ICP – 6

GIT REPO : https://github.com/YaminiEddala/Neural\_ICP6

NAME : Yamini E( 700756713 )

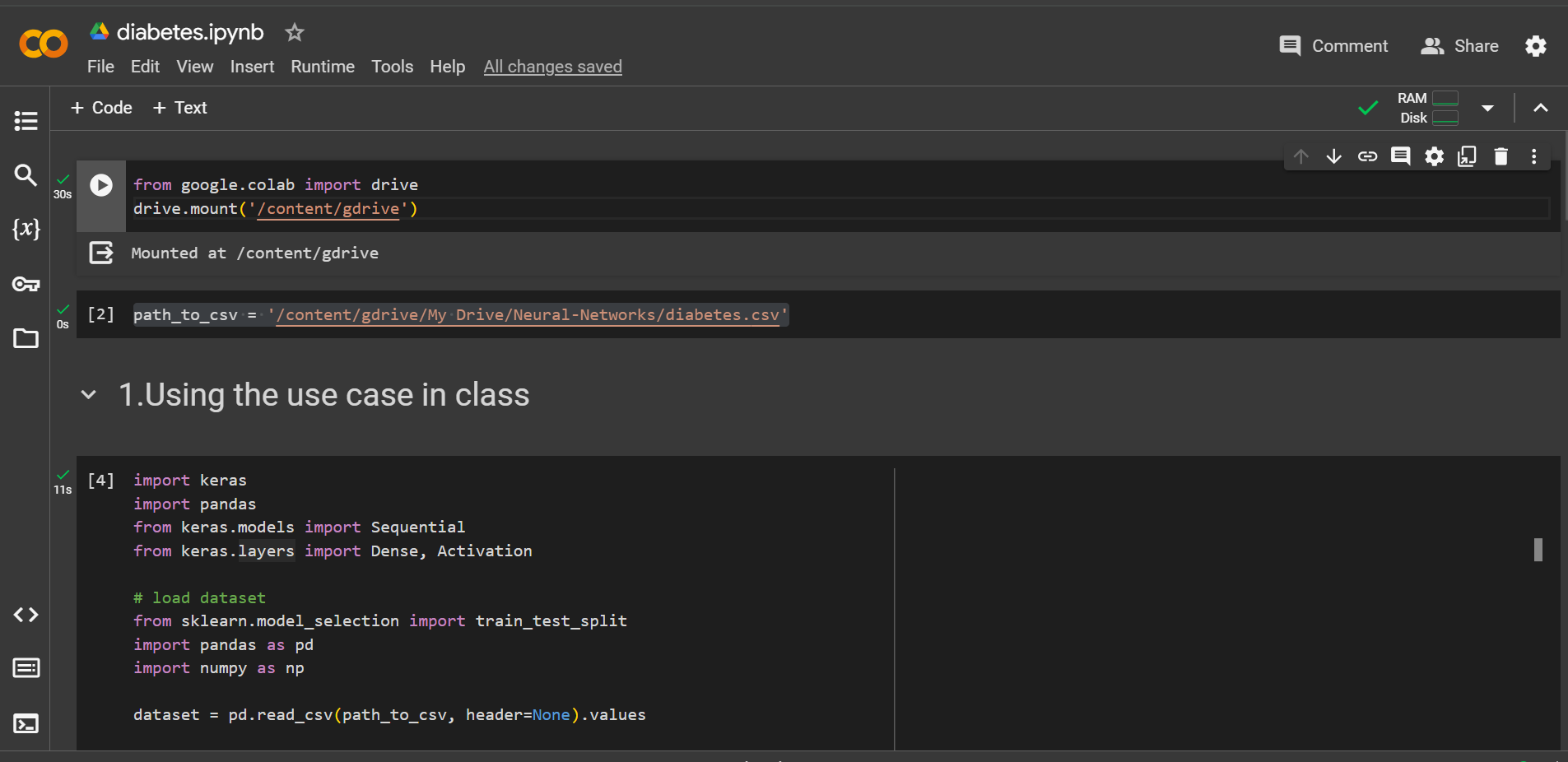
**ICP\_Basics in Keras:**

1. **Use the use case in the class:**

The code is implementing a simple neural network using the Keras API. The dataset is loaded using pandas from a CSV file and split into training and testing sets using the train\_test\_split function from scikit-learn.

The neural network has one hidden layer with 20 nodes, an input layer with 8 nodes (corresponding to the 8 features in the dataset), and an output layer with a single node (as it is a binary classification task). The activation function used in the hidden layer is ReLU, and the activation function used in the output layer is sigmoid.

The neural network is compiled using the binary\_crossentropy loss function, Adam optimizer, and accuracy as the evaluation metric. The model is then trained on the training set for 100 epochs using the fit method, and the summary of the model is printed using the summary method. Finally, the accuracy of the model is evaluated on the testing set using the evaluate method, and the loss and accuracy are printed



A screen shot of a computer program

Description automatically generated

A black and white screen

Description automatically generatedA screen shot of a computer

Description automatically generatedA screen shot of a computer

Description automatically generated’

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

The improved neural network model features three hidden layers, each with 20 neurons, in contrast to the original model's single hidden layer. This enhancement allows for more complex feature learning from the 8 input features, leading to better performance. The model utilizes a sigmoid activation function in the output layer, binary cross-entropy loss, and the Adam optimizer, with accuracy as the performance metric. Training occurs over 100 epochs with no output verbosity. The model's architecture and trainable parameters are detailed through the **summary()** method. Evaluation on the test set indicates an increase in accuracy, attributed to the model's enhanced capacity for learning and optimization due to its additional layers and parameters.

