Predictive Maintenance Adoption:

Overcoming Common Data Quality Issues

By Justin Gagne,
SENIOR DATA SCIENTIST

Very

When it comes to manufacturing, downtime is the major productivity killer.

If you've read our previous white paper, Predictive Maintenance for the Connected Plant, you know that while only 12% of plants take a predictive approach, those that do maximize their throughput and save as much as 40% when compared to inferior maintenance strategies.

Predictive maintenance more efficiently determines potential points of failure in an unexpected outage, detects incipient failure before it causes downtime, and maximizes the remaining useful life of machine components. But with so many factors to consider, preventing downtime becomes an elusive goal.

That's where machine learning (ML) comes in.

Why Use Machine Learning?

Manufacturers turn to machine learning when the amount of data they generate overwhelms traditional analytical methods. For example, in a recent project, just one of the machines we monitored sampled 1,500 data points *per second*. That's 5.4 million individual data points per hour. That amount of data is far beyond what a human can manage on their own. Even with the help of Excel and smart pivot tables, finding value in the information would be difficult. And doing all of that evaluation in time to prevent an outage would be nearly impossible.

ML synthesizes these vast amounts of data for you so that you can act on it.

The Top Three Hurdles to Predictive Maintenance Adoption

As with most projects, you should put a plan in place for implementing predictive maintenance with a clear-cut start and finish – you need to understand what product you're building and what role machine learning plays in that product. Ask yourself:

- What is your hypothesis?
- What data do you need to test it?
- Can you access this data?
- Who needs to know the results of this project so that they can act?

A predictive maintenance program can be complex, but these are the most common hurdles we see in adoption.

Sourcing and Maintaining Quality Data

Machine learning is not magic. Having the right amount of clean data makes an enormous difference in machine learning outcomes. If the data you have is not related to what you're trying to predict, you will not be successful in making those predictions. For machine learning to work, you need access to enough of the right type of data.

Data Must Be Clean and Relevant to Your Problem

How much data is enough? That depends on the problem you want to address. How accurate do your predictions need to be? How nuanced are the possible outcomes?

Most companies understand the need for clean data–removing missing data points, eliminating duplicate data, resolving impossible-to-achieve results, etc. Many companies have one of two data-related problems. They either don't have any data, or they have floods of data and don't know what is significant to the problem domain.

Overcoming the Data Source Problem

If predictive data doesn't yet exist in your facility, you can gather it by taking inventory of your available resources, retrieving the data out of your existing tooling, and incorporating new sensors to supplement where not enough data exists. If you have too much data, you're in a much better position, but you'll still want to weed out irrelevant data to reduce storage and processing costs. Finding the causal data is part of the iterative predictive maintenance workflow.

Automating the Data Engineering Process

If you begin automating your data engineering process blindly, you can easily introduce technical debt in your system by choosing a data structure that doesn't match the algorithm you need. It's best to work with production SMEs, designers, and product teams to understand their requirements.

Before Building Your Pipeline, Know Your Data

Manually analyzing your data can save you significant time and work as you begin to automate the process. Start with the problem you're trying to solve. What algorithms are likely to provide the most useful information? Now you can structure your data with all of the features you'll need to run those algorithms.

Remember that the sophistication of your approach is based on the maturity of the project. Features will likely increase over time. In the beginning, your data pipeline may be as simple as moving the data from one place to another. By planning ahead with a focus on both the problem and end-user needs, you'll avoid painting yourself into a structural corner or wasting effort on work that doesn't align with your goals.

Presenting Usable Output Data to the Right People

People often tout the power of machine learning but neglect the delivery. For example, the typical output from a machine learning model is a .csv file. We've seen companies put in significant effort to generate data only to have that .csv file buried in a folder somewhere and left to collect dust.

Identify Who Needs to Know and How They Access Information

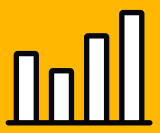
Start your project by defining who needs to know the information. Often the person is in a unique environment. They may be operating machinery and not have access to email or even a traditional computer during their shift. How they will receive the information should be a factor considered as you define the ideal output of the tool. What information do they need to see, and in what form do they need to see it?

We often find that organizations choose to display information in some form of software as a service (SaaS) application. Building a SaaS application means developing an additional fully fledged tech stack and hiring software engineers, designers, web developers, and operations specialists to manage and maintain the application.

