NNDL - Assignment - 6 ICP_Basics in Keras

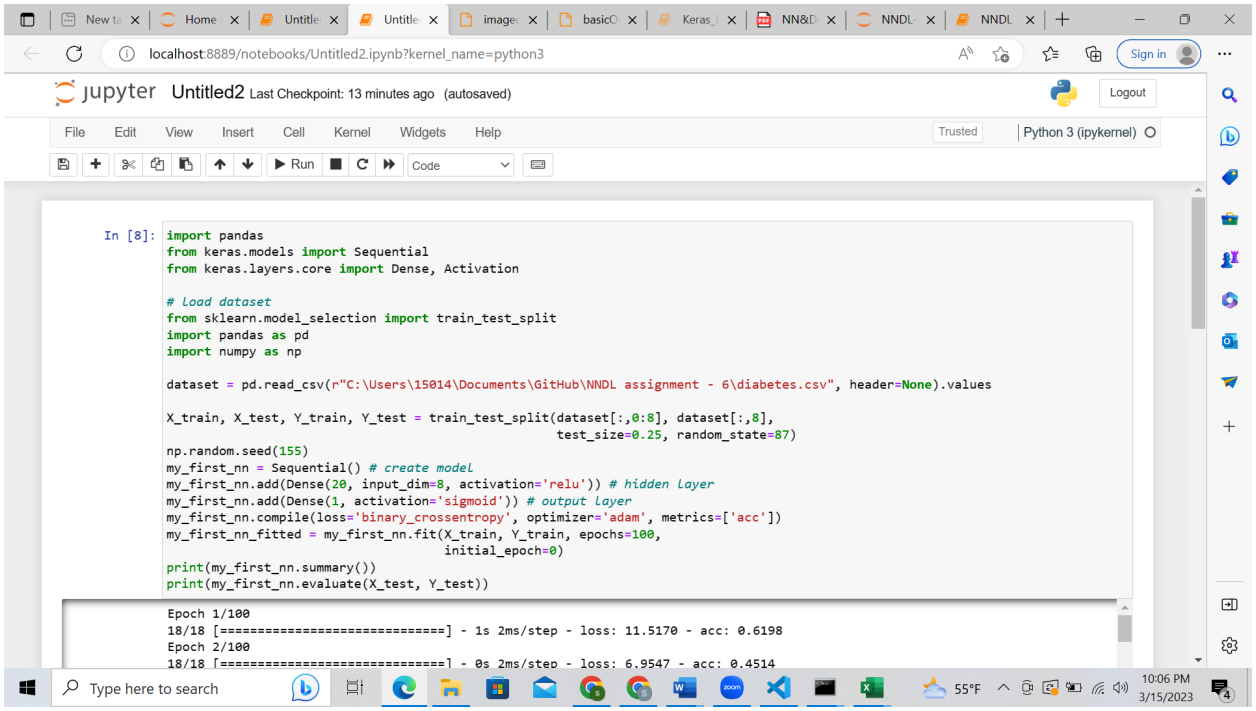
Use Case Description:
Predicting the diabetes disease

Programming elements:
Keras Basics

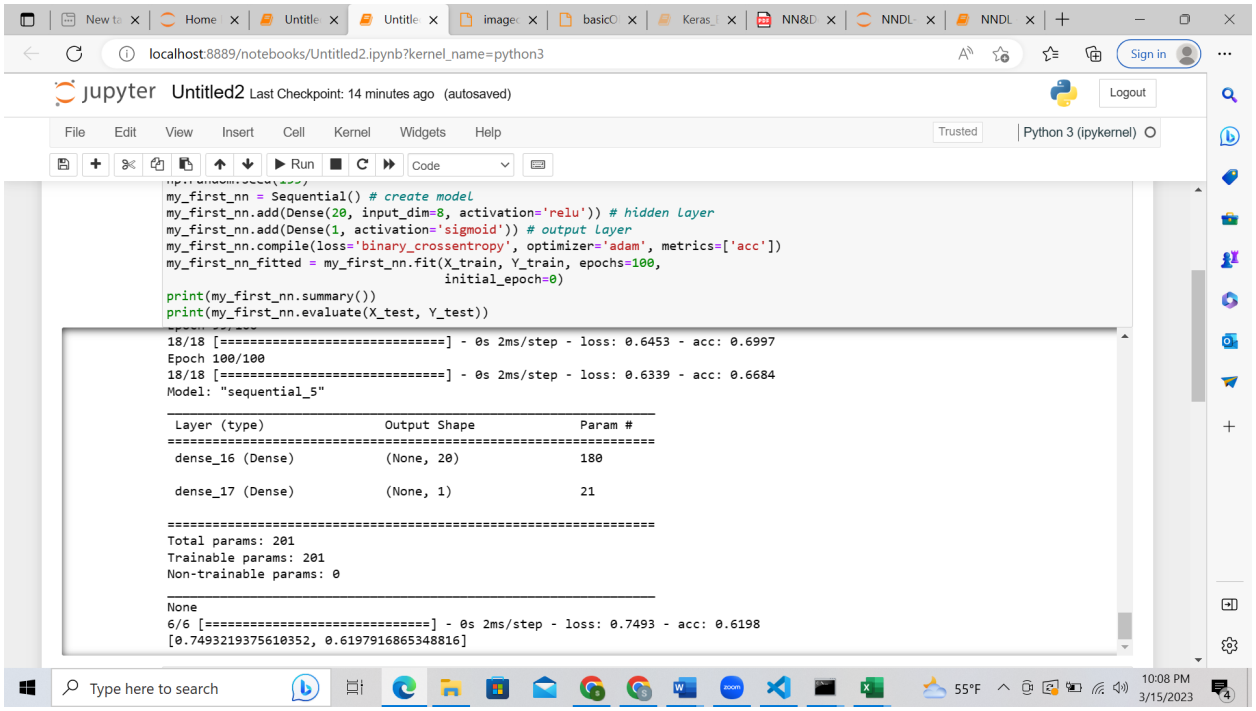
In class programming:

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

Ans
Code before adding the dense layers is -



Accuracy of the model before adding the code is -



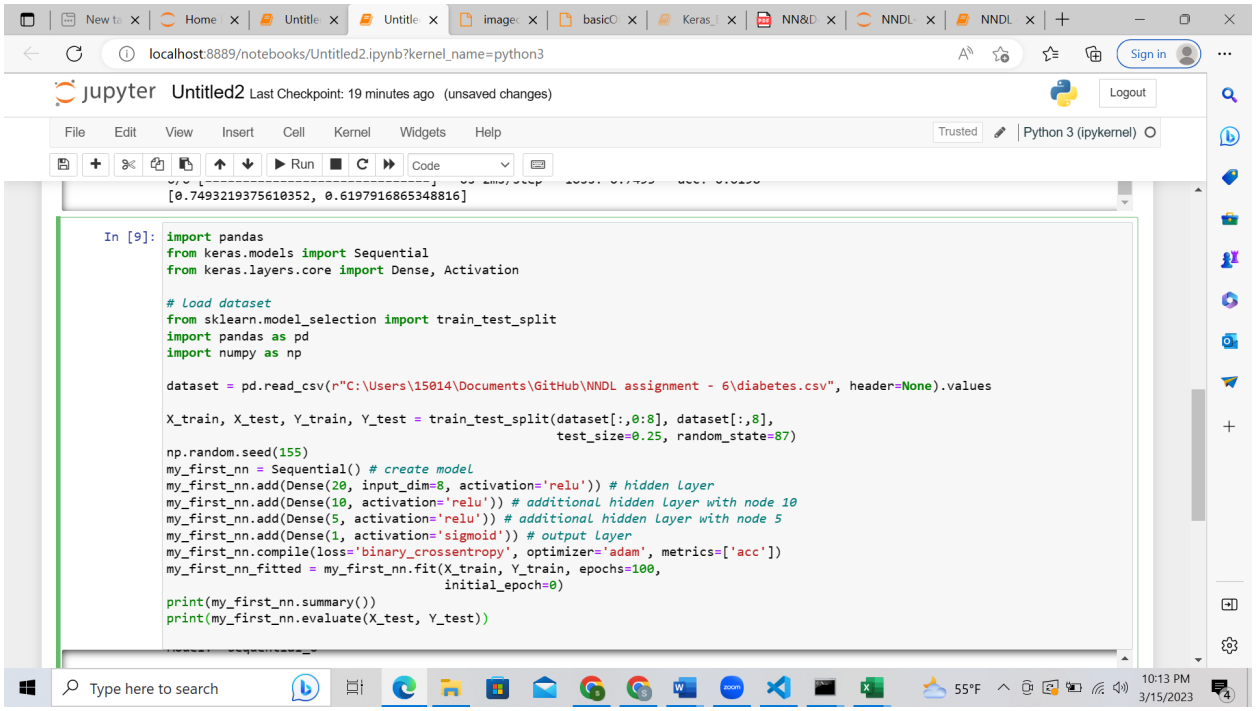
```
my_first_nn = Sequential() # create model
my_first_nn.add(Dense(20, input_dim=8, activation='relu')) # hidden layer
my_first_nn.add(Dense(1, activation='sigmoid')) # output layer
my_first_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
my_first_nn_fitted = my_first_nn.fit(X_train, Y_train, epochs=100,
                                     initial_epoch=0)
print(my_first_nn.summary())
print(my_first_nn.evaluate(X_test, Y_test))
```

```
18/18 [==============================] - 0s 2ms/step - loss: 0.6453 - acc: 0.6997
Epoch 100/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6339 - acc: 0.6684
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_16 (Dense)            (None, 20)                180

 dense_17 (Dense)            (None, 1)                 21

=================================================================
Total params: 201
Trainable params: 201
Non-trainable params: 0
_________________________________________________________________
None
6/6 [==============================] - 0s 2ms/step - loss: 0.7493 - acc: 0.6198
[0.7493219375610352, 0.6197916865348816]
```

Adding more dense layers to the code -

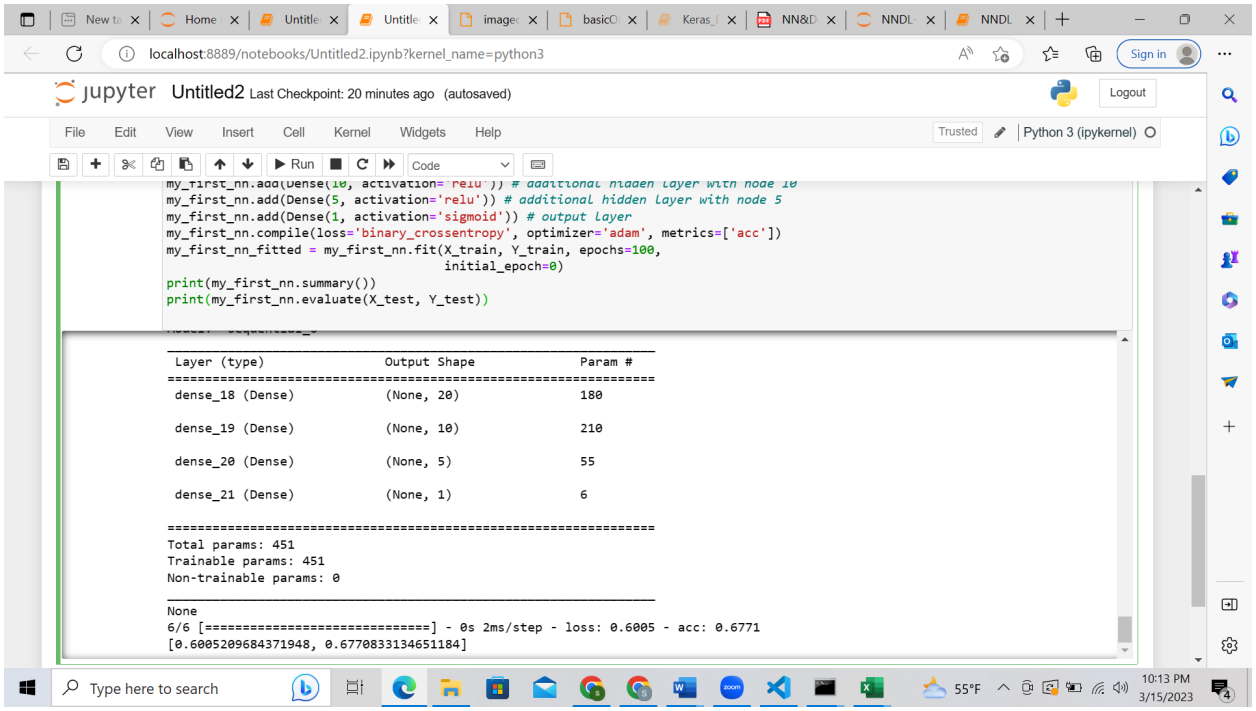
1) We can easily add more dense layers to the existing code , we can simply add more dense layers to the sequential model.
2) Here I have added two additional dense layers with 10 and 5 nodes.

