PYTHON PROJECT 2 REPORT

Team 2

INTRODUCTION:

We are going to perform different sequence classification and text classification models on some datasets where we have to predict the data based on the sequence and based on the category they belong to. In this report we are mainly going to focus on CNN and LSTM models which can be used to image classification and text classification and we are also going to apply some encoding processes which helps us to increase the accuracy of the predicted data.

OBJECTIVE:

The main Objective this report is to classify the test data based on the trained data in a data set and to increase the accuracy of the prediction with less loss.

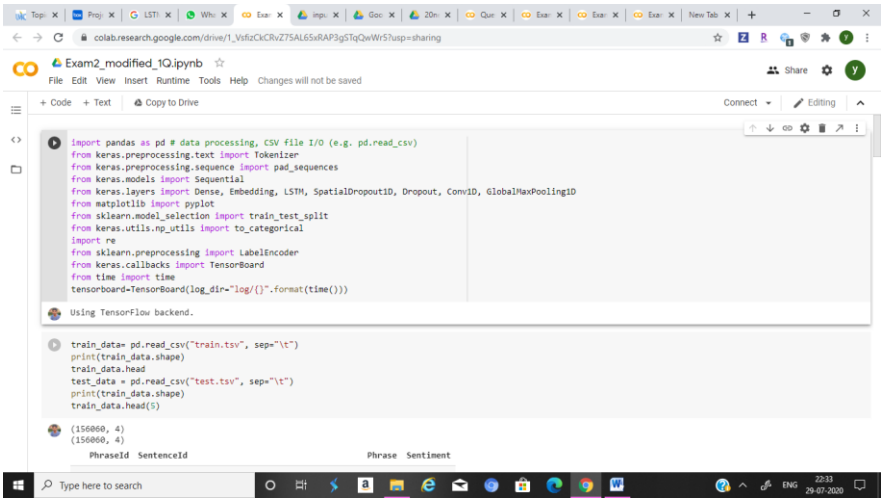
METHODS:

- In this project we are going to use the sequence classification methods that are CNN that is convolutional neural network which is used to classify the images and also text in a given dataset.
- LSTM model which is known as Long short term memory model network which is used for sequence classification which is really helpful in machine learning tasks like speech recognition etc,.
- And we are going to apply auto encoder and embedded layers to the data to see how the accuracy of the predicted data changes.

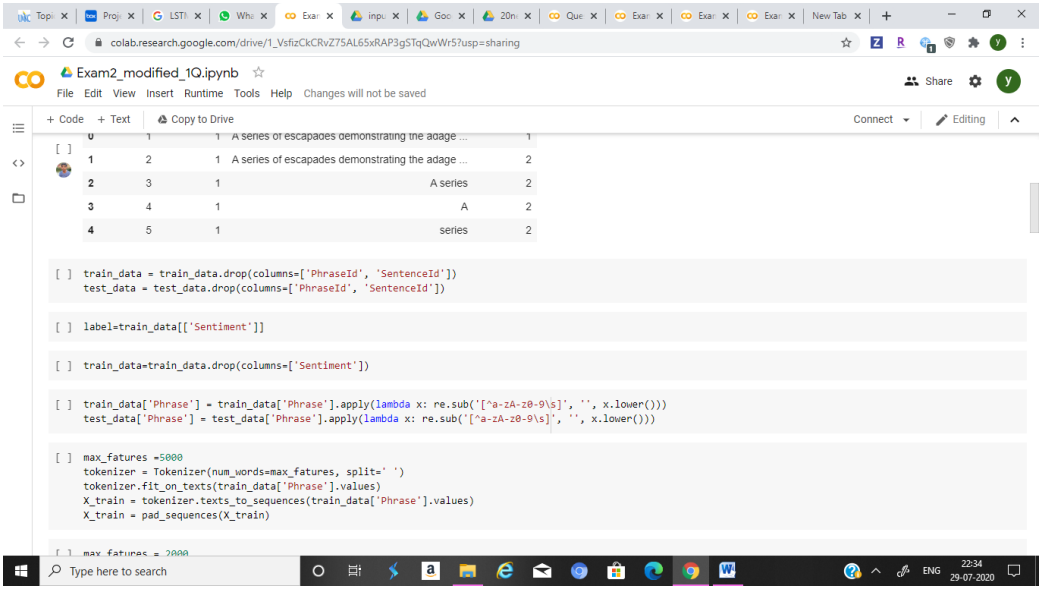
WORKFLOW:

1)

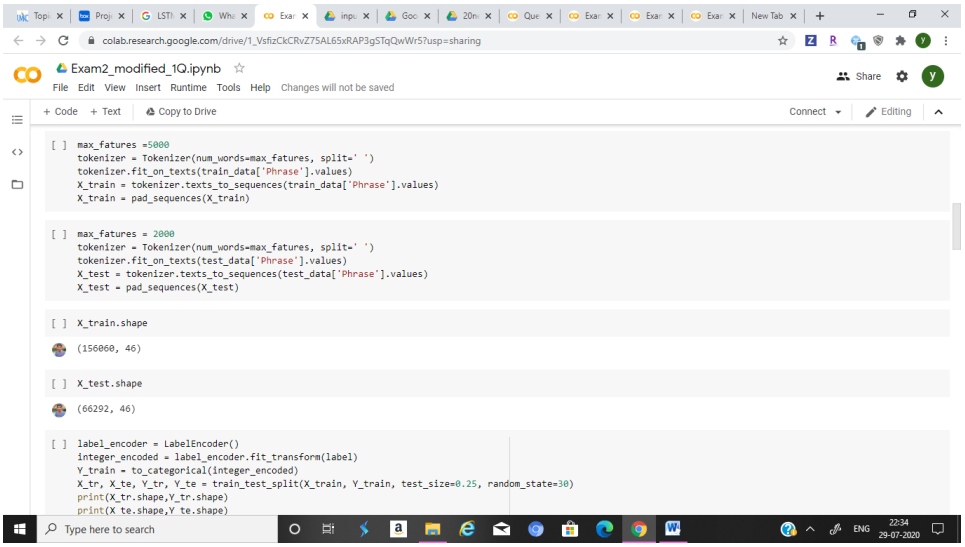
- For this we are going to perform the text classification on the review's sentiment dataset using CNN model.
- First we have to upload all the packages needed and later import the dataset to a variable and now we have to separate the train data and test data.



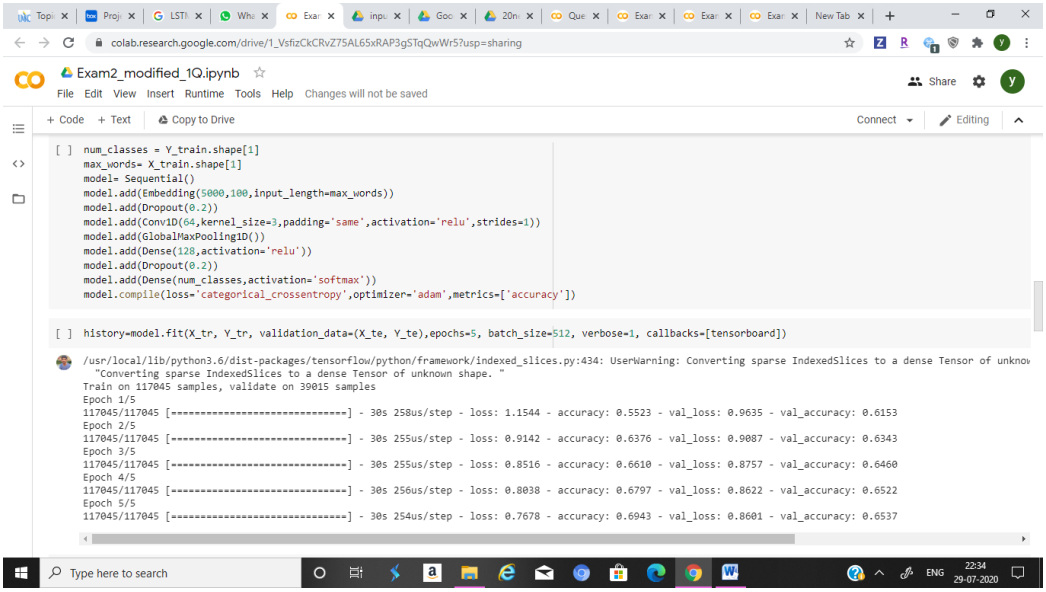
- Later that we have to perform the tokenization on the data to group the words that are similar.
- Later we have select the features and based on the features we are going to separate the data and predict it.

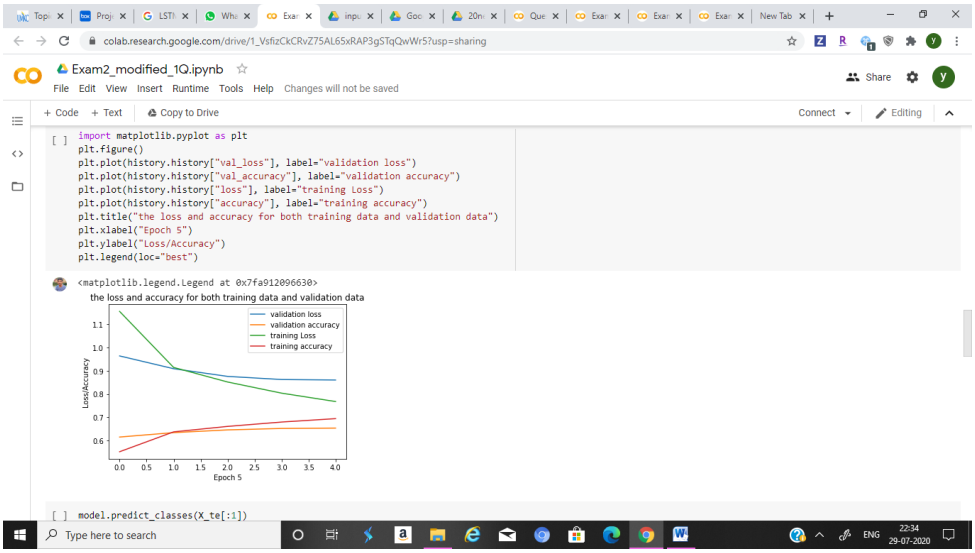


- Later apply the label encoder on the data so that all the nonnumeric datasets are converted into the string dataset and later that we have to create our model.

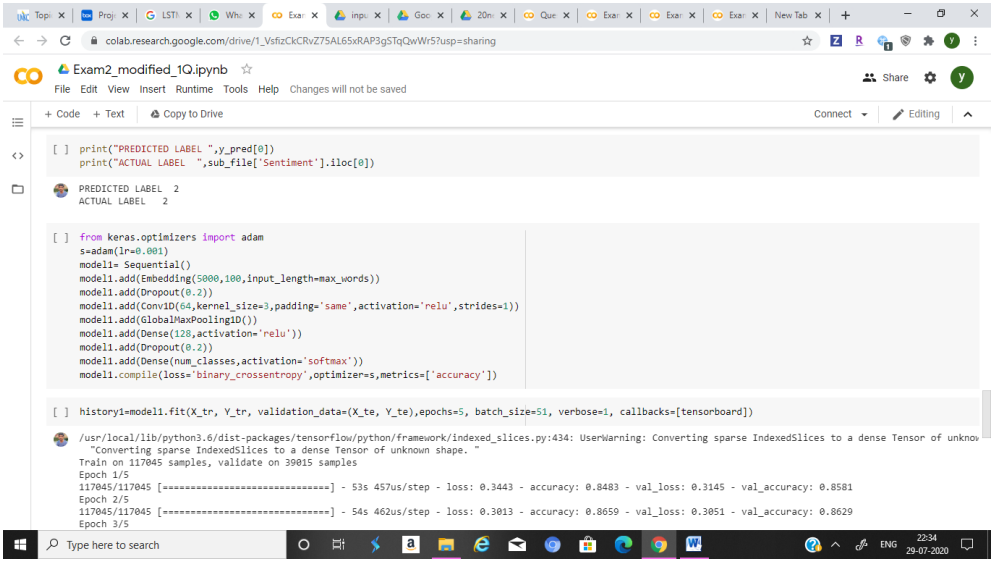


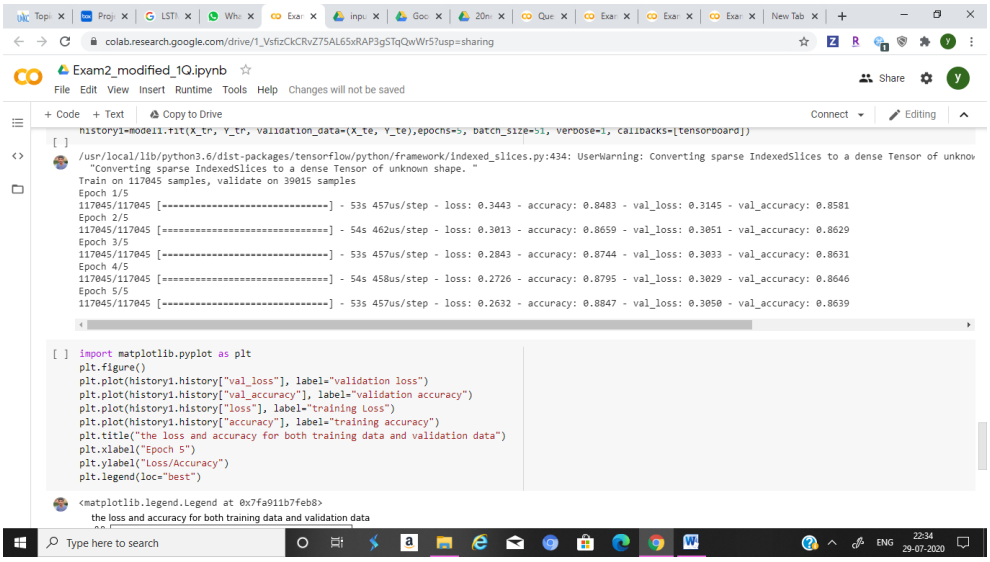
- We are using the CNN model here and we need to fit our data on to this model.
- We calculated the loss and accuracy of the data and plotted it.

