Git Link:

Video Link: https://drive.google.com/file/d/1dDt3GPRiVtSmtAqC18phJuIUzfh4\_WSn/view?usp=sharing

# TASK 1

## Use the use case in the class:

a. Add more Dense layers to the existing code and check how the accuracy changes.

A screenshot of a computer program

Description automatically generated

Explanation:

* The existing model, model\_first, is used for the diabetes dataset.
* The dataset is split into X\_train, X\_test, Y\_train, and Y\_test.
* More Dense layers are added and named as model\_second.
* model\_second is trained and evaluated.

Output:

Adding new Dense layers increased accuracy from 70% to 71.35%.

A screenshot of a computer

Description automatically generated

## Change the data source to Breast Cancer dataset \* available in the source code folder and make required changes. Report accuracy of the model.

## Normalize the data before feeding the data to the model and check how the normalization change your accuracy (code given below).

A screenshot of a computer program

Description automatically generated

**Explanation**

1. The Breast Cancer dataset is loaded using scikit-learn.
2. Standardization is applied using scikit-learn preprocessing.
3. The dataset is split into 80% training data and 20% test data, creating X\_train, X\_test, y\_train, and y\_test.
4. A Sequential model with 5 layers is created based on the number of features:
   * Input Dense layer with ReLU activation.
   * Three Dense layers with 64, 32, and 16 neurons, each using ReLU activation.
   * Output layer with 1 neuron and Sigmoid activation.
5. The model is compiled with binary crossentropy loss, Adam optimizer, and accuracy metric.
6. Training is done for 100 epochs with a batch size of 3.
7. After training, the model is evaluated on the test set, and loss, accuracy, and model summary are printed.

Output:

**The model accuracy is 94.7**

A screenshot of a computer

Description automatically generated

# TASK 2

* Plot the loss and accuracy for both training data and validation data using the history object in the source code

A screenshot of a computer code

Description automatically generated

**Explanation**

Two plots are created for loss and accuracy of both training and validation using the model's history object.

**Output:**

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generated

* Plot one of the images in the test data, and then do inferencing to check what is the prediction of the model on that single image.

A screen shot of a computer code

Description automatically generated

Explanation:

1. A random index is selected based on the number of test images.
2. The selected image is reshaped and normalized before prediction.
3. The predicted class is obtained and printed.
4. The image is displayed along with the predicted class.

Output:

Predicted class is 4:

A yellow and blue symbol

Description automatically generated

* We had used 2 hidden layers and Relu activation. Try to change the number of hidden layer and the activation to tanh or sigmoid and see what happens.

A screenshot of a computer code

Description automatically generated

**Explanation and Output:**

The model is modified by changing the number of hidden layers and using the Tanh activation function. This adjustment increases accuracy from 97.8% to 98.2%.

* 4. Run the same code without scaling the images and check the performance?

A screenshot of a computer program

Description automatically generated

**Explanation:**

The model structure remains the same, including the number of layers and activation functions. However, the input size is changed because the model is trained without scaling the images. This adjustment impacts the model's performance and accuracy.

**Output:**

The model trained without scaling the images shows a drop in performance, with accuracy decreasing from 98.2% to 95.4%.

A screenshot of a computer

Description automatically generated