ML based Predictive Maintenance at IoT Edge

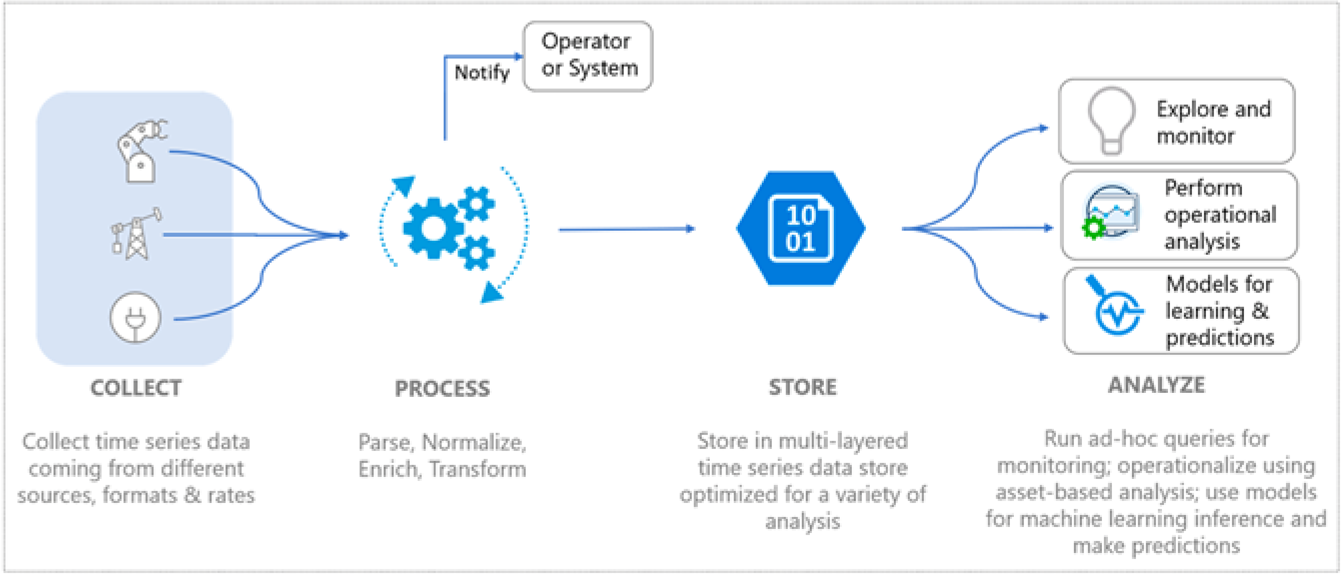
Dr. Rajiv Misra

Professor, Dept. of Computer Science & Engg. Indian Institute of Technology Patna rajivm@iitp.ac.in

Lecture Overview

- In this lecture, we combine the Machine Learning (ML) and IoT together.
- The primary objective of this lecture is to introduce the processing of IoT data with machine learning, specifically on the edge.
- While we touch many aspects of a general machine learning workflow, this lecture is not intended as an in-depth introduction to machine learning
- We do not attempt to create a highly optimized model for the use case, it just illustrates the process of creating and using a viable model for IoT data processing.

ML Development at IoT Edge


Notify
Operator or System
10 01
Explore and monitor
Perform operational analysis
Models for learning & predictions
COLLECT
Collect time series data coming from different sources, formats & rates
PROCESS
Parse, Normalize, Enrich, Transform
STORE
Store in multi-layered time series data store optimized for a variety of analysis
ANALYZE
Run ad-hoc queries for monitoring; operationalize using asset-based analysis; use models for machine learning inference and make predictions

Machine Learning: Background

Artificial intelligence (A.I.) is defined as the property of machines that mimic human intelligence as characterized by behaviours such as cognitive ability, memory, learning, and decision making.

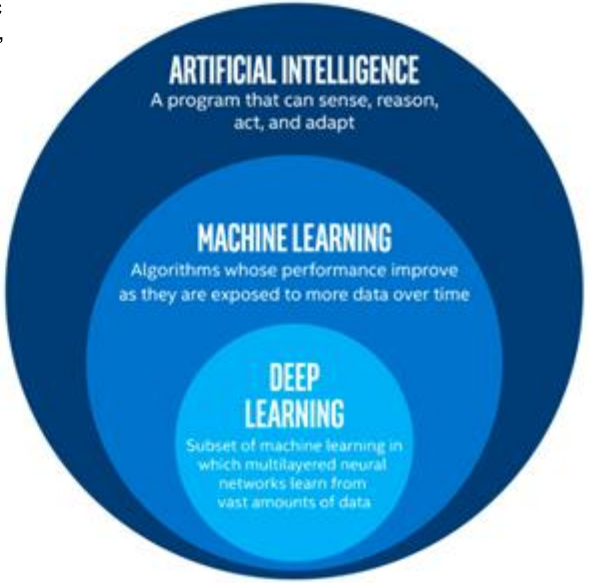
Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

"Deep" machine learning can use labeled datasets, also known as supervised learning, to inform its algorithm, but it doesn't necessarily require a labeled dataset.

Deep learning can ingest unstructured data in its raw form (e.g., text or images), and it can automatically determine the set of features which distinguish different categories of data from one another.

The "deep" in deep learning is just referring to the number of layers in a neural network.

"Non-deep", machine learning is more dependent on human intervention to learn. Human experts determine the set of features to understand the differences between data inputs, usually requiring more structured data to learn.

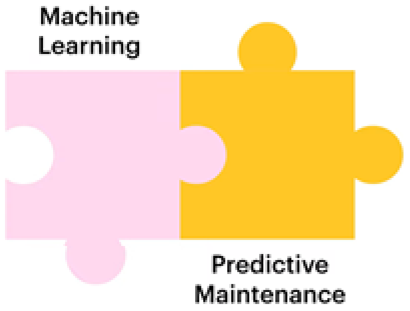

ARTIFICIAL INTELLIGENCE
A program that can sense, reason, act, and adapt
MACHINE LEARNING
Algorithms whose performance improve as they are exposed to more data over time
DEEP LEARNING
Subset of machine learning in which multilayered neural networks learn from vast amounts of data

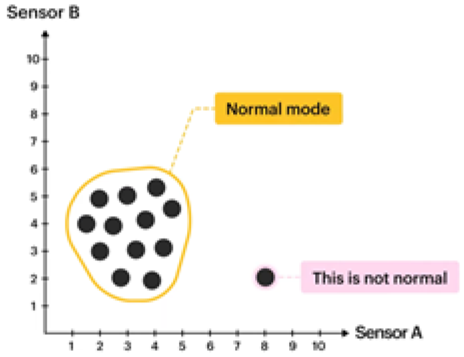
ML for Predictive Maintenance: Example

Using simple machine learning techniques we can create a simple model of a machine with normal operating conditions for any application and determine the values that fall outside of that normal area.

Example: Train a model for the motor vibration with two sensors namely A and B, in normal operating conditions. That means, using normal data points, model has good understanding of what the motor vibration value could be approximately when the motor is operating in normal mode and without any problems.

Now, let's say, one day at a random point in time, model observes that the value of sensor A is 8, and at the same time, the value for sensor B is 2. This is clearly an unusual value. The trained model can easily say that this new value is not normal and can indicate that there might be something wrong with the motor. This is how machine learning works to detect the unusual behavior of a machine.