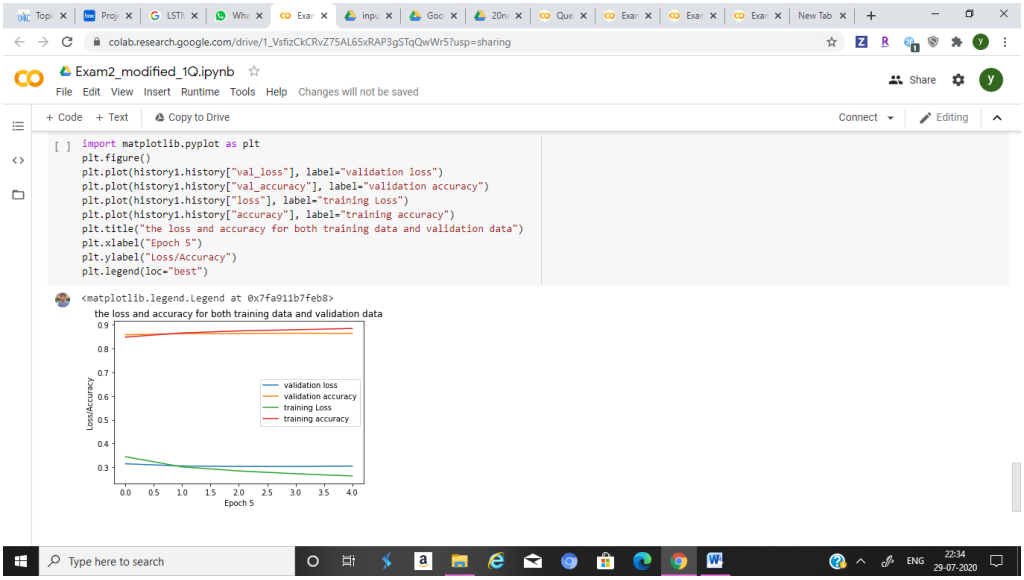




- Later we need to include the embedding layer to the model so if the accuracy changes and we did observe that accuracy is improved .
- Later we observed the overfitting problem in this, this overfitting can be removed by using learning rate .

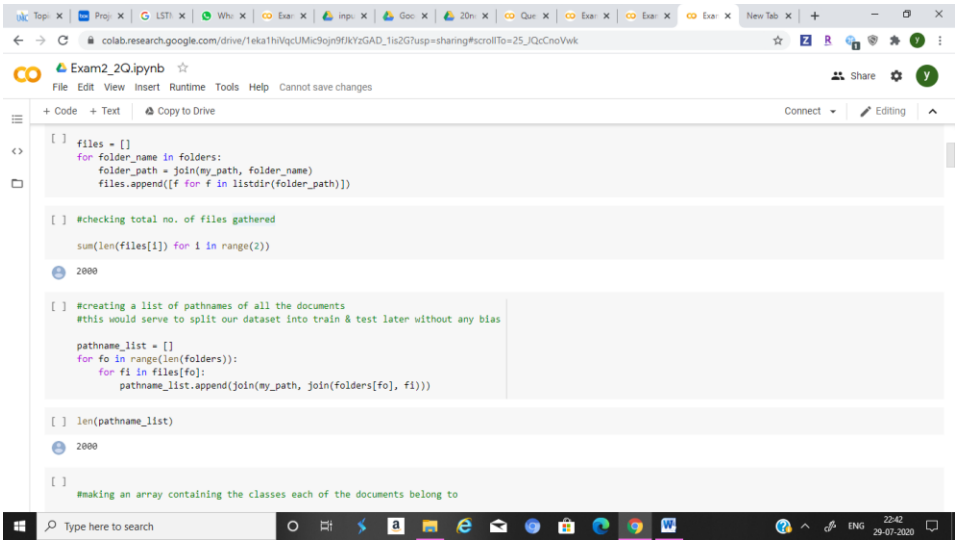




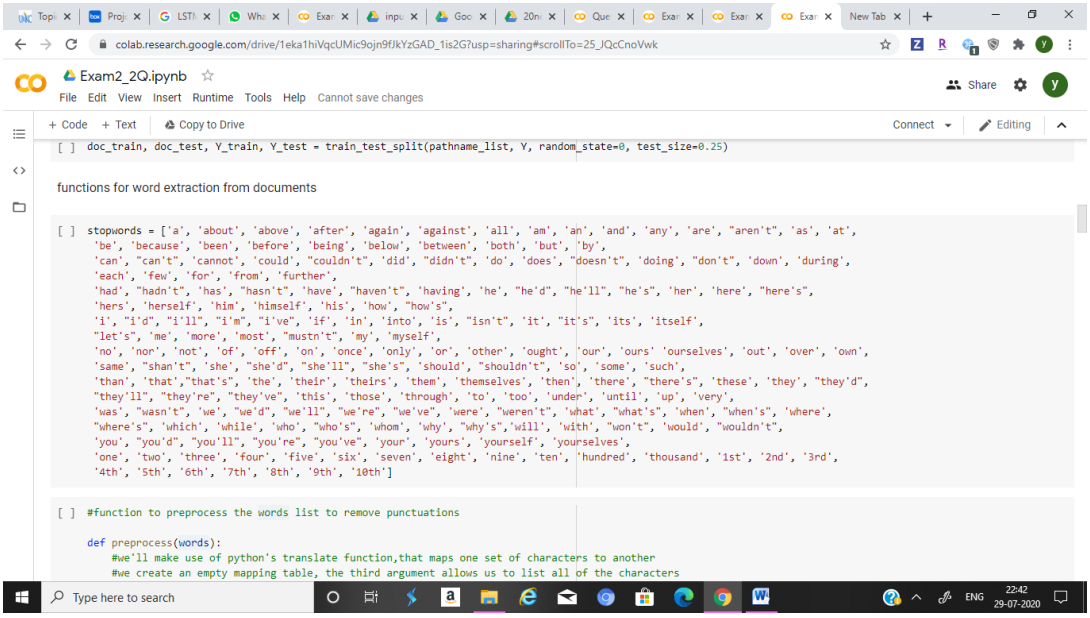


2) We are going to import all the packages that are needed for the program and we are going to use the 20 news groups dataset for this task, our task here is to classify the text using LSTM model.

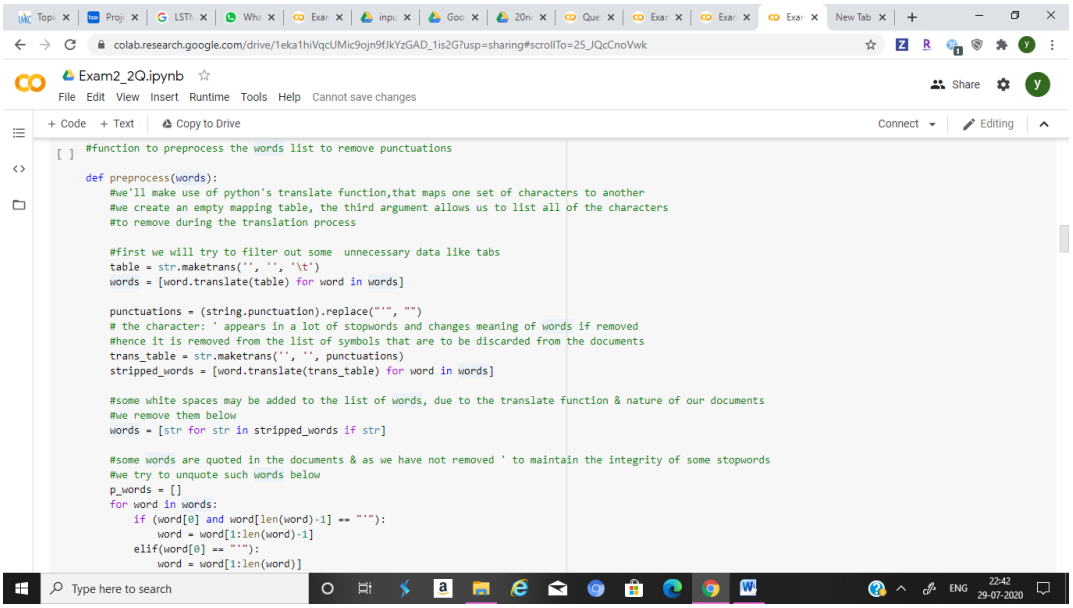
- so first we are going to create a file and add the 20news groups data onto that file later, we are gonna create a add this file to the already existing file, the 20newsgroups data is a live data so for more information belong to same category we going to create different files and append them later.

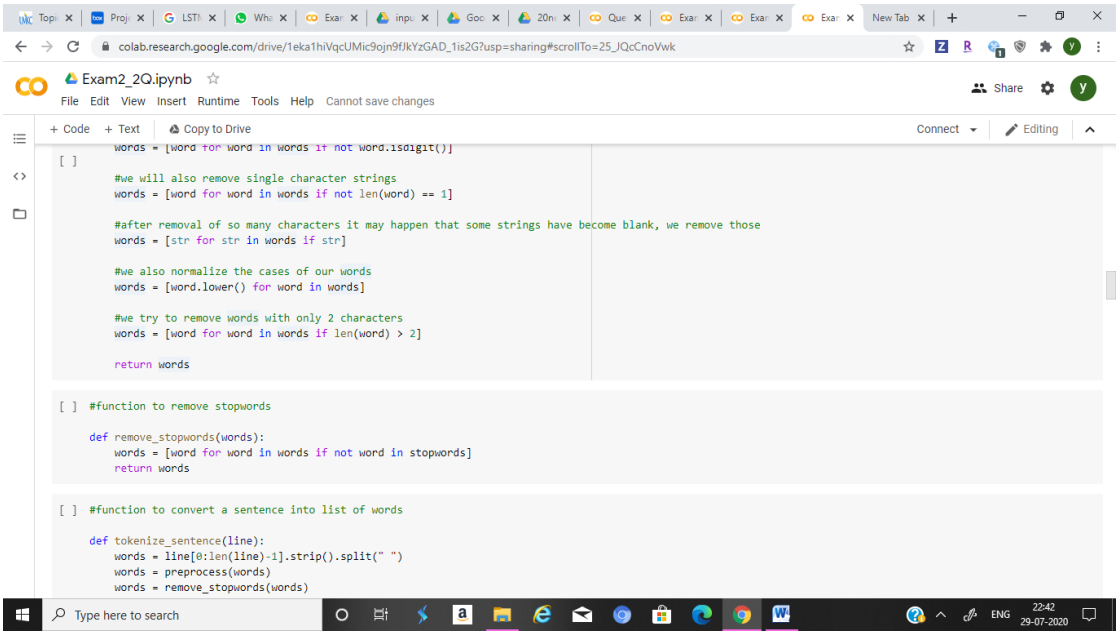


- Now split the data into training data and testing data using train test split model.
- Later select the stop words and separate them from the data file, we need to separate the stop words because they make the prediction go wrong sometime and these words are not really useful in the prediction.

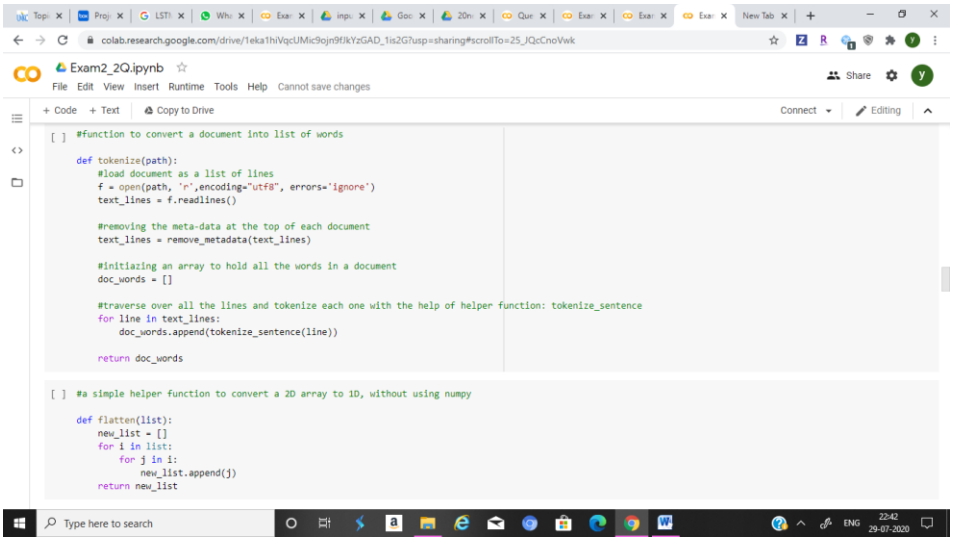


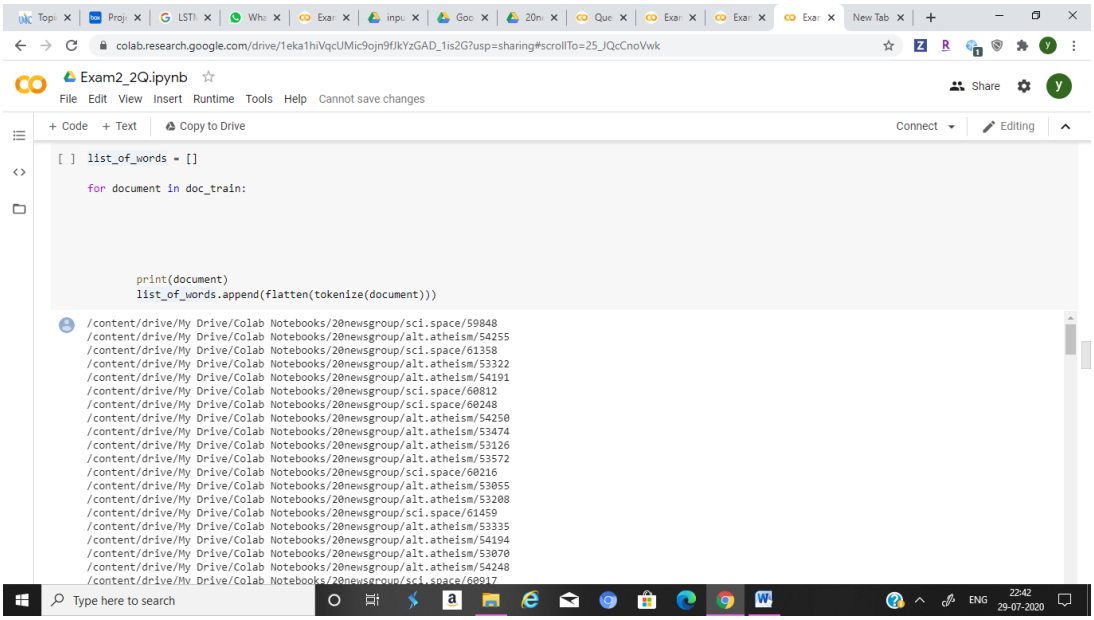
- Later, based on the certain features we extract the data and tokenize the data so the data falls under the same category, later that remove the stop words and after that now we have the train and test data calculate the length of the data and now we can print the data under the tokenizing.

