Spring 2024: CS5720 Neural Networks & Deep Learning - ICP-6
Bhanu Chandrika Lakkimsetti (700747439)
ICP_Basics in Keras

GitHub Link: https://github.com/bhanuchandrika99/NNDL_ICP_6

Use Case Description: Predicting the diabetes disease
Programming elements: Keras Basics

In class programming:

1. Use the use case in the class:

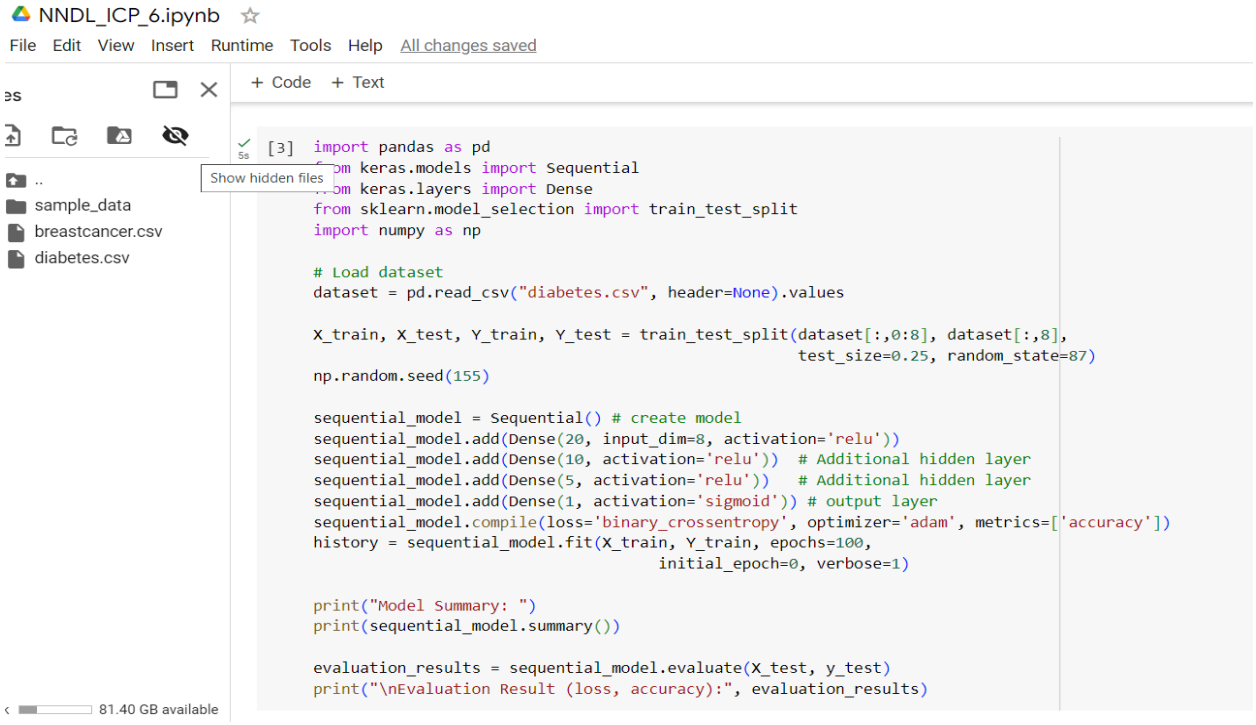
+ Code + Text

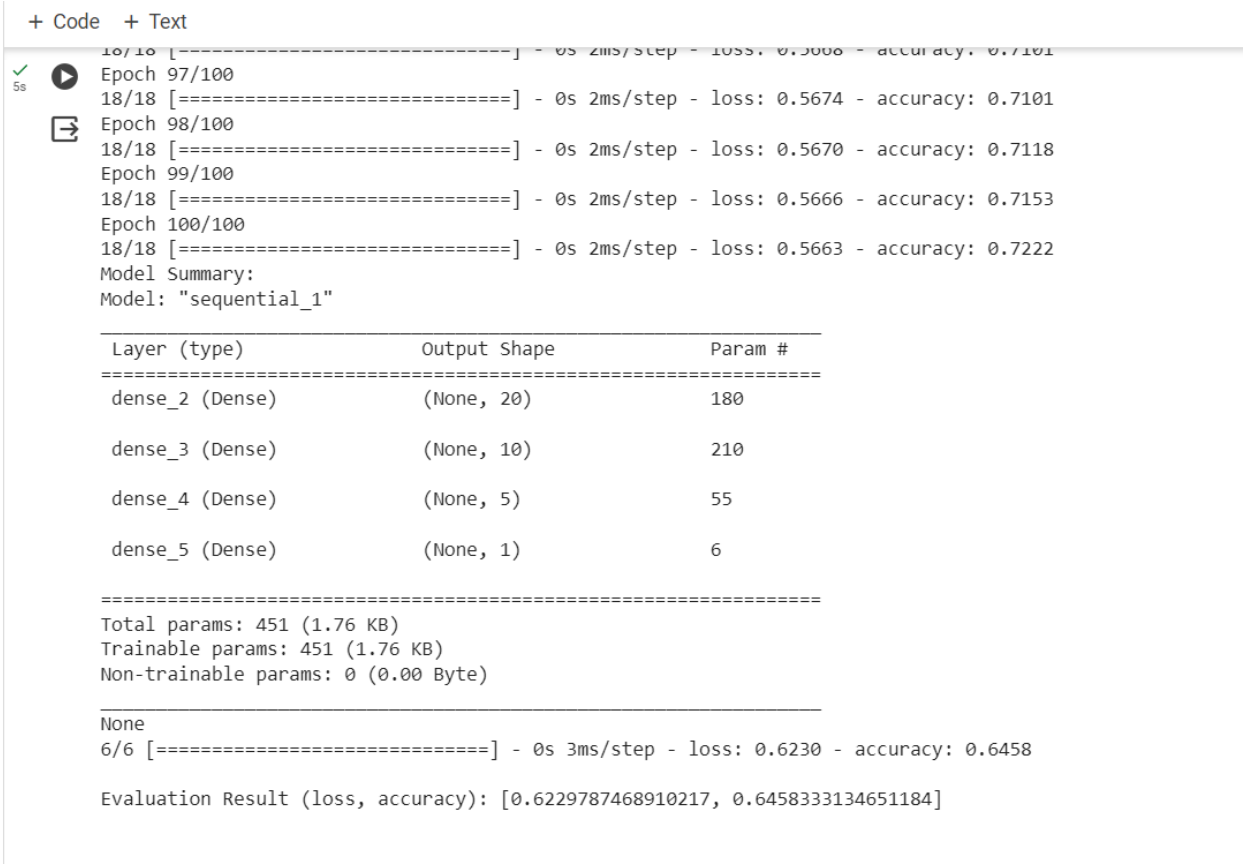
```
Epoch 1/100
18/18 [==============================] - 2s 2ms/step - loss: 22.6529 - accuracy: 0.6615
Epoch 2/100
18/18 [==============================] - 0s 2ms/step - loss: 13.0603 - accuracy: 0.6389
Epoch 3/100
18/18 [==============================] - 0s 2ms/step - loss: 9.5961 - accuracy: 0.6024
Epoch 4/100
18/18 [==============================] - 0s 2ms/step - loss: 7.7556 - accuracy: 0.6007
Epoch 5/100
18/18 [==============================] - 0s 2ms/step - loss: 6.3340 - accuracy: 0.5781
Epoch 6/100
18/18 [==============================] - 0s 2ms/step - loss: 5.1204 - accuracy: 0.5608
Epoch 7/100
18/18 [==============================] - 0s 2ms/step - loss: 4.0866 - accuracy: 0.5451
Epoch 8/100
18/18 [==============================] - 0s 2ms/step - loss: 3.0615 - accuracy: 0.5417
Epoch 9/100
18/18 [==============================] - 0s 2ms/step - loss: 2.1218 - accuracy: 0.5451
Epoch 10/100
18/18 [==============================] - 0s 2ms/step - loss: 1.4289 - accuracy: 0.5868
Epoch 11/100
18/18 [==============================] - 0s 2ms/step - loss: 1.0858 - accuracy: 0.5990
