

---

# Trend Prediction

## IoT Time Series Forecasting

### **Presented By:**

- Hossam Khairullah
- Mohamed Ashraf El-Melegy
- Mostafa Mohamed Fathy

### **Supervised By:**

- Information Technology Institute (ITI)
    - SpimeSenseLabs
-

# Agenda

1. Motivation
2. Dataset
3. Objective & Scope
4. Available Forecasting Techniques
5. Our Model
6. Deployment
7. Live Demo

---

# Motivation



# Motivation

Time Series Forecasting has always been a very important area of research in many domains.

For example:

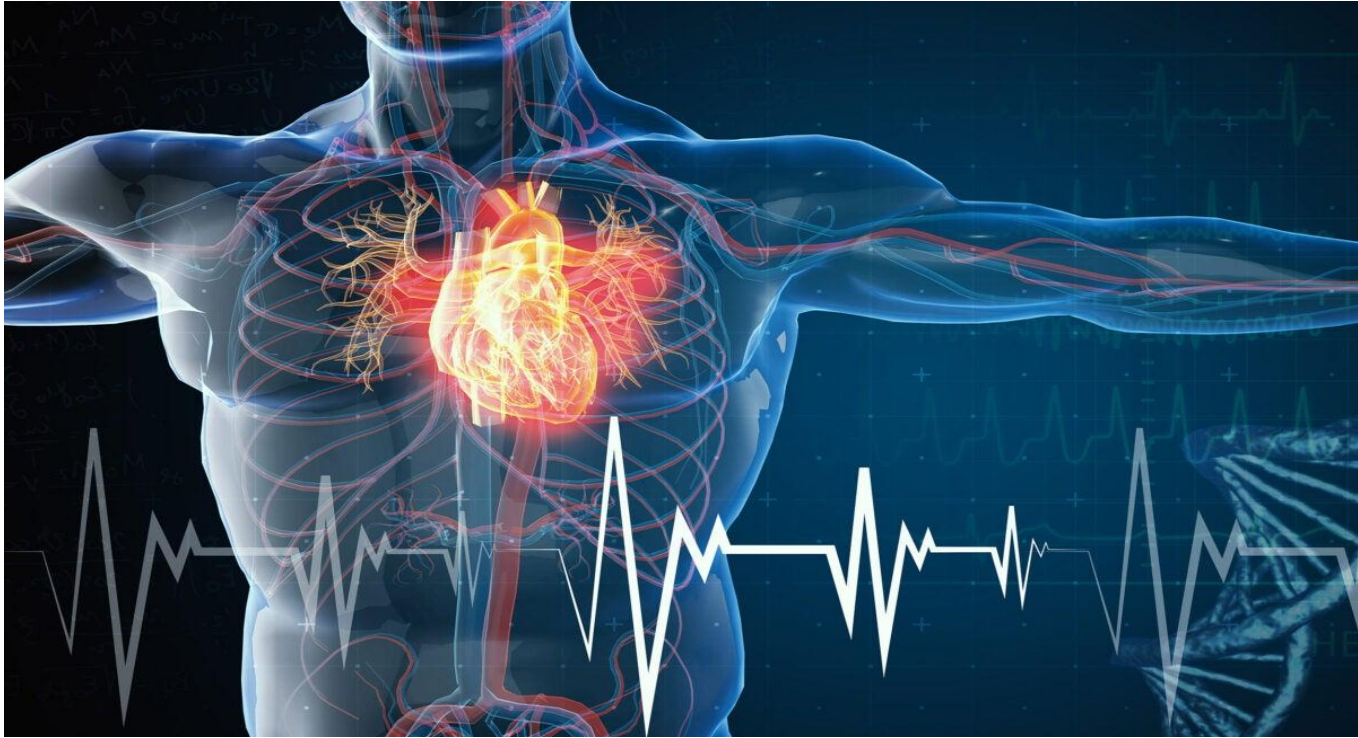
- Manufacturing
- Medicine
- Energy
- Weather
- Smart Cities

