**CS5590 APLS- DEEP LEARINING PROGRAMMING**

**LAB 2**

**BY,**

**TEAM 6**

**KOLLURI, NIKHITHA: 12**

**INAKOLLU, SRI NAGA BHUVANESHWARI: 9**

**DUKKIPATI, SRI SAI NITHIN CHOWDARY: 4**

1. **Introduction:**

This lab has programs related to the concepts Word Embeddings, Loss functions, Optimizers, Convolution Neural Networks, LSTM, Implementing Text Generator, Autoencoder, Dimension reduction on the image, TensorBoard and TensorFlow graphs and sessions.

1. **Objectives**

To implement a few programs that cover all the above mentioned concepts. If necessary compare and contrast two methods and provide the result based on the solutions obtained.

1. Build a linear Regression Model using Sequential model, show graph on tensorboard and change few parameters and plot the loss.
2. Implement logistic regression, normalize the data, plot the loss on tensorboard, change any three hyperparameters and report the accuracy.
3. Image Classification using CNN.
4. Text Classification using CNN.
5. Text Classification using LSTM.
6. Compare and report which model is best for Text Classification by tuning the hyper parameters.
7. Encoding and decoding using Autoencoders on a particular image.
8. **Datasets Used**

* California\_housing
* Heart.csv
* Natural\_images
* Movie Reviews
* Mnist

1. **Approaches/Methods:**

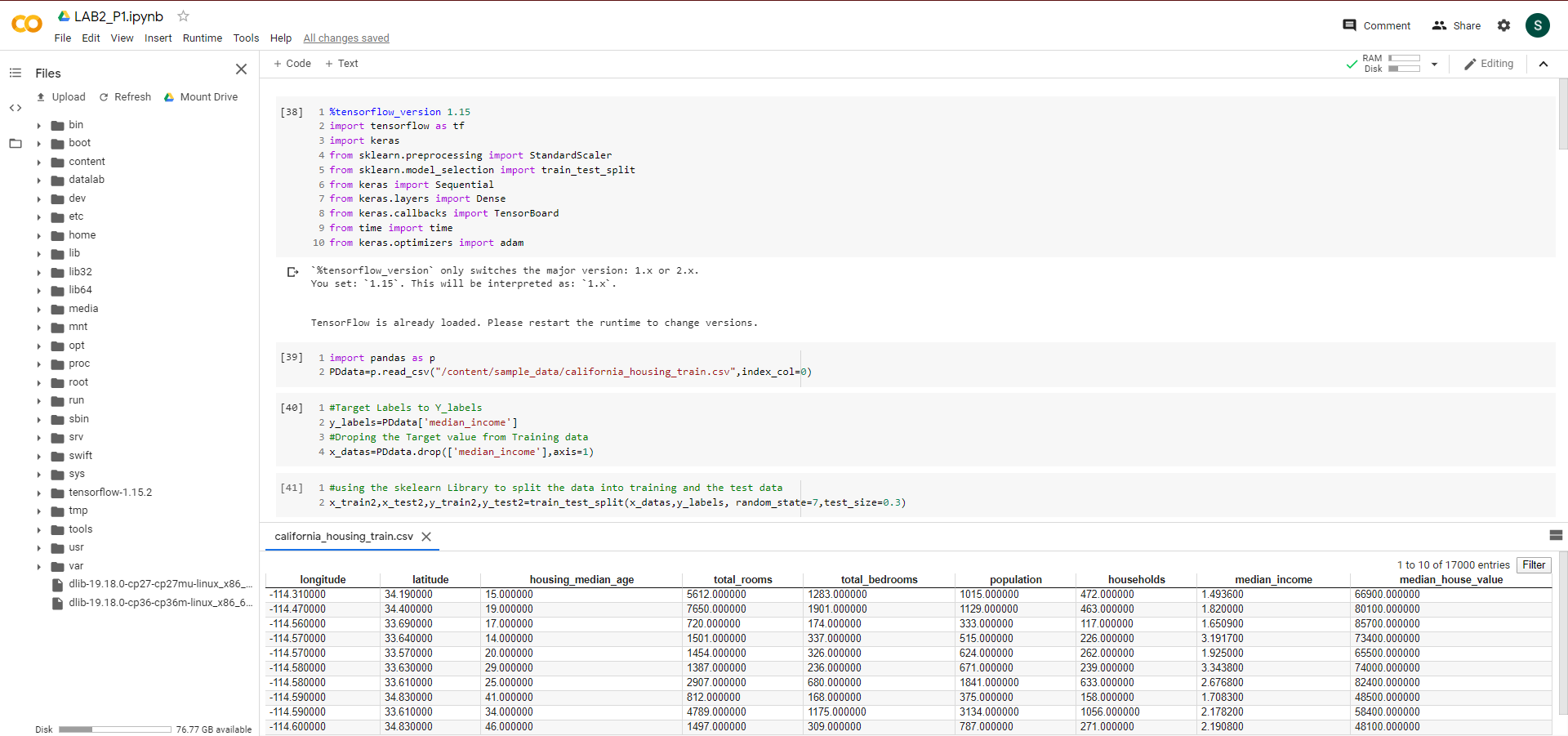
**1. LINEAR REGRESSION**

**ANS:**

We have used California housing data set from google colab sample dataset which was not used in any of our ICP

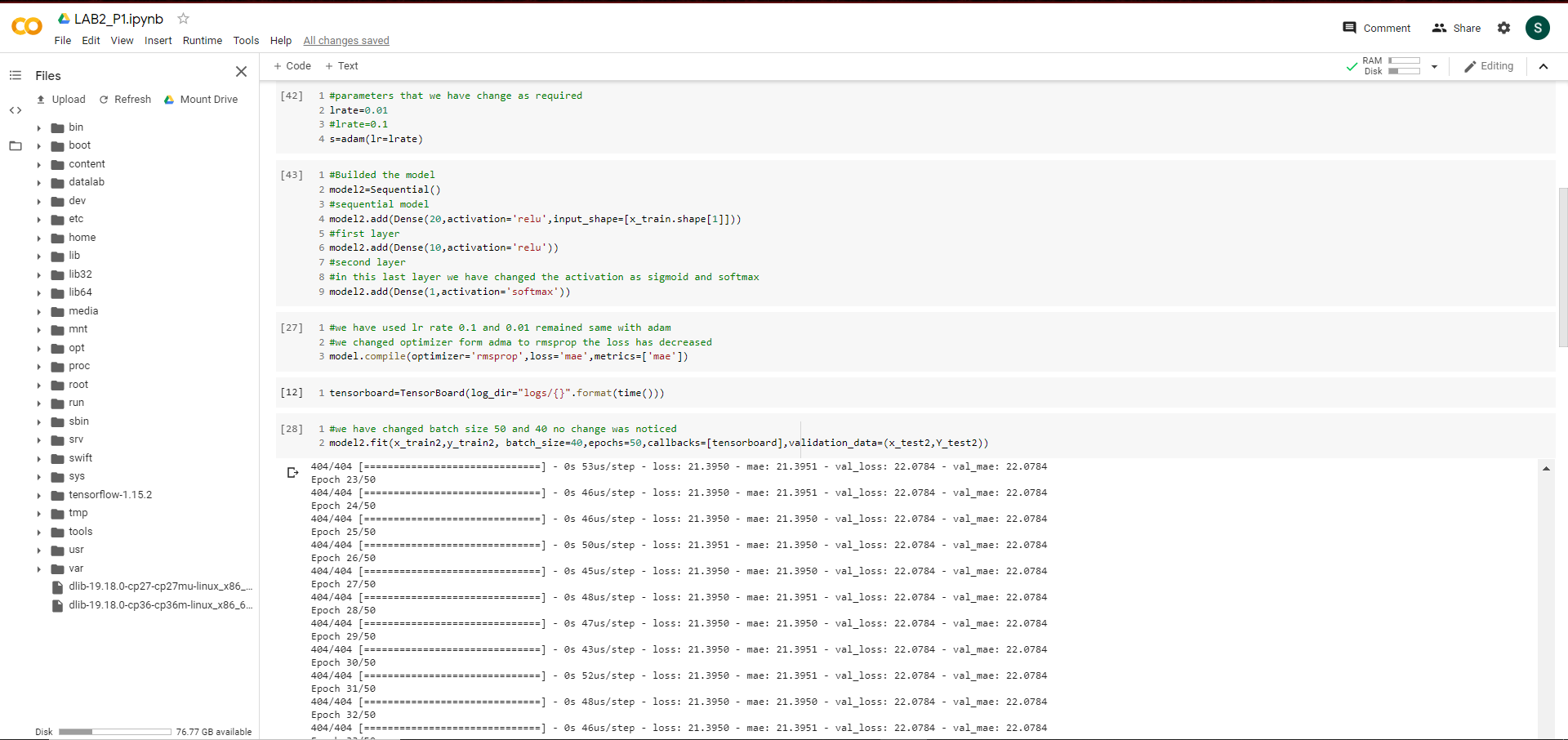
STEP 1:

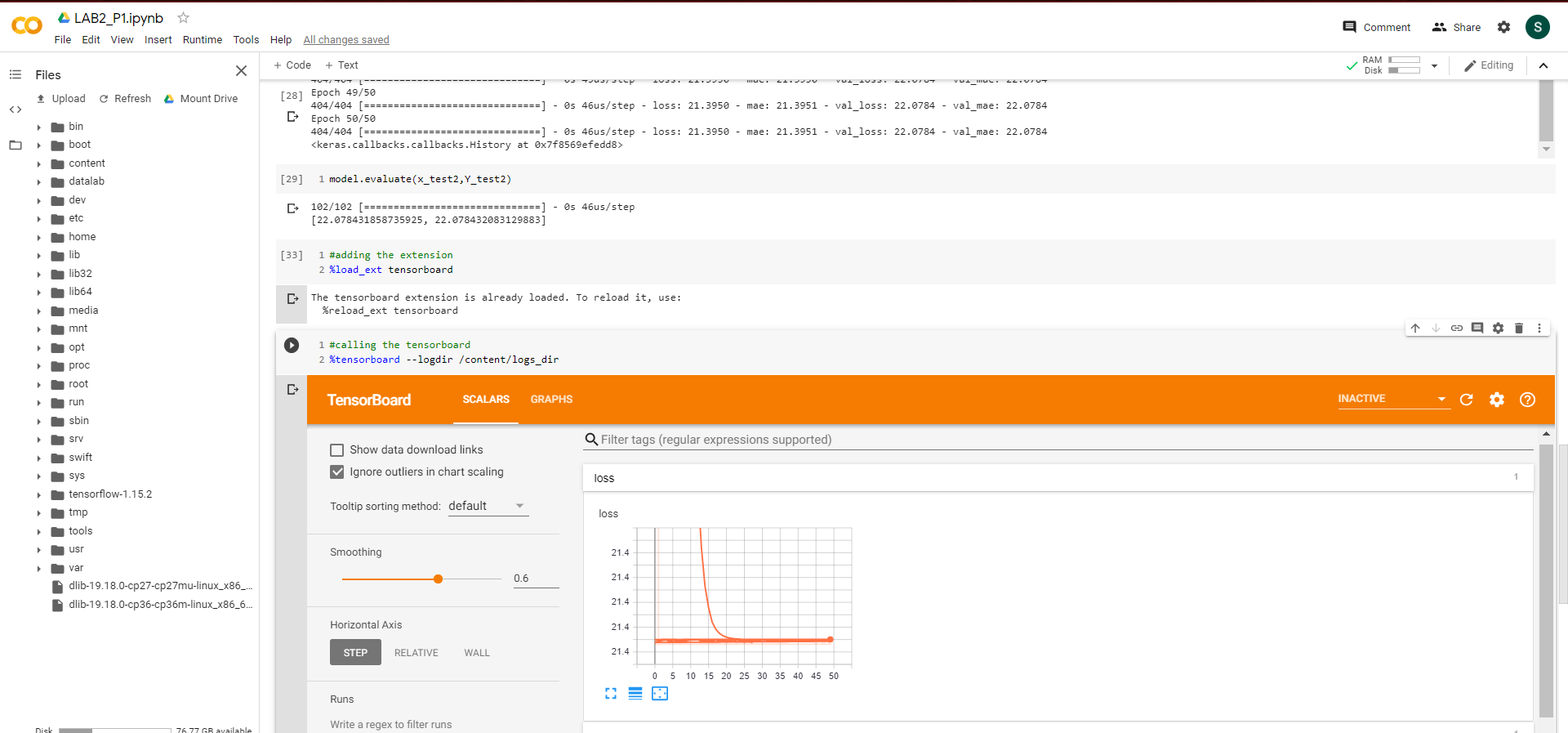
In the first step we have imported all the libraries that are required, after that we have loaded the data using given csv file in colab sample data set using pandas data frame. We have used train\_test\_split from sklearn for divide our test and training data and also shuffle our data.



