# Soft Computing Techniques for Predictive Maintenance

Machine Health Monitoring and Evaluation using IOT
NITTTR, Chandigarh

Dr. Pooja
Associate Professor, CSE
SET, Sharda University, UP
pooja.1@sharda.ac.in
www.poojaangurala.com

## Soft Computing

- One may see soft computing as an attempt to mimic natural creatures: plants, animals, human beings, which are soft, flexible, adaptive and clever.

- In this sense soft computing is the name of a family of problem-solving methods that have analogy with biological reasoning and problem solving (sometimes referred to as cognitive computing).

- The role model for soft computing is the human mind.

- Soft computing is based on techniques such as **artificial neural networks**, **fuzzy logic**, **genetic algorithms**, **machine learning**, and **expert systems**.

- Soft computing deals with approximate models and gives solutions to complex real-life problems.
- Soft computing is tolerant of imprecision, uncertainty, partial truth, and approximations.
- Although soft computing theory and techniques were first introduced in 1980s, it has now become a major research and study area in automatic control engineering.
- The techniques of soft computing are nowadays being used successfully in many domestic, commercial, and industrial applications.

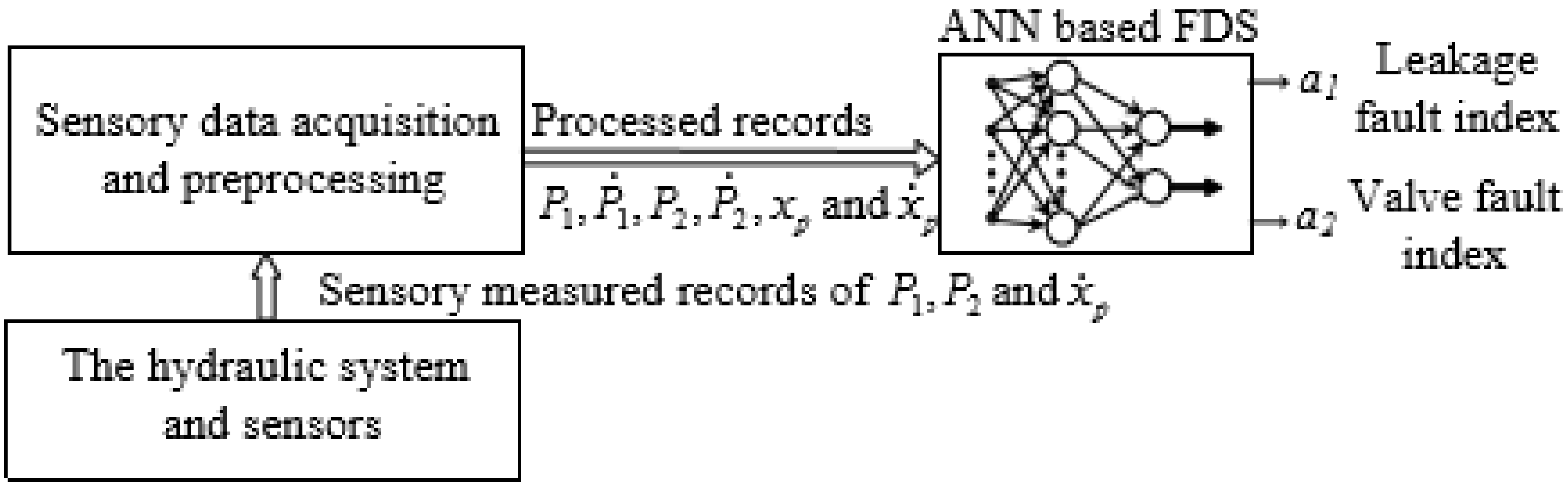
Dogan Ibrahim, “An Overview of Soft Computing”, Procedia Computer Science, Volume 102, pp. 34-38, 2016.

# Neural Network

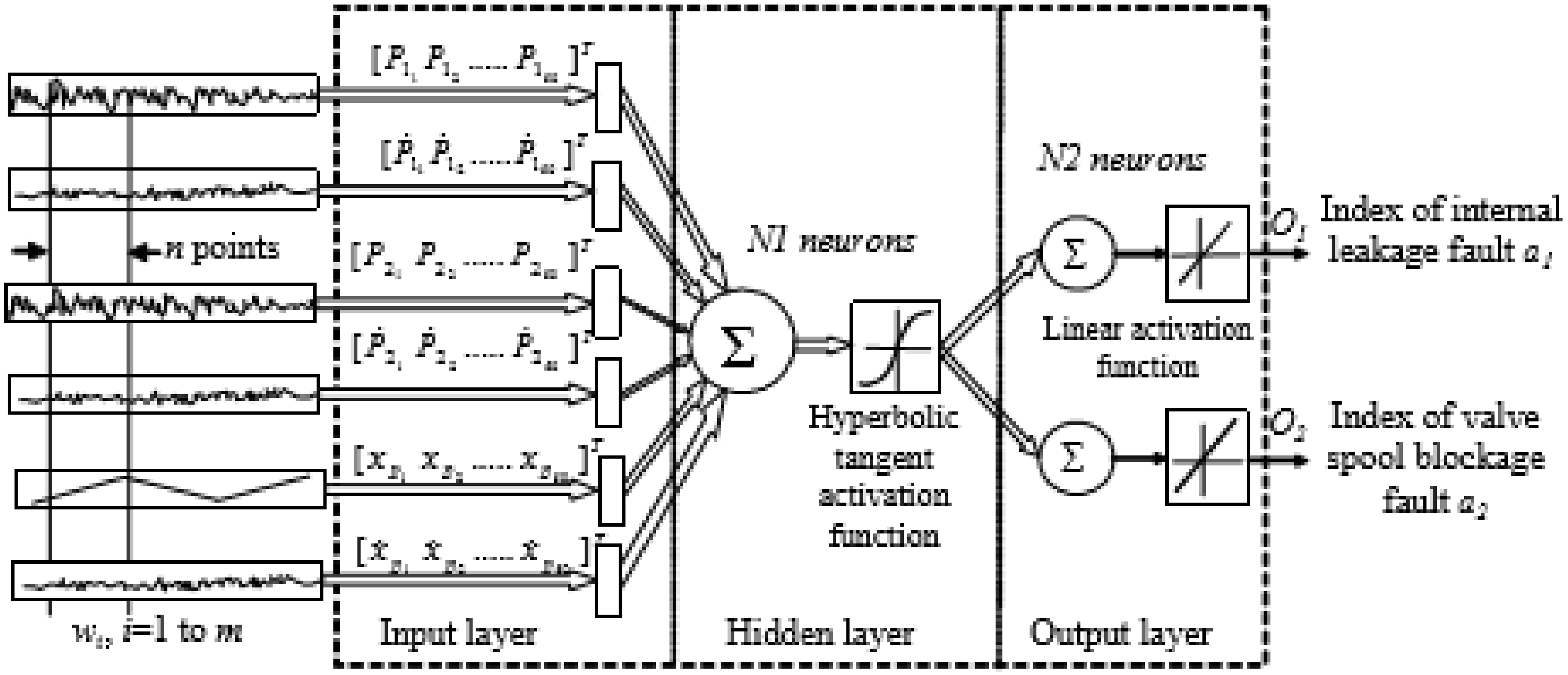
- Artificial neural network (ANN) is an information-processing approach. It works like the biological nervous systems like how the brain processes the information in the human body.
- The network architecture or topology (including number of nodes in hidden layers, network connections, initial weight assignments and activation functions) played a key role in the ANN performance and depended on the problem at hand.

