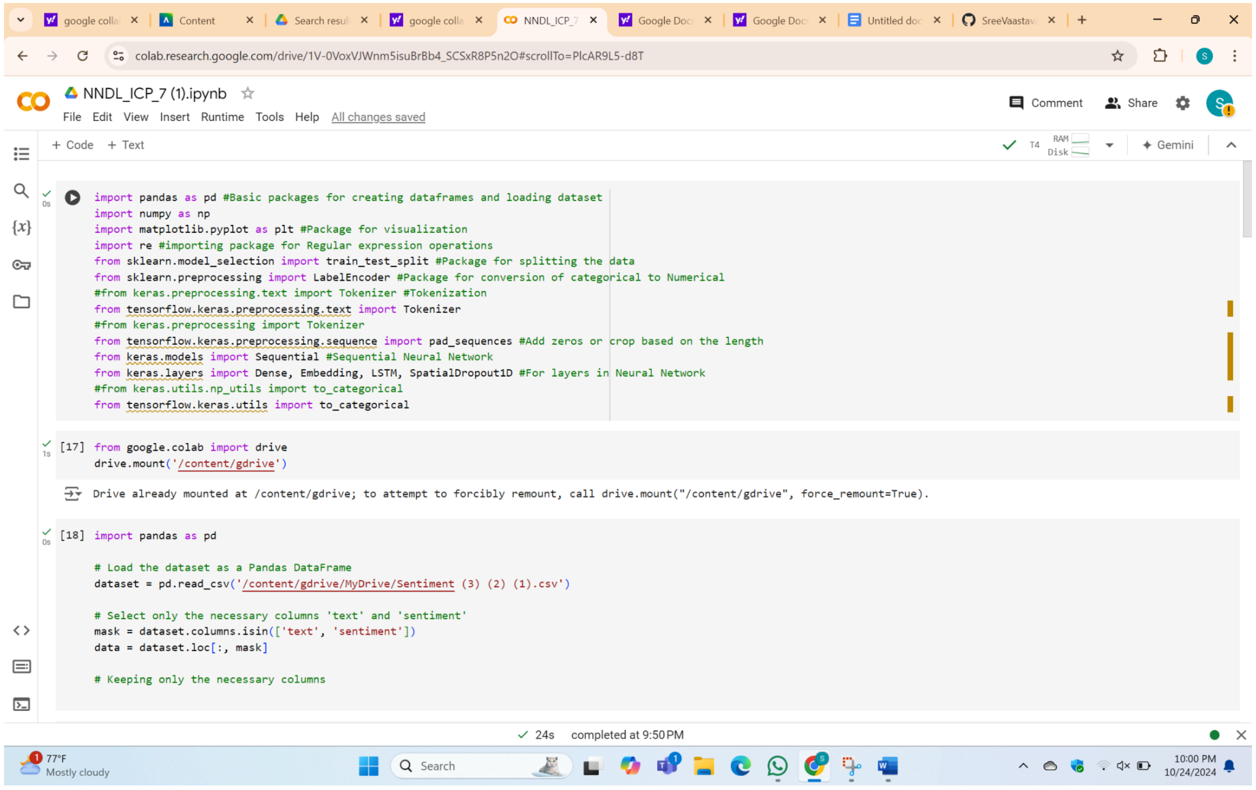
Neural Networks and Deep Learning
ICP - 7

Name : Sree Vaatsava Nalluri
Github Link: https://github.com/SreeVaastava/NNDL_ICP_7
Video Link:
https://drive.google.com/file/d/1mexMGrjNTXQa2by_zrWTsmklv6fIIXjv/view?usp=sharing


colab.research.google.com/drive/1V-0VoxVJWnm5isuBrBb4_SCSxR8P5n2O#scrollTo=PlCAR9L5-d8T
NNDL_ICP_7 (1).ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Comment
Share
Gemini