A screenshot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

I have further made changes to the model. The model has an input layer, three hidden layers with 20, 22, and 24 neurons respectively, and an output layer with one neuron. The activation function used in the hidden layers is ReLU, and the output layer uses the sigmoid activation function.

The model is then compiled with binary\_crossentropy loss and adam optimizer. The metrics parameter is set to ['acc'] to evaluate the accuracy of the model during training.

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

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

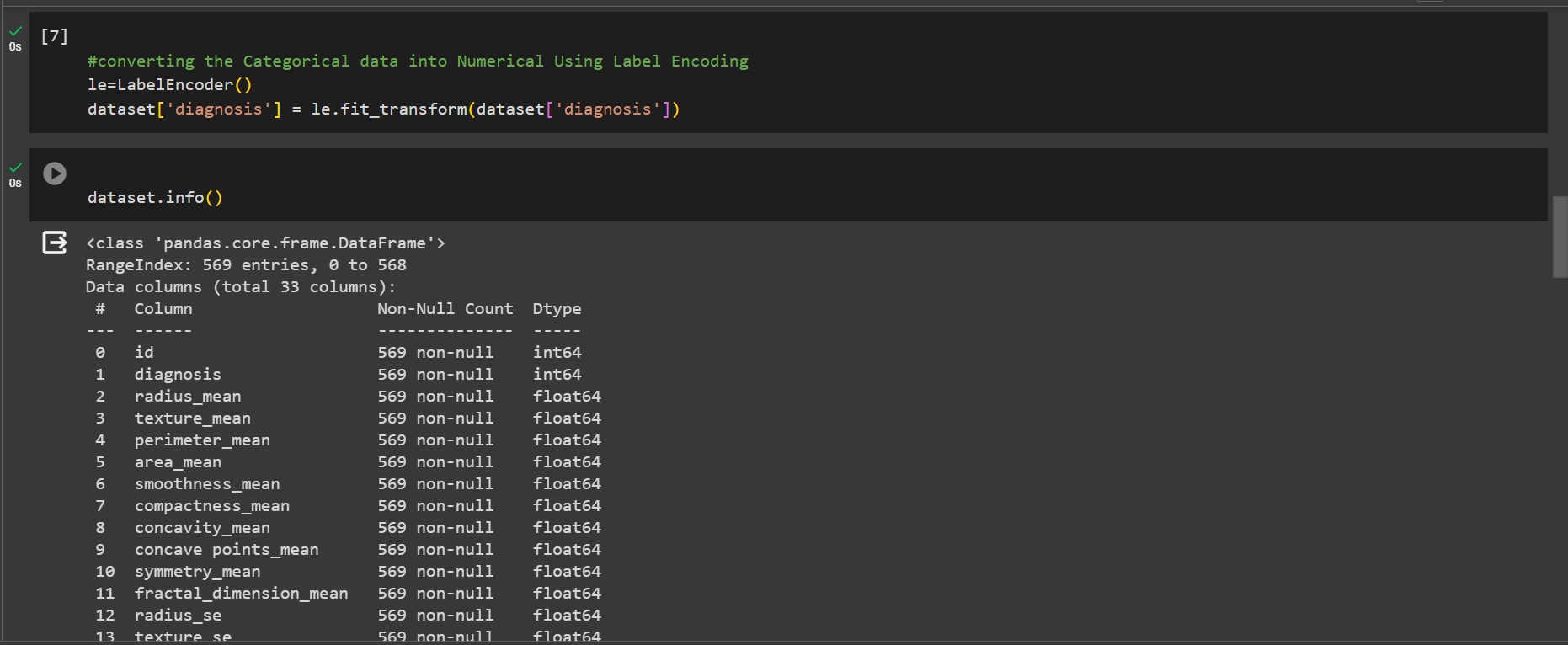
I have now changed the previous dataset to a new dataset which is from the breastcancer.csv file. The code to this question is available in the python file named breastcancer.ipynb.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated



A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

1. **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()**

my\_first\_nn.add(Dense(1, activation='sigmoid')) adds a dense output layer with 1 neuron and sigmoid activation function.

my\_first\_nn.compile(loss='binary\_crossentropy', optimizer='adam',metrics=['acc']) compiles the model with binary\_crossentropy loss function, adam optimizer, and accuracy metric.

sc = StandardScaler() creates a StandardScaler object.

X\_train = sc.fit\_transform(X\_train) fits the StandardScaler object to the training data and standardizes it.

X\_test = sc.transform(X\_test) standardizes the test data using the previously fitted StandardScaler.

my\_first\_nn\_fitted = my\_first\_nn.fit(X\_train, Y\_train, epochs=100, verbose=0, initial\_epoch=0) trains the model on the standardized training data for 100 epochs.

This output below shows the summary of a neural network model with two layers: a hidden layer with 20 neurons, and an output layer with a single neuron. The input dimension of the hidden layer is 30, which is not shown in the summary because it is automatically inferred from the input data. In this case, the model achieves an accuracy of 0.9650 and a loss of 0.1907 on the test data

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A black screen with colorful text

Description automatically generated

 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

Description automatically generated

A screenshot of a computer

Description automatically generated

A graph with a line and a line

Description automatically generated with medium confidence

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

Description automatically generated

A screenshot of a computer program

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 program

Description automatically generated

A black screen with white text

Description automatically generated

A screen shot of a computer code

Description automatically generated

A black and white screen

Description automatically generated

A computer screen shot of a black screen

Description automatically generated

A black screen with white text

Description automatically generated

A screen shot of a computer

Description automatically generated

A black and white screen

Description automatically generated

A black and white text

Description automatically generated

Run the same code without scaling the images and check the performance?

A screenshot of a computer program

Description automatically generated

A screen shot of a computer code

Description automatically generated

A screen shot of a computer

Description automatically generated