## Neural Networks

- The artificial neural networks (ANNs) have the capability to perform pattern recognition and diagnosis that are difficult to describe in terms of analytical diagnosis algorithms since they can learn input patterns by themselves .
- classifies the input data vectors to different categories according to may be fault types


Sensory data acquisition and preprocessing
Processed records
$P_1, \dot{P}_1, P_2, \dot{P}_2, x_p$ and $\dot{x}_p$
ANN based FDS
$a_1$
Leakage fault index
$a_2$
Valve fault index
Sensory measured records of $P_1, P_2$ and $\dot{x}_p$
The hydraulic system and sensors

Ahemd El-Betar, Magdy M. Abdelhamed*, Ahmed El-Assal and Roubi Abdelsatar, “Fault Diagnosis of a Hydraulic Power System Using an Artificial Neural Network ”

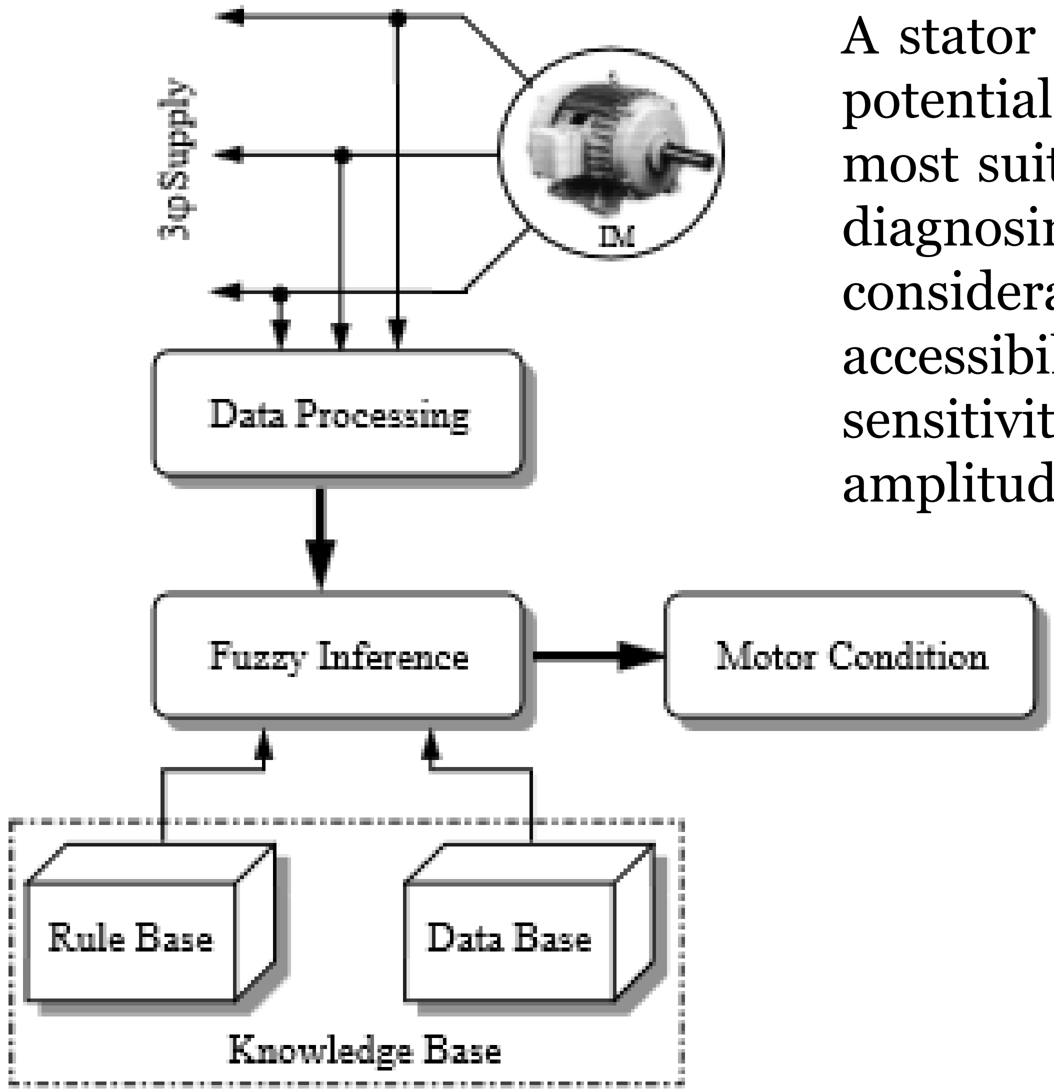


$[P_{1_1}\, P_{1_2} \ldots P_{1_m}]^T$
$[\dot{P}_{1_1}\, \dot{P}_{1_2} \ldots \dot{P}_{1_m}]^T$
$n$ points
$[P_{2_1}\, P_{2_2} \ldots P_{2_m}]^T$
$[\dot{P}_{2_1}\, \dot{P}_{2_2} \ldots \dot{P}_{2_m}]^T$
$[x_{s_1}\, x_{s_2} \ldots x_{s_m}]^T$
$[\dot{x}_{s_1}\, \dot{x}_{s_2} \ldots \dot{x}_{s_m}]^T$
$w_{i}$, $i$=1 to $m$
Input layer
N1 neurons
Hyperbolic tangent activation function
Hidden layer
N2 neurons
Linear activation function
Output layer
$O_1$
Index of internal leakage fault $a_1$
$O_2$
Index of valve spool blockage fault $a_2$

[Ahemd El-Betar, Magdy M. Abdelhamed*, Ahmed El-Assal and Roubi Abdelsatar, "Fault Diagnosis of a Hydraulic Power System Using an Artificial Neural Network "]

