

NEURAL NETWORK ASSIGNMENT 5

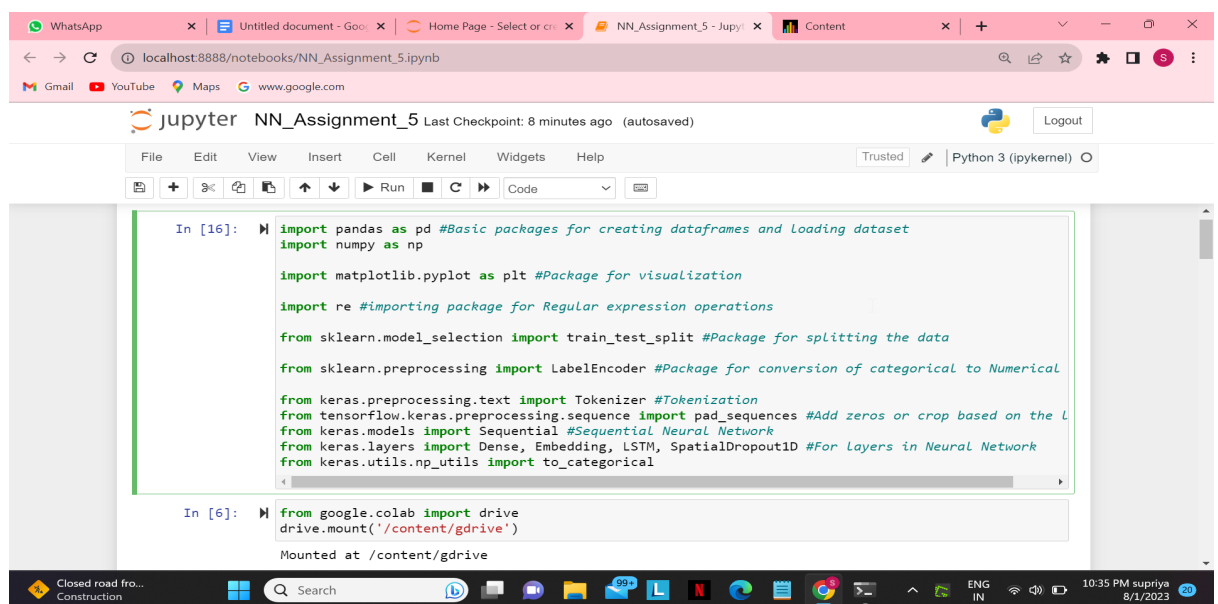
NAME: SUPRIYA SAMA

700744510

Video Link:

<https://drive.google.com/file/d/1t9Rs4KIZKY4bTdyQJWGV4l-CGt8R1s5c/view?usp=sharing>

1. I have imported the pandas, numpy, matplotlib and other packages.



The screenshot shows a Jupyter Notebook titled 'NN_Assignment_5' running on a local host. The code in the notebook is as follows:

```
In [16]: import pandas as pd #Basic packages for creating dataframes and loading dataset
import numpy as np

import matplotlib.pyplot as plt #Package for visualization

import re #importing package for Regular expression operations

from sklearn.model_selection import train_test_split #Package for splitting the data

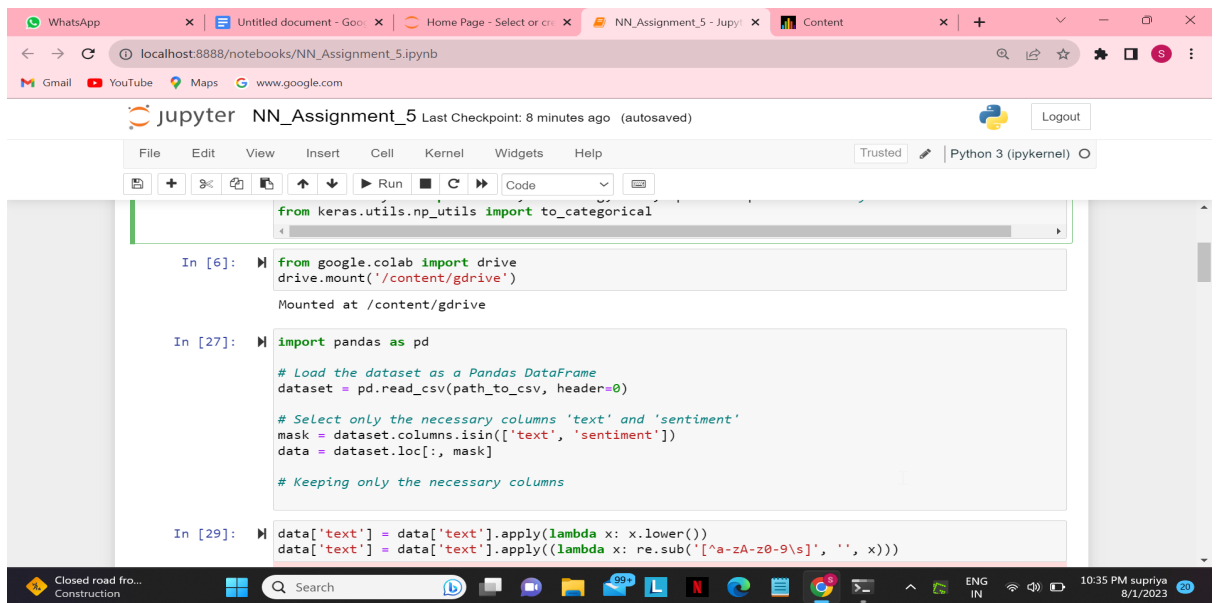
from sklearn.preprocessing import LabelEncoder #Package for conversion of categorical to Numerical

from keras.preprocessing.text import Tokenizer #Tokenization
from tensorflow.keras.preprocessing.sequence import pad_sequences #Add zeros or crop based on the L
from keras.models import Sequential #Sequential Neural Network
from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D #For Layers in Neural Network
from keras.utils.np_utils import to_categorical

In [6]: from google.colab import drive
drive.mount('/content/gdrive')

Mounted at /content/gdrive
```