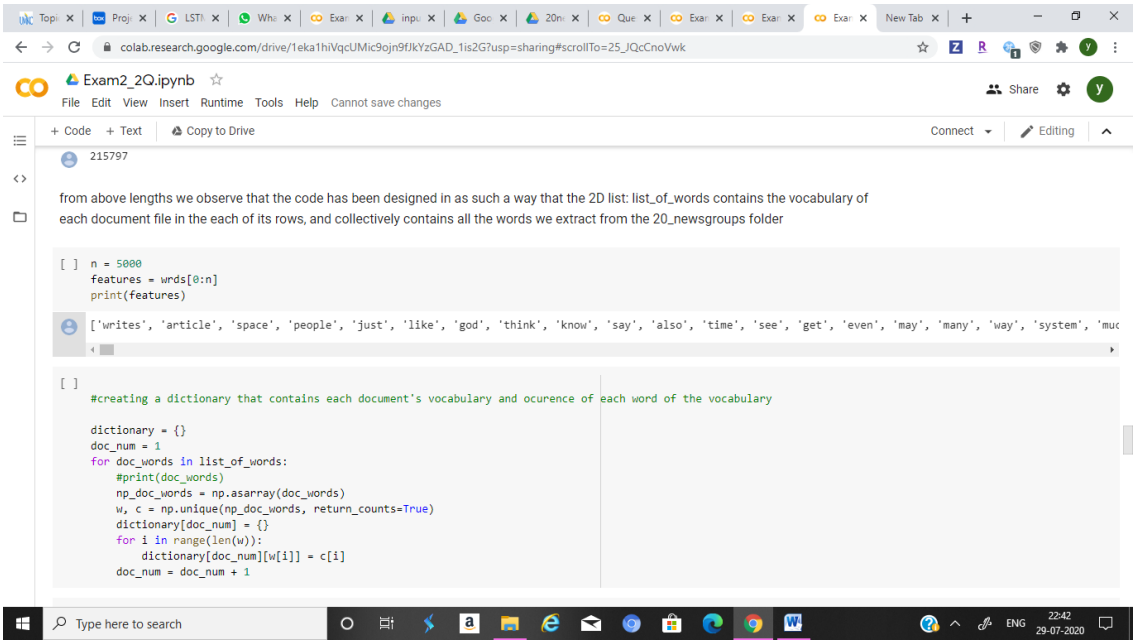


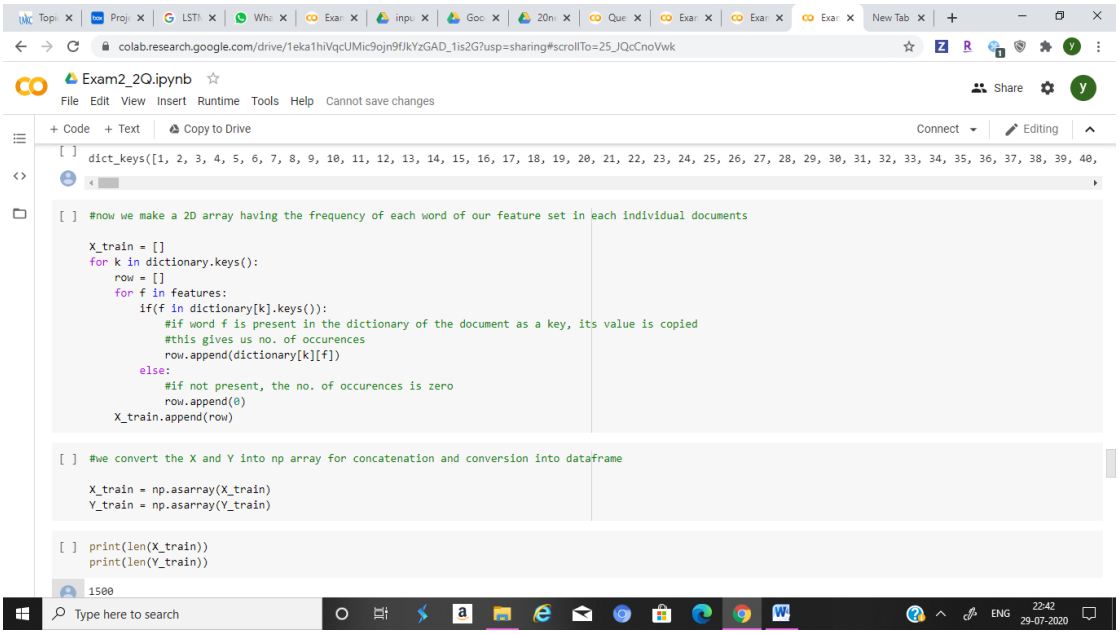


- Later, now encode the data so that all the datatypes of the words will be same later,and we are going to apply the embedding layer In it and then create the model and later that apply the model for the filtered data.

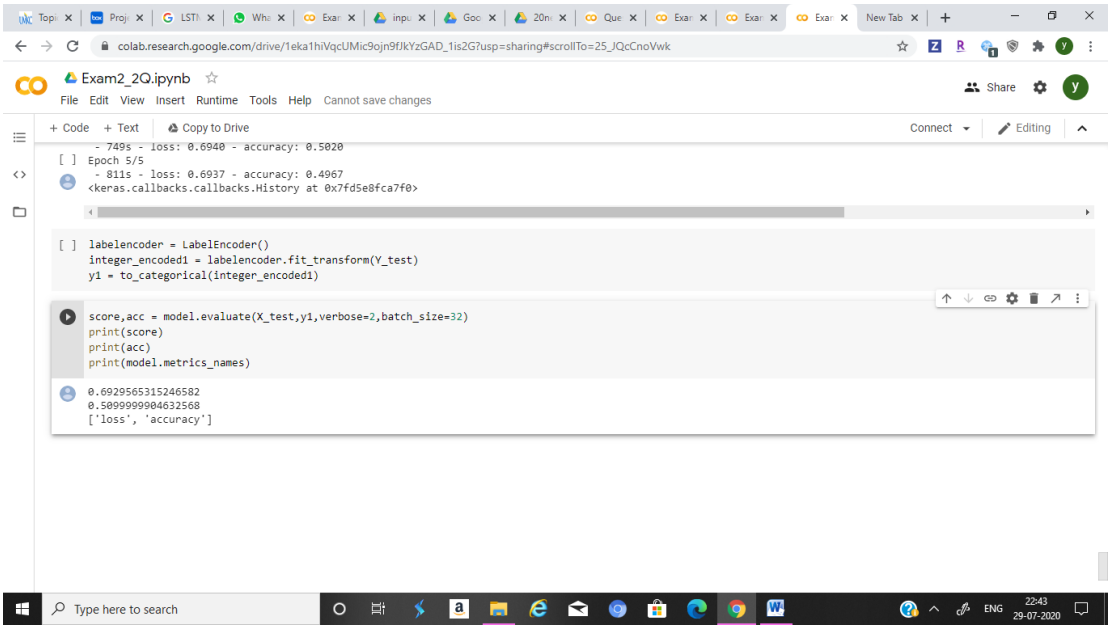

colab.research.google.com/drive/1eka1hiVqcUMic9ojn9fJkYzGAD_1is2G?usp=sharing#scrollTo=25_JQcCnoVwk
Exam2_2Q.ipynb
File Edit View Insert Runtime Tools Help Cannot save changes
Share
+ Code + Text Copy to Drive
Connect Editing
[ ] #function to convert a document into list of words
def tokenize(path):
#load document as a list of lines
f = open(path, 'r',encoding="utf8", errors='ignore')
text_lines = f.readlines()
#removing the meta-data at the top of each document
text_lines = remove_metadata(text_lines)
#initiazing an array to hold all the words in a document
doc_words = []
#traverse over all the lines and tokenize each one with the help of helper function: tokenize_sentence
for line in text_lines:
doc_words.append(tokenize_sentence(line))
return doc_words
[ ] #a simple helper function to convert a 2D array to 1D, without using numpy
def flatten(list):
new_list = []
for i in list:
for j in i:
new_list.append(j)
return new_list
Type here to search
ENG
22:42
29-07-2020


colab.research.google.com/drive/1eka1hiVqcUMic9ojn9fJkYzGAD_1is2G?usp=sharing#scrollTo=25_JQcCnoVwk
Exam2_2Q.ipynb
File Edit View Insert Runtime Tools Help Cannot save changes
Share
+ Code + Text Copy to Drive
Connect Editing
[ ] list_of_words = []
for document in doc_train:
print(document)
list_of_words.append(flatten(tokenize(document)))
/content/drive/My Drive/Colab Notebooks/20newsgroup/sci.space/59848
/content/drive/My Drive/Colab Notebooks/20newsgroup/alt.atheism/54255
/content/drive/My Drive/Colab Notebooks/20newsgroup/sci.space/61358
/content/drive/My Drive/Colab Notebooks/20newsgroup/alt.atheism/53322
/content/drive/My Drive/Colab Notebooks/20newsgroup/alt.atheism/54191
/content/drive/My Drive/Colab Notebooks/20newsgroup/sci.space/60812
/content/drive/My Drive/Colab Notebooks/20newsgroup/sci.space/60248
/content/drive/My Drive/Colab Notebooks/20newsgroup/alt.atheism/54250
/content/drive/My Drive/Colab Notebooks/20newsgroup/alt.atheism/53474
/content/drive/My Drive/Colab Notebooks/20newsgroup/alt.atheism/53126
/content/drive/My Drive/Colab Notebooks/20newsgroup/alt.atheism/53572
/content/drive/My Drive/Colab Notebooks/20newsgroup/sci.space/60216
/content/drive/My Drive/Colab Notebooks/20newsgroup/alt.atheism/53055
/content/drive/My Drive/Colab Notebooks/20newsgroup/alt.atheism/53208
/content/drive/My Drive/Colab Notebooks/20newsgroup/sci.space/61459
/content/drive/My Drive/Colab Notebooks/20newsgroup/alt.atheism/53335
/content/drive/My Drive/Colab Notebooks/20newsgroup/alt.atheism/54194
/content/drive/My Drive/Colab Notebooks/20newsgroup/alt.atheism/53070
/content/drive/My Drive/Colab Notebooks/20newsgroup/alt.atheism/54248
/content/drive/My Drive/Colab Notebooks/20newsgroup/sci.space/60917
Type here to search
ENG
22:42
29-07-2020





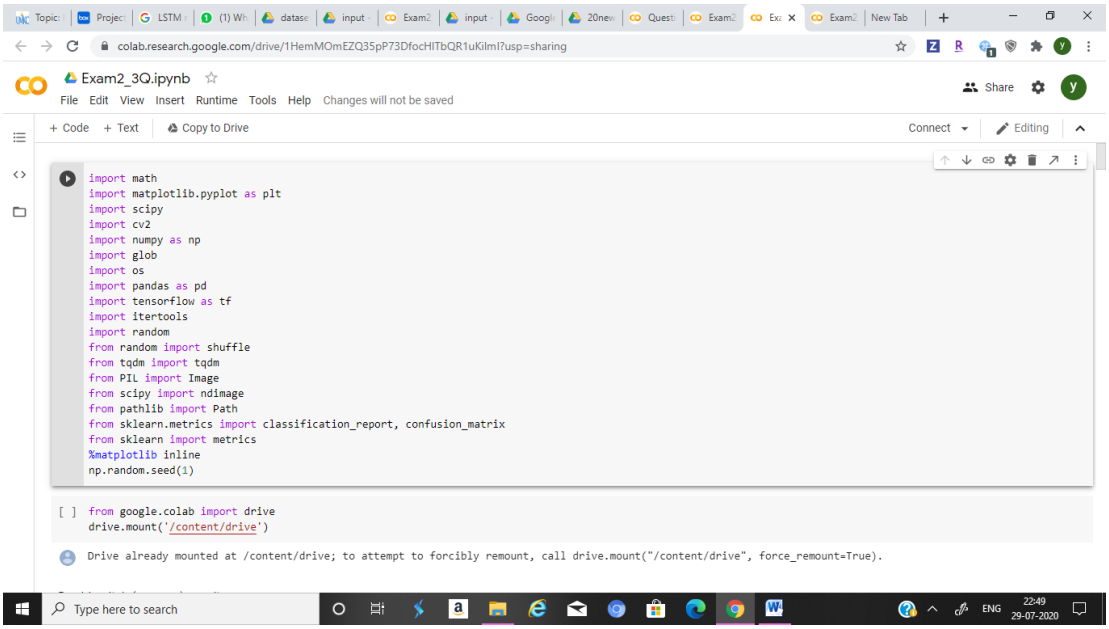
- Later that predict the accuracy and loss.
- And later we are plotting the accuracy and loss using matplot lib.

