Anomaly Detection in IIoT: A Case Study using Machine Learning

Gauri Shah*
Persistent Systems Ltd., Pune, India
c-gauri_shah@persistent.co.in

Aashis Tiwari
Persistent Systems Ltd., Pune, India
aashis_tiwari@persistent.co.in

ABSTRACT

In this paper, we explore multiple machine learning techniques applied for anomaly detection in IIoT data from engine-based machines. We evaluate sensor data on different engine characteristics such as fuel usage, engine load, and oil pressure to gauge when a particular engine shows anomalous behavior and may experience a failure. In particular, we use methods such as multi-variate linear regression, Gaussian mixture models, and time-series data analysis to detect outliers in the machine behavior. Timely detection of such anomalies helps the maintenance staff to perform preventive maintenance and ensure maximum up-time for the machines. In addition, anomalous behavior may not always indicate failure but simply inefficient usage of the machines; we also try to optimize these inefficiencies.

1 INTRODUCTION

The Industrial Internet of Things (IIoT) is the integration and linking of big data, analytical tools and wireless networks with physical and industrial equipment, or otherwise applying meta-level networking functions, to distributed systems. [1]. The data is collected from sensors which are embedded in machines deployed in remote locations in the field. One of the biggest challenges in such large-scale industrial machine deployments is keeping track of the functioning of the machines, and minimizing downtime. There is a huge benefit in being able to predict accurately the failure of a particular machine. If we fail to detect a malfunction before it occurs, there is a significant penalty to repairing the machine, which includes the lost working time of the machine. If we predict a malfunction too far ahead of when it actually happens, then the preventive maintenance is wasted in the sense that it could have been done later and more work could have been completed by the machine. In an ideal scenario, sensor data from all the deployed machines should be studied and outliers in the machine behavior should be identified as soon as possible. New alerts should be displayed to an administrator in charge of the maintenance, who can then decide what corrective action should be taken. This approach does not require the administrator to constantly monitor all the machines but only focus on the ones which have been identified to have a potential problem.

With historical sensor data, a machine learning system can be used to learn the normal behavioral patterns of a given machine. It is likely that such machines already have well-defined running specifications allowing the use of simple rule-based systems to identify anomalies. For example, if a machine has an expected engine speed of 1800RPM but it is running at 2200RPM for a prolonged period of time, then it could be cause for concern. But it is also likely that this is the normal behavior of the machine when it starts up for the first time after a long cooling period. In practice, we find that machines deviate from the expected running values due to a variety of factors such as tuning, frequency of engine servicing, weather, etc. Further, each machine degrades differently so these running values have to be updated periodically as the pre-defined thresholds may not always apply. Hence, simple statistical methods are not sufficient here, and it is essential to *learn* the behavior of the machine over time. We use machine learning techniques such as regression analysis and Gaussian Mixture Models for learning.

*Authors are listed alphabetically by last name.

2 OVERVIEW

In this paper, we study the behavior of engine-based machines based on certain characteristics to detect when a particular machine is malfunctioning. We have used historical data from real machines deployed in the field over a period of three months. Readings from the machines are obtained per minute for the following parameters: Coolant Temperature, Engine Load, Fuel Pressure, Fuel Usage, Oil Pressure, System Voltage, Engine Speed, Air Intake

Temperature, and Total Fuel Used. Out of these nine parameters, we only consider the relevant ones for different use cases based on the domain knowledge of field experts. We use this data to learn the behavior of the machine in the steady state; the learnt model is then deployed at run-time in the field to detect anomalies.

In particular, we studied the following three use cases:

1. **Fuel Economy Degradation**
   If the profile of fuel usage increases for an engine load profile on a particular machine, it signifies fuel economy degradation. We study the relationship between the engine load and the fuel usage. We expect to see that the fuel usage increases as the engine load increases, within certain limits. If we see that the fuel usage is higher than the expected learnt values, we flag it as an anomaly.

2. **Oil Pressure Profile**
   The relationship between the engine speed and the oil pressure is similar to the relationship between engine load and fuel usage. The oil pressure varies with the engine speed but it remains at a comparatively stable value depending on whether the engine is running or in an idle state. We model this relationship and flag anomalies which deviate significantly from the learnt model.