# Motivation: Manufacturing





# Motivation: Medicine



# Motivation: Energy

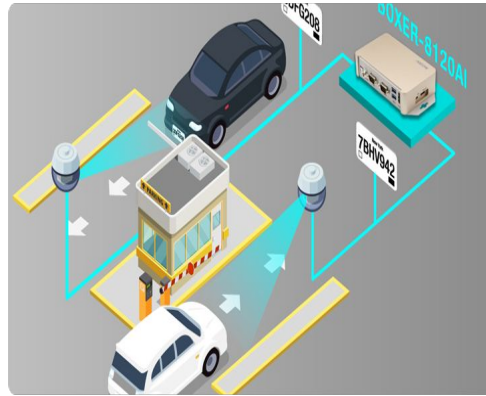


# Motivation: Weather





# Motivation: Smart Cities

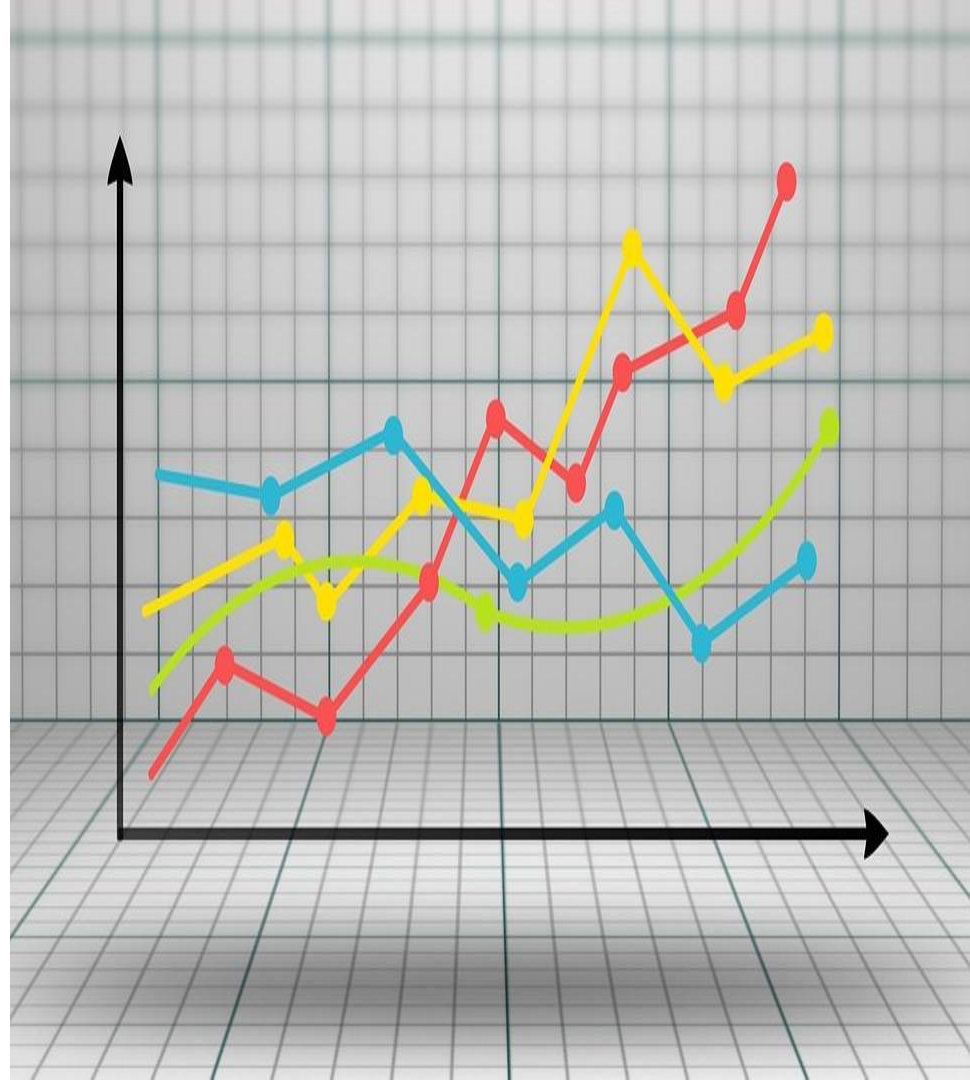


# Time Series Analysis



In mathematics, a series of data points indexed (or listed or graphed) in time order. Most commonly, it is a sequence taken at equally spaced points in time. That's:

# Time Series

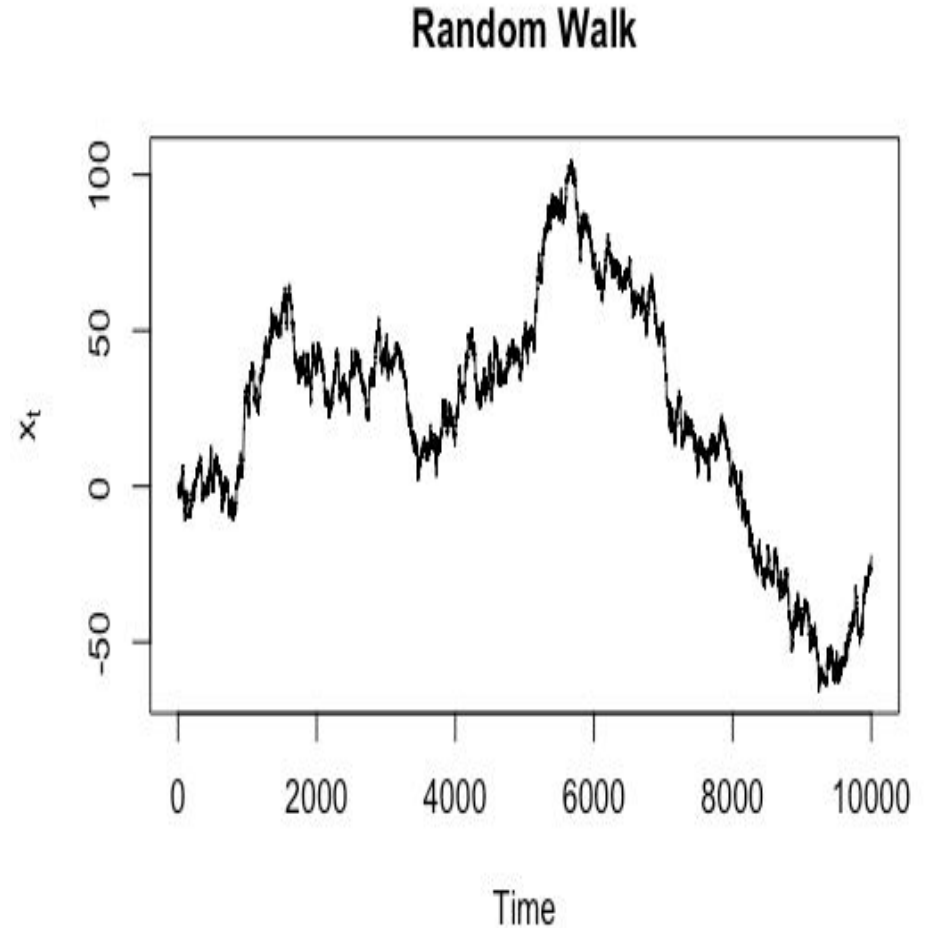


# Time Series: Types

- Non-Stationary
  - ◆ Random Walk
  - ◆ White Noise
  - ◆ Trend Stationary
- Strict Stationary