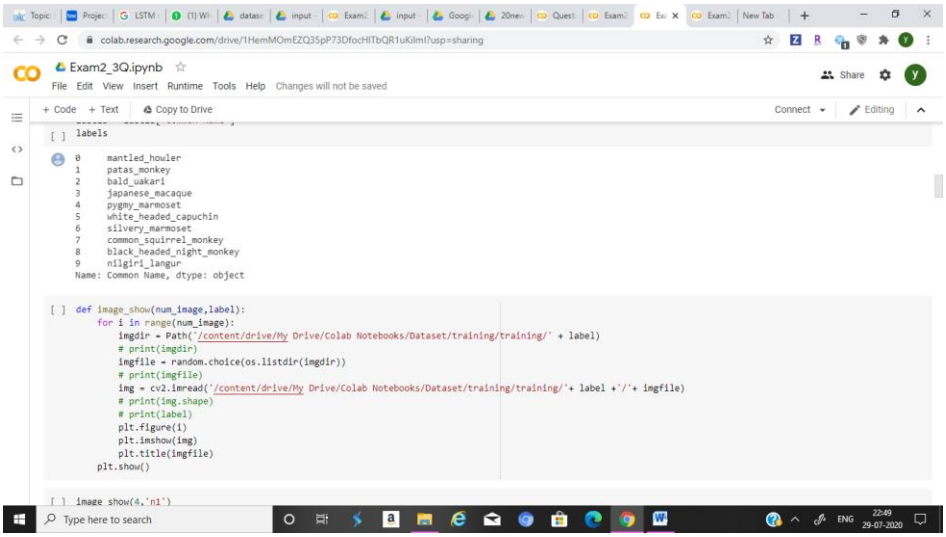


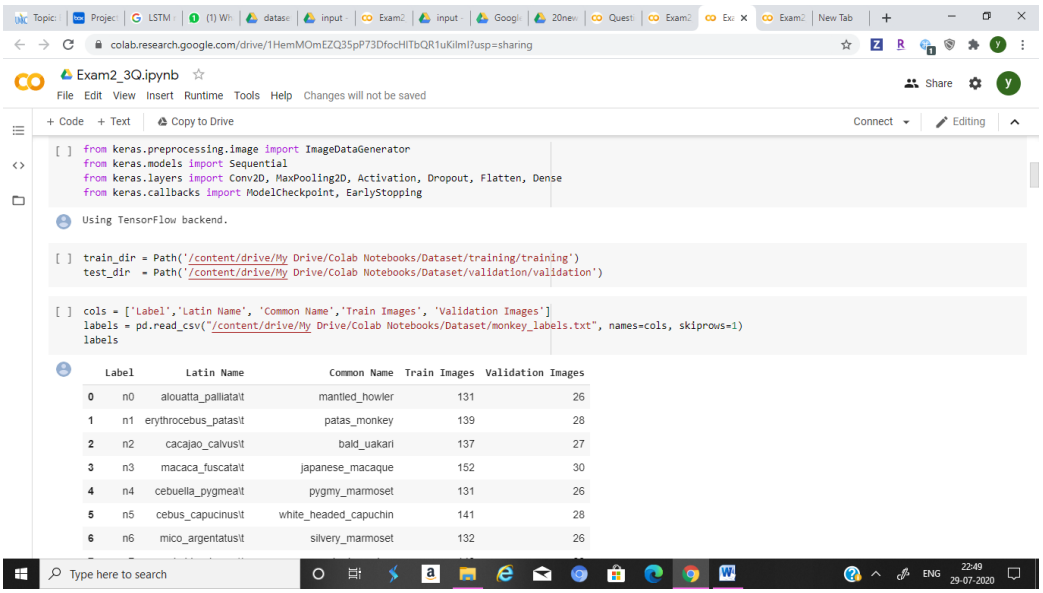
- And we observed a overfitting problem in this.

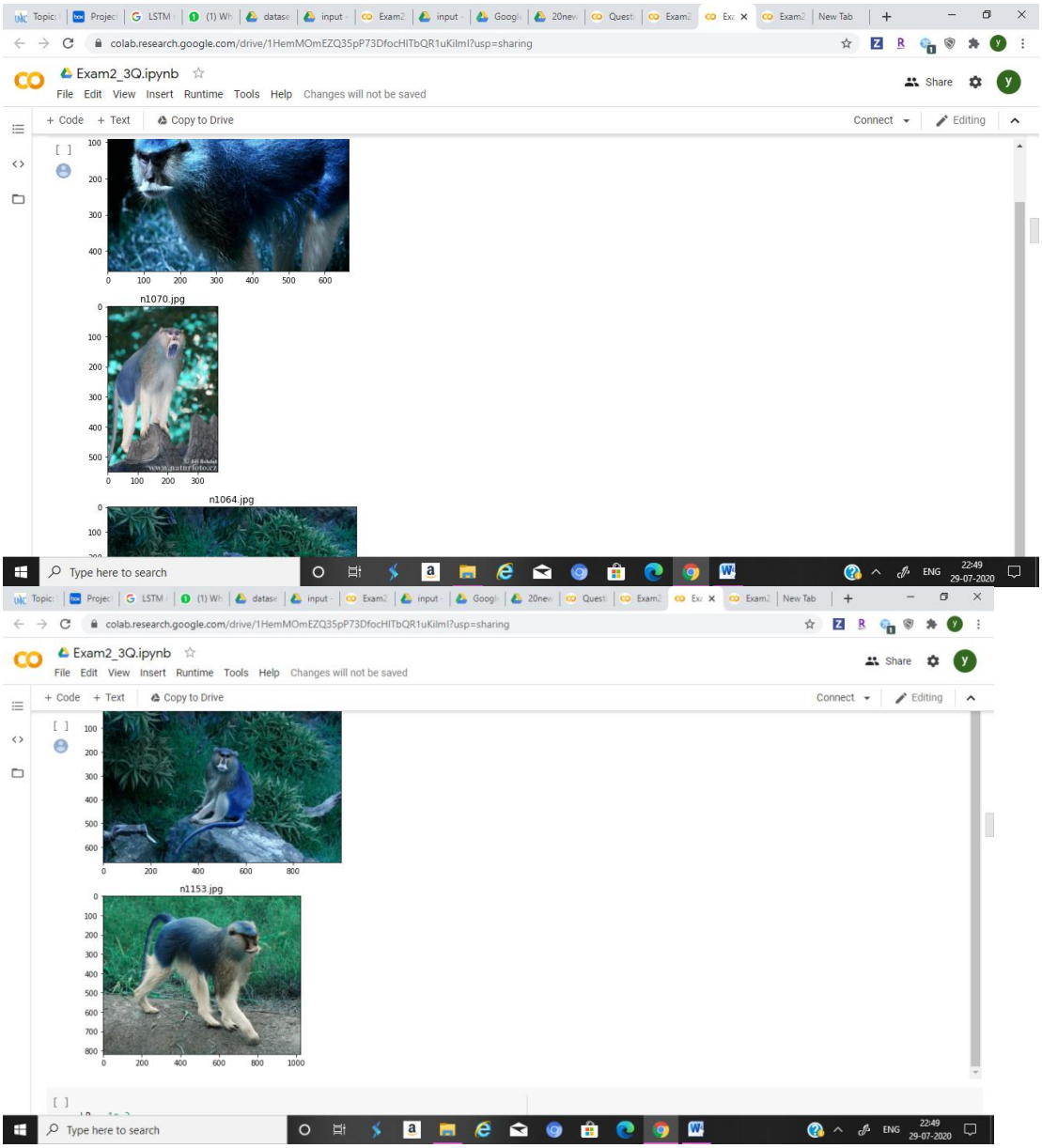
3)

- In this task we are going to perform the image classification using CNN model, so in this we train our model with some images and later our model will predict the test image based on the training image.

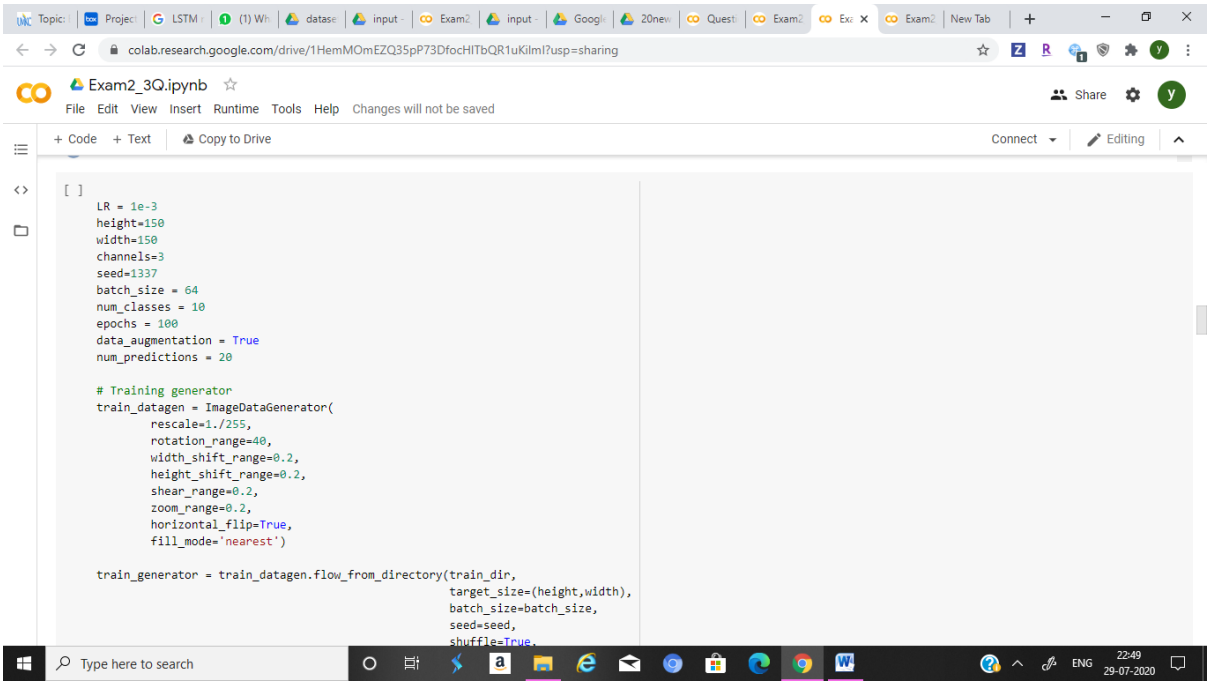


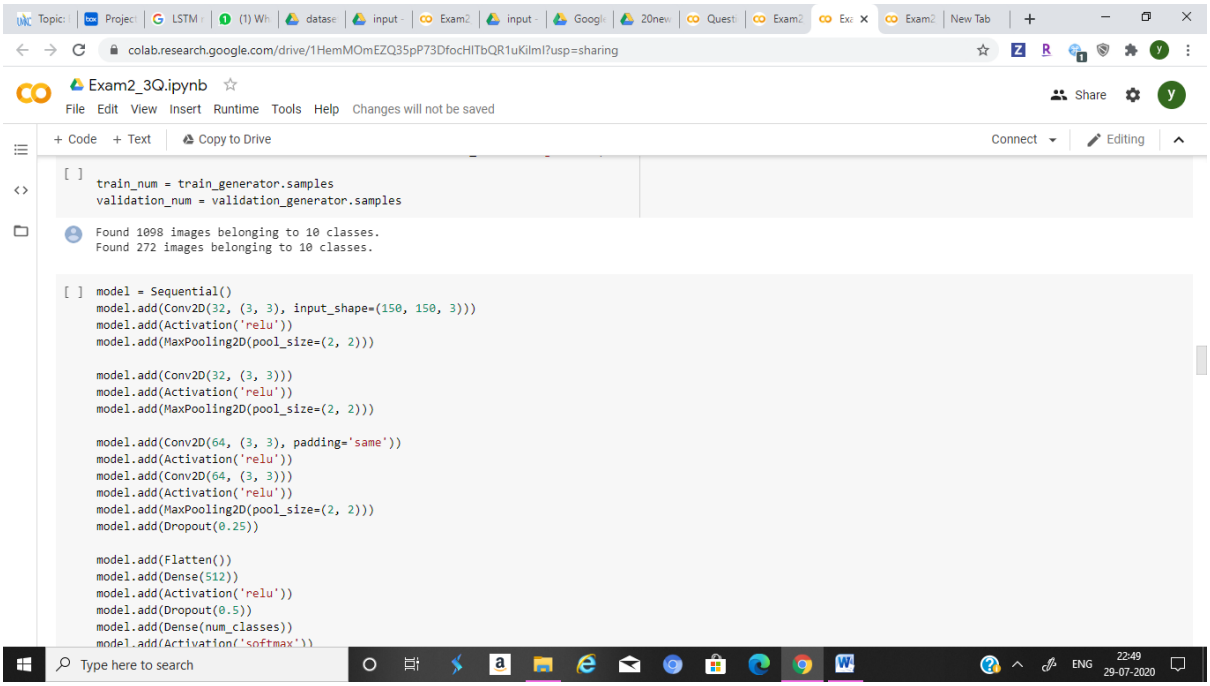

colab.research.google.com/drive/1HemMOmEZQ35pP73DfocHITbQR1uKilmI?usp=sharing
Exam2_3Q.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
Share
+ Code + Text Copy to Drive
Connect Editing
[ ] labels
0 mantled_howler
1 patas_monkey
2 bald_uakari
3 japanese_macaque
4 pygmy_marmoset
5 white_headed_capuchin
6 silvery_marmoset
7 common_squirrel_monkey
8 black_headed_night_monkey
9 nilgiri_langur
Name: Common Name, dtype: object
[ ] def image_show(num_image,label):
for i in range(num_image):
imgdir = Path('/content/drive/My Drive/Colab Notebooks/Dataset/training/training/' + label)
# print(imgdir)
imgfile = random.choice(os.listdir(imgdir))
# print(imgfile)
img = cv2.imread('/content/drive/My Drive/Colab Notebooks/Dataset/training/training/'+ label +'/'+ imgfile)
# print(img.shape)
# print(label)
plt.figure(1)
plt.imshow(img)
plt.title(imgfile)
plt.show()
[ ] image_show(4,'n1')
Type here to search
ENG 22:49 29-07-2020