At Very, we understand these challenges, so we know that data science is only one part of the complete solution – and how essential it is for you to fully grasp the potential implications of your project before you get started.

3 Keys to Overcome Predictive Maintenance Adoption Hurdles:

Source the right amount of relevant data



Manually **analyze** some of your data before building the data pipeline

Research who will be consuming the data and how they will receive it

Starting with Predictive Maintenance

Predictive maintenance delivers a massive return on investment, and it will work for you if you're willing to put in the effort. Very has helped many companies improve their maintenance performance – we can also help you through the challenges.

Prediction requires a large amount of clean historical data. Often, when we first start projects, there is simply not enough information about the machines and their failures available to build reliable, supervised models.

What do you do if you want to start using data to inform your maintenance procedures, but you don't have historical data about the downtime events? Because of differences in environment, monitoring machines – even machines created to do the same job – can be like starting from scratch.

Very approaches projects by starting simple and increasing complexity over time as we vet how the simple approaches work. We always want to deliver some type of user-facing product for every step along the way.

If we don't have enough *labeled data* (for example, if the machines are newly installed), we'll start by using basic statistics to build outlier detectors to detect anomalies within the sensor values.

Predictive maintenance delivers a massive return on investment.

When detecting anomalies, we look for data points that do not fit within a group. For example, if you are looking at the vibrations on a particular bearing, these outliers could predict a deviation from the norm that may indicate an impending failure. This strategy can be applied to all types of data to discover and label scenarios that lead to device failures.

Making the Best Predictions with Available Data

Before diving into how to use machine learning to make the best predictions with the available data, it's necessary to understand a critical distinction between the two primary ways machines learn. When we use the word "prediction," it implies the use of historical data – lots of it – to make educated guesses about the likelihood of recurring incidents. This data, labeled with the type of failure, is where the distinction between learning methods lies.

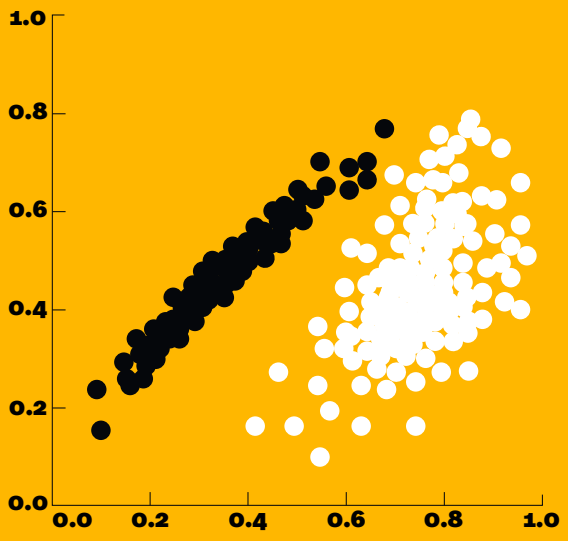
Supervised vs. Unsupervised Machine Learning

Supervised machine learning requires a labeled training set. This means there is a known output we can use to train the model. For example, if we are trying to predict when a bearing will fail, we must have a known set of data on failed bearings that we can use to train the model.

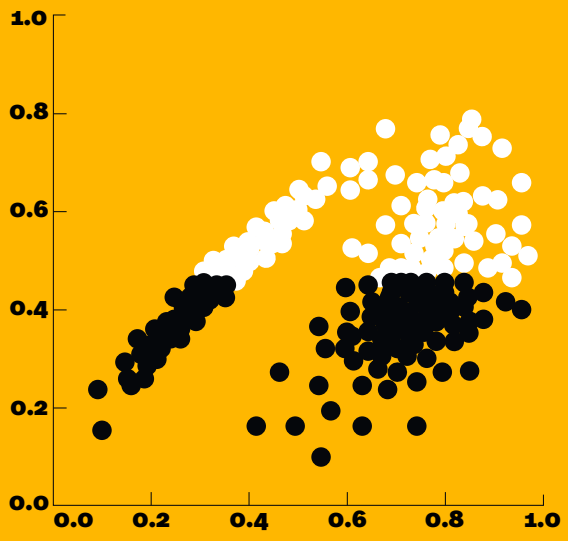
If we do not know what the output should be, or do not have enough historical labeled data on which to train the model, we can use unsupervised machine learning. As another example, if we are trying to predict if a bearing will fail or not fail, we would need a labeled training set to train the model. But if we are predicting how long a bearing will last, we do not need a labeled training set because we do not know what the output should be – the bearing either fails or doesn't.