Code after adding the dense layers is -



```
In [9]: import pandas
from keras.models import Sequential
from keras.layers.core import Dense, Activation

# Load dataset
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np

dataset = pd.read_csv(r"C:\Users\15014\Documents\GitHub\NNDL assignment - 6\diabetes.csv", header=None).values

X_train, X_test, Y_train, Y_test = train_test_split(dataset[:,0:8], dataset[:,8],
                                                    test_size=0.25, random_state=87)
np.random.seed(155)
my_first_nn = Sequential() # create model
my_first_nn.add(Dense(20, input_dim=8, activation='relu')) # hidden layer
my_first_nn.add(Dense(10, activation='relu')) # additional hidden layer with node 10
my_first_nn.add(Dense(5, activation='relu')) # additional hidden layer with node 5
my_first_nn.add(Dense(1, activation='sigmoid')) # output layer
my_first_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
my_first_nn_fitted = my_first_nn.fit(X_train, Y_train, epochs=100,
                                     initial_epoch=0)
print(my_first_nn.summary())
print(my_first_nn.evaluate(X_test, Y_test))
```

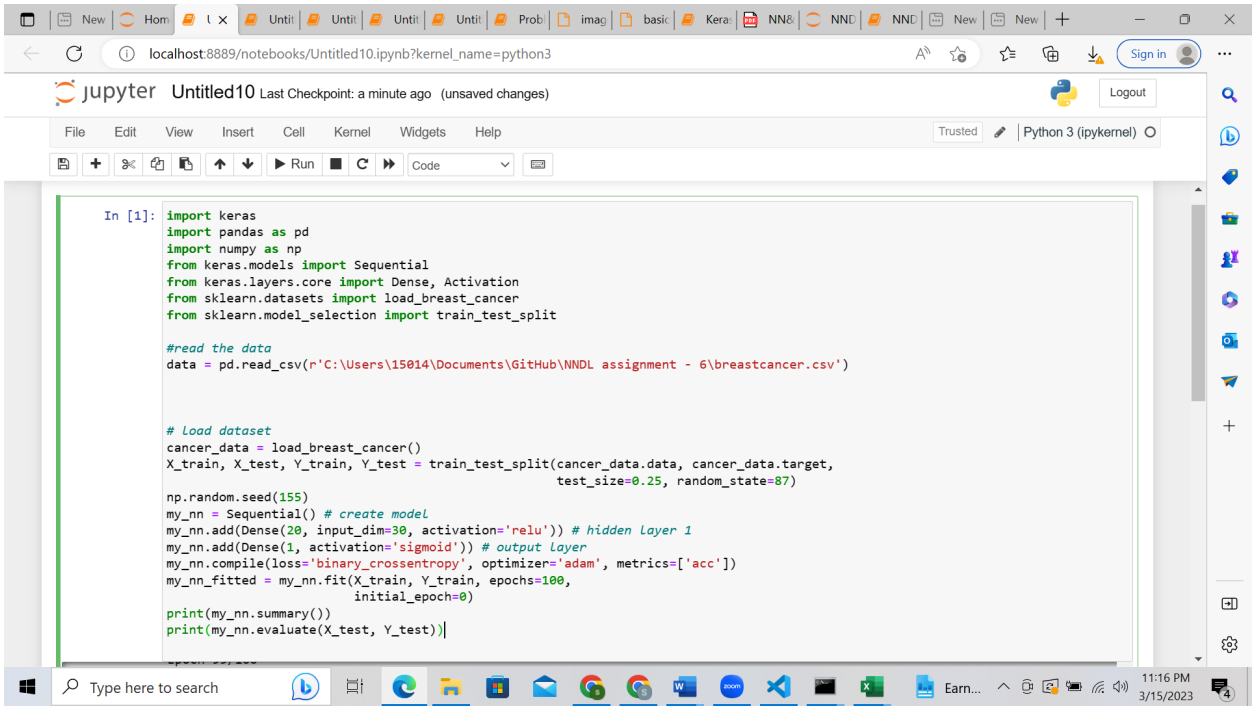
Accuracy of the model after adding the additional dense layers is



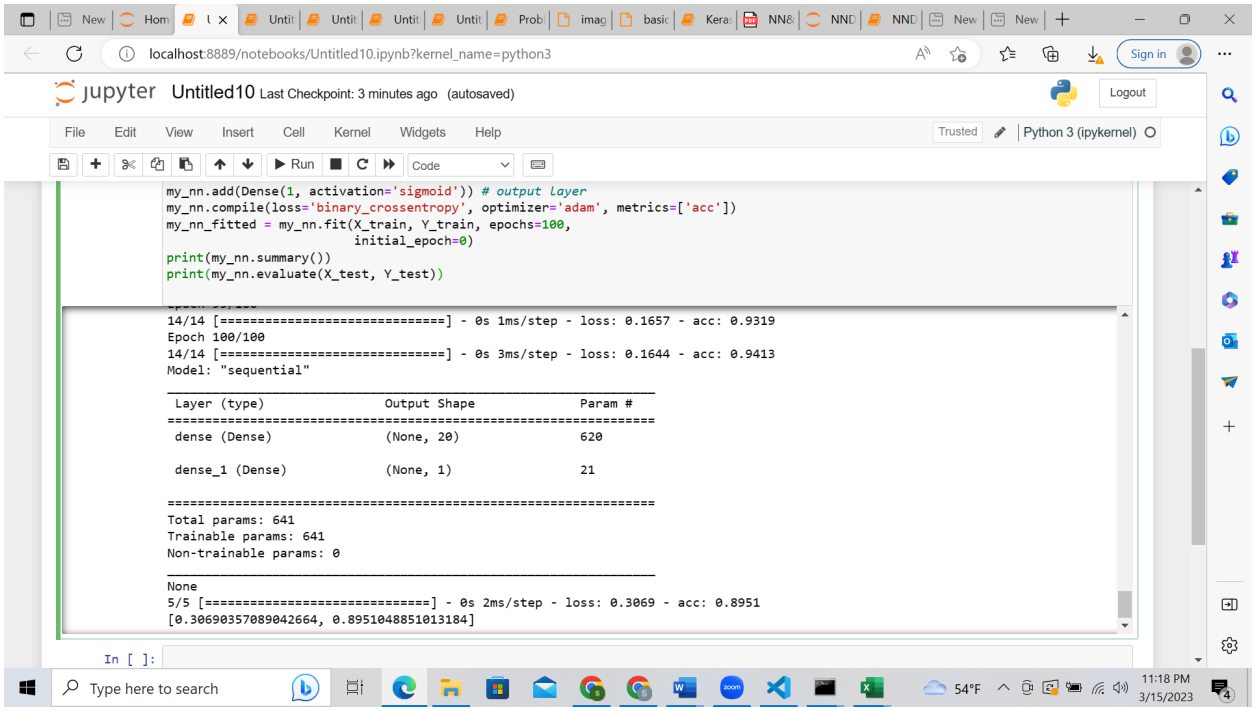
Accuracy of the model increased from 0.6198 to 0.6771 after adding 2 more dense layers.Therefore, by adding additional layers the model has more parameters to learn from the training data which may lead to better accuracy.

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

Ans)
Code change after changing the data source to Breast Cancer dataset



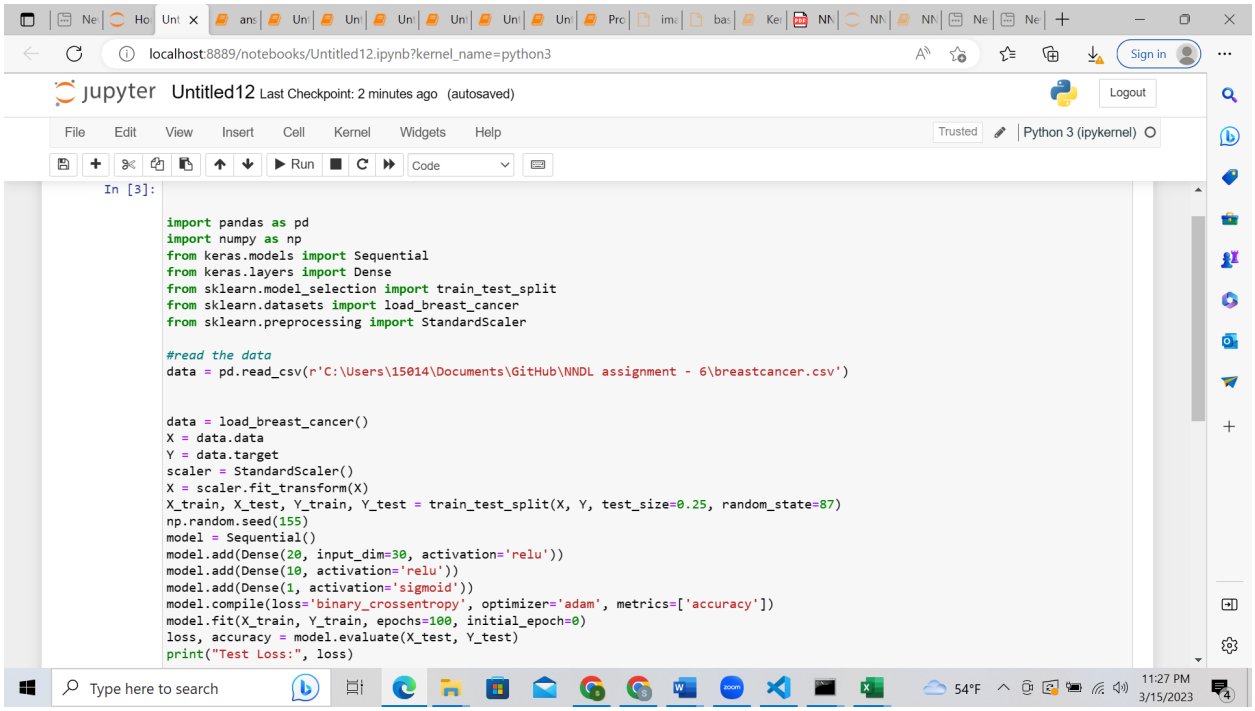
Accuracy of the model after changing the data source to the breast cancer :