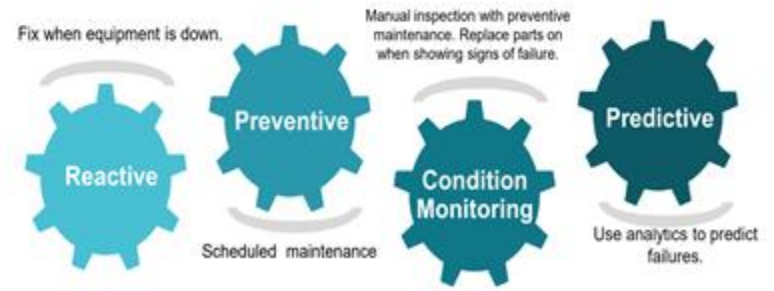




Predictive Maintenance: Introduction

In the past, companies have used reactive maintenance, which focused on preparing an asset once failures had occurred.

Then they moved to preventive maintenance, also known as the schedule-based or planned maintenance. This refers to performing periodic maintenance based on manufacturers' recommendation. The focus was on reducing the failures by replacing parts based on worst case lifetimes for critical pieces of manufacturing tooling.

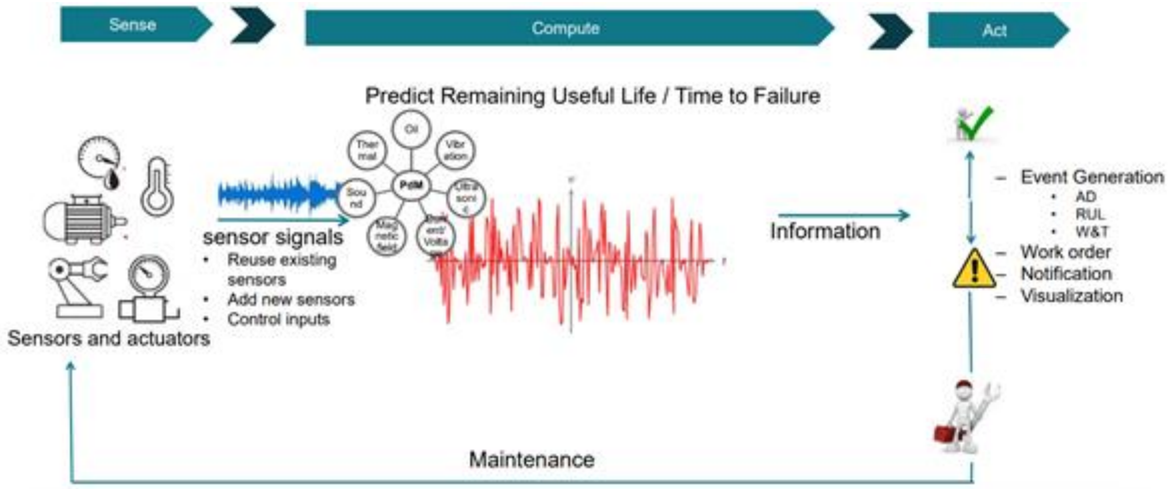

Fix when equipment is down.
Reactive
Preventive
Scheduled maintenance
Manual inspection with preventive maintenance. Replace parts on when showing signs of failure.
Condition Monitoring
Predictive
Use analytics to predict failures.

Next came condition-based maintenance methods, which repairs or replaces equipment when they begin to show signs of failure. However, this condition-based method requires an experienced maintenance team to inspect the equipment at regular intervals.

With the explosion of computers and sensors, companies are now engaging in machine-led condition-based maintenance to reduce costs while improving the uptime of factories. Predictive maintenance takes condition-based maintenance a step father. In this methodology, machine learning analytics are used to predict a machine's failure early by examining the real-time sensor data and detecting changes in machine health status.

Predictive Maintenance: Introduction

Predictive maintenance employs advanced analytics, on the machine data collected from end sensor nodes to draw meaningful insights that more accurately predict machine failures. It is comprised of three steps; sense, compute, and act.



Predictive Maintenance: Introduction

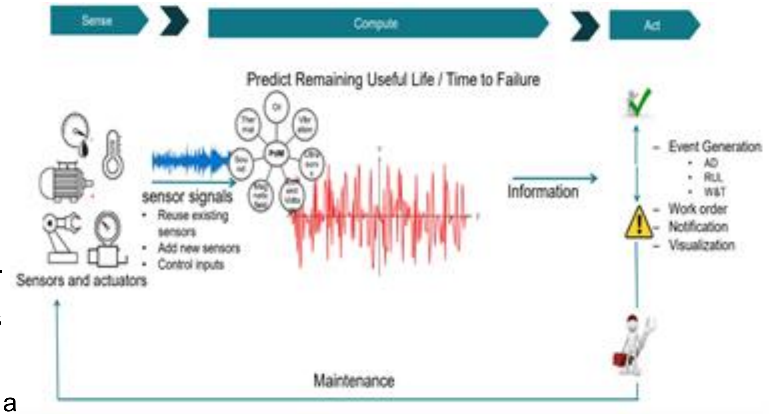
Data is collected from sensors that are already available in machines or by adding new sensors, or by using control inputs.

Depending upon the machine types and the required failure analysis, different sensor signals, such as temperature, sound, magnetic field, current, voltage, ultrasonic, vibration are analyzed to predict the failure.

The predicted information from sensor data analysis is used to generate an event, work order, and notification.

The sensor data is also used to visualize the machine's overall operating condition.

An action is taken when the event reports an anomaly, a machine that is nearing the end of its useful life, or when wear and tear is detected in machine parts.



Predictive Maintenance: Problems

Will this equipment fail in a given period of time?

What is the remaining useful life or the time to failure?

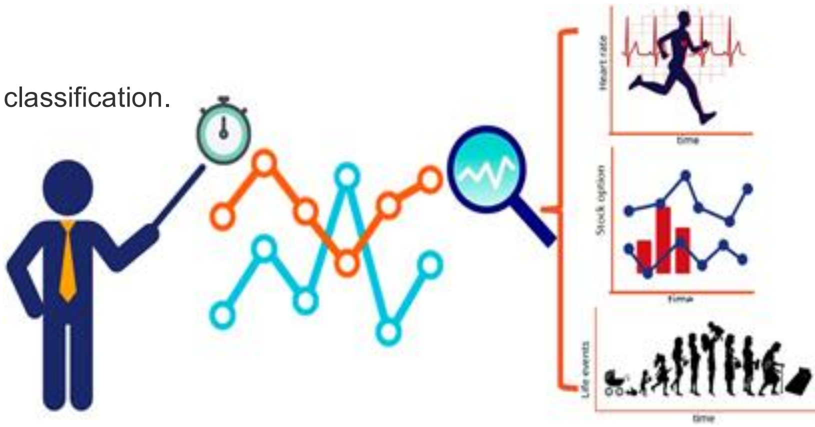
How to quantify wear and tear of expandable components.

- This is a subset of remaining useful life and focuses on shorter living subsystems.

For detecting anomalies in equipment behavior.

- With further analysis, it can provide failure classification.

To optimize equipment settings.