Use Unsupervised Learning for Unstructured Data

Unsupervised learning can help you make sense of data you've gathered by automatically discovering features correlating with the results you hope to achieve. In unsupervised machine learning, the algorithm must find the output independently. Two standard techniques to be aware of are clustering and the use of autoencoders.



Clustering Algorithm Success



Clustering Algorithm Failure

SOURCE: https://machinelearningmastery.com/clustering-algorithms-with-python/

Clustering

Clustering is a technique where you group similar data together and then label those groups with a name. For example, if you have many data points about bearings and group them by how long they last, you might call one group “Long Lasting Bearings” and another group “Short Lasting Bearings.” In this way, you have discovered two groups of bearings with different lifetimes – without knowing what those lifetimes are ahead of time.

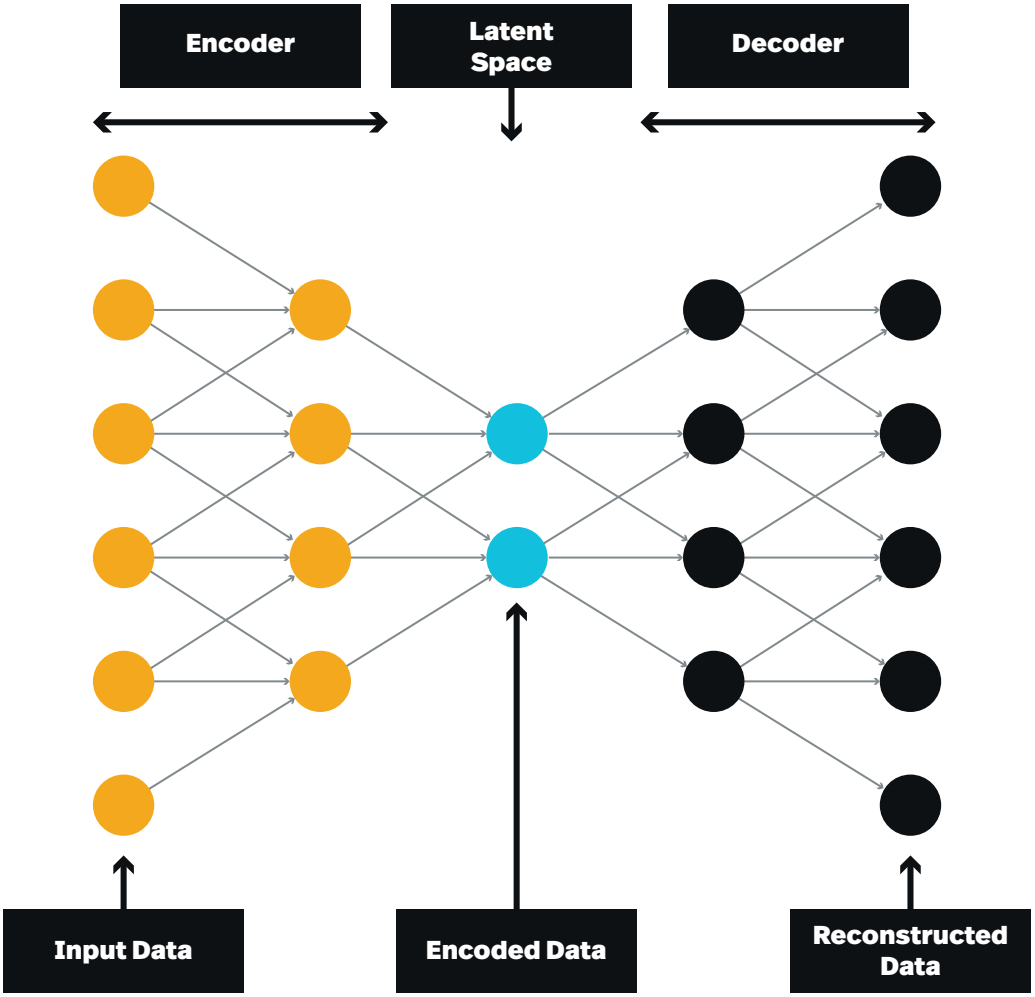
Imagine your facility has bearings that operate under a high vibrational frequency with a short lifespan. Others operate under low vibrational frequency and last longer. If the frequency and lifespan of these two types of bearings are different enough, you might expect to see two different clusters when you plot individual bearings on an x (lifespan) and y (frequency) plot. Using this information, you can group future bearings into one of these different clusters to estimate the lifespan of the bearing and to determine when to schedule maintenance.