# Time Series: Types

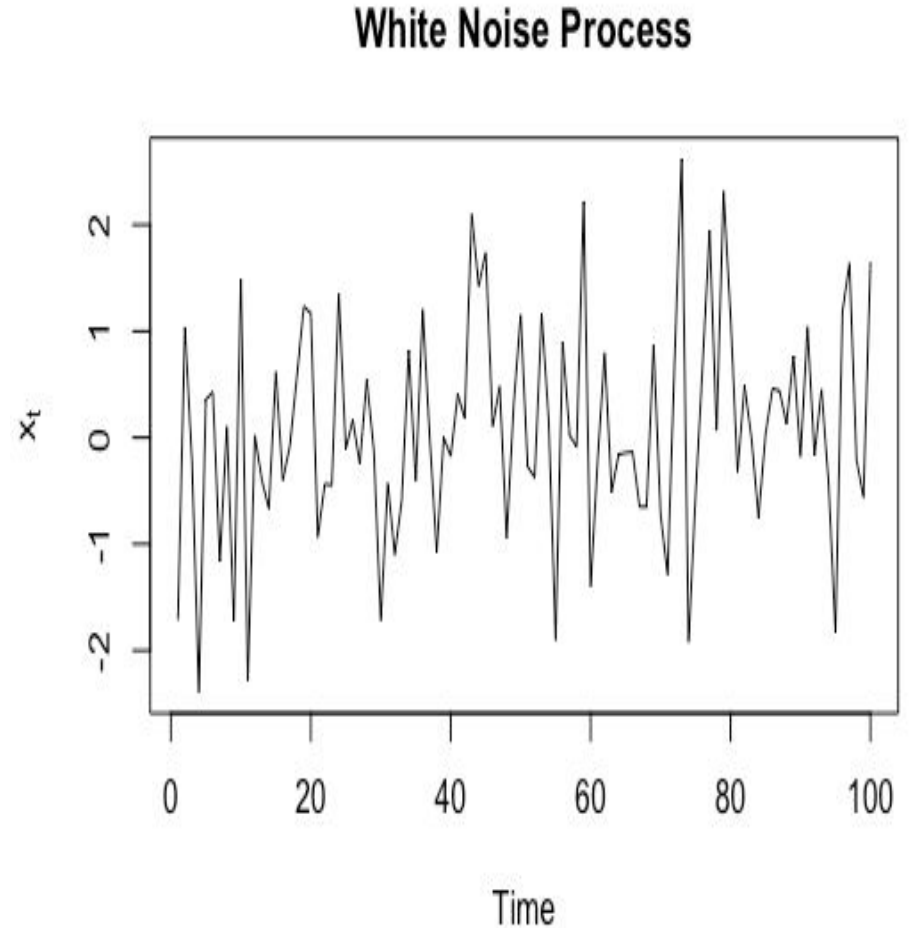
## Random Walk





# Time Series: Types

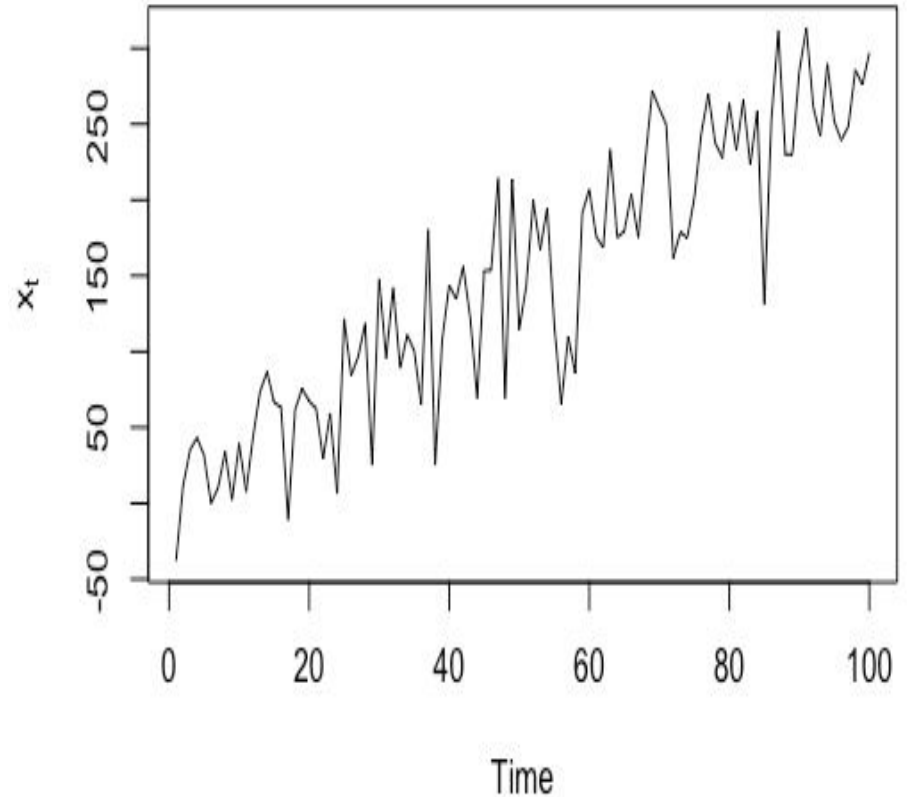
## White Noise



# Time Series: Types

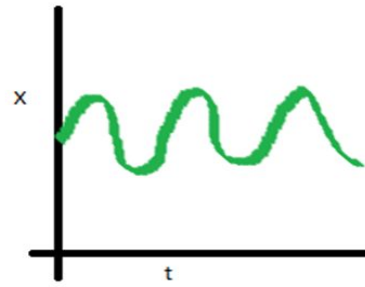
## Trend Stationary

Trend Stationary Process

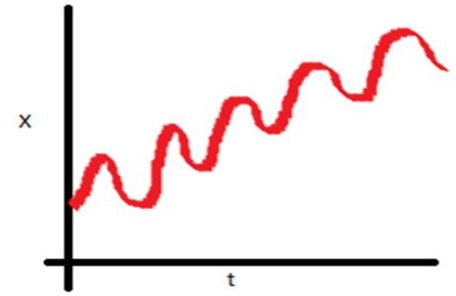


# Time Series: Types

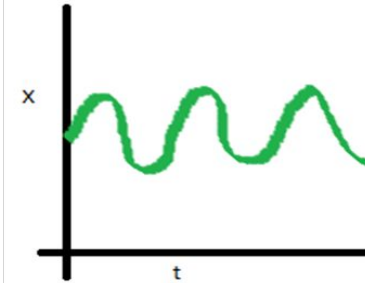
## Stationary



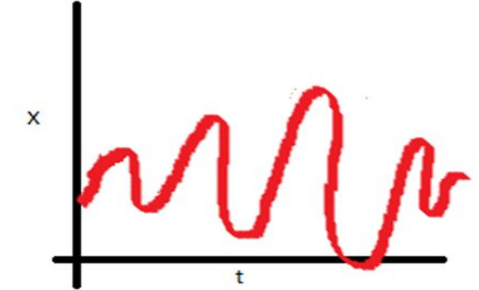
Stationary series



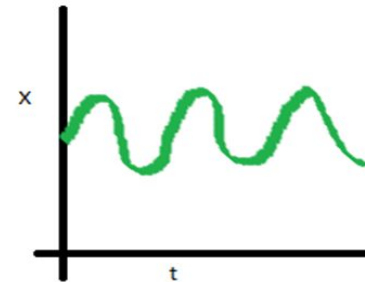
Non-Stationary series



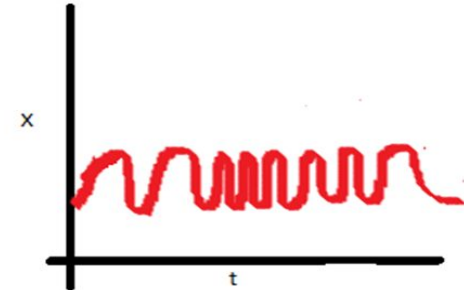
Stationary series



