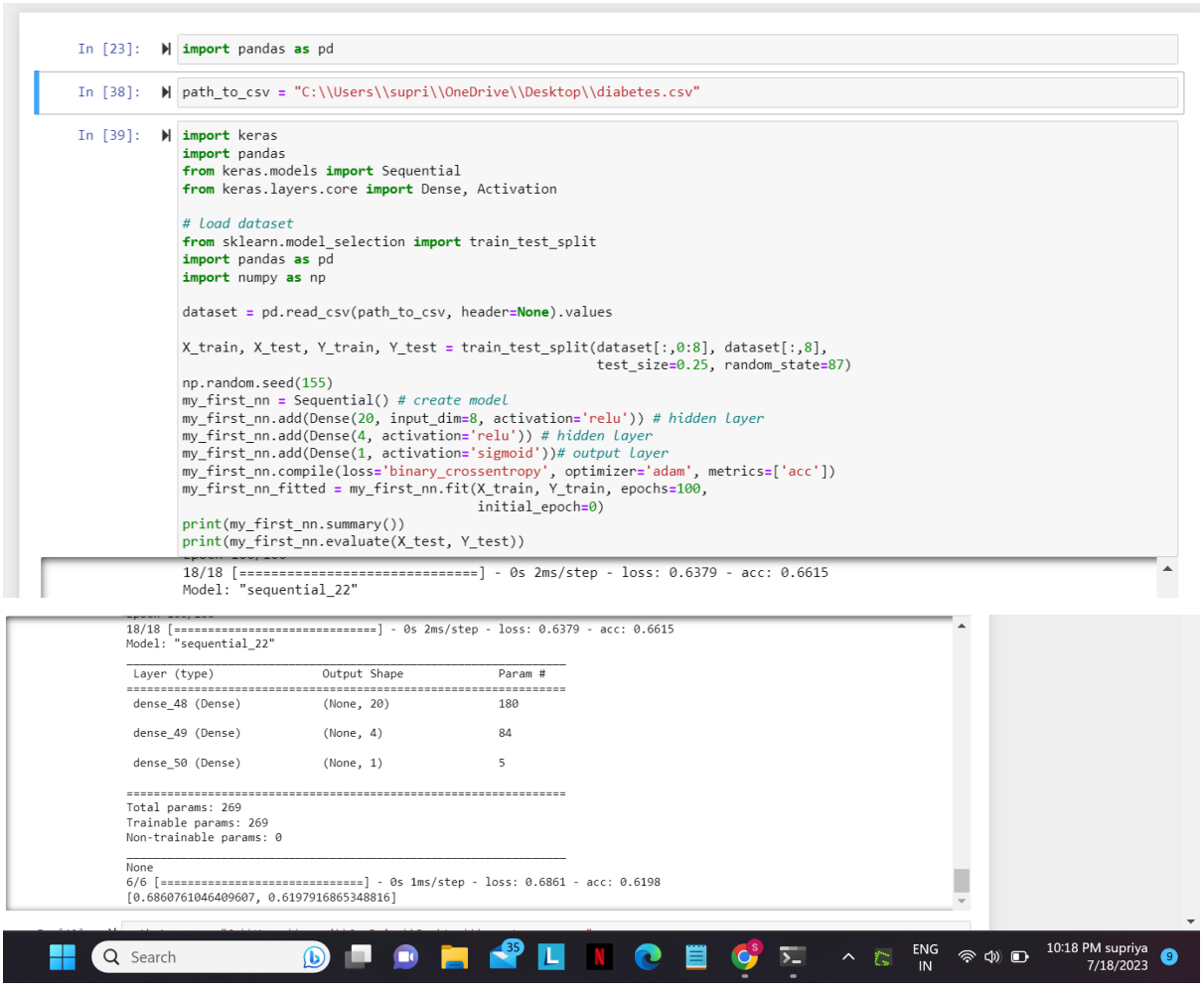
NEURAL NETWORKS ASSIGNMENT 2

NAME: SUPRIYA SAMA
700744510

Video Link:
https://drive.google.com/file/d/14-b-SH2YtRGorSQQNwjcAvfzbAuFC5sK/view?usp=sharing

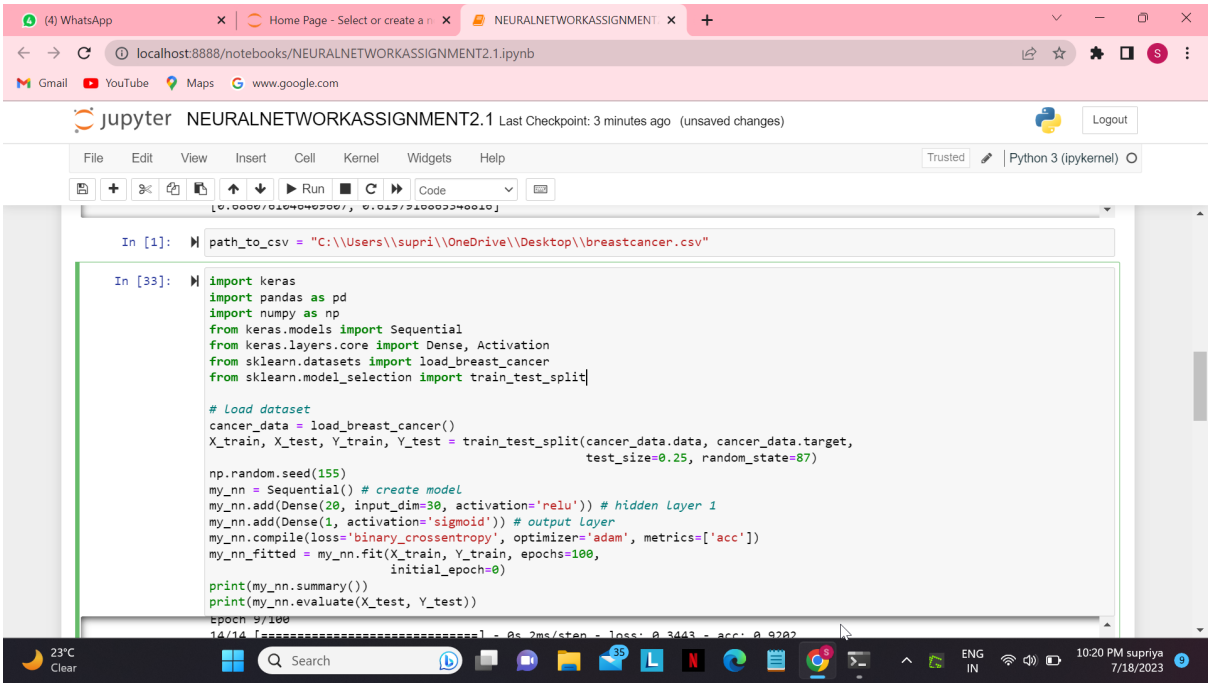
1. **Use the use case in the class:**
   Add more Dense layers to the existing code and check how the accuracy changes.

