**ICP3**

**700756561**

**TEJENDRA KONDURU**

**Program1:**

**import numpy as np**

**from keras.models import Sequential**

**from keras.layers import Dense**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.datasets import load\_breast\_cancer**

**from sklearn.preprocessing import StandardScaler**

**# Load dataset**

**from google.colab import files**

**uploaded = files.upload()**

**dataset = pd.read\_csv("diabetes.csv", header=None).values**

**# Split the dataset into training and testing sets**

**X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(dataset[:, 0:8], dataset[:, 8], test\_size=0.25, random\_state=87)**

**# Set random seed for reproducibility**

**np.random.seed(155)**

**# Create the model with additional Dense layers**

**model = Sequential()**

**model.add(Dense(20, input\_dim=8, activation='relu'))  # Hidden layer**

**model.add(Dense(15, activation='relu'))  # Additional layer**

**model.add(Dense(2, activation='relu'))  # Additional layer**

**model.add(Dense(1, activation='sigmoid'))  # Output layer**

**# Compile the model**

**model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])**

**# Fit the model**

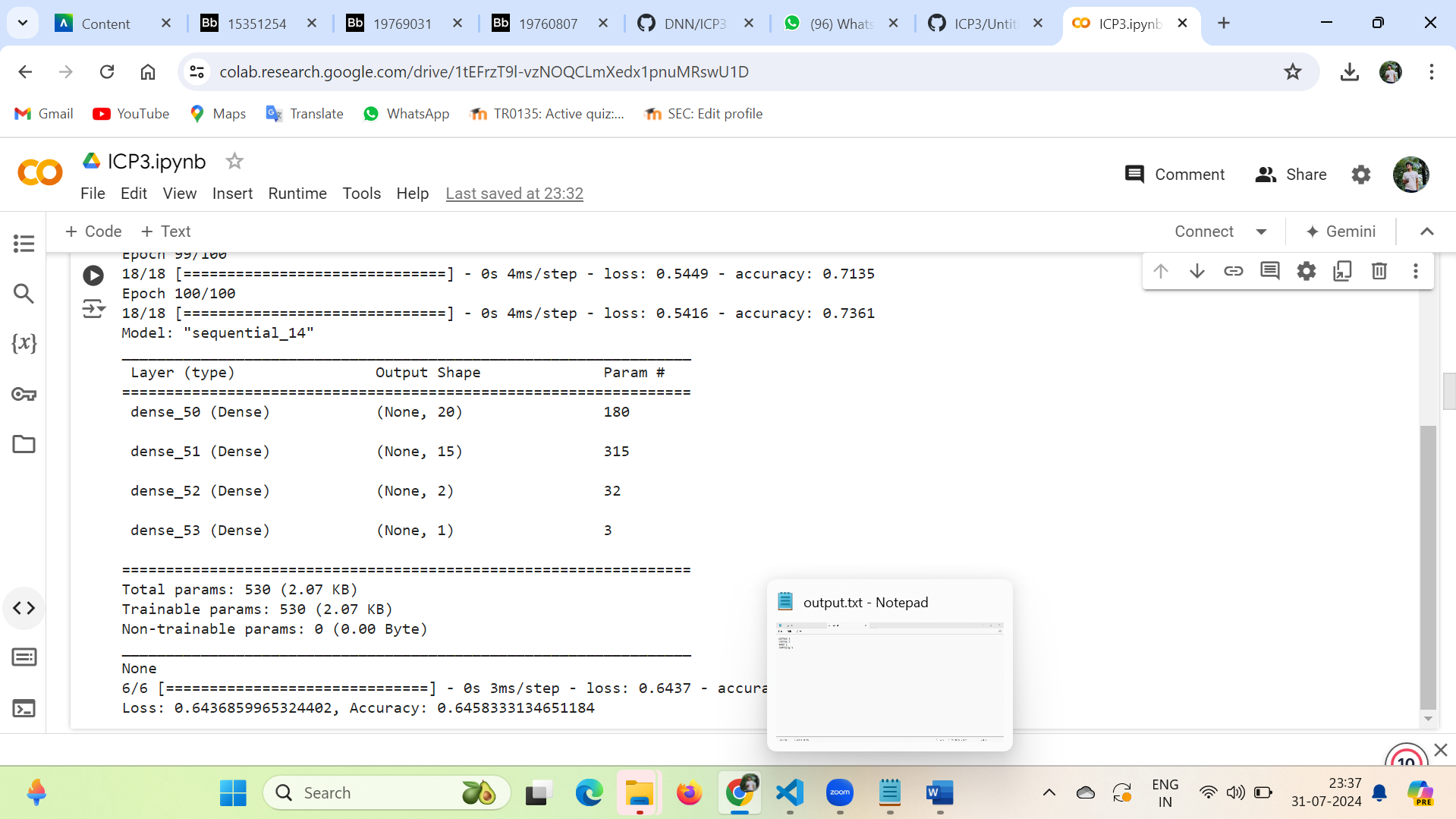
**model.fit(X\_train, Y\_train, epochs=100, initial\_epoch=0, verbose=1)**