Machine Learning Workflow: Predictive Maintenance

A six-step process:

Define the problem and the outcome

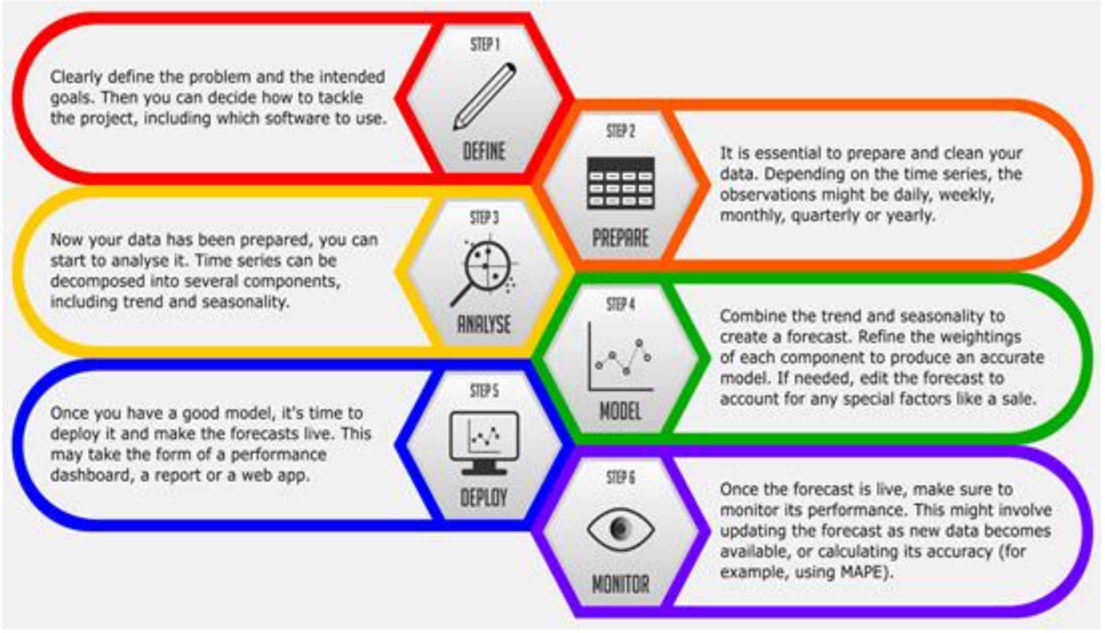
Prepare the data

Analyse the time series

Model the predict using insight from the analysis

Deploy the predictive model

Monitor the predictive performance


STEP 1
DEFINE
Clearly define the problem and the intended goals. Then you can decide how to tackle the project, including which software to use.
STEP 2
PREPARE
It is essential to prepare and clean your data. Depending on the time series, the observations might be daily, weekly, monthly, quarterly or yearly.
STEP 3
ANALYSE
Now your data has been prepared, you can start to analyse it. Time series can be decomposed into several components, including trend and seasonality.
STEP 4
MODEL
Combine the trend and seasonality to create a forecast. Refine the weightings of each component to produce an accurate model. If needed, edit the forecast to account for any special factors like a sale.
STEP 5
DEPLOY
Once you have a good model, it's time to deploy it and make the forecasts live. This may take the form of a performance dashboard, a report or a web app.
STEP 6
MONITOR
Once the forecast is live, make sure to monitor its performance. This might involve updating the forecast as new data becomes available, or calculating its accuracy (for example, using MAPE).

Machine Learning Workflow: Define

As in any successful project, the first step is to clearly define the problem.

This includes the motivation behind creating a predictive and the intended goals and outcomes.

After this, you can decide how to tackle the task at hand, including which software to use at each stage.

For example, we might use Excel for data preparation, R for the analysis and modelling and Power BI for deployment.

Machine Learning Workflow: Prepare

It is essential to properly prepare and clean the data that will be used to create the prediction.

Data cleansing might involve removing duplicated or inaccurate records, or dealing with missing data points or outliers.

In the case of a predictive maintenance project, the data will take the form of a time series.

Depending on what is being predicted, the observations might be daily, weekly, monthly, quarterly or yearly.

Machine Learning Workflow: Analyse

Once the data has been prepared, the next step is to analyse it.

For a time series, this involves decomposing the series into its constituent parts. These include trend and seasonal effects.

The trend is the long-term overall pattern of the data and is not necessarily linear.

Seasonality is a recurring pattern of a fixed length which is caused by seasonal factors.

Machine Learning Workflow: Modelelling

Predictions are created by combining the trend and seasonality components.

There are functions that can do this for you in Excel, or it can be done by hand in a statistical package like R.

If modelling manually, refine the weightings of each component to produce a more accurate model.

The model can be edited to account for any special factors that need to be included.

However, be careful to avoid introducing bias into the prediction and making it less accurate.

Whether using Excel or R, your model will include prediction intervals (or confidence intervals). These show the level of uncertainty in the prediction at each future point.

Machine Learning Workflow: Deploy

Once you are happy with your model, it's time to deploy it and make the predictions live.

This means that decision makers within the business or organisation can utilise and benefit from your predictions.

Deployment may take the form of a visualisation, a performance dashboard, a graphic or table in a report, or a web application.

You may wish to include with the prediction intervals calculated in the previous step.

These show the user the limits within which each future value can be expected to fall between if your model is correct.

Machine Learning Workflow: Monitor

After the prediction goes live, it is important to monitor its performance.

A common way of doing this is to calculate the accuracy using an error measurement statistic.

Popular measures include the *mean absolute percentage error (MAPE)* and the *mean absolute deviation (MAD)*.

Depending on what is being predicted, it may be possible to update your model as new data becomes available.

This should also lead to a more accurate prediction of future values.

Machine Learning Methods: Predictive Maintenance

Problem definition: Classification and Regression approach

– Classification: Will it fail?
- Multi-class classification: Will it fail for reason X?

– Regression: After how long will it fail?

• Methods:

– Traditional machine learning:
- Decision trees: Random forests, gradient boosting trees, isolation forest
- SVM (Support Vector Machines)

– Deep learning approach:
- CNN (Convolution Neural Network)/Multilayer Perceptrons (MLPs)
- RNN (Recurrent Neural Network)/LGTM (Long Short Term Memory)/GRU (Gated Recurrent Unit)

– Hybrid of deep learning and Physics-Based Modeling (PBM):
- Use PBM to generate training data where lacking
- Use PBM to reduce the problem space (feature engineering)
- Use PBM to inform and validate DL models (e.g., to identify catastrophic failures, most notably in scenarios with low amounts of training data and a high degree of mission criticality)

Deep Learning Methods

Deep learning has proven to show superior performance in certain domains such as object recognition and image classification.

It has also gained popularity in domains such as finance where time-series data plays an important role.

Predictive Maintenance is also a domain where data is collected over time to monitor the state of an asset with the goal of finding patterns to predict failures which can also benefit from certain deep learning algorithms.

Among the deep learning methods, Long Short Term Memory (LSTM) networks are especially appealing to the predictive maintenance domain due to the fact that they are very good at learning from sequences.

This fact lends itself to their applications using time series data by making it possible to look back for longer periods of time to detect failure patterns.

Deep Learning Methods: Multilayer Perceptrons (MLPs)

Generally, neural networks like Multilayer Perceptrons or MLPs provide capabilities that are offered by few algorithms, such as:

- **Robust to Noise.** Neural networks are robust to noise in input data and in the mapping function and can even support learning and prediction in the presence of missing values.
- **Nonlinear.** Neural networks do not make strong assumptions about the mapping function and readily learn linear and nonlinear relationships.
- **Multivariate Inputs**. An arbitrary number of input features can be specified, providing direct support for multivariate forecasting.
- **Multi-step Forecasts**. An arbitrary number of output values can be specified, providing direct support for multi-step and even multivariate forecasting.

For these capabilities alone, feedforward neural networks may be useful for time series forecasting.