Epoch 12/100
18/18 [==============================] - 0s 2ms/step - loss: 0.9967 - accuracy: 0.6389
Epoch 13/100
18/18 [==============================] - 0s 2ms/step - loss: 0.8826 - accuracy: 0.6615
Epoch 14/100
18/18 [==============================] - 0s 2ms/step - loss: 0.8023 - accuracy: 0.6684
```

+ Code + Text

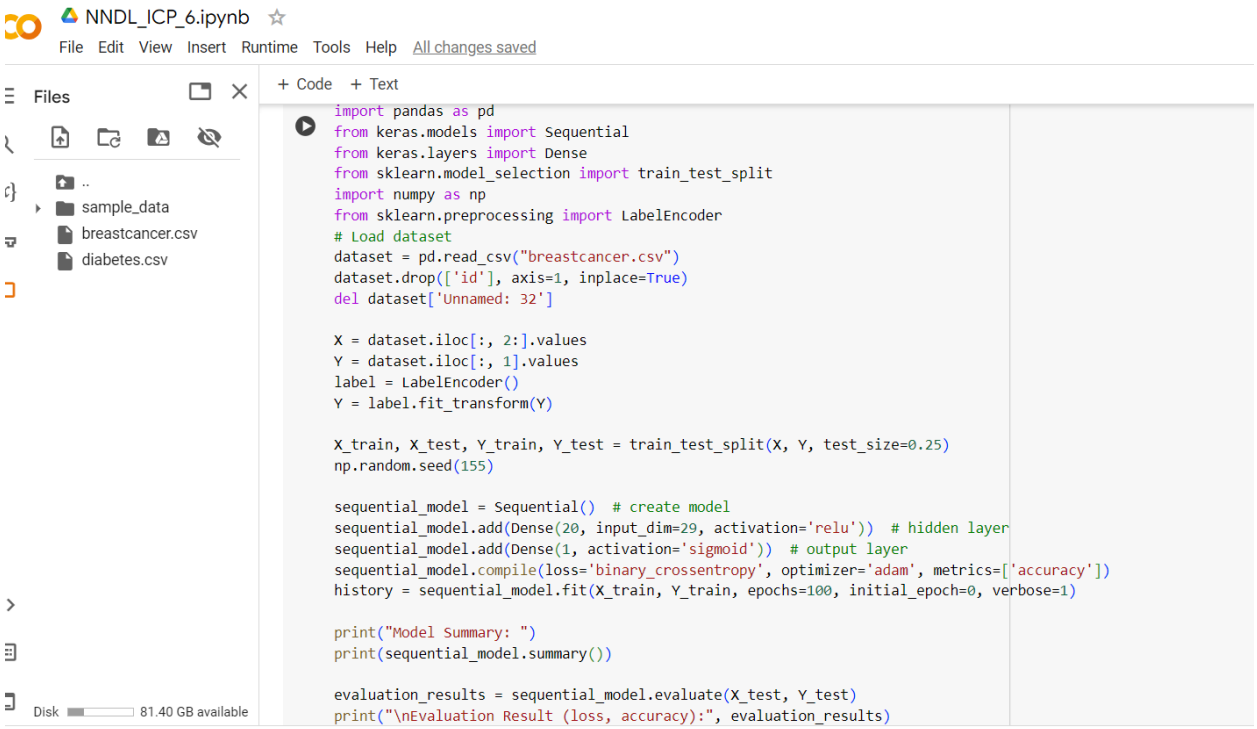
```
18/18 [==============================] - 0s 3ms/step - loss: 0.5731 - accuracy: 0.7344
Epoch 95/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6055 - accuracy: 0.7066
Epoch 96/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5670 - accuracy: 0.7101
Epoch 97/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5554 - accuracy: 0.7188
Epoch 98/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5633 - accuracy: 0.7222
Epoch 99/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5734 - accuracy: 0.7118
Epoch 100/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6140 - accuracy: 0.6944
Model Summary:
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_10 (Dense)            (None, 20)                180

 dense_11 (Dense)            (None, 1)                 21

=================================================================
Total params: 201 (804.00 Byte)
Trainable params: 201 (804.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
6/6 [==============================] - 0s 3ms/step - loss: 0.6438 - accuracy: 0.6823

Evaluation Result (loss, accuracy): [0.6437555551528931, 0.6822916865348816]
```