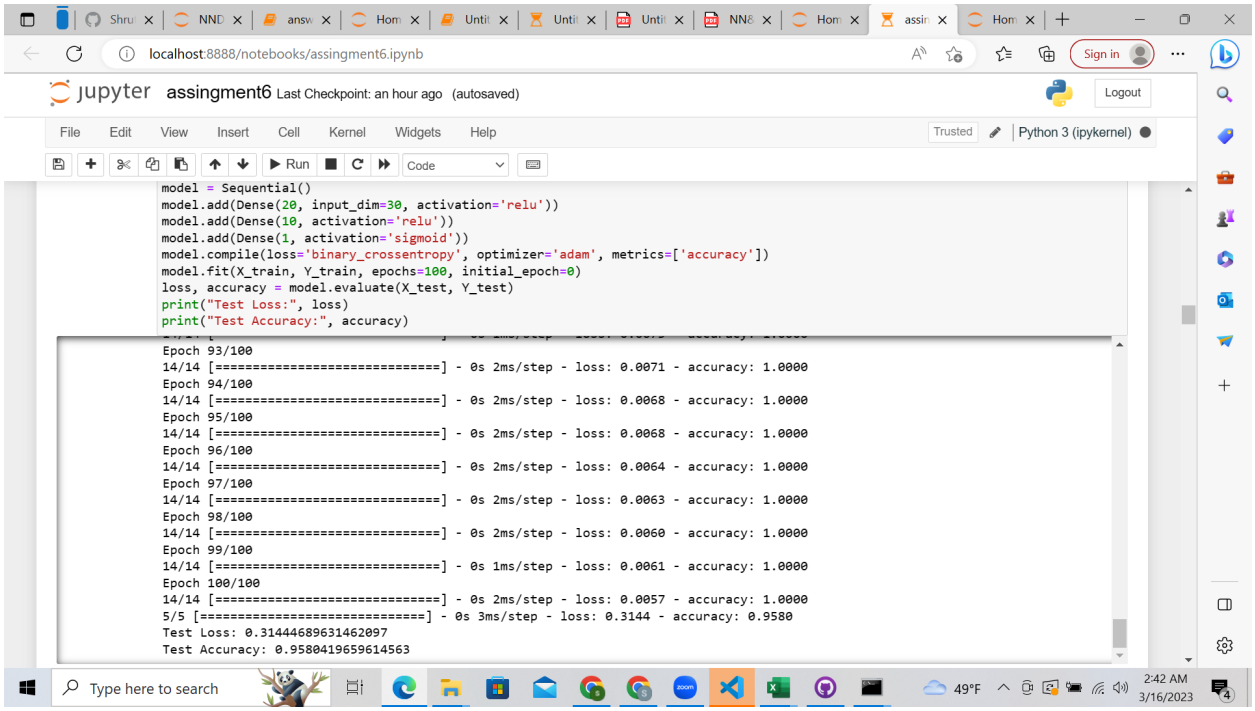
c) Normalize the data before feeding the data to the model and check how the normalization change your accuracy (code given below).

from sklearn.preprocessing
import StandardScaler sc = StandardScaler()

Ans)



Output

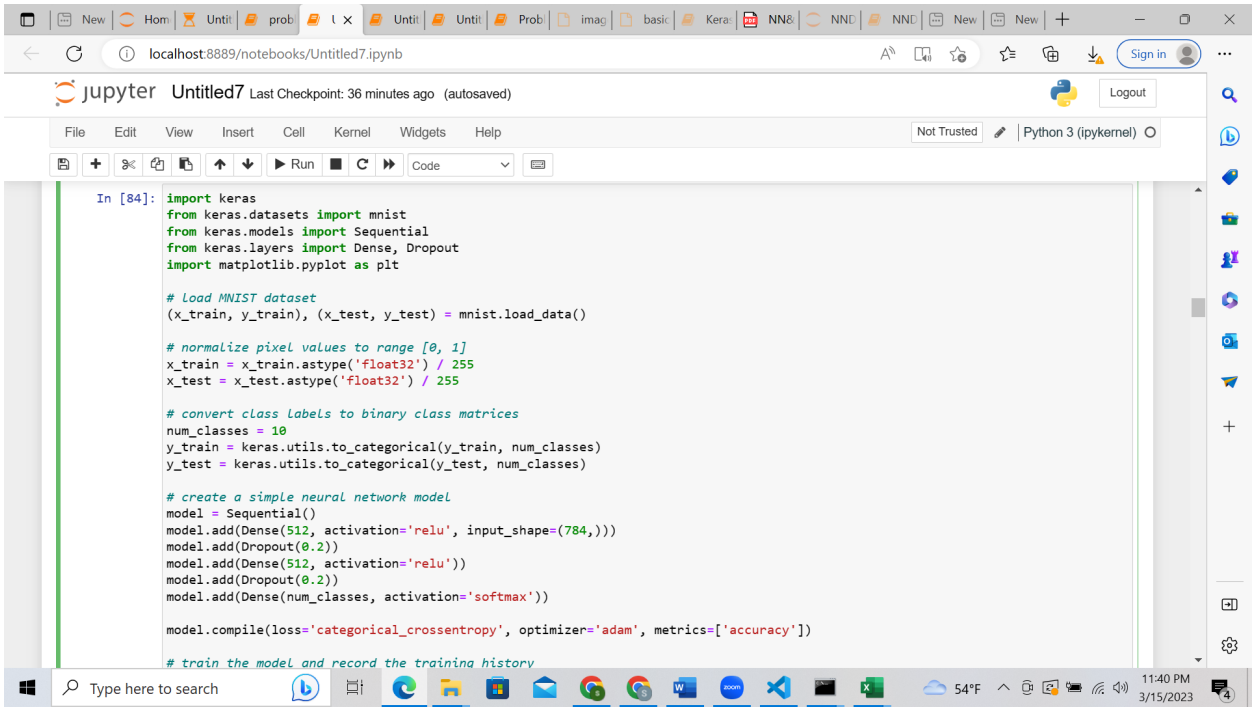


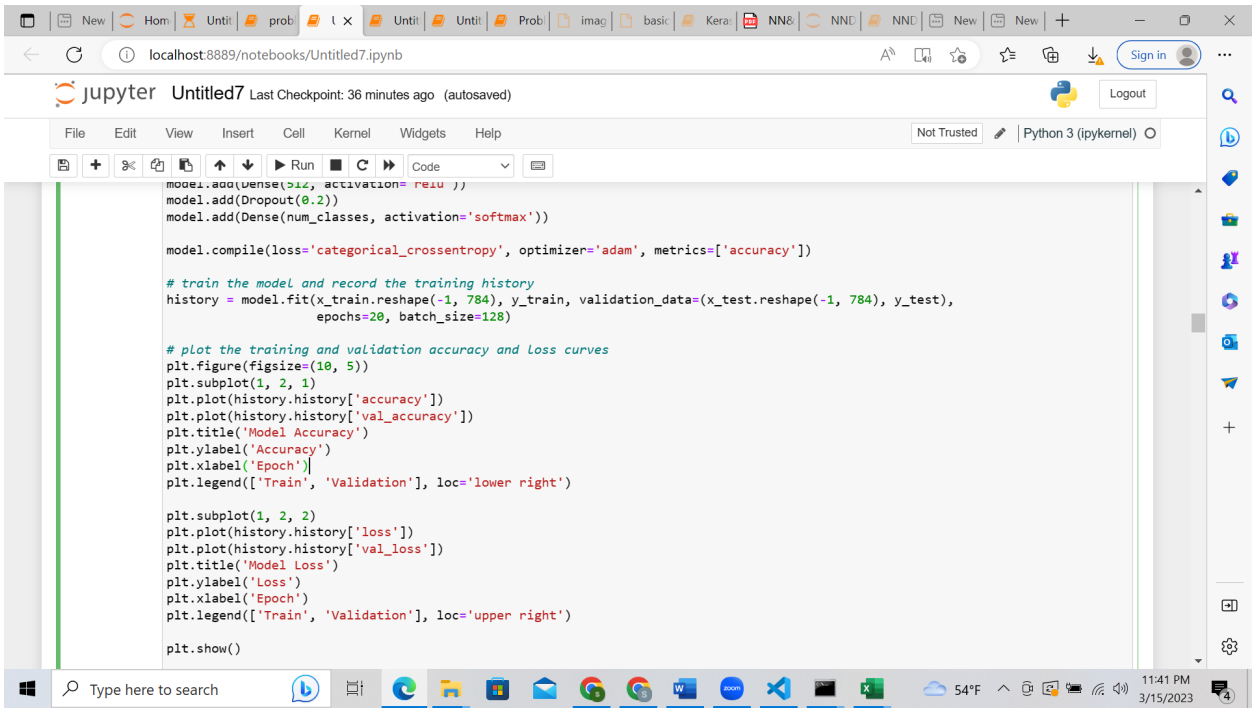
2)Use Image Classification on the handwritten digits data set (mnist)

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

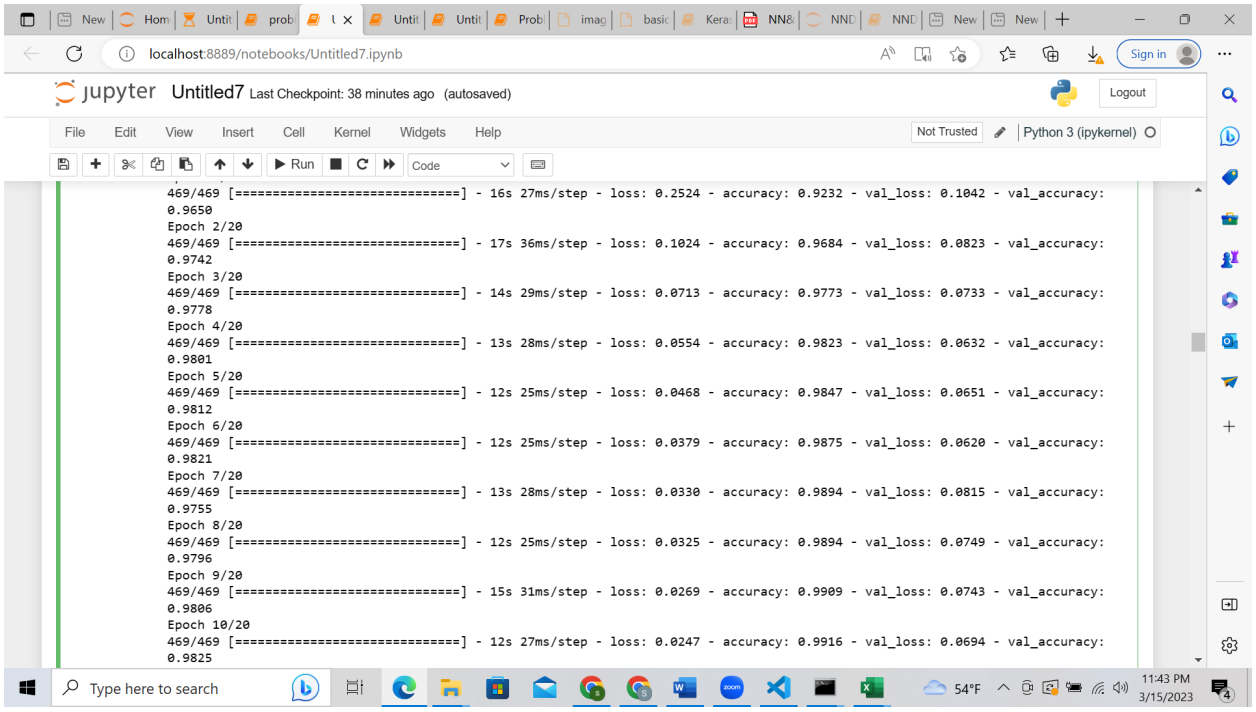
Ans)