Deep Learning Methods: Convolutional Neural Networks (CNNs)

Convolutional Neural Networks or CNNs are a type of neural network that was designed to efficiently handle image data.

The ability of CNNs to learn and automatically extract features from raw input data can be applied to time series forecasting problems. A sequence of observations can be treated like a one-dimensional image that a CNN model can read and distill into the most salient elements.

- **Feature Learning.** Automatic identification, extraction and distillation of salient features from raw input data that pertain directly to the prediction problem that is being modeled.

CNNs get the benefits of Multilayer Perceptrons for time series forecasting, namely support for multivariate input, multivariate output and learning arbitrary but complex functional relationships, but do not require that the model learn directly from lag observations. Instead, the model can learn a representation from a large input sequence that is most relevant for the prediction problem.

Deep Learning Methods: Long Short-Term Memory Networks (LSTMs)

Long Short Term Memory networks – usually just called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997), and were refined and popularized by different researchers.

LSTM add the explicit handling of order between observations when learning a mapping function from inputs to outputs, not offered by MLPs or CNNs. They are a type of neural network that adds native support for input data comprised of sequences of observations.

- **Native Support for Sequences.** Recurrent neural networks directly add support for input sequence data.

This capability of LSTMs has been used to great effect in complex natural language processing problems such as neural machine translation where the model must learn the complex interrelationships between words both within a given language and across languages in translating from one language to another.

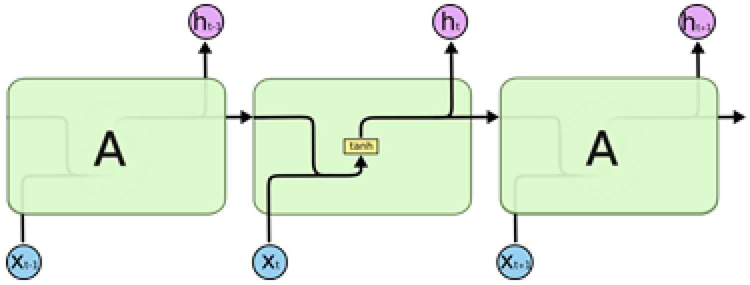
- **Learned Temporal Dependence.** The most relevant context of input observations to the expected output is learned and can change dynamically.

Deep Learning Methods: Long Short-Term Memory Networks (LSTMs)

The model both learns a mapping from inputs to outputs and learns what context from the input sequence is useful for the mapping, and can dynamically change this context as needed.

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior.

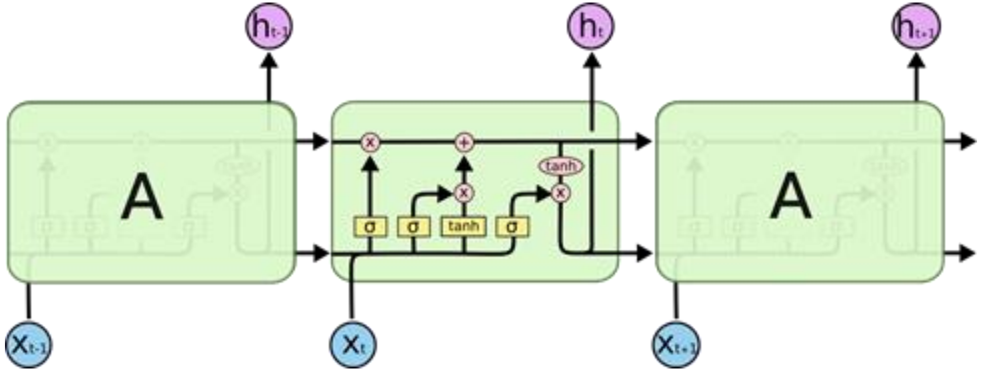
All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.



Deep Learning Methods: Long Short-Term Memory Networks (LSTMs)

LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.

In the below diagram, each line carries an entire vector, from the output of one node to the inputs of others. The pink circles represent pointwise operations, like vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation, while a line forking denote its content being copied and the copies going to different locations.

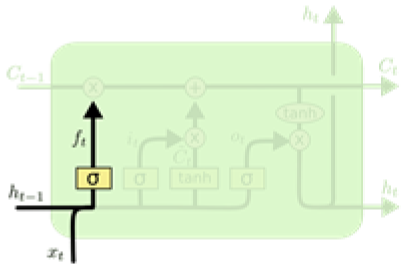


Deep Learning Methods: Long Short-Term Memory Networks (LSTMs)

An LSTM has three of gates, to protect and control the cell state. The first part is called **Forget gate**, the second part is known as the **Input gate** and the last one is the **Output gate**.

Forget Gate: The first step in our LSTM is to decide what information we're going to throw away from the cell state. This decision is made by a sigmoid layer called the "forget gate layer." It looks at h_{t-1} and x_t, and outputs a number between 0 and 1 for each number in the cell state C_{t-1}.

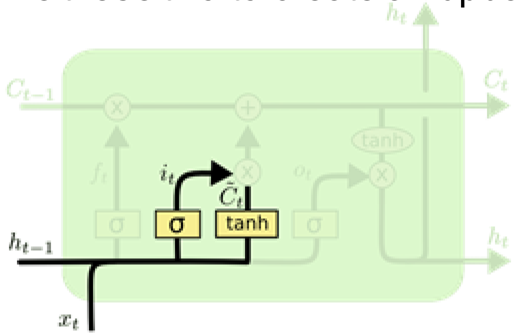
A 1 represents "completely keep this" while a 0 represents "completely get rid of this."



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Deep Learning Methods: Long Short-Term Memory Networks (LSTMs)

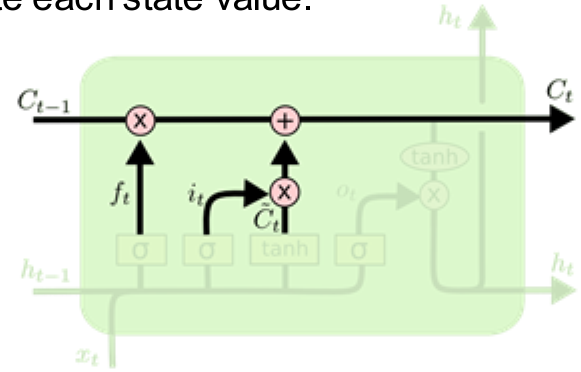
Input Gate: The next step is to decide what new information we're going to store in the cell state. This has two parts. First, a sigmoid layer called the "input gate layer" decides which values we'll update. Next, a *tanh* layer creates a vector of new candidate values, $\tilde{C}_t$, that could be added to the state. In the next step, we'll combine these two to create an update to the state.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Deep Learning Methods: Long Short-Term Memory Networks (LSTMs)

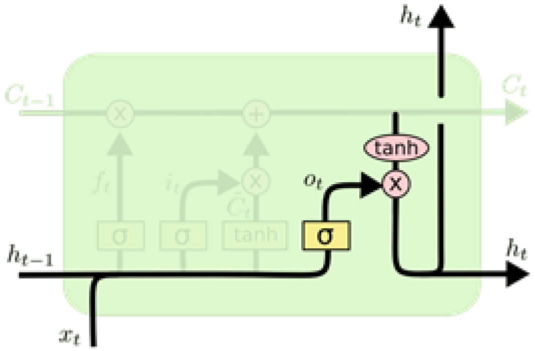
It's now time to update the old cell state, C_{t-1}, into the new cell state C_t. The previous steps already decided what to do, we just need to actually do it. We multiply the old state by f_t, forgetting the things we decided to forget earlier. Then we add $i_t * \tilde{C}_t$. This is the new candidate values, scaled by how much we decided to update each state value.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Deep Learning Methods: Long Short-Term Memory Networks (LSTMs)

Output gate: Finally, we need to decide what we're going to output. This output will be based on our cell state, but will be a filtered version. First, we run a sigmoid layer which decides what parts of the cell state we're going to output. Then, we put the cell state through tanh (to push the values to be between −1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Performance Metric: R-squared

The stationary R-squared is used in time series forecasting as a measure that compares the stationary part of the model to a simple mean model. It is defined as,

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$

Where SS_{res} denotes the sum of squared residuals from expected values and SS_{tot} denotes the sum of squared deviations from the dependent variable's sample mean. It denotes the proportion of the dependent variable's variance that may be explained by the independent variable's variance. A high R^2 value shows that the model's variance is similar to that of the true values, whereas a low R^2 value suggests that the two values are not strongly related.