STEP 2:

We have set some parameter for best fitting our model and in the next step we have builded our model, the model that we have used is Sequential model. We have used 3layers in our model with 1 hidden layer. We have used Tensor Board for logging our data and display what changes we got when we have changed given hyper perameters.

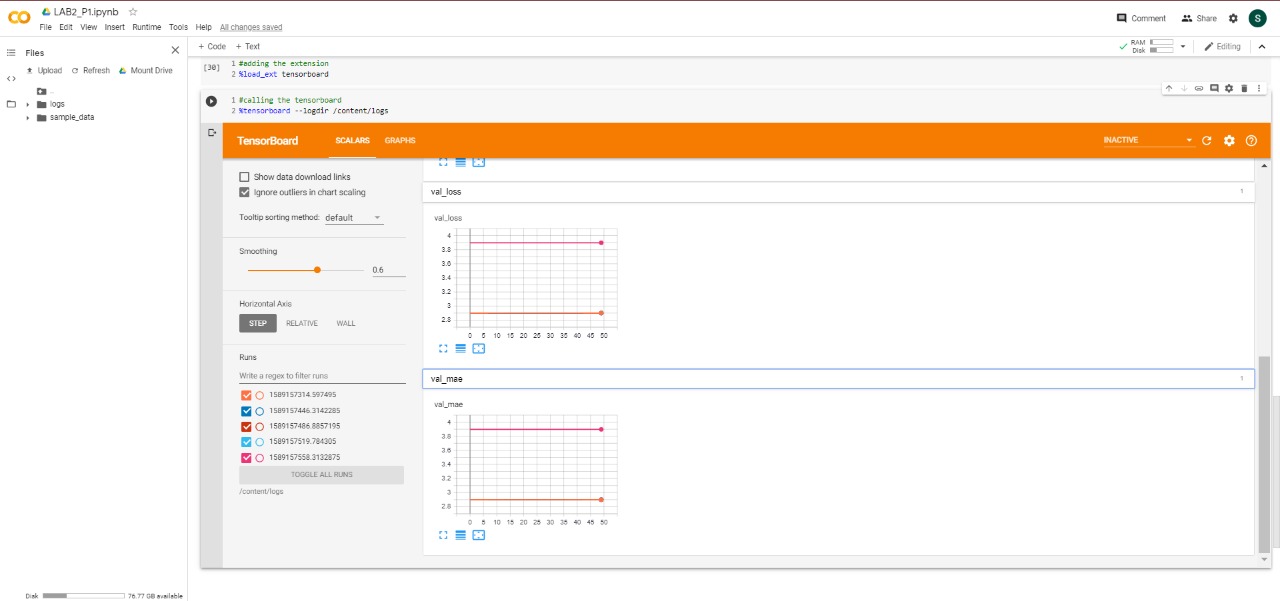


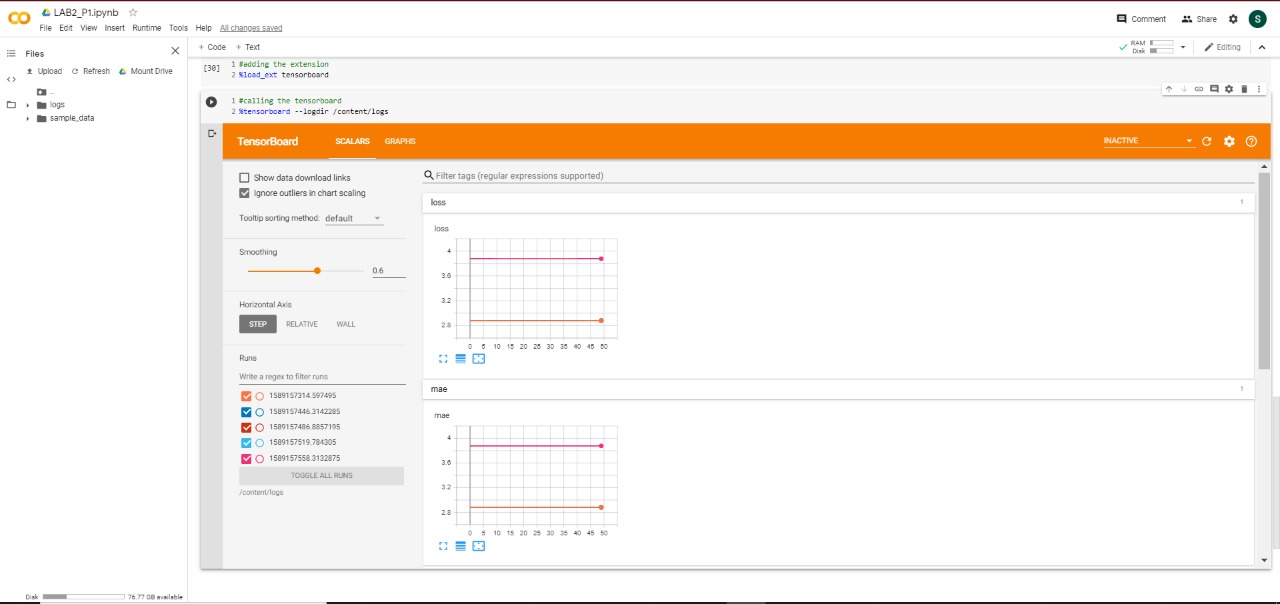


We have logged our data with different parameters,

**1a. Plot on Tensor Board**

**Ans:**





**1b. Report changes**

**Ans:**

1. learning rate
2. batch size
3. optimizer
4. activation function

**Learning Rate:**

We have used learning rate 0.1 and 0.01. for 0.1 the loss comparatively high when compared to 0.01 learning rate.

**Batch Size:**

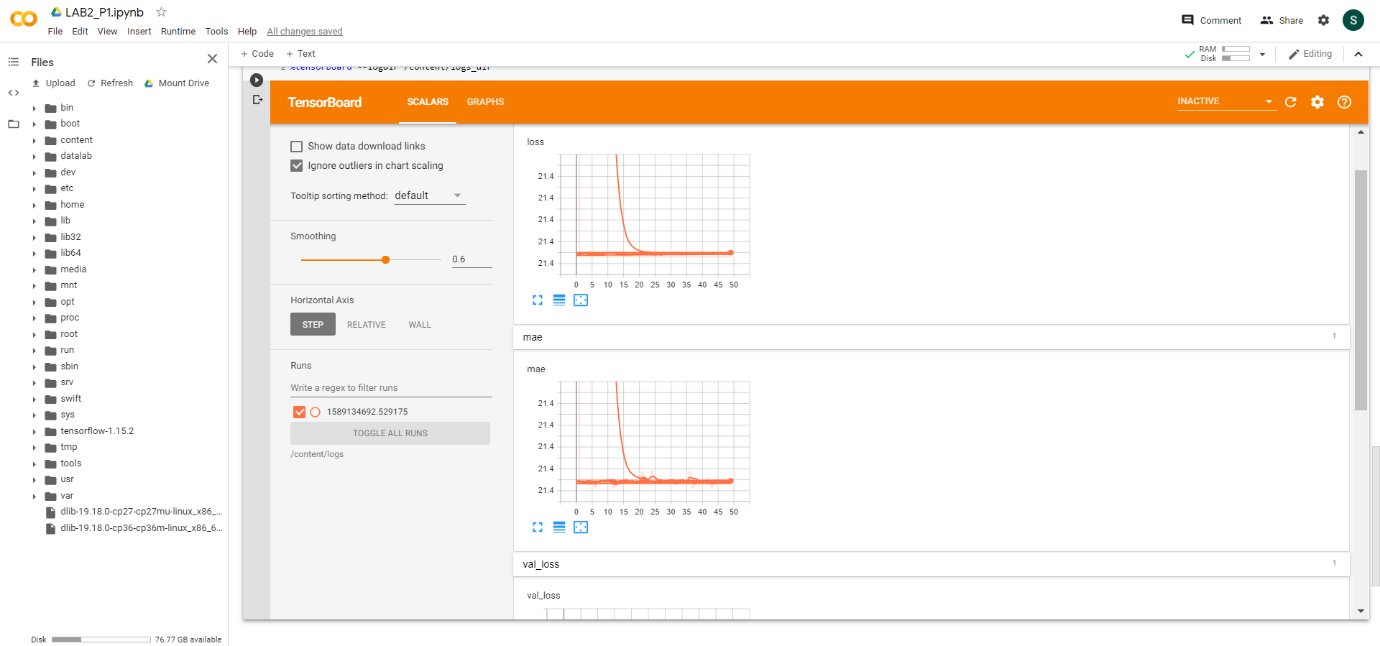
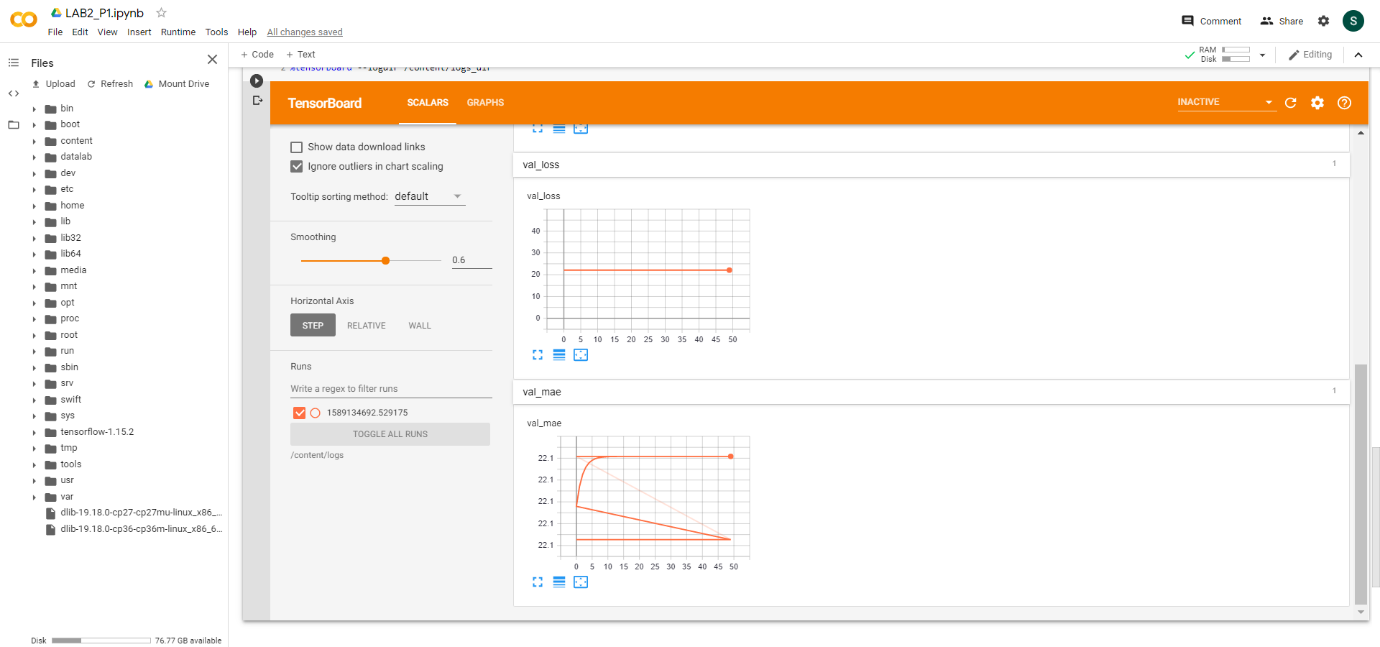
We have used batch size 40 and 50 and there was a minimal changes in loss and mae.

