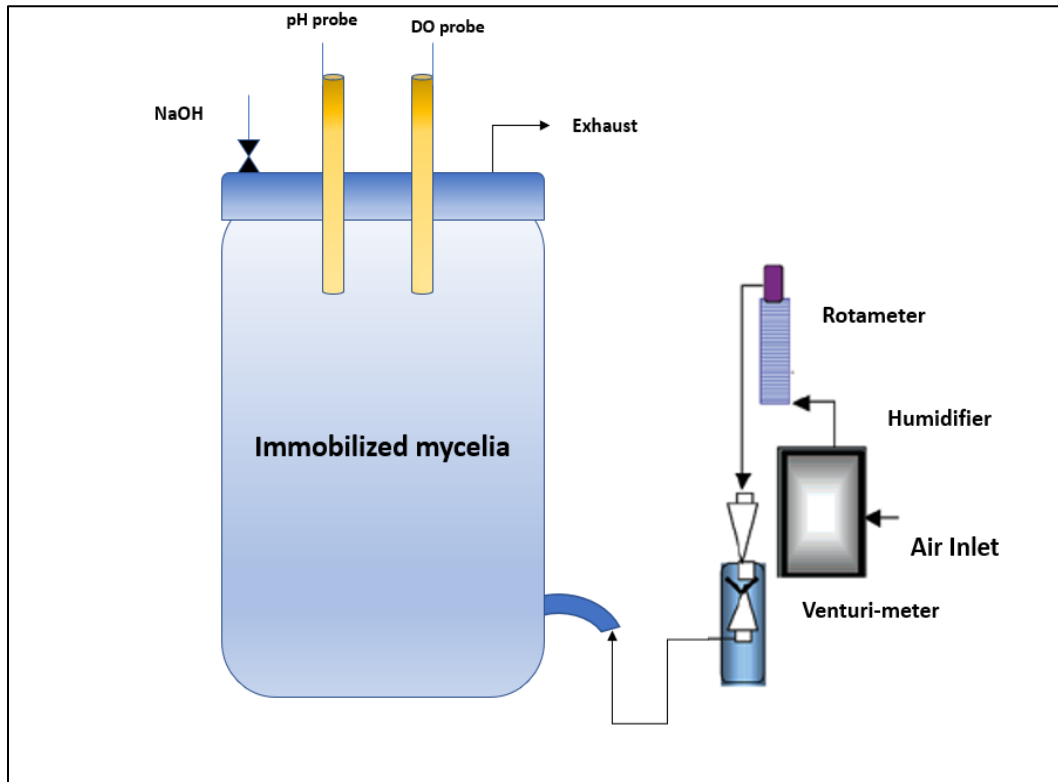
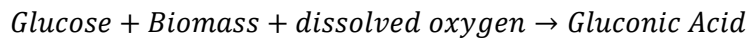


# Bio-Reactor Case Study Report



The following reaction takes place inside the above bio-reactor:



The Experimental Data of the above bio-reactor was provided which consists of the following columns:

1. Batch Number
2. Glucose Concentration (x1) (g/L)
3. Biomass Concentration (x2) (g/L)
4. Dissolved Oxygen (x3) (g/L)
5. Gluconic Acid percentage yield

Using the given dataset based on the experimental results, we need to find a model which must be able to **predict the Percentage yield of Gluconic Acid** for a given set of input  $\{x_1, x_2, x_3\}$ .

## Approach

Before we start modelling, we need to be sure that the data is properly cleaned, i.e., there are no missing or null values, and there are no outliers in the given dataset. Since the dataset is not so big and have only 46 rows, the absence of any missing values is easily identified. In order to identify outliers, I made use of Box and Whiskers Plot to identify the outliers. From the plot, I was able to identify that no outliers is present in the given dataset.

Now, since we have checked the data for any outliers and finally obtained the cleaned data, we can now proceed with modelling.

