**CS5590 APL**

**-**

**Python Programming /Deep Learning**

**LAB2**

**Team Members:**

**Ahmed Siddiqui (30)**

**Vamsi Draksharam (14)**

**Maseerah Muradabadi (23)**

**Introduction: -**

The assignment is targeted to cover D/L concepts and trying hands on neural network models with different dataset using KERAS.

**Objectives: -**

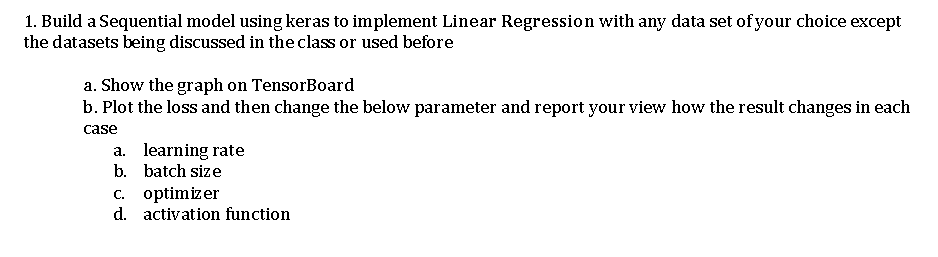
* Creation of regression and evaluating the model with MAE and Loss score.
* Creation of CNN models
* Training text classification & image classification model with different dataset and check the accuracy score for each model.
* Creating LSTM model for text classification.
* Comparing CNN and LSTM model for text classification

**Methods: -**

* Created notebooks in GoogleColab.
* Downloaded the dataset from Kaggle.
* Trained model with that dataset.
* Plotted using TensorBoard.

**Workflow: -**

**Problem 1 Statement:**



**Objective:**

The objective for the first problem is analysing the Linear Regression Model on any Sample given Dataset.

Showing the graph in TensorBoard Plot the loss and then change the below parameter to see how result changes in each case a. learning rate b. batch size c. optimizer d. activation function

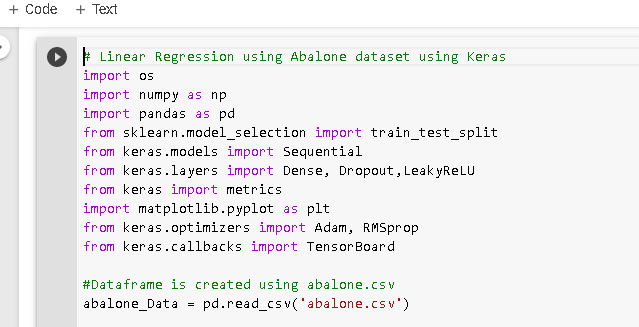
**Approach:**

To implement Linear Regression model we have taken Abalone Dataset from UCI Data Repository. Using this dataset we can predict the age of abalone from physical measurement. We have used the various Deep Learning libraries and packages like Keras Tensorboard, optimizer and Activation functions to evaluate the model.

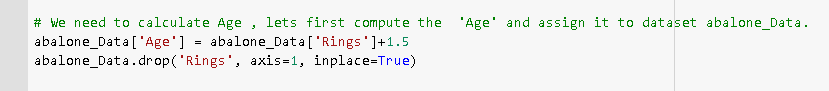
More details are given in workflow section.

**Output/ Workflow:**

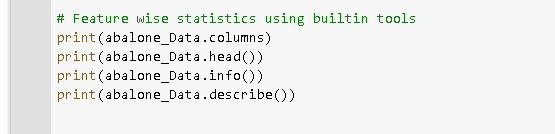
A. Loading the dataset and creating the Panda dataframe.

****

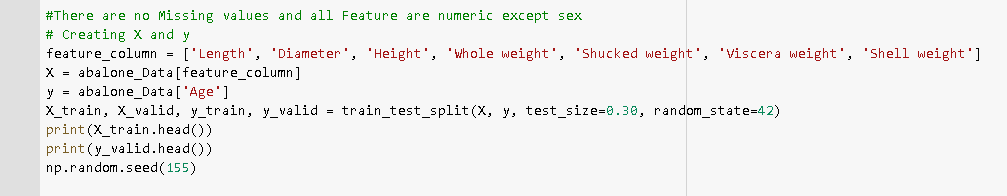
B. Created the Age Column in abalone data panda dataframe.

****

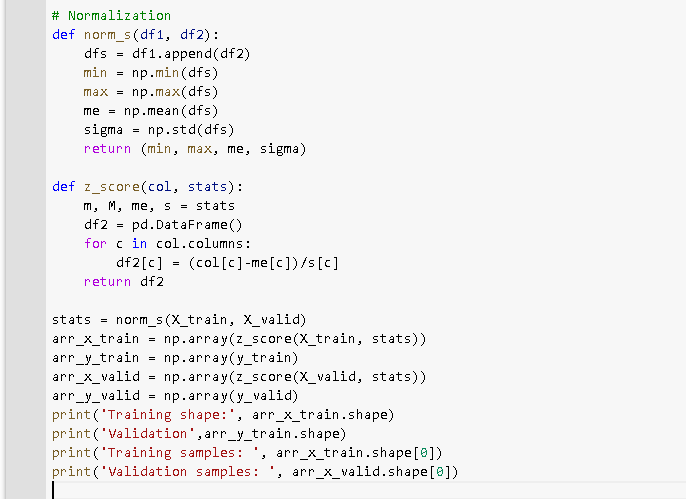
C. Feature Statistic on given dataset

****

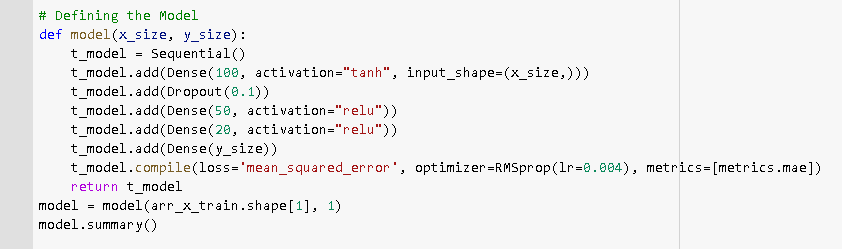
D. Creating X,Y train and validation dataframe

****

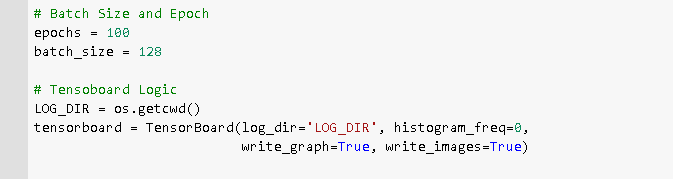
E. Normalizing the train and validation dataframe.

****

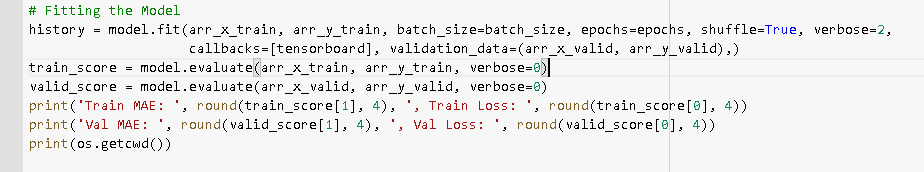
F. Model Definition

****

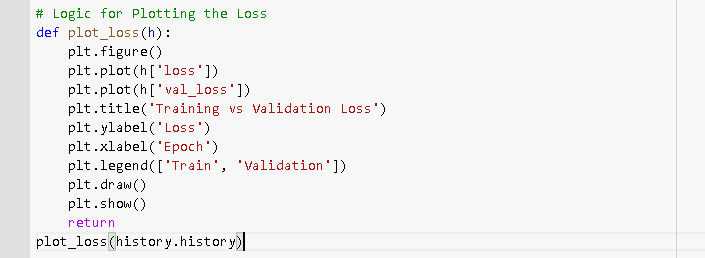
G. Defining the Number Epoch , batch size and defining tensorboard logic for graph.

****

H. Fit the Model and calculating the Train and validation score

****

I.At the last we have plotted the loss graph

****

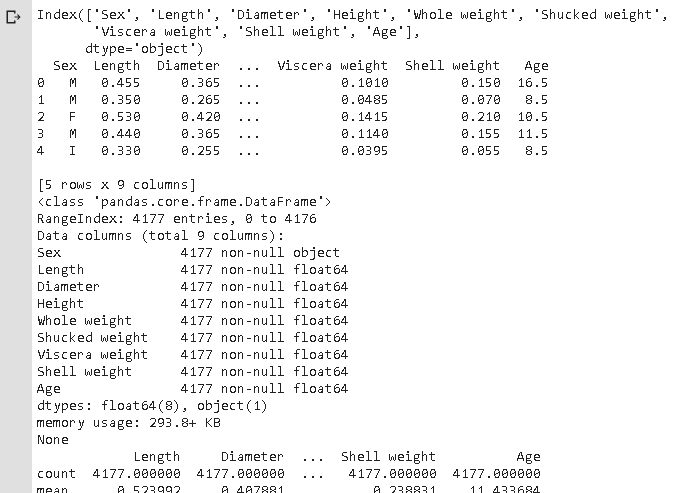
**Evaluation:**

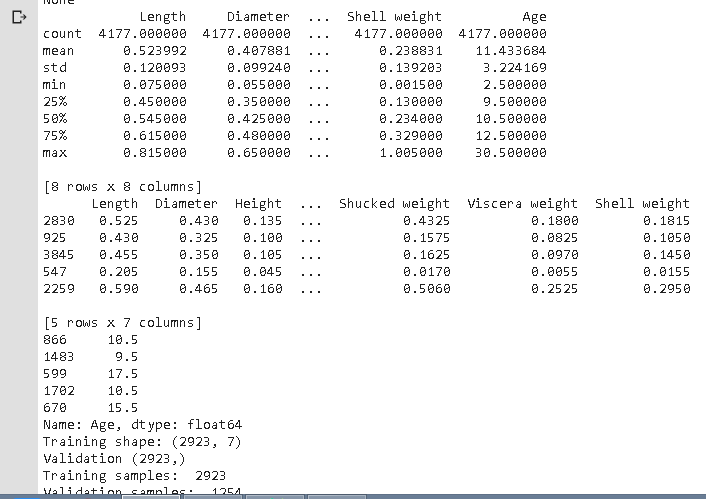
In order to evaluate the we run our program and get the train loss statistics. As we can see below for RMSprop optimizer, epoch=800 and batch size=128 and learning rate of 0.004 we got the minimum train loss.

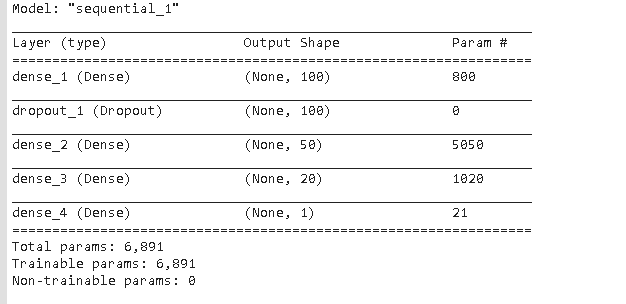
Changed the optimizer ,batch size activation function and learning rate and evaluated the train loss as shown below.