**Optimizer:**

We have used and Adam and rmsprop when we compared both rmsprop has given less loss compared to Adam in our case.

**Activation Function:**

We have used sigmoid and SoftMax as activation function of our last layer when we compared the results that loss was less when we have using sigmoid as last layer activation function.



we have logged our results for every change and plotted on tensor board that is the reason there are many lines plotted on graph.

**2. Logistic Regression on heart disease data**

**ANS:**

STEP 1:

Import all libraries that are required. Tensor Flow, Keras, Sklearn, time, pandas.

STEP 2:

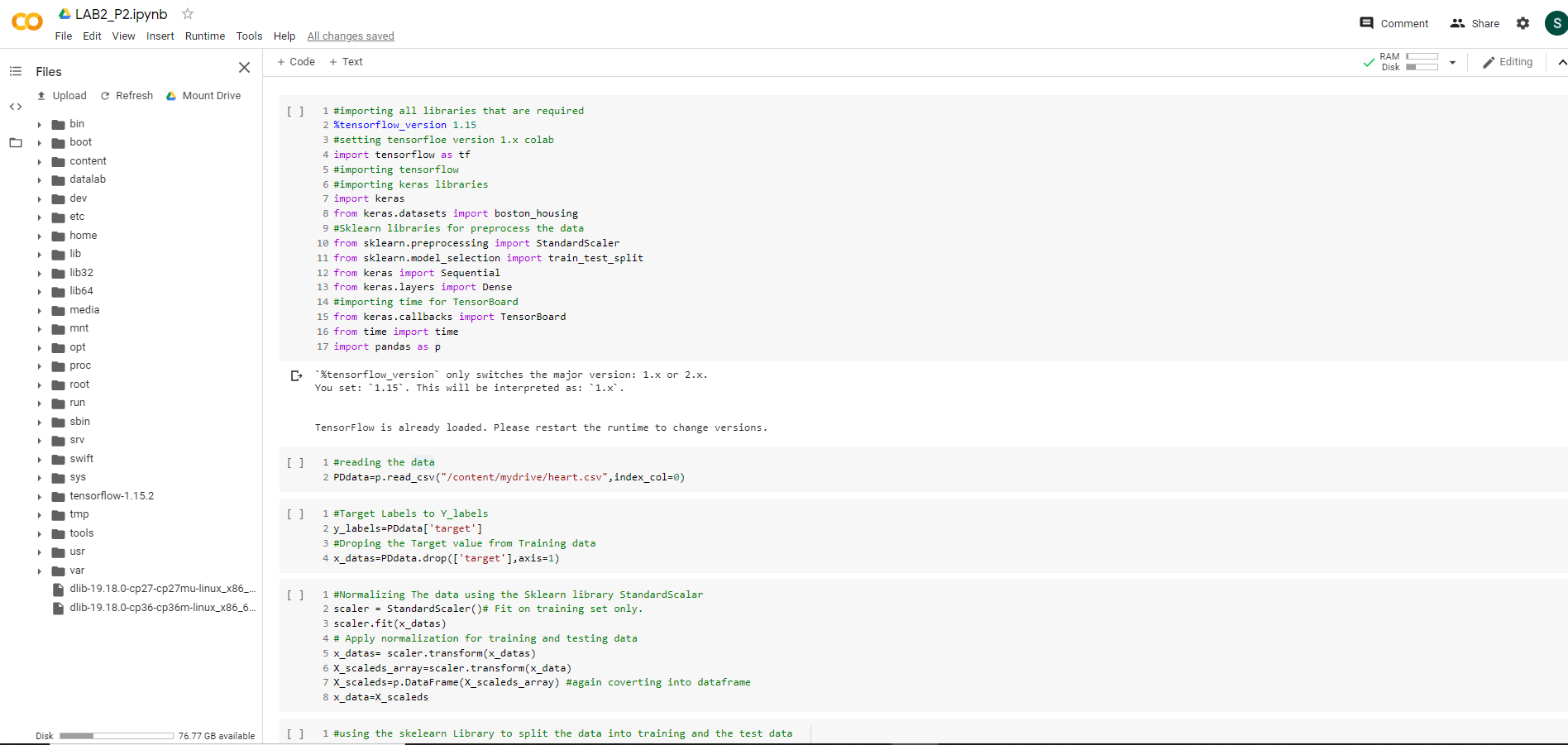
Read the data from heart.csv and we split have split the data and target variable.

**2a. Normalise the data**

**ANS:**

STEP 3:

We have normalized the data using sklearn library Standard Scalar.



STEP 4:

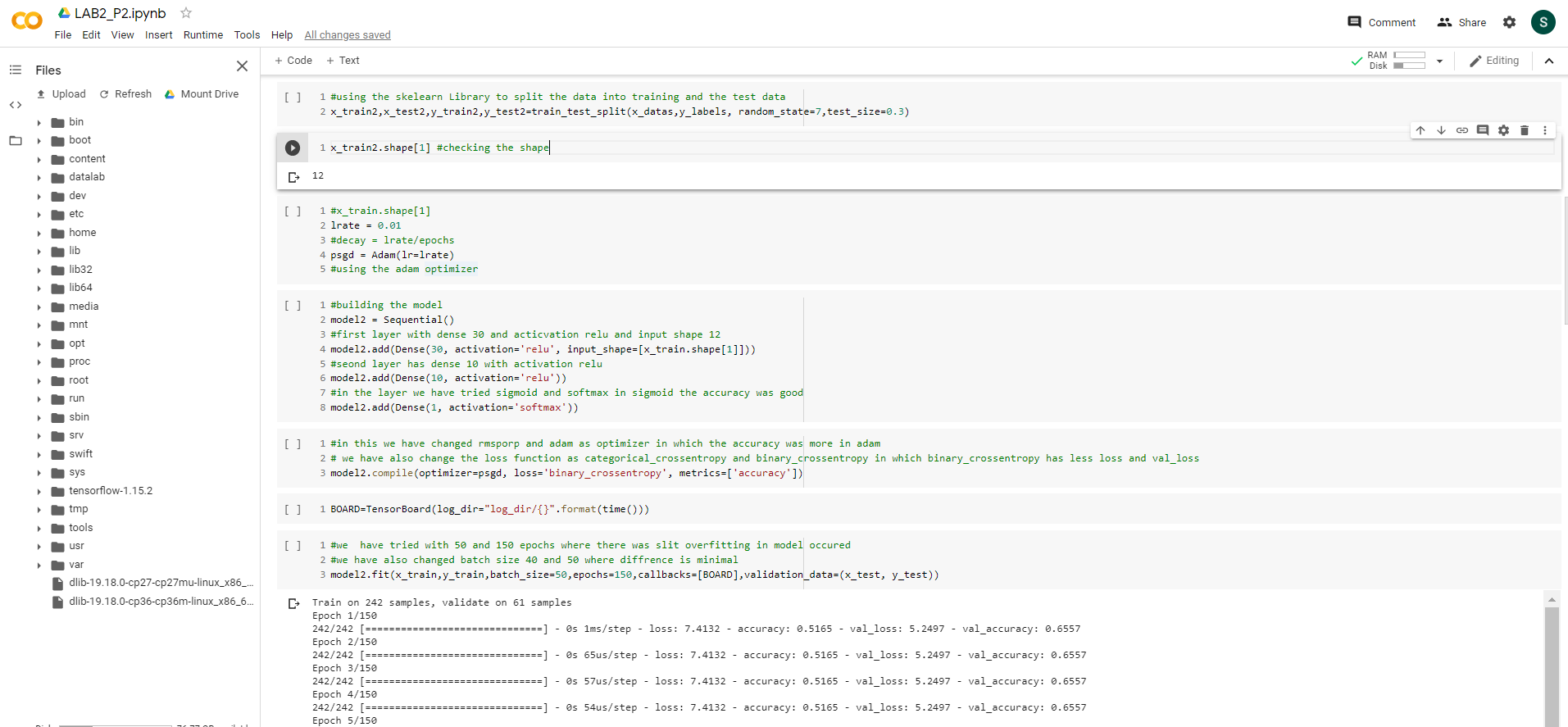
In this step we have split the train and test data using train\_test\_split from Sklearn library.

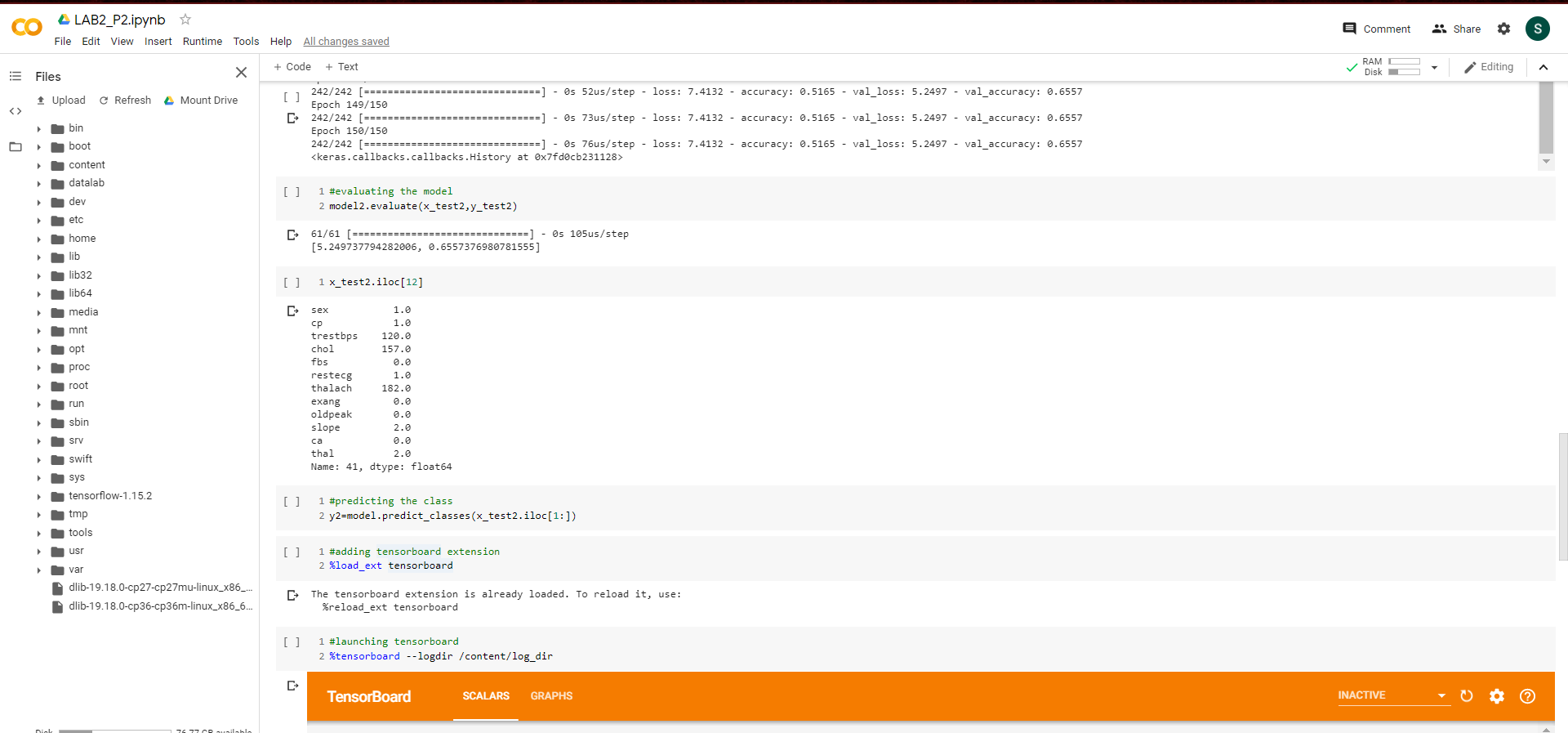
STEP 5:

Build a model with 3 layer with one hidden layer. After building the model we have compiled the model.