Non-Stationary series



Stationary series

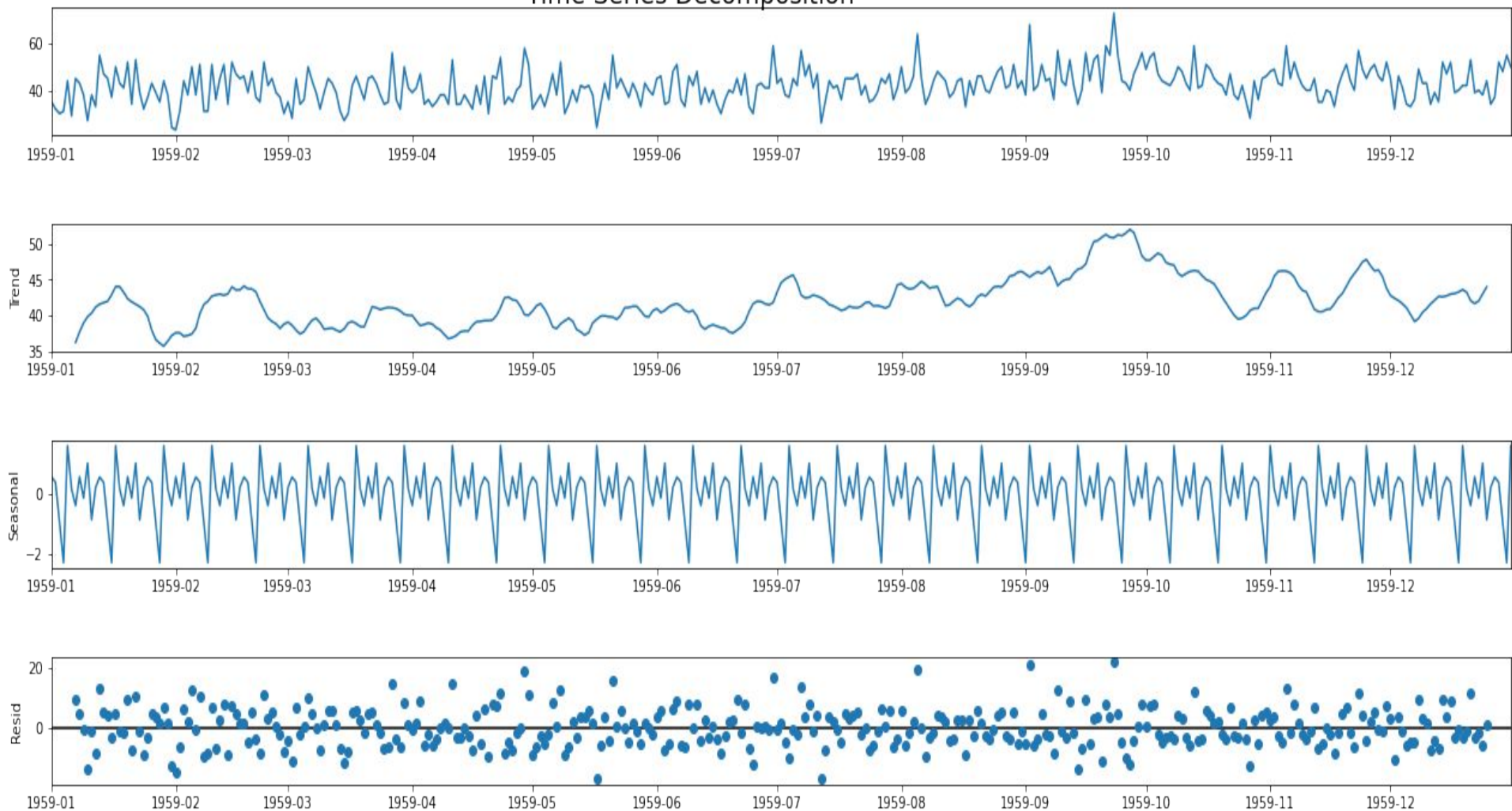


Non-Stationary series

# Time Series: Decomposition

- Long Term Trend
  - ◆ Up Trend
  - ◆ Downtrend
- Seasonality
  - ◆ Based on frequency
  - ◆ Annually, Monthly, Daily, etc...
- Noise (Residuals)

# Time Series Decomposition





# Objective & Scope



# Objective & Scope

## → Main Objective:

- ◆ Building a stable Time Series Predictor works for IoT sensor readings in order to forecast the upcoming readings of that sensor.
- ◆ Deployment on Master of Things (MoT).
- ◆ Running of any script in time frame less than 2 seconds.