Let's load the dataset as pandas Dataframe.



The screenshot shows a Jupyter Notebook interface with the following code cells:

```
from keras.utils.np_utils import to_categorical
```

```
In [6]: from google.colab import drive
drive.mount('/content/gdrive')
Mounted at /content/gdrive
```

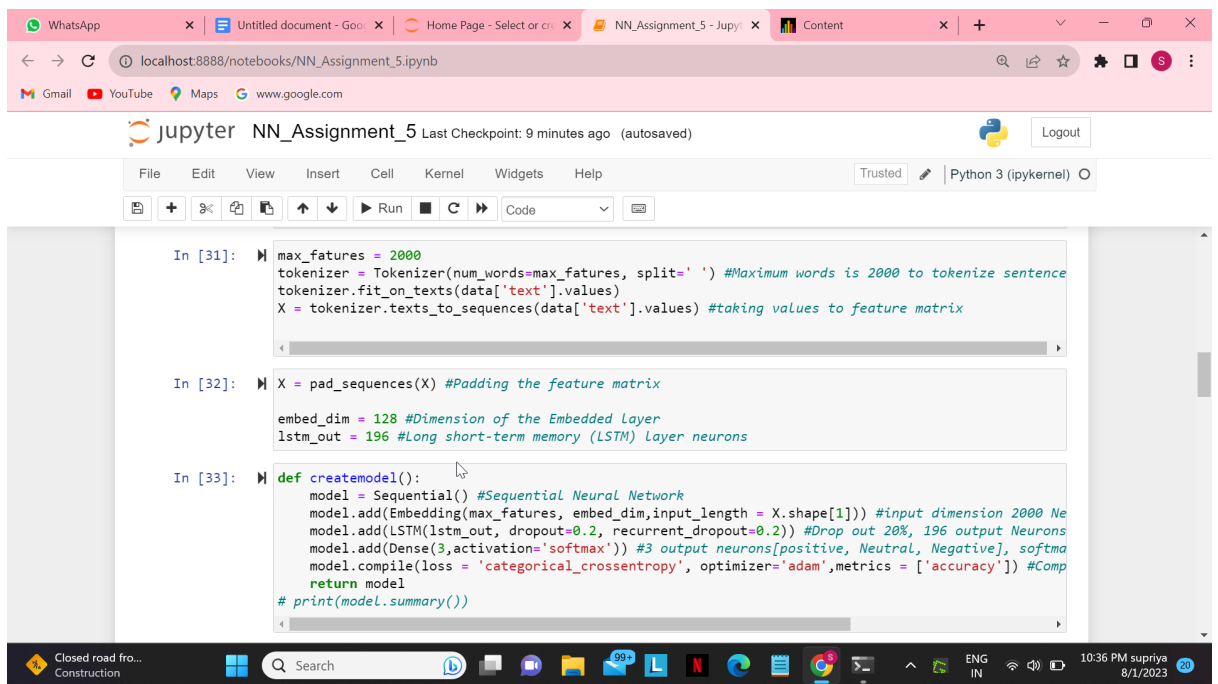
```
In [27]: import pandas as pd

# Load the dataset as a Pandas DataFrame
dataset = pd.read_csv(path_to_csv, header=0)

# Select only the necessary columns 'text' and 'sentiment'
mask = dataset.columns.isin(['text', 'sentiment'])
data = dataset.loc[:, mask]

# Keeping only the necessary columns
```

```
In [29]: data['text'] = data['text'].apply(lambda x: x.lower())
data['text'] = data['text'].apply(lambda x: re.sub('[^a-zA-z0-9\s]', '', x))
```



