Deep Learning

Goal:

Neural Networks
Python, keras, tensorflow, pytorch
Convolutional Neural Network for image and video processing
Recurrent Neural Network and NLP text and language processing

Task : 1 Binary Classification

AIM: Given a age dataset, predict whether person will buy insurance or not

Step : 1 Draw a linear regression best fit line using given data into x & y

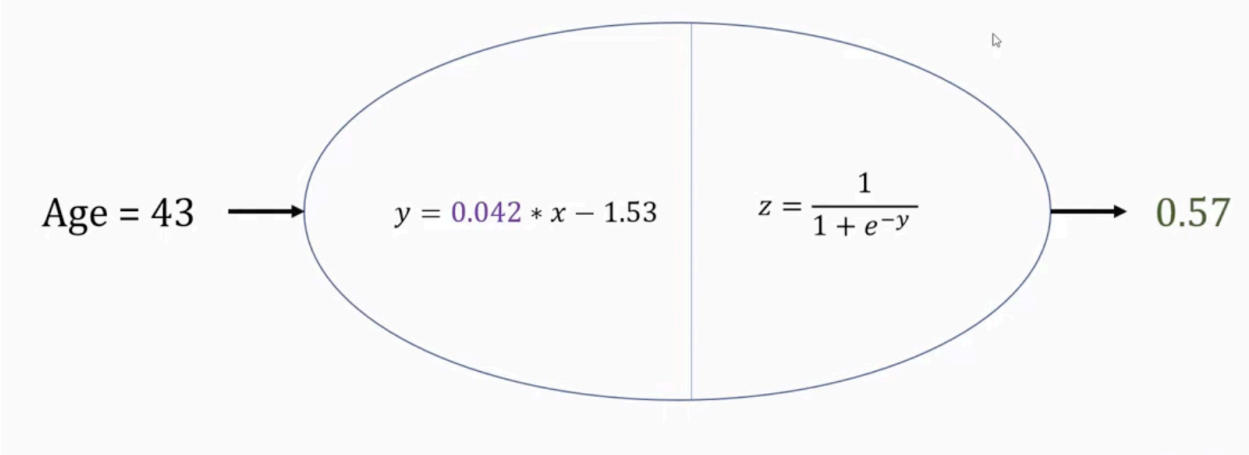
Formula : y = mx + b

Step : 2 Use y and construct sigmoid curve using sigmoid function Logistic Regression

Formula : 1/(1 + (e)^-z)

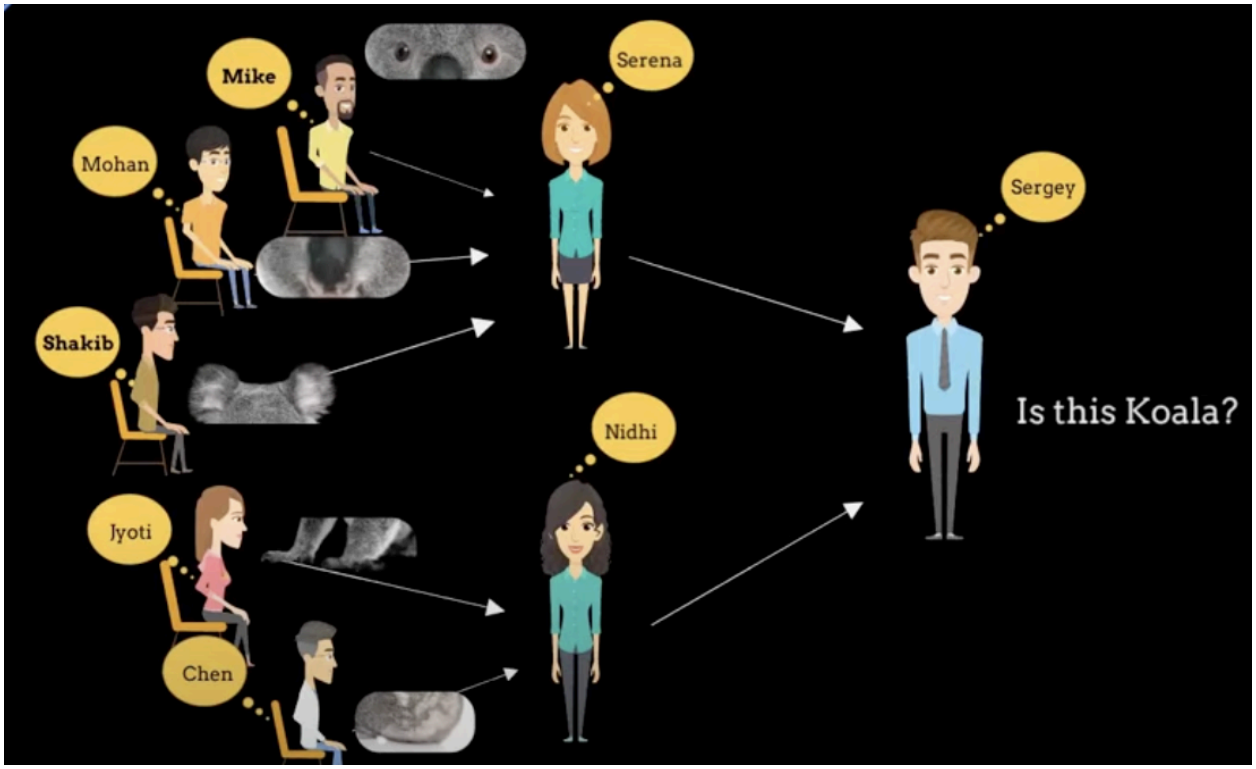
Logistic Regression - Sigmoid Curve to cover the data points - 0 or 1

Y = 0.042 * x - 1.53 (x - age)



Consider the circle as a neuron - in which it contains a activation function - and gets 0 or 1 for prediction

Neurons Explanation



Consider a set of members

Team - 1

mike, mohan, and shakib for training of predicting koalas face

Mike - eyes
Mohan - nose
Shakib - ears

Team - 2

Jyoti - legs
Chen - stomach

They all are trained for specific body parts and their numbers are sent to Serena and Nidhi to predict a number greater or less than 0.5

This prediction values goes to Sergey which should satisfy equation = face*0.6 + body*0.4

They predict it is not a koala by the above computations and values.