****

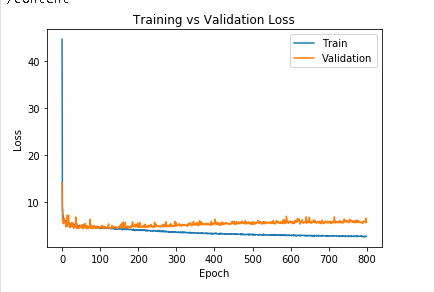
**Other output :**

****

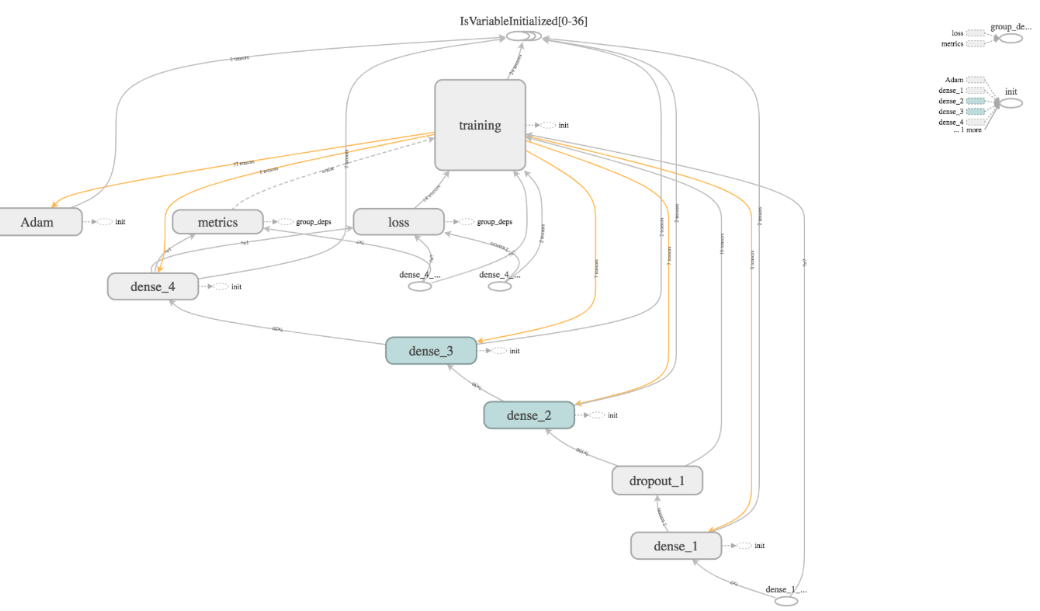
****

****

**Loss Graph:**

****

### Tensorboard Graph:

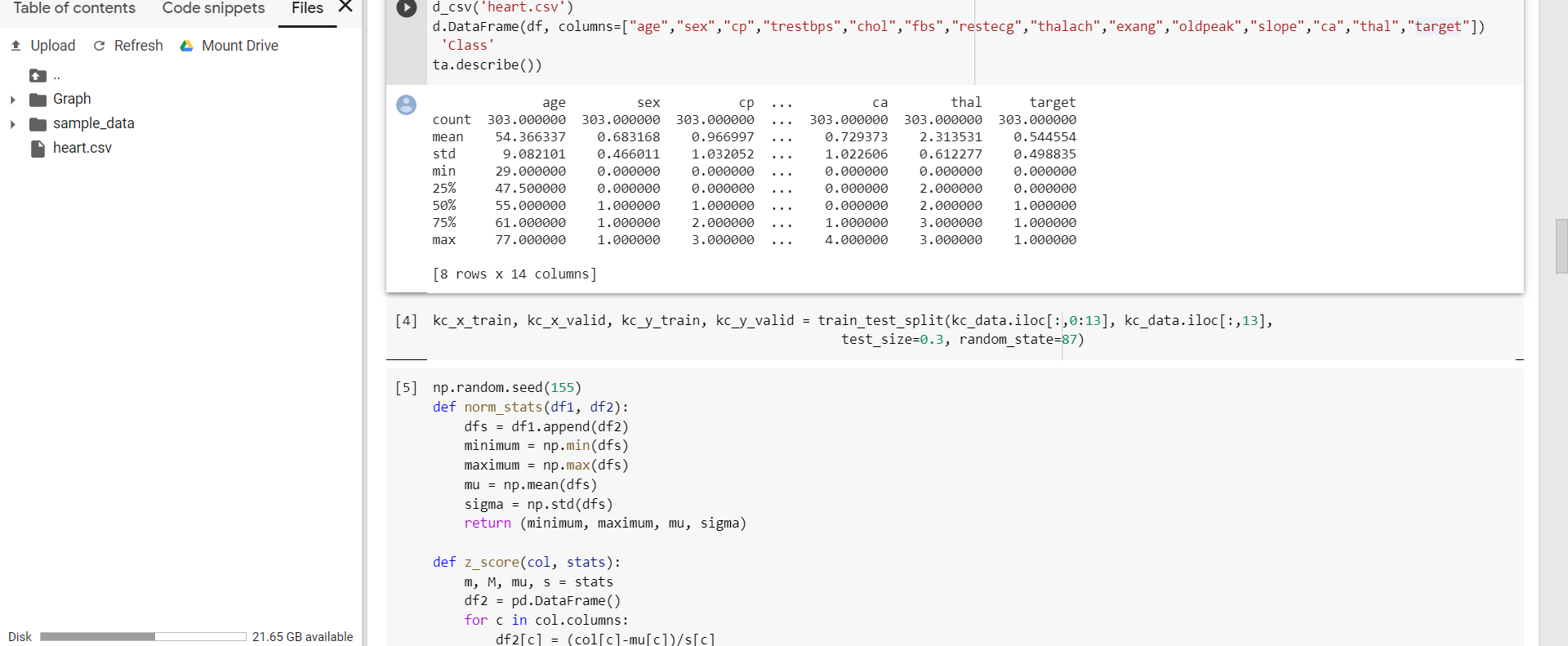
****

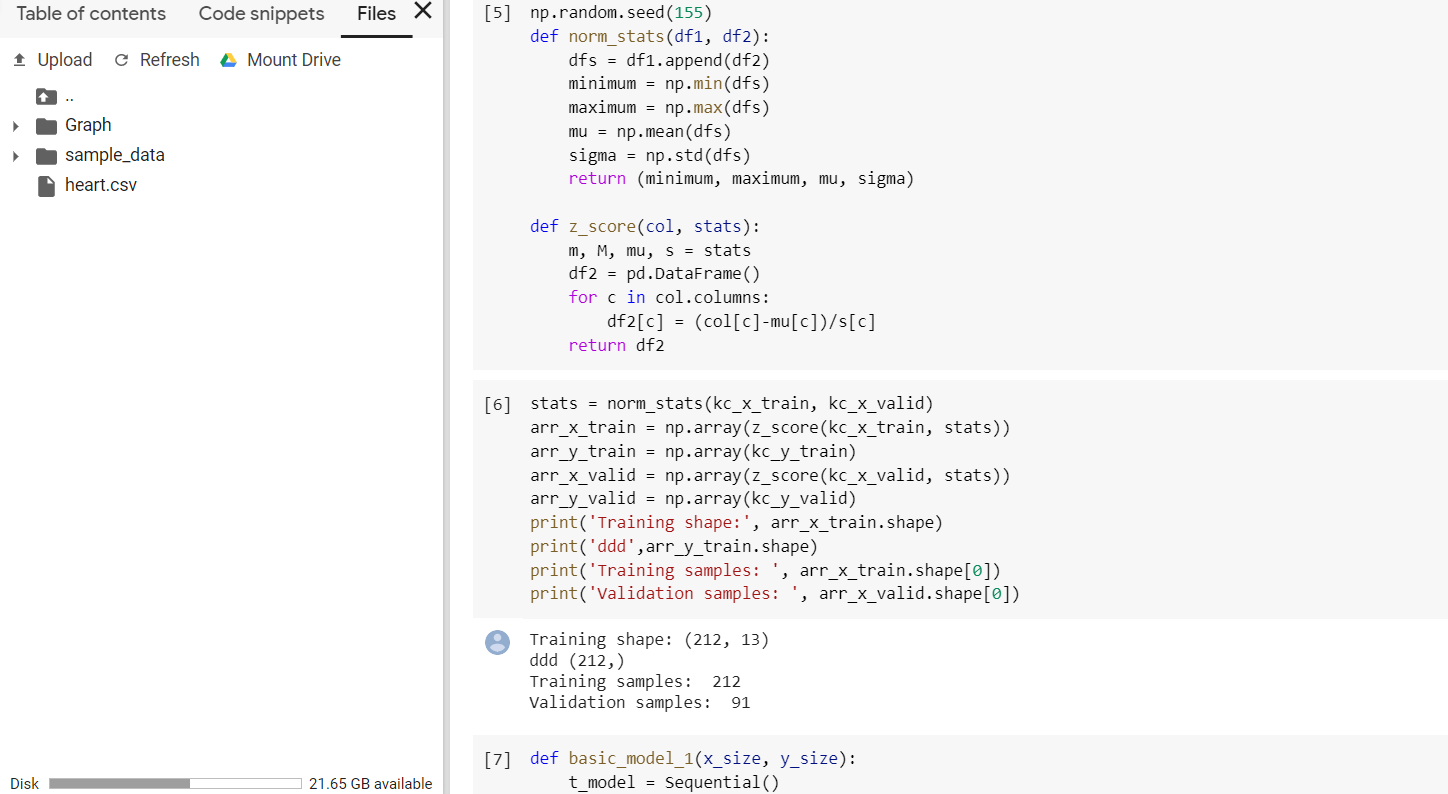
**Conclusion:**

Using the evaluation above we can see that the RMSprop gives the better training loss as compared to Adam optimizer and also Epoch and learning rate plays very significant role in same.