3. **Oil Pressure Stabilization Gradient**
   When the engine starts up after being idle for a long period, the oil pressure starts at a high value because the oil temperature is low. However over a reasonable period of time, the oil pressure has to stabilize to a normal value; if the pressure drops too suddenly or remains at the initial high value, it could indicate an issue with the oil condition. Then we should trigger an alert to change the oil or update the oil change schedule.

2.1 Implementation

Sensor reading data from the various engines are collected in real time and loaded into a proprietary big-data system. This data is available for consumption via REST APIs, and simple connectors for analysis and visualization. The historical data post-ETL process is available for export in standard formats (such as csv, tsv etc.) using an interactive UI for analysis.

The implementation of these use cases was done in Python using Jupyter notebooks [3]. We used standard python packages such as `numpy` for numerical computations, `pandas` for data ingestion and handling, and `bokeh` and `matplotlib` for visualization.

3 RELATED WORK

Anomaly Detection is an important field in IIoT today. There are many different techniques for anomlay detection [10, 11] such as density-based techniques (k-nearest neighbors, local outlier factor), once-class SVMs, fuzzy-logic based outlier detection, ensemble techniques, etc. We note that we cannot directly use of any of these techniques in our work because the default behavior of the machines is not known upfront; we are more interested in learning this behavior rather than the actual anomaly detection. The latter can be done using a simple rule-based system once the expected pattern is understood.

Popular IIoT use cases for anomaly detection include:

- Reducing operational costs and preempting service calls, thereby increasing the life of machines.
- Streamlining the manufacturing processes by detecting real-time issues in assembly lines.
- Proactive operational maintenance by reducing downtime and bottlenecks.

We use Linear Regression and Gaussian Mixture Models to learn the default behavior of machines in order to account for variations due to multiple reasons. Our method for learning the linear relation between engine load and fuel usage can be applied in other settings to learn the relationships between machine parameters. We do not have to restrict ourselves to linear regression or having just one independent variable that affects the value of the dependent variable. Even in our own use cases, we can use this type of analysis to learn the relationship between oil pressure and engine load. Further, our technique of using Gaussian Mixture Models to learn the idle and running time values of oil pressure, can be used to learn the default values of other parameters of industrial machines. Here we need to know approximately how many potential clusters of values are possible.

4 USE CASE DETAILS

In this section, we look at the details of each of the use cases listed above and evaluate the approaches used to find the solutions.

4.1 Fuel Economy Degradation

In this use case we learn the relationship between engine load and fuel usage. First, we look at a histogram of number of times a particular value of engine load appears in the data for a given fixed value of fuel usage. We bucket the fuel usage values in groups of 10 as seen in Figure 1. We notice that we get a high concentration of engine load values around 28 when fuel usage is around 30, and engine load values around 10 when the fuel usage is around 10. This graph suggests that there may be a possible linear relation between engine load and fuel usage. So we use a linear regression model [2] to fit the data as shown in Figure 2.

However we notice that a bunch of points are located below the linear regression line, and the hypothesis does not appear to fit the data points well. In order to solve this problem, we looked at three different approaches.

1. Most of the outlier points lie in the range $[0, 40]$ of engine load values so we counted the number of data values in that range. We found that very few points lie in that range so they can be safely ignored.