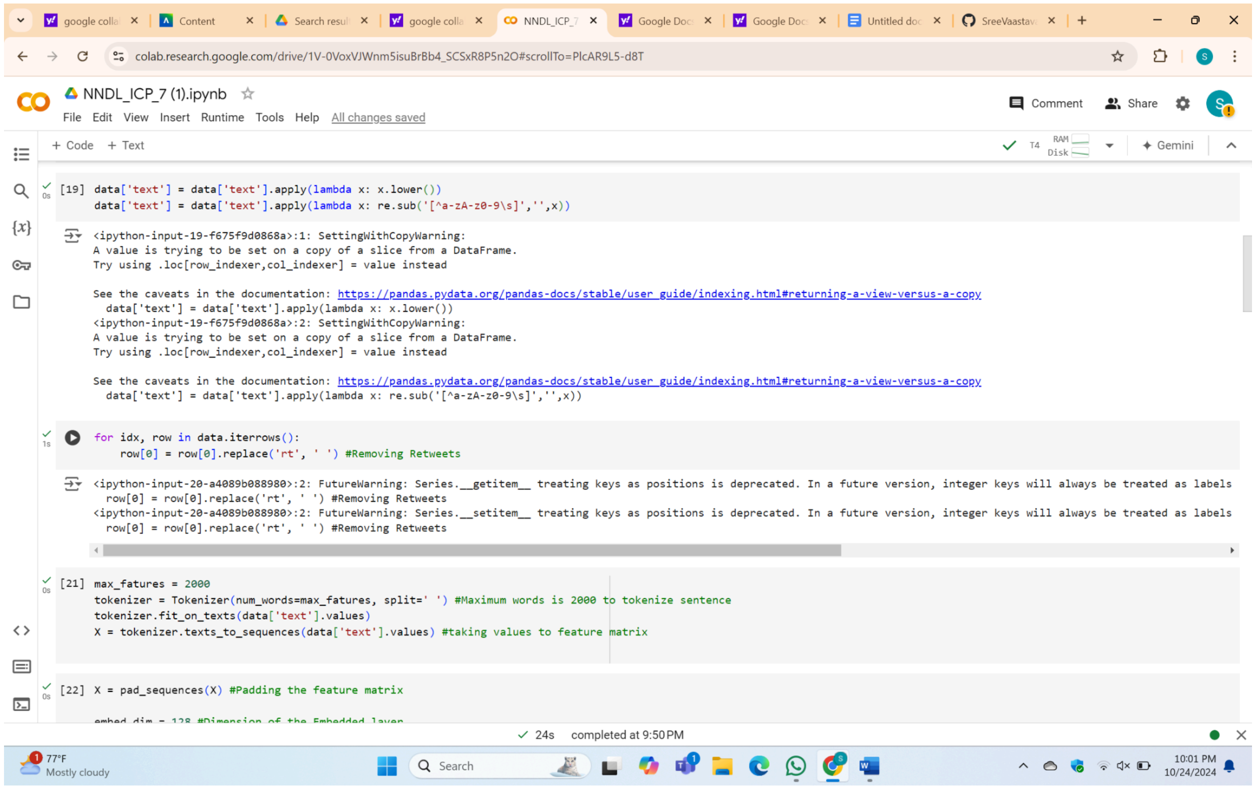
```
import pandas as pd #Basic packages for creating dataframes and loading dataset
import numpy as np
import matplotlib.pyplot as plt #Package for visualization
import re #importing package for Regular expression operations
from sklearn.model_selection import train_test_split #Package for splitting 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.text import Tokenizer
#from keras.preprocessing import Tokenizer
from tensorflow.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.np_utils import to_categorical
from tensorflow.keras.utils import to_categorical
```

```
[17] from google.colab import drive
     drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
```

```
[18] import pandas as pd

     # Load the dataset as a Pandas DataFrame
     dataset = pd.read_csv('/content/gdrive/MyDrive/Sentiment (3) (2) (1).csv')

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

     # Keeping only the necessary columns
```

✓ 24s completed at 9:50 PM


colab.research.google.com/drive/1V-0VoxVJWnm5isuBrBb4_SCSxR8P5n2O#scrollTo=PlcAR9L5-d8T
NNDL_ICP_7 (1).ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Gemini