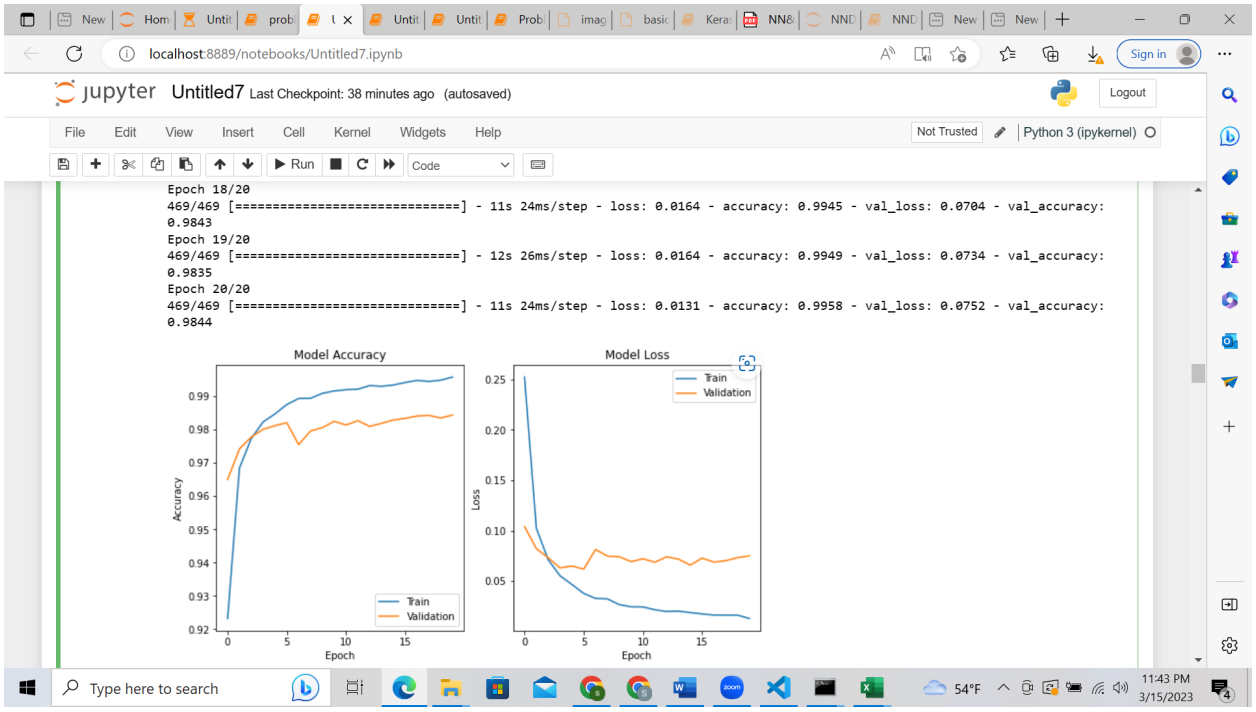
Code for the above code is



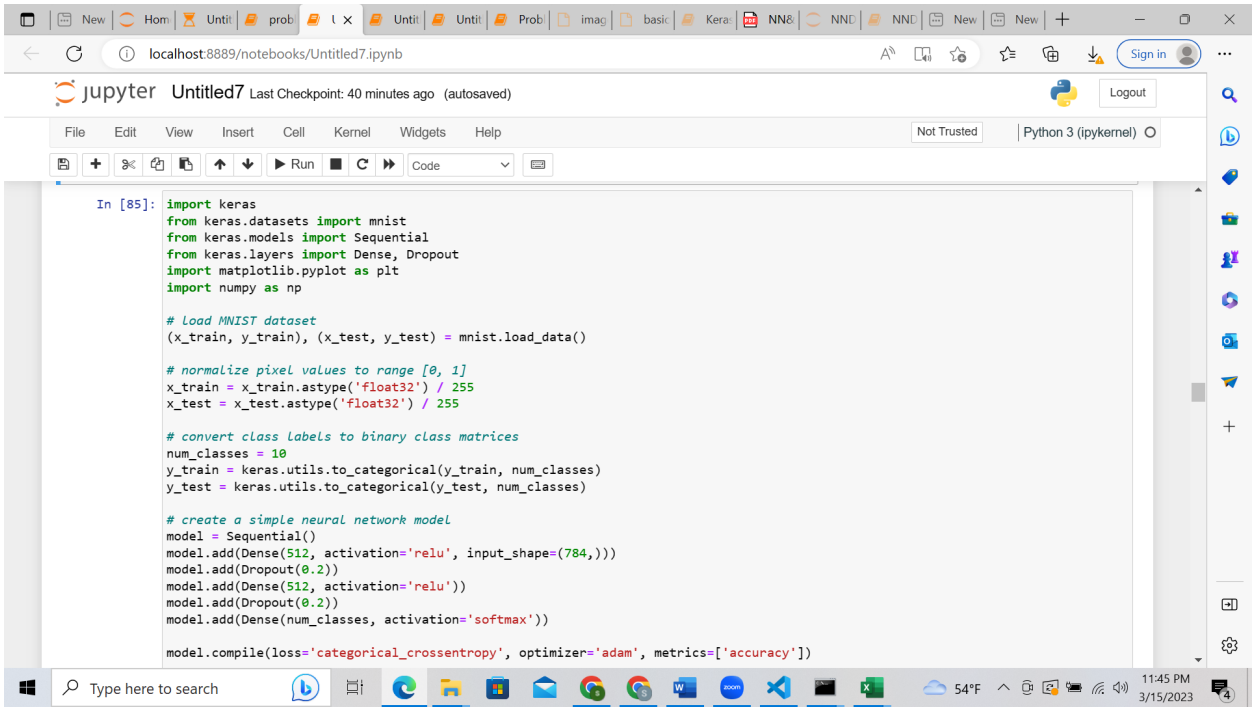


Output of the code

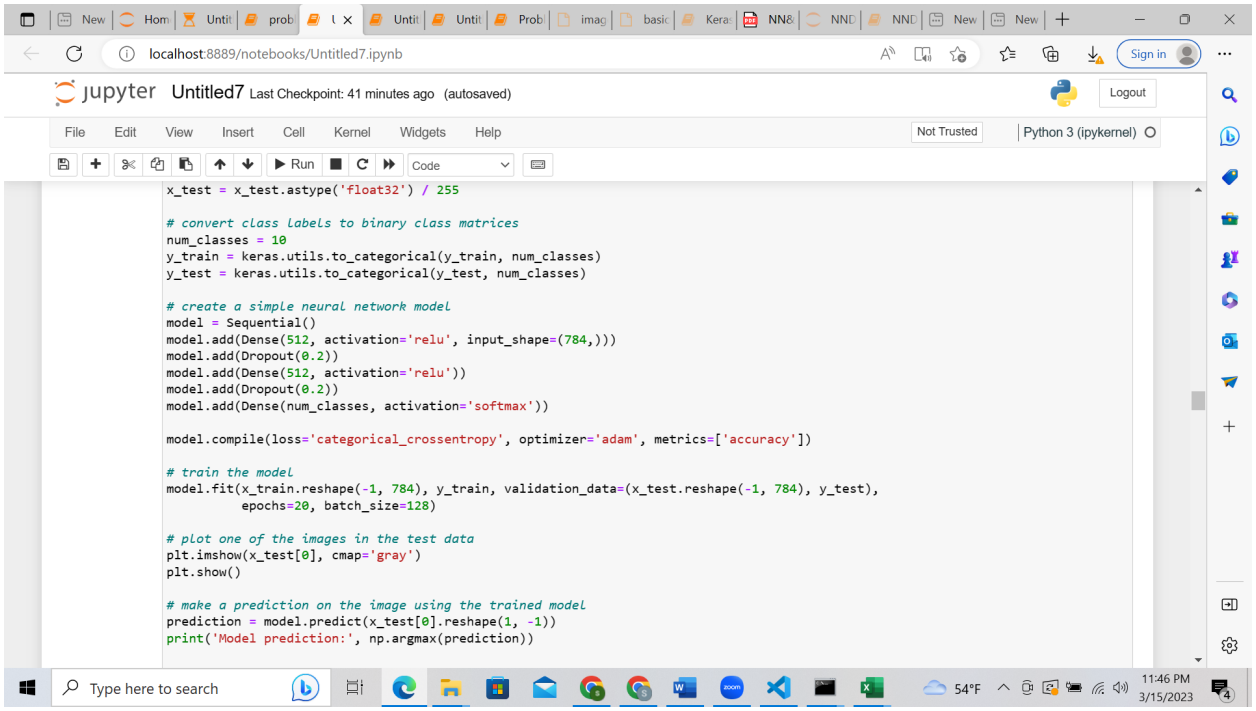




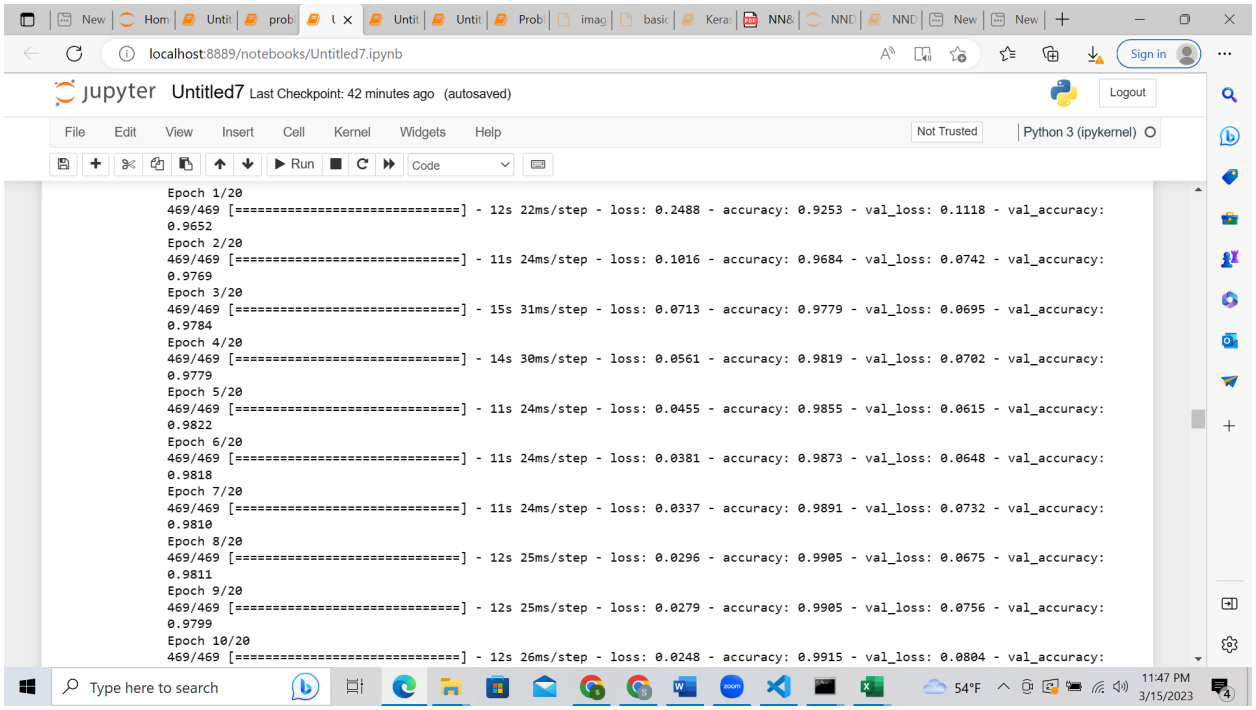
b). Plot one of the images in the test data, and then do inferencing to check what is the prediction of the model on that single image.