## → Scope:

- ◆ Univariate Predictor.
- ◆ Predictor doesn't forecast long sequence.

# Related Work

# Forecasting

# Methods

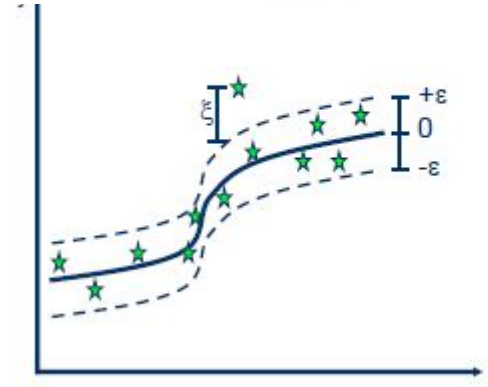
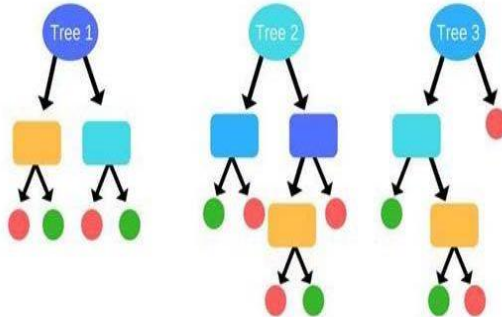
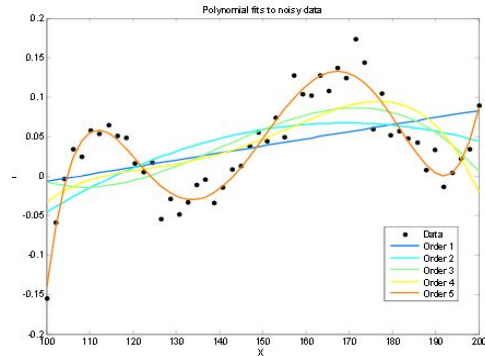


# Forecasting: Traditional Methods

- Naive Method
- Simple Average
- Moving Average
- Weighted Moving Average
- Exponential Smoothing

# Forecasting: Classic ML

- Polynomial Regression
- Random Forests
- Support Vector Machine

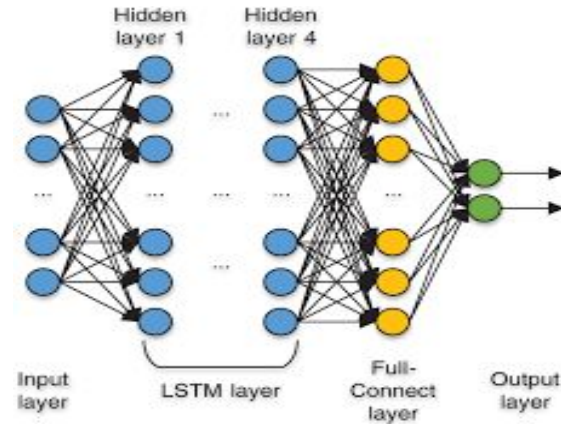
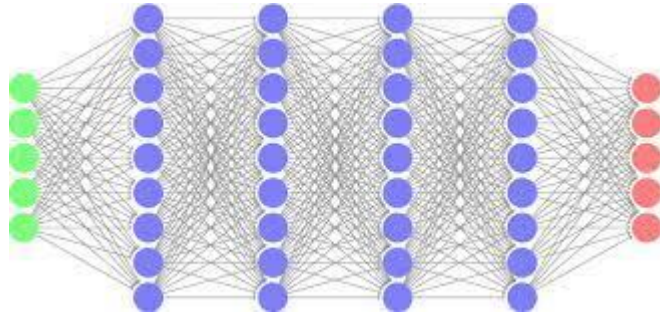