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

```
<ipython-input-19-f675f9d0868a>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['text'] = data['text'].apply(lambda x: x.lower())
<ipython-input-19-f675f9d0868a>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['text'] = data['text'].apply(lambda x: re.sub('[^a-zA-z0-9\s]','',x))
```

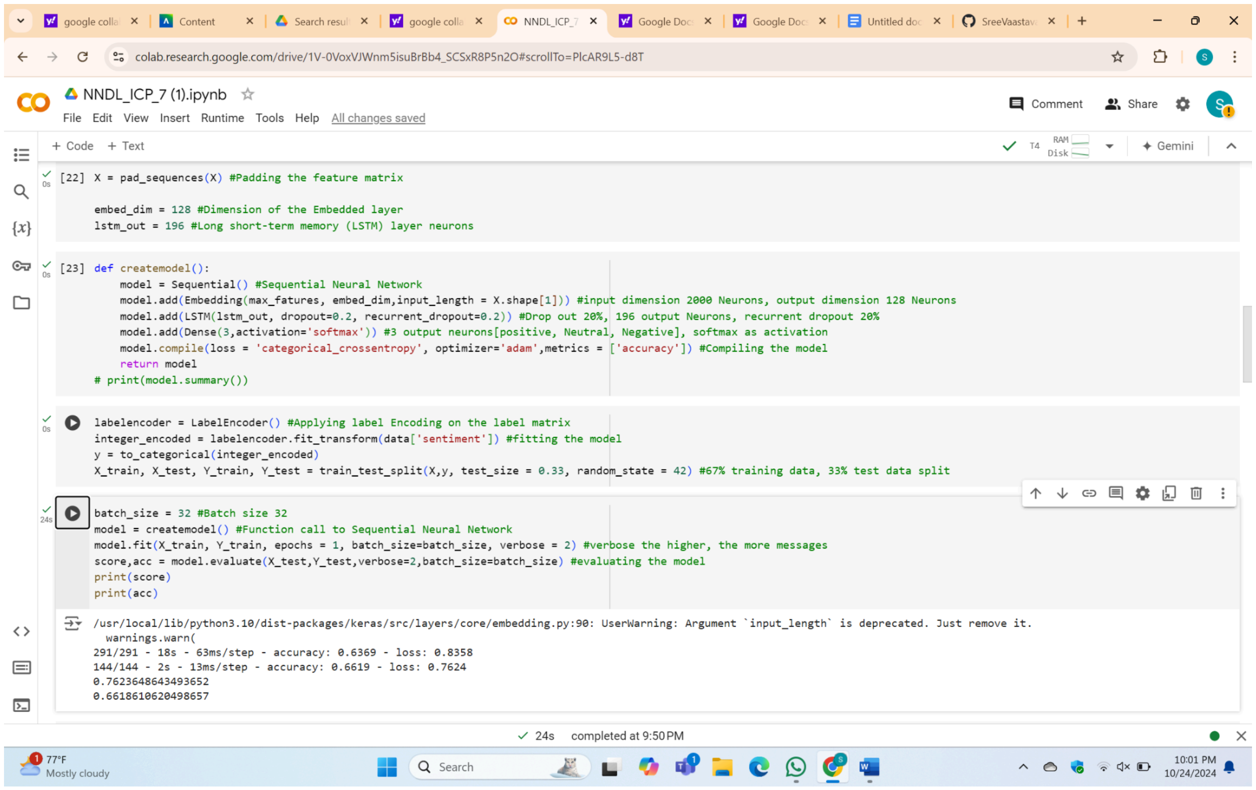
```
for idx, row in data.iterrows():
    row[0] = row[0].replace('rt', ' ') #Removing Retweets
```

```
<ipython-input-20-a4089b088980>:2: FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a future version, integer keys will always be treated as labels
  row[0] = row[0].replace('rt', ' ') #Removing Retweets
<ipython-input-20-a4089b088980>:2: FutureWarning: Series.__setitem__ treating keys as positions is deprecated. In a future version, integer keys will always be treated as labels
  row[0] = row[0].replace('rt', ' ') #Removing Retweets
```

```
[21] 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
```

```
[22] X = pad_sequences(X) #Padding the feature matrix
```