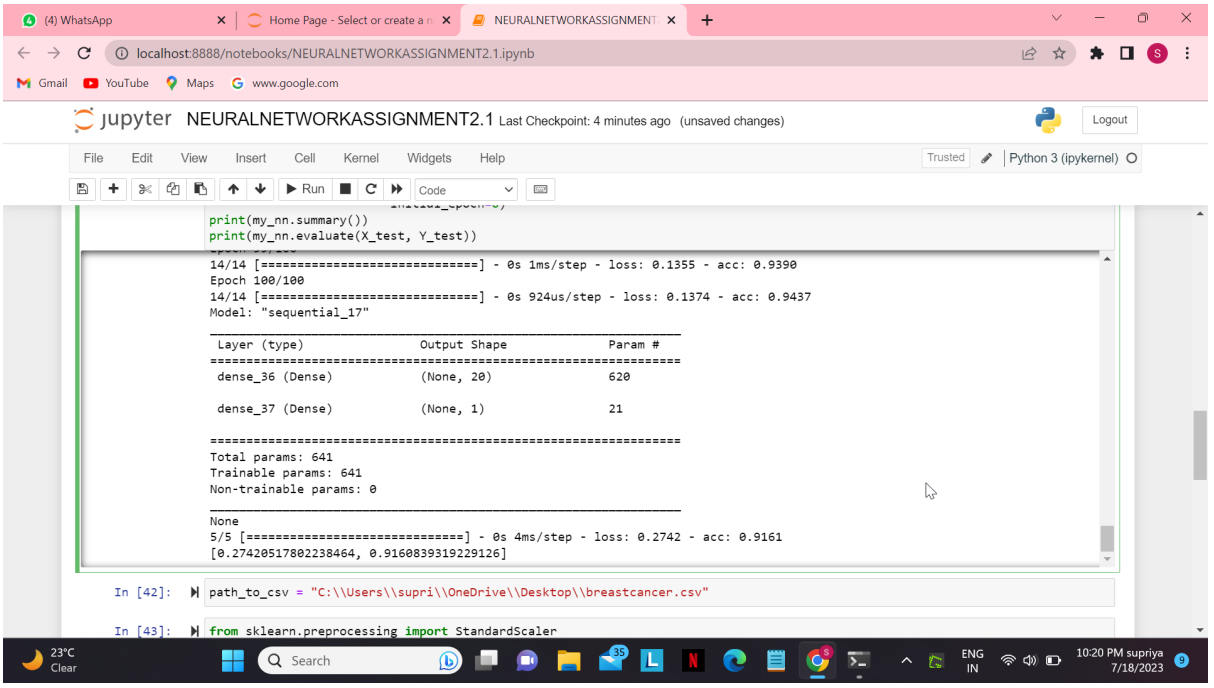
   Here I have added one more dense layer for the code and we got the accuracy of 0.6615, whereas with one layer we got the accuracy of 0.6671.
   Accuracy is more than there is only one Dense layer.



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

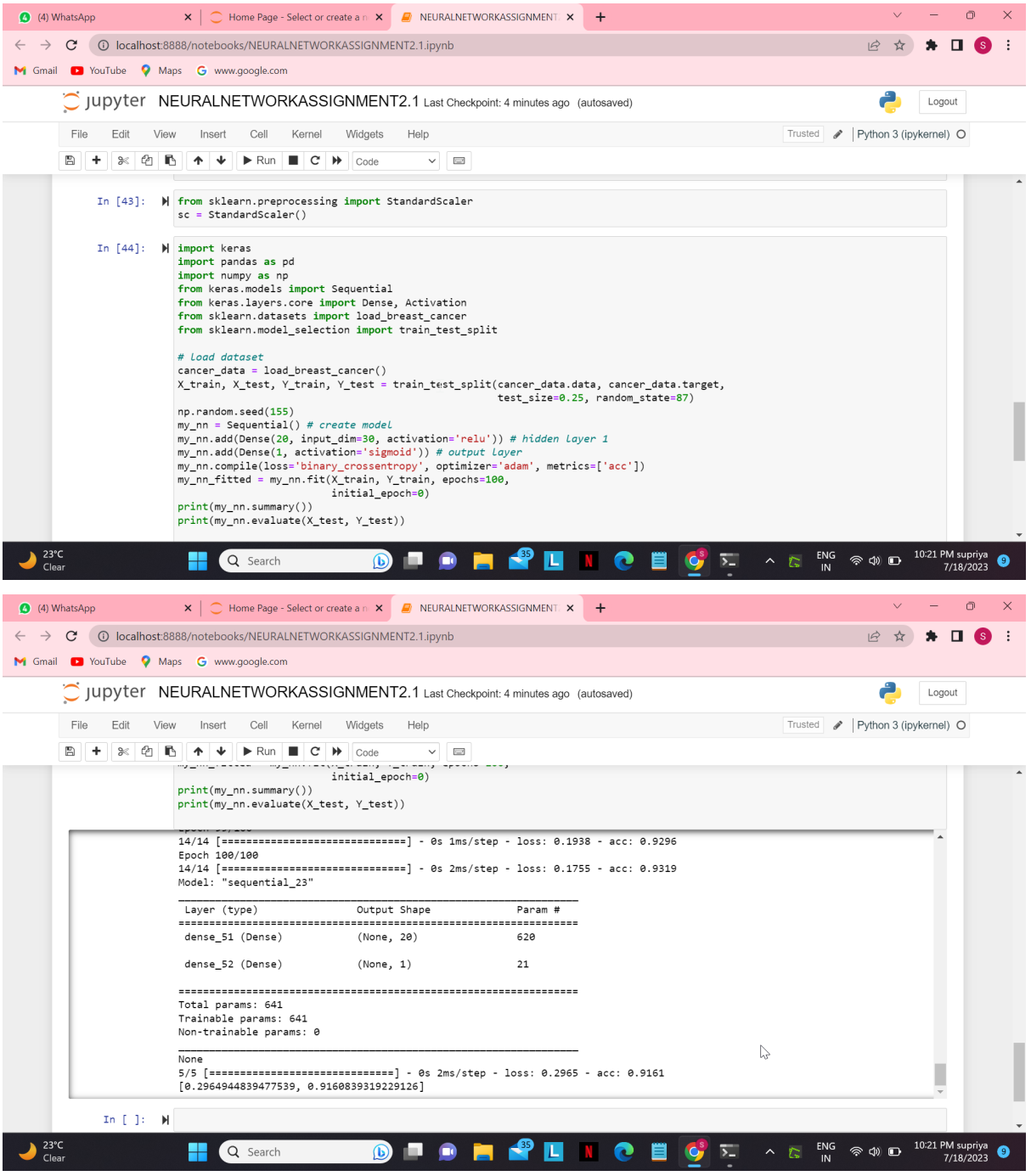
Here first I loaded the dataset then I created the model.





We can see that the accuracy is 0.9161 and the loss is 0.2742.

3. **Normalize the data before feeding the data to the model and check how the normalization change your accuracy.**



