ICP9

Student Name: **Srinivas Musinuri**
Student id: 700758813

GitHub Link: https://github.com/srinivasmusinuri/700758813_NNDL_ICP9

Video Link:
https://drive.google.com/file/d/1kfUiHDebaPb8omGf-FshK6ijLGYTpzw2/view?usp=sharing

1. Save the model and use the saved model to predict on new text data (ex, “A lot of good things are happening. We are respected again throughout the world, and that's a great thing.@realDonaldTrump”)

Importing the necessary libraries and Sentiment.csv dataset

700758813_NNDL_ICP9.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

```
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 keras.preprocessing.sequence import pad_sequences #Add zeros or crop based on the length
from keras.models import Sequential #Sequential Neural Network
from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D #For layers in Neural Network
from keras.utils import to_categorical
```

```
import pandas as pd

# Load the dataset as a Pandas DataFrame
dataset = pd.read_csv('Sentiment.csv')

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

Iterating through the rows and removing retweets and tokenizing and padding the feature matrix.

700758813_NNDL_ICP9.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

```
[4] for idx, row in data.iterrows():
        row[0] = row[0].replace('rt', ' ') #Removing Retweets
```

```
[5] 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
```

```
[6] 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
```

700758813_NNDL_ICP9.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[7] def createmodel():
        model = Sequential() #Sequential Neural Network
        model.add(Embedding(max_fatures, embed_dim,input_length = X.shape[1])) #input dimension 2000 Neurons, output dimension 128 Neurons
        model.add(LSTM(lstm_out, dropout=0.2, recurrent_dropout=0.2)) #Drop out 20%, 196 output Neurons, recurrent dropout 20%
        model.add(Dense(3,activation='softmax')) #3 output neurons[positive, Neutral, Negative], softmax as activation
        model.compile(loss = 'categorical_crossentropy', optimizer='adam',metrics = ['accuracy']) #Compiling the model
        return model
    # print(model.summary())
```

```
[8] 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 split
```

700758813_NNDL_ICP9.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

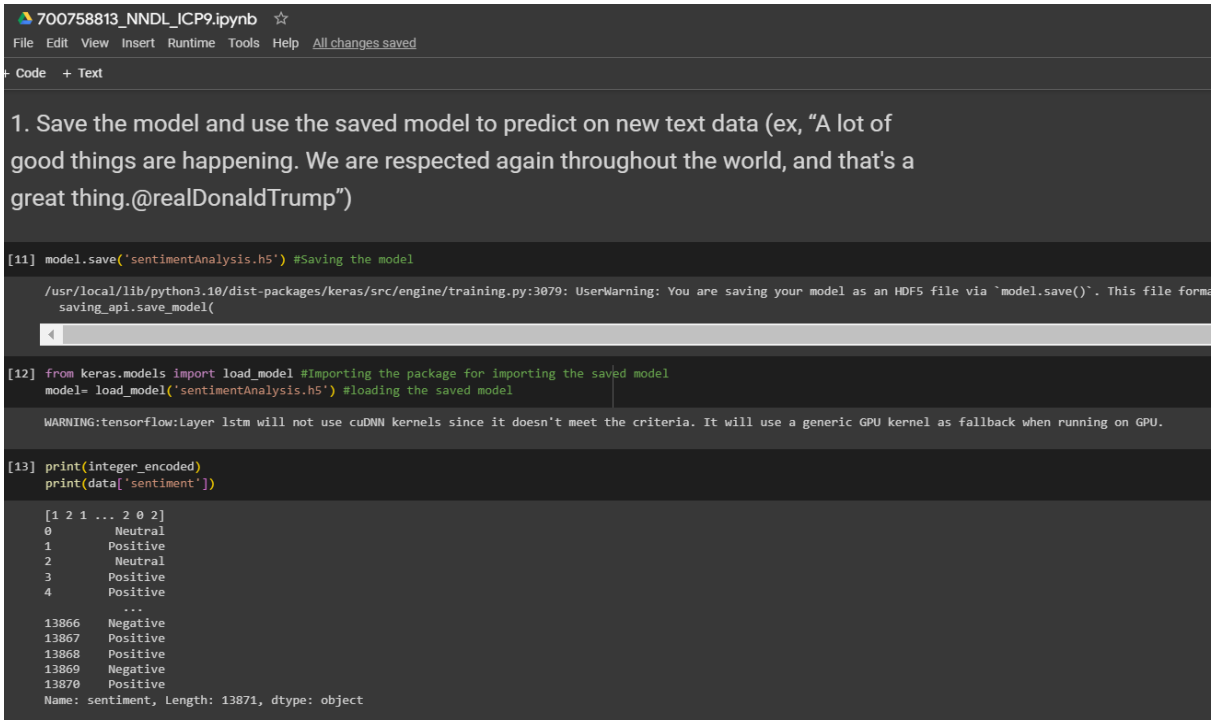
```
[9] batch_size = 32 #Batch size 32
    model = createmodel() #Function call to Sequential Neural Network
    model.fit(X_train, Y_train, epochs = 1, batch_size=batch_size, verbose = 2) #verbose the higher, the more messages
    score,acc = model.evaluate(X_test,Y_test,verbose=2,batch_size=batch_size)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