Performance Metric: Mean Absolute Error (MAE)

The MAE is defined as the average of the absolute difference between forecasted and true values. Where y_i is the expected value and x_i is the actual value (shown below formula). The letter n represents the total number of values in the test set.

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

The MAE shows us how much inaccuracy we should expect from the forecast on average. MAE = 0 means that the anticipated values are correct, and the error statistics are in the original units of the forecasted values.

The lower the MAE value, the better the model; a value of zero indicates that the forecast is error-free. In other words, the model with the lowest MAE is deemed superior when comparing many models.

Performance Metric: Mean Absolute Percentage Error (MAPE)

MAPE is the proportion of the average absolute difference between projected and true values divided by the true value. The anticipated value is F_t, and the true value is A_t. The number n refers to the total number of values in the test set.

$$\mathrm{MAPE} = \frac{100}{n} \sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right|$$

It works better with data that is free of zeros and extreme values because of the in-denominator. The MAPE value also takes an extreme value if this value is exceedingly tiny or huge.

Performance Metric: Mean Squared Error (MSE)

MSE is defined as the average of the error squares. It is also known as the metric that evaluates the quality of a forecasting model or predictor. MSE also takes into account variance (the difference between anticipated values) and bias (the distance of predicted value from its true value).

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y_i})^2$$

Where *y'ᵢ* denotes the predicted value and *yᵢ* denotes the actual value. The number n refers to the total number of values in the test set. MSE is almost always positive, and lower values are preferable. This measure penalizes large errors or outliers more than minor errors due to the square term (as seen in the formula above).

Performance Metric: Root Mean Squared Error(RMSE)

This measure is defined as the square root of mean square error and is an extension of MSE. Where y'_i denotes the predicted value and y_i denotes the actual value. The number n refers to the total number of values in the test set. This statistic, like MSE, penalizes greater errors more.

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}$$

This statistic is likewise always positive, with lower values indicating higher performance. The RMSE number is in the same unit as the projected value, which is an advantage of this technique. In comparison to MSE, this makes it easier to comprehend.

Use Case: Prognostics and Health Management

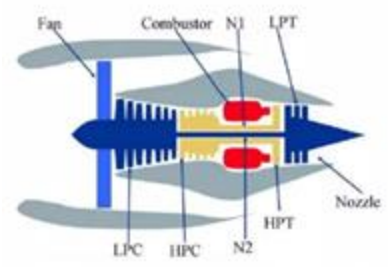
The objective of this use case is to build an LSTM model that can predict the number of remaining operational cycles before failure in the test set, i.e., the number of operational cycles after the last cycle that the engine will continue to operate. Also provided a vector of true Remaining Useful Life (RUL) values for the test data.

The data was generated using C-MAPSS, the commercial version of MAPSS (Modular Aero-Propulsion System Simulation) software. This software provides a flexible turbofan engine simulation environment to conveniently simulate the health, control, and engine parameters.

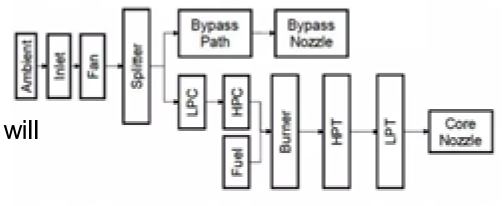
The simulated aircraft sensor values is used to predict two scenarios, so that maintenance can be planned in advance:

* Regression models: The question to ask is "Given these aircraft engine operation and failure events history, can we predict when an in-service engine will fail?"

* Binary classification: We re-formulate this question “Is this engine going to fail within w1 cycles?”



Simplified diagram of engine simulation



A layout showing various modules and their connections as modeled in the simulation

LSTM model: Dataset

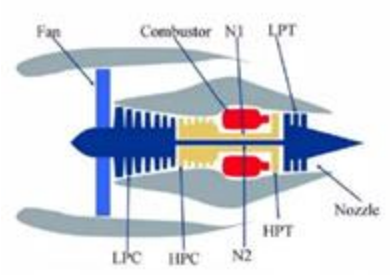
Dataset consists of multiple multivariate time series, such data set is divided into training and test subsets. Each time series is from a different engine. The engine is operating normally at the start of each time series and develops a fault at some point during the series.

In the training set, the fault grows in magnitude until system failure. In the test set, the time series ends some time prior to system failure.
Public dataset (Nasa Turbo fan)

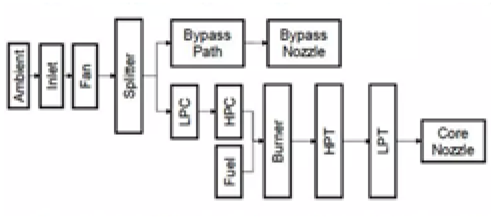
- Damage propagation for aircraft engine
- Run to failure simulation

Aircraft gas turbine. Dataset contains time series (cycles) for all measurements of 100 different engines.

The data used in this use-case is taken from the https://www.nasa.gov/intelligent-systems-division



Simplified diagram of engine simulation



A layout showing various modules and their connections as modeled in the simulation

LSTM model: Data Ingestion

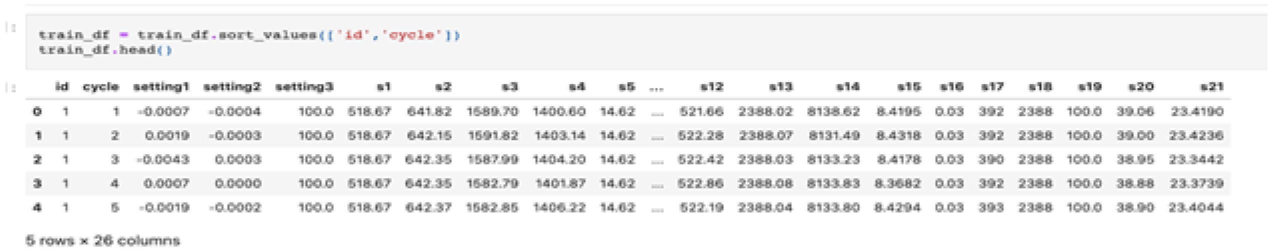
We ingest the training, test and ground truth datasets.

The training data consists of multiple multivariate time series with "cycle" as the time unit, together with 21 sensor readings for each cycle.

Each time series can be assumed as being generated from a different engine of the same type.

The testing data has the same data schema as the training data. The only difference is that the data does not indicate when the failure occurs.