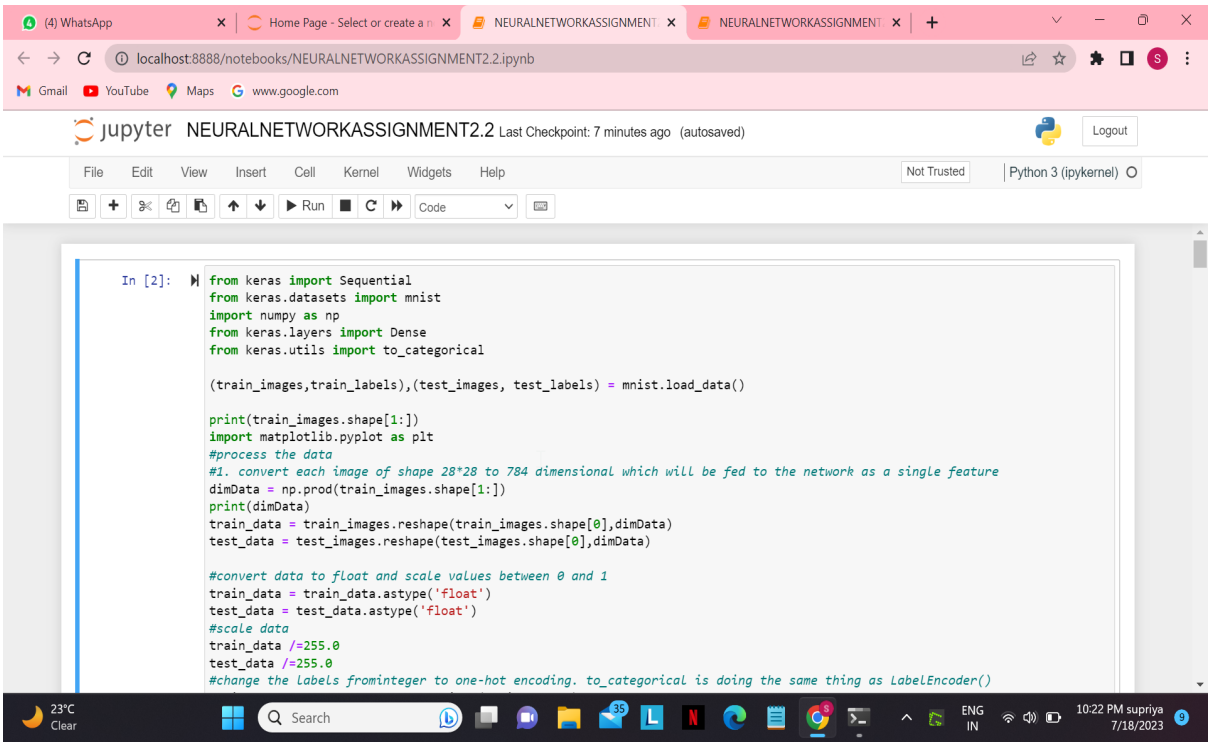
```
In [43]: from sklearn.preprocessing import StandardScaler
         sc = StandardScaler()

In [44]: import keras
         import pandas as pd
         import numpy as np
         from keras.models import Sequential
         from keras.layers.core import Dense, Activation
         from sklearn.datasets import load_breast_cancer
         from sklearn.model_selection import train_test_split

         # Load dataset
         cancer_data = load_breast_cancer()
         X_train, X_test, Y_train, Y_test = train_test_split(cancer_data.data, cancer_data.target,
                                                             test_size=0.25, random_state=87)
         np.random.seed(155)
         my_nn = Sequential() # create model
         my_nn.add(Dense(20, input_dim=30, activation='relu')) # hidden layer 1
         my_nn.add(Dense(1, activation='sigmoid')) # output layer
         my_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
         my_nn_fitted = my_nn.fit(X_train, Y_train, epochs=100,
                                  initial_epoch=0)
         print(my_nn.summary())
         print(my_nn.evaluate(X_test, Y_test))
```

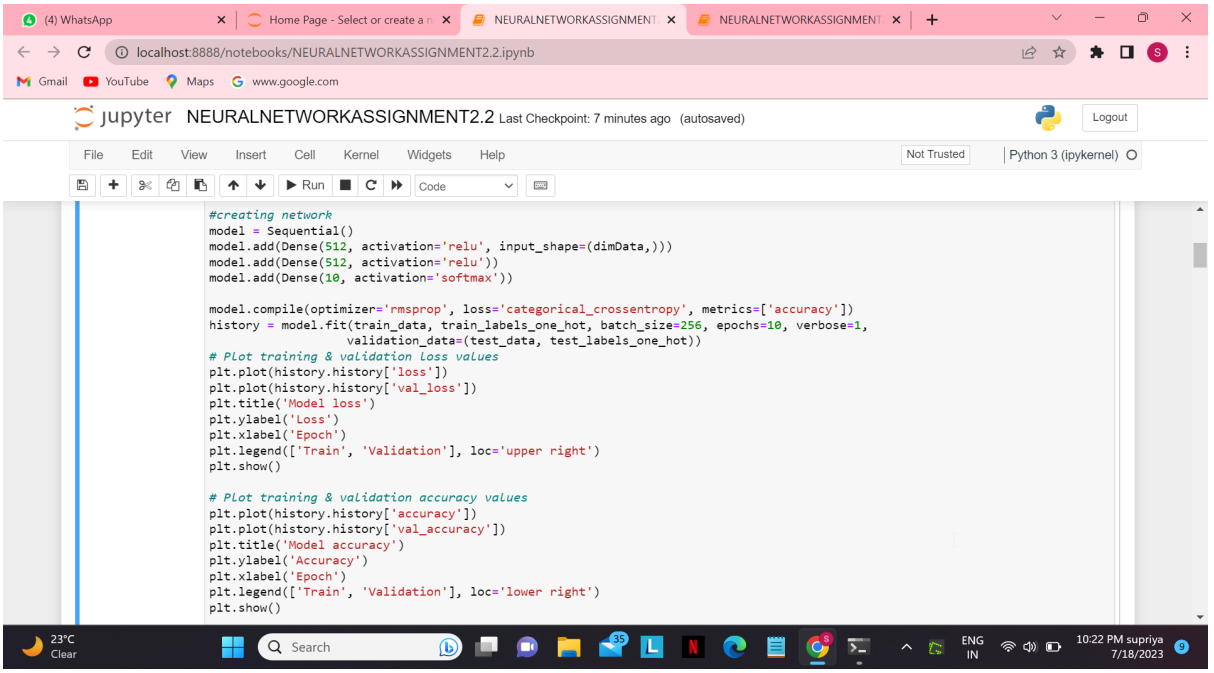
```
14/14 [==============================] - 0s 1ms/step - loss: 0.1938 - acc: 0.9296
Epoch 100/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1755 - acc: 0.9319
Model: "sequential_23"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_51 (Dense)            (None, 20)                620

 dense_52 (Dense)            (None, 1)                 21

=================================================================
Total params: 641
Trainable params: 641
Non-trainable params: 0
_________________________________________________________________
None
5/5 [==============================] - 0s 2ms/step - loss: 0.2965 - acc: 0.9161
[0.2964944839477539, 0.9160839319229126]
```

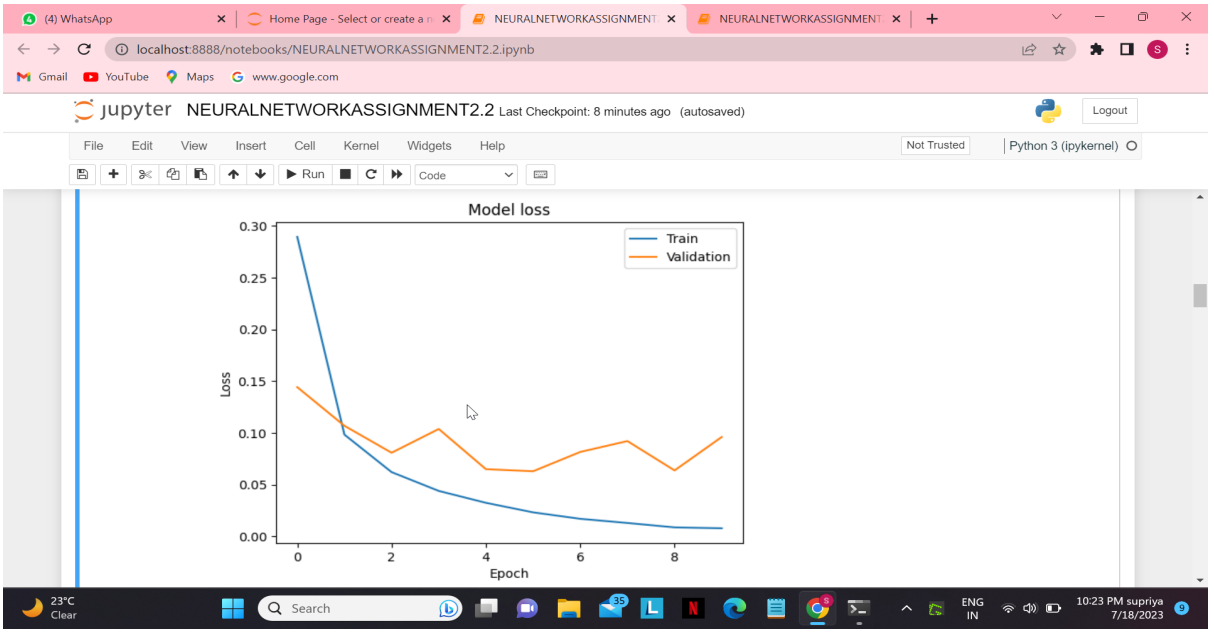
We got a loss of 0.2965 and accuracy of 0.9161.