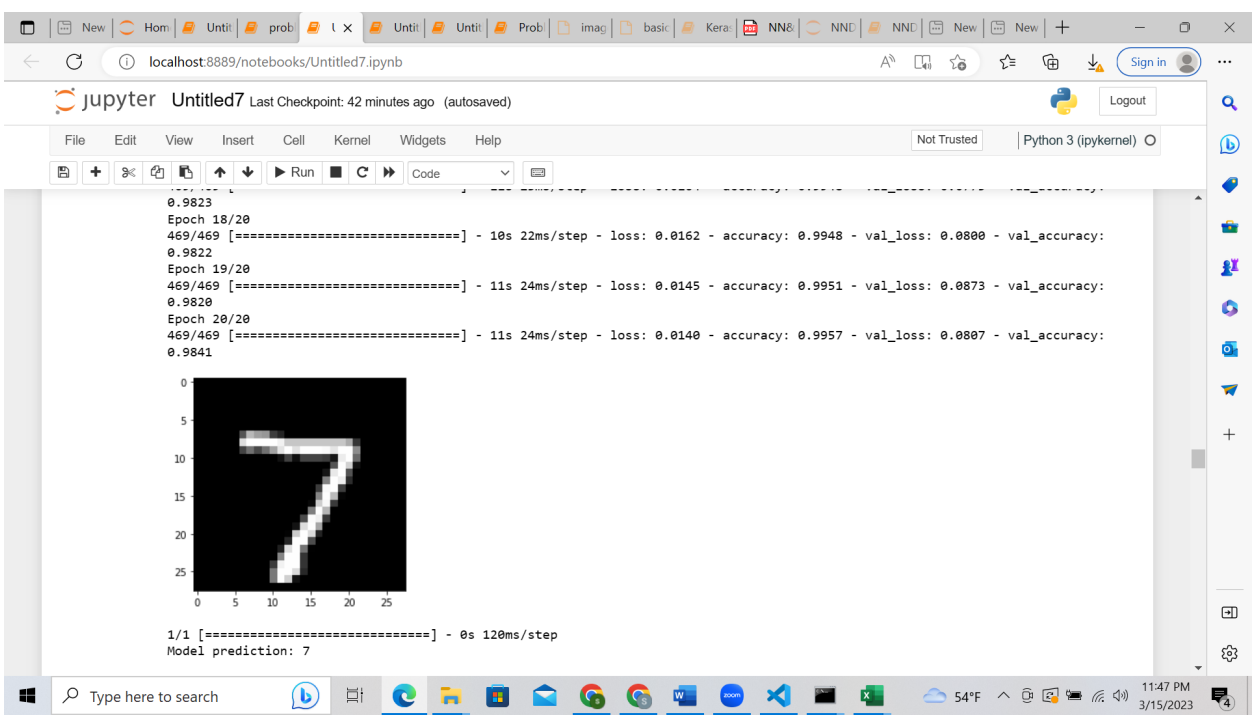


```
In [85]:
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

# Load MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# normalize pixel values to range [0, 1]
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255

# convert class labels to binary class matrices
num_classes = 10
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

# create a simple neural network model
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```



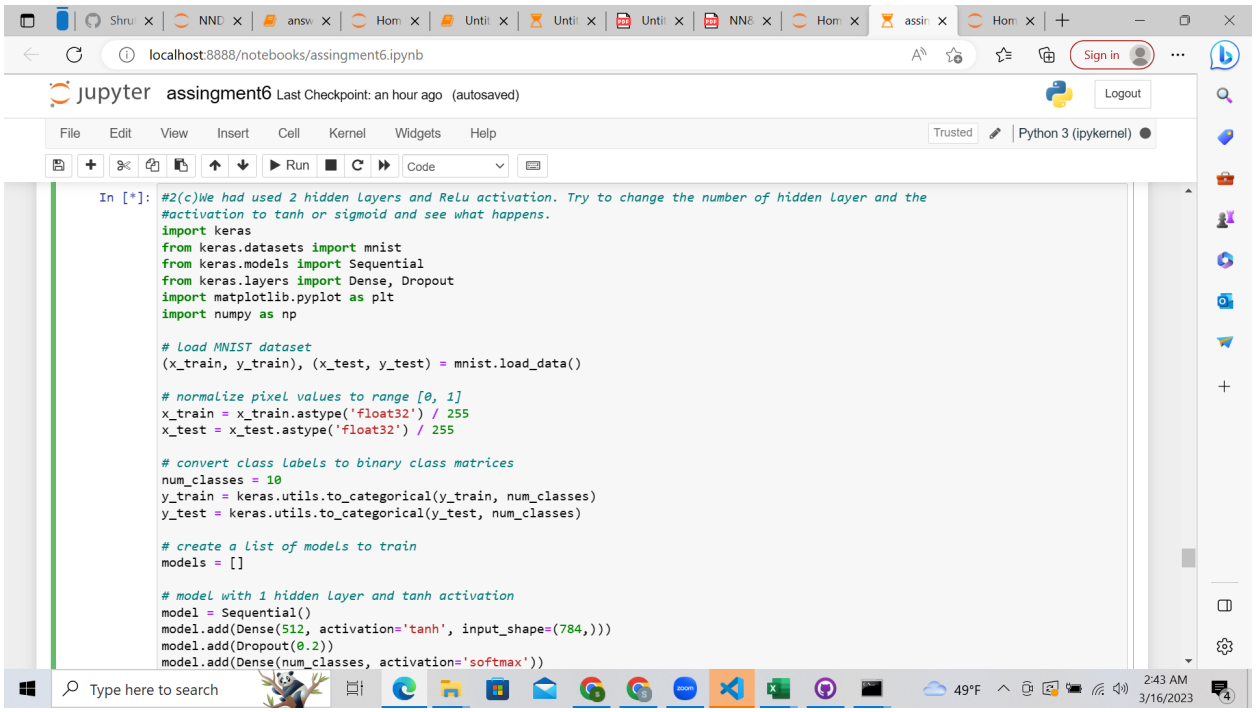
```
x_test = x_test.astype('float32') / 255

# convert class labels to binary class matrices
num_classes = 10
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

# create a simple neural network model
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# train the model
model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.reshape(-1, 784), y_test),
          epochs=20, batch_size=128)