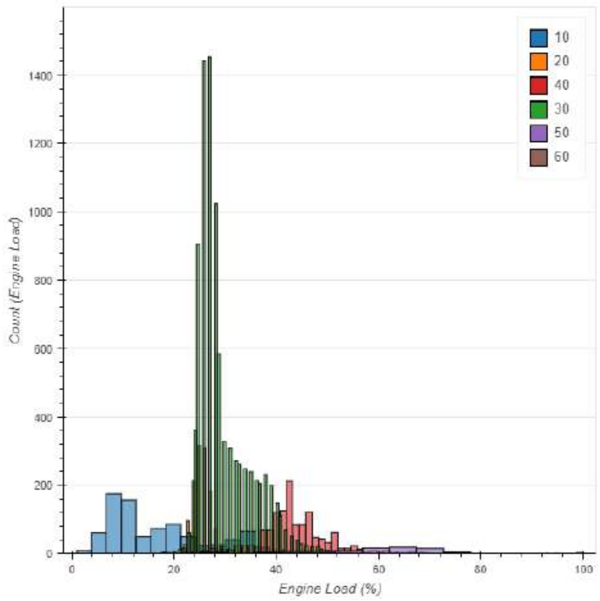

Figure 1: Histogram of engine load values for buckets of fuel usage values.

2. We can plot piece-wise segmented linear regression in two ranges from $[0-20]$ and $(20-70]$. This approach, although reasonable, also did not appear to fit all the data points well.

3. We considered multi-variate linear regression as we noticed that most of the engine load values are focused around a couple of fuel usage values as seen in Figure 1. We separated the values into two groups depending on whether the engine is running or it is in an idle state based on the engine speed. We separately fit two different linear regression hypotheses on these two sets of points, and this approach gives us the best result as seen in Figure 3.

4.1.1 Anomaly Detection

We use the following configurable parameters for anomaly detection:

1. Sliding Window (P): The time slice to be considered for anomaly detection.

2. Threshold (T): Magnitude of deviation from the stable regression values.

3. Percent Samples ($N\%$): Minimum percentage of samples above the threshold value.

Once we have learned this model, the anomaly detection is defined as follows: we raise an alarm if more than $N\%$ of readings deviate from the model by a certain threshold value T in specific sliding window of time P.

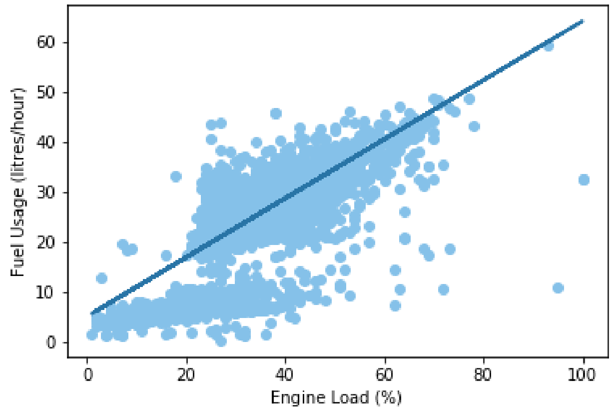

Figure 2: Linear Regression between engine load and fuel usage.

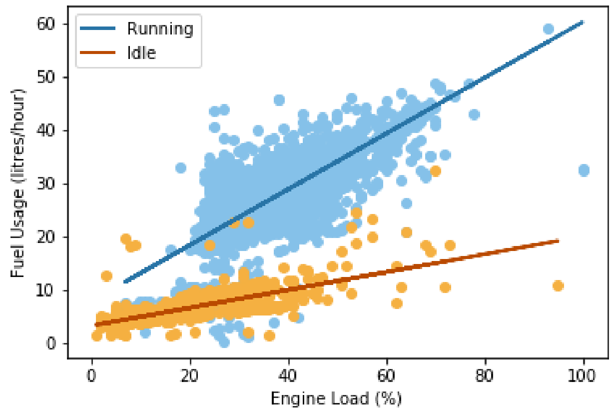

Figure 3: Linear Regression for Engine in Running State and Idle State.

4.2 Oil Pressure Profile

Unusually low or high oil pressure in an engine regardless of performance is an indication of an anomaly and requires immediate investigation. It may also indicate abnormal changes in other factors of the engine such as oil temperature, coolant temperature, etc. Although there is an effect of engine wear-and-tear on oil pressure values, it is not very significant and the values should remain within a small range. Nonetheless, the oil-pressure usage profile requires re-evaluation from time to time.