STEP 6:

We have used Tensor Board for logging the data for each parameter change. Finally, we have fit the model.



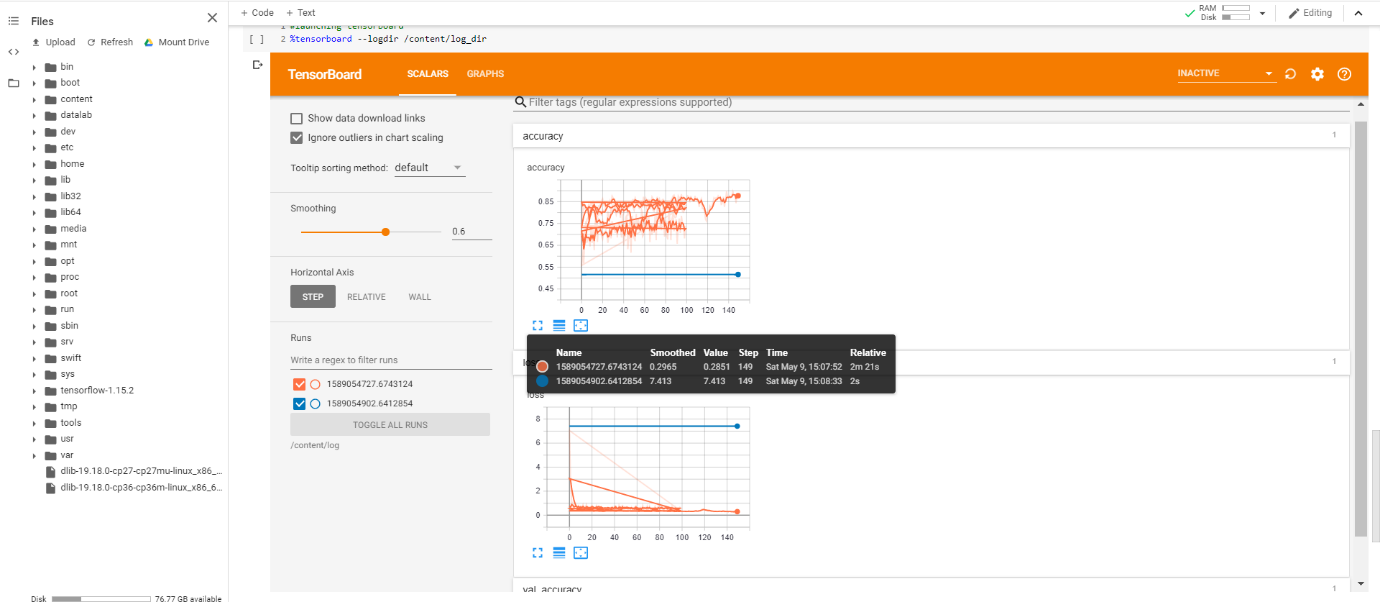


**2b. TENSOR BOARD**

**ANS**

STEP 7:

We have launched Tensor Board for seeing the graph plotted using log files.



**2C. Hyper parameters**

**ANS**

Parameters that we have changed is Optimizer, Batch size, Activation function

OPTIMIZER:

When we have the changed the optimizer from Adam to rmsprop the accuracy got decreased and loss also got increased. In our case Adam was the best Optimizer.

Batch size:

We have used 40 & 50 as batch there was minimal difference in accuracy(Can see in plotted graph) the computational was less in batch 50 when compared to batch size 40.

Activation function:

We have used sigmoid and SoftMax as activation function in the last layer. When we have used sigmoid, we got best fit in our model when compared to softmax the reason was simple we are dealing with binary classification which makes us to choose sigmoid as activation function in the last layer.

**3. Image classification**

**ANS:**

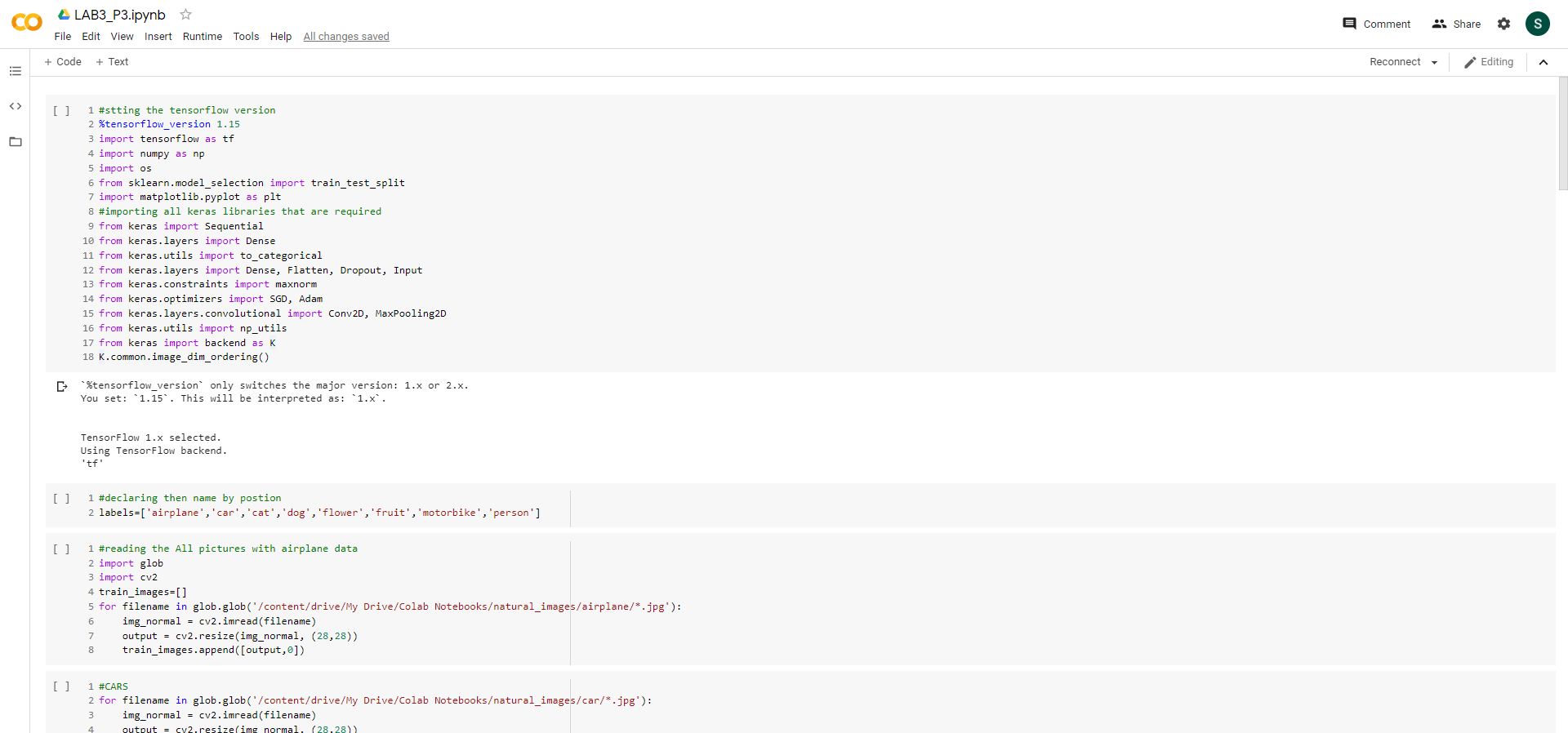
We have used natural-images as dataset for training our model

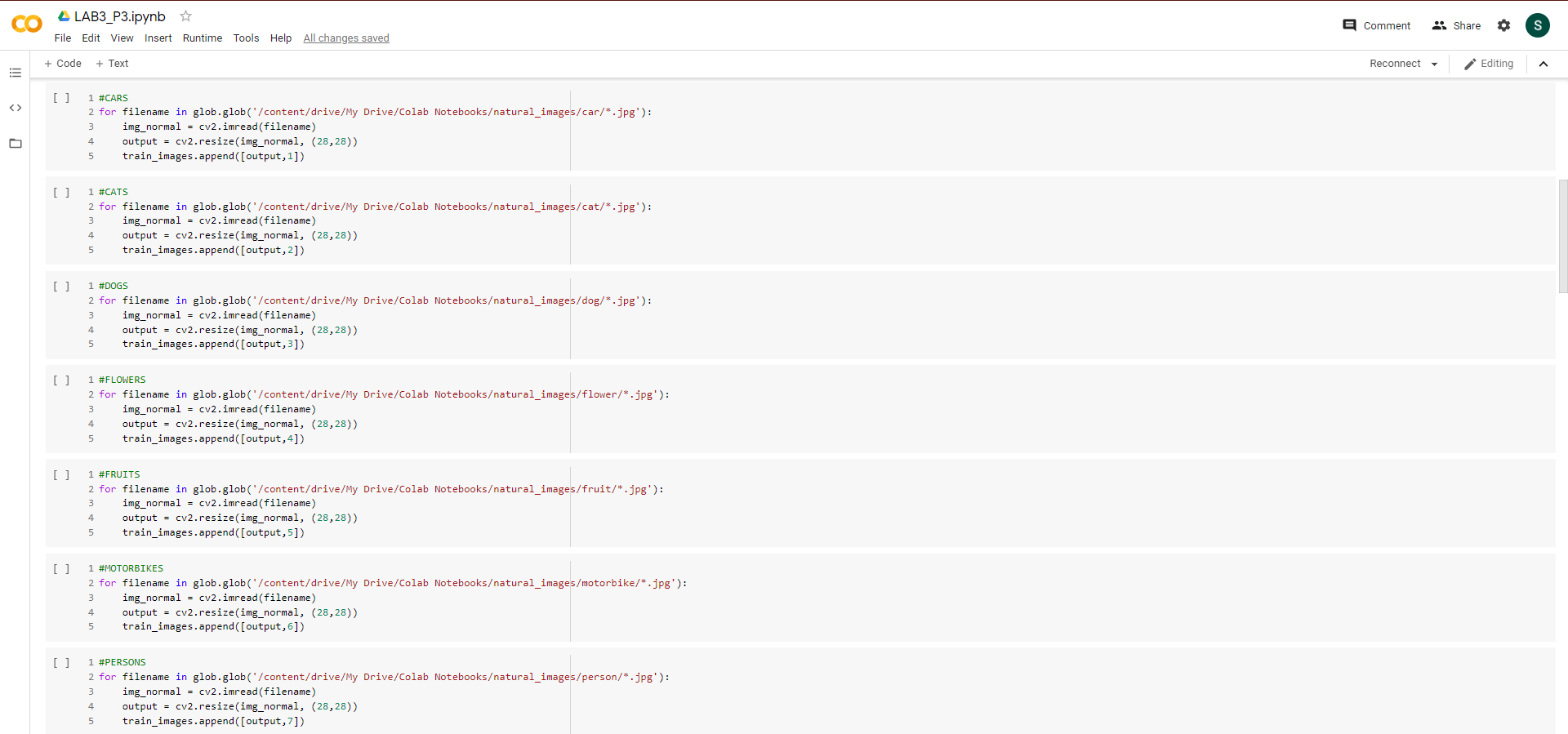
STEP 1:

We have imported all the libraries that are required. We are using convolution 2D layer for our classification.

STEP 2:

We have downloaded the data from Kaggle and uploaded the data to google drive to read and use in our program. We have added more data to original data for better classification. We have used glob to iterate through all file with same type. We have loaded the data and label into same list as nested list. While loading the data we have used open cv library for resizing the image and imread the image.