**Program-2**





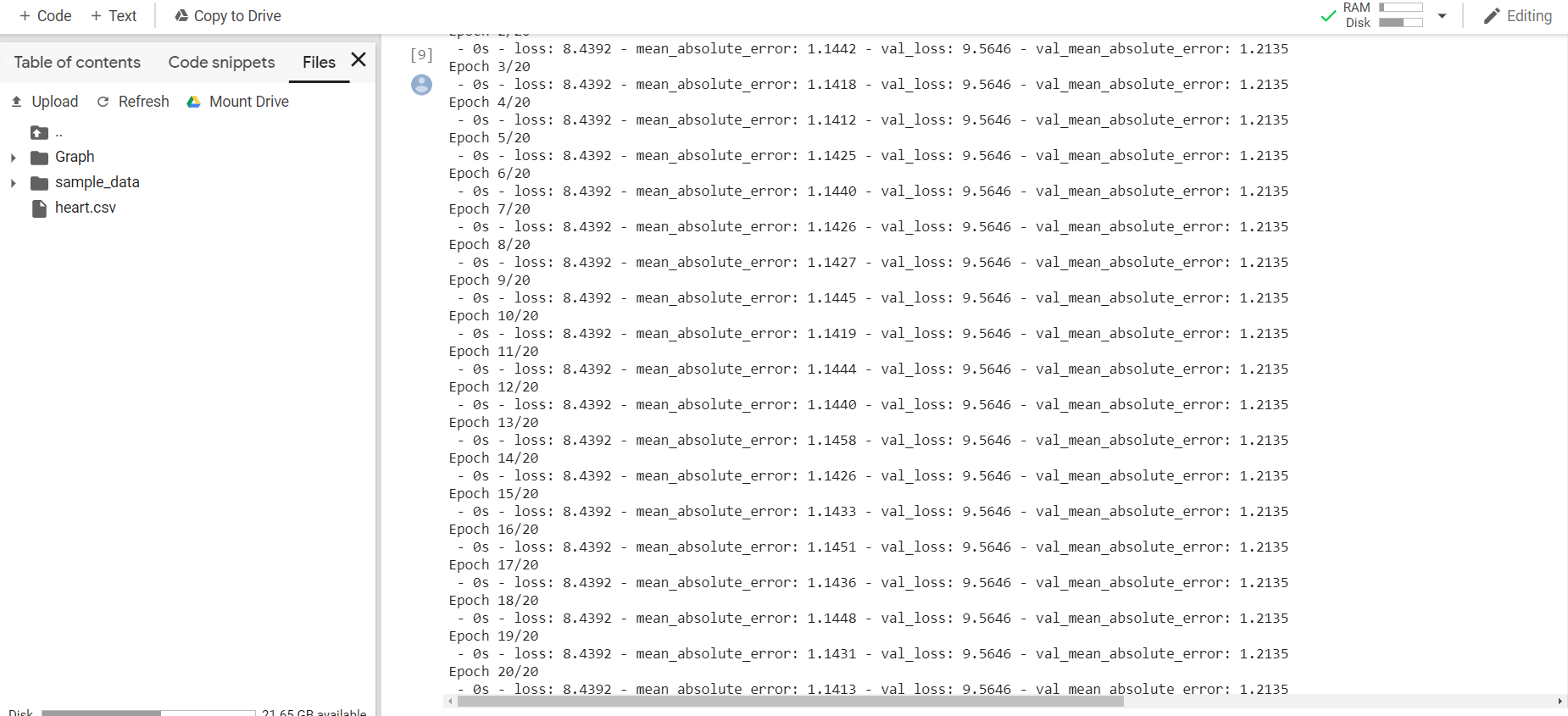


#The basic\_model\_2 is different from the basic\_model\_1 but performs the same task with different structure





# Changed verbose to 2 and observed execution.





# Prepared the plotting

# Summarized history for MAE

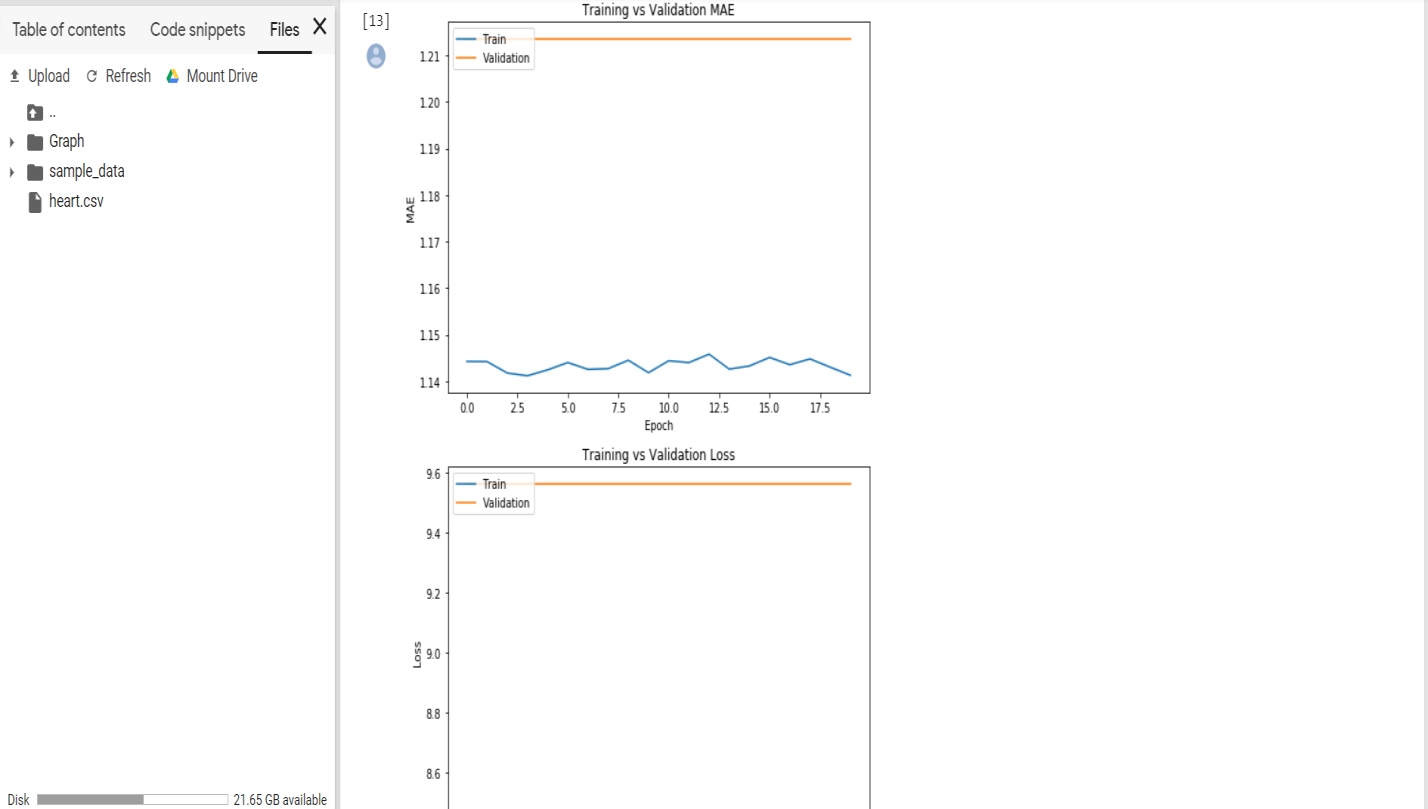
# Summarized history for loss

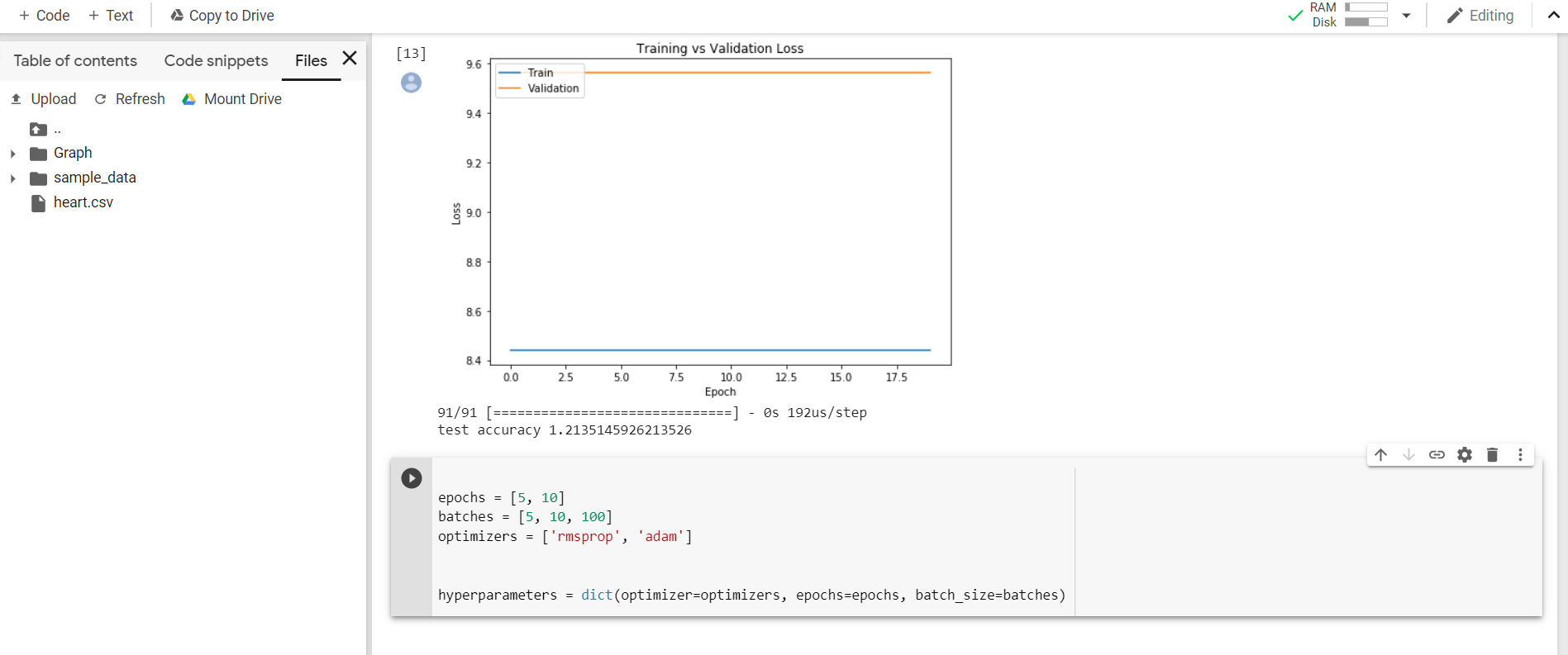
# Plotted it all in IPython (non-interactive)