# plot one of the images in the test data
plt.imshow(x_test[0], cmap='gray')
plt.show()

# make a prediction on the image using the trained model
prediction = model.predict(x_test[0].reshape(1, -1))
print('Model prediction:', np.argmax(prediction))
```

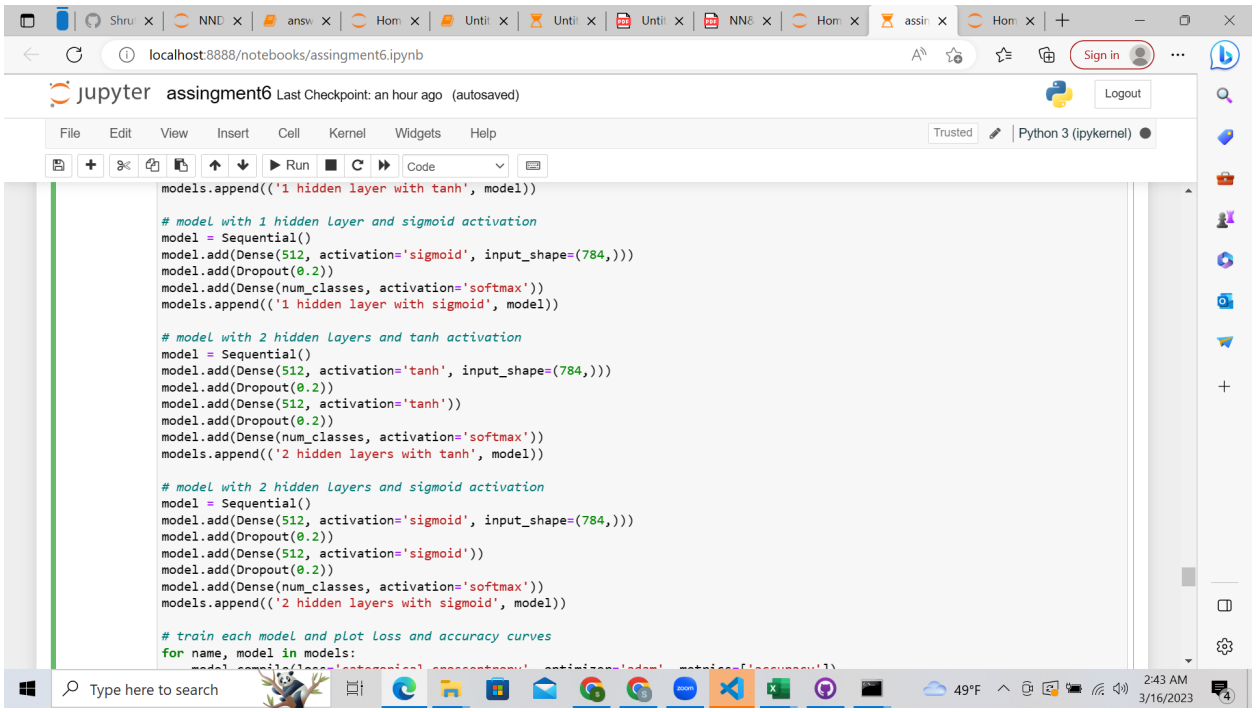

localhost:8889/notebooks/Untitled7.ipynb
jupyter Untitled7 Last Checkpoint: 42 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Epoch 1/20
469/469 [==============================] - 12s 22ms/step - loss: 0.2488 - accuracy: 0.9253 - val_loss: 0.1118 - val_accuracy:
0.9652
Epoch 2/20
469/469 [==============================] - 11s 24ms/step - loss: 0.1016 - accuracy: 0.9684 - val_loss: 0.0742 - val_accuracy:
0.9769
Epoch 3/20
469/469 [==============================] - 15s 31ms/step - loss: 0.0713 - accuracy: 0.9779 - val_loss: 0.0695 - val_accuracy:
0.9784
Epoch 4/20
469/469 [==============================] - 14s 30ms/step - loss: 0.0561 - accuracy: 0.9819 - val_loss: 0.0702 - val_accuracy:
0.9779
Epoch 5/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0455 - accuracy: 0.9855 - val_loss: 0.0615 - val_accuracy:
0.9822
Epoch 6/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0381 - accuracy: 0.9873 - val_loss: 0.0648 - val_accuracy:
0.9818
Epoch 7/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0337 - accuracy: 0.9891 - val_loss: 0.0732 - val_accuracy:
0.9810
Epoch 8/20
469/469 [==============================] - 12s 25ms/step - loss: 0.0296 - accuracy: 0.9905 - val_loss: 0.0675 - val_accuracy:
0.9811
Epoch 9/20
469/469 [==============================] - 12s 25ms/step - loss: 0.0279 - accuracy: 0.9905 - val_loss: 0.0756 - val_accuracy:
0.9799
Epoch 10/20
469/469 [==============================] - 12s 26ms/step - loss: 0.0248 - accuracy: 0.9915 - val_loss: 0.0804 - val_accuracy:
Type here to search
54°F
11:47 PM
3/15/2023

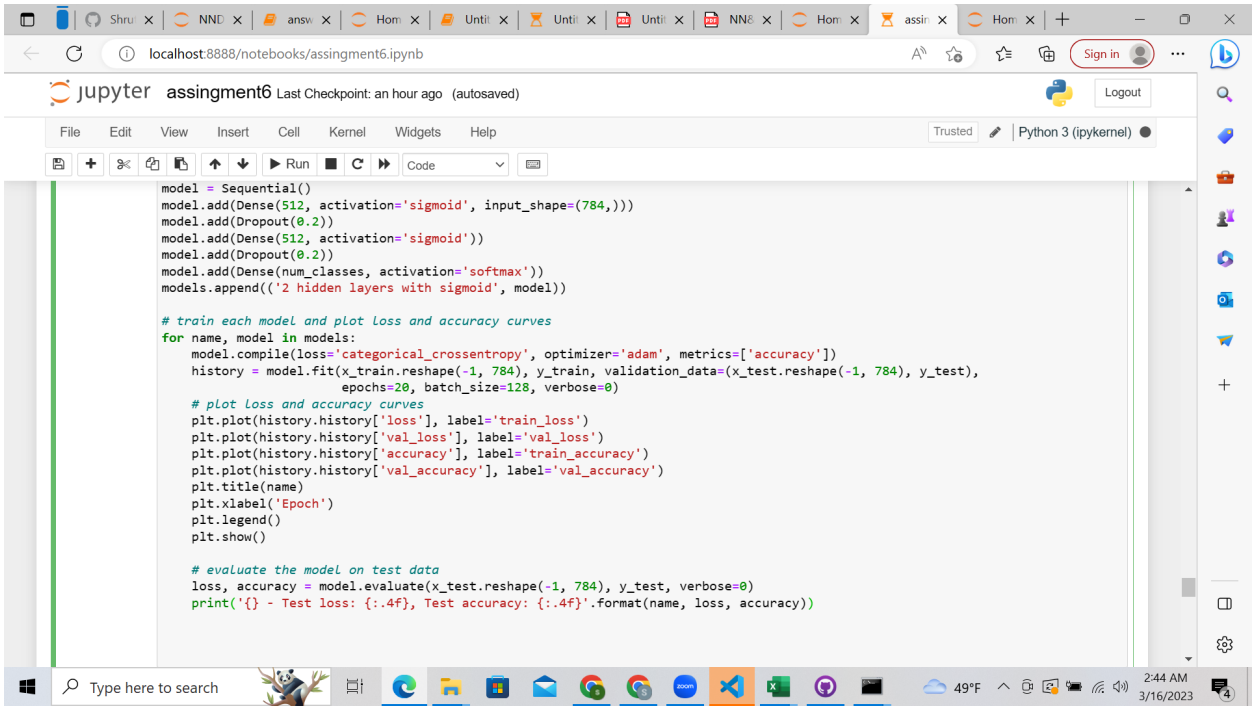

localhost:8889/notebooks/Untitled7.ipynb
jupyter Untitled7 Last Checkpoint: 42 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
0.9823
Epoch 18/20
469/469 [==============================] - 10s 22ms/step - loss: 0.0162 - accuracy: 0.9948 - val_loss: 0.0800 - val_accuracy:
0.9822
Epoch 19/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0145 - accuracy: 0.9951 - val_loss: 0.0873 - val_accuracy:
0.9820
Epoch 20/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0140 - accuracy: 0.9957 - val_loss: 0.0807 - val_accuracy:
0.9841
0
5
10
15
20
25
0 5 10 15 20 25
1/1 [==============================] - 0s 120ms/step
Model prediction: 7
Type here to search
54°F
11:47 PM
3/15/2023

c.)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.

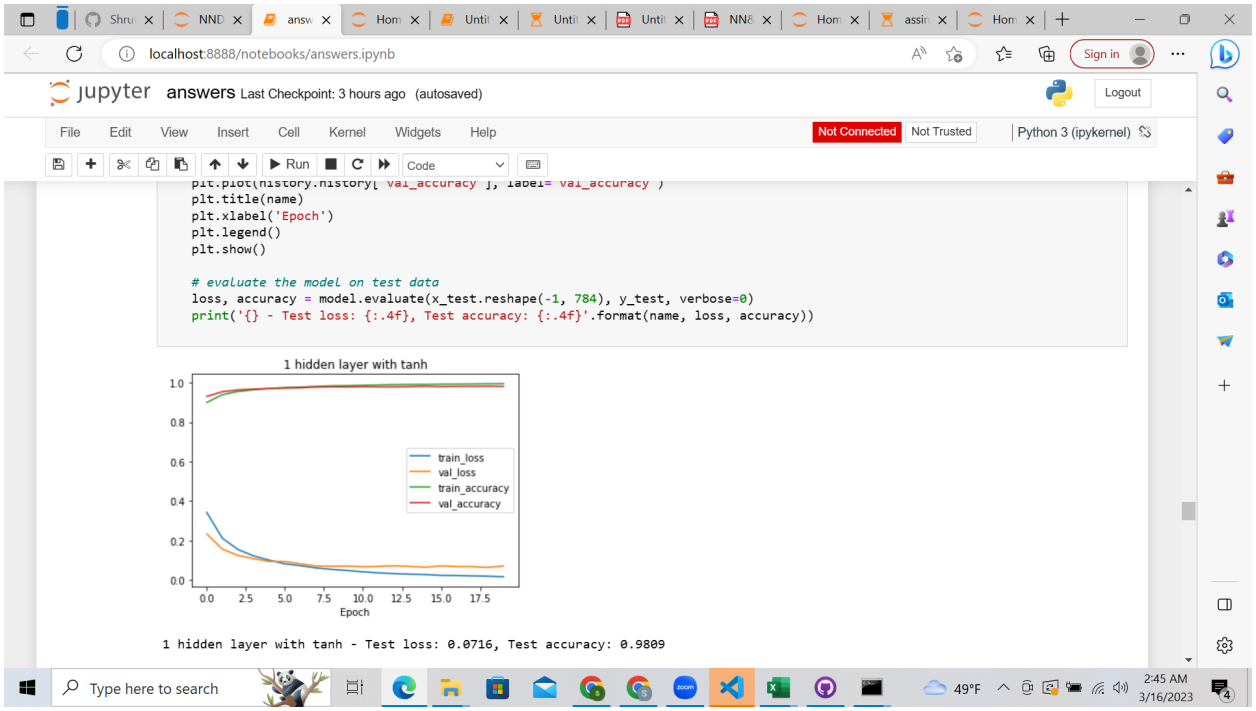
Ans)
Code

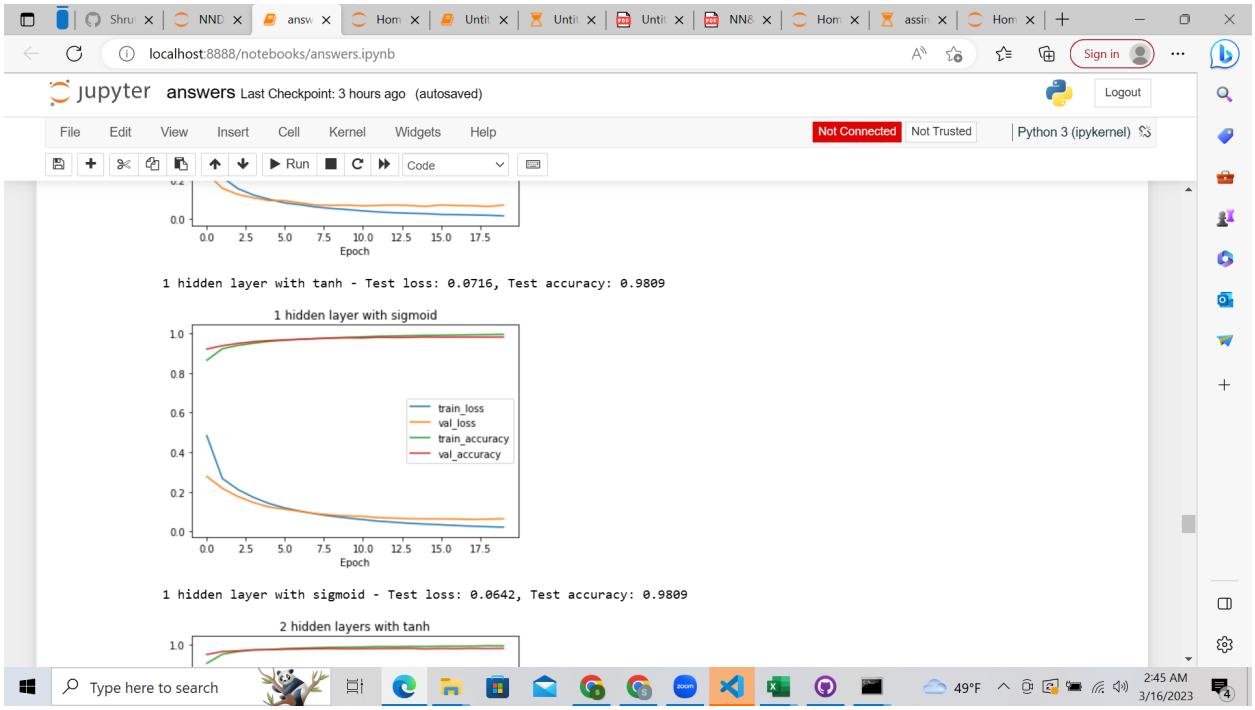


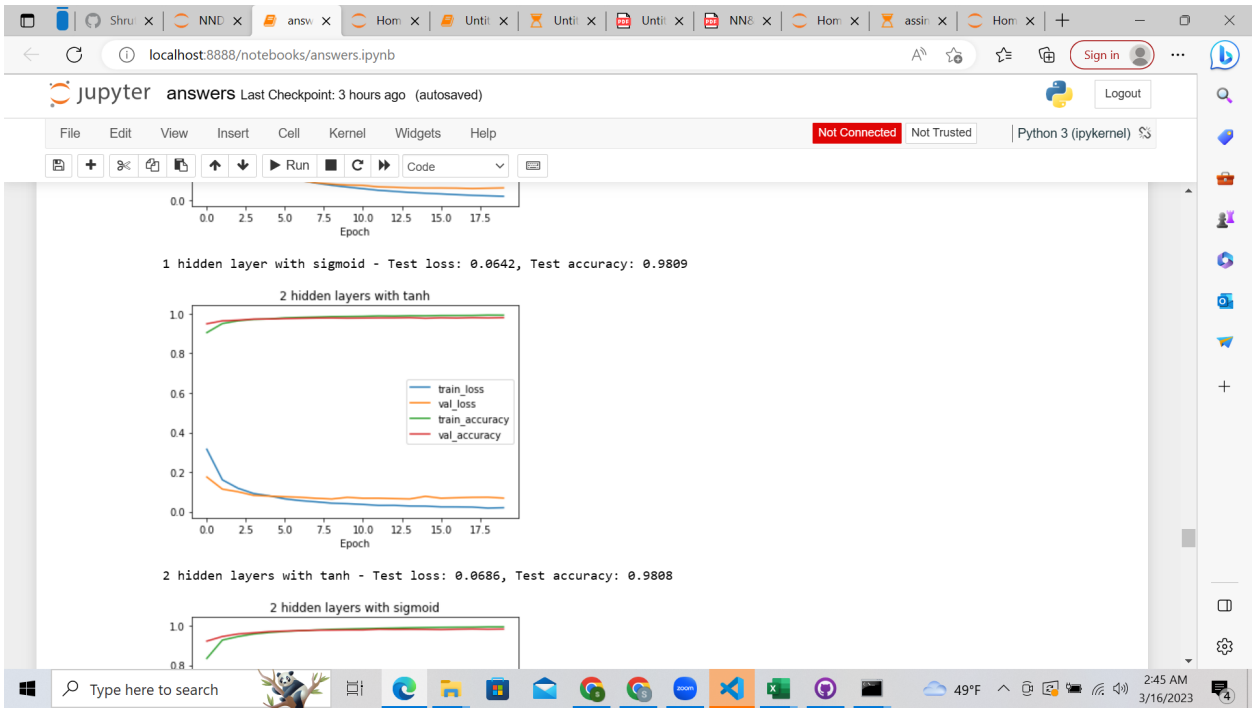


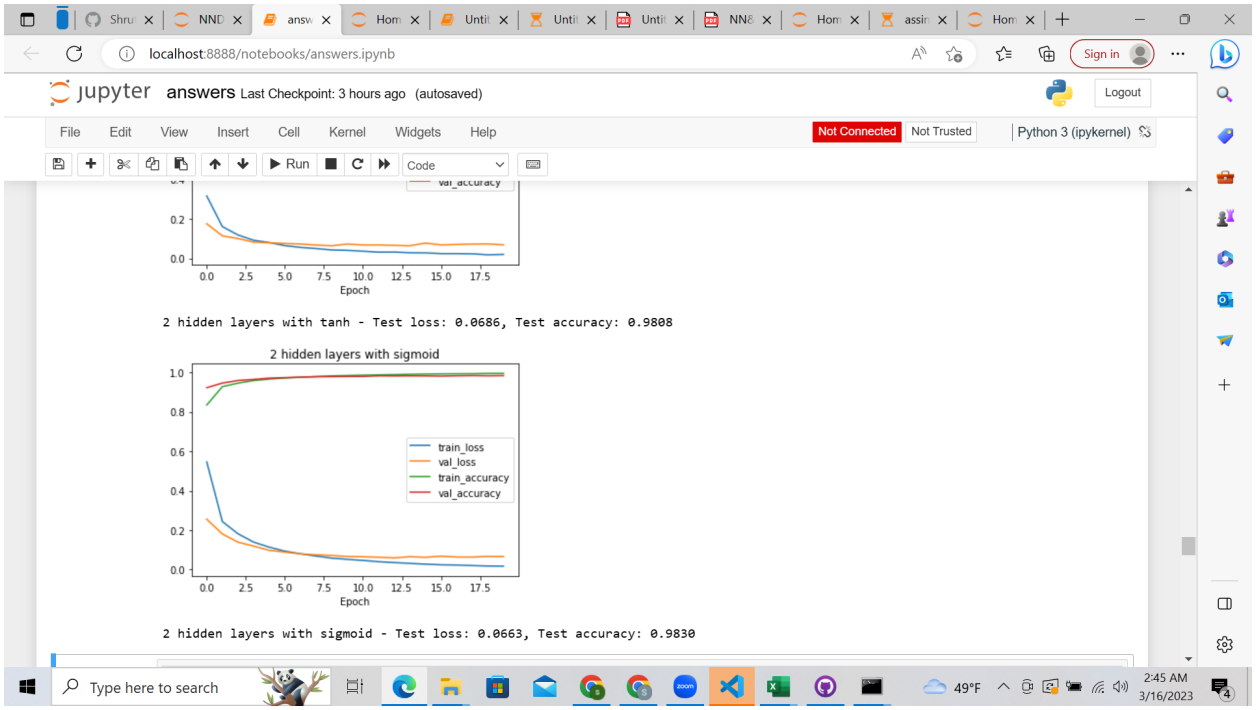

localhost:8888/notebooks/assingment6.ipynb
jupyter assingment6 Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
model = Sequential()
model.add(Dense(512, activation='sigmoid', input_shape=(784,)))
model.add(Dropout(0.2))
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'])
history = model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.reshape(-1, 784), y_test),
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))
Type here to search
49°F
2:44 AM
3/16/2023

Output


localhost:8888/notebooks/answers.ipynb
jupyter answers Last Checkpoint: 3 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Connected
Not Trusted
Python 3 (ipykernel)
Run
Code
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))
1 hidden layer with tanh
train_loss
val_loss
train_accuracy
val_accuracy
Epoch
1 hidden layer with tanh - Test loss: 0.0716, Test accuracy: 0.9809
Type here to search
49°F
2:45 AM
3/16/2023

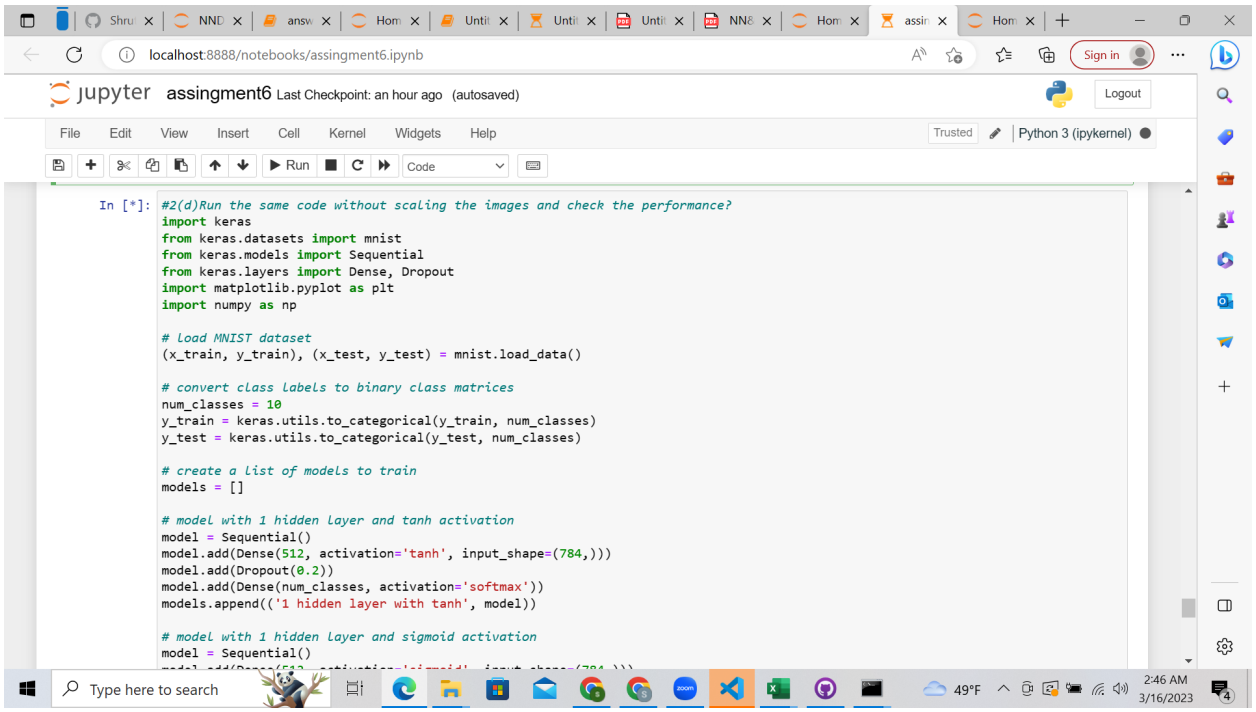

jupyter answers Last Checkpoint: 3 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Connected
Not Trusted
Python 3 (ipykernel)
Run
Code
Epoch
1 hidden layer with tanh - Test loss: 0.0716, Test accuracy: 0.9809
1 hidden layer with sigmoid
train_loss
val_loss
train_accuracy
val_accuracy
Epoch
1 hidden layer with sigmoid - Test loss: 0.0642, Test accuracy: 0.9809
2 hidden layers with tanh
Type here to search
49°F
2:45 AM
3/16/2023

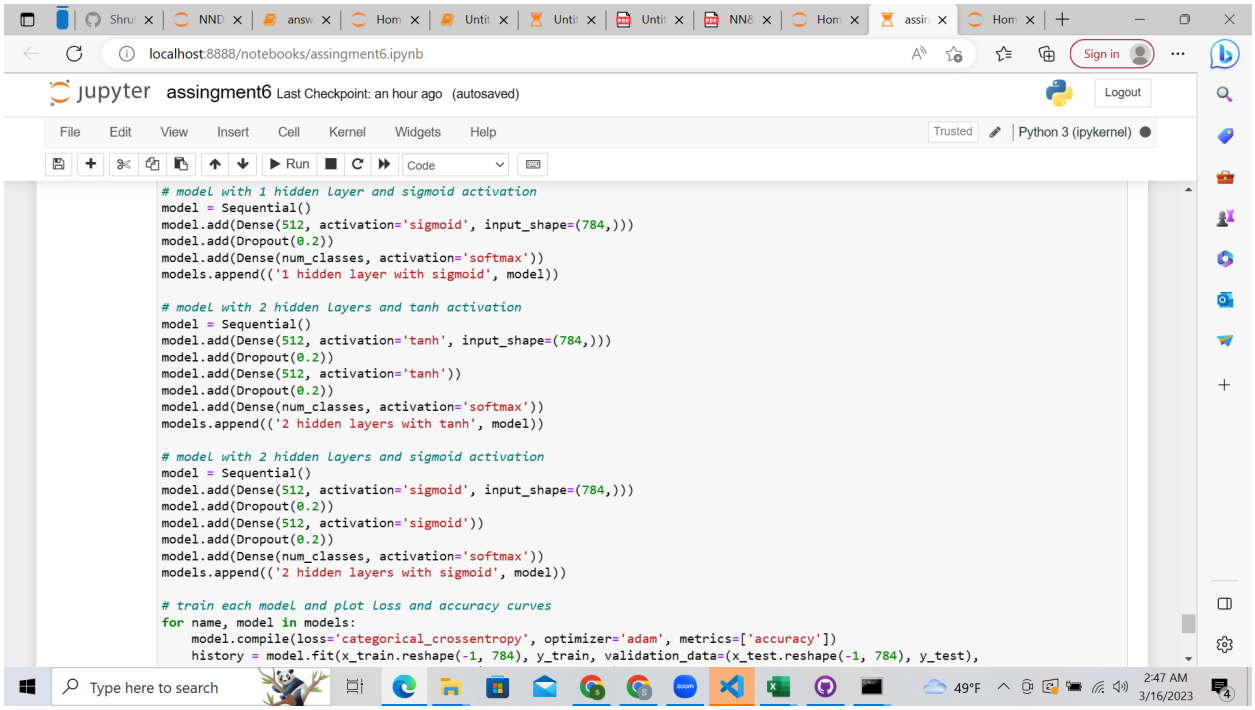

jupyter answers Last Checkpoint: 3 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Connected
Not Trusted
Python 3 (ipykernel)
Run
Code
Epoch
1 hidden layer with sigmoid - Test loss: 0.0642, Test accuracy: 0.9809
2 hidden layers with tanh
train_loss