# Forecasting: Deep Learning

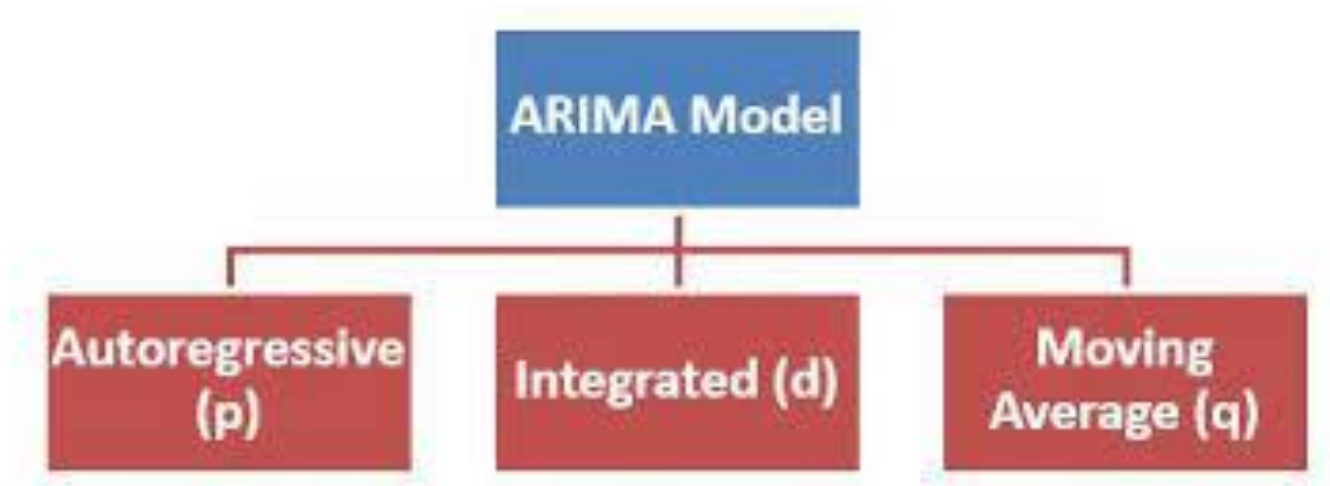
→ Deep Neural Networks

→ Recurrent Neural Networks (RNNs)



# Forecasting: ARIMA

AutoRegressive Moving Average



# Forecasting: ARIMA

→ ARMA

→ ARIMA

→ SARIMA

→ Auto ARIMA

# Our ARIMA Model



# Our ARIMA Model

- Overview
- Mathematically
- In Code
- Training Approach
- Evaluation



# Our ARIMA Model: Overview

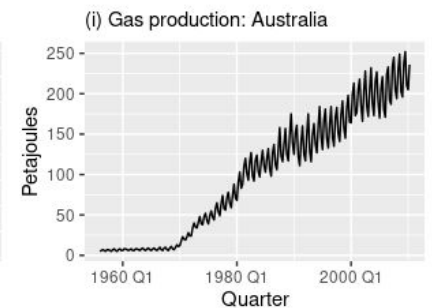
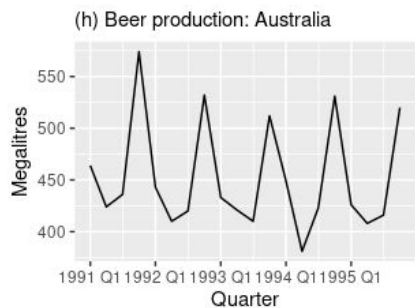
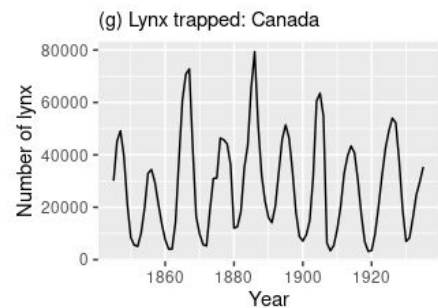
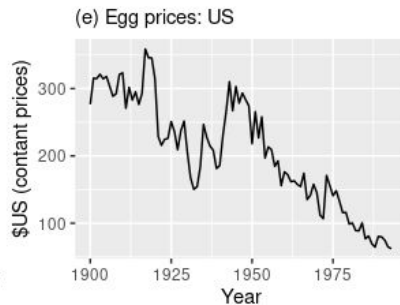
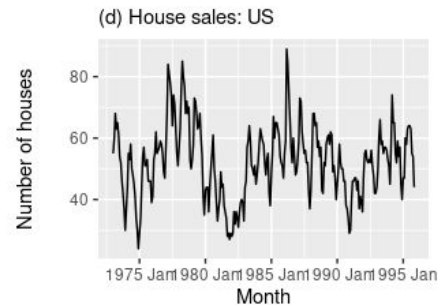
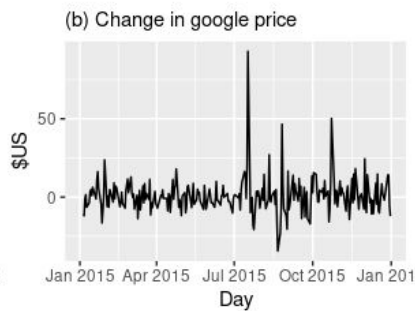
ARIMA(p, d, q)

[Demo Sheet](#)

- **AR (p)**: refers to a model that shows a changing variable that regresses on its own lagged, or prior, values.
- **I (d)**: represents the differencing of raw observations to allow for the time series to become stationary.
- **MA (q)**: incorporates the dependency between an observation and a residual error.