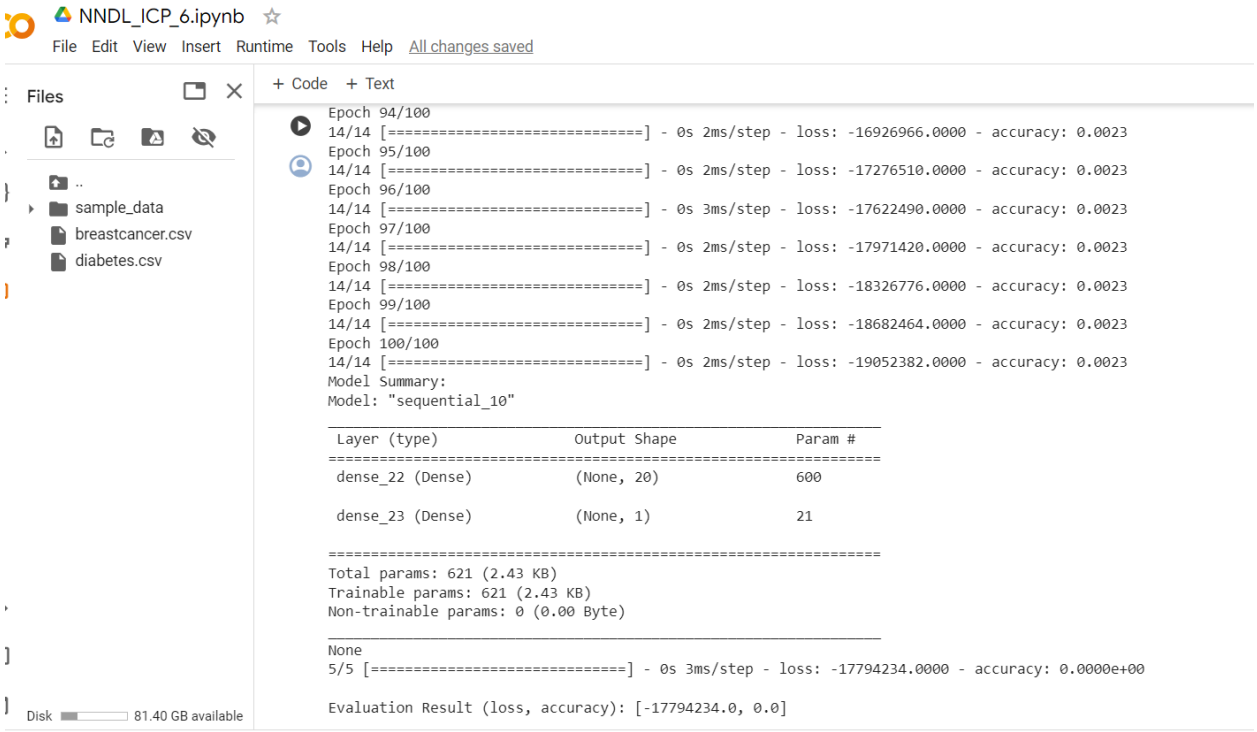
a. Add more Dense layers to the existing code and check how the accuracy changes.