# Created hyperparameter space

# Created hyperparameter options



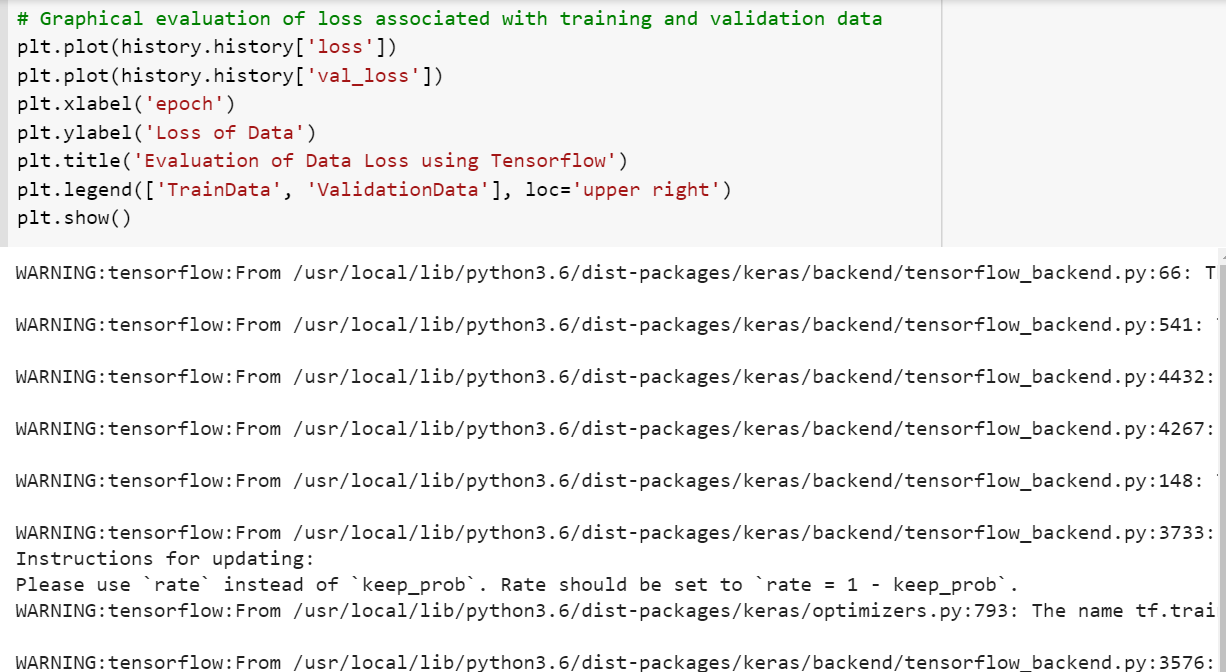
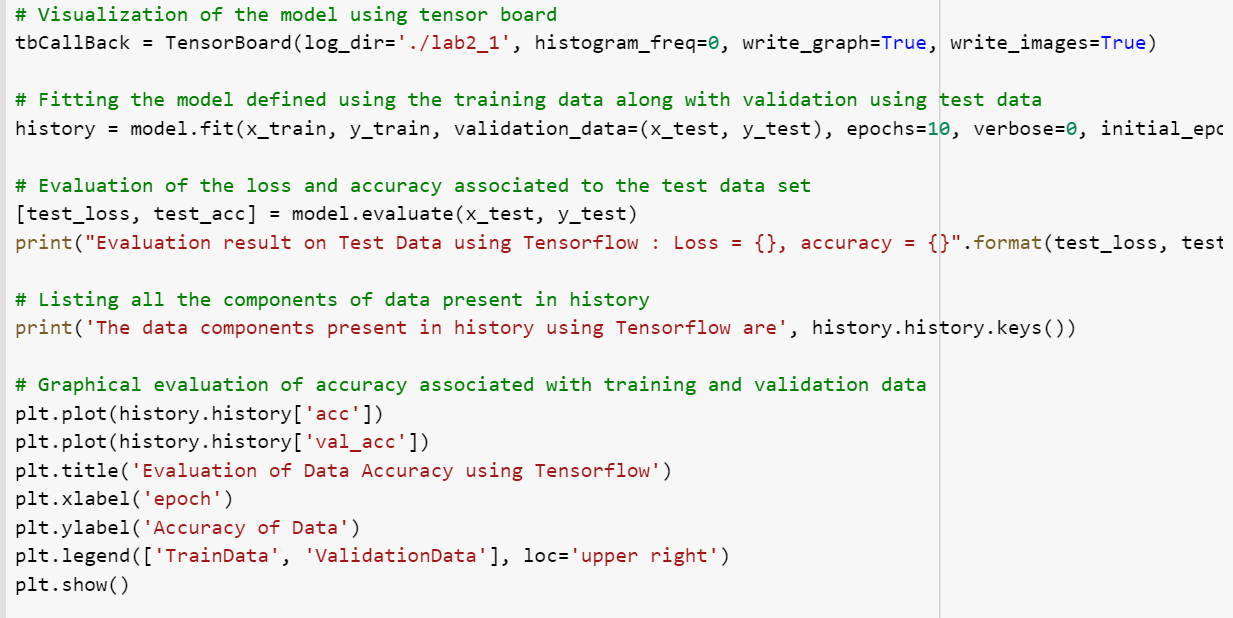
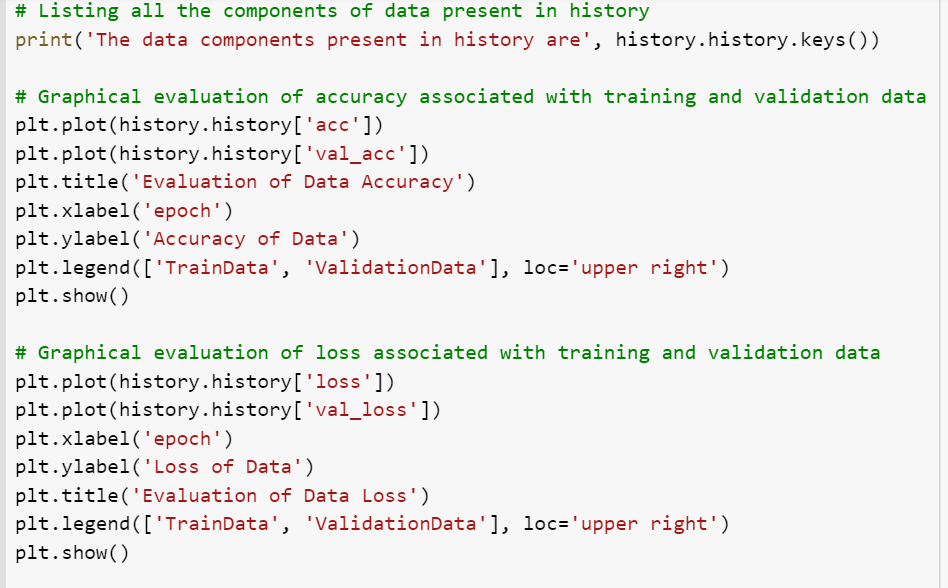
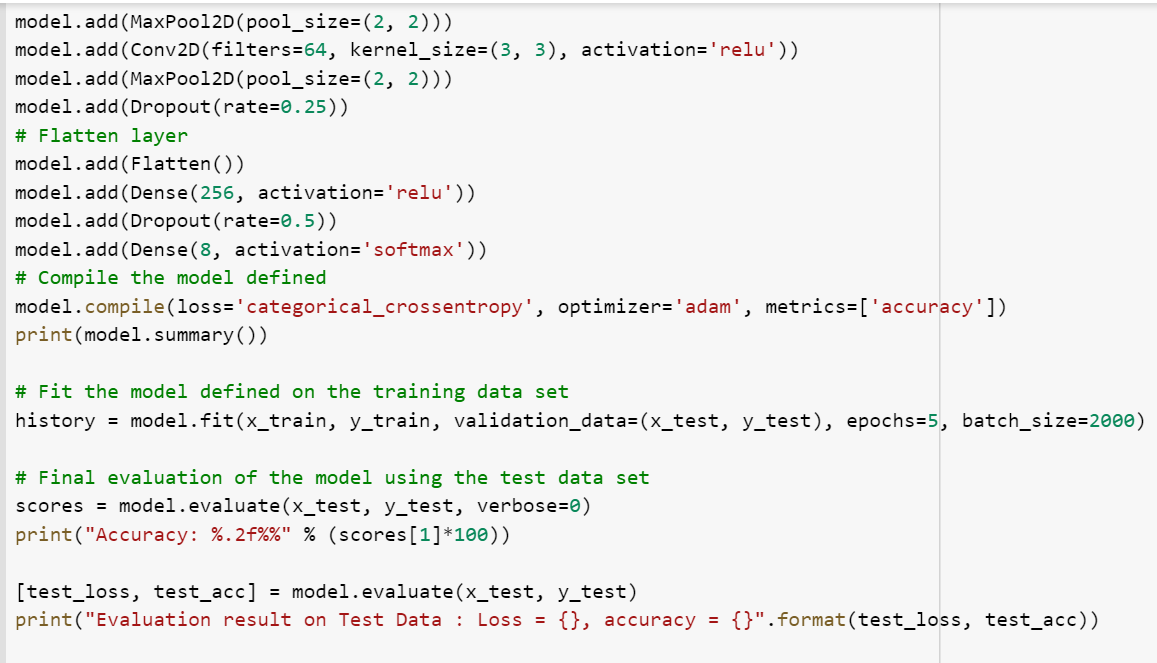
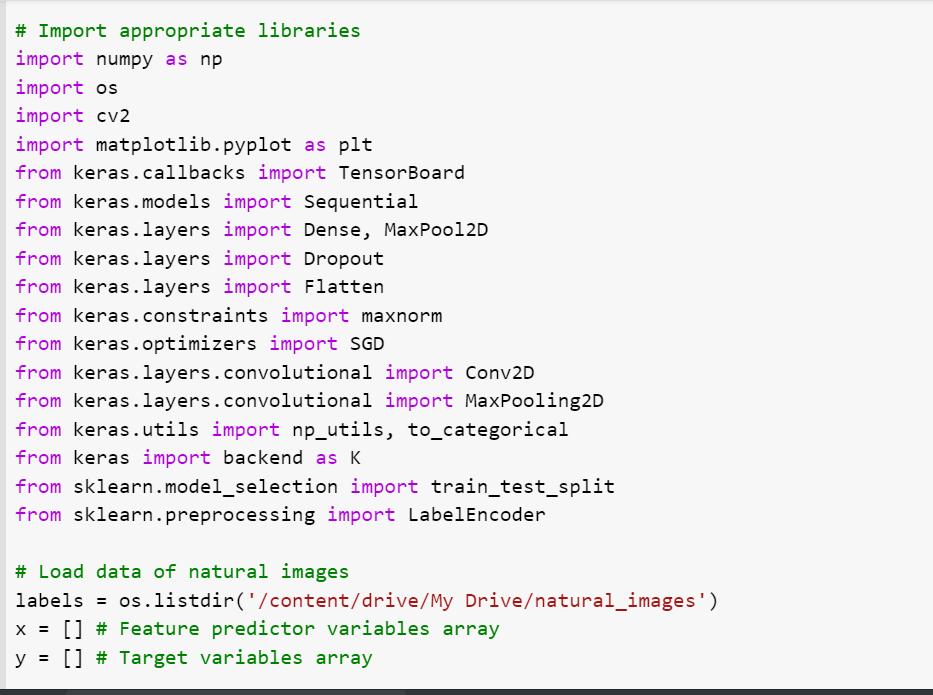