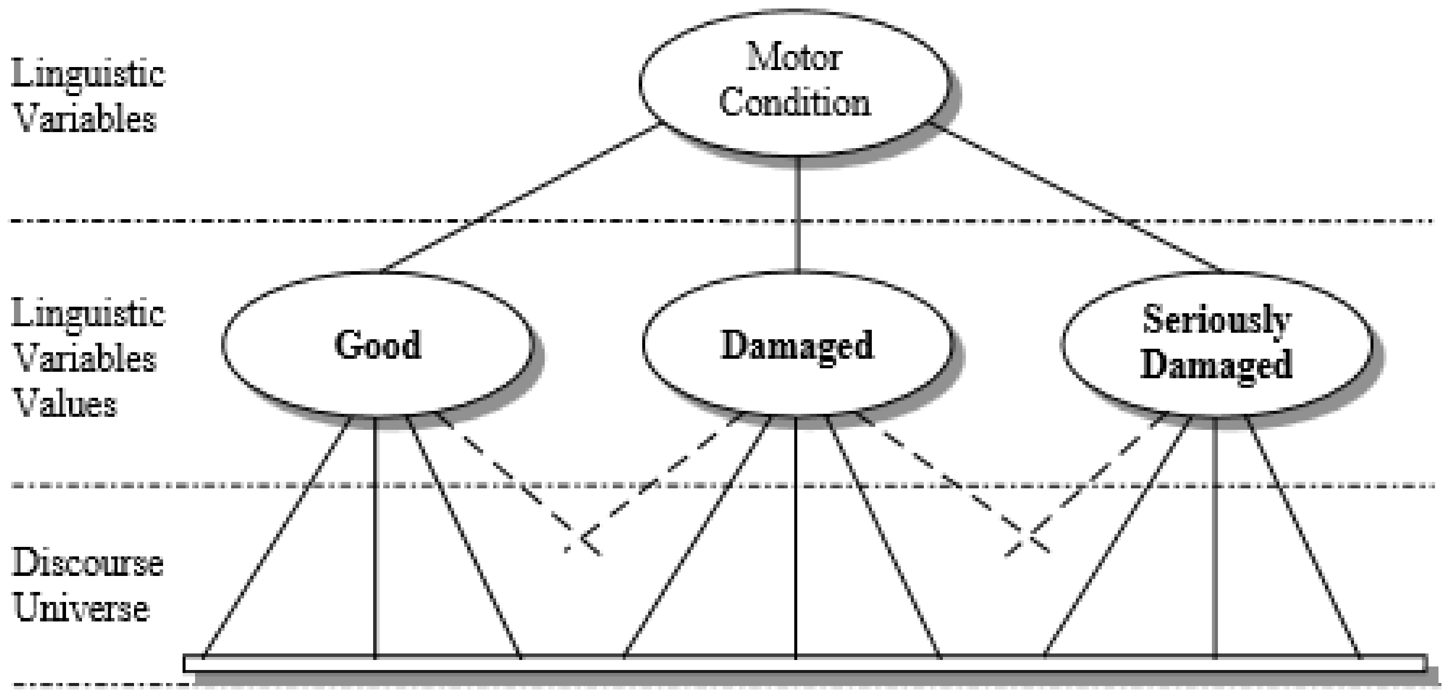
# Fuzzy Logic

- FL is a multi-valued logic that allows the intermediate values between conventional evaluations like true/false, yes/no, high/low, and so on. The FL helps in providing a variety of ways to solve a control or classification problem.



A stator current signal contains potential fault information. The most suitable measurements for diagnosing the faults under consideration, in term of easy accessibility, reliability, and sensitivity, are the stator current amplitudes Ia, Ib, and Ic.

A Simple Fuzzy Logic Approach for Induction Motors Stator Condition Monitoring

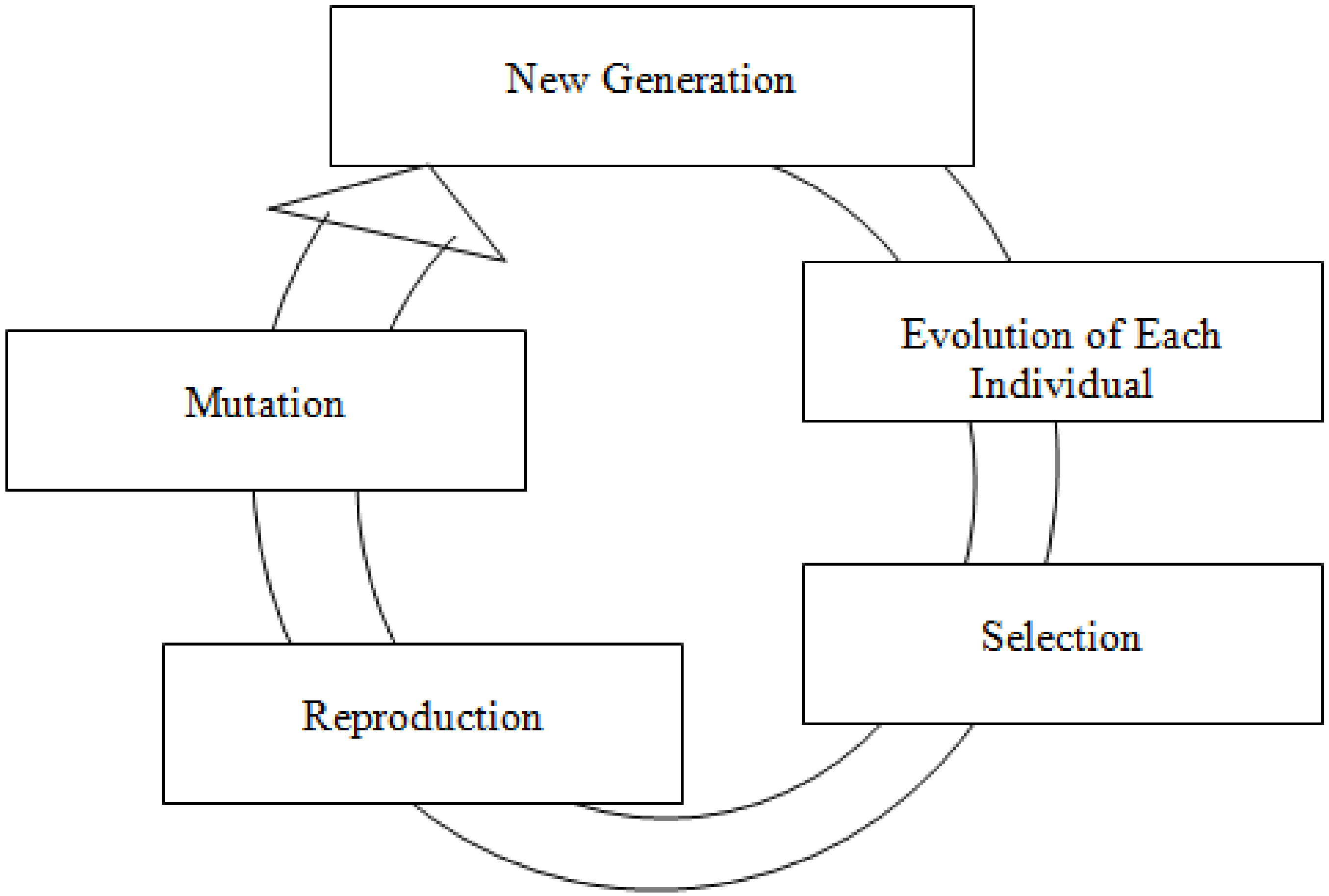


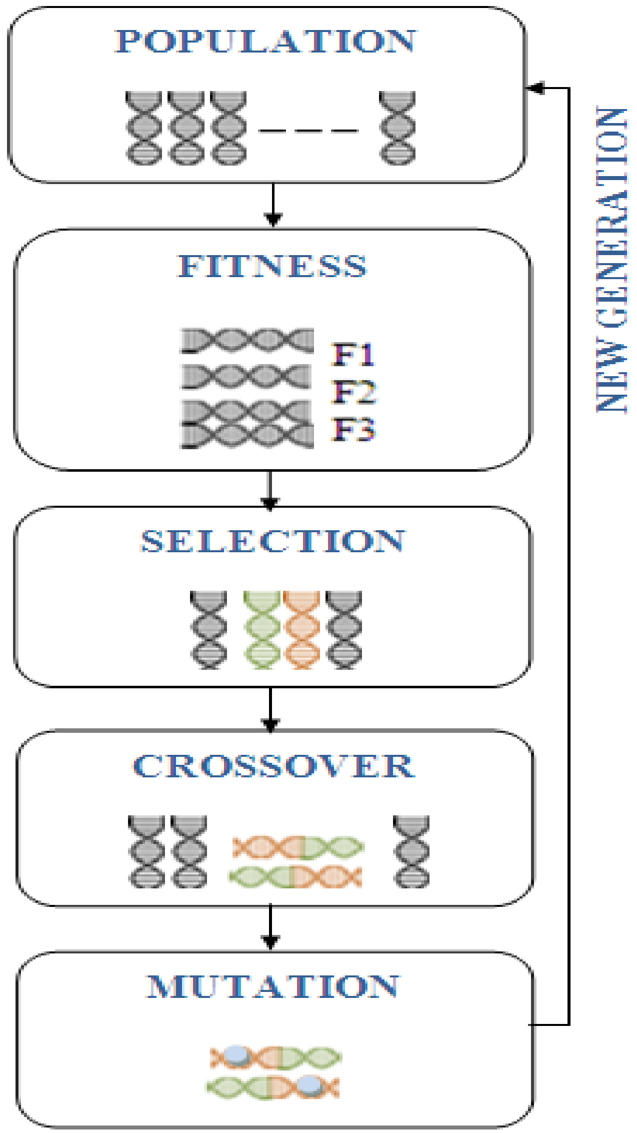
A Simple Fuzzy Logic Approach for Induction Motors Stator Condition Monitoring

