ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TU1BC003

SYSTEMS MODELLING, AI/ML ALGORITHMS APPLIED TO CONTROL SYSTEMS

S. A. Mnisi†, South African Radio Astronomy Observatory, Cape Town, South Africa

Abstract

The 64 receptor (with 20 more being built) radio telescope in the Karoo, South Africa, comprises of a large number of devices and components connected to the Control And Monitoring (CAM) system via the Karoo Array Telescope Communication Protocol (KATCP). KATCP is used extensively for internal communications between CAM components and other subsystems. A KATCP interface exposes requests and sensors; sampling strategies are set on sensors, ranging from several updates per second to infrequent on-change updates. The sensor samples are of different types, from small integers to text fields. The samples and associated timestamps are permanently stored and made available for scientists, engineers and operators to query and analyse. In this paper, I present how to apply Machine Learning tools which utilise data-driven algorithms and statistical models to analyse data sets and then draw inferences from identified patterns or make predictions based on them. The algorithms learn from the data as they run against it, as opposed to traditional rules-based analytical systems that follow explicit instructions. Since this involves data preprocessing, I will touch on how the MeerKAT telescope data storage infrastructure (Katstore) manages the voluminous variety, velocity and volume of this data.

OVERVIEW OF SENSOR DATA STORAGE

Before delving into how Machine Learning (ML) tools can be applied to the MeerKAT Control And Monitoring (CAM), it is important to first give an overview of the storage infrastructure, Katstore[1]. CAM Software components send data points(samples) to Katstore to save and make available for analysis. Katstore stores the values, status and other information about sensors in the CAM system[2]. These samples received by Katstore are keyed on time and sensor name. This makes Katstore a time series database (TSDB), purposely built to have a fixed index on time. The data in Katstore is immutable(no update on a sample is allowed) and only grows over time. It can be seen as an append-only database and samples do not need to arrive in chronological order.

The samples are packed as JavaScript Object Notation (JSON)[3] objects by the software components that collected the samples. Any valid JSON is accepted, with Katstore only requiring that each sample contains the keys name and time, where name is the sensor name and time is the time in Coordinated Universal Time (UTC).

Making each sample a document removes the need for application knowledge in Katstore and future-proofs the implementation. New fields can be added and removed without requiring changes to Katstore and there is no fixed schema for a sensor sample. The software components that collect the samples publish the samples at intervals that can be configured per sensor. Katstore subscribes to the per sensor archive subject on the message bus and stores the published samples.

A sensor is a fundamental concept in KATCP and a collection of sensor types are available. The following types are currently supported: integer, float, boolean, timestamp, discrete, address and string. Sensors always have a status and the following statuses are supported: unknown, nominal, warn, error, failure, unreachable and inactive. In KATCP sensor sampling is performed by the server based on a sampling strategy provided by the client, this allow every connection to set up a unique sampling strategy.

```
{
      "name": "m000_rsc_rxl_cryostat_pressure",
      "time": 1505982067.202219,
      "value": 1013.25,
      "status": "nominal",
      "value_ts": 1505977839.44
}
```

Example 1: Sensor data sample.

There are several sampling strategies available ranging from a fixed time interval (period) to on value change (event). The sensor data, Example 1, that is collected and stored in Katstore is enormous and continues to grow over time.

MeerKAT CAM has many software components, some components connect to hardware devices and others connect to software components. All inter-component communication is done with Karoo Array Telescope Communication Protocol (KATCP). Components can call requests on connected components for control purposes. A KATCP request is analogous to method or command calls of other platforms. For monitoring purposes, KATCP provides the concept of sensors. For the purpose of archiving, the components that make up the MeerKAT CAM system publish sensor samples to different subjects on the message bus. The publish rate is controlled by the system configuration. Katstore subscribes to the archive subjects and stores the samples to the buffer. For the samples to be stored, all the MeerKAT CAM software

* smnisi@sarao.ac.za

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TU1BCO03

components and the hardware devices need to be in operation, preferably optimal state.

APPLICATION OF MACHINE LEARNING IN MEERKAT CAM

Currently, SARAO's maintenance staff relies on preventive maintenance to mitigate breakdowns.

Preventive maintenance happens at times that are pre-set, often long in advance. This is usually driven by previous events and the knowledge and experience of engineers and operators. It includes routine, periodic, planned, or time-based maintenance. It often prevents breakdowns, but unfortunately it can be inexact, which may lead to expensive maintenance before it's needed or to unnoticed weaknesses in the maintenance process. Furthermore, and of higher importance is the strict Radio Frequency Interference Mitigation policy that SARAO enforces on site in the Karoo which is aimed at shielding the equipment from electrical devices. This is very costly if neglected as it causes damage to the Telescopes and related hardware. The costly nature of the hardware also means that inexact, unwarranted maintenance is at the very least costly.