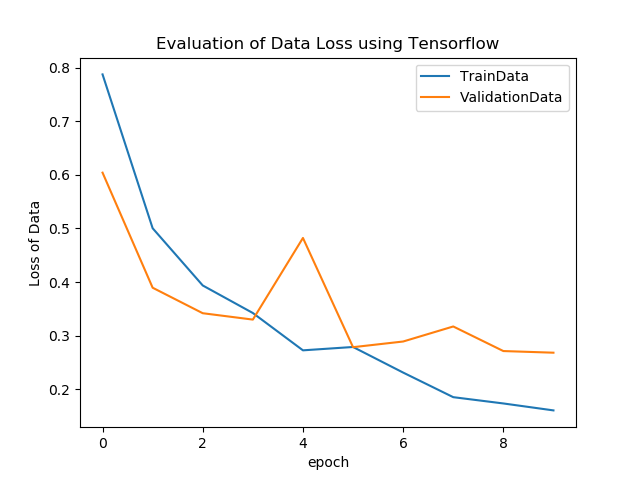
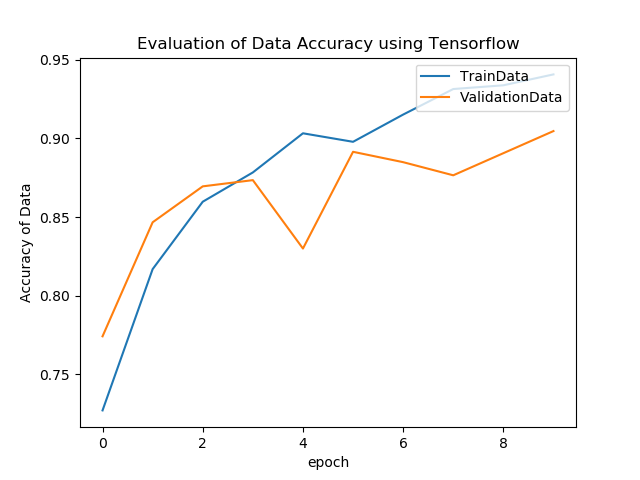
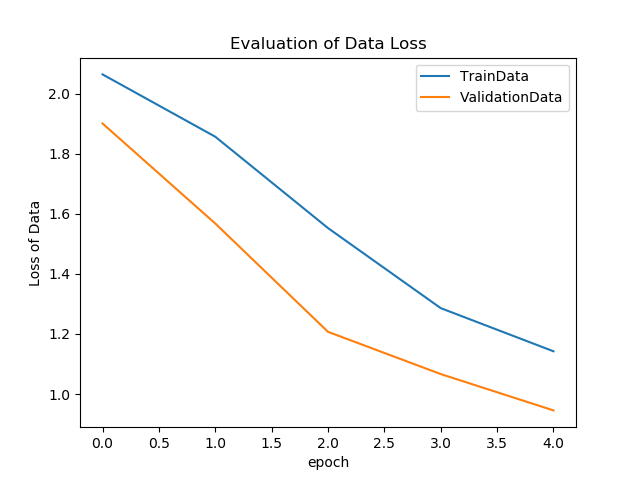
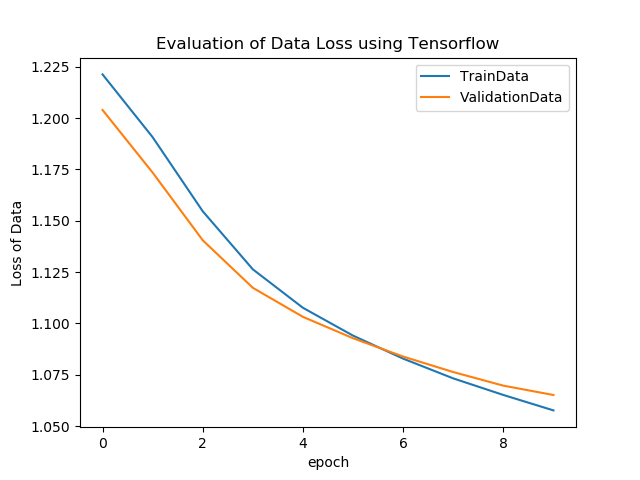
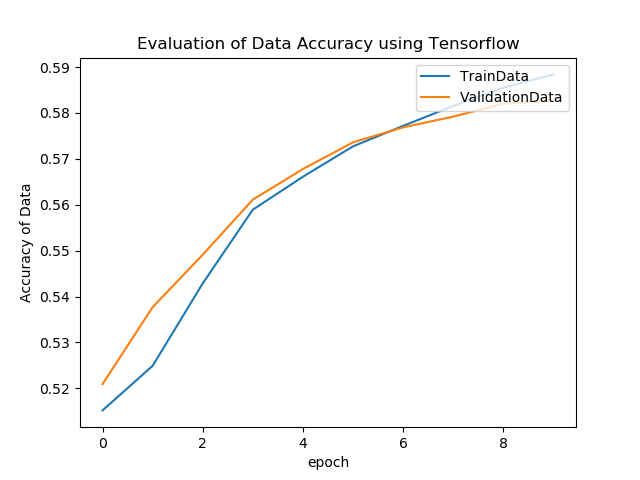
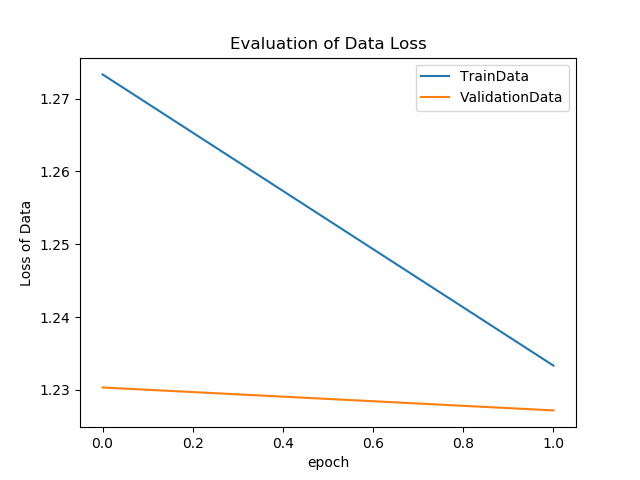
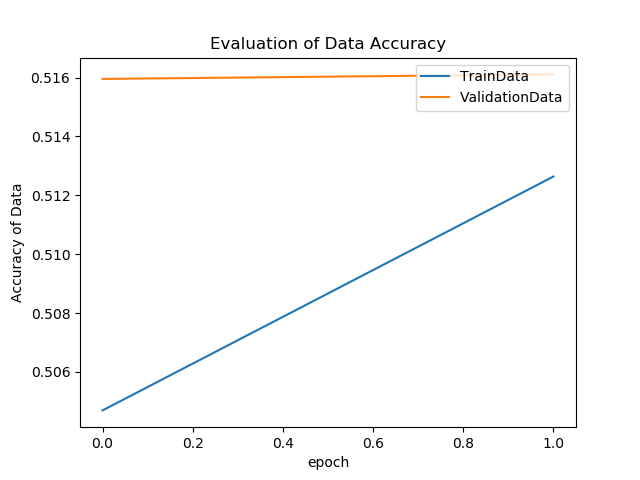
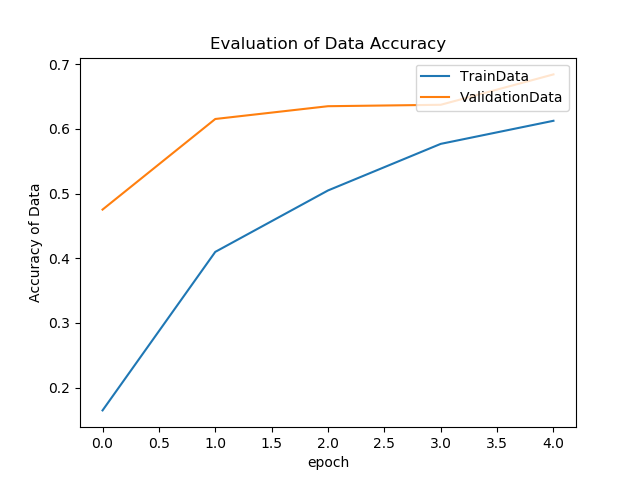
**Program-3**

Implement the image classification with CNN model on anyone of the following datasets

Code:

****

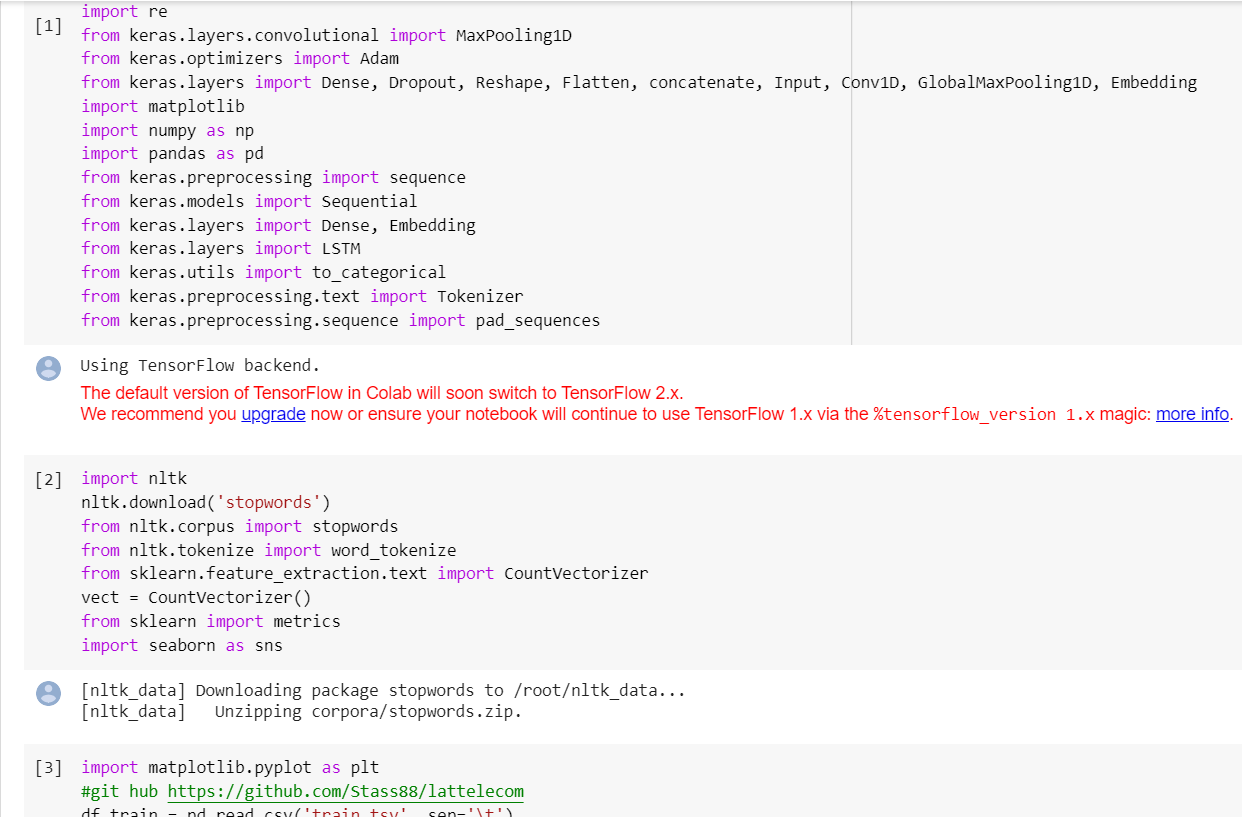
**Output:**

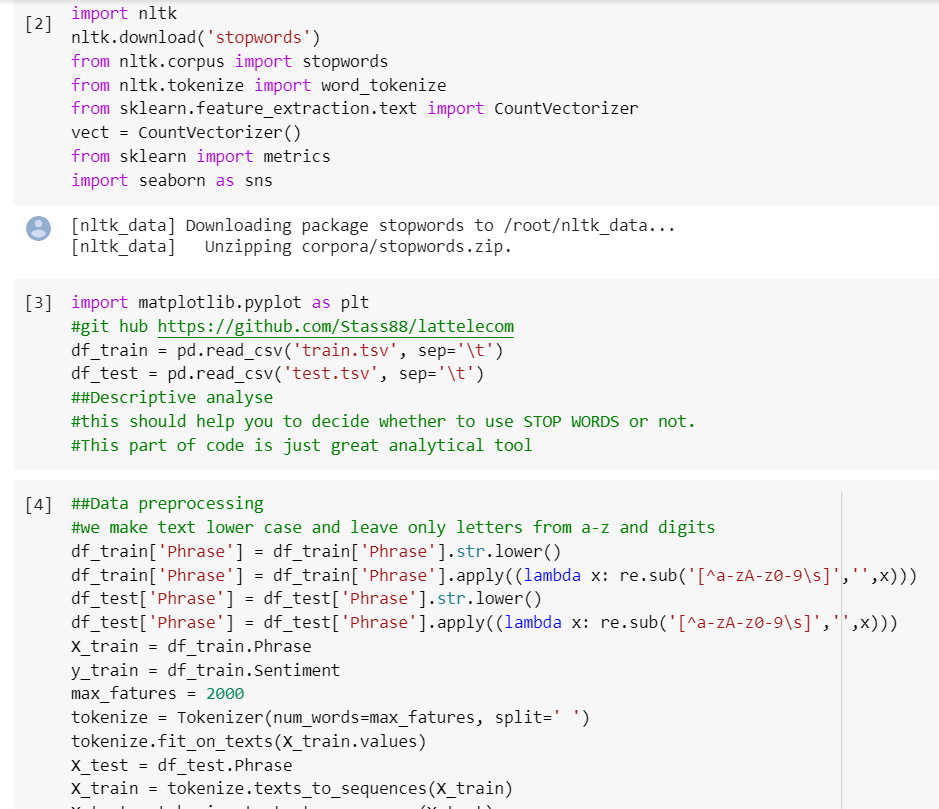
****

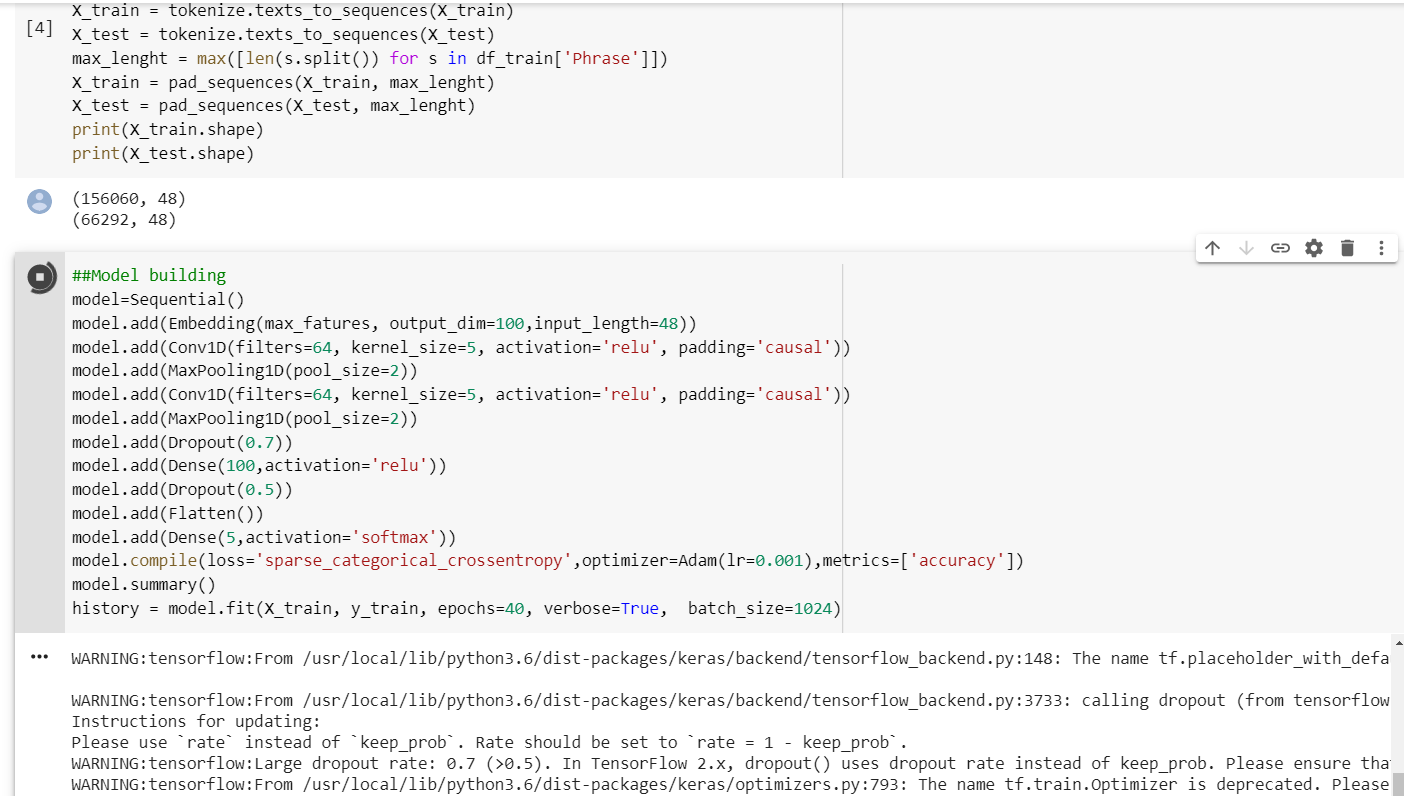
**Program-4**

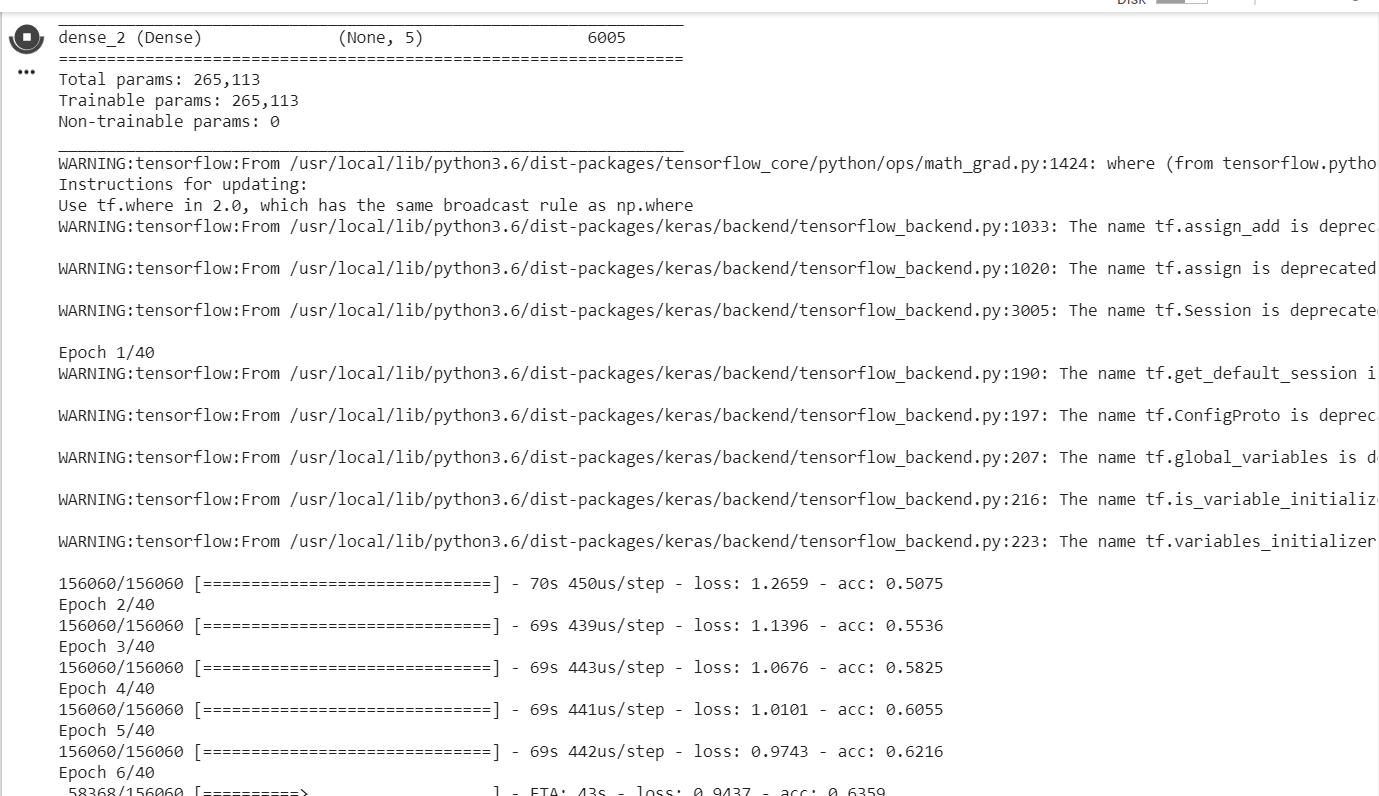
**Code:**

Implement the text classification with CNN model on the following movie reviews dataset

