A major problem faced by businesses in asset-heavy industries such as manufacturing is the significant costs associated with delays in the production process due to mechanical problems. Most of these businesses are interested in predicting these problems in advance so that they can proactively fix these issues before they occur which will reduce the costly impact caused by downtime.

The business problem for this example is about predicting problems caused by component failures such that the question "What is the probability that a machine will fail in the near future due to a failure of a certain component?" can be answered. The problem is formatted as a multi-class classification problem and a machine learning algorithm is used to create the predictive model that learns from historical data collected from machines.

Datasets:

Common data sources for predictive maintenance problems are:

(1) Failure history: The failure history of a machine or component within the machine.

(2) Maintenance history: The repair history of a machine, e.g. error codes, previous maintenance activities or component replacements.

(3) Machine conditions and usage: The operating conditions of a machine e.g. data collected from sensors.

(4) Machine features: The features of a machine, e.g. engine size, make and model, location.

(5) Operator features: The features of the operator, e.g. gender, past experience.

The data for this example comes from 4 different sources which are real-time telemetry data collected from machines, error messages, historical maintenance records that include failures and machine information such as type and age.

The first data source is the telemetry time-series data which consists of **voltage, rotation, pressure, and vibration** measurements collected from 100 machines in **real time averaged over every hour collected during the year 2015**.

The second major data source is the error logs. These are **non-breaking errors thrown while the machine is still operational and do not constitute as failures.** The **error date and times** are rounded to the closest hour since the telemetry data is collected at an hourly rate.

Maintenance are the **scheduled and unscheduled** maintenance records which correspond to both **regular inspection of components as well as failures.** A **record is generated if a component is replaced during the scheduled inspection or replaced due to a breakdown.** The **records that are created due to breakdowns will be called failures**. Maintenance data has both 2014 and 2015 records.

Machines data set includes some information about the machines: model type and age.

Failures are the records of component replacements **due to failures.** Each record has a **date and time, machine ID, and failed component type.**

The first step in predictive maintenance applications is feature engineering which requires bringing the different data sources together to create features that best describe a machines health condition at a given point in time.

**Lag Features from Telemetry**

Telemetry data almost always comes with time-stamps which makes it suitable for calculating lagging features. A common method is to pick a window size for the lag features to be created and compute rolling aggregate measures such as mean, standard deviation, minimum, maximum, etc. to represent the short term history of the telemetry over the lag window. The rolling mean and standard deviation of the telemetry data over the last 3 hour lag window is calculated for every 3 hours.

For capturing a longer term effect, 24 hour lag features are also calculated.

Next, the columns of the feature datasets created earlier are merged to create the final feature set from telemetry.

Like telemetry data, errors come with timestamps. An important difference is that the **error IDs are categorical values** and **should not be averaged over time intervals like the telemetry measurements.** Instead, we count the number of errors of each type in a **lagging window. We begin by reformatting the error data** to have one entry per machine per time at which at least one error occurred.

Finally, we can compute the total number of errors of each type over the last 24 hours, for time points taken every three hours.

Creating lagging features from **maintenance data** is not as straightforward as for telemetry and errors, so the features from this data are generated in a more custom way. This type of ad-hoc feature engineering is very common in predictive maintenance since domain knowledge plays a big role in understanding the predictors of a problem. So, the days since last component replacement are calculated for each component type as features from the maintenance data.

# **Training, Validation and Testing**

# When working with time-stamped data as in this example, record partitioning into training, validation, and test sets should be performed carefully to prevent overestimating the performance of the models. In predictive maintenance, the features are usually generated using lagging aggregates: records in the same time window will likely have identical labels and similar feature values. These correlations can give a model an "unfair advantage" when predicting on a test set record that shares its time window with a training set record. We therefore partition records into training, validation, and test sets in large chunks, to minimize the number of time intervals shared between them.

Predictive models have no advance knowledge of future chronological trends: in practice, such trends are likely to exist and to adversely impact the model's performance. To obtain an accurate assessment of a predictive model's performance, we recommend training on older records and validating/testing using newer records.

For both of these reasons, a time-dependent record splitting strategy is an excellent choice for predictive maintenance models. The split is effected by choosing a point in time based on the desired size of the training and test sets: all records before the time point are used for training the model, and all remaining records are used for testing. (If desired, the timeline could be further divided to create validation sets for parameter selection.) To prevent any records in the training set from sharing time windows with the records in the test set, we remove any records at the boundary -- in this case, by ignoring 24 hours' worth of data prior to the time point.