## Genetic Algorithms

- A genetic algorithm is a search heuristic that is inspired by Charles Darwin’s theory of “survival of the fittest”.
- A genetic algorithm maintains a population of candidate solutions for the problem in hand, and make it evolve by iteratively applying a set of stochastic operators.

## How genetic Algorithms work?

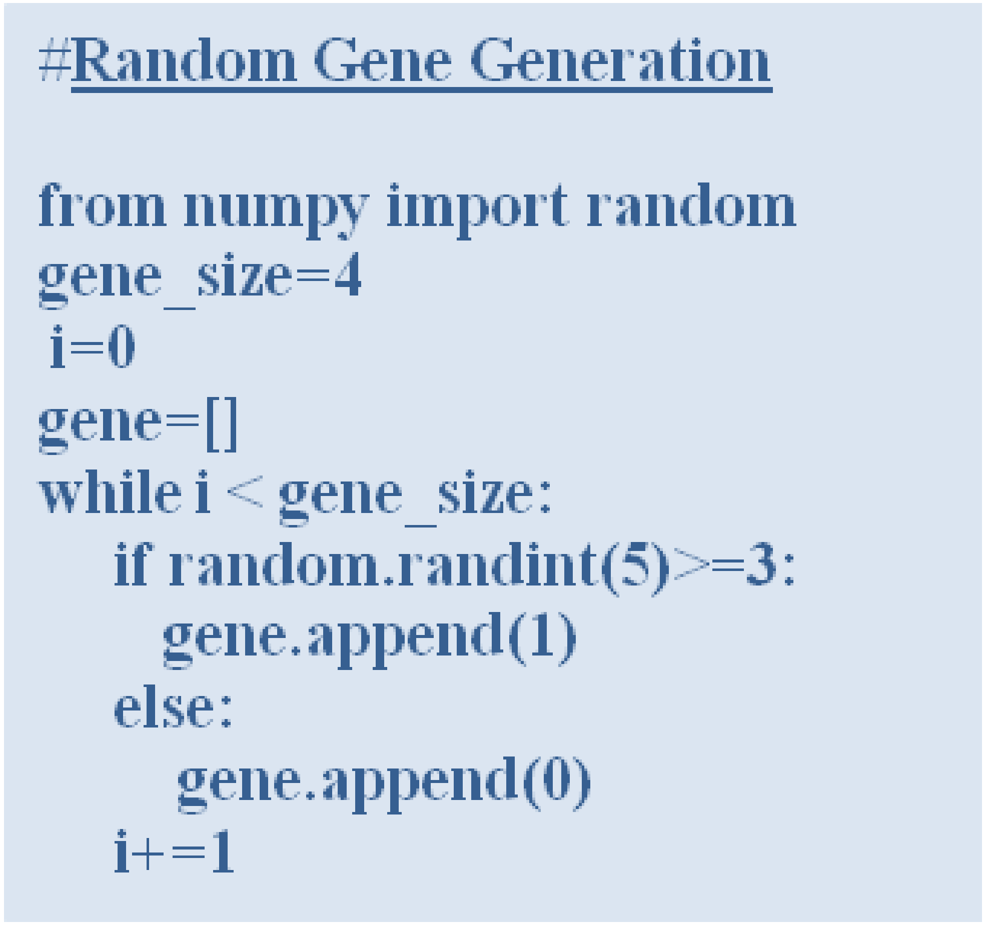



POPULATION
FITNESS
F1
F2
F3
SELECTION
CROSSOVER
MUTATION
NEW GENERATION

# Genetic Algorithms

- A Genetic Algorithm based optimization procedure is used to search for the most cost-effective maintenance schedule, considering both production gains and maintenance expenses. The algorithm is implemented in a simulated environment and benchmarked against several traditional maintenance strategies, such as corrective maintenance, scheduled maintenance and condition-based maintenance.

Zimin (Max) Yang, Dragan Djurdjanovic , Jun Ni, "Maintenance scheduling in manufacturing systems based on predicted machine degradation"

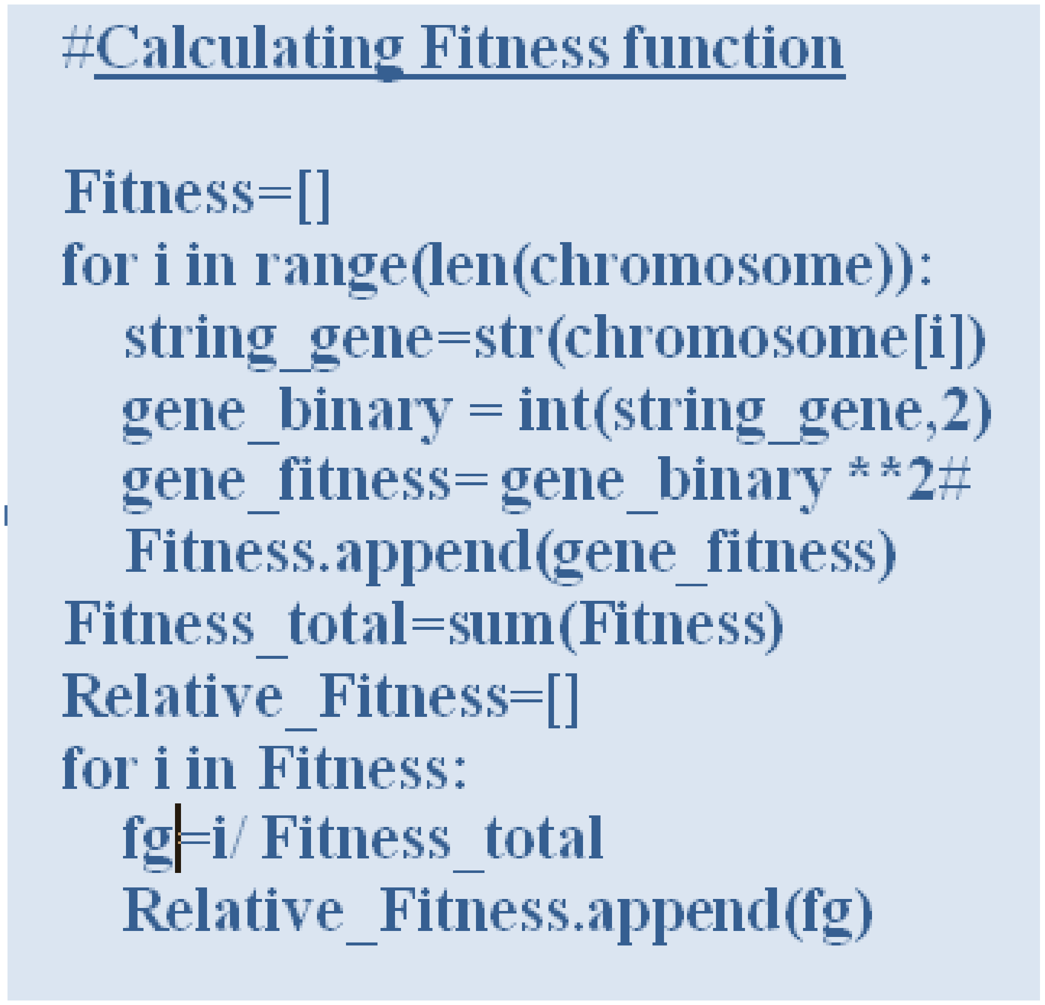


```
#Random Gene Generation

from numpy import random
gene_size=4
 i=0
gene=[]
while i < gene_size:
    if random.randint(5)>=3:
       gene.append(1)
    else:
        gene.append(0)
    i+=1
```

```
chromosome = ['01011', '11101', '00101', '00100']
```