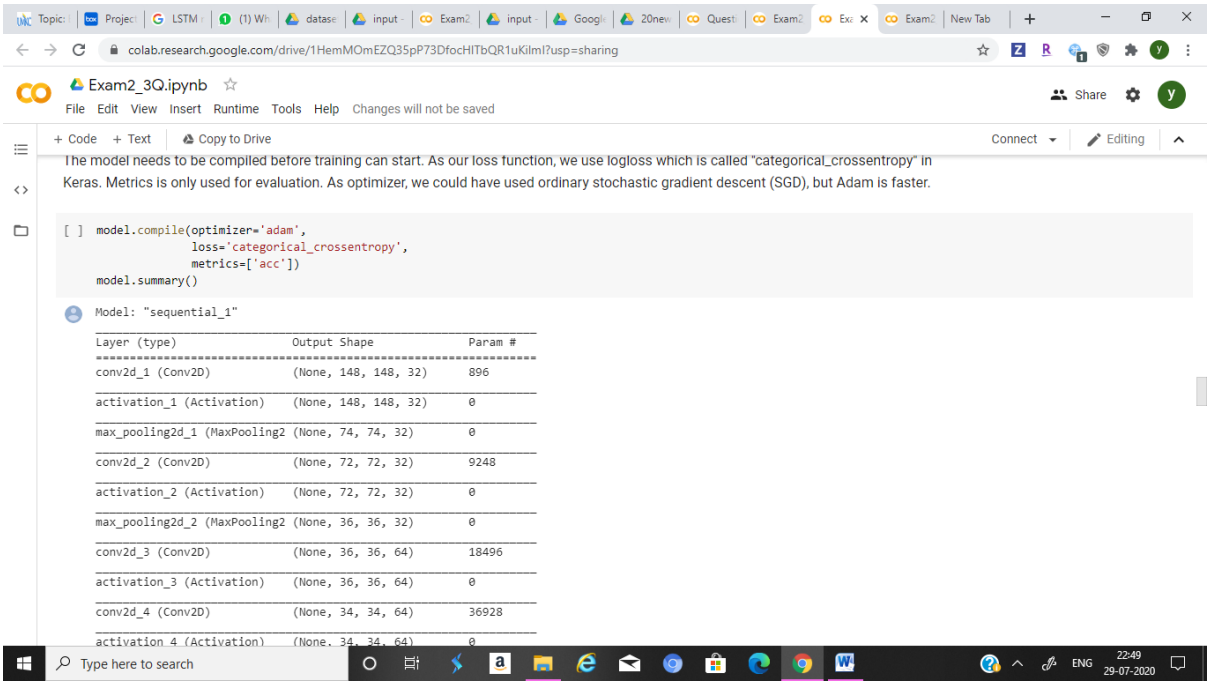

colab.research.google.com/drive/1HemMOmEZQ35pP73DfocHITbQR1uKilmI?usp=sharing
Exam2_3Q.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
Share
+ Code + Text Copy to Drive
Connect Editing
[ ] from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Activation, Dropout, Flatten, Dense
from keras.callbacks import ModelCheckpoint, EarlyStopping
Using TensorFlow backend.
[ ] train_dir = Path('/content/drive/My Drive/Colab Notebooks/Dataset/training/training')
test_dir = Path('/content/drive/My Drive/Colab Notebooks/Dataset/validation/validation')
[ ] cols = ['Label','Latin Name', 'Common Name','Train Images', 'Validation Images']
labels = pd.read_csv("/content/drive/My Drive/Colab Notebooks/Dataset/monkey_labels.txt", names=cols, skiprows=1)
labels
Label Latin Name Common Name Train Images Validation Images
0 n0 alouatta_palliata\t mantled_howler 131 26
1 n1 erythrocebus_patas\t patas_monkey 139 28
2 n2 cacajao_calvus\t bald_uakari 137 27
3 n3 macaca_fuscata\t japanese_macaque 152 30
4 n4 cebuella_pygmea\t pygmy_marmoset 131 26
5 n5 cebus_capucinus\t white_headed_capuchin 141 28
6 n6 mico_argentatus\t silvery_marmoset 132 26
Type here to search
ENG 22:49 29-07-2020



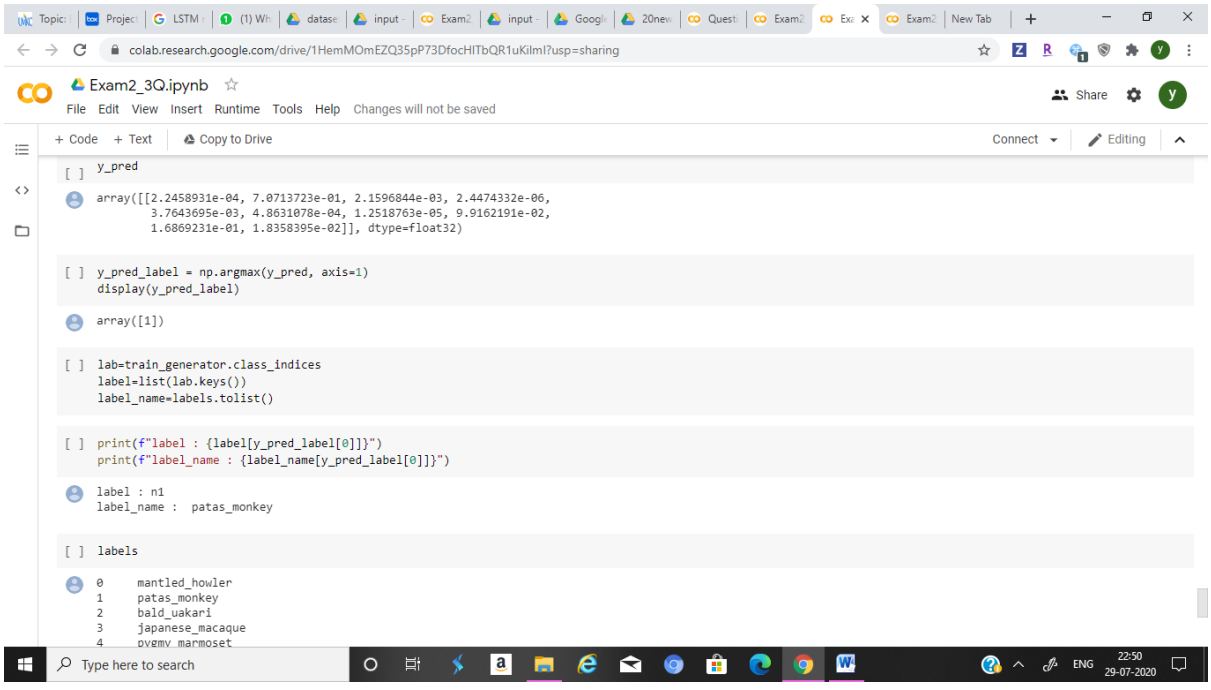
- So, first import all the required packages and later that we are going to read the dataset into a variable,now in this select the columns which you are gonna use for the prediction and select the test column and save them in different variables and print the test column, here we are taking the common name column as the test column.
- Later, now we have to train the model, for that import the images that belong to different categories and later that specify the properties for that certain category, later apply this model on the test image and check the output.

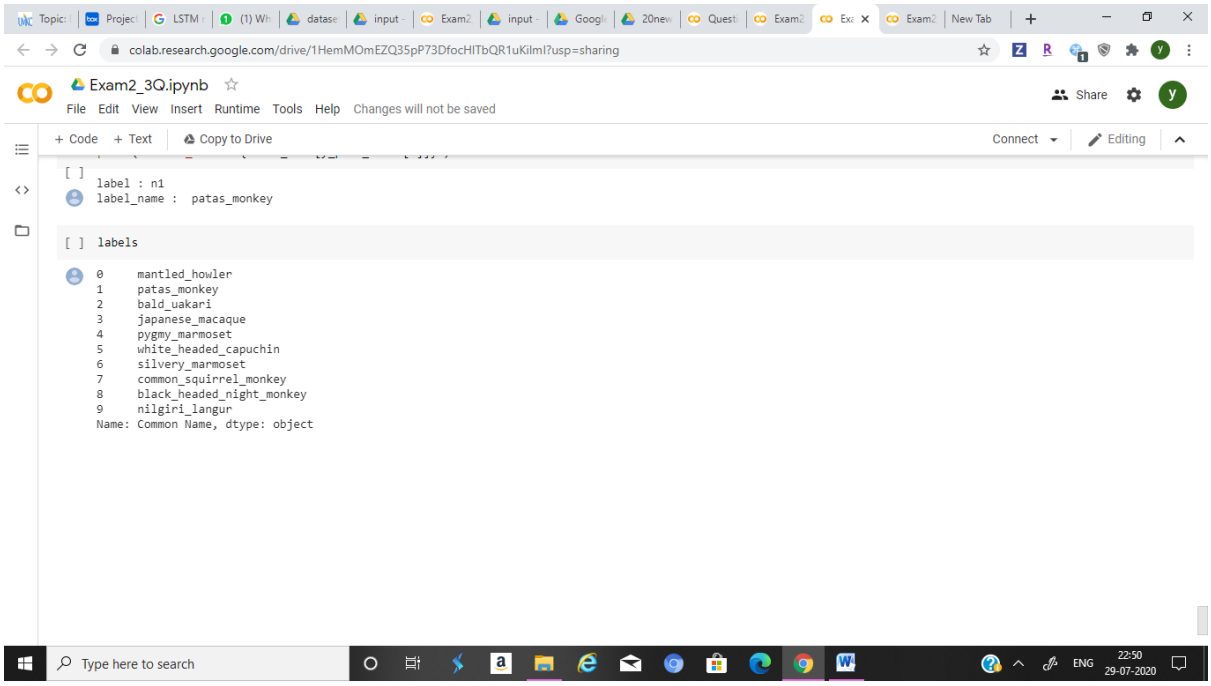

colab.research.google.com/drive/1HemMOmEZQ35pP73DfocHITbQR1uKilml?usp=sharing
Exam2_3Q.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
Share
+ Code + Text Copy to Drive
Connect Editing
[ ]
LR = 1e-3
height=150
width=150
channels=3
seed=1337
batch_size = 64
num_classes = 10
epochs = 100
data_augmentation = True
num_predictions = 20
# Training generator
train_datagen = ImageDataGenerator(
rescale=1./255,
rotation_range=40,
width_shift_range=0.2,
height_shift_range=0.2,
shear_range=0.2,
zoom_range=0.2,
horizontal_flip=True,
fill_mode='nearest')
train_generator = train_datagen.flow_from_directory(train_dir,
target_size=(height,width),
batch_size=batch_size,
seed=seed,
shuffle=True,
Type here to search
ENG
22:49
29-07-2020


colab.research.google.com/drive/1HemMOmEZQ35pP73DfocHITbQR1uKilml?usp=sharing
Exam2_3Q.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
Share
+ Code + Text Copy to Drive
Connect Editing
[ ]
train_num = train_generator.samples
validation_num = validation_generator.samples
Found 1098 images belonging to 10 classes.
Found 272 images belonging to 10 classes.
[ ] model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(150, 150, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes))
model.add(Activation('softmax'))
Type here to search
ENG
22:49
29-07-2020



- And calculate the accuracy and loss and predicted value to the actual value.

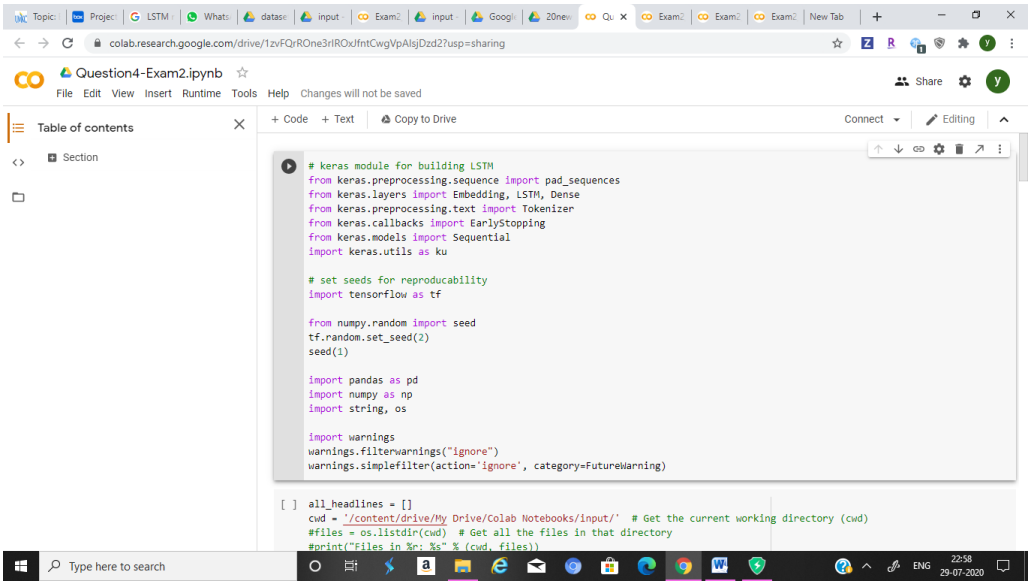




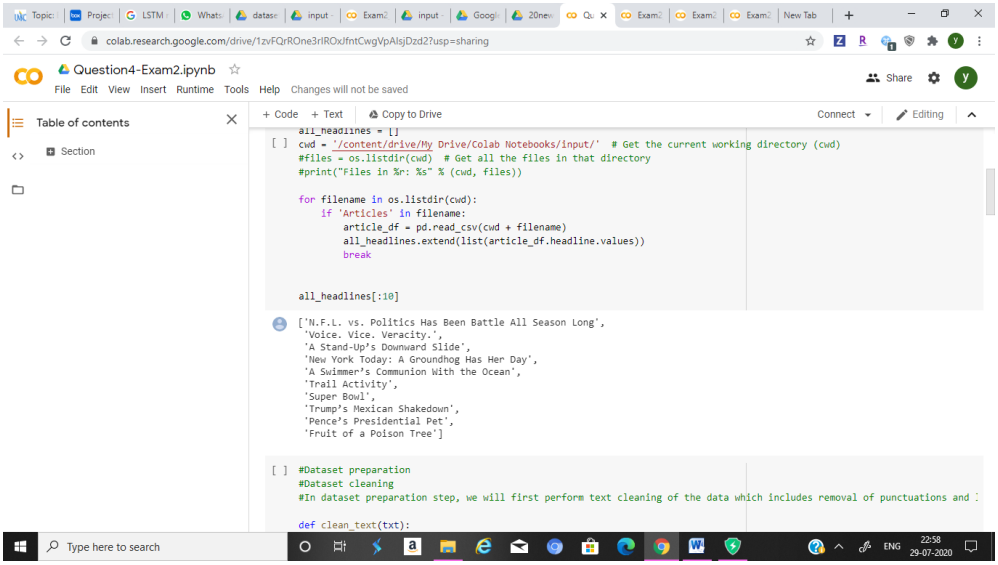
- We are going to plot the accuracy and loss.
- After that apply scaling on the data , so this improves the accuracy because scaling helps the model to consider all the data in an equal value.
- After that calculate the accuracy and loss and plot them.