# Our ARIMA Model: Mathematically

- Stationarisation Techniques
- Autoregressive
- Integration
- Moving Average



# Our ARIMA Model: Mathematically

$$y'_t = y_t - y_{t-1}$$

$$y_t - y_{t-1} = \varepsilon_t \rightarrow y_t = y_{t-1} + \varepsilon_t : \text{random walk}$$

$$y_t = c + y_{t-1} + \varepsilon_t : \text{random walk w/ drift}$$

$$y''_t = y'_t - y'_{t-1} = y_t - 2y_{t-1} + y_{t-2}$$

$$By_t = y_{t-1}; B^2y_t = y_{t-2} \rightarrow B^d y_t = y_{t-d}$$

$$y'_t = y_t - y_{t-1} = (1 - B)y_t$$

$$y''_t = y'_t - y'_{t-1} = y_t - 2y_{t-1} + y_{t-2} = (1 - 2B + B^2)y_t = (1 - B)^2 y_t$$

$$(1 - B)^d y_t : d\text{-th order difference}$$

# Our ARIMA Model: Mathematically

$$\hat{y}_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \dots + \phi_p y_{t-p} + \varepsilon_t$$

$$\hat{y}_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \dots + \theta_q \varepsilon_{t-q}$$

# Our ARIMA Model: Mathematically

$$\hat{y}'_t = c + \phi_1 y_{t-1} + \dots + \phi_p y_{t-p} + \theta_1 \varepsilon_{t-1} + \dots + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

$$(1 - B\phi_1 - \dots - B^p\phi_p)(1 - B)^d y_t = c + (1 + B\theta_1 + \dots + B^q\theta_q)\varepsilon_t$$

$$\Phi(B)(y'_t - \mu) = \Theta(B)\varepsilon_t; c = \mu(1 - \phi_1 - \dots - \phi_p); \mu = \frac{1}{T} \sum_{t=0}^T y'_t$$

# Our ARIMA Model: In Code

## → Pure JS Implementation

- ◆ Pros: has no dependency on any other frameworks.
- ◆ Cons: in absence of any low level optimisation, combined with time limit requirement; data records are kept under 128 & results could be significantly inaccurate.



# Our ARIMA Model: In Code

```
826 module.exports = {  
827   array, empty, diff, dot, ndim, reshape, shape, sum, transpose, diag, ones,  
828   zeros, eye, arange, vstack, hstack, NDArray, linalg, linspace, random,  
829   cumsum, mean, std, prod  
830 }
```

```
class AutoRegressionIntegratedMovingAverage extends GradientDescent {  
  
  constructor(order = [1, 0, 0], KWArgs = { learningRate: 1e-3 }) {  
    super(KWArgs.learningRate || 1e-3, KWArgs);  
    [this._p, this._d, this._q] = order;  
    this._update = function (gradient, m, vt1 = 0) {  
      this._W = this._W.add(this.vt(gradient, m, vt1));  
    };  
  }  
}
```

# Our ARIMA Model: In Code

## → TensorFlow JS Implementation

- ◆ **Pros:** Problem could be implemented as a neural network; Optimisations allow for faster processing (up to x20); which in turn encourages usage of more data records, plus allows for training on several loss functions.
- ◆ **Cons:** Still not fully deployed yet, Miss the integration part

# Our ARIMA Model: In Code

```
function buildModel(inputShape, optimizer, loss, metrics) {  
  // Define input, which has a size of inputShape  
  const inputLayer = tf.input({ shape: inputShape });  
  
  // Output dense layer uses linear activation.  
  const denseLayer1 = tf.layers.dense({ units: 1 });  
  
  // Obtain the output symbolic tensor by applying the layers on the inputLayer.  
  const output = denseLayer1.apply(inputLayer);  
  
  // Create the model based on the inputs.  
  const model = tf.model({ inputs: inputLayer, outputs: output });  
  
  model.compile({  
    optimizer: optimizer,  
    loss: loss,  
    metrics: metrics,  
  });  
}
```

# Framework Of Time Series ARIMA Modeling (Box-Jenkins)

1. Visualize the time series

2. Stationarize the series

3. Plot ACF/PACF charts and find optimal parameters

4. Build the ARIMA model

5. Make Predictions

# Our ARIMA Model: Training Approach

We need to tune some hyperparameters to help the model adapt the data in an appropriate way, So how to choose correct value for:

- Epochs
- Learning Rate
- $p$
- $d$
- $q$

# Our ARIMA Model: Training Approach

For tuning  $(p, d, q)$  there are some visual and statistical methods such as:

- Augmented Dickey–Fuller test (ADF)
- Kwiatkowski-Phillips-Schmidt-Shin test (KPSS)
- Autocorrelation Function (ACF)
- Partial Autocorrelation Function (PACF)

# Our ARIMA Model: Evaluation Metrics

- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- Log Likelihood
- Akaike's Information Criteria (AIC)
- Bayesian Information Criteria (BIC)



# Our ARIMA Model: Evaluation Metrics

Metrics in mathematical forms

Log Likelihood

$$LL = -\frac{n}{2} \ln(2\pi\sigma^2) - \frac{1}{2n\sigma^2} \sum_{i=0}^n \varepsilon^2$$

Akaike's Information Criteria

$$AIC = 2(p + q + k + 1) - 2LL; k = 1 \text{ if } c \neq 0, k = 0 \text{ if } c = 0$$