- The fitness function F(x) for each of these gene can be calculated as x**2



```
#Calculating Fitness function

Fitness=[]
for i in range(len(chromosome)):
    string_gene=str(chromosome[i])
    gene_binary = int(string_gene,2)
    gene_fitness= gene_binary **2#
    Fitness.append(gene_fitness)
Fitness_total=sum(Fitness)
Relative_Fitness=[]
for i in Fitness:
    fg=i/ Fitness_total
    Relative_Fitness.append(fg)
```

```
#Selection for Crossover

CO_G=[]#crossover gene
j=0
while j<=2:# for two gene
   for i in range(len(Fitness)-1):
      g=max(Fitness)#finding gene with highest fitness
      if Fitness[i]==g:
         CO_G.append(chromosome[i])
      Fitness[i]=0
   j+=1

#Crossover Implementation

newgen1=crossgene0[:3]+crossgene1[3:]#bit flipping crossover
newgen2=crossgene1[:3]+crossgene0[3:]
newPop=[]#next generation population
newPop.append(newgen1)#newgeneraton from crossed chromosomes
newPop.append(newgen2)
```

```
#Mutation

j=0
while j<1:
    for i in range(len(newPop)):
        bm=random.randint(5)#random point generation for gene mutation
        newpop1[i]=list(newpop1[i])
        if newpop1[i][bm]==0:
            #newpop1[i]=list(newpop1[i])
            newpop1[i][bm]=1
            #newpop1[i]=str(newpop1)
        else:
            newpop1[i][bm]=0
        s=''
        for g in range(len(newpop1[0])):
            ps=str(newpop1[0][g])
            s=s+ps
        newpop1[i]=s #mutated chromosomes of new generation
    j+=1
```

# Predictive Maintenance (PdM)

- Preventive maintenance : aims to prepone the time of preventative measures before the time of a potential asset failure. One of its draw back:-the current condition of an asset does not influence its maintenance schedule.
- Condition Based Monitoring (one implementation of PdM) includes measurements of the condition of the assets into its maintenance planning.

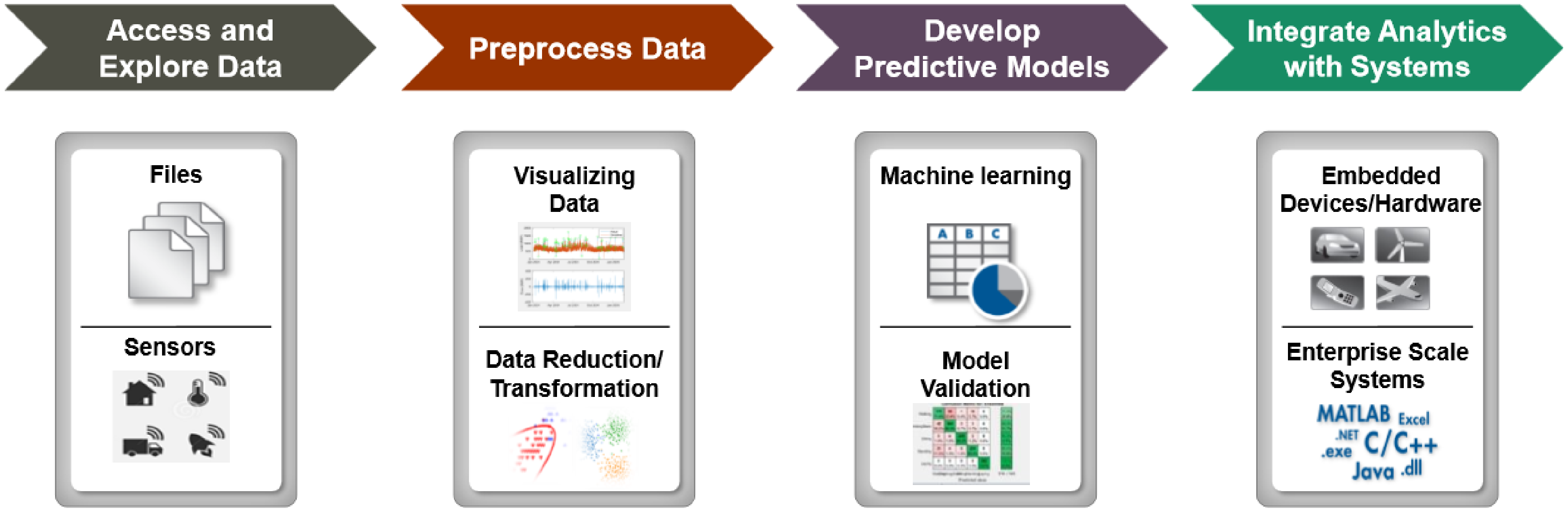
- Forecast failures of assets by analyzing data related to assets and tries to find an optimal point in time for maintenance.
- Predictive maintenance detect the anomalies and failure patterns and provide early warnings.
- These warnings can enable efficient maintenance of those components.
- Frequently recorded characteristics are for example the temperature or vibration in a machine.

## Predictive Maintenance

The success of predictive maintenance models depend on three main components: having the right data available, framing the problem appropriately and evaluating the predictions properly.

Predictive maintenance savings come in two forms:

- Avoid or minimize the downtimes. This will avoid an unhappy customer, save money, and sometimes save lives.
- Optimize the periodic maintenance operations.