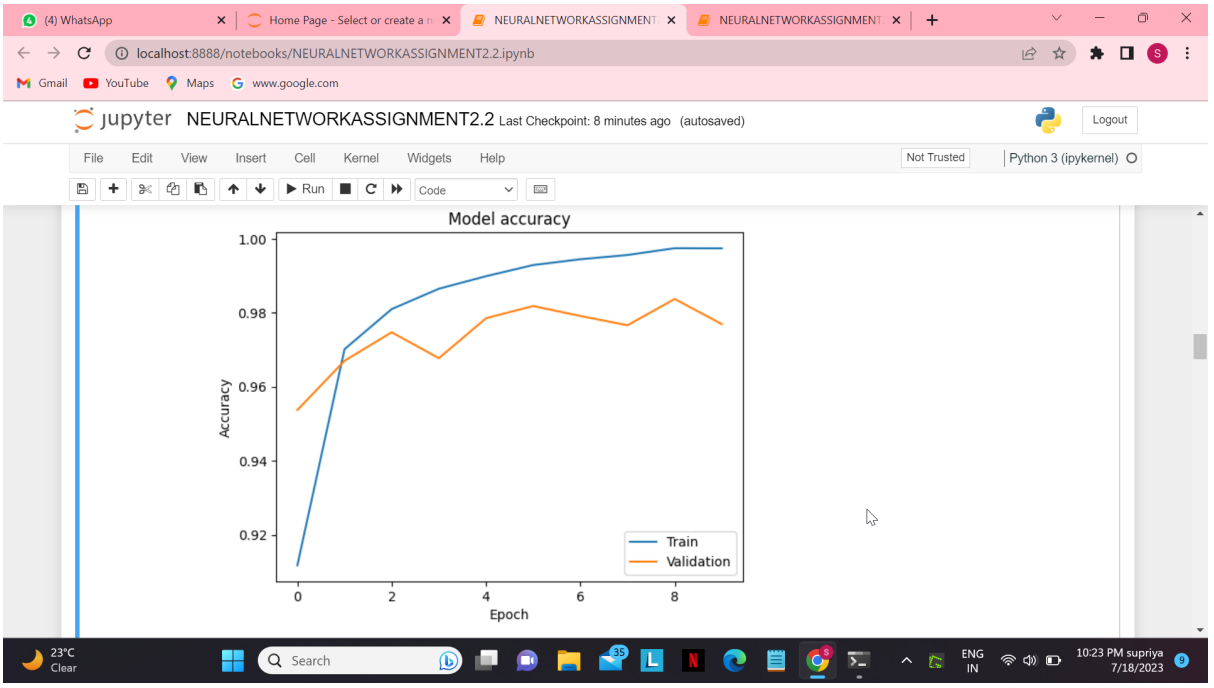
Use Image Classification on the hand written digits data set (mnist)

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


(4) WhatsApp
Home Page - Select or create a n
NEURALNETWORKASSIGNMENT
NEURALNETWORKASSIGNMENT
localhost:8888/notebooks/NEURALNETWORKASSIGNMENT2.2.ipynb
Gmail
YouTube
Maps
www.google.com
jupyter NEURALNETWORKASSIGNMENT2.2 Last Checkpoint: 7 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [2]:
from keras import Sequential
from keras.datasets import mnist
import numpy as np
from keras.layers import Dense
from keras.utils import to_categorical
(train_images,train_labels),(test_images, test_labels) = mnist.load_data()
print(train_images.shape[1:])
import matplotlib.pyplot as plt
#process the data
#1. convert each image of shape 28*28 to 784 dimensional which will be fed to the network as a single feature
dimData = np.prod(train_images.shape[1:])
print(dimData)
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('float')
test_data = test_data.astype('float')
#scale data
train_data /=255.0
test_data /=255.0
#change the labels frominteger to one-hot encoding. to_categorical is doing the same thing as LabelEncoder()
23°C Clear
Search
ENG IN
10:22 PM supriya 7/18/2023


(4) WhatsApp
Home Page - Select or create a n
NEURALNETWORKASSIGNMENT
NEURALNETWORKASSIGNMENT
localhost:8888/notebooks/NEURALNETWORKASSIGNMENT2.2.ipynb
Gmail
YouTube
Maps
www.google.com
jupyter NEURALNETWORKASSIGNMENT2.2 Last Checkpoint: 7 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
#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))
# Plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper right')
plt.show()
# Plot training & validation accuracy values
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='lower right')
plt.show()
23°C Clear
Search
ENG IN
10:22 PM supriya 7/18/2023