STEP3:

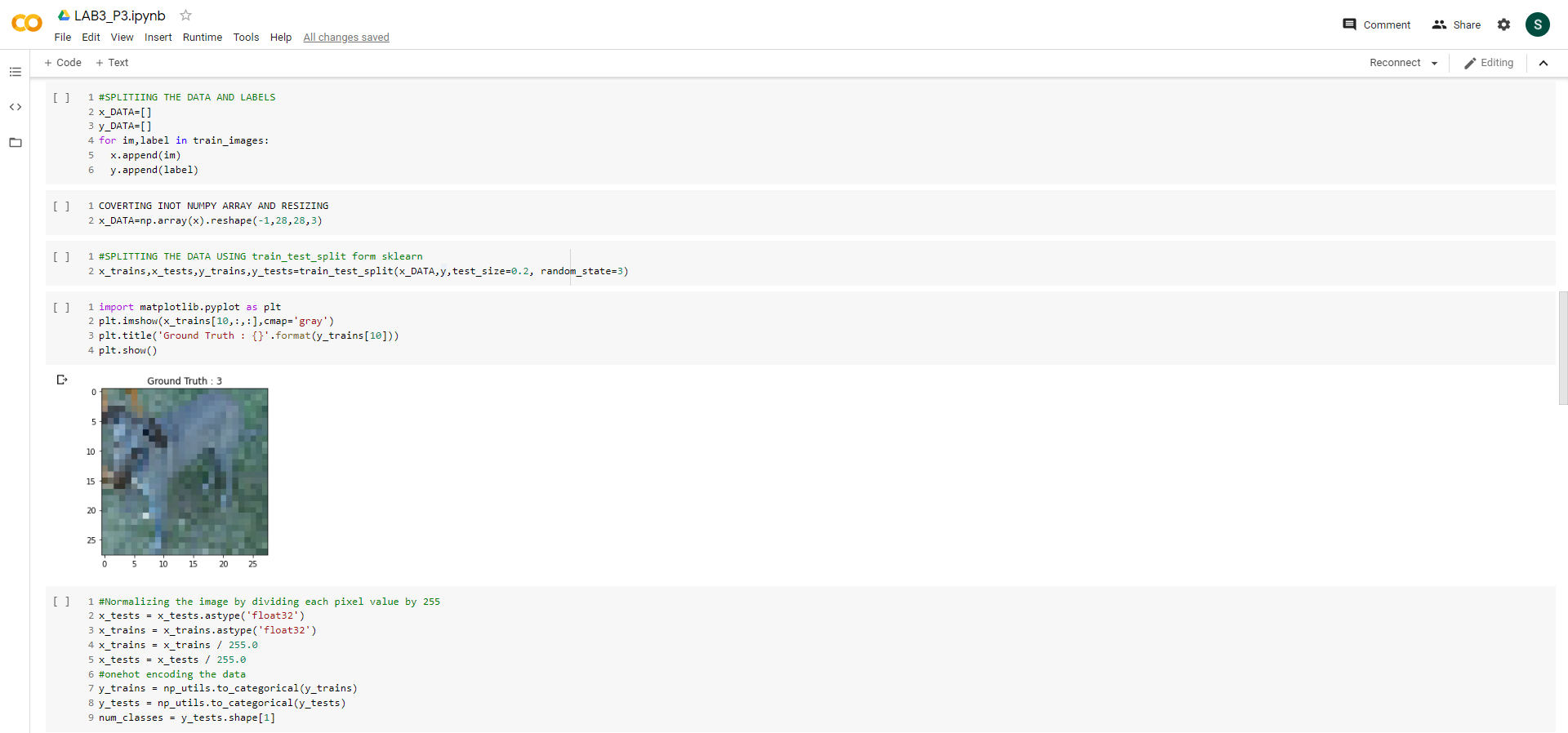
After loading the data to list we have split the image and label to different lists X and Y.

STEP 4:

After splitting the data, we have reshaped the image data array to 28,28,3 shape. After reshaping into array, we have split the train and test data using train\_test\_split from Sklearn library.

STEP 5:

Plotted image for confirming whether everything is cool or not. After plotting the data we have normalized the data by diving each pixel by 255.0 for easy computation. After normalizing we have one hot encoded the label for feeding into our model.



STEP 6:

We have built the model using sequential model with 1 con2d layer and one max pooling layer. We have used softmax as activation in the last layer as we are dealing with more than 2 classes.

STEP 7:

Compiled the model with Adam optimizer and loss as binary\_crossentropy. After compiling the model we have fit the model.



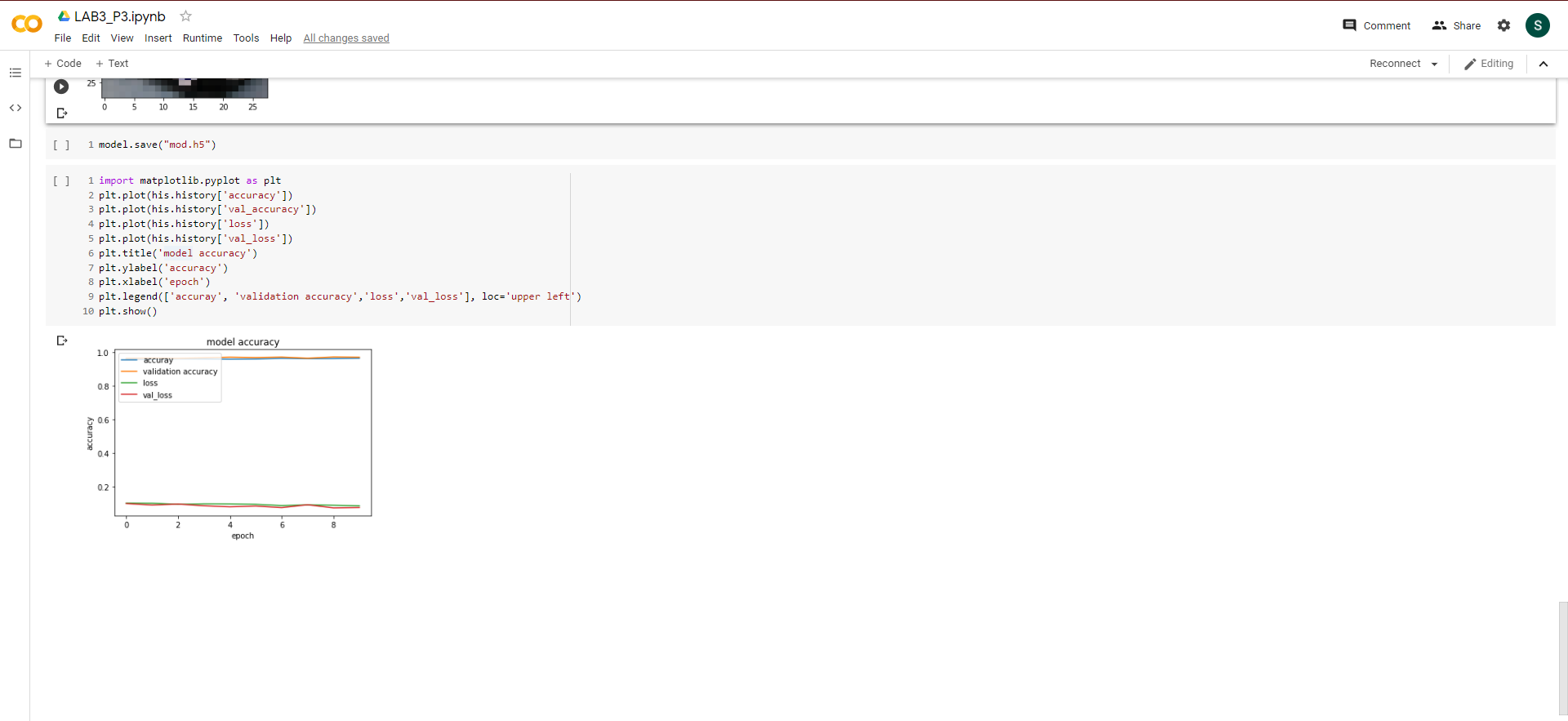
STEP 8:

We have got best fit of model at 10 epochs after the fit we have predicted the image from test data, and we have also plotted the image.



STEP 9:

We have plot loss, accuracy, validation accuracy, validation loss. In the below graph we can see that we got best fit for our model.

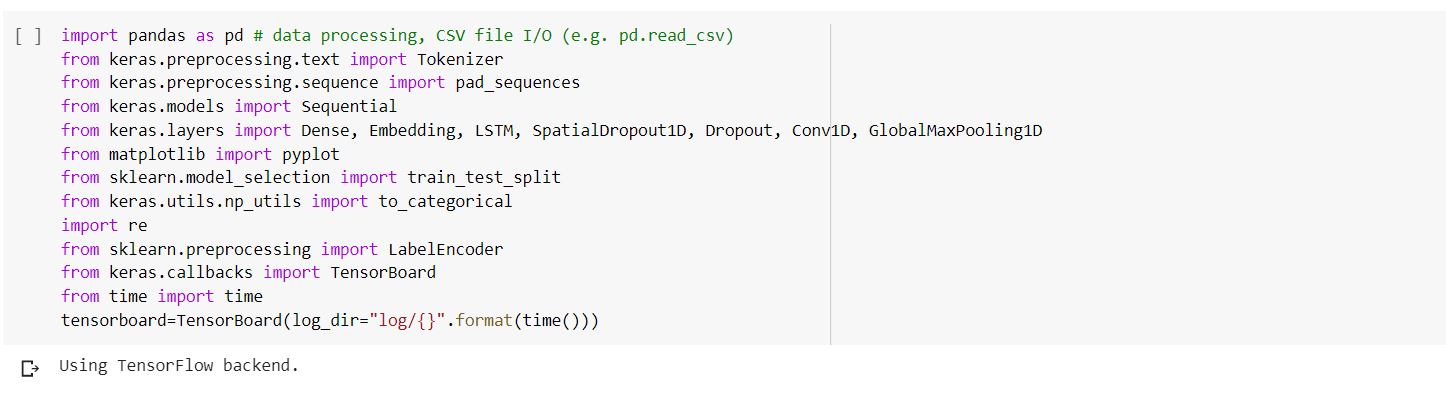


**4.** **Text classification using CNN model.**

**ANS:** We have downloaded the Kaggle Movie reviews dataset and have performed text classification using the CNN Model.

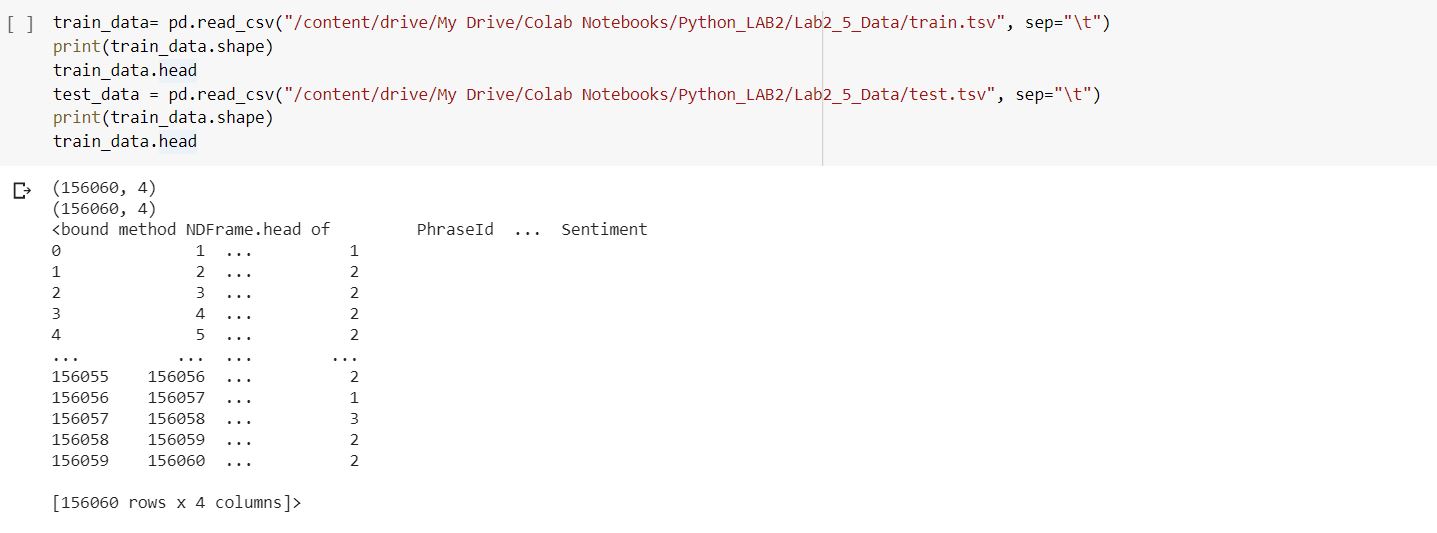
STEP 1:

We have imported all the libraries that are required. We are using CNN layer for our classification.