Therein lies the opportunity to investigate in Machine Learning techniques with the ability to transmit and analyse data in real time, which means that "**in-operation"** equipment condition rather than calendars become the foundation for maintenance protocols exactly when and where it's needed.

HOW MACHINE LEARNING WORKS

Machine learning (ML) is a subfield of Artificial Intelligence (AI) focused on building computer systems that learn from data. Depending on the nature of the data and the desired outcome, one of four learning models can be used: *supervised, unsupervised, semi-supervised, or reinforcement*. Within each of these learning models, one or more algorithmic techniques may be applied – relative to the data sets in use and the intended results. These Machine learning algorithms are designed to classify objects, find patterns, predict outcomes, and make informed decisions. ML algorithms can be used one at a time or combined to achieve the best possible accuracy when complex and more unpredictable data is involved.

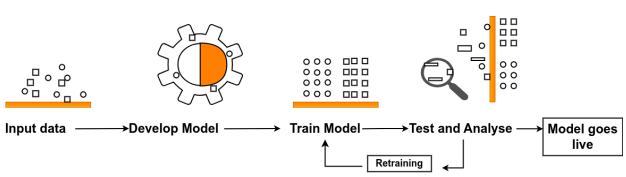

Figure 1: How Machine Learning works.

Figure 1, demonstrates how ML works. ML starts with raw data(input data) which could be from various sources depending on a specific domain. The raw data is split into training data and testing data. Training data is the subset of the raw data that is used to train the ML model and testing data is used to verify the accuracy of the model. The data is gathered and prepared to be used as training data or data that the machine learning model will be trained on, and more data is preferred. A machine learning model is then built and supplied with the data. The model then trained with the data to find patterns and make predictions. The model can be tweaked which includes changing/varying its parameters in an effort to help push towards accurate results. The training stage goes through numerous iterations and tests and analysis of the results is performed until there is confidence in accuracy of the model before it is deployed to a live environment.

Some data is excluded from the training data to be used as evaluation data, which tests how accurate the machine learning model is when it is given new data. The result is a model that can be used with different sets of data in the future.

Finally, once the model achieves a high accuracy rate, a decision can be made to deploy it live where it continues to be monitored.

MACHINE LEARNING APPLICABLE FOR MEERKAT CAM

The vast amount of sensor data that is archived in Katstore and logs from various components presents a challenge in that it is humanly impossible to look across all the data and identify patterns. There are a number of Machine Learning strategies that can be applied to MeerKAT CAM and the glaring one is in predictive maintenance and monitoring. First, a look at what predictive maintenance is and how it could be applied to MeerKAT CAM and other possible scenarios that are possibly applicable to MeerKAT CAM.

Predictive Maintenance

Predictive maintenance (PdM) is a type of condition-based maintenance that monitors the condition of hardware assets through sensor devices. These sensor devices supply data in real-time, which is then used to predict when the asset will require maintenance and thus, prevent equipment failure. Predictive maintenance is the most advanced type of maintenance currently available. With time-based maintenance, one runs the risk of performing too much maintenance or not enough. On the other hand, with reactive maintenance, maintenance is performed when needed, but at the cost of unscheduled downtime. Predictive maintenance solves these issues. With PdM, maintenance is only scheduled when specific conditions are met and before the asset breaks down.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TU1BC003

The proactive nature of predictive maintenance enhances preventive maintenance by providing continuous insights on the ***actual*** condition of the equipment rather than relying on the ***expected*** condition of the equipment based on a historical baseline. With predictive maintenance, corrective maintenance is only carried out when there is a need to do so, and so avoids incurring unnecessary maintenance costs and machine downtime. Predictive maintenance uses time series historical and failure data to predict the future potential health of equipment and so anticipate problems in advance. This could enable SARAO to optimise maintenance scheduling and improve reliability.

Predictive maintenance also differs from preventive maintenance in the diversity and breadth of real-time data used in monitoring the equipment. Various condition monitoring techniques such as sound (ultrasonic acoustics), temperature (thermal), lubrication (oil, fluids) and vibration analysis can identify anomalies and provide advance warnings of potential problems. A rising temperature in a component, for example, could indicate airflow blockages or wear and tear; unusual vibrations could indicate misalignment of moving parts; changes in sound can provide early warnings of defects that can't be picked up by the human ear.

The key is "the right information at the right time." In knowing which equipment needs maintenance, work can be better scheduled, and unplanned costly shutdowns are replaced by shorter and fewer planned shutdowns, thus improving equipment reliability, minimising potential data loss and reducing maintenance costs. In addition, this extends the useful lifetime of equipment and optimises the handling of spare parts. It also increases the productivity of operations and monitoring teams.