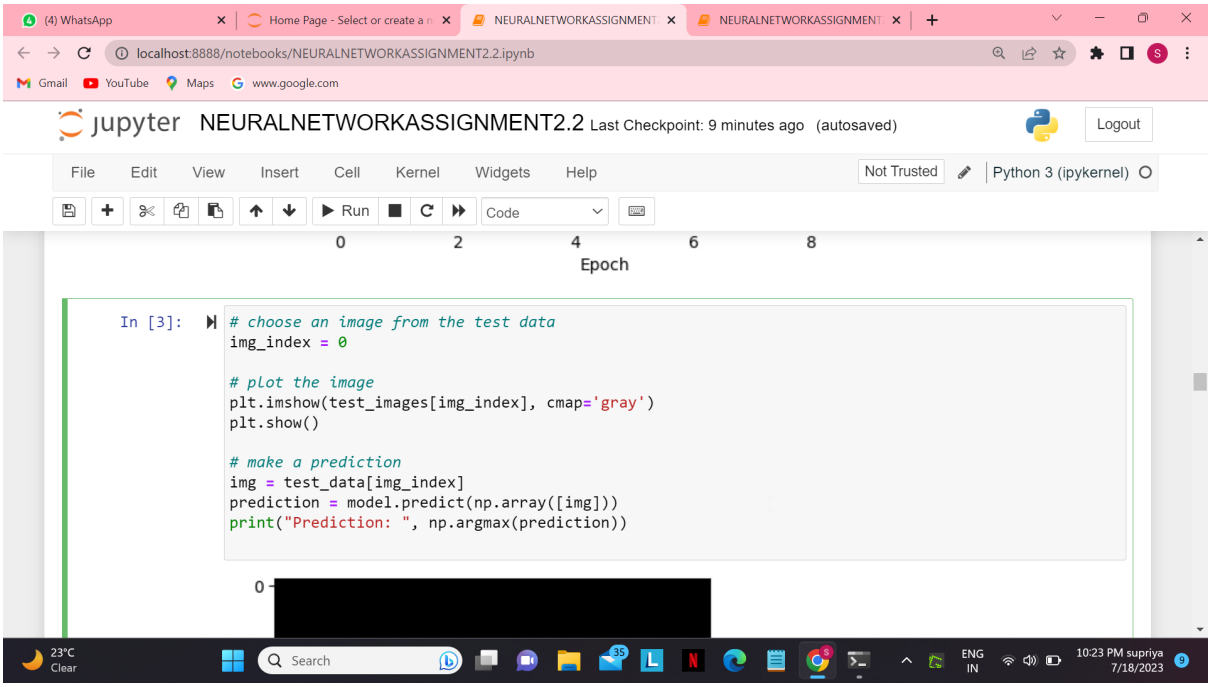


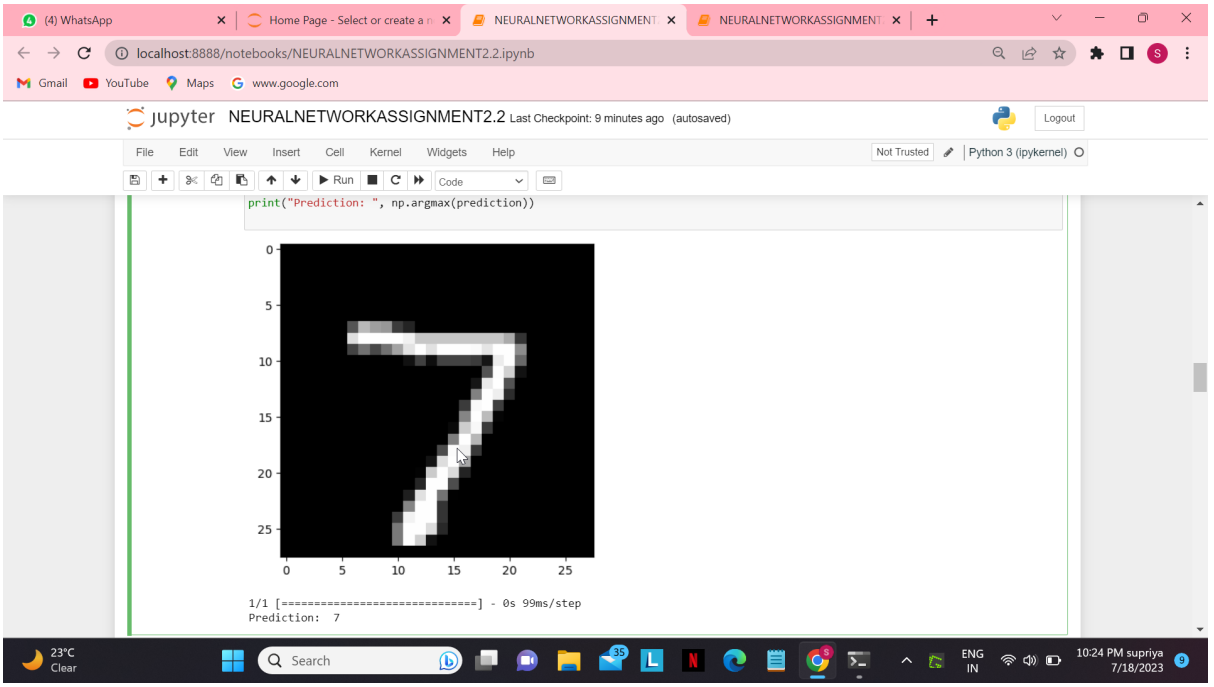


Here we can see the plot of the loss and accuracy for both training data and validation data.

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.

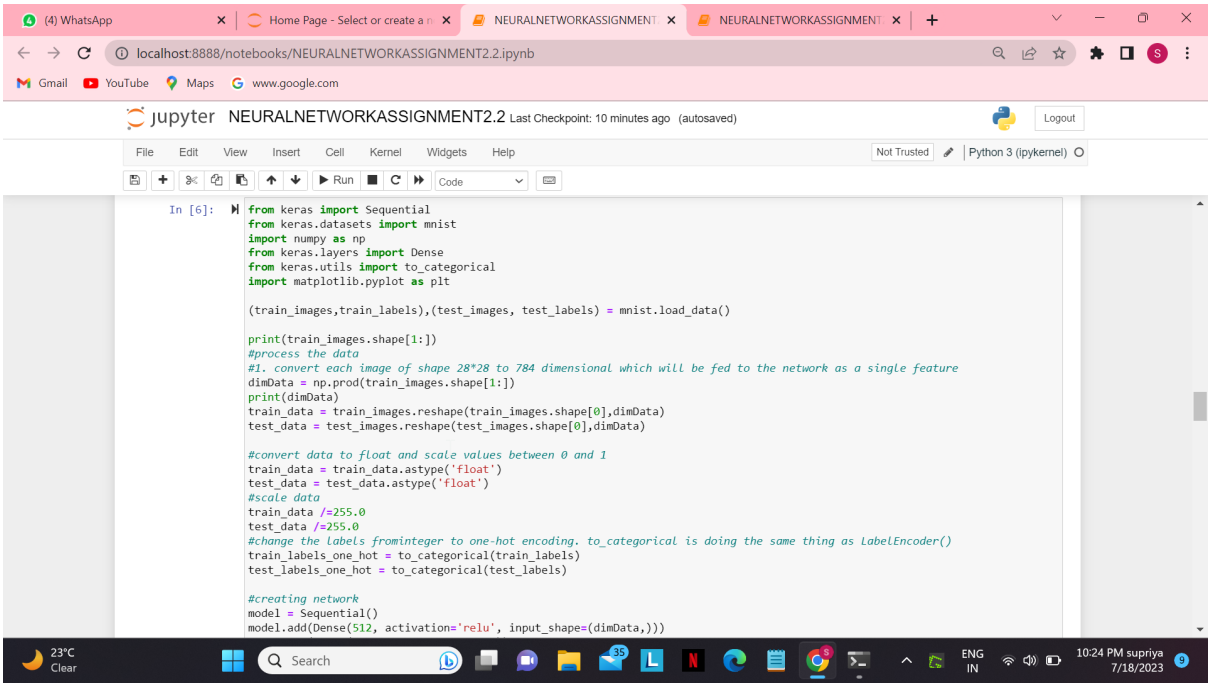
Here first we have to choose the image from the test data, then plot the image.