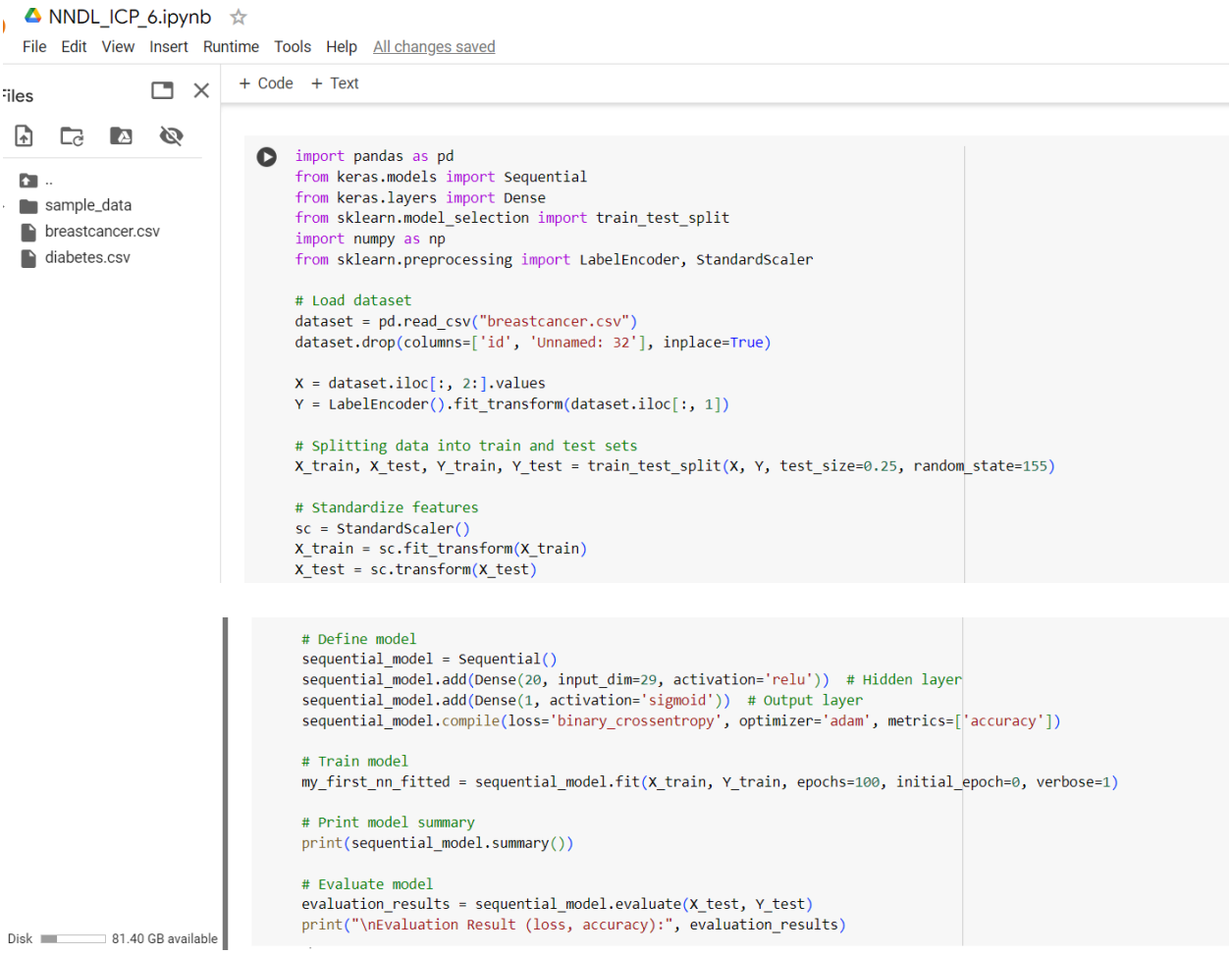


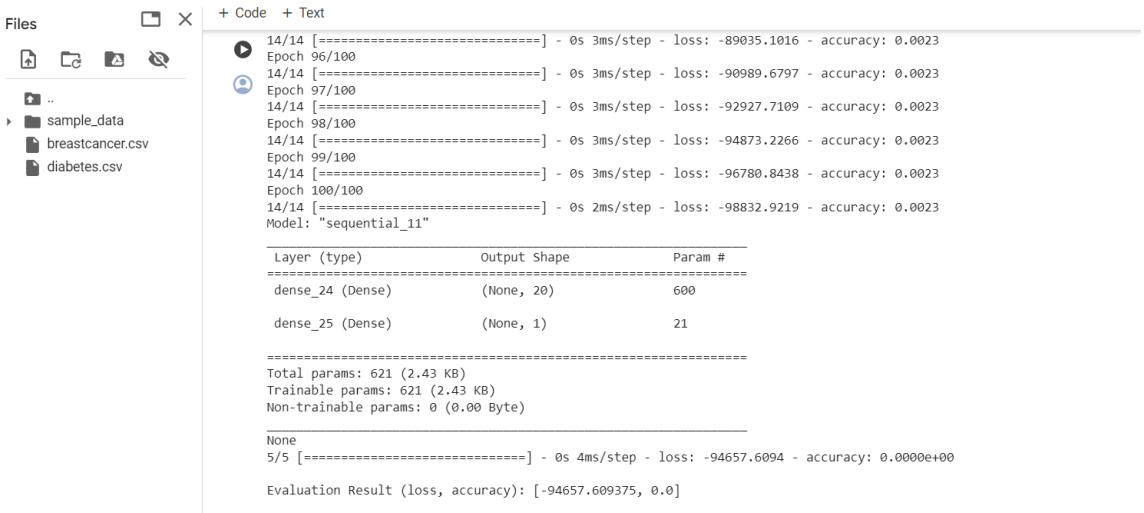
2. Change the data source to Breast Cancer dataset * available in the source code folder and make required changes. Report accuracy of the model.