When the load on the engine is high, the engine operates at a higher speed (RPM), hence the oil pressure goes up and vice-versa. Using the domain knowledge from field experts, we know the engine is running when the RPM value is high (~ 1900), and it is idle if RPM values are lower (~ 850). We use historical readings from the engine for oil pressure and engine speed for modeling the system profile. Our goal is to automatically learn the stable range of oil pressure values given the engine speed, and raise an alert if it deviates from the learnt values for a long period of time.

We did a preliminary investigation by visualizing the oil pressure and engine speed values with the help of histograms. Figures 4 and 5 shows that the oil pressure values in a particular engine state (idle or running) follows a Gaussian distribution. We can safely ignore the higher values of oil pressure as they are engine warm-up readings, which stabilize over time (see use case 3).

Fig 6 indicates the concentration of oil pressure values at various engine speed levels, indicating high density around 850RPM and 1900RPM. From the figures, we also observe the oil pressure values around 160kPa for the idle and 300kPa for running states. The underlying data is observed to be from a mixture of a finite number of Gaussian distributions, where the individual components are related to the engine speed levels respectively. We model the profile using Gaussian Mixture Models (GMM), where the mean values of the components indicate the stable oil pressure values.

4.2.1 Hyper-Parameter Search and Model Tuning

The GMM [4] implementation in `scikit-learn` [5] uses Expectation Minimization (EM) algorithm [6] for fitting the model, and uses the Bayesian Information Criterion (BIC score) [7] to assess the optimal number of clusters. A lower BIC score indicates a better model fit. We do an efficient hyper-parameter search using various covariance types and the number of components for the model selection. We tried our experiments with four covariance types: Spherical, Diagonal, Full, and Tied, and up to four components.

As seen in Figure 7, the BIC score reduces with increasing number of components as expected. However, from our domain knowledge we know that there should be two components for the two states of the engine: running and idle. We observed that the spherical covariance type with two components has the least BIC score as seen in Figure 7. We report the means of the individual components (cluster centroids) as the mean value of oil pressure for the two engine states. The mean oil pressure values are as follows:

1. Idle State: Centroid at 156.83kPa (with 16.42% data points matching this centroid)

2. Running State: Centroid at 295.22kPa (with 83.58% data points matching this centroid).

These values match well with the expected values given above. Note that when we consider four components also, we get values that effectively indicate two components as shown below:

1. Centroid at 0kPa (with 0.79% data points)

2. Centroid at 157.36kPa (with 11.58% data points)

3. Centroid at 289.34kPa (with 67.16% data points)

4. Centroid at 292.91kPa (with 20.47% data points)

4.2.2 Anomaly Detection

We use the following configurable parameters for anomaly detection:

1. Low Mean: The idle-state oil pressure mean derived from the GMM.

2. High Mean: The running-state oil pressure mean derived from the GMM.

3. Sliding Window(P): The time slice to be considered for anomaly detection.

4. Threshold (T): Magnitude of deviation from the stable mean values.

5. Percent Samples ($N\%$): Minimum percentage of samples above the threshold value.

Once we have learned this model, the anomaly detection is defined exactly like the earlier use case: we raise an alarm if more than $N\%$ of readings deviate from the model (idle or running state) by a certain threshold value T in specific sliding window of time P.

4.3 Oil Pressure Stabilization Gradient

Consider an engine that is started after a long period of idling. The oil pressure suddenly increases but it should stabilize in a certain time period thereafter. If it does not stabilize and it either drops too low or stays at the initial peak value, it could indicate an issue with the oil condition. We were interested in determining how long it takes for the oil pressure to stabilize once the engine has started up. For this use case, we did a time-series analysis of the data; typically the engine starts up in the morning and run through the day. We broke up the data set into multiple readings spanning a single day, with the first reading being when the engine first starts up.

We ran into a problem of *missing data* because some of the sensor readings were missing; in this case getting accurate readings for the entire window of stabilization is critical as otherwise we will learn an incorrect value for the stabilization period. We discarded the readings for those days where the following conditions were applicable:

- Values were missing for the entire day.

- More than 30% of readings were missing for first 30 minutes.

- The engine was started but not doing any useful work within first 30 minutes.