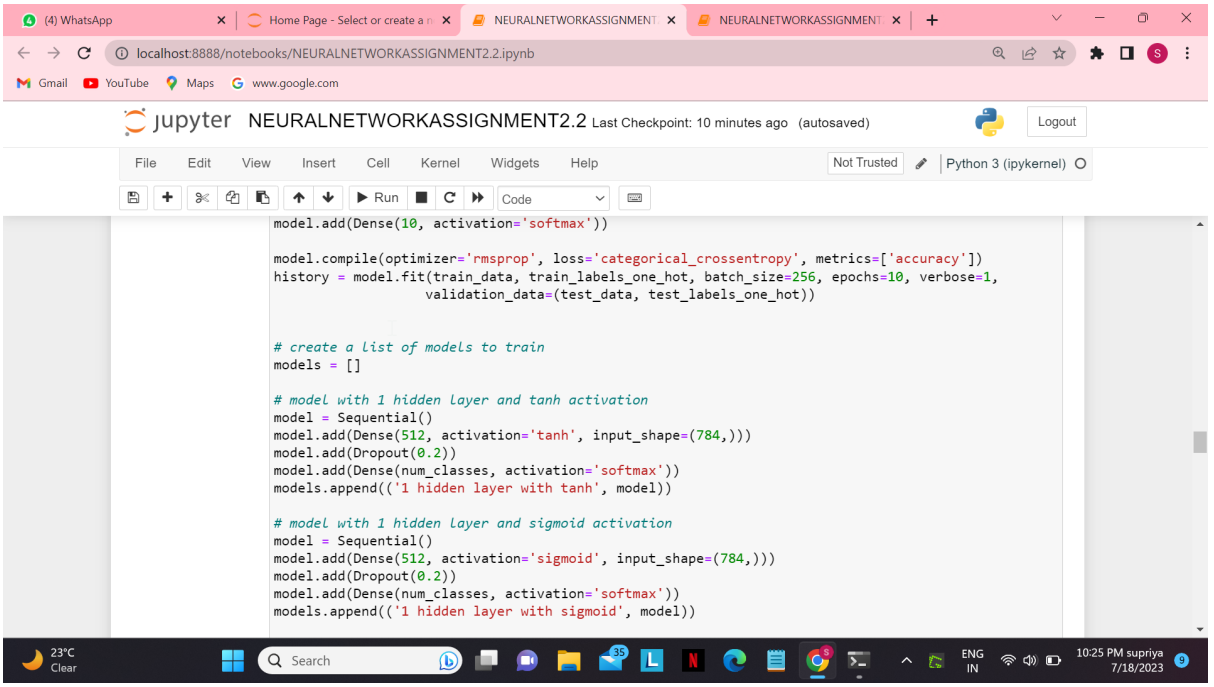




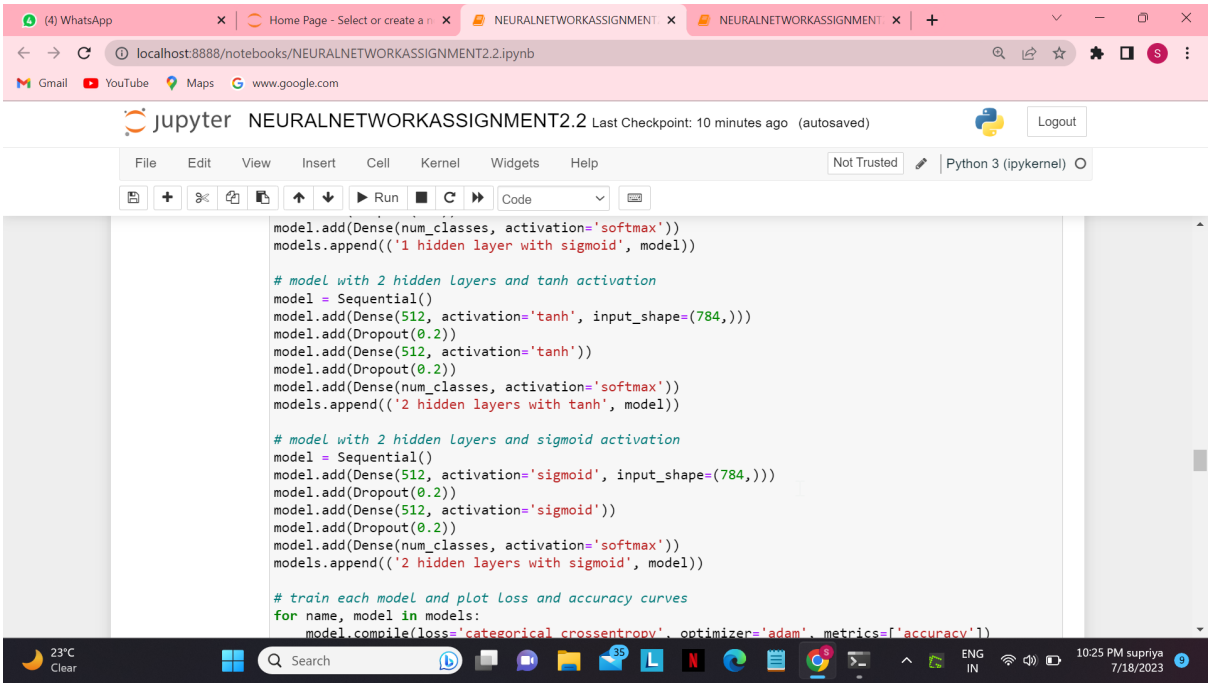
We got the prediction 7.

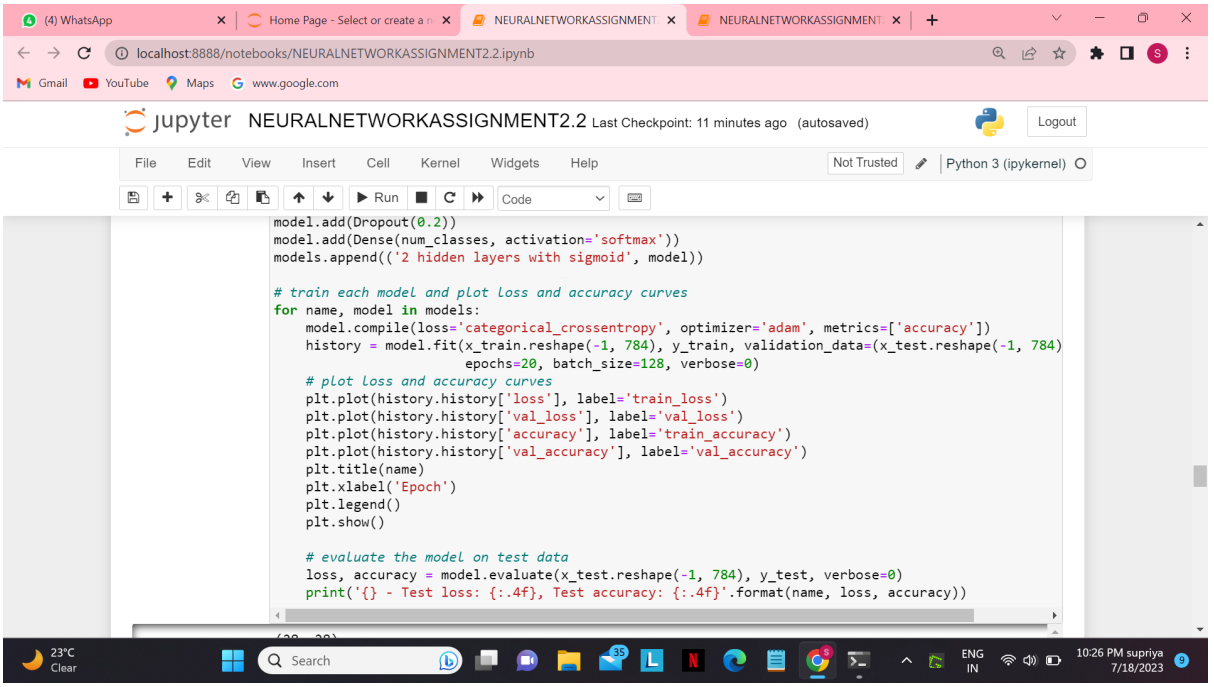
3. We had used 2 hidden layers and Relu activation. Try to change the number of hidden layer and the activation to tanh or sigmoid and see what happens.