Finally, the ground truth data provides the number of remaining working cycles for the engines in the testing data.



```
train_df = train_df.sort_values(['id','cycle'])
train_df.head()
```

	id	cycle	setting1	setting2	setting3	s1	s2	s3	s4	s5	...	s12	s13	s14	s15	s16	s17	s18	s19	s20	s21
0	1	1	-0.0007	-0.0004	100.0	518.67	641.82	1589.70	1400.60	14.62	...	521.66	2388.02	8138.62	8.4195	0.03	392	2388	100.0	39.06	23.4190
1	1	2	0.0019	-0.0003	100.0	518.67	642.15	1591.82	1403.14	14.62	...	522.28	2388.07	8131.49	8.4318	0.03	392	2388	100.0	39.00	23.4236
2	1	3	-0.0043	0.0003	100.0	518.67	642.35	1587.99	1404.20	14.62	...	522.42	2388.03	8133.23	8.4178	0.03	390	2388	100.0	38.95	23.3442
3	1	4	0.0007	0.0000	100.0	518.67	642.35	1582.79	1401.87	14.62	...	522.86	2388.08	8133.83	8.3682	0.03	392	2388	100.0	38.88	23.3739
4	1	5	-0.0019	-0.0002	100.0	518.67	642.37	1582.85	1406.22	14.62	...	522.19	2388.04	8133.80	8.4294	0.03	393	2388	100.0	38.90	23.4044

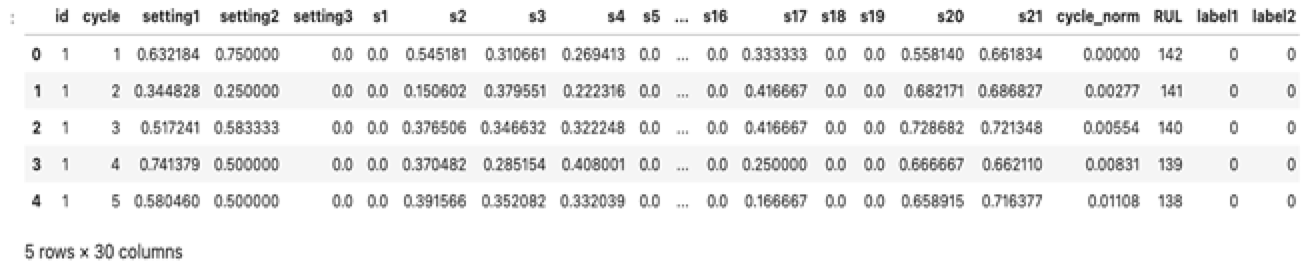
5 rows × 26 columns

LSTM model: Data Preparation and Feature Engineering

First step is to generate labels for the training data which are Remaining Useful Life (RUL), label1 and label2.

Each row can be used as a model training sample where the *s_k* columns are the features and the RUL is the model target. The rows are treated as independent observations and the measurement trends from the previous cycles are ignored. The features are normalized to μ = 0, σ = 1 and PCA is applied.

For the LSTM model, opt for more advanced feature engineering and chose to incorporate the trends from the previous cycles. In this case, each training sample consists of measurements at cycle i as well as i-5, i-10, i-20, i-30, i-40. The model input is a 3D tensor with shape (n, 6, 24) where n is the number of training samples, 6 is the number of cycles (timesteps), and 24 is the number of principal components (features).



	id	cycle	setting1	setting2	setting3	s1	s2	s3	s4	s5	...	s16	s17	s18	s19	s20	s21	cycle_norm	RUL	label1	label2
0	1	1	0.632184	0.750000	0.0	0.0	0.545181	0.310661	0.269413	0.0	...	0.0	0.333333	0.0	0.0	0.558140	0.661834	0.00000	142	0	0
1	1	2	0.344828	0.250000	0.0	0.0	0.150602	0.379551	0.222316	0.0	...	0.0	0.416667	0.0	0.0	0.682171	0.686827	0.00277	141	0	0
2	1	3	0.517241	0.583333	0.0	0.0	0.376506	0.346632	0.322248	0.0	...	0.0	0.416667	0.0	0.0	0.728682	0.721348	0.00554	140	0	0
3	1	4	0.741379	0.500000	0.0	0.0	0.370482	0.285154	0.408001	0.0	...	0.0	0.250000	0.0	0.0	0.666667	0.662110	0.00831	139	0	0
4	1	5	0.580460	0.500000	0.0	0.0	0.391566	0.352082	0.332039	0.0	...	0.0	0.166667	0.0	0.0	0.658915	0.716377	0.01108	138	0	0

5 rows × 30 columns

LSTM model: Modelling

When using LSTMs in the time-series domain, one important parameter to pick is the sequence length which is the window for LSTMs to look back.

This may be viewed as similar to picking window_size = 5 cycles for calculating the rolling features which are rolling mean and rolling standard deviation for 21 sensor values.

The idea of using LSTMs is to let the model extract abstract features out of the sequence of sensor values in the window rather than engineering those manually. The expectation is that if there is a pattern in these sensor values within the window prior to failure, the pattern should be encoded by the LSTM.

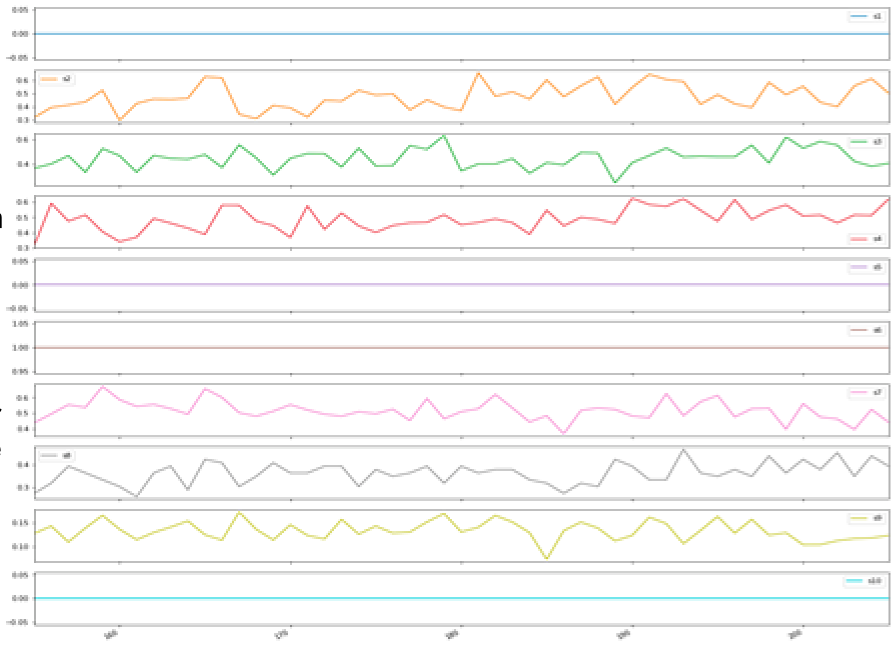
One critical advantage of LSTMs is their ability to remember from long-term sequences (window sizes) which is hard to achieve by traditional feature engineering. For example, computing rolling averages over a window size of 50 cycles may lead to loss of information due to smoothing and abstracting of values over such a long period, instead, using all 50 values as input may provide better results. While feature engineering over large window sizes may not make sense, LSTMs are able to use larger window sizes and use all the information in the window as input.

LSTM model: Modelling

Let's first look at an example of the sensor values 50 cycles prior to the failure for engine id 3.

We will be feeding LSTM network this type of data for each time step for each engine id.

