**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

The given code implements a neural network using Keras to predict diabetes based on a dataset. The neural network consists of multiple Dense layers, and the code trains the model on the training data and evaluates its performance on the test data. The model's accuracy is recorded after training and testing.

To improve the model's accuracy, two additional Dense layers have been added to the neural network, increasing its complexity. After adding the new layers, the code trains the updated model on the training data and evaluates its performance on the test data again to see how the accuracy changes with the added layers.

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

**A screenshot of a computer program

Description automatically generated**

The given code implements a neural network using Keras to classify breast cancer data. It uses the breast cancer dataset from sklearn, splits it into training and testing sets, and builds a neural network with two Dense layers (with 20 and 10 neurons, respectively) followed by an output Dense layer (with 1 neuron) using the ReLU activation function. The model is trained on the training data and evaluated on the test data to calculate the loss and accuracy.

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

**from sklearn.preprocessing import StandardScaler**

**sc = StandardScaler()**

**A screenshot of a computer code

Description automatically generated**

The given code implements a neural network using Keras to classify breast cancer data after preprocessing it with StandardScaler. It scales the input features, splits the data into training and testing sets, and builds a neural network with two Dense layers (with 20 and 10 neurons, respectively) followed by an output Dense layer (with 1 neuron) using the ReLU activation function for the hidden layers and the sigmoid activation function for the output layer. The model is trained on the training data and evaluated on the test data to calculate the loss and accuracy of the model in predicting breast cancer.

**PROBLEM 2**

1. **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 program

Description automatically generated**

**OUTPUT**

**A graph of a graph of a model

Description automatically generated**

The code provided plots the model's training and testing accuracy as well as the training and testing loss over epochs. These plots help visualize the performance of the neural network during training and provide insights into potential overfitting or underfitting.

1. **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 screenshot of a computer program

Description automatically generated

**OUTPUT**

**A number seven in a black background

Description automatically generated**

The code displays the first test image from the "test\_images" dataset using a grayscale colormap. Then, it uses the trained neural network model to predict the label of the displayed image and prints the predicted class label.

1. **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 program

Description automatically generated**

**OUTPUT**

**A screenshot of a graph

Description automatically generated**

The provided code trains two neural network models with different activation functions (tanh and sigmoid) and the same architecture. The models have four Dense layers with decreasing neurons in each layer. The final Dense layer has 10 neurons with a softmax activation function for multi-class classification. The models are trained using RMSprop optimizer and categorical cross-entropy loss for 10 epochs, and the training progress is recorded in the "history" variable for each model.

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

**A screenshot of a computer program

Description automatically generated**

**OUTPUT**

**A screenshot of a computer

Description automatically generated**

The provided code prepares the image data and their corresponding labels for training and testing. It then builds a neural network model with two hidden Dense layers, each having 512 neurons with the ReLU activation function, and an output Dense layer with 10 neurons and a softmax activation function for multi-class classification. The model is trained using RMSprop optimizer and categorical cross-entropy loss for 10 epochs. The training progress is recorded in the "history" variable.

**GitHub URL:** **https://github.com/TriveniBala/NEURAL-NETWORKS-ASSIGNMENT-2**