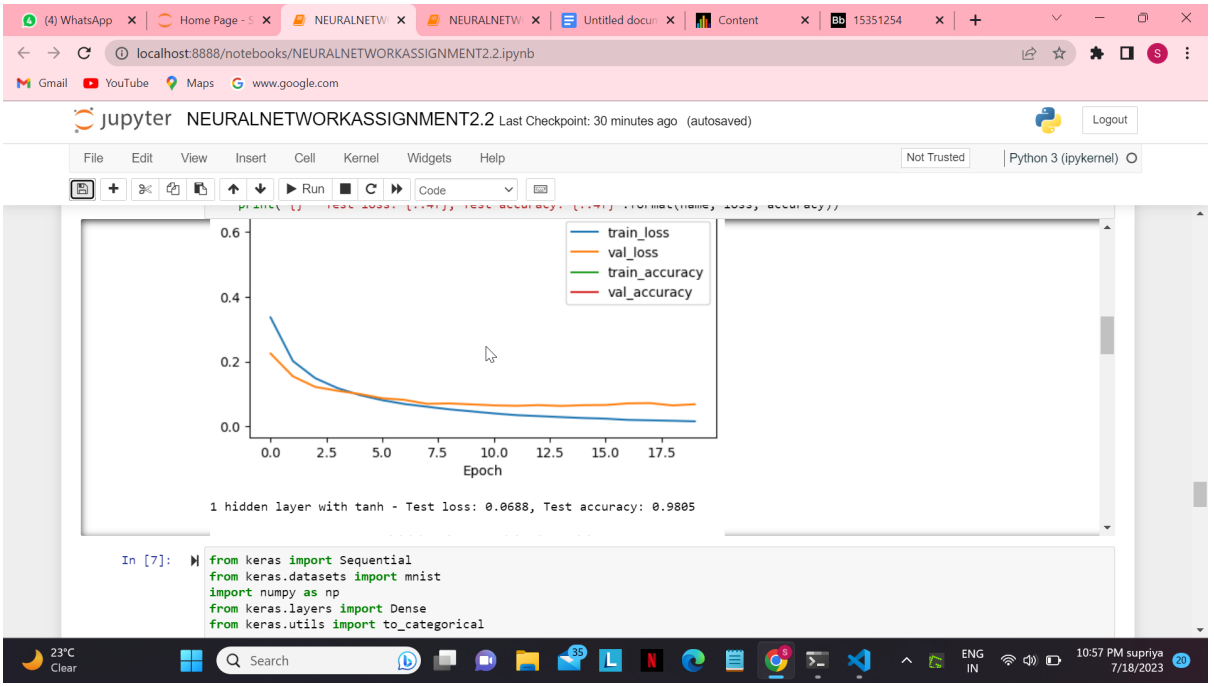




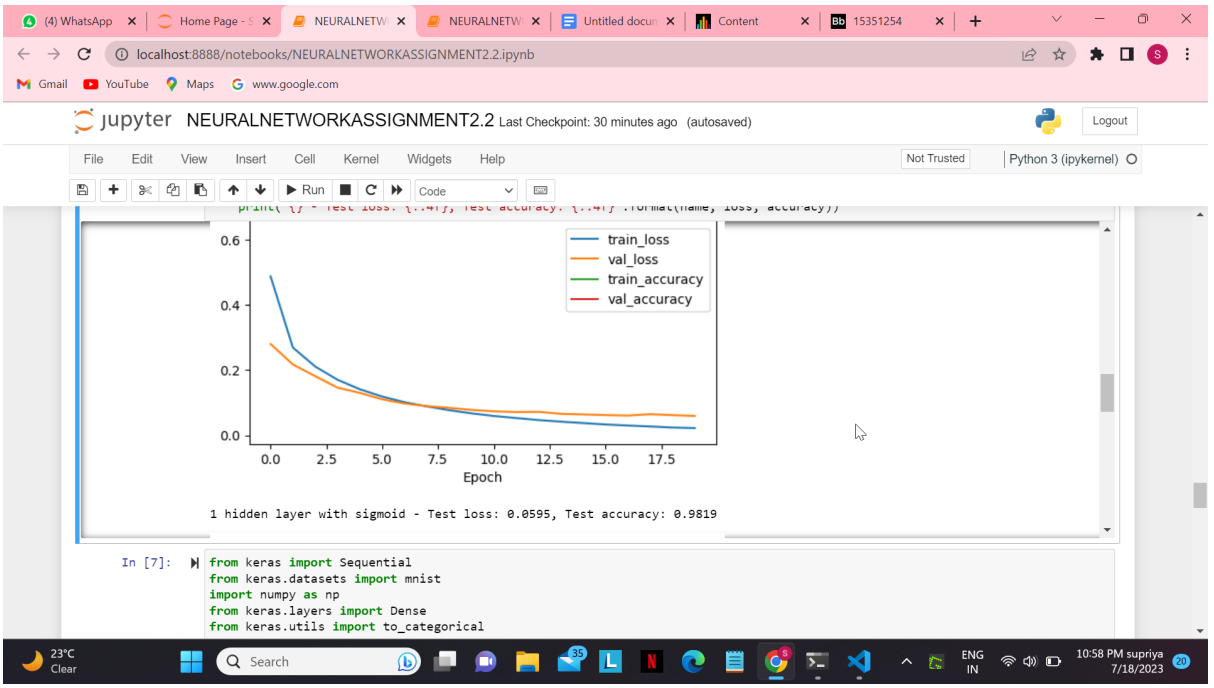
Here we used **tanh** and **sigmoid** activations.


(4) WhatsApp
Home Page - Select or create a n
NEURALNETWORKASSIGNMENT
NEURALNETWORKASSIGNMENT
localhost:8888/notebooks/NEURALNETWORKASSIGNMENT2.2.ipynb
Gmail
YouTube
Maps
www.google.com
jupyter NEURALNETWORKASSIGNMENT2.2 Last Checkpoint: 10 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
model.add(Dense(num_classes, activation='softmax'))
models.append(('1 hidden layer with sigmoid', model))
# model with 2 hidden layers and tanh activation
model = Sequential()
model.add(Dense(512, activation='tanh', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('2 hidden layers with tanh', model))
# model with 2 hidden layers and sigmoid activation
model = Sequential()
model.add(Dense(512, activation='sigmoid', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='sigmoid'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('2 hidden layers with sigmoid', model))
# train each model and plot loss and accuracy curves
for name, model in models:
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
23°C
Clear
Search
ENG
IN
10:25 PM supriya
7/18/2023