## Code and Explanation

```
function performance = performanceFn(mse,R)
    performance = (1 - R*R) + mse;
end
```

---

```
function [Parr, R2, ape, mse, netarr, trarr] = determineHiddenLayerSize(x, y, maxSize)
    Parr = zeros(maxSize, 1, "double");
    R2 = zeros(maxSize, 1, "double");
    mse = zeros(maxSize, 1, "double");
    ape = zeros(maxSize, 1, "double");
    netarr = cell(maxSize, 1);
    trarr = cell(maxSize, 1);

    for i = 1:maxSize
        % Defining the architecture of ANN
        hiddenLayerSize = i;
        net = fitnet(hiddenLayerSize, 'trainlm');
        net.divideFcn = 'dividerand';
        net.divideParam.trainRatio = 0.70;
        net.divideParam.testRatio = 0.15;
        net.divideParam.valRatio = 0.15;

        net.trainParam.showWindow = 0;
        [net, tr] = train(net, x, y); % training the model

        netarr{i} = net;
        trarr{i} = tr;

        yPredicted = net(x);

        [R, ~, ~] = regression(y, yPredicted, 'one');
        R2(i) = R*R;
        mse(i) = mean((yPredicted-y).^2);
        ape(i) = mean((abs(yPredicted-y)/y)*100);

        Parr(i) = performanceFn(mse(i), R);
    end
end
```

---

```

[x, target] = bioReactor();

maxSize = input("Enter maxSize:");

resMinP = 1e7;
resMinPidx = -1;

[Parr, R2, ape, mse, netarr, trarr] = determineHiddenLayerSize(x, target, maxSize);
for i = 1:maxSize
    if(resMinP > Parr(i))
        resMinP = Parr(i);
        resMinPidx = i;
    end
end

disp("Performance value:"); disp(resMinP);
disp("R2 value:"); disp(R2(resMinPidx));
disp("MSE value:"); disp(mse(resMinPidx));
disp("APE value:"); disp(ape(resMinPidx));

network = netarr{resMinPidx};
y = network(x);

figure, plotperform(trarr{resMinPidx});
figure, plottrainstate(trarr{resMinPidx});
figure, plotregression(target, y);

genFunction(network, 'myNeuralNetworkFunction');
y = myNeuralNetworkFunction(x);

disp(y);

save('bioReacterWorkspace');

```

---

performanceFn(mse,R)

Finds the performance of the model based on the MSE (Mean Squared Error) and R value, according to the equation:  $(1 - R^2) + \text{MSE}$ .

Lower the value, better is the model.

Argument Details:

mse = MSE (Mean Squared Error)

R = Pearson's correlation coefficient ( $R^2$  = coefficient of determination)

---

determineHiddenLayerSize(x, y, maxSize)