**# Print the model summary**

**print(model.summary())**

**evaluation = model.evaluate(X\_test, Y\_test)**

**print(f"Loss: {evaluation[0]}, Accuracy: {evaluation[1]}")**

****

**Program2:**

**import numpy as np**

**import matplotlib.pyplot as plt**

**from keras.datasets import mnist**

**from keras.models import Sequential**

**from keras.layers import Dense**

**from keras.utils import to\_categorical**

**# Load the MNIST dataset**

**(train\_images, train\_labels), (test\_images, test\_labels) = mnist.load\_data()**

**# Process the data**

**dimData = np.prod(train\_images.shape[1:])**

**train\_data = train\_images.reshape(train\_images.shape[0], dimData)**

**test\_data = test\_images.reshape(test\_images.shape[0], dimData)**

**# Convert data to float and scale values between 0 and 1**

**train\_data = train\_data.astype('float32')**

**test\_data = test\_data.astype('float32')**

**train\_data /= 255.0**

**test\_data /= 255.0**

**# Change the labels from integer to one-hot encoding**

**train\_labels\_one\_hot = to\_categorical(train\_labels)**

**test\_labels\_one\_hot = to\_categorical(test\_labels)**

**# Creating the network**

**model = Sequential()**

**model.add(Dense(512, activation='relu', input\_shape=(dimData,)))**

**model.add(Dense(512, activation='relu'))**

**model.add(Dense(10, activation='softmax'))**

**model.compile(optimizer='rmsprop', loss='categorical\_crossentropy', metrics=['accuracy'])**

**# Train the model**

**history = model.fit(train\_data, train\_labels\_one\_hot, batch\_size=256, epochs=10, verbose=1,**

**validation\_data=(test\_data, test\_labels\_one\_hot))**

**# Task 1: Plot the loss and accuracy for both training data and validation data**

**def plot\_history(history):**

**plt.figure(figsize=(12, 4))**

**plt.subplot(1, 2, 1)**

**plt.plot(history.history['loss'], label='Training Loss')**

**plt.plot(history.history['val\_loss'], label='Validation Loss')**

**plt.title('Loss')**

**plt.xlabel('Epochs')**

**plt.ylabel('Loss')**

**plt.legend()**

**plt.subplot(1, 2, 2)**

**plt.plot(history.history['accuracy'], label='Training Accuracy')**

**plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')**

**plt.title('Accuracy')**

**plt.xlabel('Epochs')**

**plt.ylabel('Accuracy')**

**plt.legend()**

**plt.show()**

**plot\_history(history)**

**# Task 2: Plot one of the images in the test data and predict**

**def plot\_image\_and\_predict(index):**

**plt.imshow(test\_images[index], cmap='gray')**

**plt.title(f"True Label: {test\_labels[index]}")**

**plt.show()**

**# Preprocess the image**

**img = test\_images[index].reshape(1, dimData).astype('float32') / 255.0**

**# Predict the class**

**prediction = model.predict(img)**

**predicted\_class = np.argmax(prediction)**

**print(f"Predicted Label: {predicted\_class}")**

**plot\_image\_and\_predict(0)**

**# Task 3: Change the number of hidden layers and activation function to tanh or sigmoid**

**def build\_and\_train\_model(layers, activation):**

**model = Sequential()**

**model.add(Dense(512, activation=activation, input\_shape=(dimData,)))**

**for \_ in range(layers - 1):**

**model.add(Dense(512, activation=activation))**

**model.add(Dense(10, activation='softmax'))**

**model.compile(optimizer='rmsprop', loss='categorical\_crossentropy', metrics=['accuracy'])**

**history = model.fit(train\_data, train\_labels\_one\_hot, batch\_size=256, epochs=10, verbose=1,**

**validation\_data=(test\_data, test\_labels\_one\_hot))**

**return history**

**# Example with 3 hidden layers and 'tanh' activation**

**history\_tanh = build\_and\_train\_model(3, 'tanh')**

**plot\_history(history\_tanh)**

**# Example with 3 hidden layers and 'sigmoid' activation**

**history\_sigmoid = build\_and\_train\_model(3, 'sigmoid')**

**plot\_history(history\_sigmoid)**

**# Task 4: Run the same code without scaling the images and check the performance**

**train\_data\_no\_scale = train\_images.reshape(train\_images.shape[0], dimData).astype('float32')**

**test\_data\_no\_scale = test\_images.reshape(test\_images.shape[0], dimData).astype('float32')**

**model\_no\_scale = Sequential()**

**model\_no\_scale.add(Dense(512, activation='relu', input\_shape=(dimData,)))**

**model\_no\_scale.add(Dense(512, activation='relu'))**

**model\_no\_scale.add(Dense(10, activation='softmax'))**

**model\_no\_scale.compile(optimizer='rmsprop', loss='categorical\_crossentropy', metrics=['accuracy'])**

**history\_no\_scale = model\_no\_scale.fit(train\_data\_no\_scale, train\_labels\_one\_hot, batch\_size=256, epochs=10, verbose=1,**

**validation\_data=(test\_data\_no\_scale, test\_labels\_one\_hot))**

**plot\_history(history\_no\_scale)**

**A screenshot of a computer screen

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**GitHub Link:**

<https://github.com/teja375/DNN/tree/main/ICP3>