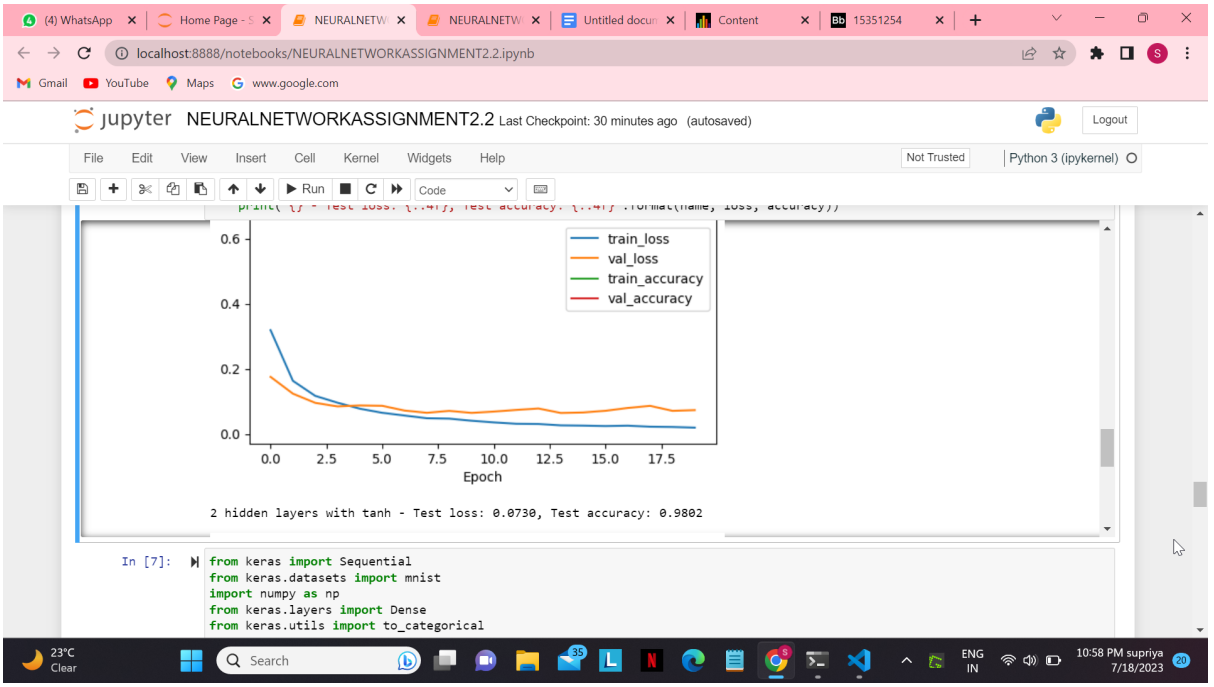

(4) WhatsApp
Home Page - Select or create a n
NEURALNETWORKASSIGNMENT
NEURALNETWORKASSIGNMENT
localhost:8888/notebooks/NEURALNETWORKASSIGNMENT2.2.ipynb
Gmail
YouTube
Maps
www.google.com
jupyter NEURALNETWORKASSIGNMENT2.2 Last Checkpoint: 11 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('2 hidden layers with sigmoid', model))
# train each model and plot loss and accuracy curves
for name, model in models:
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
history = model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.reshape(-1, 784)
epochs=20, batch_size=128, verbose=0)
# plot loss and accuracy curves
plt.plot(history.history['loss'], label='train_loss')
plt.plot(history.history['val_loss'], label='val_loss')
plt.plot(history.history['accuracy'], label='train_accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.title(name)
plt.xlabel('Epoch')
plt.legend()
plt.show()
# evaluate the model on test data
loss, accuracy = model.evaluate(x_test.reshape(-1, 784), y_test, verbose=0)
print('{} - Test loss: {:.4f}, Test accuracy: {:.4f}'.format(name, loss, accuracy))
23°C
Clear
Search
ENG
IN
10:26 PM supriya
7/18/2023



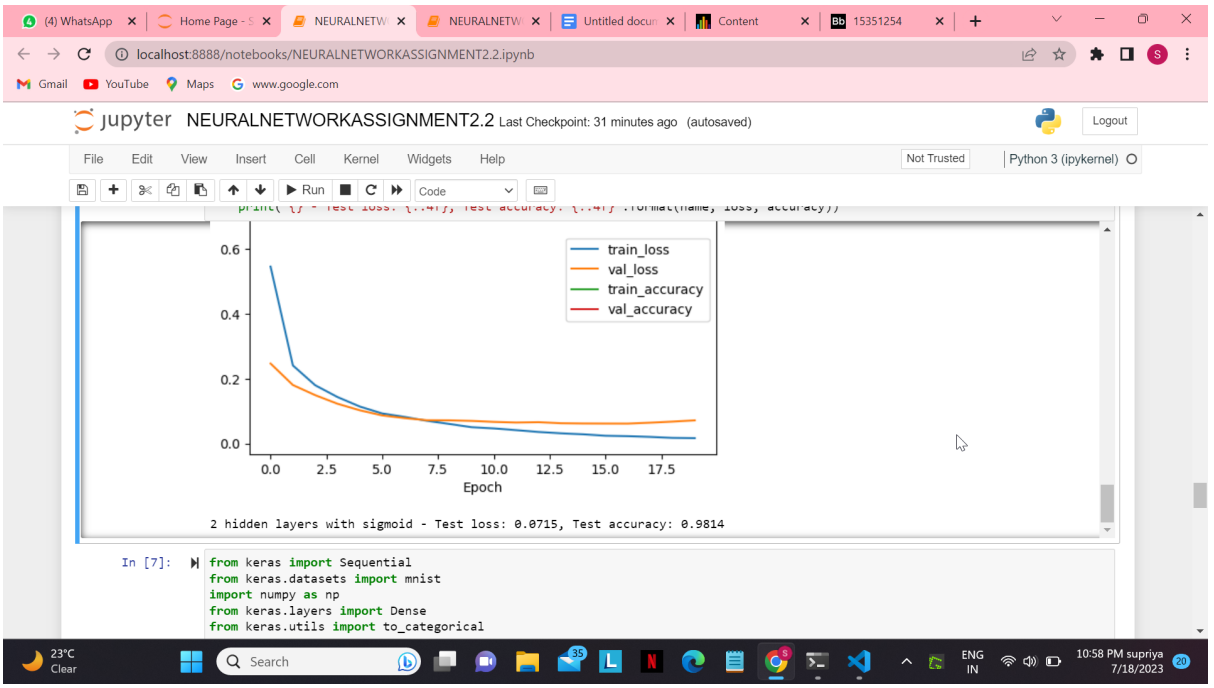
We can see the loss and accuracy for 1 hidden layer with tanh.



We can see the loss and accuracy for 1 hidden layer with sigmoid.



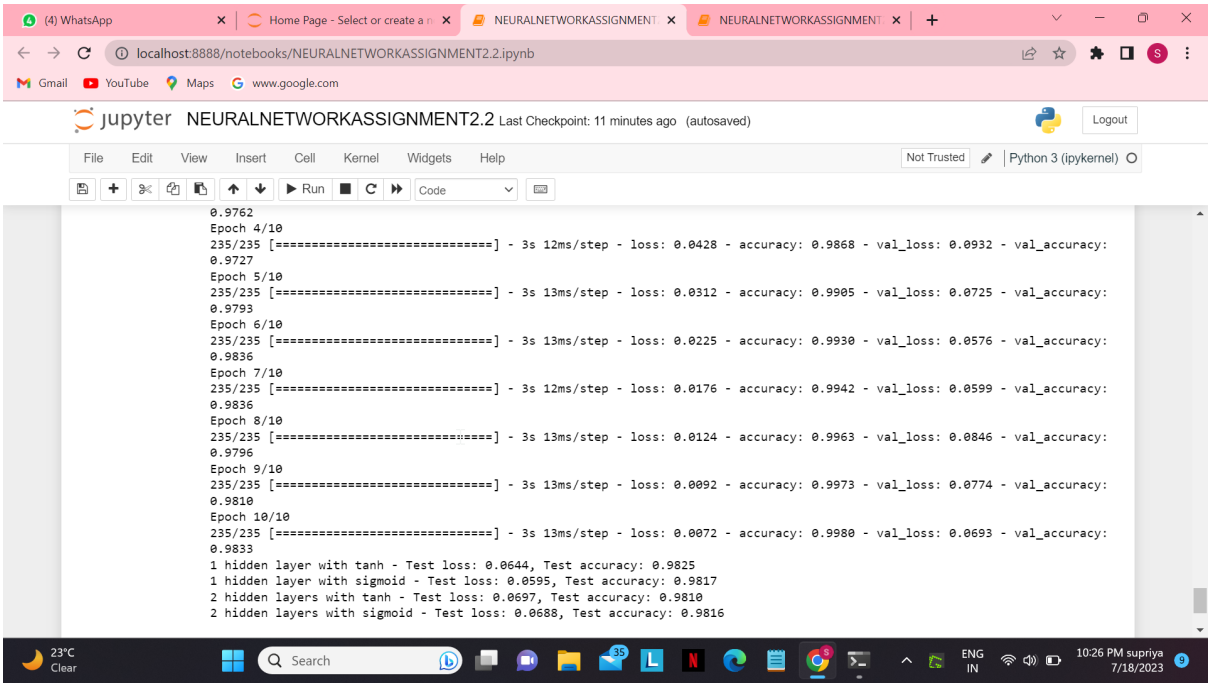
We can see the loss and accuracy for 2 hidden layer with tanh.



We can see the loss and accuracy for 2 hidden layer with sigmoid.

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

I run the same code without scaling the images. Here we can see the output, which is different from the earlier one.



We can see both the loss and accuracy has changed.