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()

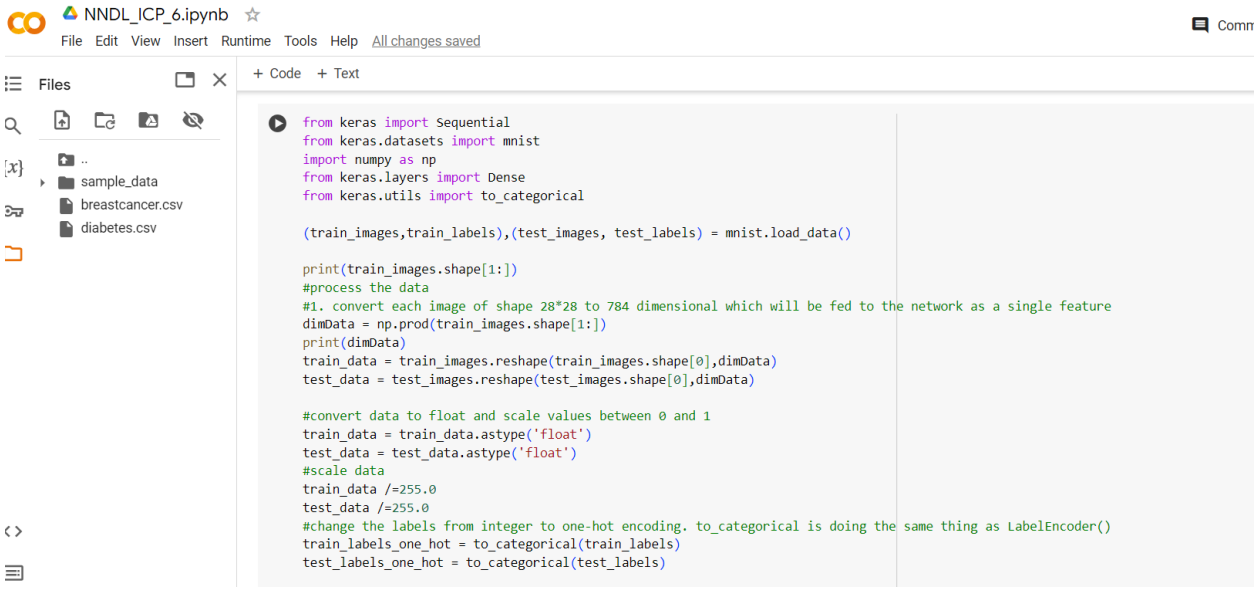


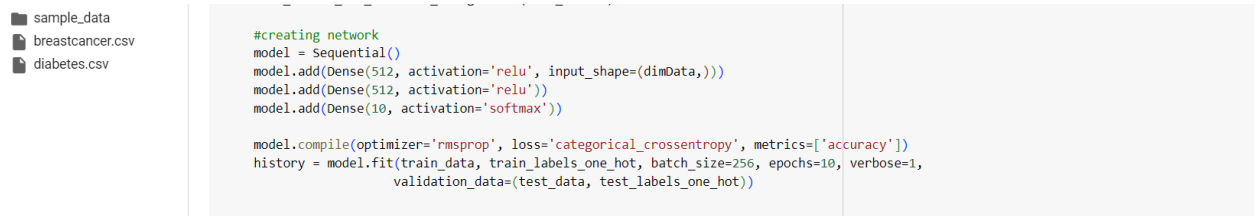


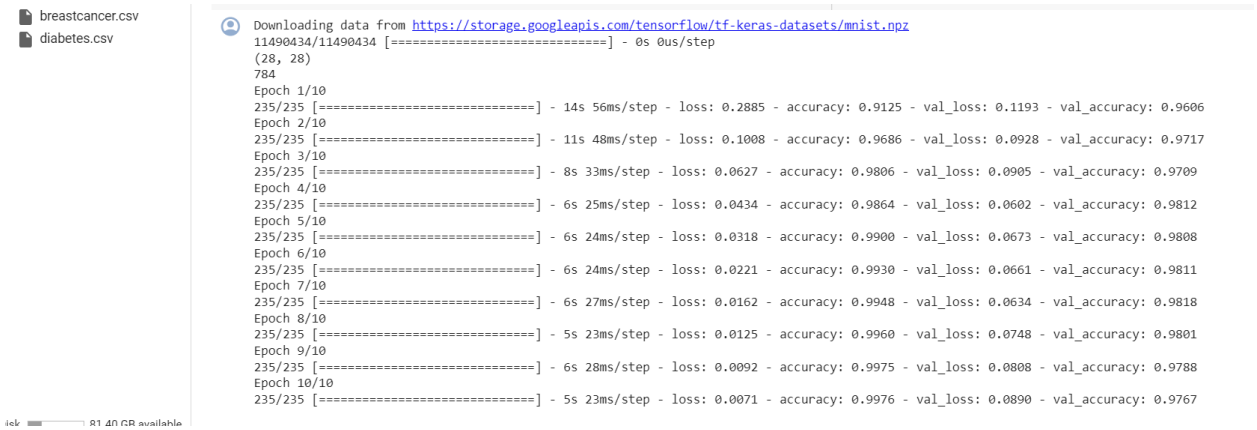
Note :Breast Cancer dataset is designated to predict if a patient has Malignant (M) or Benign = B cancer

In class programming:

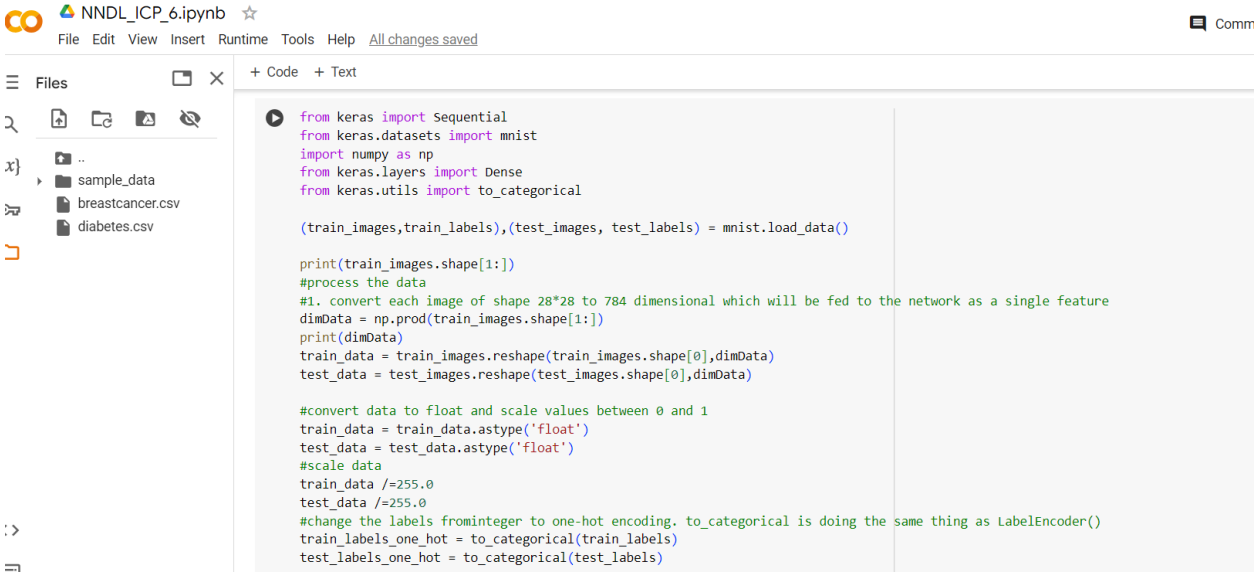
Use Image Classification on the hand written digits data set (mnist)

