541	33 %
2800	1471	32 %
2600	1611	35 %
Total	4623	

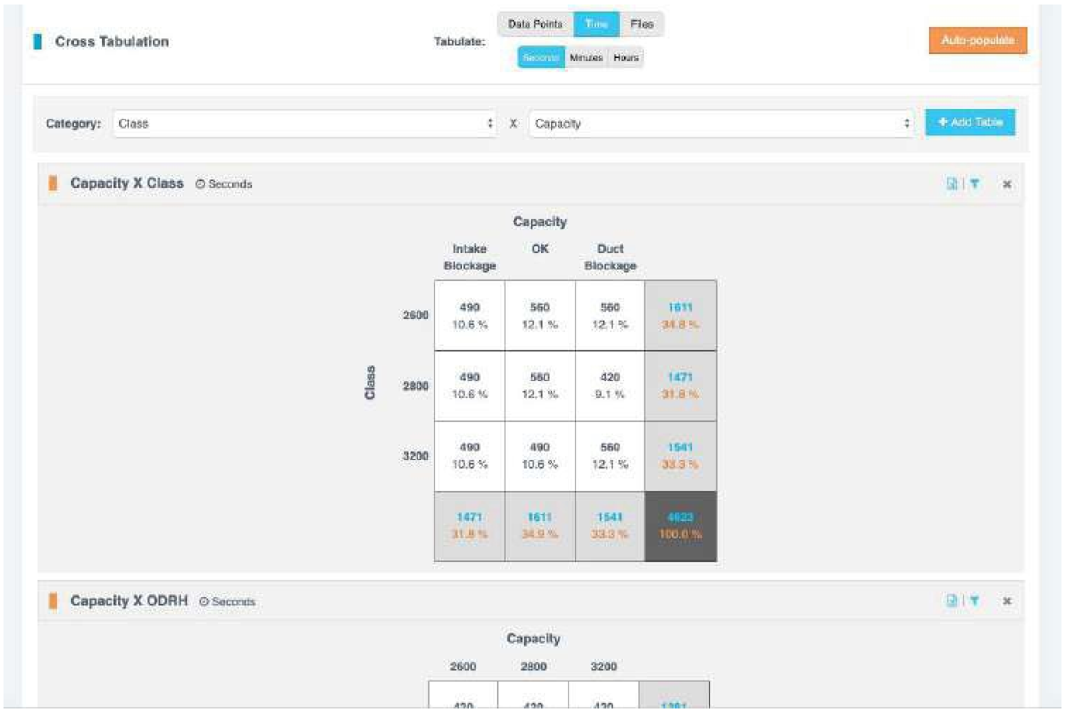
ODTemp

Value	Count	
60	1611	35 %
40	1821	39 %
20	1191	26 %
Total	4623	

Class

Name	Count	
Duct Blockage	1541	33 %
OK	1611	35 %
Intake Blockage	1471	32 %
Total	4623	

Screenshot of a Data Coverage report in Reality AI Tools, run automatically whenever a file is loaded. Shows amount of data collected for values of target and metadata variables, for tracking against a Data Coverage Matrix in a Data Collection Plan.