Access and Explore Data
Preprocess Data
Develop Predictive Models
Integrate Analytics with Systems
Files
Sensors
Visualizing Data
Data Reduction/ Transformation
Machine learning
Model Validation
Embedded Devices/Hardware
Enterprise Scale Systems
MATLAB Excel .NET .exe C/C++ Java .dll

"Big Data and Machine Learning for Predictive Maintenance" MatLab expo 2017

# Data for predictive maintenance

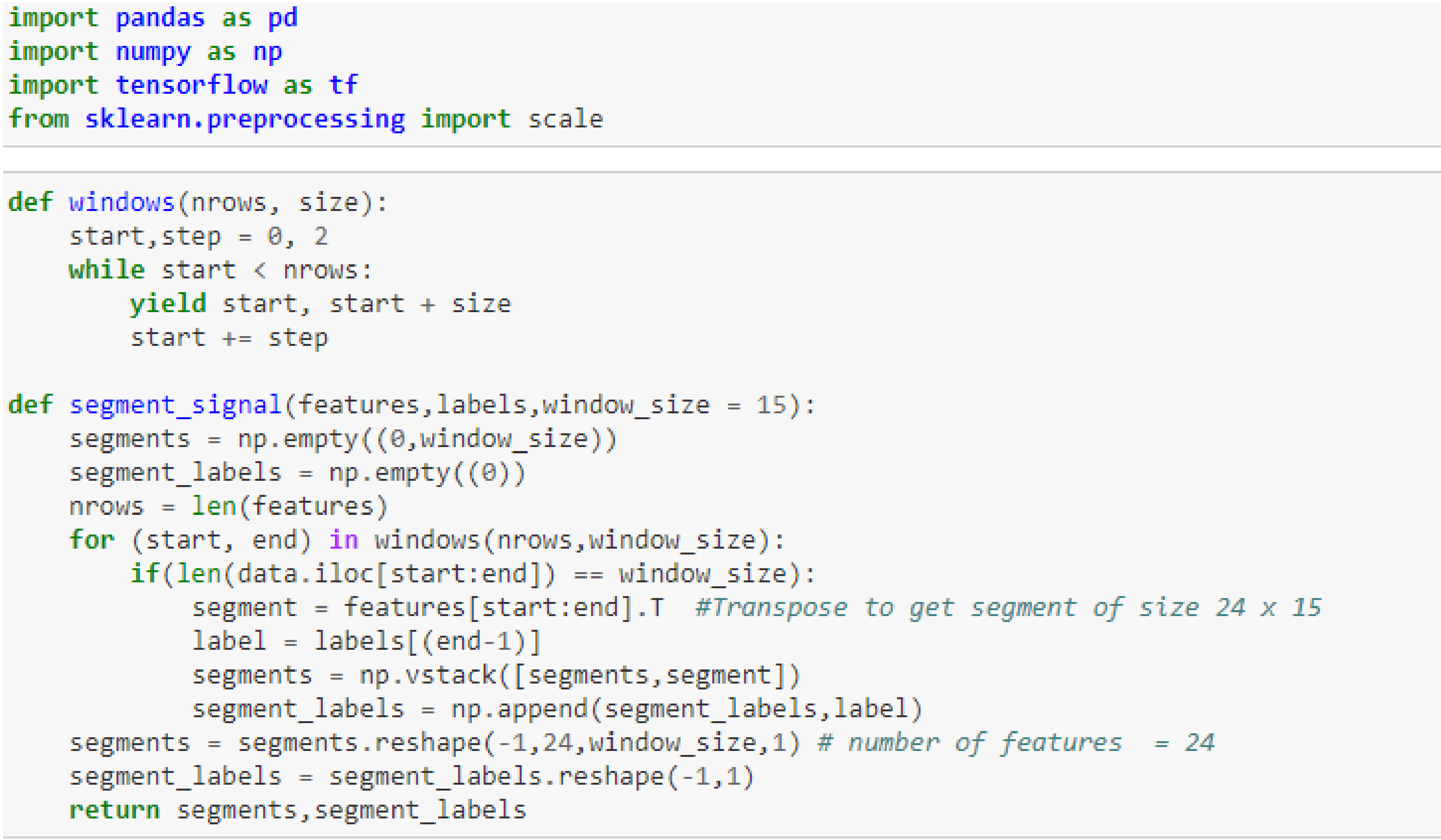
- Time series data.
- Includes timestamp, a set of sensor readings collected at the same time as timestamps, and device identifiers.

## PdM Approaches

- <u>Classification approach</u> - predicts whether there is a possibility of failure in next n-steps.
- <u>Regression approach</u> - predicts how much time is left before the next failure. Also called as <u>Remaining Useful Life</u> (RUL).

## Estimation-of-Remaining-Useful-Life-using-CNN

- https://github.com/aqibsaeed/Estimation-of-Remaining-Useful-Life-using-CNN/blob/master/Estimation%20of%20RUL%20using%20CNN.ipynb

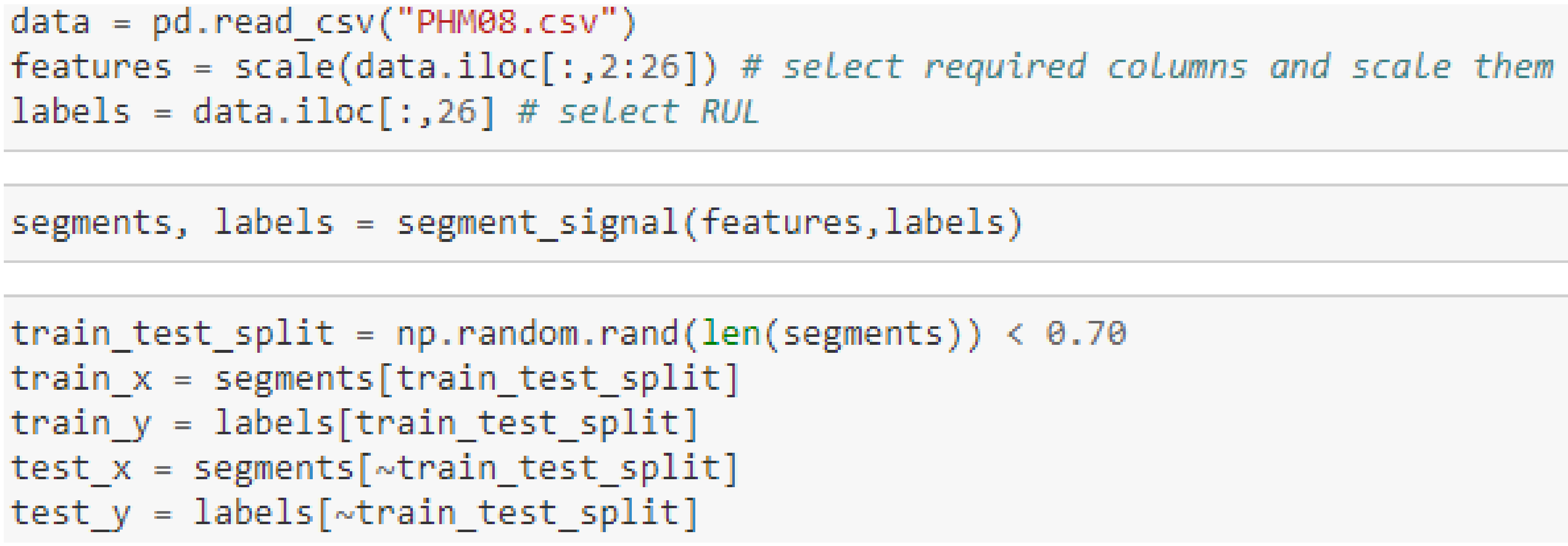


```
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.preprocessing import scale
```

```
def windows(nrows, size):
    start,step = 0, 2
    while start < nrows:
        yield start, start + size
        start += step

def segment_signal(features,labels,window_size = 15):
    segments = np.empty((0,window_size))
    segment_labels = np.empty((0))
    nrows = len(features)
    for (start, end) in windows(nrows,window_size):
        if(len(data.iloc[start:end]) == window_size):
            segment = features[start:end].T  #Transpose to get segment of size 24 x 15
            label = labels[(end-1)]
            segments = np.vstack([segments,segment])
            segment_labels = np.append(segment_labels,label)
    segments = segments.reshape(-1,24,window_size,1) # number of features  = 24
    segment_labels = segment_labels.reshape(-1,1)
    return segments,segment_labels
```

https://github.com/aqibsaeed/Estimation-of-Remaining-Useful-Life-using-CNN/blob/master/Estimation%20of%20RUL%20using%20CNN.ipynb