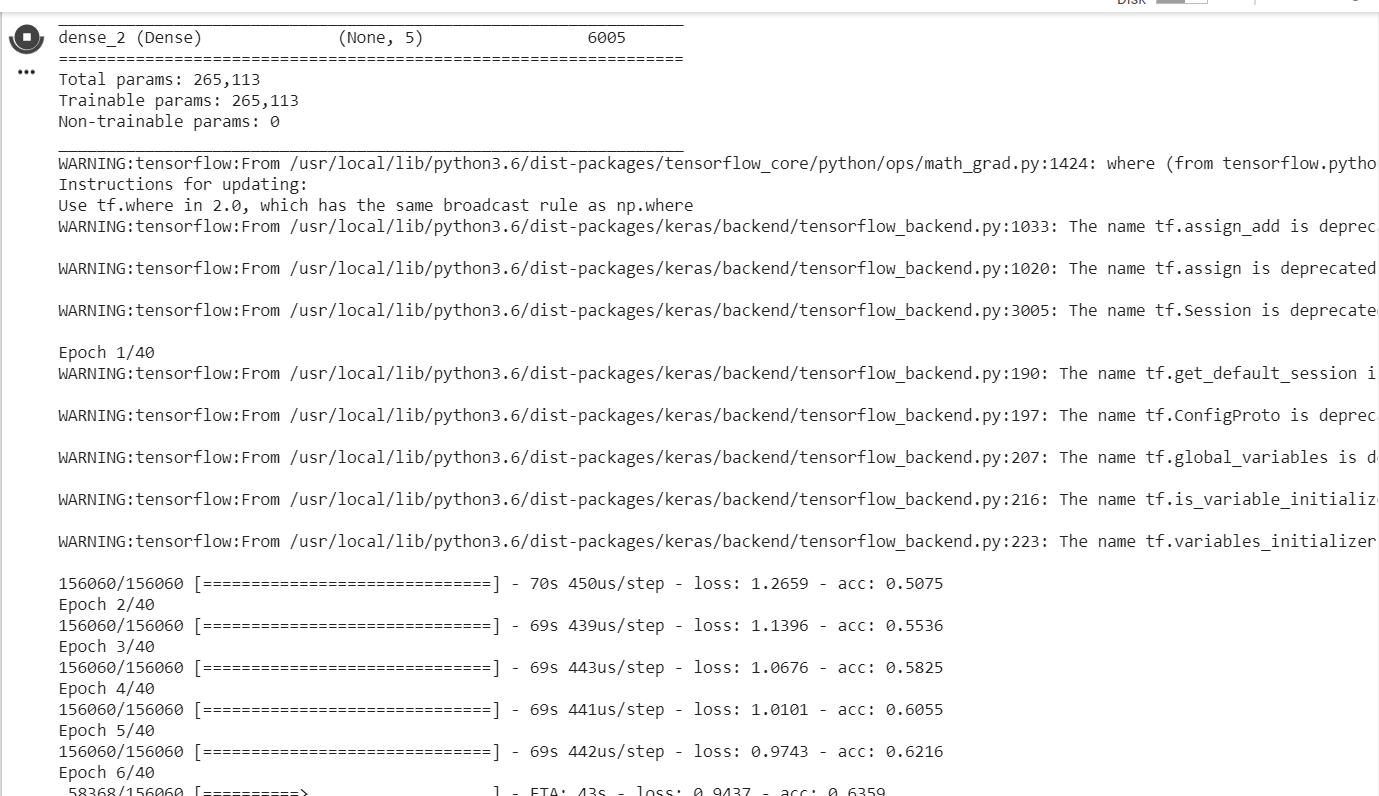
****

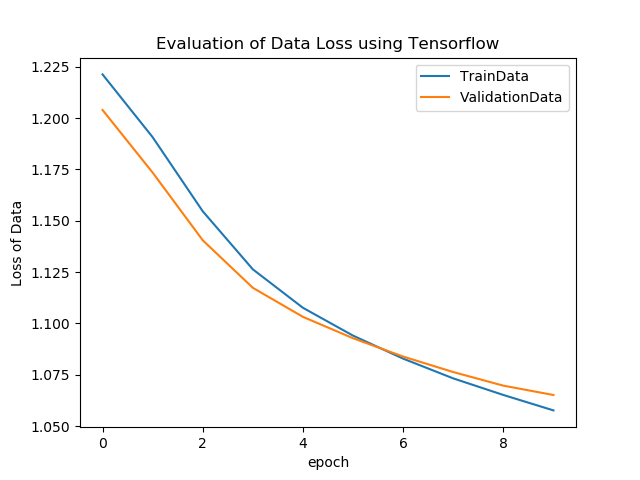
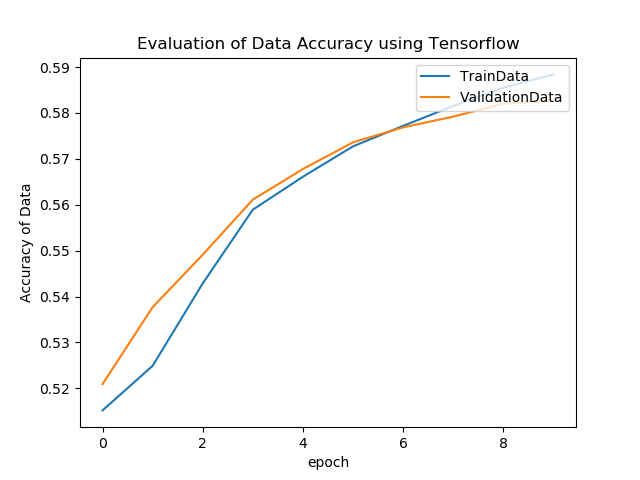
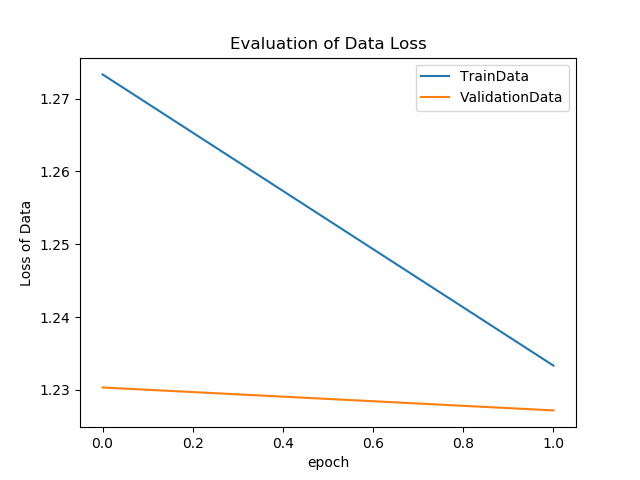
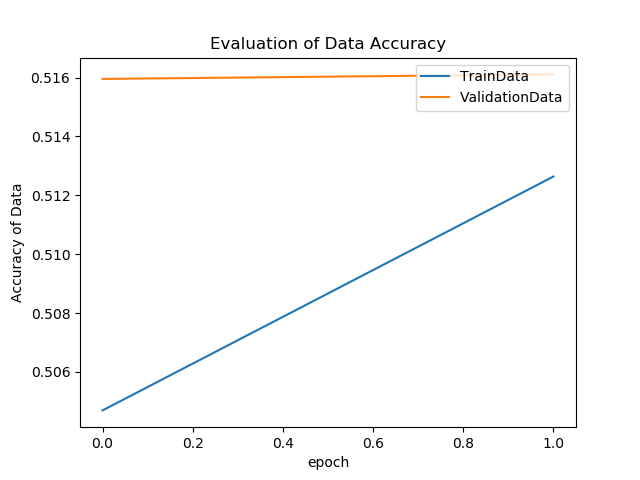
****