The screenshot shows a Jupyter Notebook interface with the following code cells:

```
In [31]: max_fatures = 2000
tokenizer = Tokenizer(num_words=max_fatures, split=' ') #Maximum words is 2000 to tokenize sentence
tokenizer.fit_on_texts(data['text'].values)
X = tokenizer.texts_to_sequences(data['text'].values) #taking values to feature matrix
```

```
In [32]: X = pad_sequences(X) #Padding the feature matrix

embed_dim = 128 #Dimension of the Embedded Layer
lstm_out = 196 #Long short-term memory (LSTM) Layer neurons
```

```
In [33]: def createmodel():
    model = Sequential() #Sequential Neural Network
    model.add(Embedding(max_fatures, embed_dim, input_length = X.shape[1])) #input dimension 2000 Ne
    model.add(LSTM(lstm_out, dropout=0.2, recurrent_dropout=0.2)) #Drop out 20%, 196 output Neurons
    model.add(Dense(3, activation='softmax')) #3 output neurons [positive, Neutral, Negative], softma
    model.compile(loss = 'categorical_crossentropy', optimizer='adam', metrics = ['accuracy']) #Comp
    return model
# print(model.summary())
```

Now, let's apply label encoding.

The screenshot shows a Jupyter Notebook titled 'NN_Assignment_5' running on a local server. The interface includes a top bar with navigation icons and a menu bar with options like File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The notebook contains three code cells:

```
In [34]: labelencoder = LabelEncoder() #Applying Label Encoding on the Label matrix
integer_encoded = labelencoder.fit_transform(data['sentiment']) #fitting the model
y = to_categorical(integer_encoded)
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size = 0.33, random_state = 42) #67% training data, 33% test data
```

```
In [35]: batch_size = 32 #Batch size 32
model = createmodel() #Function call to Sequential Neural Network
model.fit(X_train, Y_train, epochs = 1, batch_size=batch_size, verbose = 2) #verbose the higher, the more messages
score, acc = model.evaluate(X_test, Y_test, verbose=2, batch_size=batch_size) #evaluating the model
print(score)
print(acc)
```

The output for cell [35] shows a warning from TensorFlow and training progress:

```
WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
291/291 - 56s - loss: 0.8208 - accuracy: 0.6530 - 56s/epoch - 193ms/step
144/144 - 2s - loss: 0.7517 - accuracy: 0.6796 - 2s/epoch - 11ms/step
0.751739501953125
0.679554402822166
```

```
In [36]: print(model.metrics_names) #metrics of the model
['loss', 'accuracy']
```

The bottom status bar shows the system temperature (29°C), search bar, and system clock (10:37 PM, 8/1/2023).

The screenshot shows the same Jupyter Notebook interface with two additional code cells:

```
In [37]: model.save('sentimentAnalysis.h5') #Saving the model
```

```
In [38]: from keras.models import load_model #Importing the package for importing the saved model
model = load_model('sentimentAnalysis.h5') #Loading the saved model
```

The output for cell [38] shows the same TensorFlow warning as before.

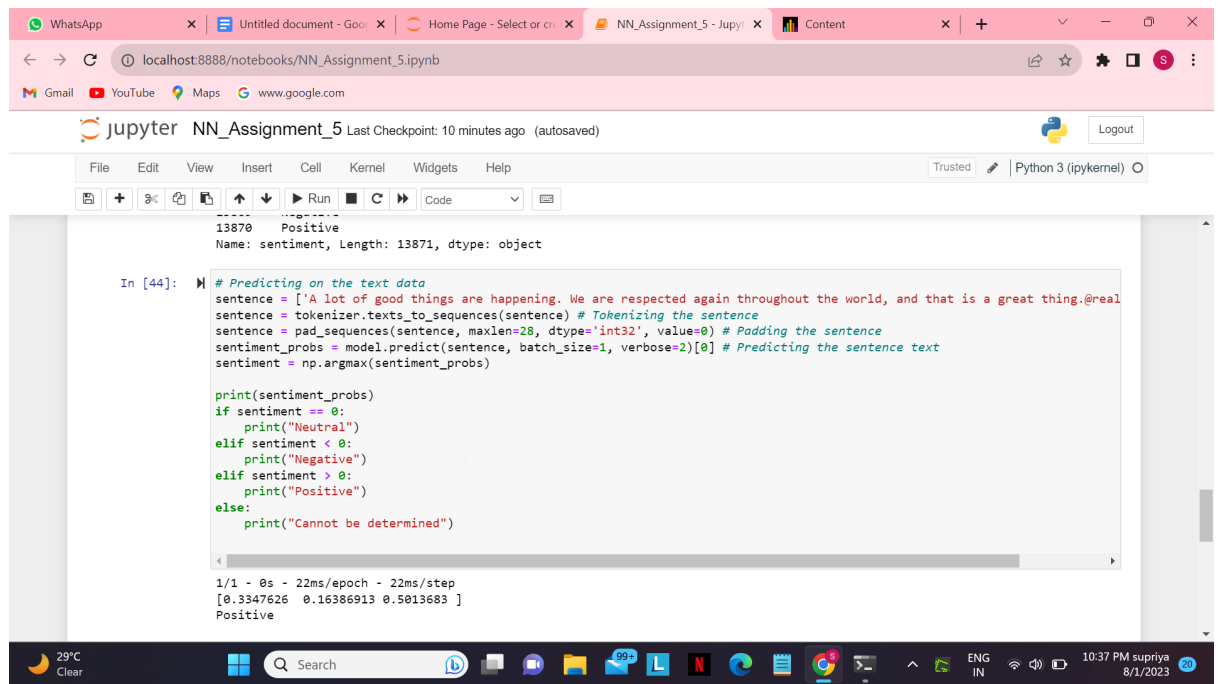
```
In [39]: print(integer_encoded)
print(data['sentiment'])
```

