# Jr Data Scientist opportunity

# Implementation of neural network (from scratch) for Bank-data, Iris data, Breast cancer data and MNIST data

**Data sets used – Bank_data.csv (Comma separated value)**

**Description**: Data were extracted from images that were taken from genuine and forged banknote-like specimens. For digitization, an industrial camera usually used for print inspection was used. The final images have 400x 400 pixels. Due to the object lens and distance to the investigated object grey-scale pictures with a resolution of about 660 dpi were gained. Wavelet Transform tool were used to extract features from images.

**Bank data features are**: Image.Var, Image.Skew, Image.Curt, Entropy , Class
**Independent variable**:  Image.Var, Image.Skew, Image.Curt, Entropy
**Target vaiable**: Class (0 and 1)
**No of oberservation in Bank data sets are**: (1372, 5)

**Exploratory data analysis(EDA)**
- Checking null value: There are no null values in the data sets
- Checking data types
- Checking distribution of each column associated with target variable –  the distributions of the provided data sets are not normal
- Split the datasets into train and test: and get to know model behaviour
- Standard scalar – Making the distribution to normal distribution.

**Implementation of neural network from scratch**

Since we are having a train data of Independent features - (960, 4) Target feature - (960,)
**Implementation design/Architecture of Neural Network:**
Let us assume:

$\quad$ **X** = Input layer,
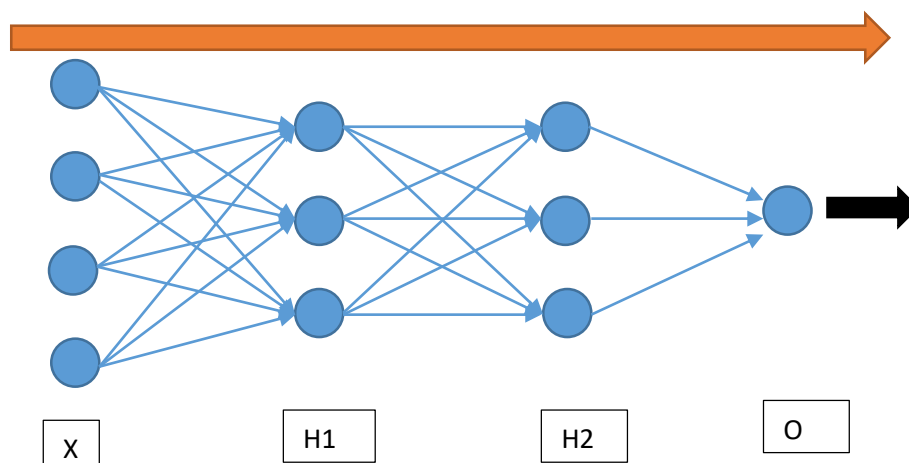$\quad$ **H1** = Hidden layer_1 (3 nodes),
$\quad$ **H2** = Hidden layer_2 (3 nodes),
$\quad$ **O** = Output layer (1 node)

**Feed Forward Neural network (flow of information from left to right)**
At each layer except Input layer two operations are done:
1) Z - Linear combination summation of weights and input layer (Weighted sum of all the inputs connected to the node)
2) A - Activation function for that node

**Weight Associated with each layer - (no of features * no of nodes in hidden layer)** and is initialised as by uniform or normal distribution

**In our Example:**

X = (960, 4), Y = (960,), W1(First hidden layer) = (4,3), W2(Second hidden layer) = (3,3),

W3(Output layer) = (3,1), Activation function for hidden layer = Ada act(K0 + K1X),

Output layer = Softmax function

| Each layer function | Calculation of output for each layer for bank datasets |
|---|---|
| At First layer:<br>Z1 = X.dot(W1) + b1<br>A1 = Ada_act(Z1) | At First layer:<br>Z1 = (960, 4).dot(4, 3) + (1,3) => (960,3)<br>A1 = Ada_act(960,3) => (960,3) |
| At second layer:<br>Z2 = A1.dot(W2) + b2<br>A2 = Ada_act(Z2) | At second layer:<br>Z2 = (960, 3).dot(3, 3) + (1,3) => (960,3)<br>A2 = Ada_act(960,3) => (960,3) |
| At output layer:<br>Z3 = A2.dot(W3) + b2<br>A3 = softmax(Z3 | At output layer:<br>Z3 = (960, 3).dot(3, 1) + (1,1) => (960,1)<br>A3 = Ada_act(960,1) => (960,1) |

**Backword propagation (flow of information from right to left)**
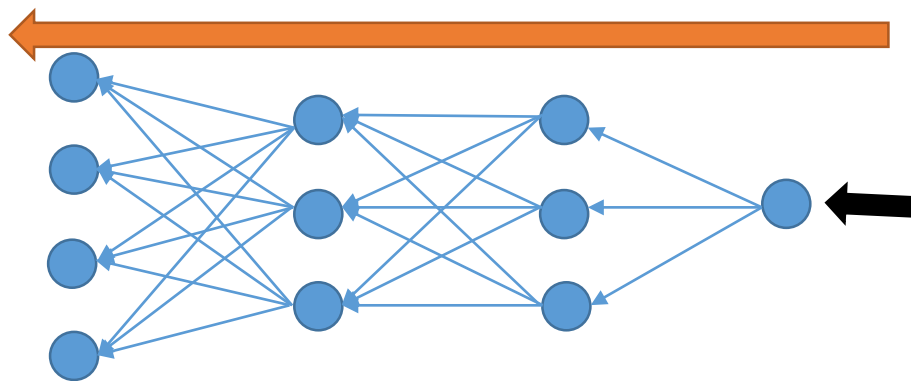


**Calculate the error from the output layer:**

**Error = (Predicted - Actual)**

Error at the output layer is due to the information flow from left to right. Hence error is associated with each layer. Weights associated with each layer is updated to minimise the error through back propagation (right to left)

**Error**

dZ3 = A3 - Y

**Error needs to propagate to second layer**

dW3 = A2.T.dot(dZ3) (change of error at layer output layer)

**Average of error at output layer**

db3 = avg of error(dZ3)

Similarly propagate all the way to input layer and update the weights

**Weight update**
New weight = old weight – alpha * rate of change of weight at that particular layer

Repeat the process until error reach to minimum.

**Test your model for test data and evaluate the model data**

Define the performance metrics:
In case of classification: f1 score, accuracy, recall, specificity


**Similarly, Neural network is implemented on _Iris data, MNIST data, and Breast cancer data_ sets, depending upon the number of attributes input and output of the neural network changes.**