****

****

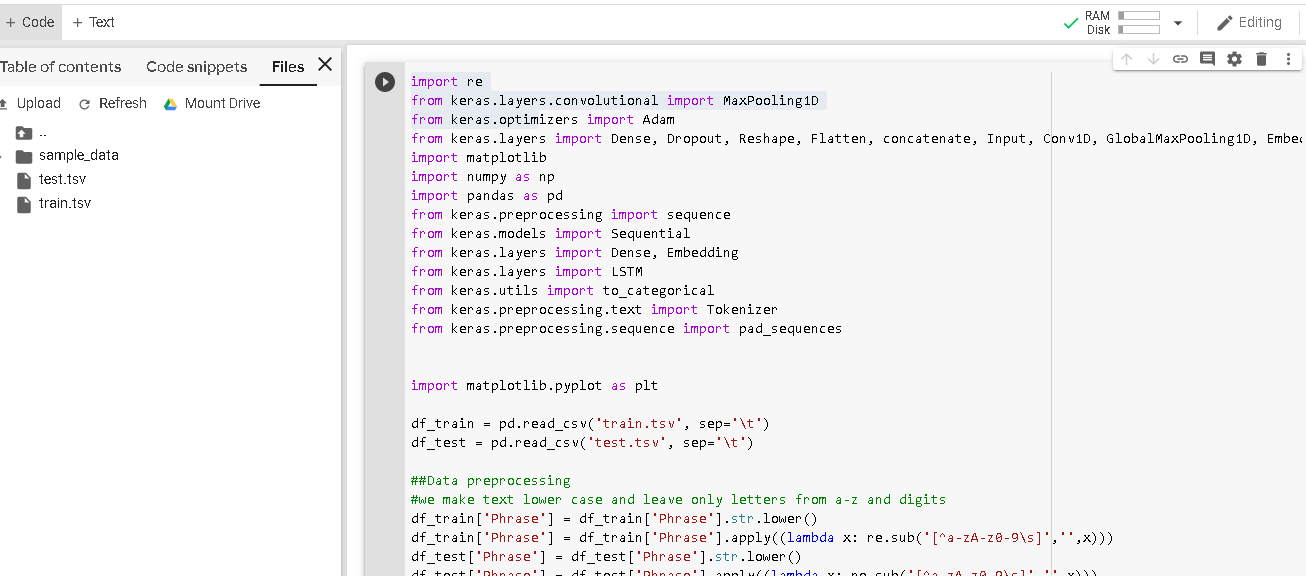
**Output**

****

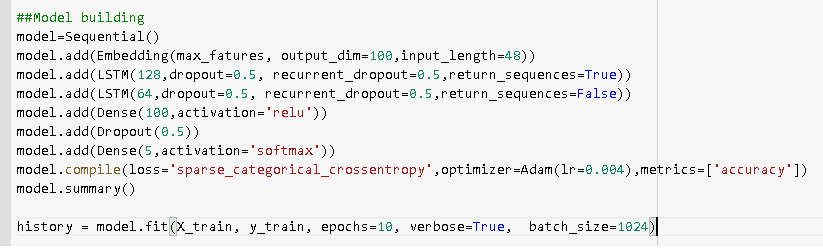
****

**Program-5**

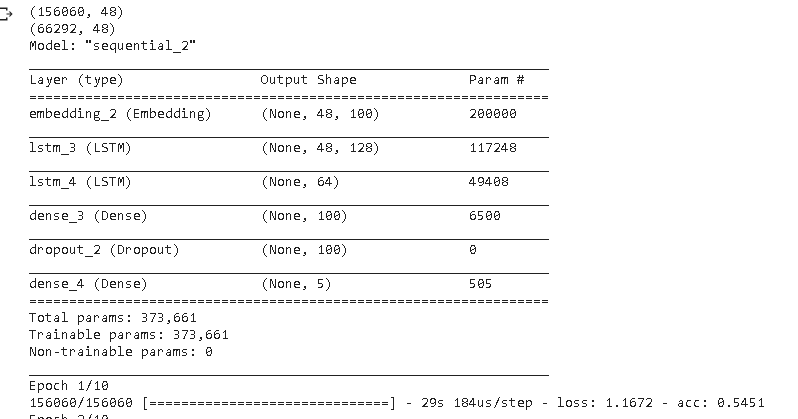
**Code**

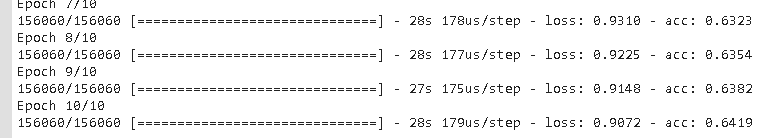
****

****

****

**Output:**

****

****

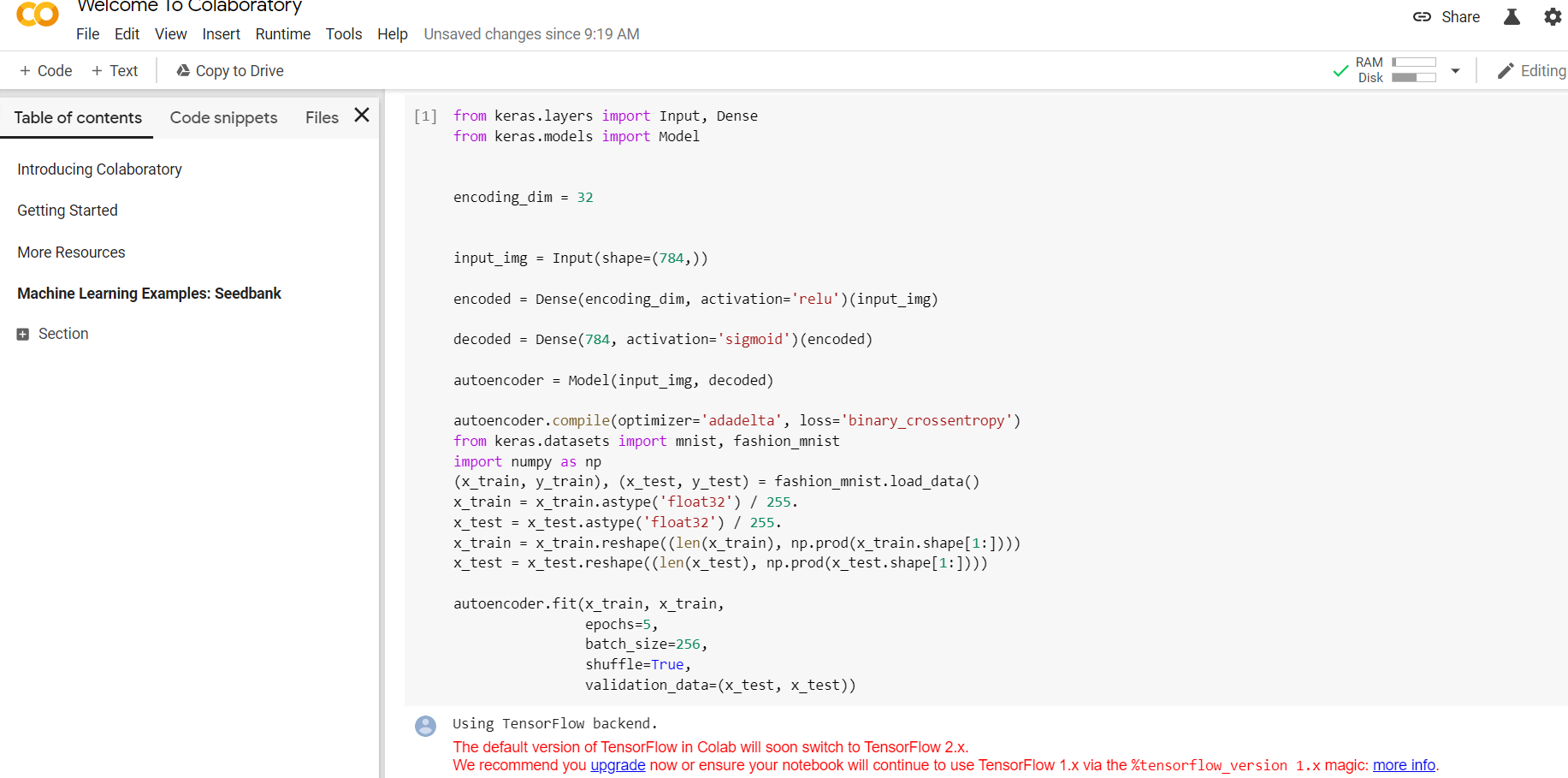
**Program-6**

|  |  |
| --- | --- |
| Model | Accuracy |
| CNN | 63 |
| LSTM | 65 |

The accuracy of LSTM model of text classification is higher than of CNN, this is because of cell state and recurrent dropout. Dropping low priority feature over random dropping is always essential. So **LSTM** model is **best** for text classification.

Increasing the Epoch value increased accuracy in both cases

**Program-7**



*# here is the size of our encoded representations*

encoding\_dim = 32

*# here is our input placeholder*

input\_img = Input(shape=(784,))

*# "encoded" is the encoded representation of the input*

encoded = Dense(encoding\_dim, activation='relu')(input\_img)

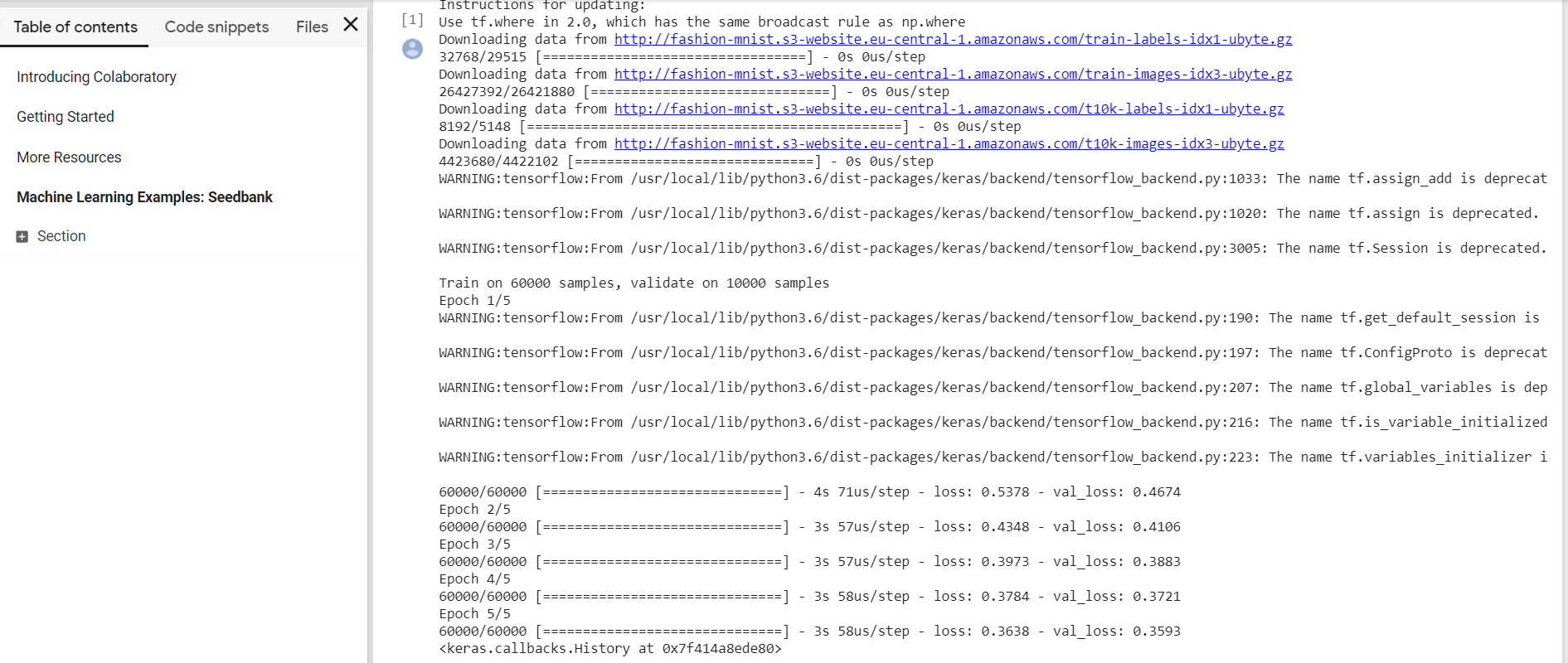
*# "decoded" is the lossy reconstruction of the input*

decoded = Dense(784, activation='sigmoid')(encoded)

*# the model below maps an input to its reconstruction*

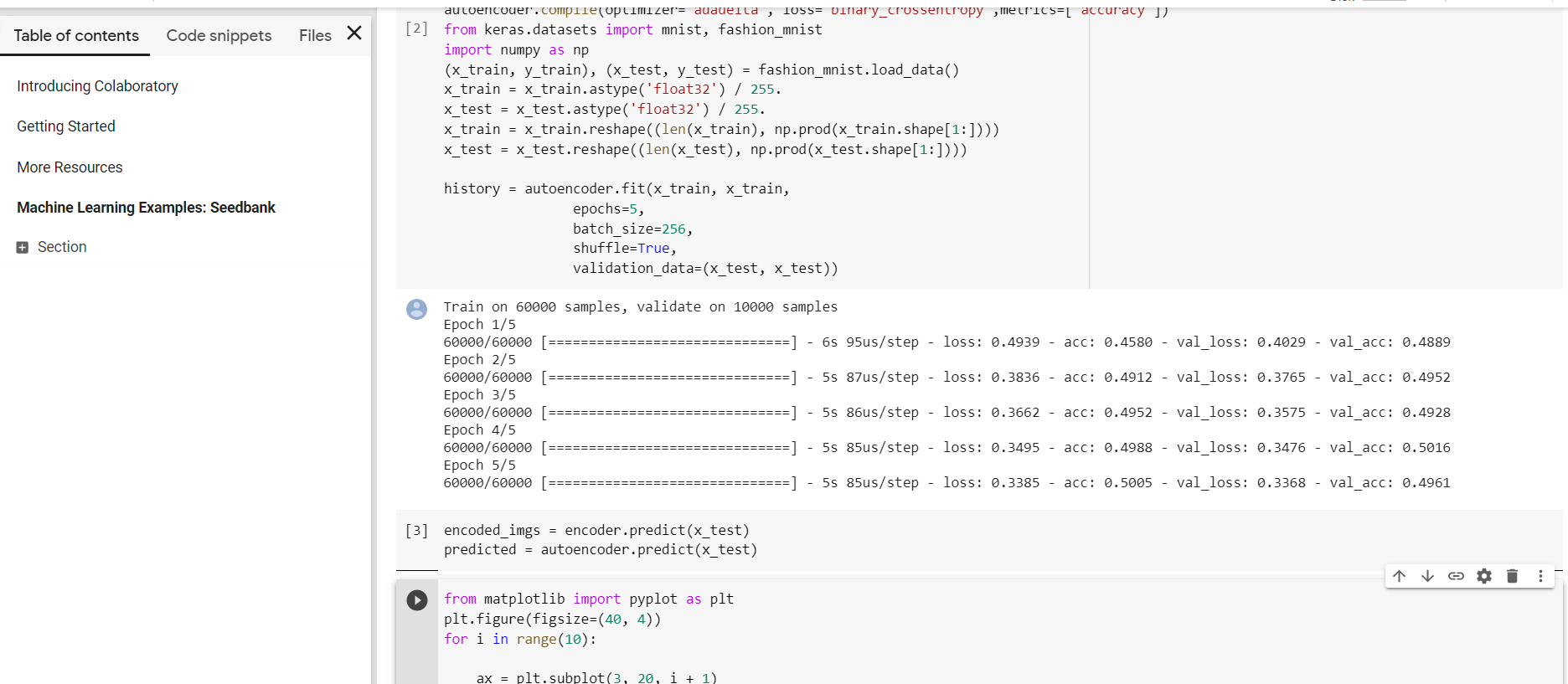
autoencoder = Model(input\_img, decoded)

*# this model maps an input to its encoded representation*





*#Addition of one hidden layer*





*# Display of original images*

ax = plt.subplot(3, 20, i + 1)

plt.imshow(x\_test[i].reshape(28, 28))

plt.gray()

ax.get\_xaxis().set\_visible(**False**)

ax.get\_yaxis().set\_visible(**False**)

*# Display of the encoded images*

ax = plt.subplot(3, 20, i + 1 + 20)

plt.imshow(encoded\_imgs[i].reshape(8,4))

plt.gray()

ax.get\_xaxis().set\_visible(**False**)

ax.get\_yaxis().set\_visible(**False**)

*# Display of the reconstructed images*



Visualization using Matplotlib