4)

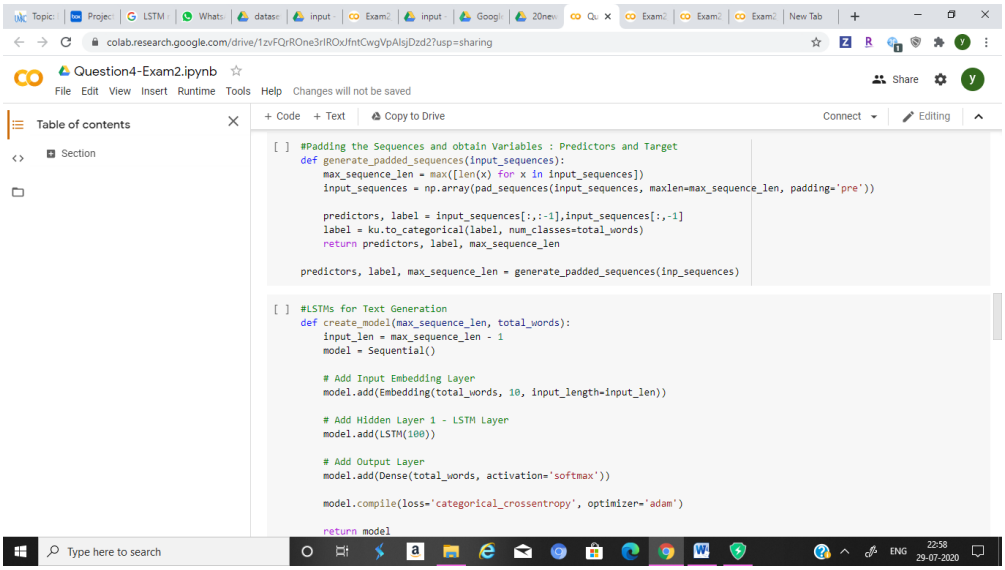
- In this task we are going to train our model on the 20 news groups data headlines and comments and we are going to predict the headline of the dataset.
- For this first import all the packages needed and now we have to import the dataset.

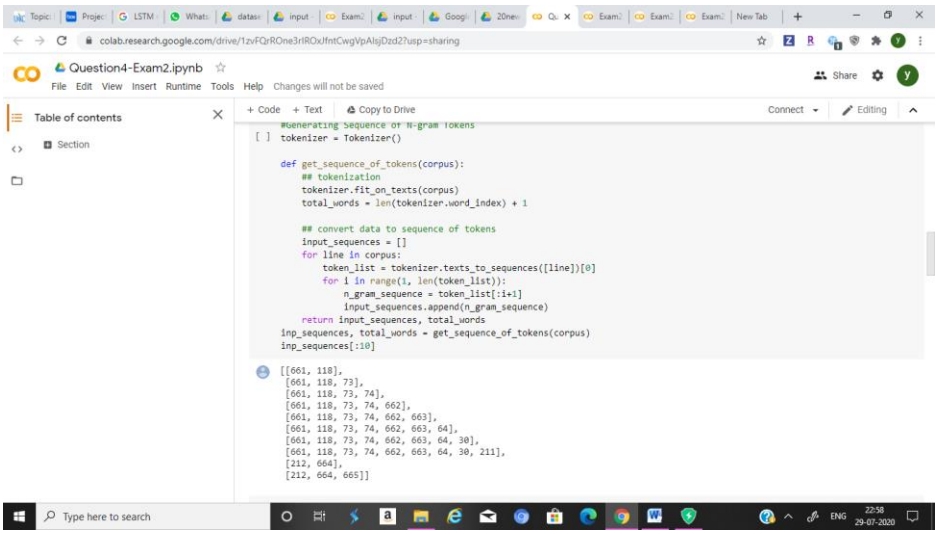


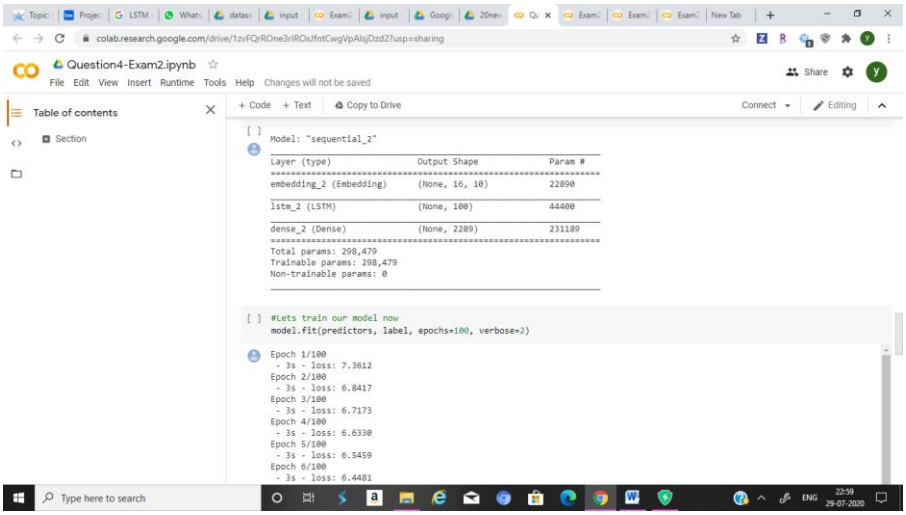
- After that extract all the headlines and comments from the dataset and save them in a folder and we are going to train this data to the model.

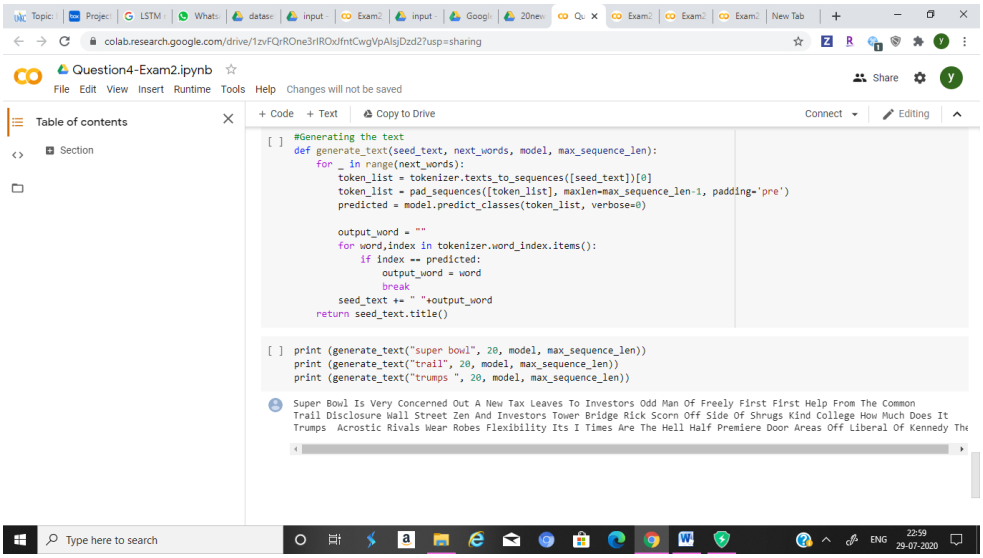


- Now we are going to give the model a sample data and it has to predict the comment and headline for the given data.
- Now, separate the data as train and test data using train test split model.
- After that we have to apply the tokenization to the dataset and later store it, now create a model to predict the data which is a sequential model and feed the trained and test data to the model.


Question4-Exam2.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
Share
Table of contents
Section
+ Code + Text Copy to Drive
Connect Editing
[ ] #Padding the Sequences and obtain Variables : Predictors and Target
def generate_padded_sequences(input_sequences):
max_sequence_len = max([len(x) for x in input_sequences])
input_sequences = np.array(pad_sequences(input_sequences, maxlen=max_sequence_len, padding='pre'))
predictors, label = input_sequences[:,:-1],input_sequences[:,-1]
label = ku.to_categorical(label, num_classes=total_words)
return predictors, label, max_sequence_len
predictors, label, max_sequence_len = generate_padded_sequences(inp_sequences)
[ ] #LSTMs for Text Generation
def create_model(max_sequence_len, total_words):
input_len = max_sequence_len - 1
model = Sequential()
# Add Input Embedding Layer
model.add(Embedding(total_words, 10, input_length=input_len))
# Add Hidden Layer 1 - LSTM Layer
model.add(LSTM(100))
# Add Output Layer
model.add(Dense(total_words, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam')
return model
Type here to search
ENG 22:58 29-07-2020


Question4-Exam2.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
Share
Table of contents
Section
+ Code + Text Copy to Drive
Connect Editing
#Generating Sequence of N-gram Tokens
[ ] tokenizer = Tokenizer()
def get_sequence_of_tokens(corpus):
tokenization
tokenizer.fit_on_texts(corpus)
total_words = len(tokenizer.word_index) + 1
convert data to sequence of tokens
input_sequences = []
for line in corpus:
token_list = tokenizer.texts_to_sequences([line])[0]
for i in range(1, len(token_list)):
n_gram_sequence = token_list[:i+1]
input_sequences.append(n_gram_sequence)
return input_sequences, total_words
inp_sequences, total_words = get_sequence_of_tokens(corpus)
inp_sequences[:10]
[[661, 118],
[661, 118, 73],
[661, 118, 73, 74],
[661, 118, 73, 74, 662],
[661, 118, 73, 74, 662, 663],
[661, 118, 73, 74, 662, 663, 64],
[661, 118, 73, 74, 662, 663, 64, 30],
[661, 118, 73, 74, 662, 663, 64, 30, 211],
[212, 664],
[212, 664, 665]]
Type here to search
ENG 22:58 29-07-2020

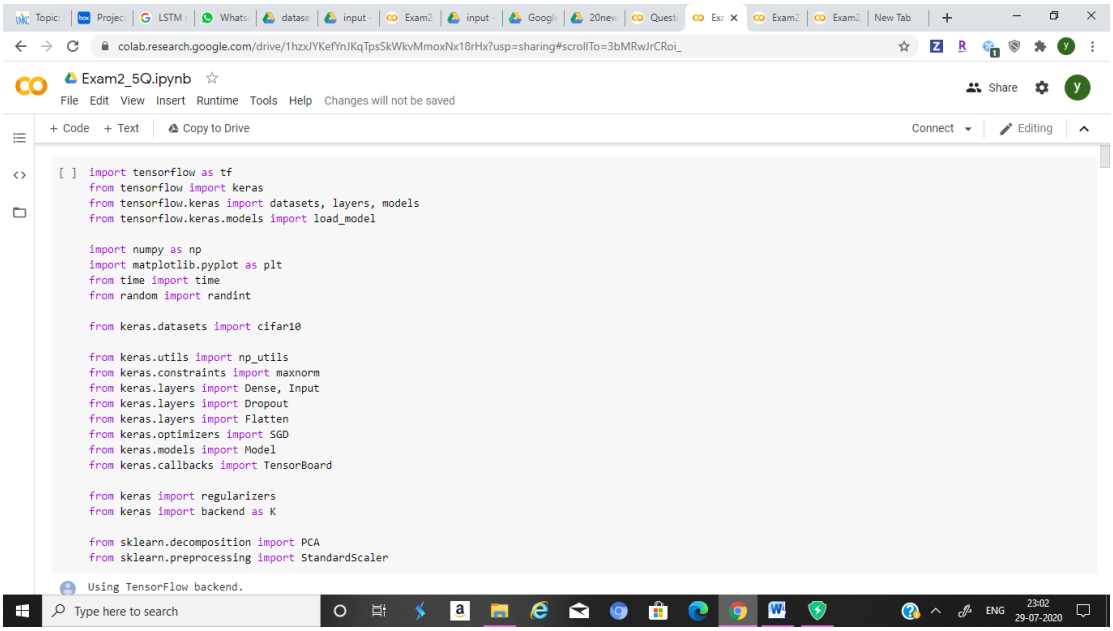

Question4-Exam2.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
Share
Table of contents
Section
+ Code + Text Copy to Drive
Connect Editing
[ ]
Model: "sequential_2"
Layer (type) Output Shape Param #
embedding_2 (Embedding) (None, 16, 10) 22890
lstm_2 (LSTM) (None, 100) 44400
dense_2 (Dense) (None, 2289) 231189
Total params: 298,479
Trainable params: 298,479
Non-trainable params: 0
[ ] #Lets train our model now
model.fit(predictors, label, epochs=100, verbose=2)
Epoch 1/100
- 3s - loss: 7.3612
Epoch 2/100
- 3s - loss: 6.8417
Epoch 3/100
- 3s - loss: 6.7173
Epoch 4/100
- 3s - loss: 6.6330
Epoch 5/100
- 3s - loss: 6.5459
Epoch 6/100
- 3s - loss: 6.4481
Type here to search
ENG 22:59 29-07-2020

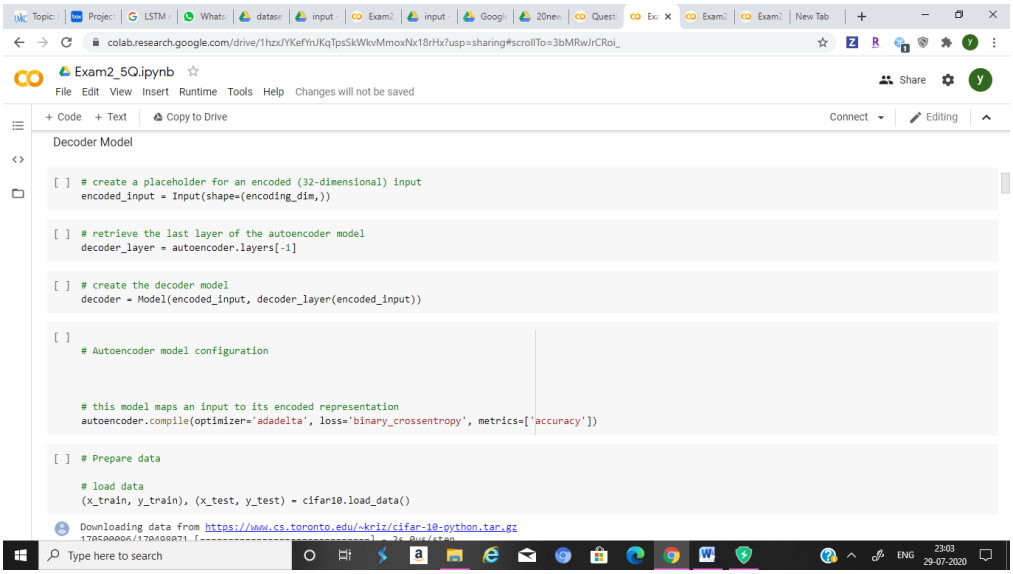


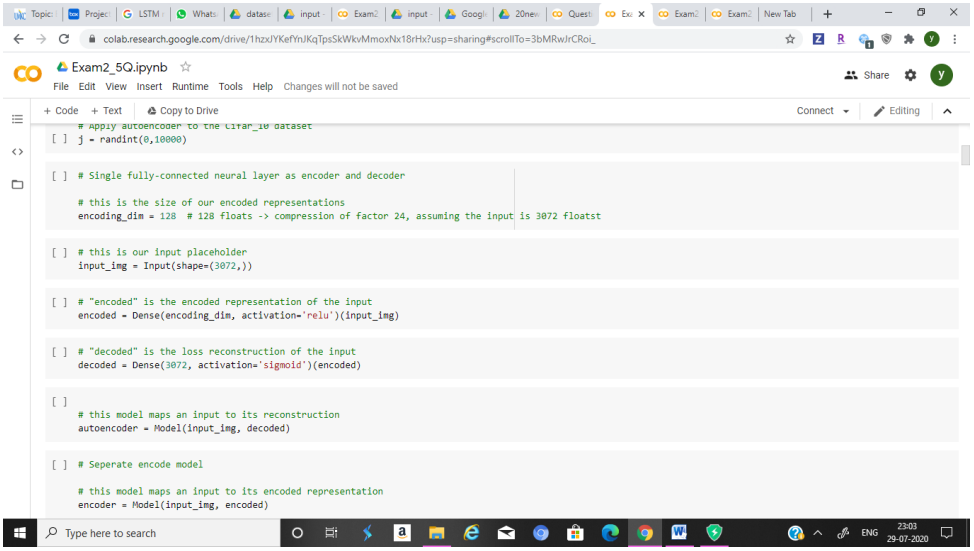
- Calculate the accuracy of the predicted data and observe the output.