Some sample graphs of the remaining time-series data are shown in Figure 8.
We then plotted the histogram of time (in minutes) taken for the oil pressure to cool down to a value within a small window of 290kPa. Although there are few readings for some days, we see that the oil pressure roughly stabilizes in under 13 minutes under normal operating conditions. Once we learn the time window for oil pressure stabilization during start-up, we can then use it for the testing phase. If the oil

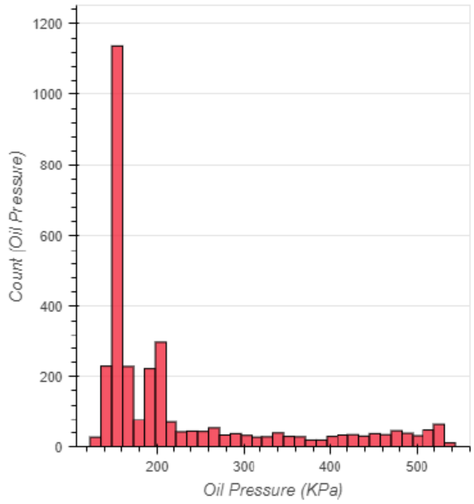

Figure 4: Histogram of oil pressure values in idle engine state.

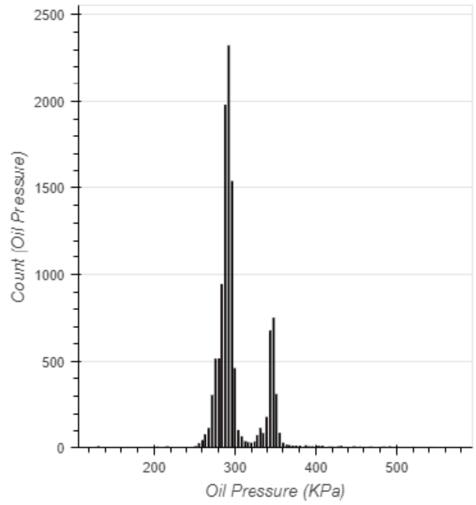

Figure 5: Histogram of oil pressure values in running engine state.

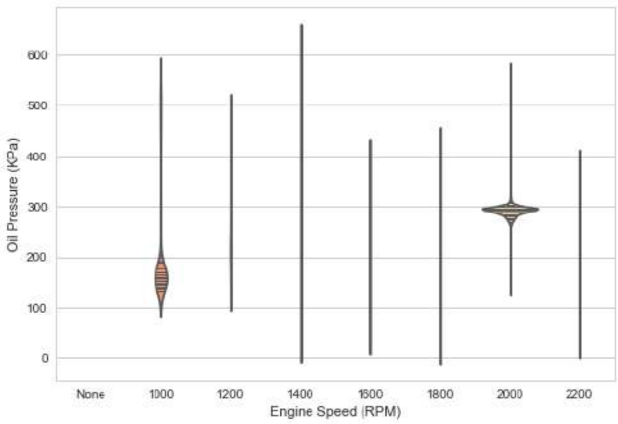

Figure 6: Spread of oil pressure values for given engine speed values.

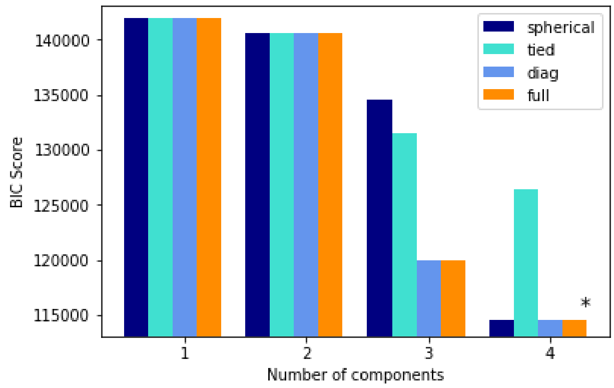

Figure 7: BIC scores used in GMM to determine stable oil pressure values. ∗ indicates the best result across the different number of components and various covariance types.

pressure does not stabilize around 290kPa within 13 minutes since when the engine started up, an alert is raised to the maintenance team.