Uncovering this data helps to determine an optimal timeframe to replace bearings before the point of failure – while also avoiding premature replacements, which increase operational costs.

It’s important to note that the clustering algorithms you choose to use may not adequately differentiate between clusters. If you use these algorithms and don’t notice the clustering error, you may incorrectly estimate the life of a bearing, potentially leading to an unexpected failure.

Autoencoders

Autoencoders are artificial neural networks that encode input data into a hidden layer and then decode it as output data. These networks learn what features are important to make predictions – without being told what those features are ahead of time. They are also good at finding outliers within datasets for more sophisticated anomaly detection.



Supervised Learning Maximizes Predictive Potential

Supervised learning uses labeled data to train machine learning models. In an industrial setting, this data can represent specific failure states of the machines we're monitoring. However, labeling by hand has the potential to allow bias to interfere. Labeling may also require domain expertise that makes it difficult to scale.

Automatic Labeling

In some instances, we do not have a known output on which to train the model. If this is the case, we must label the data ourselves. Manually labeling data is often time-consuming and expensive, which is why it is best to label data systematically when possible. In some cases, however, it is necessary.

When we label data ourselves, we must be cautious. Labeling data in an unsupervised way – such as grouping bearings together by how long they last, as in the above example – can introduce bias. Let's say we decide ahead of time that bearings with lifetimes less than 100 hours are "bad" and bearings with lifetimes greater than 100 hours are "good," then we introduce bias into the data, and our model will learn from the bias instead of from the actual data. When we label data in a supervised manner – such as labeling failed bearings as "bad" and non-failed bearings as "good" – we can introduce bias into our data and still get good results from our model. However, this *does not* mean that we should introduce bias into our data!

It can at times be better to label data in an unsupervised manner and then use supervised machine learning on top of it. In this way, our model will learn what features are important for making predictions without being biased by us ahead of time. But, if you do plan to use an unsupervised classifier to label, be sure to check the labels for accuracy first. You'll introduce incorrect labeling and train your model incorrectly if your classifier is wrong. This is just one example of why you want to exercise careful testing in deployment.

Alternatively, you can label data by hand using a structured system. You could build an application requiring operators to input a failure code when a machine goes offline. You can then use this code along with the timestamp to label the sensor data generated.

Modeling Predictive Maintenance

In general, you'll use one of two predictive modeling approaches when working with labeled data.

Regression models predict the remaining useful life of a component. They estimate how much time we have left before failure. For a regression model to work, you'll need as much historical data as possible. You should track every event and, ideally, represent every type of failure.

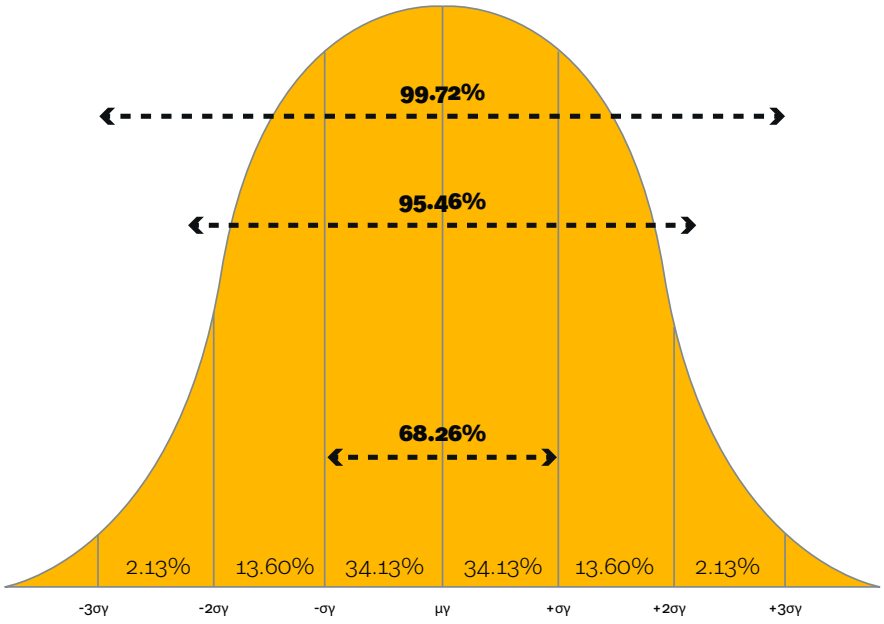
A regression model assumes that, based on the inherent (static) aspects of a system and its performance in the present, the remaining lifecycle is predictable. However, if there are several ways in which a system can fail, we must create a separate model for each possibility.