Corrected AIC

$$AIC_c = AIC + \frac{2(p+q+k+2)(p+q+k+1)}{T-p-q-k-2}$$

Bayesian Information Criteria

$$BIC = AIC + (p + q + k + 1)(\ln(T) - 2)$$

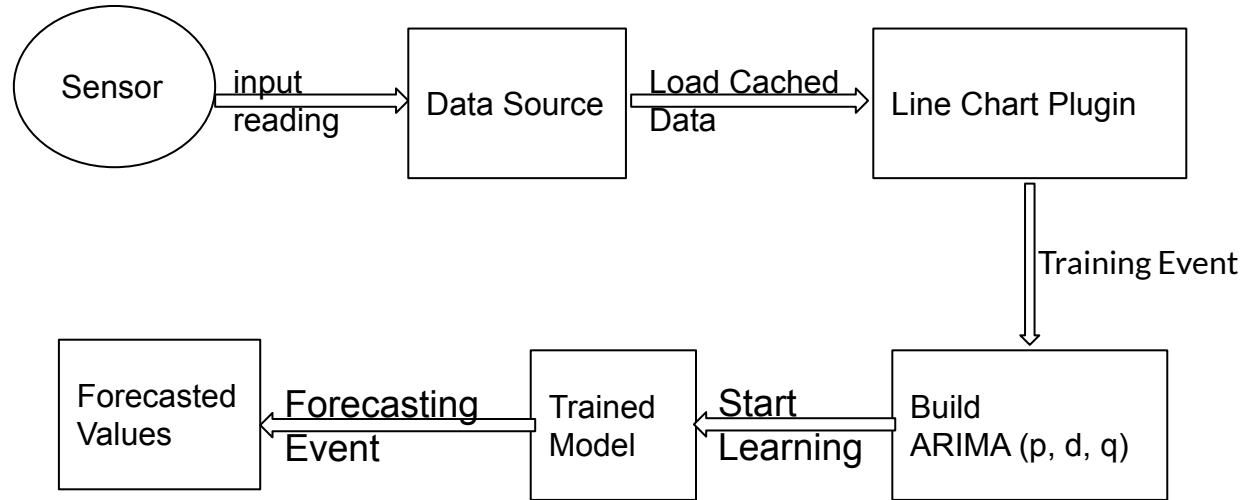
# Deployment On MoT



# Deployment: Master of Things (MoT)

- ➔ It's an Internet of Things (IoT) platform enables you to create IoT & Machine-to-machine (M2M) applications to serve you in few hours.
- ➔ It's a product of **SpimeSenseLabs**.

# Deployment: Architecture



# Deployment: Events

→ On Click Events

◆ Training

◆ Forecasting

◆ Evaluation

→ Custom Events

→ Timer Tick Event

# Deployment: Events

## → Onclick Training Event:

- ◆ Read sensor data.
- ◆ Build the model with required hyperparameters from UI.
- ◆ Start fitting the data.
- ◆ Save the trained model after finishing fitting into JSON object containing all the learned params.

# Deployment: Events

- **OnClick Forecasting Event:**
  - ◆ Read the number of needed predictionprediction steps.
  - ◆ Load the saved trained model.
  - ◆ Call forecast method.
  - ◆ Show the predictions in UI list.



# Deployment: Events

## → Onclick Evaluate Event:

- ◆ Train on 90% of sensor data.
- ◆ Forecast number of steps equals to 10% of sensor data
- ◆ Call an evaluation matrix, such as: MSE.

# Deployment: Performance

- Training Action:
  - ◆ Less than 2,000 ms
- Forecasting Action:
  - ◆ Less than 2,000 ms

# Deployment: Retraining Schedule

**Based on readings frequency:**

The higher the frequency of coming readings, the more retraining must be done, And vice versa...

# Live Demo



# Live Demo

The screenshot displays a live demo interface. On the left, a code editor shows JavaScript code for a prediction plugin. Lines 1037 through 1043 are highlighted in blue. The code defines an `AutoRegressionIntegratedMovingAverage` model and sets various parameters like `mod._initialValue`, `mod._p`, `mod._d`, `mod._q`, `mod._w`, `mod._residuals`, and `mod._lags`. It also sets a `peridos_value` parameter and uses `forecastSync` to generate predictions. On the right, a 'Plugin Properties (Predict)' panel is visible, showing a table with 'Name' and 'Value' columns. The 'Help Tip' field is empty, 'Version' is 2, 'Show border' is unchecked, and 'Border Rounded' is unchecked. Below this is a 'Parameter Description' section. At the bottom, a status bar shows 'Mouse X,Y Position: 20,308' and a copyright notice '© 2020 MasterOfThrones. All rights reserved.'

```
1033
1034 // TODO predict
1035 DataListGetAsync().then(model => {
1036   mod = new AutoRegressionIntegratedMovingAverage();
1037   mod._initialValue = model._initialValue
1038   mod._p = model._p
1039   mod._d = model._d
1040   mod._q = model._q
1041   mod._w = model._w
1042   mod._residuals = model._residuals
1043   mod._lags = model._lags
1044
1045   var peridos = parseInt(GetPluginParameterValue('peridos_value', 'Selected item'));
1046   return mod.forecastSync(peridos);
1047 }).then(predictions => {
1048   // TODO use predictions
1049   predictions.forEach(function(item, index) {
1050     predictions[index] = parseFloat(item);
1051   });
1052   SetPluginParameterValue('Pred_list', 'List captions', predictions);
1053   SetPluginParameterValue('Pred_list', 'Visible', 1);
1054   DrawPlugin('Pred_list');
1055   console.log(predictions)
1056   event.end();
1057 }).catch(event.error);
```

Name	Value
Help Tip	
Version	2
Show border	<input type="checkbox"/>
Border Rounded	<input type="checkbox"/>

Parameter Description

Mouse X,Y Position: 20,308

© 2020 MasterOfThrones. All rights reserved.

---

# Thank You

---