```
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 - 53s - loss: 0.8235 - accuracy: 0.6435 - 53s/epoch - 183ms/step
144/144 - 2s - loss: 0.7565 - accuracy: 0.6693 - 2s/epoch - 13ms/step
0.7565157413482666
0.669287919998169
```

```
[10] print(model.metrics_names) #metrics of the model
```

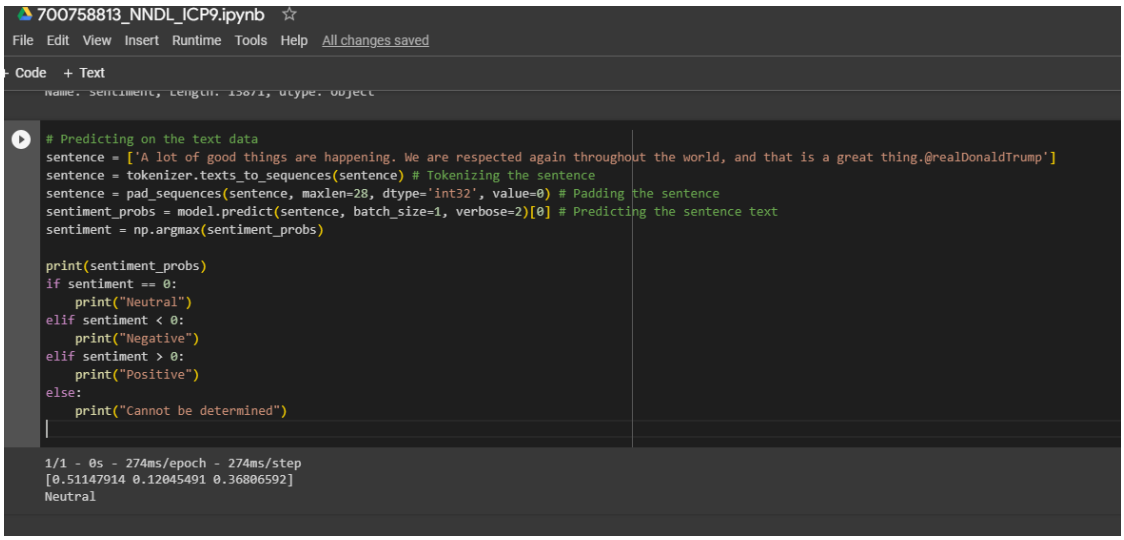
```
['loss', 'accuracy']
```

1. Save the model and use the saved model to predict on new text data (ex, “A lot of good things are happening. We are respected again throughout the world, and that's a great



OUTPUT

Predicting on the text data and got neutral as the output



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

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