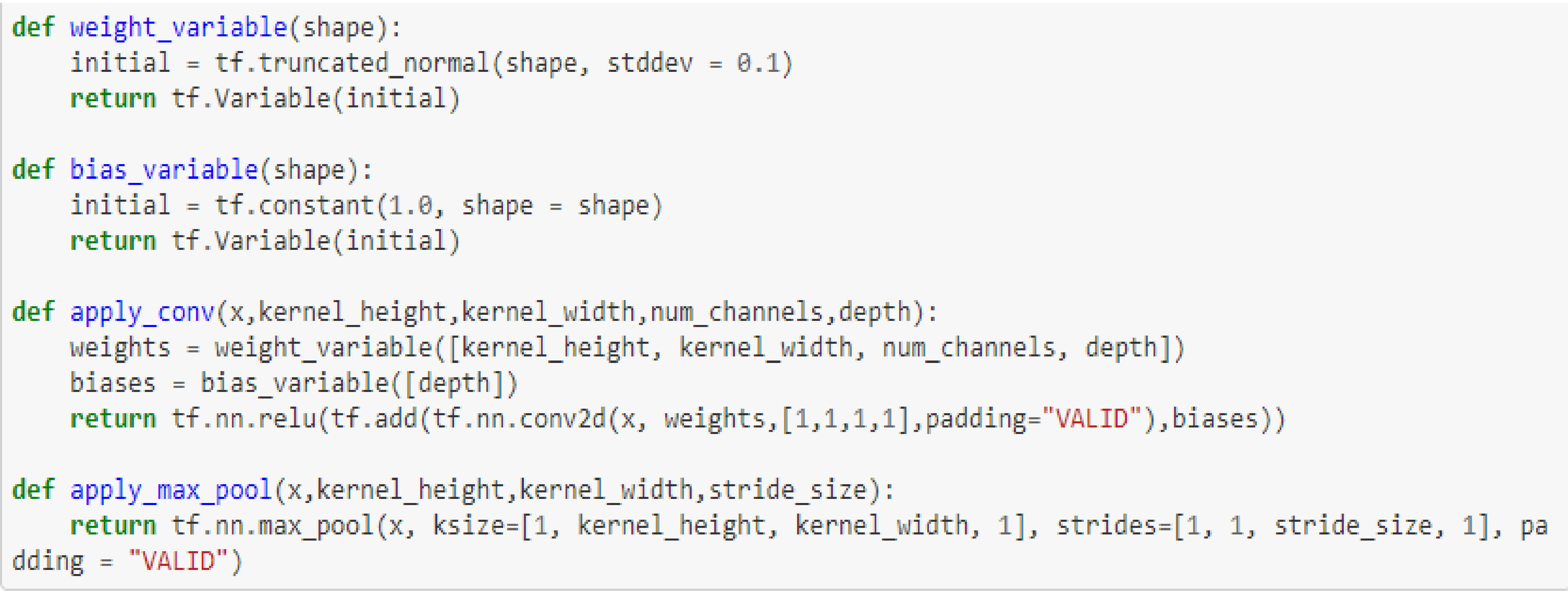


```
data = pd.read_csv("PHM08.csv")
features = scale(data.iloc[:,2:26]) # select required columns and scale them
labels = data.iloc[:,26] # select RUL
```

```
segments, labels = segment_signal(features,labels)
```

```
train_test_split = np.random.rand(len(segments)) < 0.70
train_x = segments[train_test_split]
train_y = labels[train_test_split]
test_x = segments[~train_test_split]
test_y = labels[~train_test_split]
```

A. Saxena and K. Goebel (2008). "PHM08 Challenge Data Set", NASA Ames Prognostics Data Repository (http://ti.arc.nasa.gov/project/prognostic-data-repository), NASA Ames Research Center, Moffett Field, CA



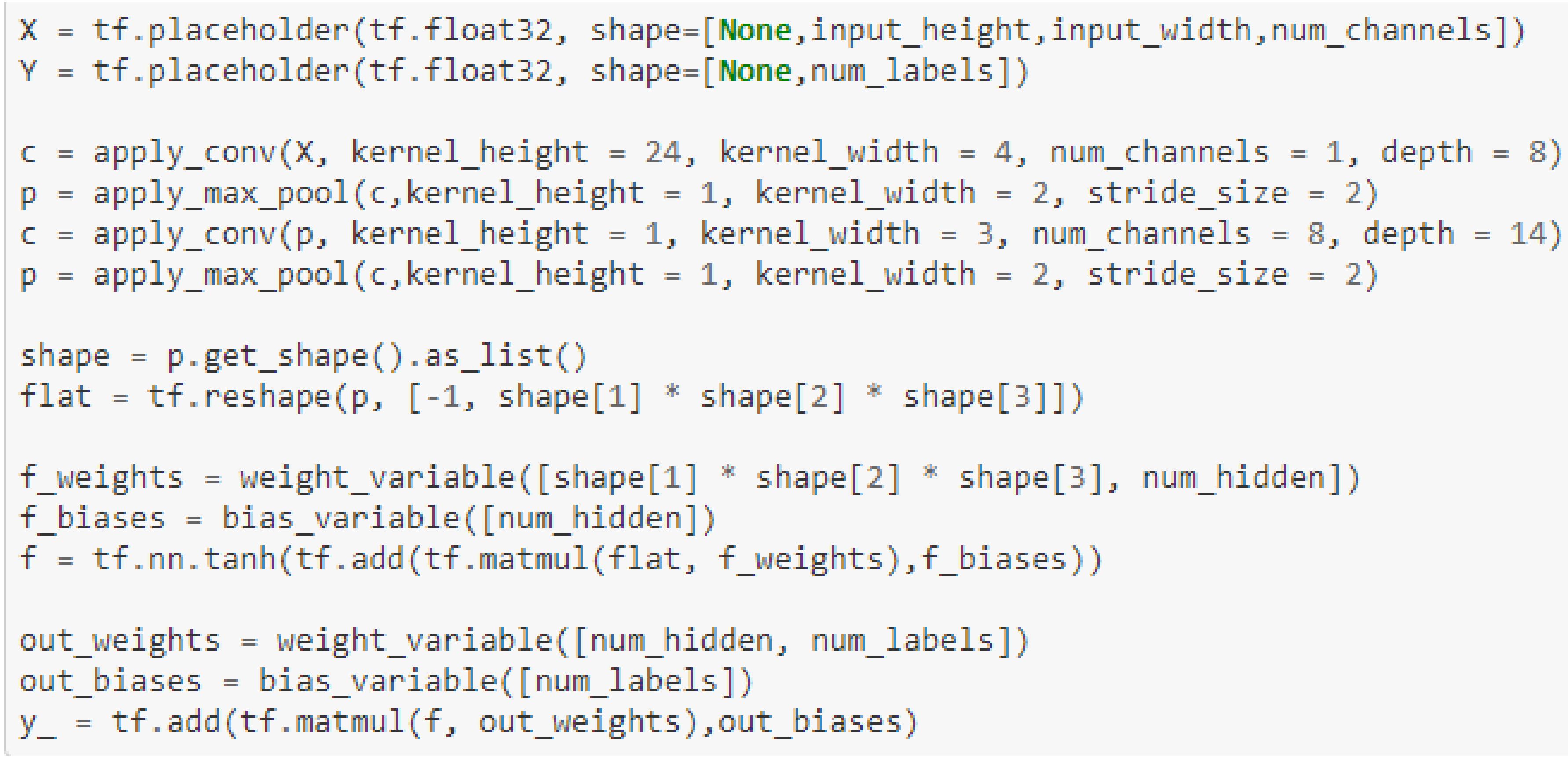
```python
def weight_variable(shape):
    initial = tf.truncated_normal(shape, stddev = 0.1)
    return tf.Variable(initial)

def bias_variable(shape):
    initial = tf.constant(1.0, shape = shape)
    return tf.Variable(initial)

def apply_conv(x,kernel_height,kernel_width,num_channels,depth):
    weights = weight_variable([kernel_height, kernel_width, num_channels, depth])
    biases = bias_variable([depth])
    return tf.nn.relu(tf.add(tf.nn.conv2d(x, weights,[1,1,1,1],padding="VALID"),biases))

def apply_max_pool(x,kernel_height,kernel_width,stride_size):
    return tf.nn.max_pool(x, ksize=[1, kernel_height, kernel_width, 1], strides=[1, 1, stride_size, 1], pa
dding = "VALID")
```

```
num_labels = 1
batch_size = 10
num_hidden = 800
learning_rate = 0.0001
training_epochs = 30
input_height = 24
input_width = 15
num_channels = 1
total_batches = train_x.shape[0] // batch_size
```