Cross Tabulation

Tabulate: Time, Seconds

Category: Class x Capacity

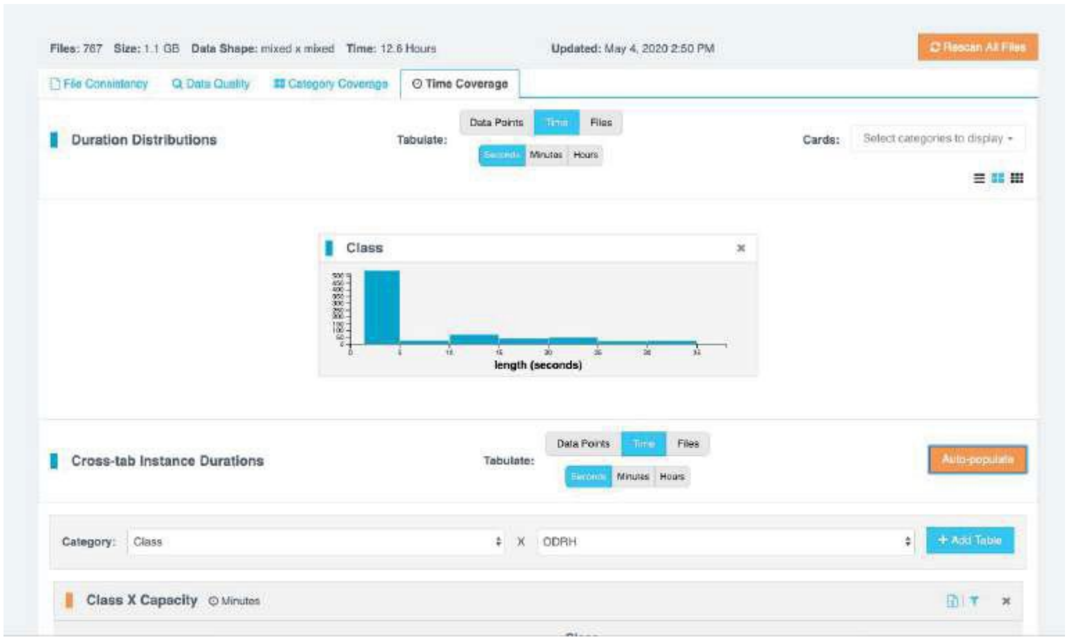
Capacity X Class ⏱ Seconds

Class	Capacity: Intake Blockage	OK	Duct Blockage	
2600	490 10.6 %	560 12.1 %	560 12.1 %	1611 34.8 %
2800	490 10.6 %	560 12.1 %	420 9.1 %	1471 31.8 %
3200	490 10.6 %	490 10.6 %	560 12.1 %	1541 33.3 %
	1471 31.8 %	1611 34.9 %	1541 33.3 %	4623 100.0 %

Capacity X ODRH ⏱ Seconds

	Capacity: 2600	2800	3200	
	420	420	420	1261

Screenshot of a Data Coverage cross-tab in Reality AI Tools to support analysis and monitoring of coverage as data is collected. Data Coverage statistics can also be downloaded for use in other reporting software as needed.

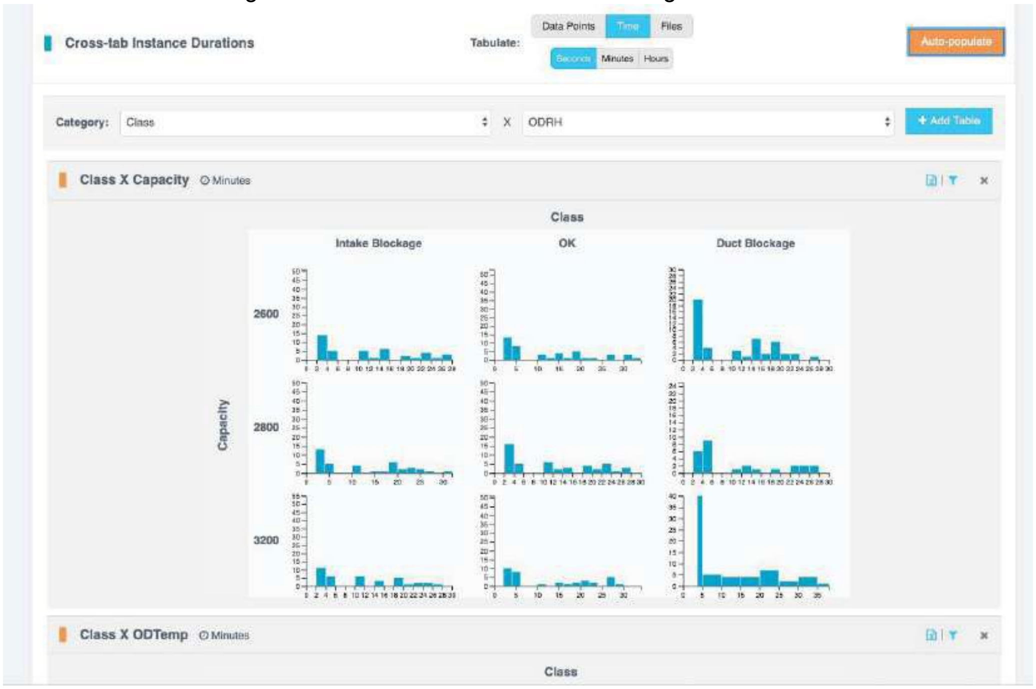


Files: 767 Size: 1.1 GB Data Shape: mixed x mixed Time: 12.6 Hours

Updated: May 4, 2020 2:50 PM

Category: Class X ODRH

Screenshot of a Time Coverage report in Reality AI Tools, run automatically whenever a file is loaded. Shows the amount of data collected for values of target and metadata variables by time-in-class. This is particularly useful for real-time processing applications where the length of the decision window or smoothing over time is a consideration.



Category: Class X ODRH

Screenshot of a Time Coverage cross-tab in Reality AI Tools. Time Coverage statistics can also be downloaded for use in other reporting or analysis software as needed.

Data Collection Process

Using Reality AI Tools for automated data readiness assessment and generation of machine learning models, it is easy to implement an Iterative Data Coverage process as discussed above.

For each coverage area, you initiate data collection and regularly run Data Readiness Assessment in Reality AI Tools. Any issues uncovered can be resolved quickly, and good data can be tested on the most recent version of the ML model. If accuracy is good, you can conclude that the model is generalizing well to the current coverage area. If not, additional data from this coverage area may be required. In any event, errors made by the model should be added to the training set and the model retrained.