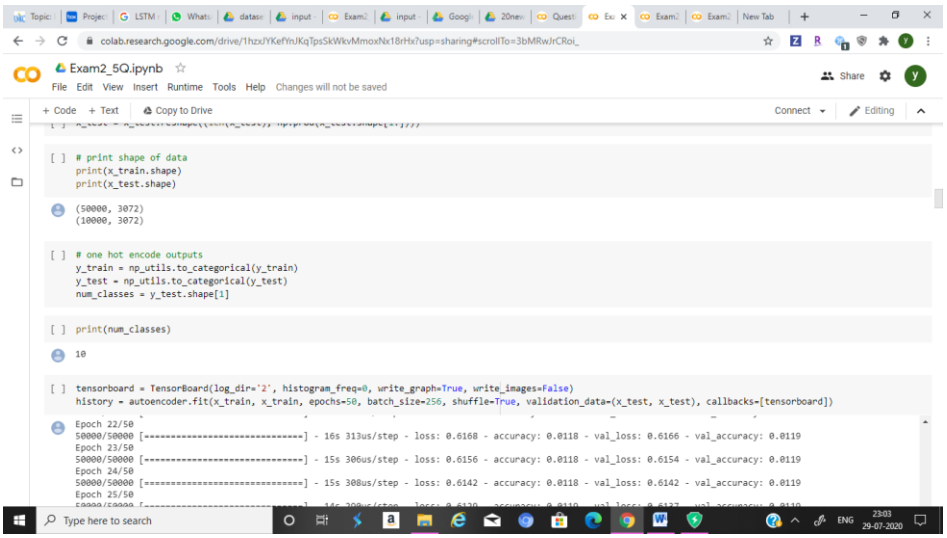
5)

- In this we are going to apply autoencoding on the data set given and later we are going to feed this data any sequential classifying model and observe the accuracy and later that we are going to apply PCA model on the data and apply the sequential model later and observe the best one.
- First we have to import all the packages needed and later that we should create a variable and load and read the dataset into it,

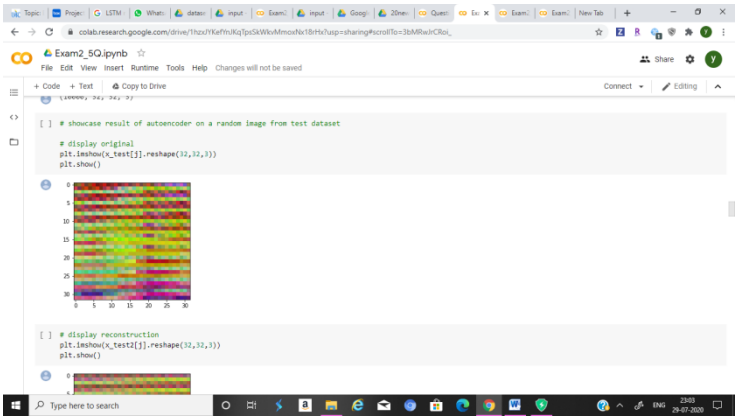


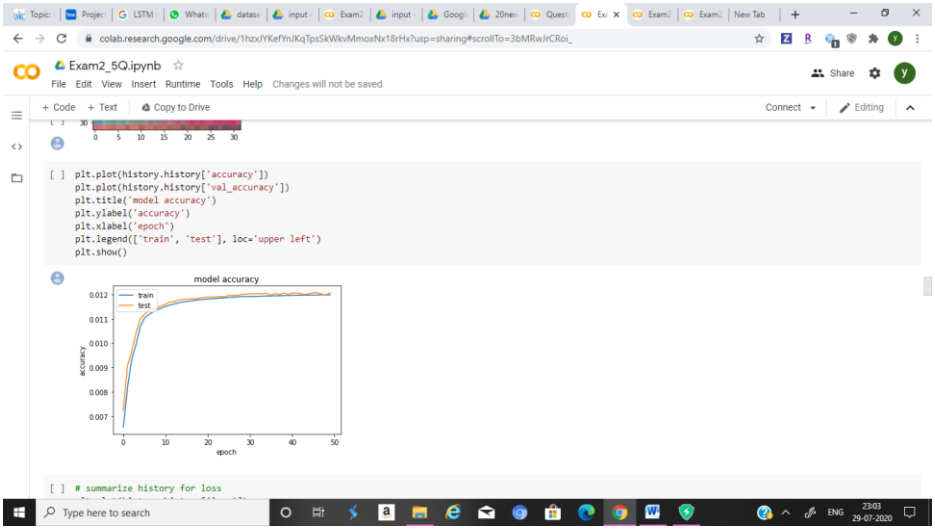

colab.research.google.com/drive/1hzxJYKefYnJKqTpsSkWkvMmoxNx18rHx?usp=sharing#scrollTo=3bMRwJrCRoi_
Exam2_5Q.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
Share
+ Code + Text Copy to Drive
Connect Editing
Decoder Model
# create a placeholder for an encoded (32-dimensional) input
encoded_input = Input(shape=(encoding_dim,))
# retrieve the last layer of the autoencoder model
decoder_layer = autoencoder.layers[-1]
# create the decoder model
decoder = Model(encoded_input, decoder_layer(encoded_input))
# Autoencoder model configuration
# this model maps an input to its encoded representation
autoencoder.compile(optimizer='adadelta', loss='binary_crossentropy', metrics=['accuracy'])
# Prepare data
# load data
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
Type here to search
ENG
23:03
29-07-2020

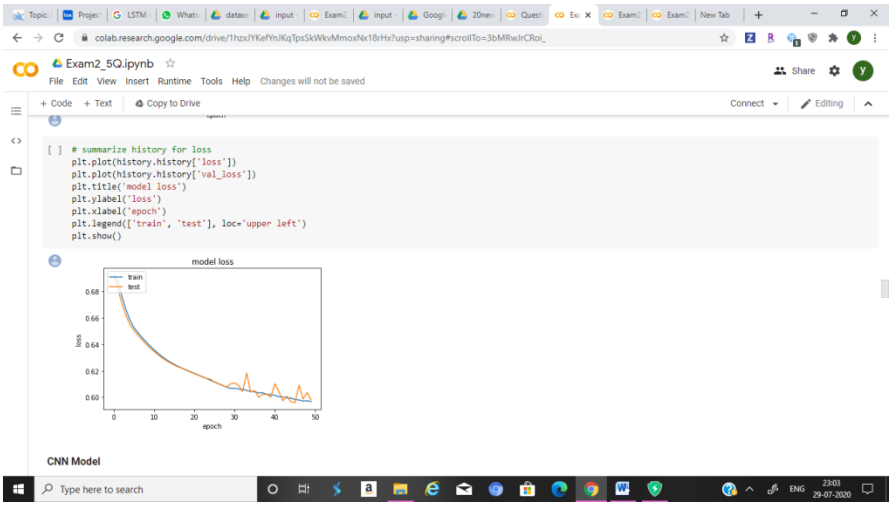

colab.research.google.com/drive/1hzxJYKefYnJKqTpsSkWkvMmoxNx18rHx?usp=sharing#scrollTo=3bMRwJrCRoi_
Exam2_5Q.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
Share
+ Code + Text Copy to Drive
Connect Editing
j = randint(0,10000)
# Single fully-connected neural layer as encoder and decoder
# this is the size of our encoded representations
encoding_dim = 128 # 128 floats -> compression of factor 24, assuming the input is 3072 floatst
# this is our input placeholder
input_img = Input(shape=(3072,))
# "encoded" is the encoded representation of the input
encoded = Dense(encoding_dim, activation='relu')(input_img)
# "decoded" is the loss reconstruction of the input
decoded = Dense(3072, activation='sigmoid')(encoded)
# this model maps an input to its reconstruction
autoencoder = Model(input_img, decoded)
# Seperate encode model
# this model maps an input to its encoded representation
encoder = Model(input_img, encoded)
Type here to search
ENG
23:03
29-07-2020

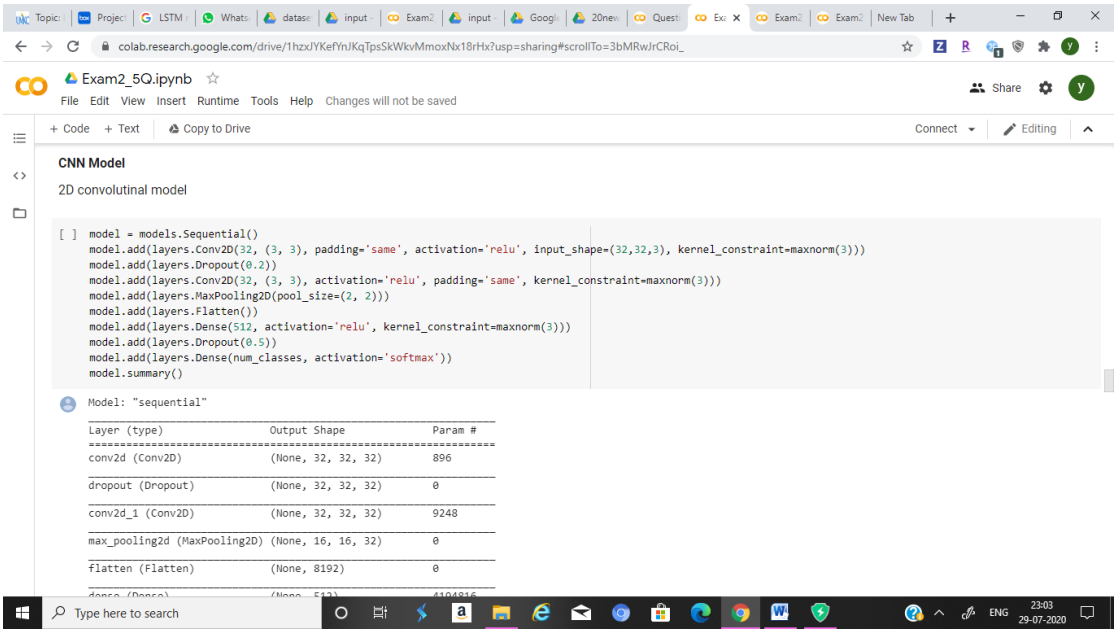

colab.research.google.com/drive/1hzxJYKefYnJKqTpsSkWkvMmoxNx18rHx?usp=sharing#scrollTo=3bMRwJrCRoi_
Exam2_5Q.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
Share
+ Code + Text Copy to Drive
Connect Editing
# print shape of data
print(x_train.shape)
print(x_test.shape)
(50000, 3072)
(10000, 3072)
# one hot encode outputs
y_train = np_utils.to_categorical(y_train)
y_test = np_utils.to_categorical(y_test)
num_classes = y_test.shape[1]
print(num_classes)
10
tensorboard = TensorBoard(log_dir='2', histogram_freq=0, write_graph=True, write_images=False)
history = autoencoder.fit(x_train, x_train, epochs=50, batch_size=256, shuffle=True, validation_data=(x_test, x_test), callbacks=[tensorboard])
Epoch 22/50
50000/50000 [==============================] - 16s 313us/step - loss: 0.6168 - accuracy: 0.0118 - val_loss: 0.6166 - val_accuracy: 0.0119
Epoch 23/50
50000/50000 [==============================] - 15s 306us/step - loss: 0.6156 - accuracy: 0.0118 - val_loss: 0.6154 - val_accuracy: 0.0119
Epoch 24/50
50000/50000 [==============================] - 15s 308us/step - loss: 0.6142 - accuracy: 0.0118 - val_loss: 0.6142 - val_accuracy: 0.0119
Epoch 25/50
Type here to search
ENG
23:03
29-07-2020

- Now we have to apply auto encoder on this data which helps to convert each datatype into a similar one so the model doesn’t create preferences among the datatypes.
- Later we are going to divide the data into train data and test data, now we are going to apply CNN model on the data and we are going to predict the images and later that we are going to calculate the accuracy.