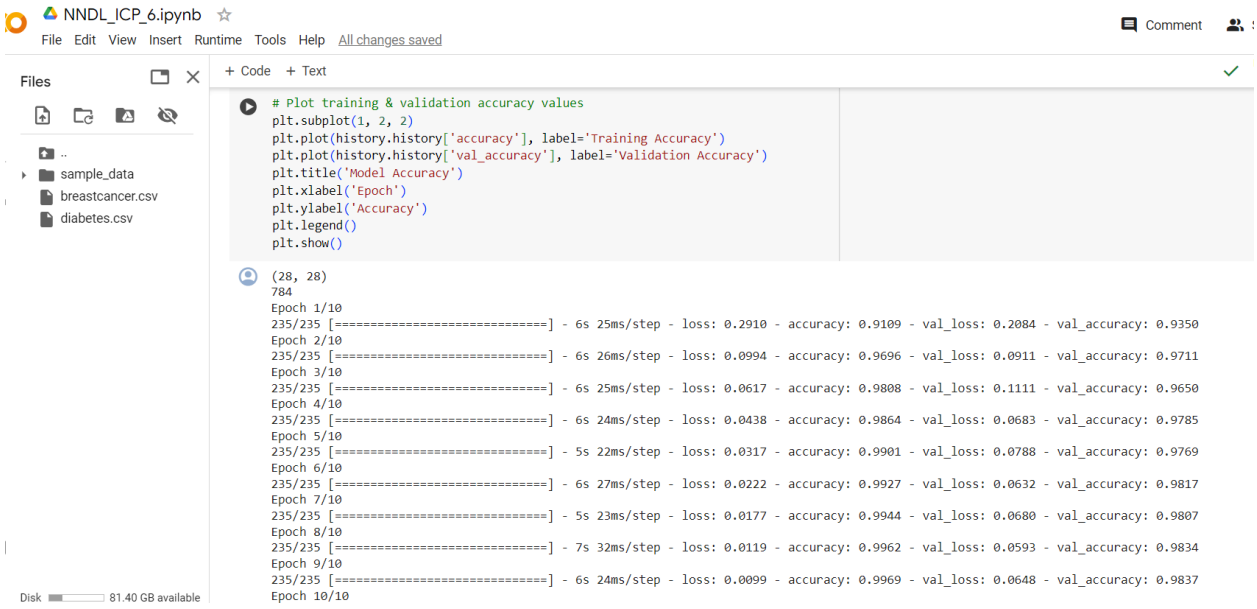


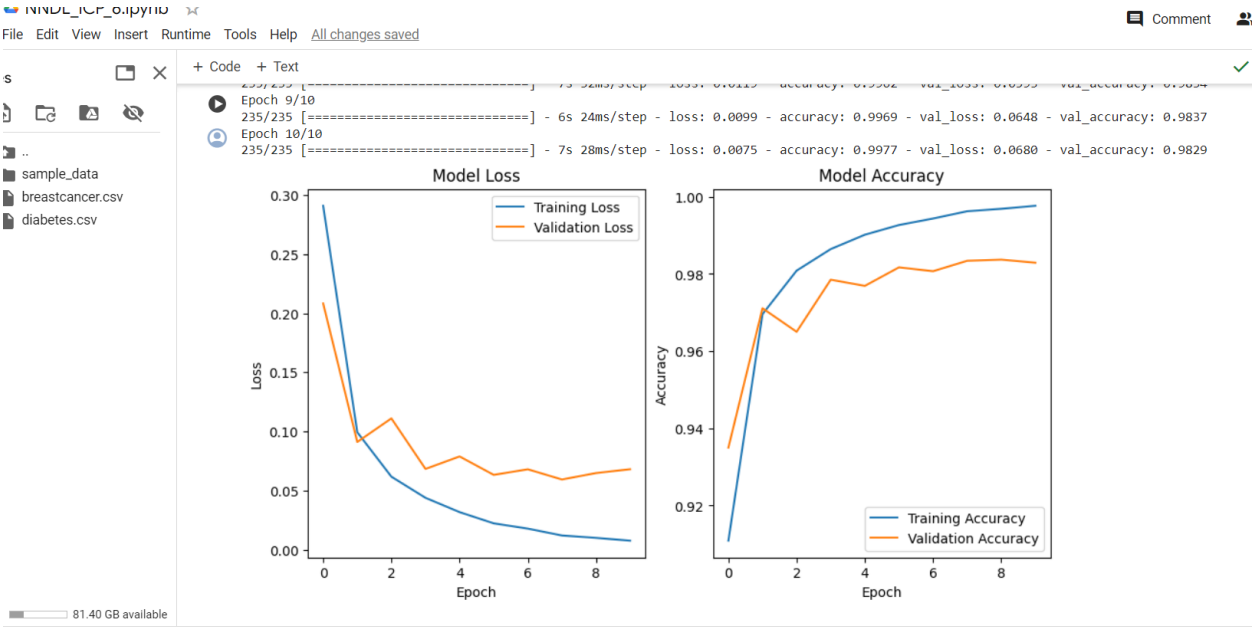




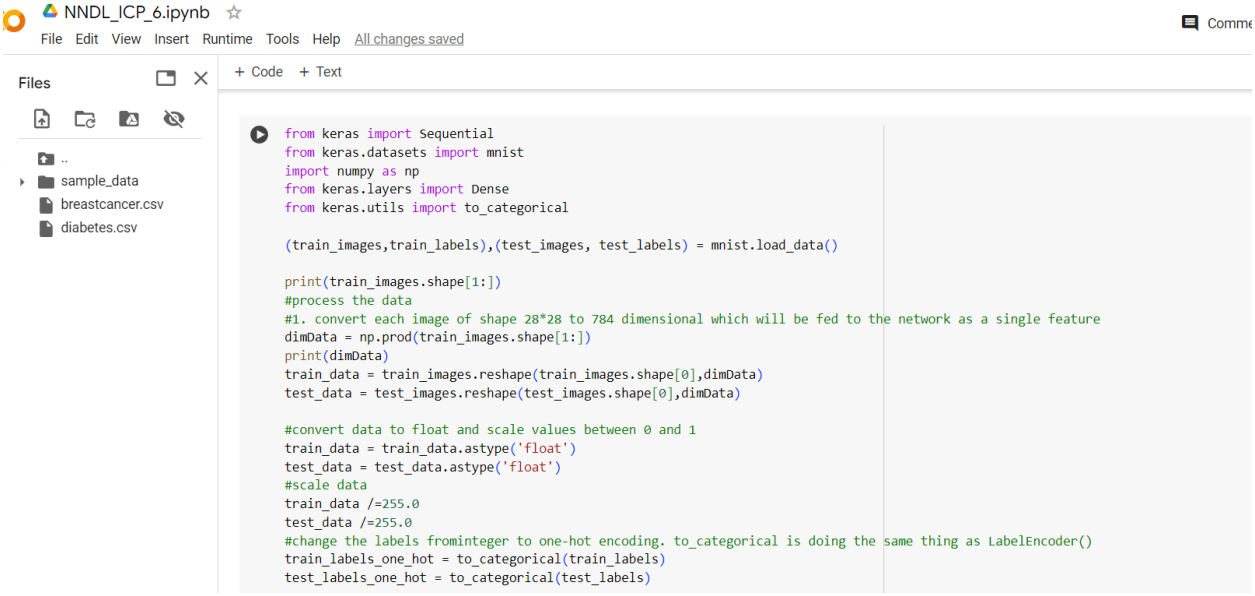
1. Plot the loss and accuracy for both training data and validation data using the history object in the source code.







2. 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.



NNDL_ICP_6.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Sl

Files

..
sample_data
breastcancer.csv
diabetes.csv

Disk 81.40 GB available

+ Code + Text

```
#creating 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'])
history = model.fit(train_data, train_labels_one_hot, batch_size=256, epochs=10, verbose=1,
                    validation_data=(test_data, test_labels_one_hot))

import matplotlib.pyplot as plt

# Plot one image from the test data
plt.figure()
plt.imshow(test_images[0], cmap='gray')
plt.title(f"True Label: {test_labels[0]}")

# Perform inference on the single image
image = test_data[0].reshape(1, dimData)
prediction = model.predict(image)
predicted_label = np.argmax(prediction)

print(f"Predicted Label: {predicted_label}")
```

```
(28, 28)
784
Epoch 1/10
235/235 [==============================] - 7s 26ms/step - loss: 0.2913 - accuracy: 0.9108 - val_loss: 0.1687 - val_accuracy: 0.9458
Epoch 2/10
```