24s completed at 9:50 PM
77°F
Mostly cloudy
Search
10:01 PM
10/24/2024


colab.research.google.com/drive/1V-0VoxVJWnm5isuBrBb4_SCSxR8P5n2O#scrollTo=PlcAR9L5-d8T
NNDL_ICP_7 (1).ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Gemini

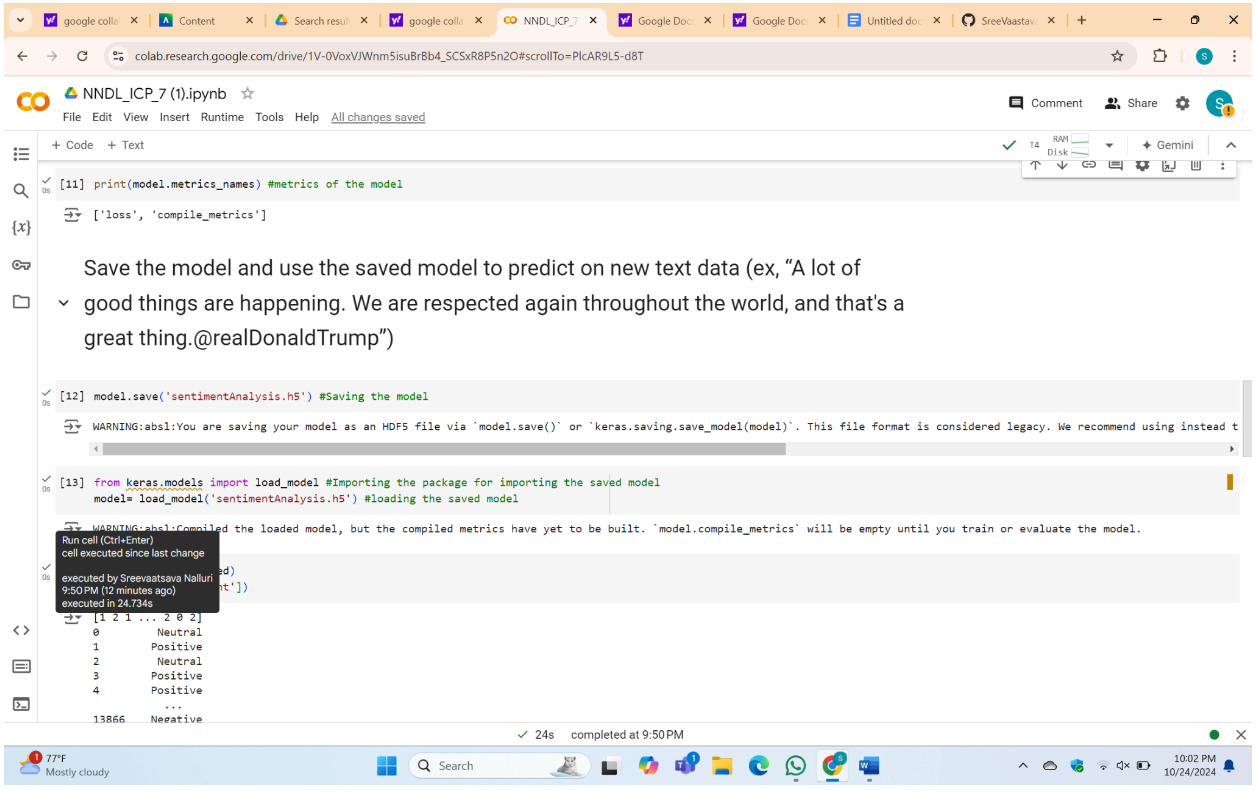
```
[22] X = pad_sequences(X) #Padding the feature matrix

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

```
[23] 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())
```

```
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
```

```
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)
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just remove it.
  warnings.warn(
291/291 - 18s - 63ms/step - accuracy: 0.6369 - loss: 0.8358
144/144 - 2s - 13ms/step - accuracy: 0.6619 - loss: 0.7624
0.7623648643493652
0.6618610620498657
```



NNDL_ICP_7 (1).ipynb

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