```
[ ]  pip install scikeras
```

```
Collecting scikeras
  Downloading scikeras-0.12.0-py3-none-any.whl (27 kB)
Requirement already satisfied: packaging>=0.21 in /usr/local/lib/python3.10/dist-packages (from scikeras) (23.2)
Requirement already satisfied: scikit-learn>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from scikeras) (1.2.2)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0.0->scikeras) (1.23.5)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0.0->scikeras) (1.11.3)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0.0->scikeras) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0.0->scikeras) (3.2.0)
Installing collected packages: scikeras
Successfully installed scikeras-0.12.0
```

700758813_NNDL_ICP9.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
from scikeras.wrappers import KerasClassifier #importing Keras classifier

from sklearn.model_selection import GridSearchCV #importing Grid search CV

model = KerasClassifier(model=createmodel,verbose=2) #initiating model to test performance by applying multiple hyper parameters
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
```

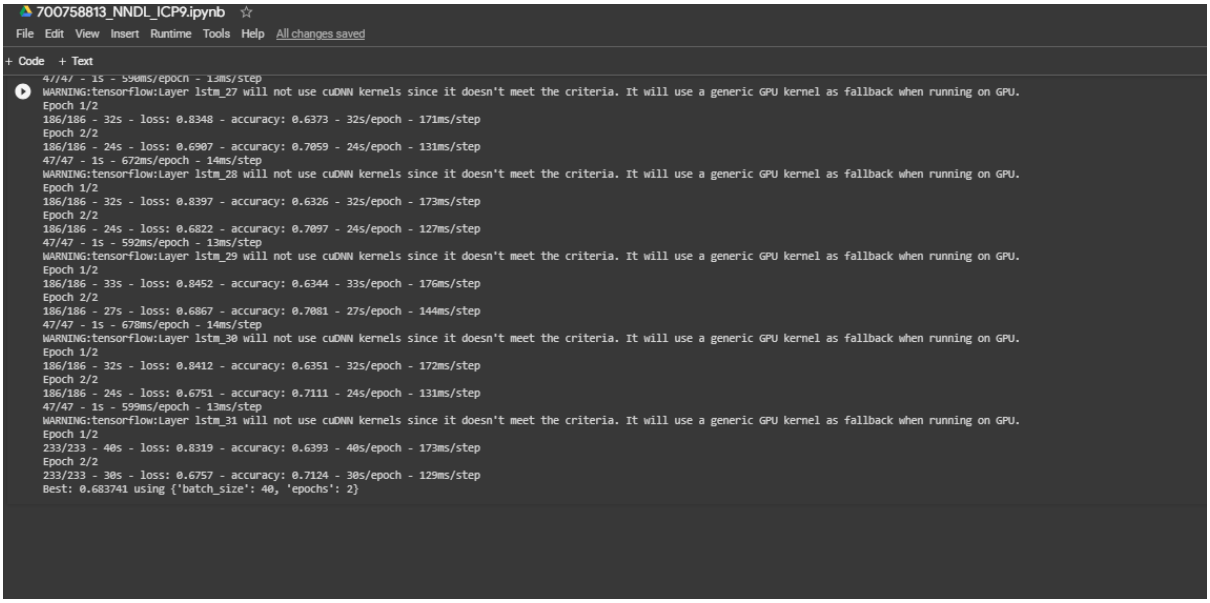
OUTPUT:

700758813_NNDL_ICP9.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Code + Text

```
744/744 - 106s - loss: 0.8321 - accuracy: 0.6416 - 106s/epoch - 142ms/step
186/186 - 2s - 2s/epoch - 11ms/step
WARNING:tensorflow:Layer lstm_4 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: 0.8298 - accuracy: 0.6455 - 108s/epoch - 145ms/step
186/186 - 2s - 2s/epoch - 11ms/step
WARNING:tensorflow:Layer lstm_5 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 - 105s - loss: 0.8220 - accuracy: 0.6491 - 105s/epoch - 141ms/step
186/186 - 2s - 2s/epoch - 12ms/step
WARNING:tensorflow:Layer lstm_6 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.
Epoch 1/2
744/744 - 104s - loss: 0.8282 - accuracy: 0.6470 - 104s/epoch - 140ms/step
Epoch 2/2
744/744 - 95s - loss: 0.6805 - accuracy: 0.7142 - 95s/epoch - 127ms/step
186/186 - 2s - 2s/epoch - 10ms/step
WARNING:tensorflow:Layer lstm_7 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.
Epoch 1/2
744/744 - 106s - loss: 0.8250 - accuracy: 0.6476 - 106s/epoch - 143ms/step
Epoch 2/2
744/744 - 94s - loss: 0.6825 - accuracy: 0.7100 - 94s/epoch - 127ms/step
186/186 - 2s - 2s/epoch - 10ms/step
WARNING:tensorflow:Layer lstm_8 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.
Epoch 1/2
744/744 - 104s - loss: 0.8248 - accuracy: 0.6451 - 104s/epoch - 140ms/step
Epoch 2/2
744/744 - 95s - loss: 0.6751 - accuracy: 0.7152 - 95s/epoch - 127ms/step
186/186 - 2s - 2s/epoch - 10ms/step
WARNING:tensorflow:Layer lstm_9 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.
Epoch 1/2
744/744 - 104s - loss: 0.8245 - accuracy: 0.6452 - 104s/epoch - 140ms/step
Epoch 2/2
744/744 - 93s - loss: 0.6760 - accuracy: 0.7139 - 93s/epoch - 125ms/step
186/186 - 2s - 2s/epoch - 11ms/step
WARNING:tensorflow:Layer lstm_10 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.
Epoch 1/2
744/744 - 106s - loss: 0.8199 - accuracy: 0.6445 - 106s/epoch - 142ms/step
Epoch 2/2
744/744 - 95s - loss: 0.6691 - accuracy: 0.7147 - 95s/epoch - 128ms/step
186/186 - 2s - 2s/epoch - 10ms/step
```


700758813_NNDL_ICP9.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
47/47 - 1s - 596ms/epoch - 13ms/step
WARNING:tensorflow:Layer lstm_27 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.
Epoch 1/2
186/186 - 32s - loss: 0.8348 - accuracy: 0.6373 - 32s/epoch - 171ms/step
Epoch 2/2
186/186 - 24s - loss: 0.6907 - accuracy: 0.7059 - 24s/epoch - 131ms/step
47/47 - 1s - 672ms/epoch - 14ms/step
WARNING:tensorflow:Layer lstm_28 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.
Epoch 1/2
186/186 - 32s - loss: 0.8397 - accuracy: 0.6326 - 32s/epoch - 173ms/step
Epoch 2/2
186/186 - 24s - loss: 0.6822 - accuracy: 0.7097 - 24s/epoch - 127ms/step
47/47 - 1s - 592ms/epoch - 13ms/step
WARNING:tensorflow:Layer lstm_29 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.
Epoch 1/2
186/186 - 33s - loss: 0.8452 - accuracy: 0.6344 - 33s/epoch - 176ms/step
Epoch 2/2
186/186 - 27s - loss: 0.6867 - accuracy: 0.7081 - 27s/epoch - 144ms/step
47/47 - 1s - 678ms/epoch - 14ms/step
WARNING:tensorflow:Layer lstm_30 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.
Epoch 1/2
186/186 - 32s - loss: 0.8412 - accuracy: 0.6351 - 32s/epoch - 172ms/step
Epoch 2/2
186/186 - 24s - loss: 0.6751 - accuracy: 0.7111 - 24s/epoch - 131ms/step
47/47 - 1s - 599ms/epoch - 13ms/step
WARNING:tensorflow:Layer lstm_31 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.
Epoch 1/2
233/233 - 40s - loss: 0.8319 - accuracy: 0.6393 - 40s/epoch - 173ms/step
Epoch 2/2
233/233 - 30s - loss: 0.6757 - accuracy: 0.7124 - 30s/epoch - 129ms/step
Best: 0.683741 using {'batch_size': 40, 'epochs': 2}