STEP 2:

Using pandas read\_csv method to read the train and test file which we have uploaded in our google drive. And just printing the head to know what are the columns that exist in the tsv file. We are printing the shape of train and test to know the number of rows.



STEP 3:

In this step we are dropping the unnecessary columns and the target column. Applying the lambda function, we are converting all the text in the Phrase column of all rows to lower case. Next we are tokenizing the Phrase text into words and adding some padding to create a string of same length for all the tokens.



STEP 4:

Here in step we use label Encoder to normalize the data. We use this encoder to convert or transform all the non-numerical data to numerical data. Next we split the data into training and testing or validation data using train\_test\_split method.

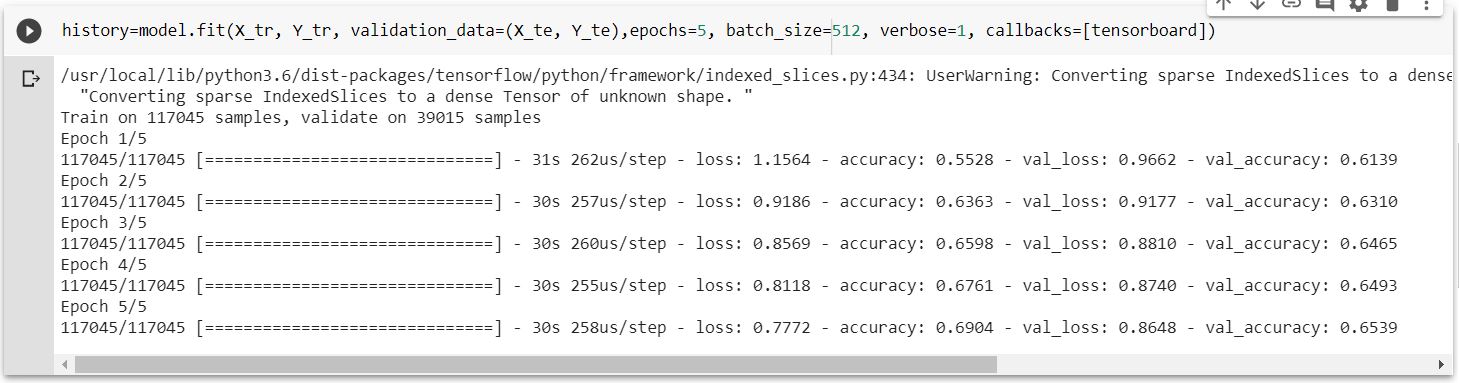
STEP 5:

Now we build a model using sequential model and add a few layers to it. We added the embedding layer, introduced some dropouts, added a Conv1D, GlobalMaxPooling1D, a dense layer and at the last an output layer with number of classes as output dimension. Using the categorical\_crossentropy as loss function, adam as optimizer we have complied the model.



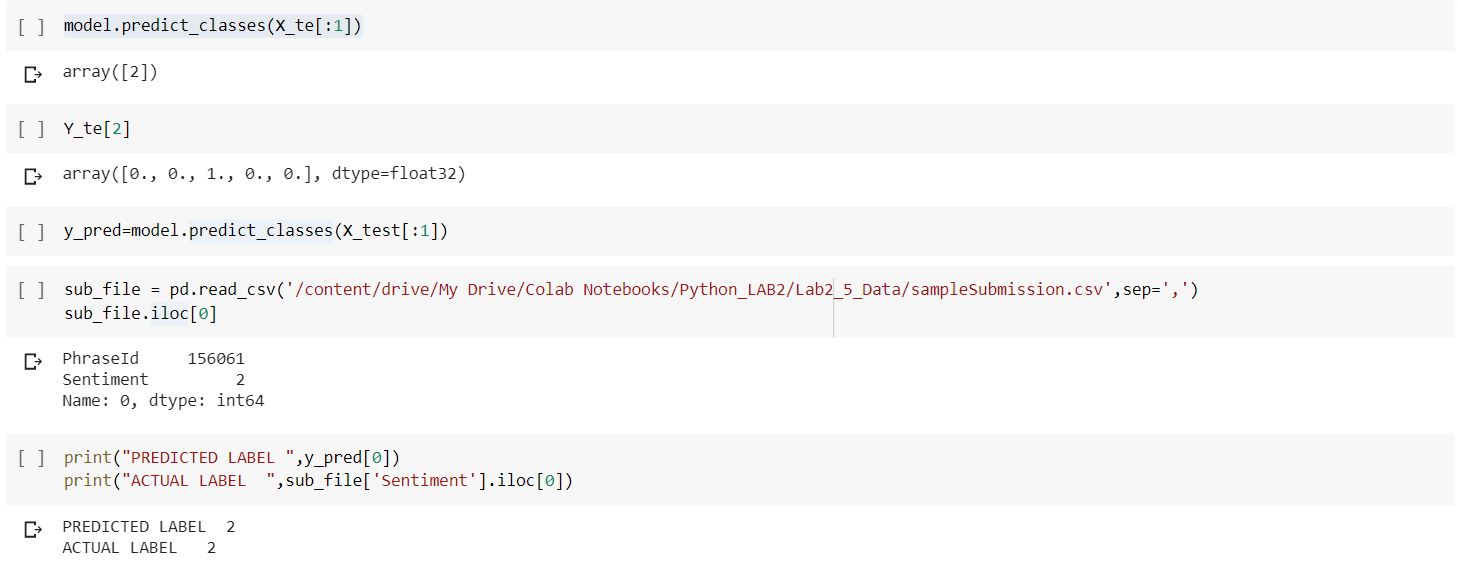
STEP 6:

Using the newly obtained test and train data split we fit the model and run the epochs to know the accuracy and loss of our model.



STEP 7:

Now using the built model, we predict the sentiment of a sentence from the test.tsv file. After predicting it we test it using data from the sampleSubmission.csv and compare the actual and predicted values.

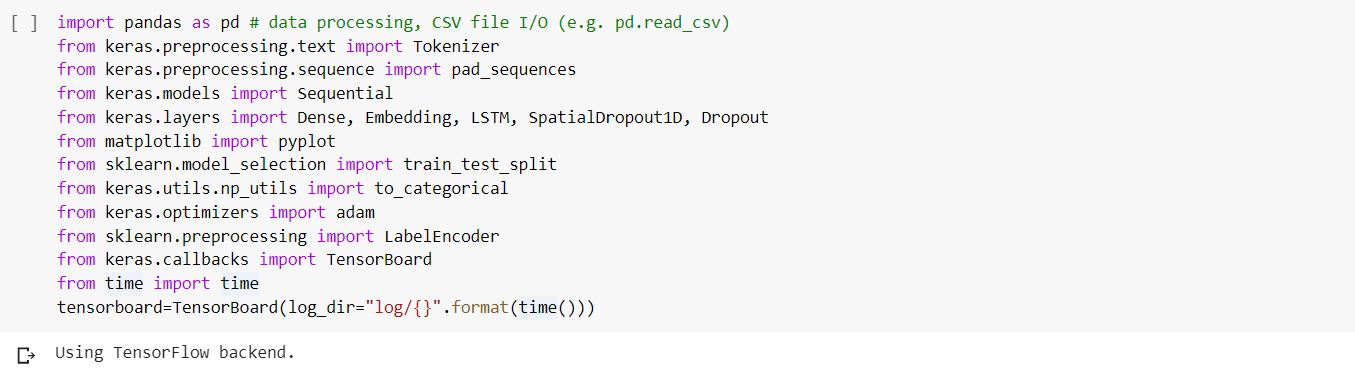


**5.** **Text classification using LSTM model.**

**ANS:** We have downloaded the Kaggle Movie reviews dataset and have performed text classification using the LSTM Model.

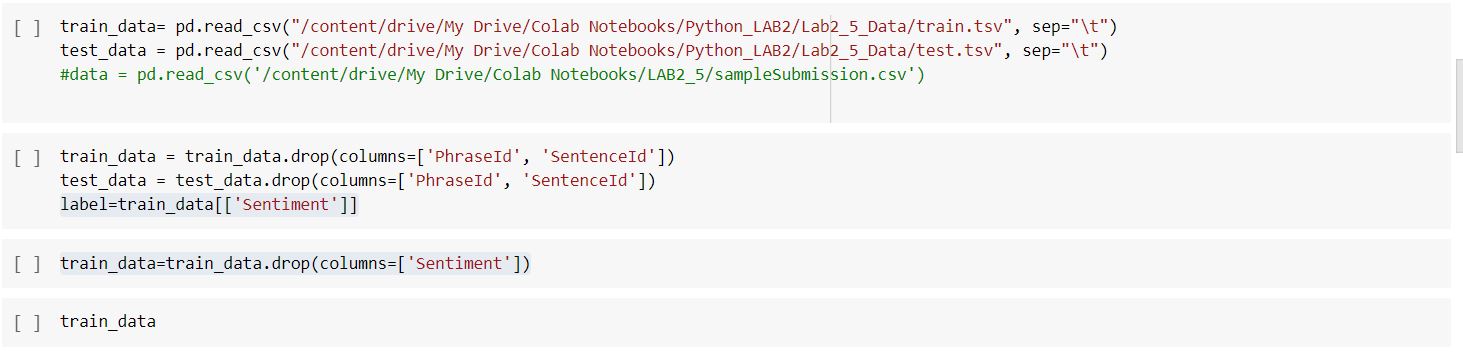
STEP 1:

We have imported all the libraries that are required. We are using LSTM layer for our classification.



STEP 2:

Using pandas read\_csv method to read the train and test file which we have uploaded in our google drive. And just printing the head to know what are the columns that exist in the tsv file. We are printing the shape of train and test to know the number of rows. After that we are dropping the unnecessary columns and the target column.



STEP 3:

In this step we are applying the lambda function, we are converting all the text in the Phrase column of all rows to lower case. Next we are tokenizing the Phrase text into words and adding some padding to create a string of same length for all the tokens.