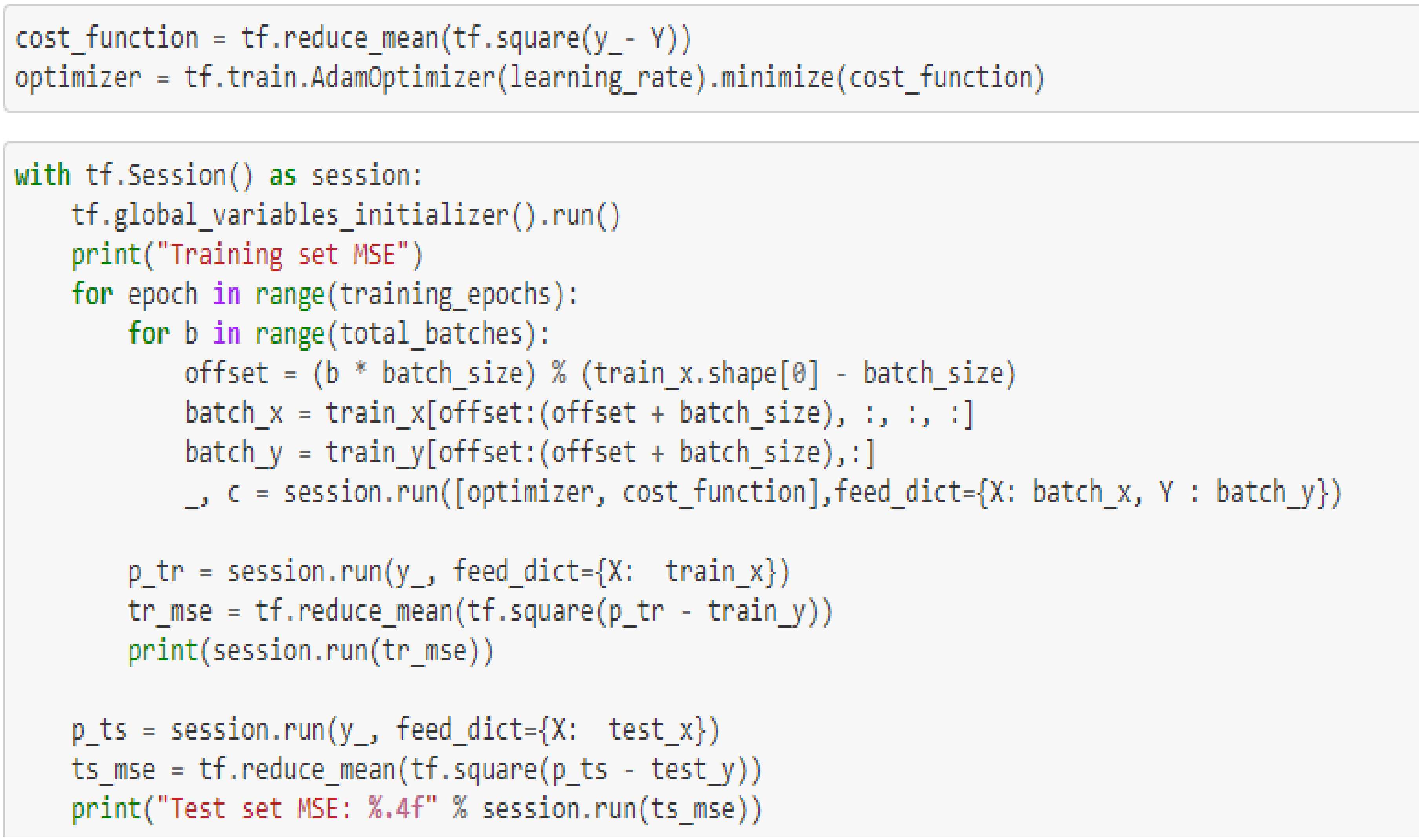
```
X = tf.placeholder(tf.float32, shape=[None,input_height,input_width,num_channels])
Y = tf.placeholder(tf.float32, shape=[None,num_labels])

c = apply_conv(X, kernel_height = 24, kernel_width = 4, num_channels = 1, depth = 8)
p = apply_max_pool(c,kernel_height = 1, kernel_width = 2, stride_size = 2)
c = apply_conv(p, kernel_height = 1, kernel_width = 3, num_channels = 8, depth = 14)
p = apply_max_pool(c,kernel_height = 1, kernel_width = 2, stride_size = 2)

shape = p.get_shape().as_list()
flat = tf.reshape(p, [-1, shape[1] * shape[2] * shape[3]])

f_weights = weight_variable([shape[1] * shape[2] * shape[3], num_hidden])
f_biases = bias_variable([num_hidden])
f = tf.nn.tanh(tf.add(tf.matmul(flat, f_weights),f_biases))

out_weights = weight_variable([num_hidden, num_labels])
out_biases = bias_variable([num_labels])
y_ = tf.add(tf.matmul(f, out_weights),out_biases)
```



```
cost_function = tf.reduce_mean(tf.square(y_- Y))
optimizer = tf.train.AdamOptimizer(learning_rate).minimize(cost_function)
```

```
with tf.Session() as session:
    tf.global_variables_initializer().run()
    print("Training set MSE")
    for epoch in range(training_epochs):
        for b in range(total_batches):
            offset = (b * batch_size) % (train_x.shape[0] - batch_size)
            batch_x = train_x[offset:(offset + batch_size), :, :, :]
            batch_y = train_y[offset:(offset + batch_size),:]
            _, c = session.run([optimizer, cost_function],feed_dict={X: batch_x, Y : batch_y})

        p_tr = session.run(y_, feed_dict={X:  train_x})
        tr_mse = tf.reduce_mean(tf.square(p_tr - train_y))
        print(session.run(tr_mse))

    p_ts = session.run(y_, feed_dict={X:  test_x})
    ts_mse = tf.reduce_mean(tf.square(p_ts - test_y))
    print("Test set MSE: %.4f" % session.run(ts_mse))
```