HOW A PREDICTIVE MAINTENANCE MODEL COULD BE BUILT FOR MEERKAT CAM

For this paper, the focus will mainly be on three types of machine learning tasks; *classification, regression, optimisation.*

Classification refers to a predictive modeling problem where a class label is predicted for a given example of input data. For example, given an email, classify if it is spam or not.

Regression analysis is a predictive modeling technique that estimates the relationship between two or more variables. It focuses on the relationship between a dependent (target) variable and an independent variable(s) (predictors). An example would be a house being predicted to sell for a value in a certain range given a number of factors(predictors).

Optimisation is the problem of finding a set of inputs to an objective function that results in a maximum or minimum function evaluation.

The following scenarios amongst others, can be considered for building a machine learning model for MeerKAT CAM as a starting point:

1. Which device is going to fail: a classification scenario
2. What is the remaining life of a device: a regression scenario
3. Optimisation scenario: This is an advanced ML scenario usually best applied once a prediction model matures

Considering the above scenarios, the first step is to build a classification or regression model for determining what is the remaining life of a machine or whether a certain device will fail. Once the model achieves maturity, then building an optimisation scenario can commence. What generally happens, is that a human takes a look into the machine learning output and makes a decision. When a machine learning model gets to a mature stage, then instead of a human making a decision, the machine learning model is left to make the decision itself.

The above scenarios could be used as a basis for brainstorming which scenario makes the best value proposition for SARAO or MeerKAT CAM.

The next step is to use the archived sensor data, logs and other static data to create features. Features in machine learning could be described as representation of data in a form that can be used to map input data to output predictions. One of the challenges is that most systems, MeerKAT CAM included, were not built with machine learning in consideration. The data needs to be processed into a form that algorithms can work on. Thus, the creation of features is needed in order to bring data into shape where the algorithm can recognise and work on it. Once the data is in shape, then an algorithm is selected to build the model.

Ultimately, the machine learning scenarios mentioned above need to at least make business sense. When working on a machine learning model, an effort should be directed towards converting a business use case into a machine learning use case. Taking the first scenario of whether a device will fail or not, there needs to be a clear definition in the use case of how the output of the machine learning model will be used.

The following questions will inform a decision for a use case which ultimately will get converted into a hypothesis.

- How the output of the PdM model will be used?
- What is the definition of device failure or breakdown?

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TU1BCO03

- What signals have patterns of breakdowns or degraded performance?
- The frequency with which signals are collected?
- How much normal and failure data is available?

Forming a hypothesis

A hypothesis in machine learning is a model that approximates a target function and performs mappings of inputs(data) to outputs(predictions). Given the classification and regression scenario of estimating when a device will fail or the device's remaining life, the following hypothesis could be drawn.

- Predict if equipment will fail in the next 'K' period
- Predict if equipment will fail in the next 'K1', 'K2'...Kn period
- Predict if equipment will fail in the next 'K' period due to fault in part 'N'
- Predict time to failure or remaining life of equipment
- Find anomalies

Once the use case has been converted into a hypothesis, then perform the data exploration exercise to determine whether the data set and the use case are in alignment for this use case or not.

Steps to build a Machine Learning Model

The following, shown in Figure 2, are general steps to go through when building a machine learning model:

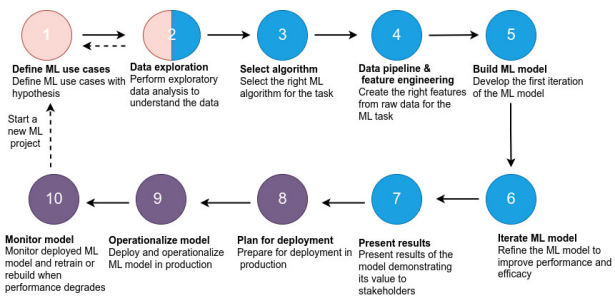

Figure 2: Steps to build a Machine Learning Model.

More often than not, during data exploration there is misalignment in terms of the data set and use case. When this happens, you take a step back and decide whether different data is needed, or more data. Sometimes, a decision could be made to change the use case. In terms of data, SARAO is in a privileged position. It will all come down to how the data is crafted into features that a machine learning model can use effectively.

What to look for in Predictive Maintenance

When building a Predictive Maintenance model, for example for a device in the Telescope Hardware Network or in some other component of MeerKAT CAM that has sensor data transmitted from it, where the aim is to determine when this device will fail - the objective is to look for patterns in the data exploration such as speed, efficiency, pressure, load etc reducing/degrade as the device gets older. Notable patterns could include the speed, efficiency, pressure etc being good/optimal when the device is new and degrading as the device gets older. Also, heat, noise, vibration could start low when the device is new and increase as it ages. These are some patterns that when spotted in the data exploration phase, could inform a decision in building a machine learning model for MeerKAT CAM.