Final call is made to the Lab man who knows it's a koala. So they made a mistake!

The error is passed to sergey to team members, so they adjust their weights which is called **Backward Error Propagation**

DEEP LEARNING FRAMEWORKS

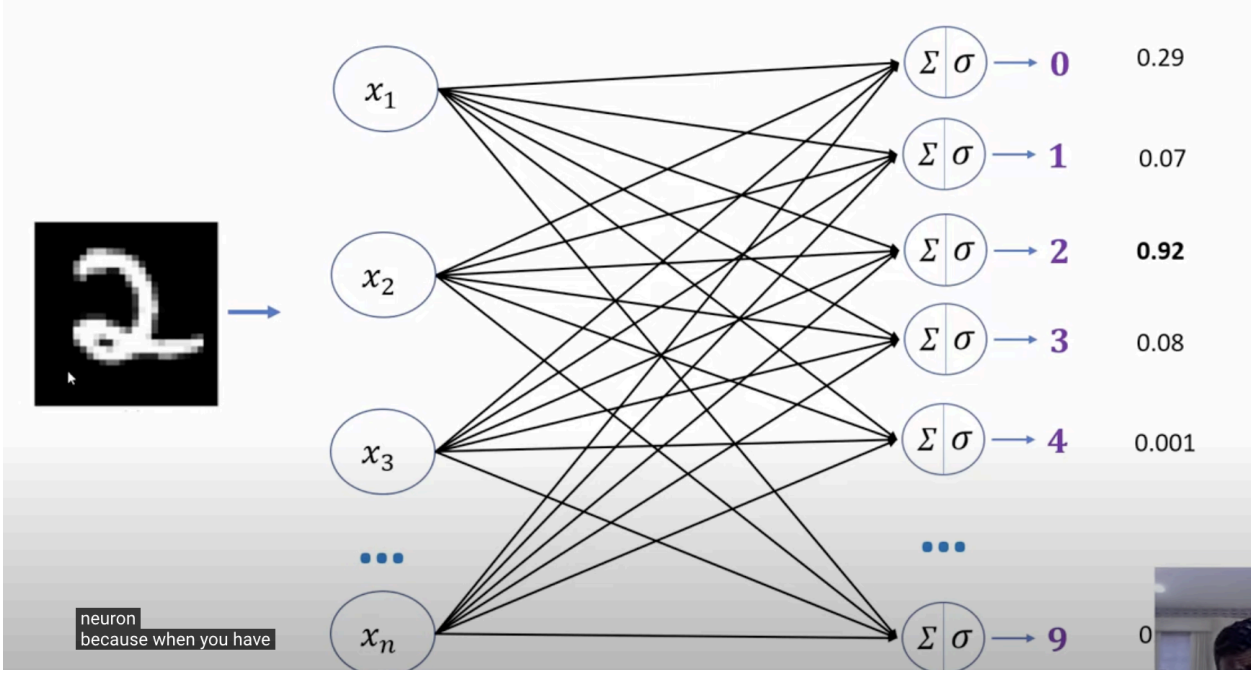
1. Pytorch - Meta
2. TensorFlow - Google

Keras is not full fledged framework but works as a wrapper for tensor flow and backend

We will use keras in-built with tensor flow to make the ease of API's

Project - I Neural Networks for Handwritten digits classification

Process



For Image Classification

Generally it is represented as a 2 dimensional array, with darker areas as 0 and white areas as 1.

Supply the image to a 2D array and you can flatten them into a 1D array. 7X7 grid - 49 input elements propagated into a neural network which has 10 output neurons. This does not have a hidden layer.

But our task contains 28 X 28 grid flattened - 784 neuron

Follow the code on Google colab - Untitled1.ipynb

- Scale the X_train and X_test for better accuracies (ie have the values between 0 and 1)
    - X_train = X_train/255
    - X_test = X_test/255

- Conversion of 2D array to 1D using reshape function from pandas
    - X_tain_flattened = X_train.reshape(len(X_train),28*28)
    - X_train.shape

- Create Neural Network with i/p and o/p neurons
    - Model = keras.Sequential([
        keras.layers.Dense(10, input_shape=(784,), activation='sigmoid')
      ])

      model.fit(X_trained_flattened, y_train, epochs = 5)

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When usin
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.8276 - loss: 0.6969
Epoch 2/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9153 - loss: 0.3056
Epoch 3/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.9197 - loss: 0.2853
Epoch 4/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.9218 - loss: 0.2784
Epoch 5/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.9236 - loss: 0.2732
<keras.src.callbacks.history.History at 0x7ffa8a4af520>
```

For the above model add hidden layer to improve performance

- Model = keras.Sequential([
    keras.layers.Dense(100, input_shape=(784,), activation='relu')
    keras.layers.Dense(10, activation='sigmoid')
  ])

  model.fit(X_trained_flattened, y_train, epochs = 5)

- Here 100 represents number of hidden layers
- Activation function used is relu