Classification models predict machine failure within a specific window of time. In this scenario, you don't need to know far in advance when or if a machine will fail – only that failure is imminent. This model supports multiple types of failure, allowing you to group incidents under the same classification. The success of a classification model depends on availability of substantive data and notable features denoting each type of failure to inform the ML model.

Regression and classification models are similar in many ways, but they fundamentally differ in that regression output variables are continuous while classification output variables are class labels.

Non-Gaussian Distributed Input Data

Because many ML algorithms assume your data is Gaussian by default, they do not work well when the data does not conform to a Gaussian (or normal) distribution. Small sample sizes often result in data with skewed, non-Gaussian distributions. So start by confirming that you have an adequate number of data points.



An example of a normal curve

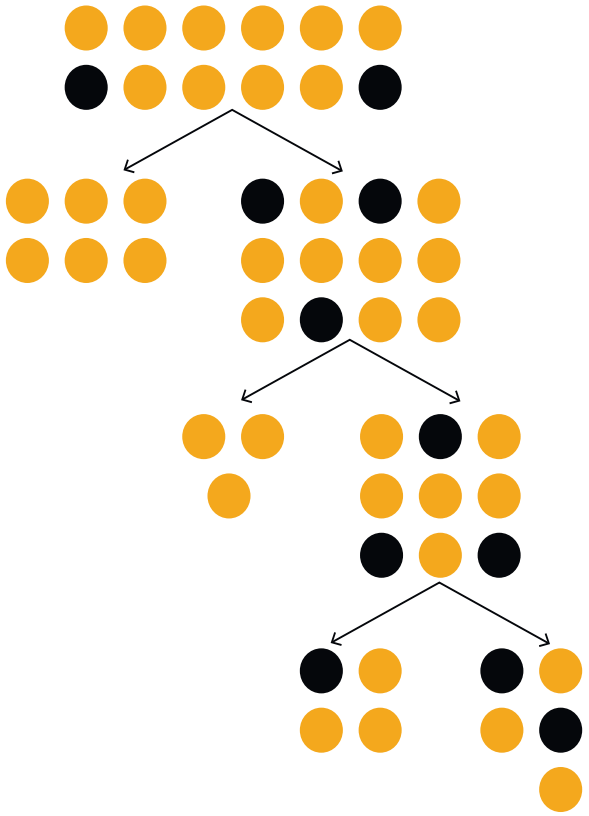
SOURCE: https://blog.antiaging.com/tyranny-tsh-thyroid-testing/

If your input data continues to remain skewed, you may wish to address this problem during feature engineering by using mathematical transformations to make the data more Gaussian. Alternatively, you could find a resolution during model selection by choosing a model (such as a tree-based model) that does not depend on Gaussian inputs.

Class/Label Imbalance

A class imbalance problem presents a unique challenge. Fraud detection and part failures could result in an imbalance due to having many examples of successful behavior and only a few examples of failure. When discussing this imbalance, we often represent it as a binary system where results can fall into one of two classes. The severity of an imbalance might be slight, but it could also be as high as 1:10,000 or more.

Fortunately, there are various ways to tackle class imbalance during training to help reduce a model's bias towards a particularly over-represented class. A simple, if naive, approach would be to oversample your underpopulated class until the class becomes proportional to the over-represented class. More sophisticated methods are available, such as Synthetic Minority Oversampling Technique (SMOTE), which generates new synthetic data points for the under-represented class.



An example of a class imbalance in a tree-based model

SOURCE: https://www.jeremyjordan.me/imbalanced-data/

Enjoy a more predictable future.

Industry 4.0 changes the way we build and manage production facilities, but too many companies get stuck trying to implement predictive maintenance.

LEARN MORE ABOUT EXTRACTING THE MOST OUT OF YOUR DATA AND PUTTING MACHINE LEARNING TO WORK FOR YOU.

REACH OUT TODAY →