5 CONCLUSIONS AND FUTURE WORK

We looked at multiple use cases of anomaly detection using sensor data from machine-based engines. Some of these use cases were solved using simple statistical analysis whereas we used machine learning techniques for some of the use cases as listed here. These solutions were very useful for the maintenance team in focusing on the anomalies and taking preventive action to fix any issues. Our solutions are being deployed in the field at the time of submission of this paper.

In section 4.2.1, we did a hyper-parameter search on number of components and covariance type where the data can be assumed to be generated by a Dirichlet Process [8]. However, the number of clusters could change over time, depicting different engine behaviors in different states. Non-Parametric mixture models, such as Dirichlet Process Mixture Model (DPMM), assume infinite clusters/components and infer the correct number of components instead of restricting them as in Gaussian Mixture Models. We could model the profile using Bayesian Inferencing [9] with prior as a Dirichlet distribution, and use the categorical distribution as conjugate prior.

In the analysis of warm-up period oil pressure gradient discussed in Section 4.3, we can take into account other features such as the effect of environmental temperature on warm-up times on cold start.

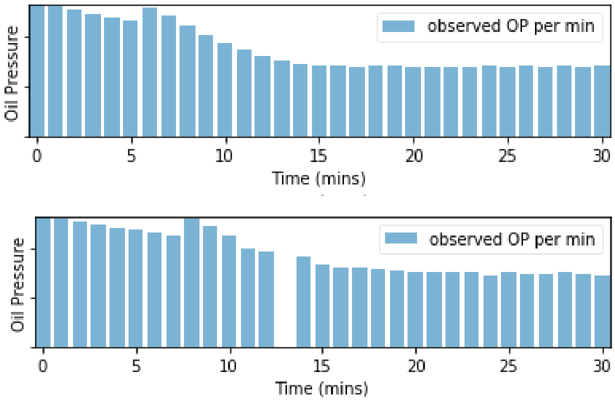

Figure 8: 2 days of oil pressure values after engine start up.

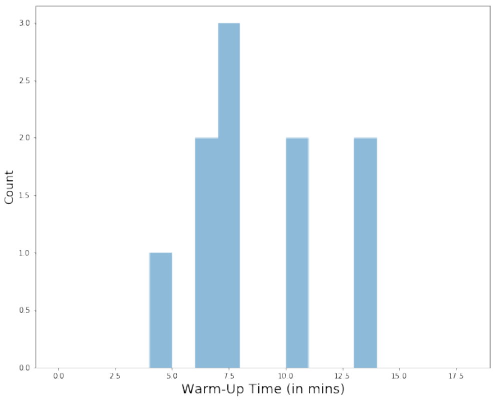

Figure 9: Histogram of time taken (in minutes) for oil pressure to stabilize after engine start up.

REFERENCES

[1] Industrial Internet. `https://www.techopedia.com/definition/30044/industrial-internet`

[2] Andrew Ng. CS 229 Machine Learning course at Stanford University. Autumn 2016. `http://cs229.stanford.edu/notes/cs229-notes1.pdf`

[3] Jupyter Notebooks. `http://jupyter.org/`

[4] Gaussian Mixture Models. `http://www.cs.cmu.edu/~./awm/tutorials/gmm.html`

[5] Gaussian Mixture Models in `scikit-learn`. `http://scikit-learn.org/stable/modules/mixture.html`

[6] Expectation Maximization Algorithm. `https://www.cs.utah.edu/~piyush/teaching/EM_algorithm.pdf`

[7] Bayesian Information Criterion. `https://en.wikipedia.org/wiki/Bayesian_information_criterion`

[8] Dirichlet Process. `https://en.wikipedia.org/wiki/Dirichlet_process`

[9] Bayesian Inference. `https://brohrer.github.io/how_bayesian_inference_works.html`

[10] Anomaly Detection. `https://en.wikipedia.org/wiki/Anomaly_detection#Popular_techniques`

[11] Varun Chandola, Arindam Banerjee, Vipin Kumar: Anomaly Detection - A Survey. ACM Computing Surveys (CSUR), Volume 41 Issue 3, July 2009.