NNDL_ICP_6.ipynb

File Edit View Insert Runtime Tools Help All changes saved

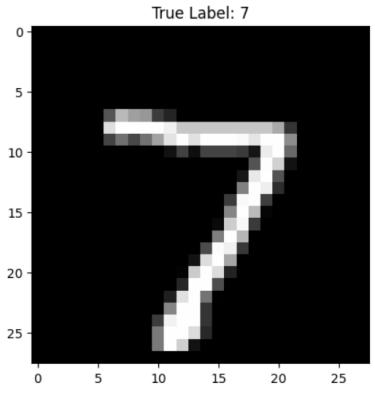
Comment Share

+ Code + Text

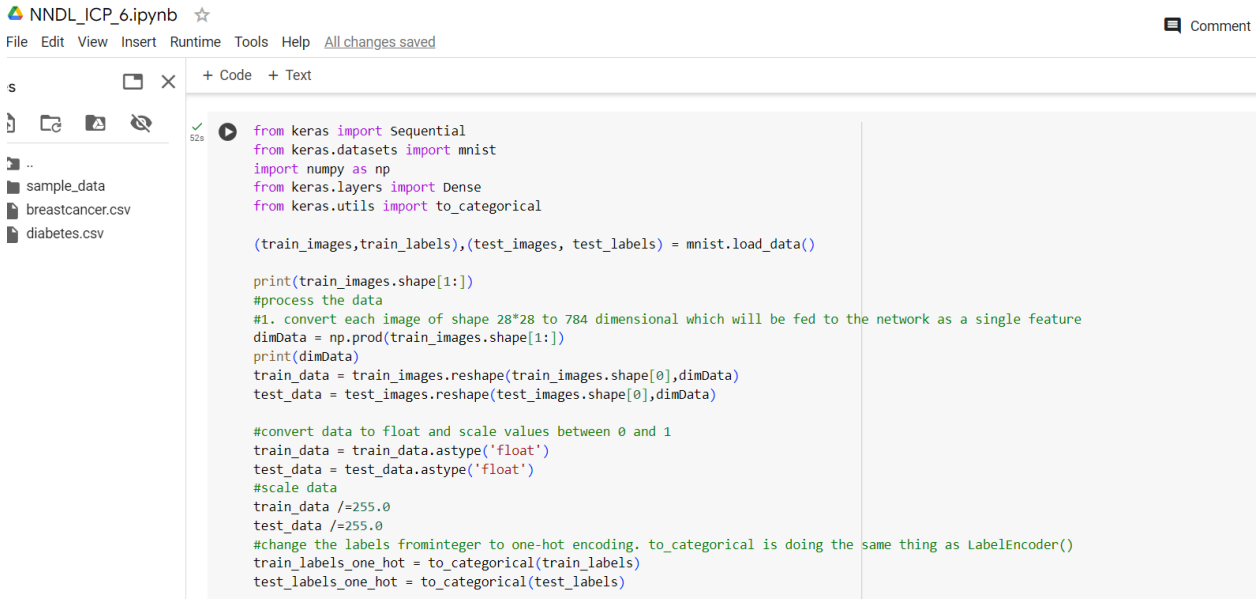
sample_data
breastcancer.csv
diabetes.csv

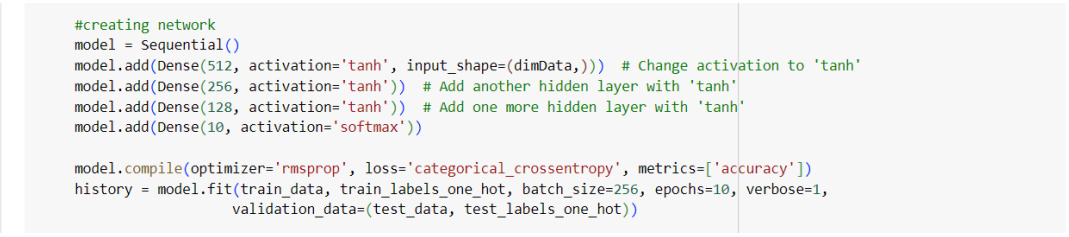
81.40 GB available

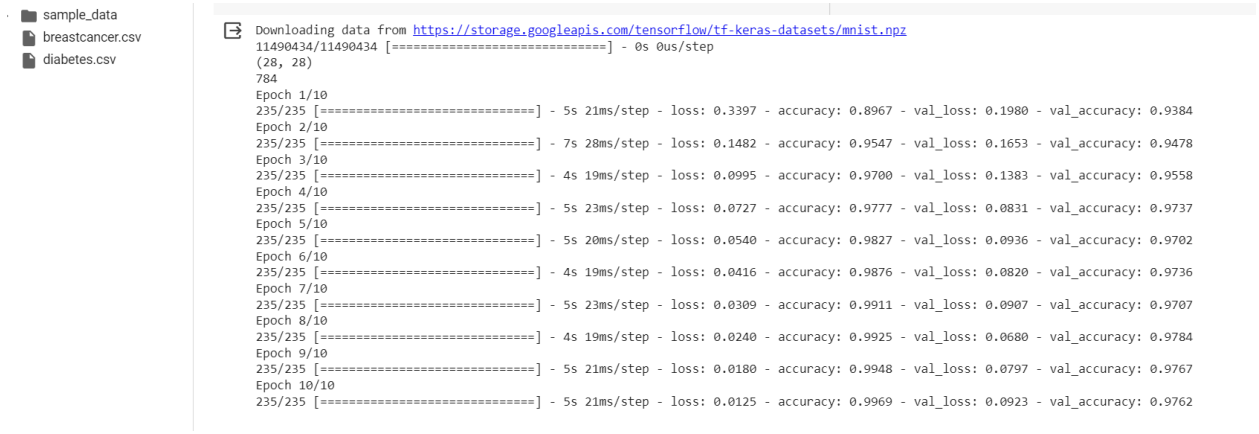
```
Epoch 9/10
235/235 [==============================] - 5s 23ms/step - loss: 0.0100 - accuracy: 0.9968 - val_loss: 0.0679 - val_accuracy: 0.9833
Epoch 10/10
235/235 [==============================] - 6s 27ms/step - loss: 0.0073 - accuracy: 0.9977 - val_loss: 0.0656 - val_accuracy: 0.9820
1/1 [==============================] - 0s 90ms/step
Predicted Label: 7
```



3. 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.







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