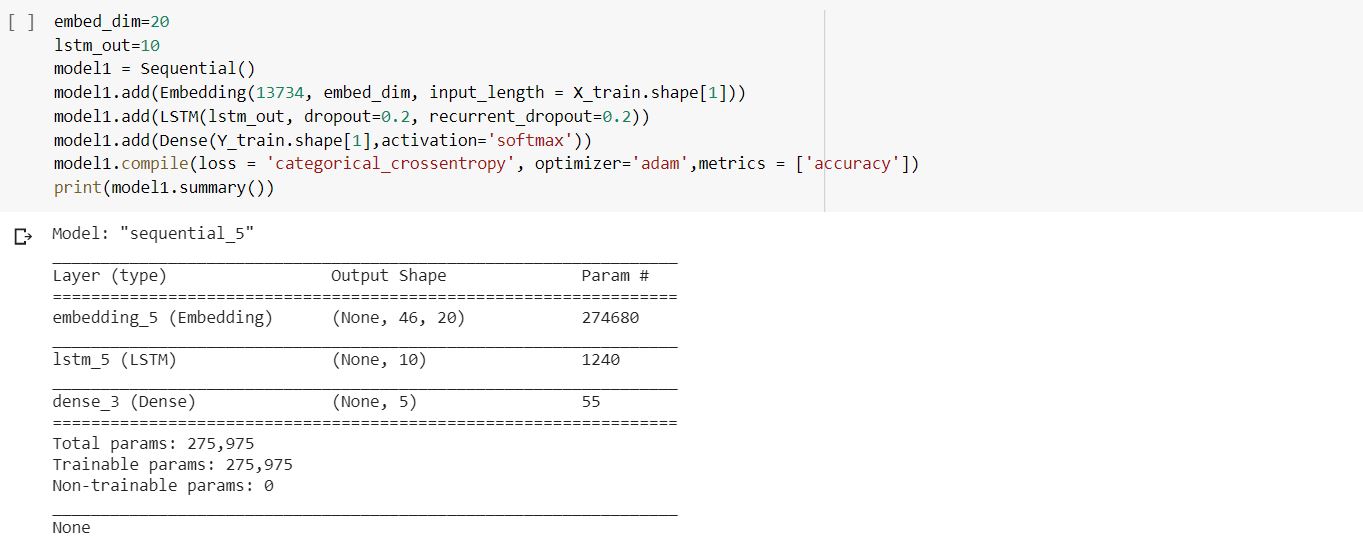
STEP 4:

Here in step we use label Encoder to normalize the data. We use this encoder to convert or transform all the non-numerical data to numerical data. Next we split the data into training and testing or validation data using train\_test\_split method.



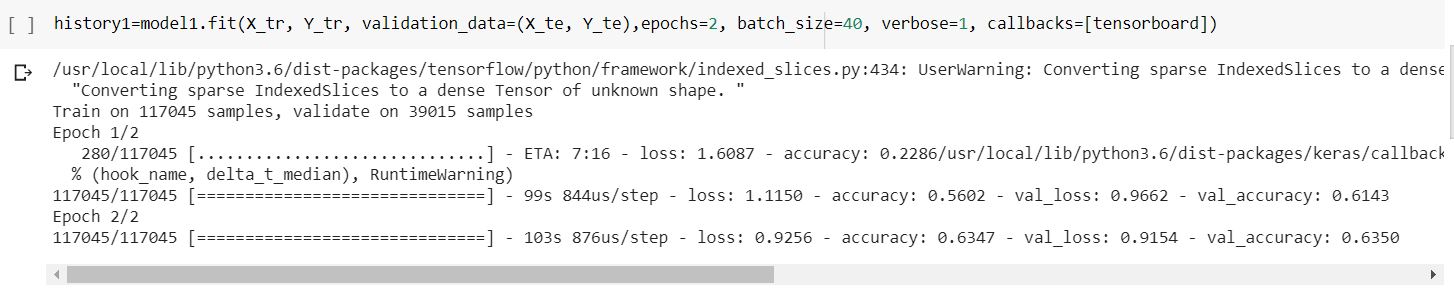
STEP 5:

Now we build a model using sequential model and add a few layers to it. We added the embedding layer, introduced some dropouts, added a LSTM and a dense layer as the last an output layer with number of classes as output dimension. Using the categorical\_crossentropy as loss function, adam as optimizer we have complied the model.



STEP 6:

Using the newly obtained test and train data split we fit the model and run the epochs to know the accuracy and loss of our model.



STEP 7:

Now using the built model, we predict the sentiment of a sentence from the test.tsv file. After predicting it we test it using data from the sampleSubmission.csv and compare the actual and predicted values.

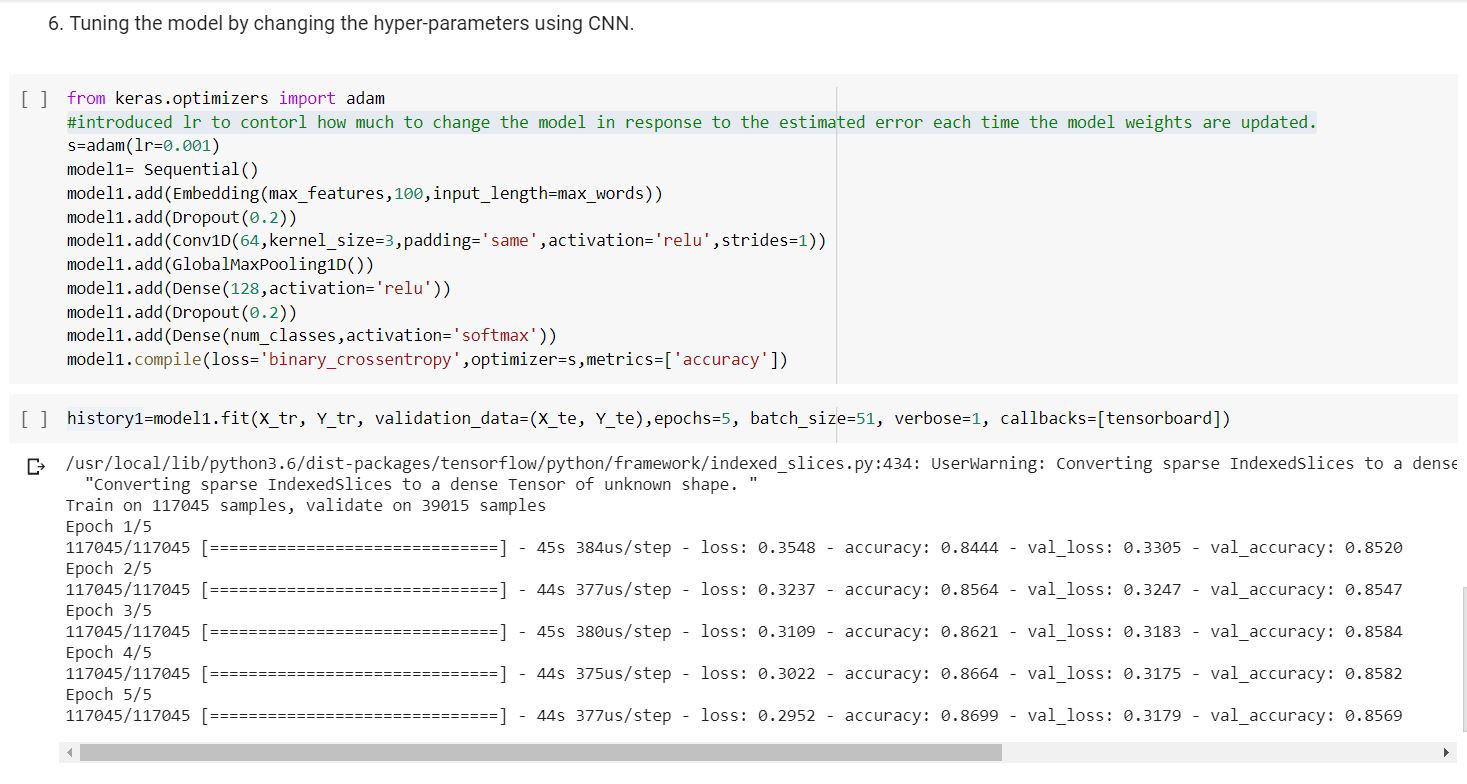


**6. Choose the best from above models and tune the hyper parameters for good accuracy.**

**ANS:** From the above two models by looking at the accuracy we can tell that LSTM is giving more accuracy than CNN model. Now we try to tune our model

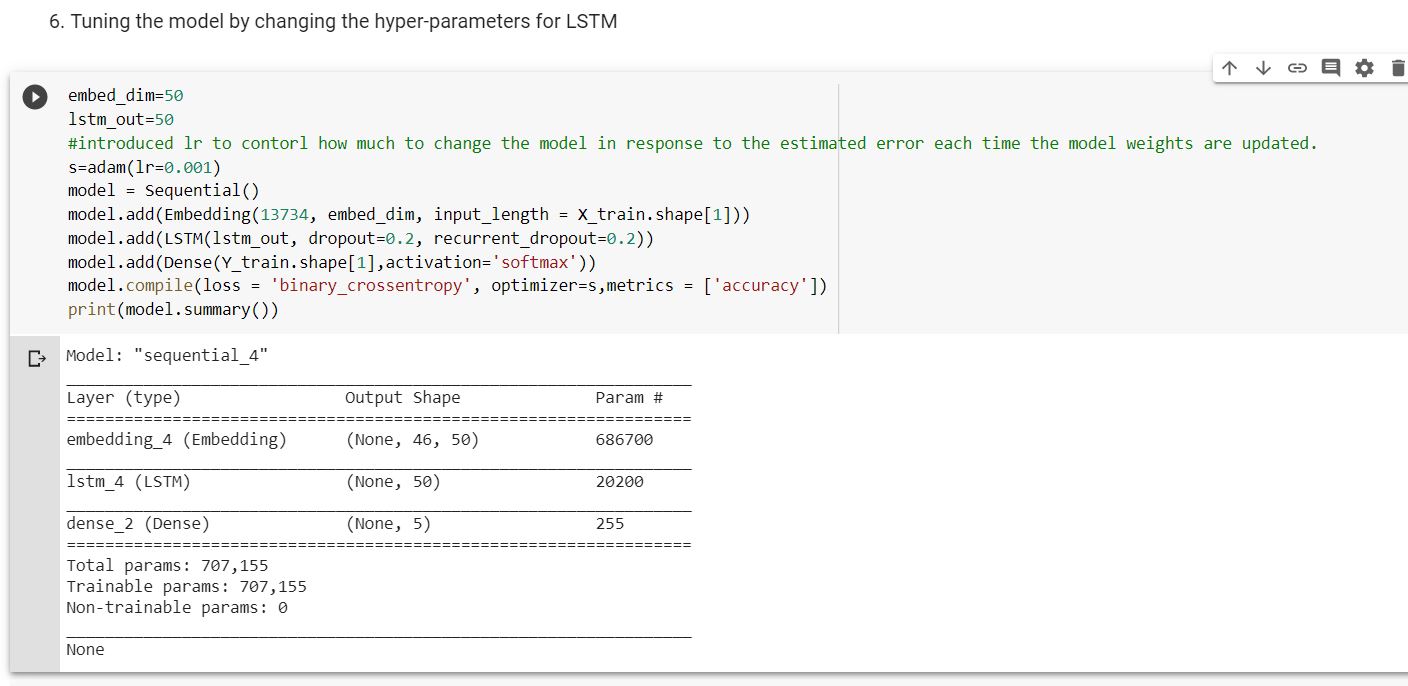
STEP 1:

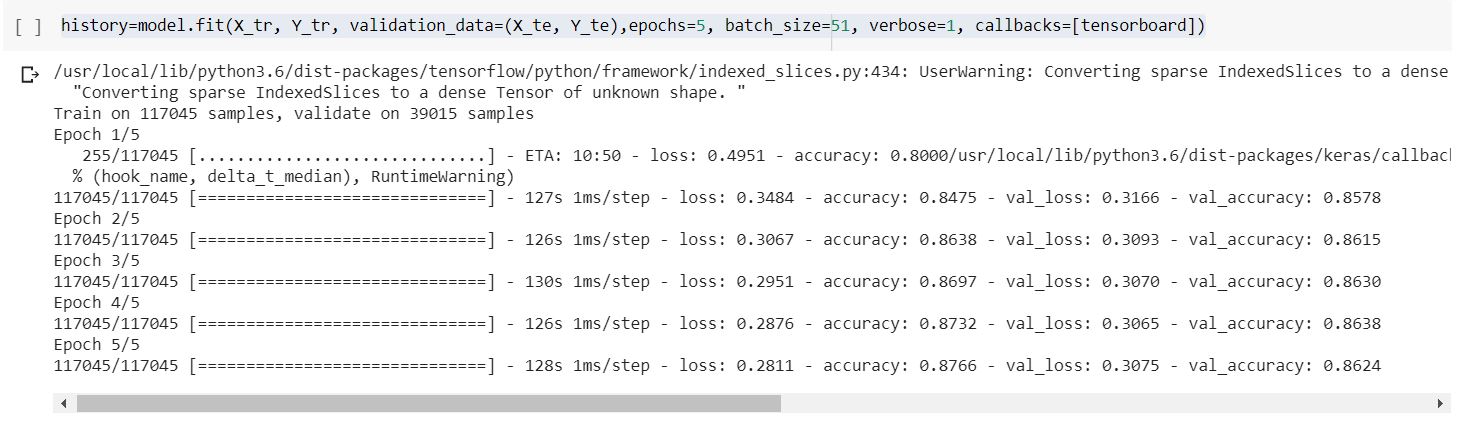
We have edited the previous programs for this execution using CNN. We have taken the model of question 4, and changed the hyper parameters. We have introduced the learning rate, changed the number of neurons, and changed the loss function to binary\_crossentropy. While fitting the model we have changed the batch size. We have obtained an accuracy of 86%, which means there is an increase in the accuracy from 69% to 86%.