LSTM layers expect an input in the shape of a numpy array of 3 dimensions (samples, time steps, features) where samples is the number of training sequences, time steps is the look back window or sequence length and features is the number of features of each sequence at each time step.



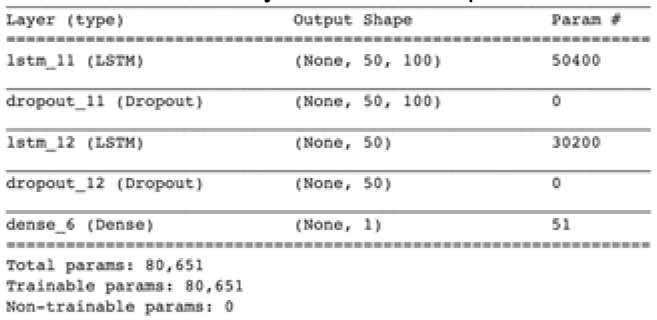
LSTM model: Network Configuration

The first layer is an LSTM layer with 100 units followed by another LSTM layer with 50 units.

Dropout is also applied after each LSTM layer to control overfitting.

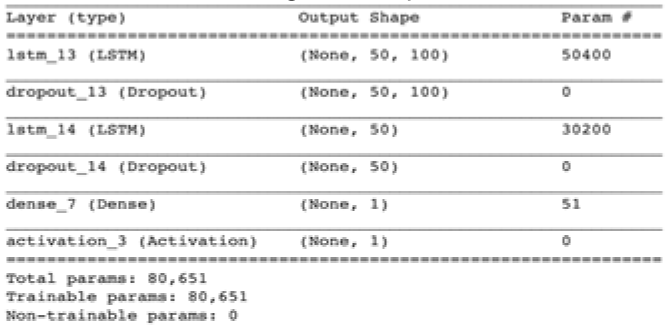
Final layer is a Dense output layer with single unit with sigmoid activation for the binary classification problem and linear activation for the regression problem.

Network for binary classification problem



```
Layer (type)                 Output Shape              Param #
=================================================================
lstm_11 (LSTM)               (None, 50, 100)           50400
_________________________________________________________________
dropout_11 (Dropout)         (None, 50, 100)           0
_________________________________________________________________
lstm_12 (LSTM)               (None, 50)                30200
_________________________________________________________________
dropout_12 (Dropout)         (None, 50)                0
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 51
=================================================================
Total params: 80,651
Trainable params: 80,651
Non-trainable params: 0
```

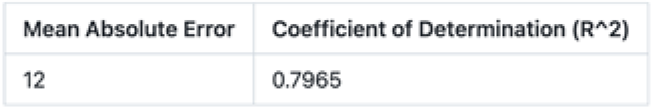
Network for regression problem



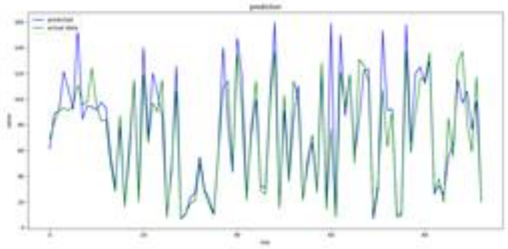
```
Layer (type)                 Output Shape              Param #
=================================================================
lstm_13 (LSTM)               (None, 50, 100)           50400
_________________________________________________________________
dropout_13 (Dropout)         (None, 50, 100)           0
_________________________________________________________________
lstm_14 (LSTM)               (None, 50)                30200
_________________________________________________________________
dropout_14 (Dropout)         (None, 50)                0
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 51
_________________________________________________________________
activation_3 (Activation)    (None, 1)                 0
=================================================================
Total params: 80,651
Trainable params: 80,651
Non-trainable params: 0
```

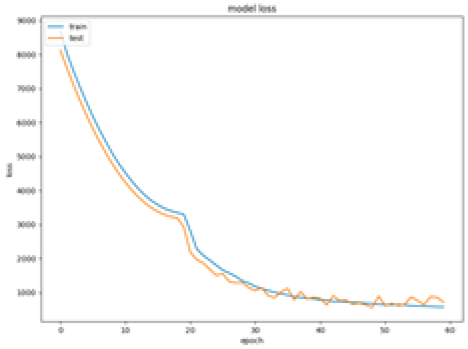
LSTM model: Model Evaluation

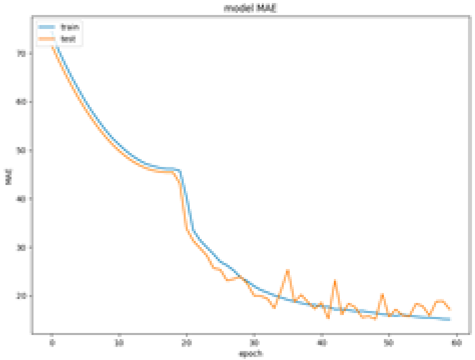
Results of Regression problem:

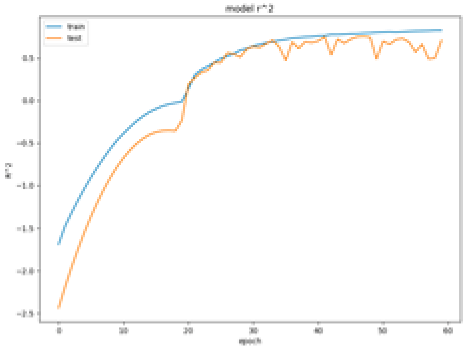


Mean Absolute Error	Coefficient of Determination (R^2)
12	0.7965





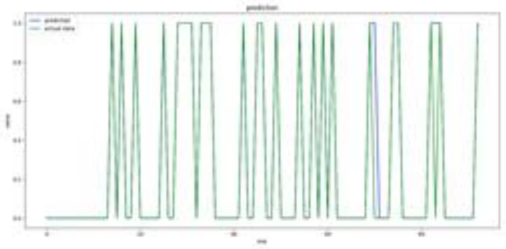


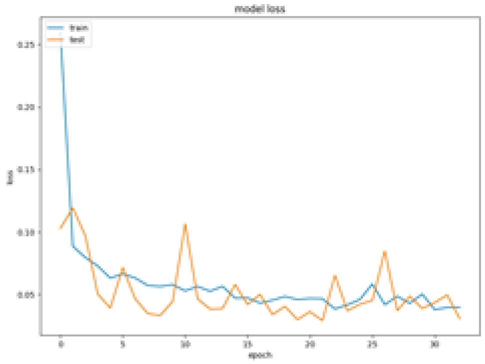


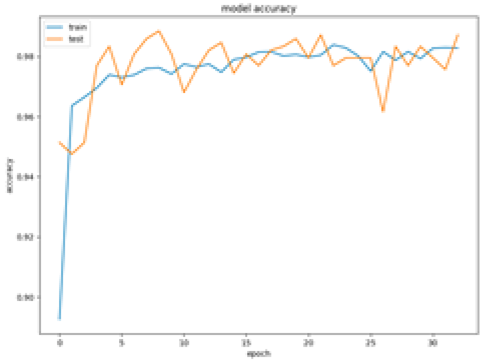
LSTM model: Model Evaluation

Results of Binary Classification problem:

Accuracy	Precision	Recall	F-Score
0.97	0.92	1.0	0.96



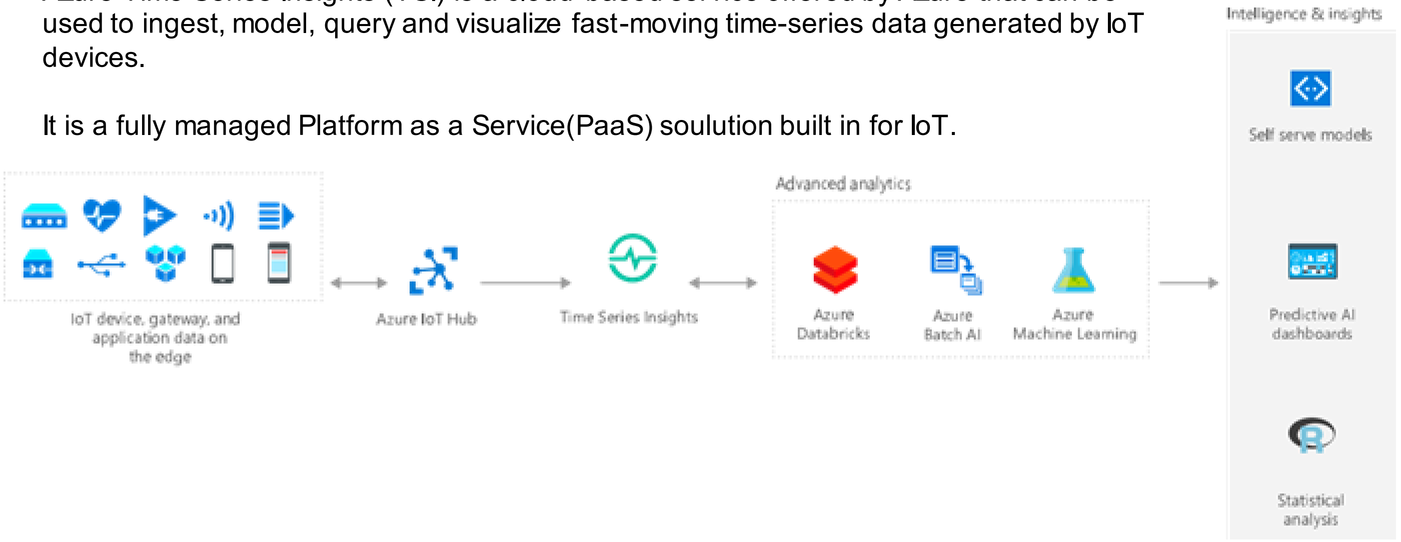




Azure Time Series Insights (PaaS): Predictive Maintenance

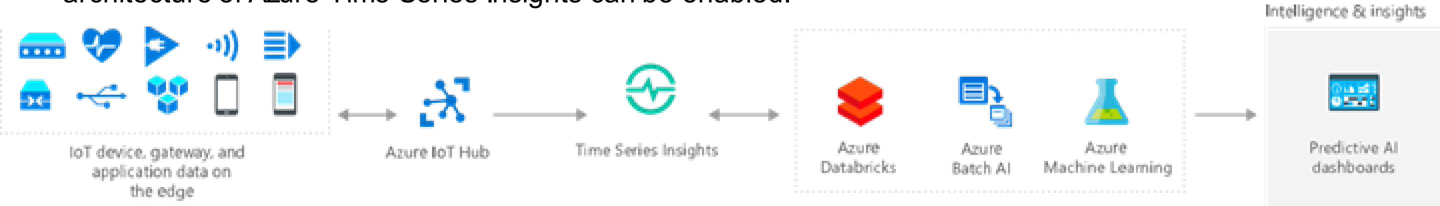
Azure Time Series Insights (TSI) is a cloud-based service offered by Azure that can be used to ingest, model, query and visualize fast-moving time-series data generated by IoT devices.

It is a fully managed Platform as a Service(PaaS) soulution built in for IoT.



Azure Time Series Insights (PaaS): Predictive Maintenance

Real-time data in the form of a time-series can be generated by various devices like mobile devices, sensors, satellites, medical devices etc.
Data from these devices can be fetched to the Azure environment using Azure IoT Hub. Azure IoT hub acts as a data integration pipeline to connect to the source devices and then fetch data and deliver it to the TSI platform.
Once the data is in the TSI, it can then be used for visualization purposes, and can be queried and aggregated accordingly. Additionally, customers can also leverage existing analytics and machine learning capabilities on top of the data available in TSI.
Data from TSI can be further processed using Azure Databricks and machine learning models can be applied based on pre-trained models that will offer predictions in real-time. This is how an overall architecture of Azure Time Series Insights can be enabled.



Azure Time Series Insights (PaaS): Components

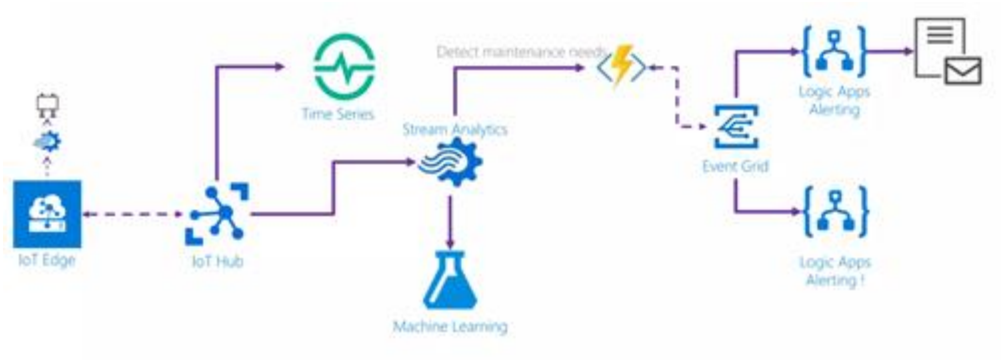
Azure TSI provides the following four components using which users can consume data from varied data sources as follows.

- **Integration** – TSI provides an easy integration from data generated using IoT devices by allowing connection between the cloud data gateways such as Azure IoT Hub and Azure Event Hubs. Data from these sources can be easily consumed in JSON structures, cleaned and then stored in a columnar store
- **Storage** – Azure TSI also takes care of the data that is to be retained in the system for querying and visualizing the data. By default, data is stored on SSDs for fast retrieval and has a data retention policy of 400 days. This supports querying historic data for up to a period of 400 days
- **Data Visualization** – Once the data is fetched from the data sources and stored in the columnar stores, it can be visualized in the form of line charts or heat maps. The visuals are provided out of the box by Azure TSI and can be leveraged for easy visual analysis
- **Query Service** – Although, visualizing the data will answer many questions, however, TSI also provides a query service using which you can integrate TSI into your custom applications.

Usually, a time series data is indexed by timestamps. Therefore, you can build your applications by using TSI as a backend service for integrating and storing the data and using the client SDK for Azure TSI for building the frontend and display visuals like line charts and heat maps.

Predictive Maintenance: Steps

Step 1: Sensor data are collected from edge devices and are forwarded to Azure IoT Hub.
Step 2: Azure IoT hub then drives these gathered data to the TSI platform and Stream Analytics.
Step 3: At TSI, data can be visualised, queried and aggregated with other services.
Step 4: Azure machine learning service provides the training of ML model or using a pretrained model on top of the data available in TSI.
Step 5: Once the training is completed, inference is provided using Azure IoT Hub and Iot Edge service.



Lecture Summary

- Understanding of predictive maintenance
- Machine learning models for predictive maintenance
- Use case of predictive maintenance using LSTM model
- Azure Time Series Insights

THANK YOU!