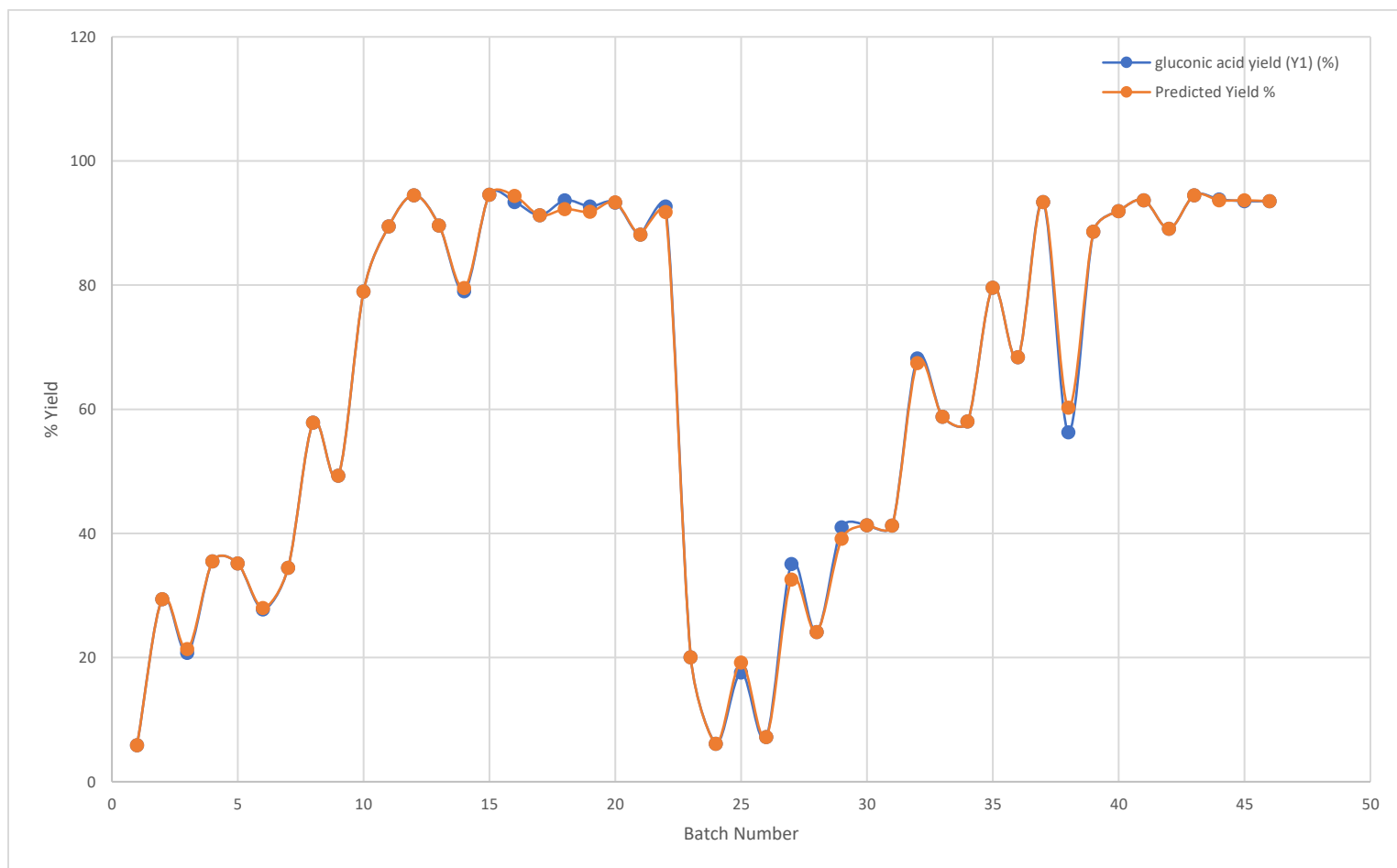
Trains the model for different layer sizes varying from 1 to maxSize. It iterates through each an every value from 1 and trains the model based on current value or the current layer size. Now, once the model is trained in the current iteration, we find the performance value that model. In this way, when the iteration is completed, we have the following arrays:

1. Performance array (Parr)
  2. Average Performance error Array (ape)
  3. Mean Squared Error Array (mse)
  4. Coefficient of Determination Array (R2)
  5. Model Array (netarr)
-

In the main code, we make use of the above defined functions and their return values to find the most suitable and optimum model. We just iterate through the arrays returned and the index value having the lowest performance score is the best model. We just maintain a variable 'network' which stores the best model after the iteration completes. Finally, this 'network' is used to get the plots and y value.

## Results

NAME	VALUE
MEAN SQUARED ERROR (MSE)	0.738319
RMSE	0.859255
R <sup>2</sup>	0.9992
AVERAGE PERCENTAGE ERROR (APE)	0.835386
NODES OR LAYER SIZE	14



From the above provided values and the graph, it is clear that the developed model is very accurate, but it shows some amount of error in some cases, but the contribution of error is very less.

**Hence, the model can be used to predict the percentage yield of Gluconic Acid from the concentration of Glucose, Biomass, and dissolved Oxygen in real life scenario.**