Datasets for Predictive Maintenance

Time series data is the most powerful and more useful for building predictive maintenance models. However, combining it with static data provides a powerful overview. Knowing when past maintenance or service was done on machinery, and adding that to the data set also makes a model more powerful.

Strategies for creating labels

A label in ML is a description that explains a piece of data to a model so that it can learn from that description. For a PdM model, the following actions are used to create labels amongst others.

- Convert failure signals into a label
- Convert degrade signals into a label

In Predictive Maintenance, one of the issues is that data labeling is a requirement in most cases. Most of the time these need to be created.

Selecting a label for classification

In the classification scenario to determine when a device will fail, a label needs to be created for classification. In MeerKAT CAM signal data is produced and archived frequently. However, when a device fails, that is the final label. In order to catch the failure earlier, for example 10 days earlier, 10 days of data needs to be tagged before failure as label data. As such, that becomes the failure data, which is a positive label. The remaining data becomes a negative label. Depending on the use case, a determination is made of how far before the failure, the failure is to be predicted and then tag that much period before failure as the label.

Selecting a label for regression

In similar fashion, when trying to predict the remaining life of a device, a deep learning model could easily be built where there is a final failure and the sensor data. The data can then be tagged at various places, for example,

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TU1BC003

how the signal was at 15% life, at 30% life, at 60% life to enrich the model.

Figure 3: Regression label example.

Figure 3 above is an example of how from failure data, calculations could determine how much life was left at various intervals before the device.

Creating features

The next step after having created the right labels is to create features. As stated earlier, features in machine learning could be described as representation of data in a form that can be used to map input data to output predictions. The Telescope Hardware Network comprises multiple devices, however standard features could accommodate most of them. These are listed in Figure 4. Tumbling or moving averages are widely used since they are generally good in showing short-term or medium-term patterns of failures.

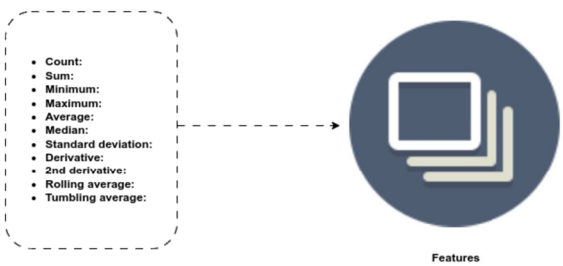

Figure 4: Feature example.

Algorithms for Predictive Maintenance

At this stage, armed with a use case, a data set and labels - it is time to select an algorithm. Figure 5 shows which algorithms are best suited for each of the scenarios.

Will it fail?	Will if faill for reason X?	After how long will it fail?	Is the behaviour anomalous?
Classification	**Multiclass classification**	**Regression**	**Anomaly detection**
RNN, LSTM	RNN, LSTM	RNN, LSTM	Conditional AutoEncoder
DNN Classification	DNN Classification	DNN Regression	AutoEncoder
Traditional ML e.g Random Forest SVM, Decision Trees	Random Forest, Decision Trees, Hidden Markov chain	RF Regression	Anomaly detection

DNN - Deep Neural Network **RNN** - Recurrent Neural Network **LSTM** - Long Short Term Memory

Figure 5: Algorithms for Predictive Maintenance.

LOOKING AHEAD

At some point, when the model has matured then one of the advanced scenarios, *Optimisation scenario,* could be considered. This is when the model is left to make a decision. For MeerKAT CAM, one of these cases could be for example, the cooling system in the Karoo Array Processor Building(KAPB) which cools all Control And Monitoring as well as Science Data Processing(SDP) servers. An optimisation scenario could be achieved by using reinforcement learning where the machine learning model makes a decision such as how much of cooling power to use. This could result in energy saving.

CONCLUSION

In order for MeerKAT CAM to keep innovating and help Astronomers and Scientists to advance future research, Machine Learning and AI are critical fields to delve in. Monitoring the operational health of the telescopes and related hardware is very crucial and machine learning could help in getting more insight which ultimately could lead to timely informed decisions.

Finally, the scale of the data that MeerKAT CAM software components send to Katstore to save and make available for analysis is such that no human can look across all the data and identify patterns and informed insights from it.

REFERENCES

[1] M.J. Slabber. 'Overview of the monitoring data archive used on MeerKAT', in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015, pp. 1155–1157. doi:10.18429/JACoW-ICALEPCS2015-THHD3O06

[2] N. Marais, "MeerKAT Control and Monitoring System Architecture", in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015, pp. 247-250. doi:10.18429/JACoW-ICALEPCS2015-MOPGF067

[3] T. Bray. The JavaScript Object Notation (JSON) Data Interchange Format. RFC 7159 (Proposed Standard), https://www.rfc-editor.org/rfc /rfc7159.txt