val_loss
train_accuracy
val_accuracy
Epoch
2 hidden layers with tanh - Test loss: 0.0686, Test accuracy: 0.9808
2 hidden layers with sigmoid
Type here to search
49°F
2:45 AM
3/16/2023

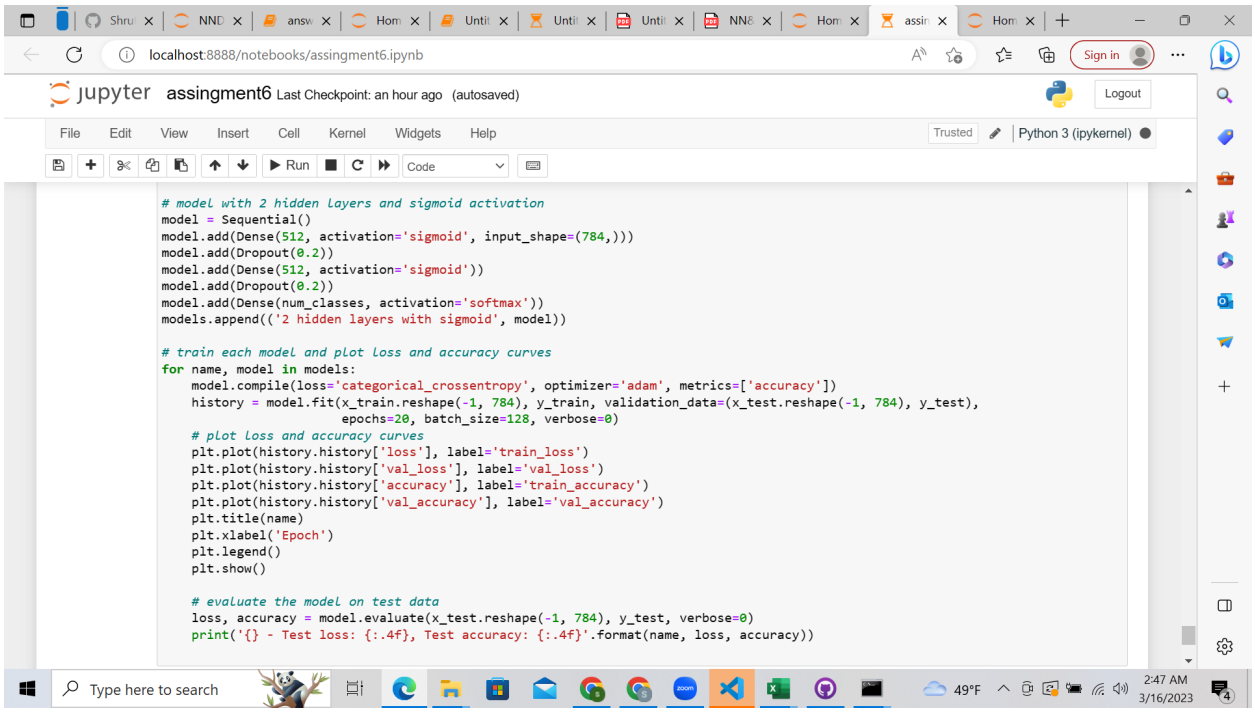


d.) Run the same code without scaling the images and check the performance?

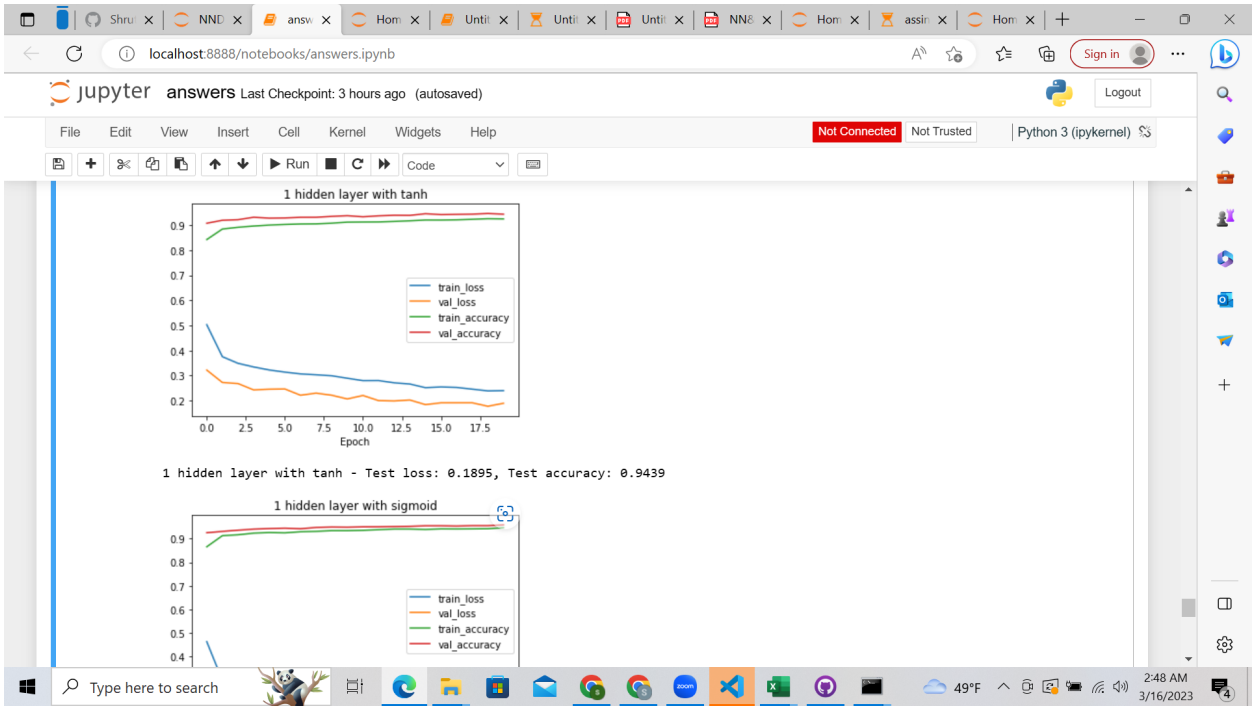
Code

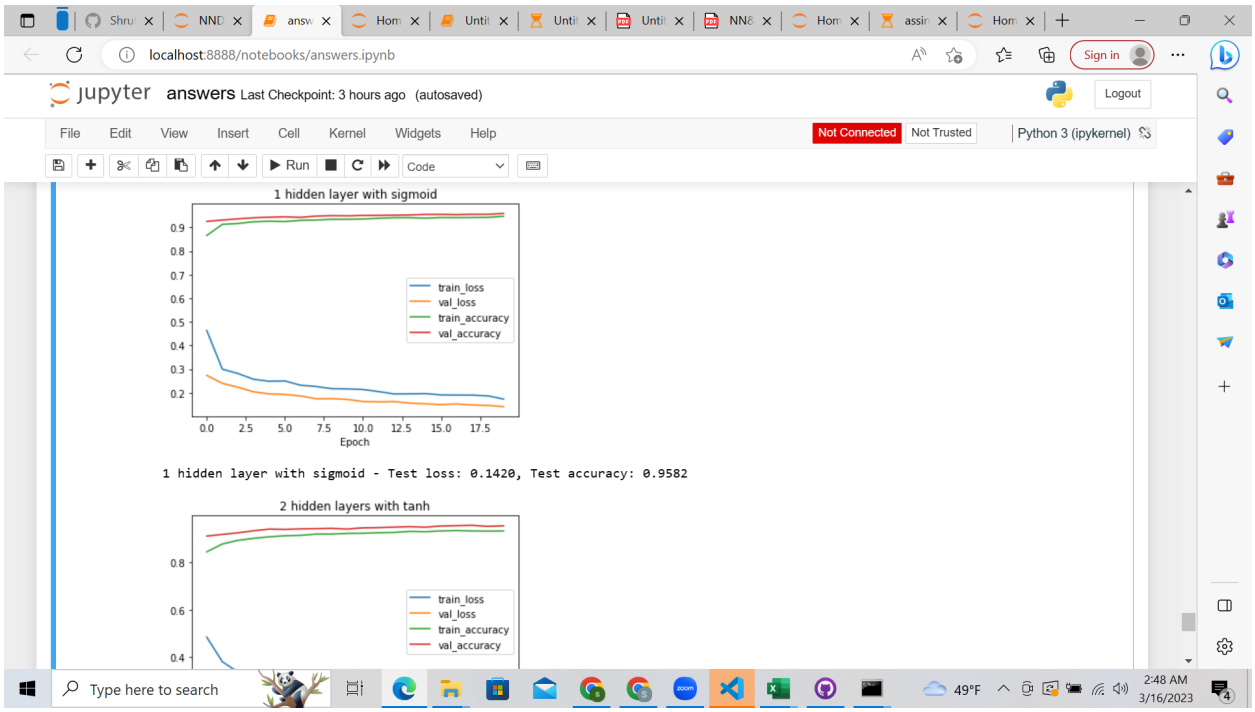


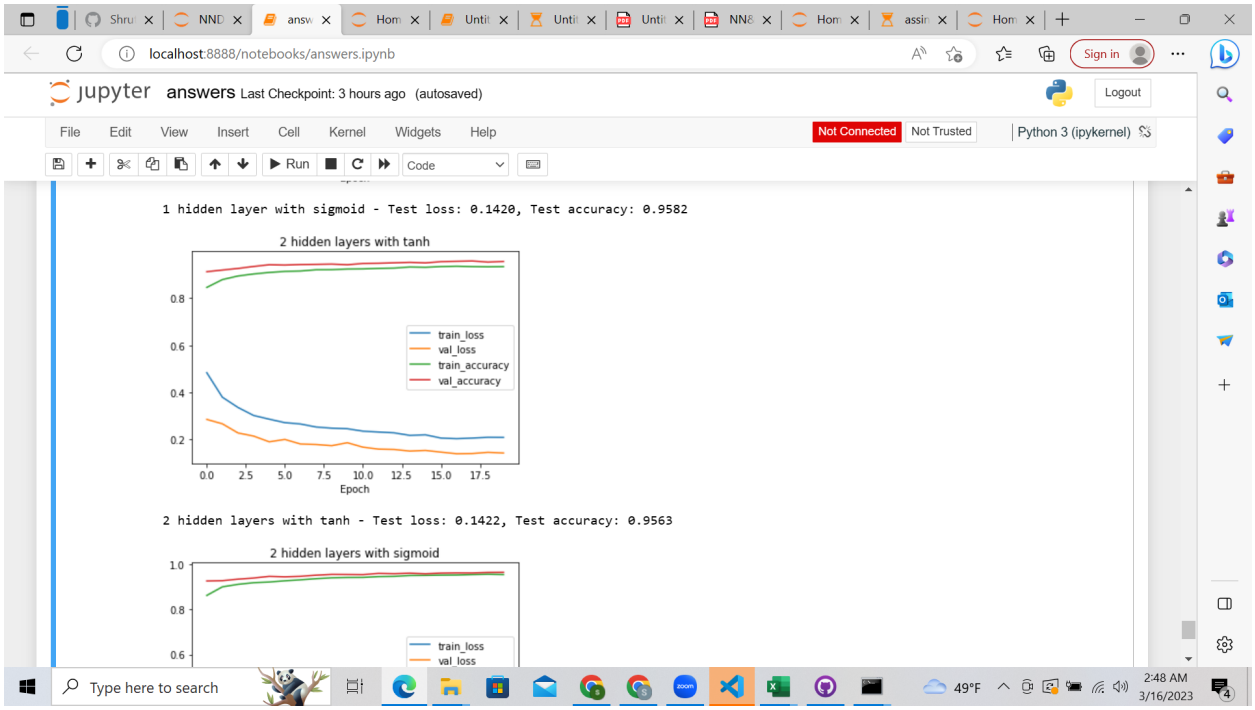

localhost:8888/notebooks/assingment6.ipynb
jupyter assingment6 Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
# model with 1 hidden layer and sigmoid activation
model = Sequential()
model.add(Dense(512, activation='sigmoid', input_shape=(784,)))
model.add(Dropout(0.2))
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'])
history = model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.reshape(-1, 784), y_test),
Type here to search
49°F
2:47 AM
3/16/2023

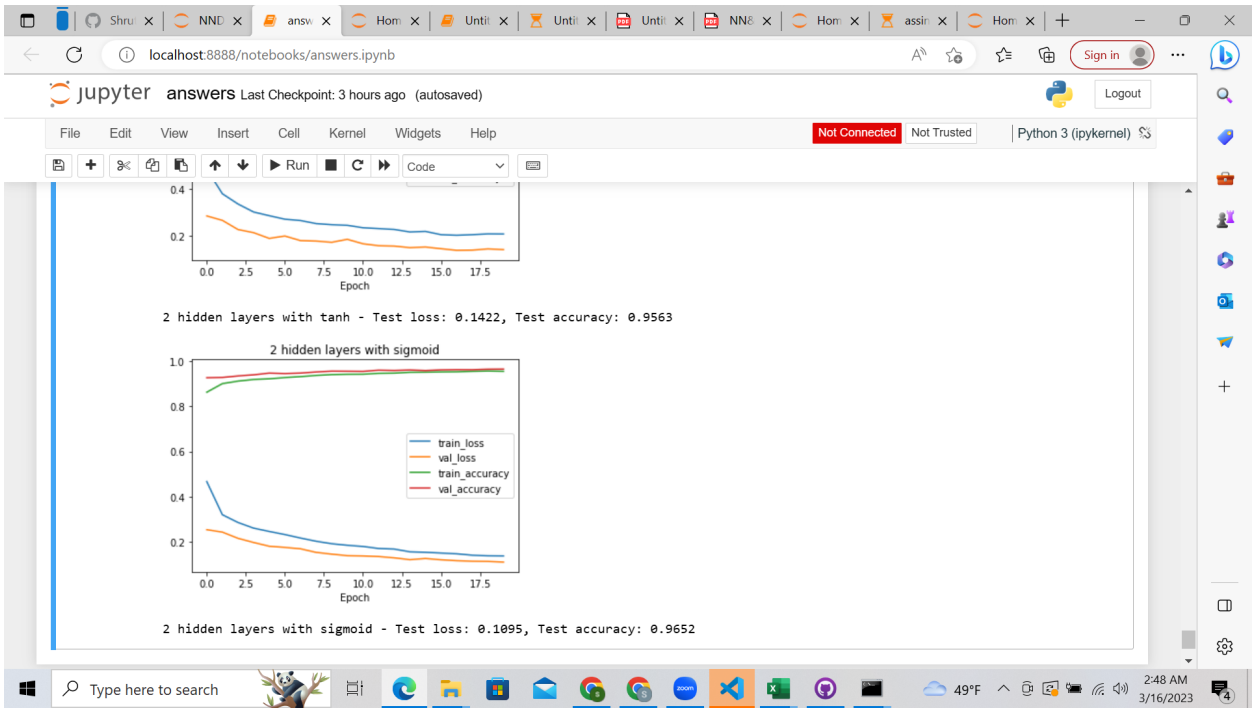

localhost:8888/notebooks/assingment6.ipynb
jupyter assingment6 Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
# model with 2 hidden layers and sigmoid activation
model = Sequential()
model.add(Dense(512, activation='sigmoid', 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'])
history = model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.reshape(-1, 784), y_test),
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))
Type here to search
49°F
2:47 AM
3/16/2023

Output


jupyter answers Last Checkpoint: 3 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Connected
Not Trusted
Python 3 (ipykernel)
Run
Code
1 hidden layer with tanh
train_loss
val_loss
train_accuracy
val_accuracy
Epoch
1 hidden layer with tanh - Test loss: 0.1895, Test accuracy: 0.9439
1 hidden layer with sigmoid
Type here to search
49°F
2:48 AM
3/16/2023


jupyter answers Last Checkpoint: 3 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Connected
Not Trusted
Python 3 (ipykernel)
Run
Code
1 hidden layer with sigmoid
train_loss
val_loss
train_accuracy
val_accuracy
Epoch
1 hidden layer with sigmoid - Test loss: 0.1420, Test accuracy: 0.9582
2 hidden layers with tanh
Type here to search
49°F
2:48 AM
3/16/2023





Git repo link : https://github.com/ShruthiVallapReddy/NNDL---Assignment---6.git