STEP 2:

We have edited the previous programs for this execution using LSTM. We have taken the model of question 5, and changed the hyper parameters. We have introduced the learning rate, changed the number of neurons, and changed the loss function to binary\_crossentropy. While fitting the model we have changed the batch size. We have obtained an accuracy of 87%, which means there is an increase in the accuracy from 63% to 87%.



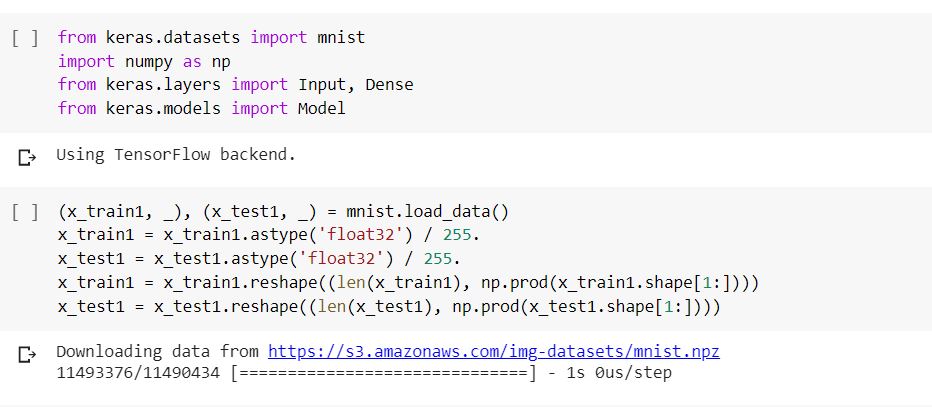


**7. AutoEncoders on MNIST data**

**ANS:** We have downloaded the MNIST dataset, and using the autoencoders we have encoded and decoded a particular image.

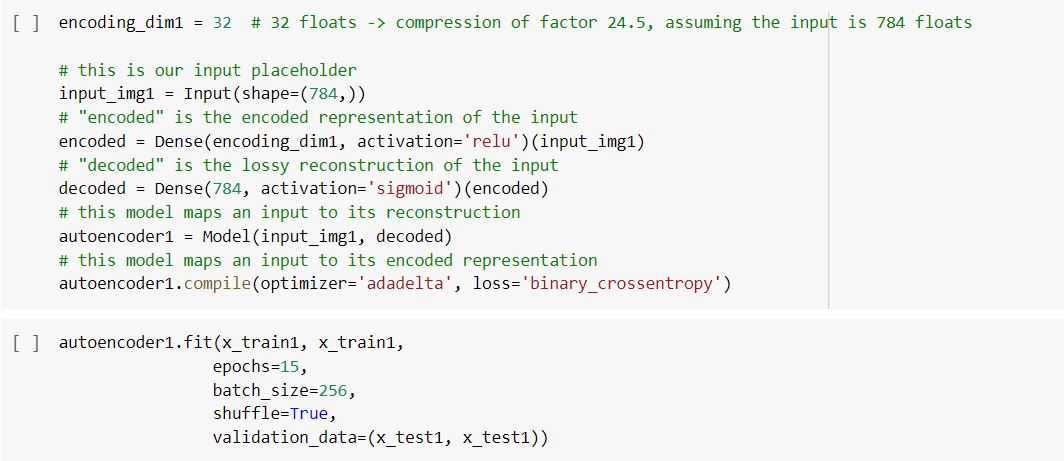
STEP 1:

We have imported all the libraries that are required. We are loading the data from mnist from keras.dataset library. Now we are scaling the data by converting it into float and dividing it by 255, so that traversing can be done between 0 and 1. Then we are reshaping the training and testing data.



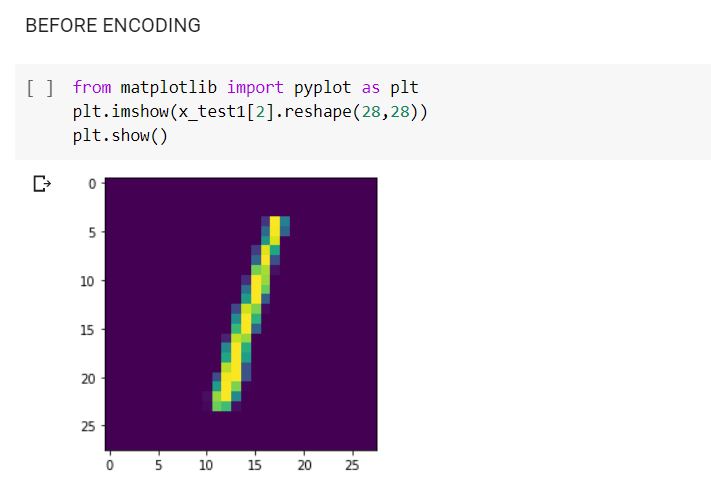
STEP 2:

In this step we perform encoding and decoding and use autoencoder to build the model and compile it using adadelta as the optimizer and binary\_crossentropy as loss. At last fit the model using 15 epochs and 256 as batch size. We use encoding to represent the input and decoding to get the lossy reconstruction of the input.



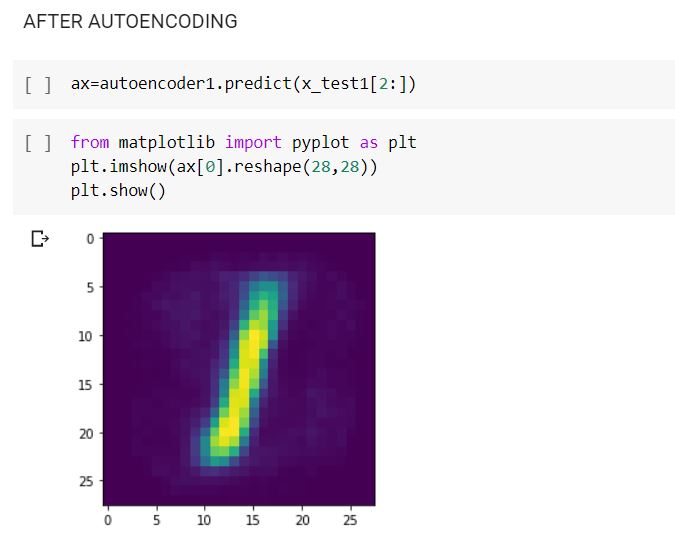
STEP 3:

As we were asked to perform the encoding and decoding on a single image. In this stp we are displaying the image before encoding, which is from the test data set.



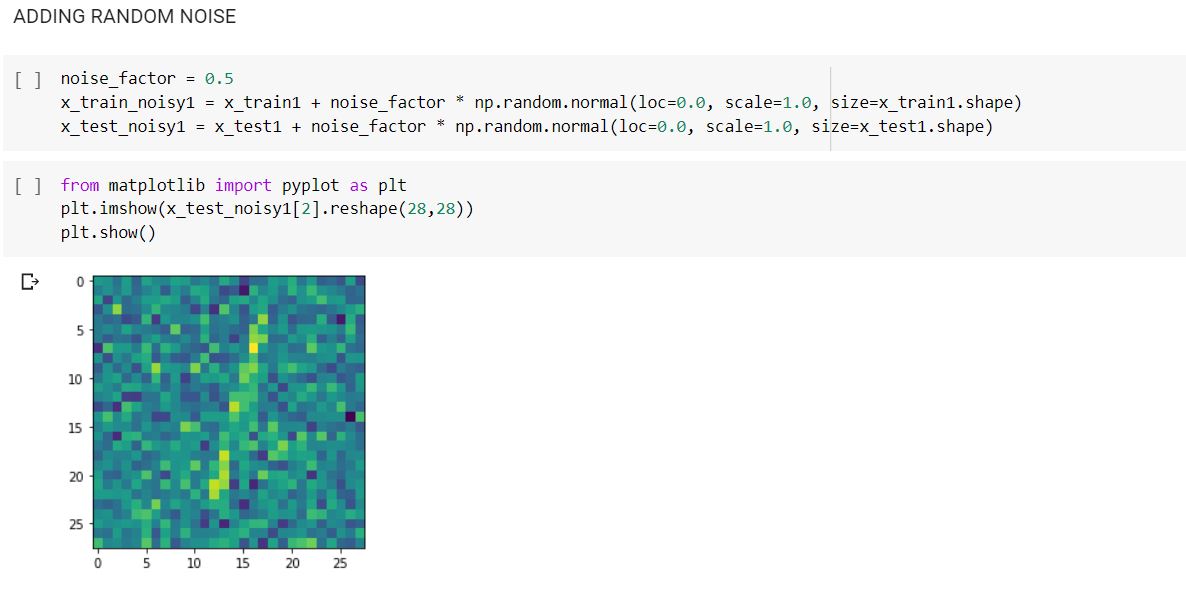
STEP 4:

Displaying the image after autoencoding.



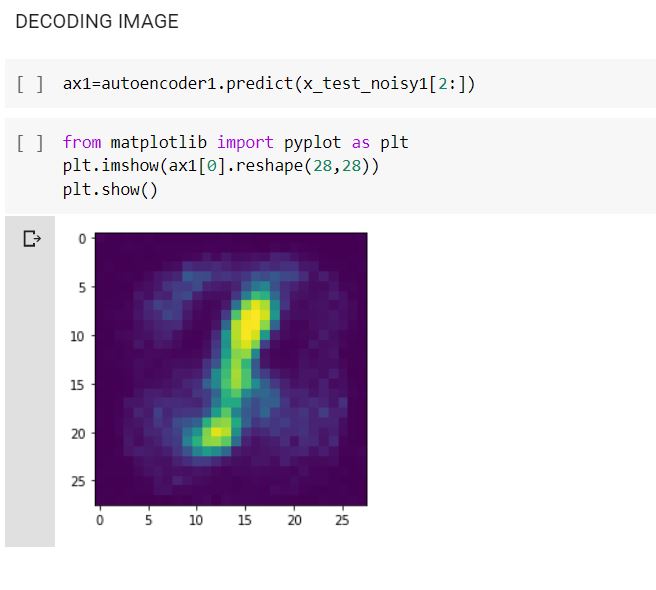
STEP 5:

Displays the image after adding some noise to it.



STEP 6:

Displaying the image after decoding the image, which is correctly predicted. This means our constructed model is working fine.



1. **Evaluation & Discussion**
2. Built a sequential model using Linear Regression, showed the graph using TensorBoard and plotted the graph by changing few mentioned hyper parameters.
3. Performed Logistic Regression on the heart dataset. We have normalized the data before feeding, showed the loss on the tensor board and we have changed three parameters and have reported the accuracy changes.
4. Performed Image classification using CNN Model.
5. Text Classification using CNN Model is performed
6. Text Classification using LSTM.
7. Compared and provided the best model between the above two models for text classification.
8. Encoded and Decoded the mnist dataset using Autoencoders and performed all this on a single image.
9. **Conclusion**

All programs were performed according to the objectives specified and the evaluation criteria is met.