The output for cell [39] shows the integer-encoded sentiment data and the original sentiment labels:

```
[1 2 1 ... 2 0 2]
0      Neutral
1      Positive
2      Neutral
3      Positive
4      Positive
...
13866   Negative
13867   Positive
13868   Positive
13869   Negative
13870   Positive
Name: sentiment, Length: 13871, dtype: object
```

The bottom status bar remains the same, showing the system temperature, search bar, and system clock.

Here we can see the output. Now predict the text data.



```
-----  
13870      Positive  
Name: sentiment, Length: 13871, dtype: object  
  
In [44]: # Predicting on the text data  
sentence = ['A lot of good things are happening. We are respected again throughout the world, and that is a great thing.@realDonaldTrump']  
tokenizer = Tokenizer(maxlen=28) # Tokenizing the sentence  
sentence = pad_sequences(sentence, maxlen=28, dtype='int32', value=0) # Padding the sentence  
sentiment_probs = model.predict(sentence, batch_size=1, verbose=2)[0] # Predicting the sentence text  
sentiment = np.argmax(sentiment_probs)  
  
print(sentiment_probs)  
if sentiment == 0:  
    print("Neutral")  
elif sentiment < 0:  
    print("Negative")  
elif sentiment > 0:  
    print("Positive")  
else:  
    print("Cannot be determined")  
  
1/1 - 0s - 22ms/epoch - 22ms/step  
[0.3347626  0.16386913 0.5013683 ]  
Positive
```

We can see the output is positive.

Now let's apply GridsearchCV on the code, for this let's import KerasClassifier and GridSearchCV.

Next initiate the model to the test performance by applying multiple hyper parameters and summarise the result.

WhatsApp x | Untitled document - Google x | Home Page - Select or create x | NN_Assignment_5 - Jupyter x | Content x | + | - | x

localhost:8888/notebooks/NN_Assignment_5.ipynb

Gmail YouTube Maps www.google.com

jupyter NN_Assignment_5 Last Checkpoint: 11 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

#2. Apply GridSearchCV on the source code provided in the class

```
In [45]: from keras.wrappers.scikit_learn import KerasClassifier #importing Keras classifier
from sklearn.model_selection import GridSearchCV #importing Grid search CV

model = KerasClassifier(build_fn=createmodel,verbose=2) #initiating model to test performance by applying multiple hyper para
batch_size=[10, 20, 40] #hyper parameter batch_size
epochs=[1, 2] #hyper parameter no. of epochs
param_grid={'batch_size':batch_size, 'epochs':epochs} #creating dictionary for batch size, no. of epochs
grid = GridSearchCV(estimator=model, param_grid=param_grid) #Applying dictionary with hyper parameters
grid_result= grid.fit(X_train,Y_train) #Fitting the model
# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_)) #best score, best hyper parameters
```

<ipython-input-45-6c99b49150f4>:4: DeprecationWarning: KerasClassifier is deprecated, use Sci-Keras (https://github.com/adriangb/scikeras) instead. See https://www.adriangb.com/scikeras/stable/migration.html for help migrating.

```
model = KerasClassifier(build_fn=createmodel,verbose=2) #initiating model to test performance by applying multiple hyper
r parameters
WARNING:tensorflow:Layer lstm_1 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU
kernel as fallback when running on GPU.
```

744/744 - 108s - loss: 0.8243 - accuracy: 0.6433 - 108s/epoch - 145ms/step
186/186 - 2s - loss: 0.7794 - accuracy: 0.6681 - 2s/epoch - 12ms/step

29°C Clear Search 99+ ENG IN 10:38 PM supriya 8/1/2023