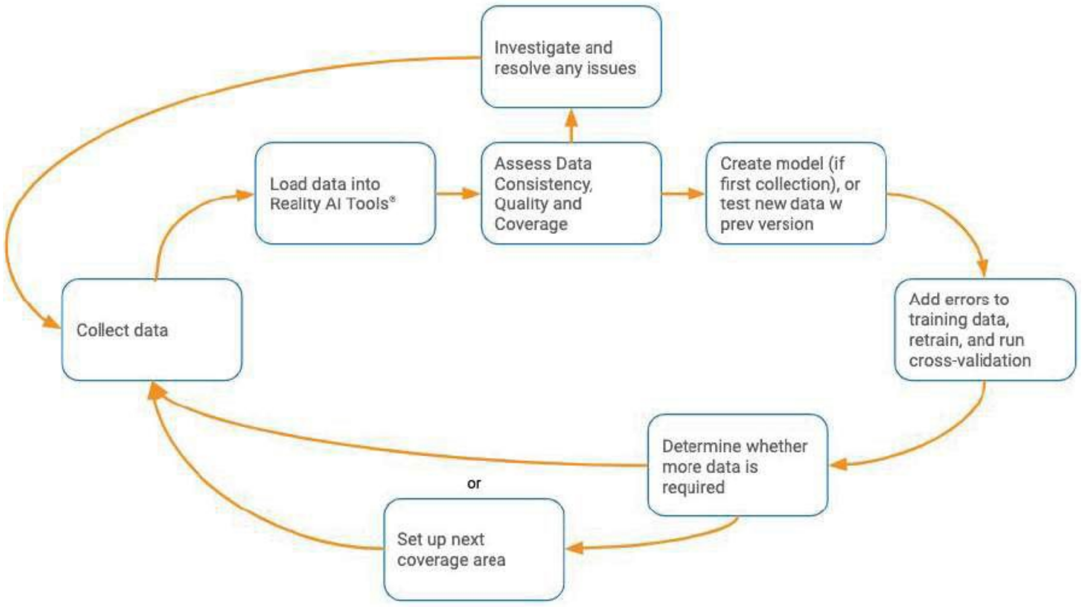

Investigate and resolve any issues
Load data into Reality AI Tools®
Assess Data Consistency, Quality and Coverage
Create model (if first collection), or test new data w prev version
Collect data
Add errors to training data, retrain, and run cross-validation
Determine whether more data is required
or
Set up next coverage area

Diagram showing process for collection using the Iterative Data Coverage method in Reality AI Tools

When doing extended field testing, you may want to run a similar process. Here the objective is also to ensure that collected data is high quality and that ML models can be continuously improved. When error rates are within acceptable limits for the use case, testing is complete.

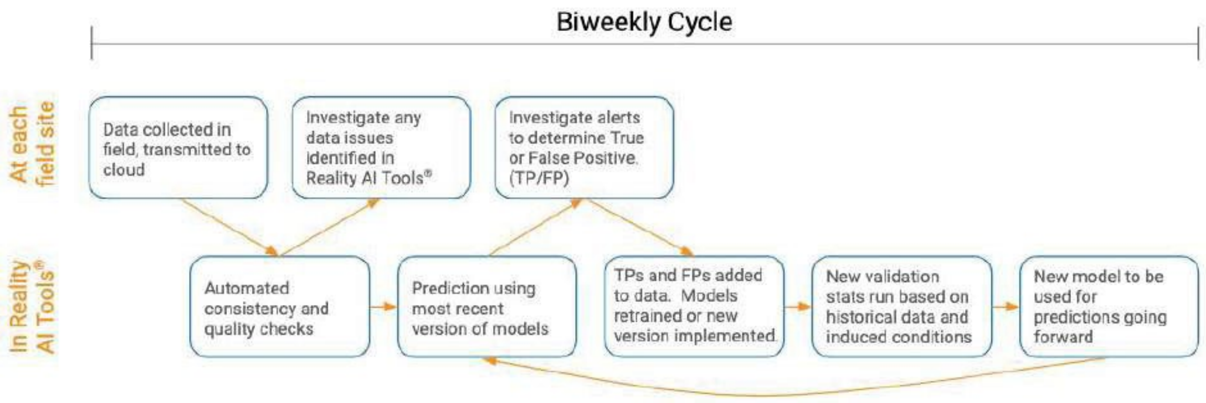

Biweekly Cycle
At each field site
In Reality AI Tools®
Data collected in field, transmitted to cloud
Investigate any data issues identified in Reality AI Tools®
Investigate alerts to determine True or False Positive (TP/FP)
Automated consistency and quality checks
Prediction using most recent version of models
TPs and FPs added to data. Models retrained or new version implemented
New validation stats run based on historical data and induced conditions
New model to be used for predictions going forward

Diagram for a typical field testing process using Reality AI Tools

Conclusion

Instrumentation and data collection are more than 80% of the cost of most machine learning projects. Reality AI software from Renesas has analytics that can help reduce the cost of both. Reality AI Tools identify the most cost-effective combinations of sensor channels, find the best sensor locations, and generate minimum component specifications. It can also help you manage the cost of data collection by finding instrumentation and data processing problems as data is gathered so you can get the most out of using synthetic augmentation techniques.

Reference Material

- www.renesas.com/realityai
- Reality AI Tools®

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu, Koto-ku, Tokyo 135-0061, Japan
https://www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
https://www.renesas.com/contact-us

Trademarks

Doc Number: R11WP0003EU0100