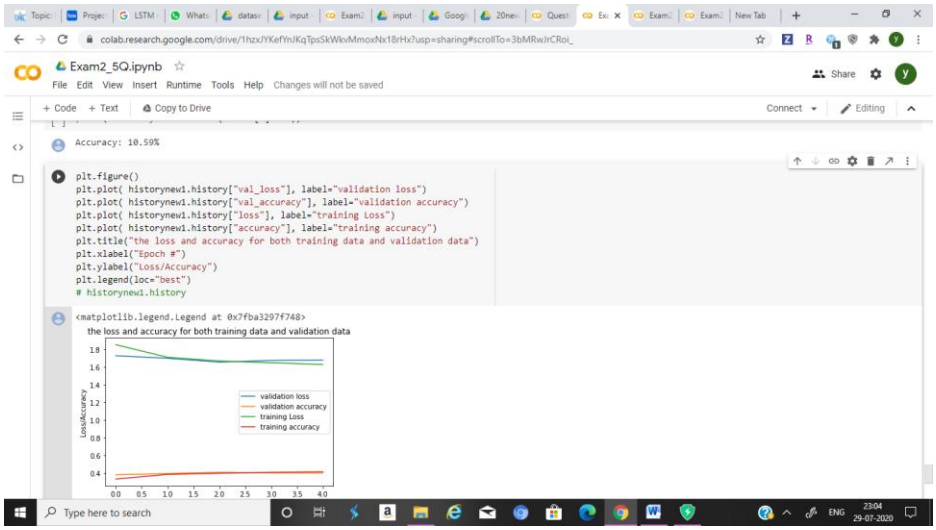


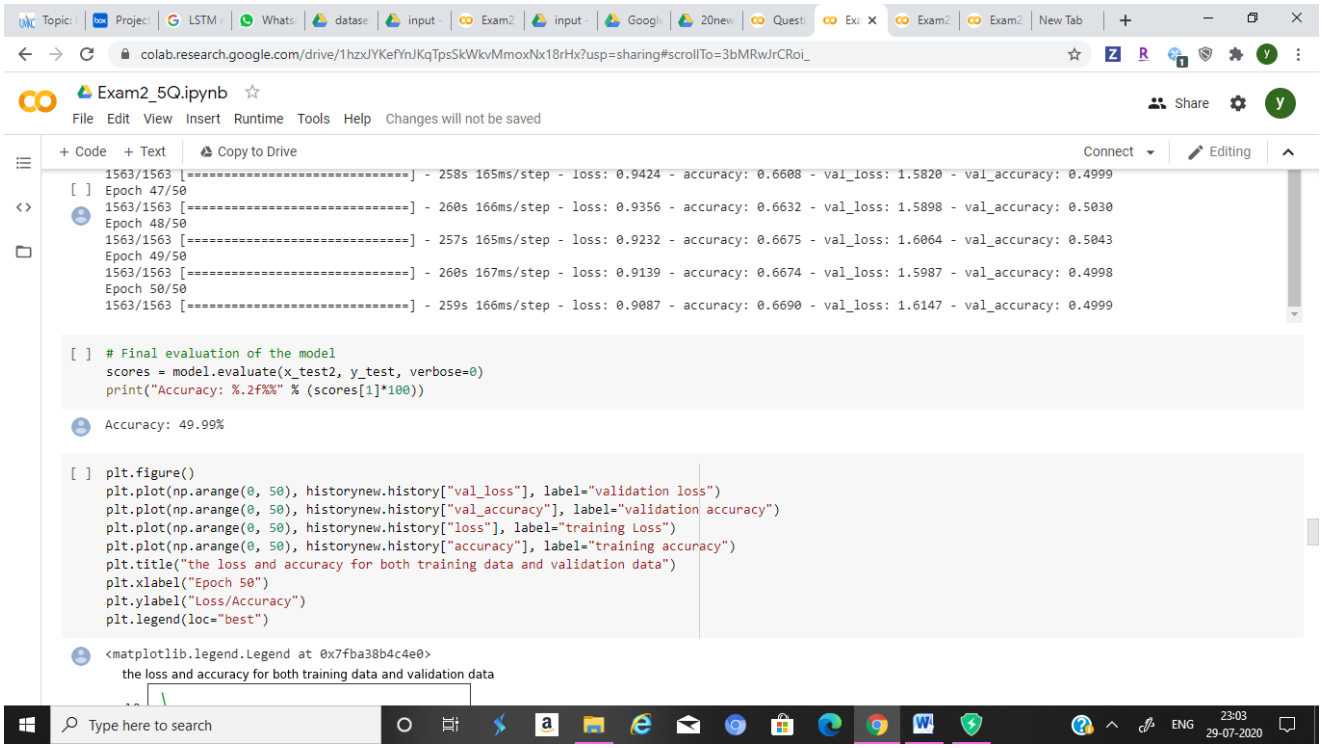


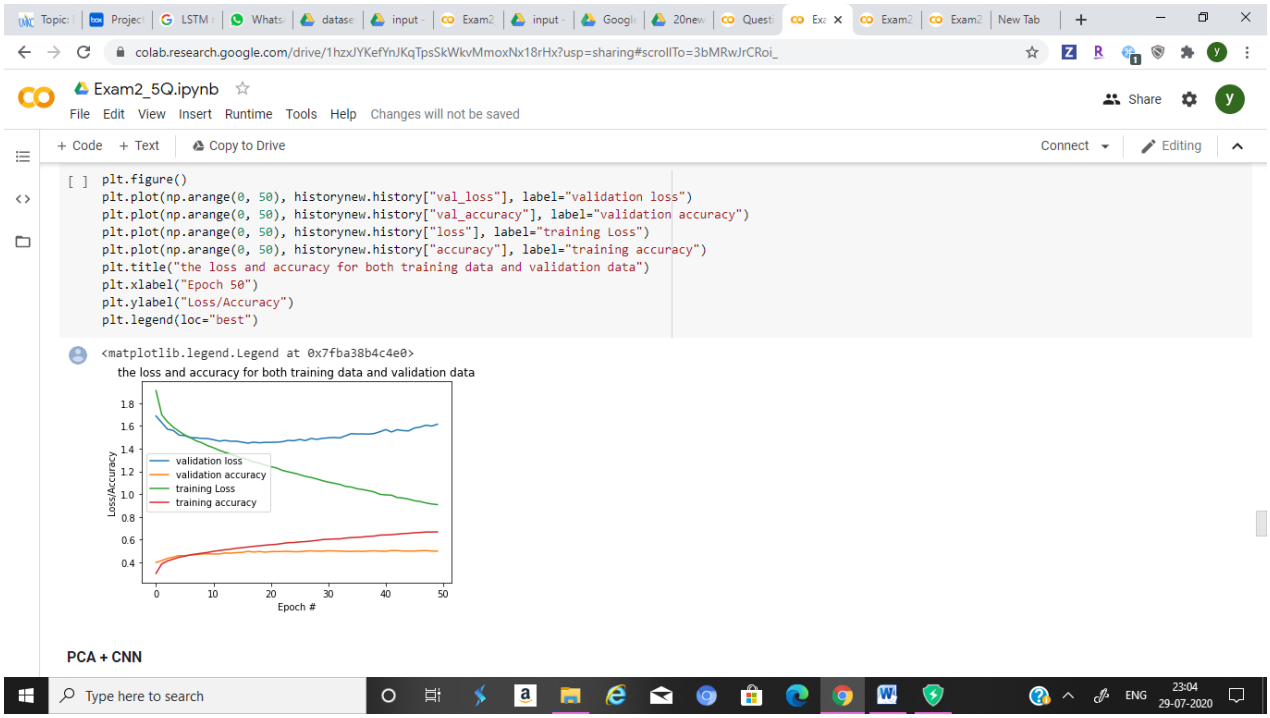


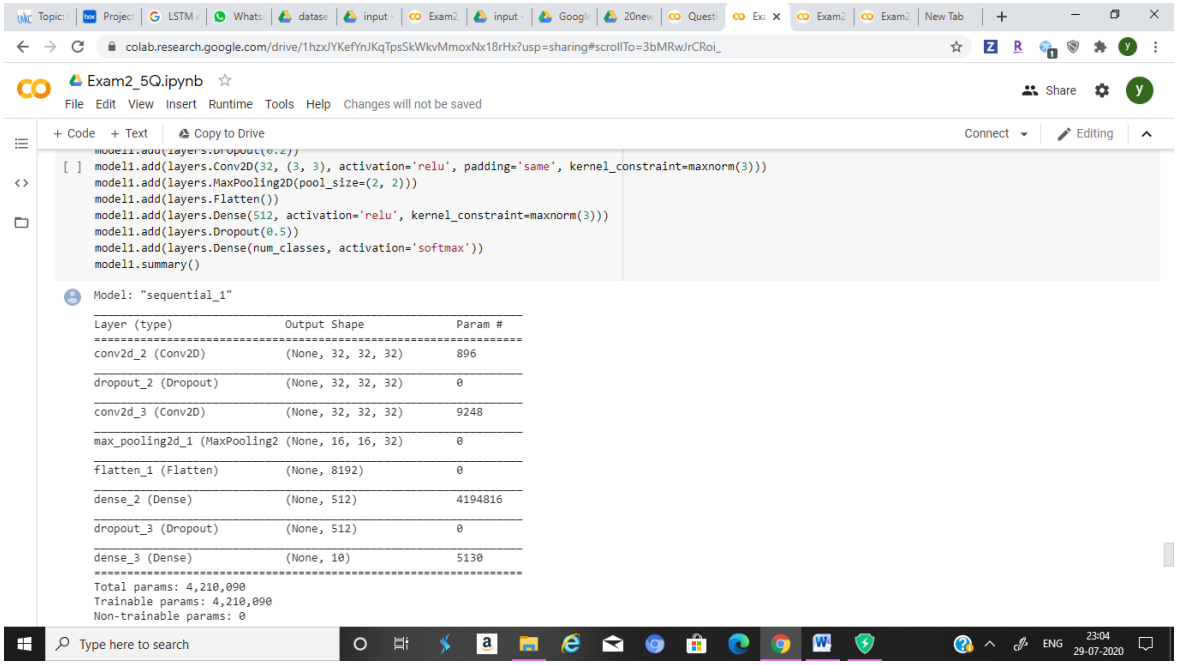


- And we are going to plot the accuracy and loss.
- Now we are going to apply PCA on the dataset and later we are going to feed this data onto CNN model and calculate the accuracy and loss again and compare the plots in the both cases and decide on which is a good model.


Exam2_5Q.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
+ Code + Text Copy to Drive
Connect Editing
Accuracy: 10.59%
plt.figure()
plt.plot(historynew1.history["val_loss"], label="validation loss")
plt.plot(historynew1.history["val_accuracy"], label="validation accuracy")
plt.plot(historynew1.history["loss"], label="training Loss")
plt.plot(historynew1.history["accuracy"], label="training accuracy")
plt.title("the loss and accuracy for both training data and validation data")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="best")
# historynew1.history
<matplotlib.legend.Legend at 0x7fba3297f748>
the loss and accuracy for both training data and validation data
validation loss
validation accuracy
training Loss
training accuracy
Loss/Accuracy
Type here to search
ENG 23:04 29-07-2020


colab.research.google.com/drive/1hzxJYKefYnJKqTpsSkWkvMmoxNx18rHx?usp=sharing#scrollTo=3bMRwJrCRoi_
Exam2_5Q.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
Share
+ Code + Text Copy to Drive
Connect Editing
1563/1563 [==========] - 258s 165ms/step - loss: 0.9424 - accuracy: 0.6608 - val_loss: 1.5820 - val_accuracy: 0.4999
Epoch 47/50
1563/1563 [==========] - 260s 166ms/step - loss: 0.9356 - accuracy: 0.6632 - val_loss: 1.5898 - val_accuracy: 0.5030
Epoch 48/50
1563/1563 [==========] - 257s 165ms/step - loss: 0.9232 - accuracy: 0.6675 - val_loss: 1.6064 - val_accuracy: 0.5043
Epoch 49/50
1563/1563 [==========] - 260s 167ms/step - loss: 0.9139 - accuracy: 0.6674 - val_loss: 1.5987 - val_accuracy: 0.4998
Epoch 50/50
1563/1563 [==========] - 259s 166ms/step - loss: 0.9087 - accuracy: 0.6690 - val_loss: 1.6147 - val_accuracy: 0.4999
# Final evaluation of the model
scores = model.evaluate(x_test2, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
Accuracy: 49.99%
plt.figure()
plt.plot(np.arange(0, 50), historynew.history["val_loss"], label="validation loss")
plt.plot(np.arange(0, 50), historynew.history["val_accuracy"], label="validation accuracy")
plt.plot(np.arange(0, 50), historynew.history["loss"], label="training Loss")
plt.plot(np.arange(0, 50), historynew.history["accuracy"], label="training accuracy")
plt.title("the loss and accuracy for both training data and validation data")
plt.xlabel("Epoch 50")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="best")
<matplotlib.legend.Legend at 0x7fba38b4c4e0>
the loss and accuracy for both training data and validation data
Type here to search
ENG 23:03 29-07-2020





- We observed that the accuracy is more when we do apply the auto encoder and then sequential model on the dataset.

DATASETS:

The datasets we are going to use in this project are review's sentiment dataset,20news groups and 10-monkey species and Cifar_10 dataset and in the text classification task we are going to use information from Newyork Times Headlines and comments.

1st Question:

https://drive.google.com/drive/folders/1StULLg3upQ24LoxDirsqSBVXzg1Q_Cij

2nd Question:

https://drive.google.com/drive/folders/1apbbjmYR_BBzl9mT2ZygSyCVPqp-6LjV

3rd Question:

https://drive.google.com/drive/folders/1wYGvd62UFyeoyq3kOlIspPti48KhqEP0

4th Question

https://drive.google.com/drive/folders/1pydNdvIxDujpfb3eJqD1W5m-o58gkfsA

5th Question

From the library

PARAMETERS:

The parameters we are going to take into consideration in order to evaluate the precision of our model and to figure out which model is best we are considering accuracy and loss.

EVALUATION AND DISCUSSION:

- While doing this project we observed that embedding layer and autoencoding and scaling helps with better accuracy.
- We did faced with the occurrence of overfitting in the data which doesn't really help with the data prediction and we did discuss on how to remove this and we figured out that using learning rate will eliminate this

CONCLUSION:

In this project we implemented sequential classification models on different data sets ,we did image classification and text classification and we observed that embedding layer, auto encoding improves the accuracy of the data prediction and when compared to CNN model LSTM has better results and in machine learning LSTM is the mostly use model in the neural networks.

VIDEO LINK:

https://umkc.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=1982343f-f8ef-4a88-9c50-ac080034d724

GITHUB Source code link:

https://github.com/RoshiniVarada/PythonProject2