```
['loss', 'compile_metrics']
```

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”)

```
[12] model.save('sentimentAnalysis.h5') #Saving the model
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead t
```

```
[13] from keras.models import load_model #Importing the package for importing the saved model
     model= load_model('sentimentAnalysis.h5') #loading the saved model
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
```

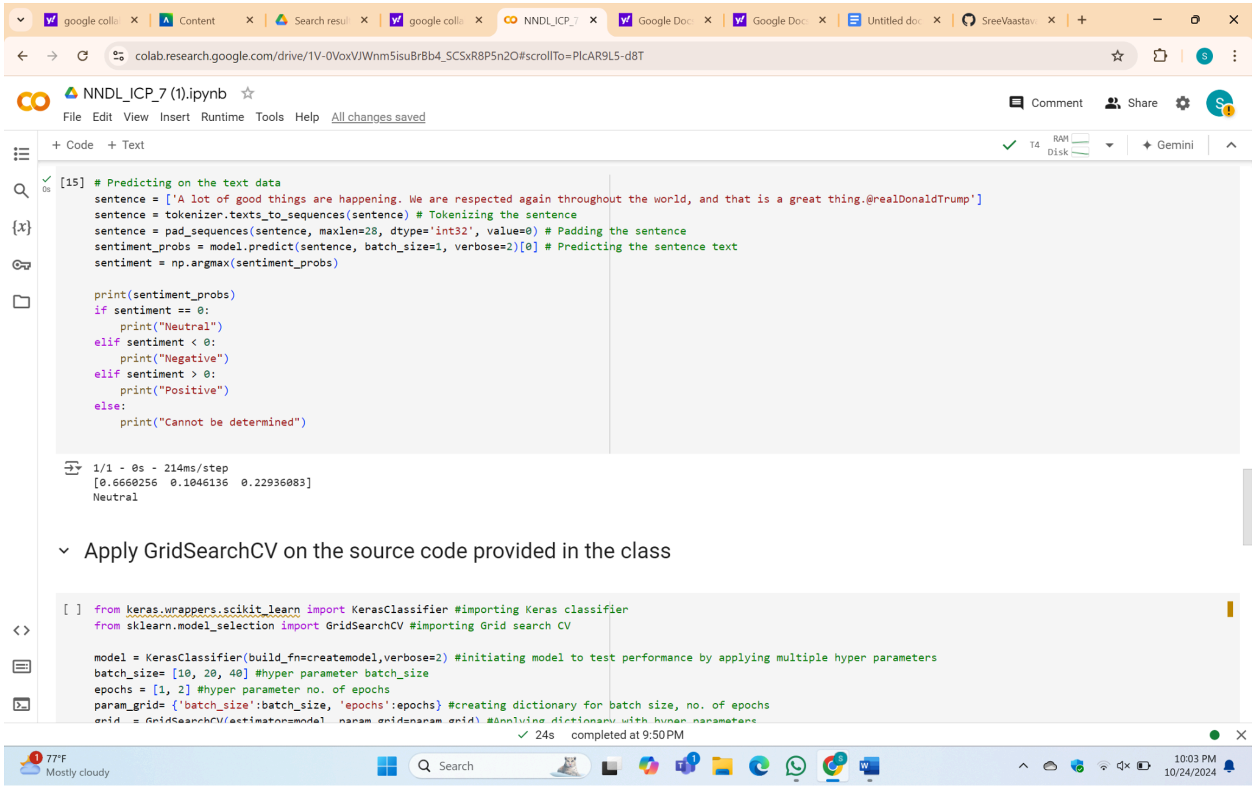
Run cell (Ctrl+Enter)
cell executed since last change

executed by Sreevaatsava Nalluri
9:50 PM (12 minutes ago)
executed in 24.734s

```
ed)
nt'])
```

```
[1 2 1 ... 2 0 2]
0          Neutral
1         Positive
2          Neutral
3         Positive
4         Positive
            ...
13866     Negative
```

✓ 24s completed at 9:50 PM



NNDL_ICP_7 (1).ipynb

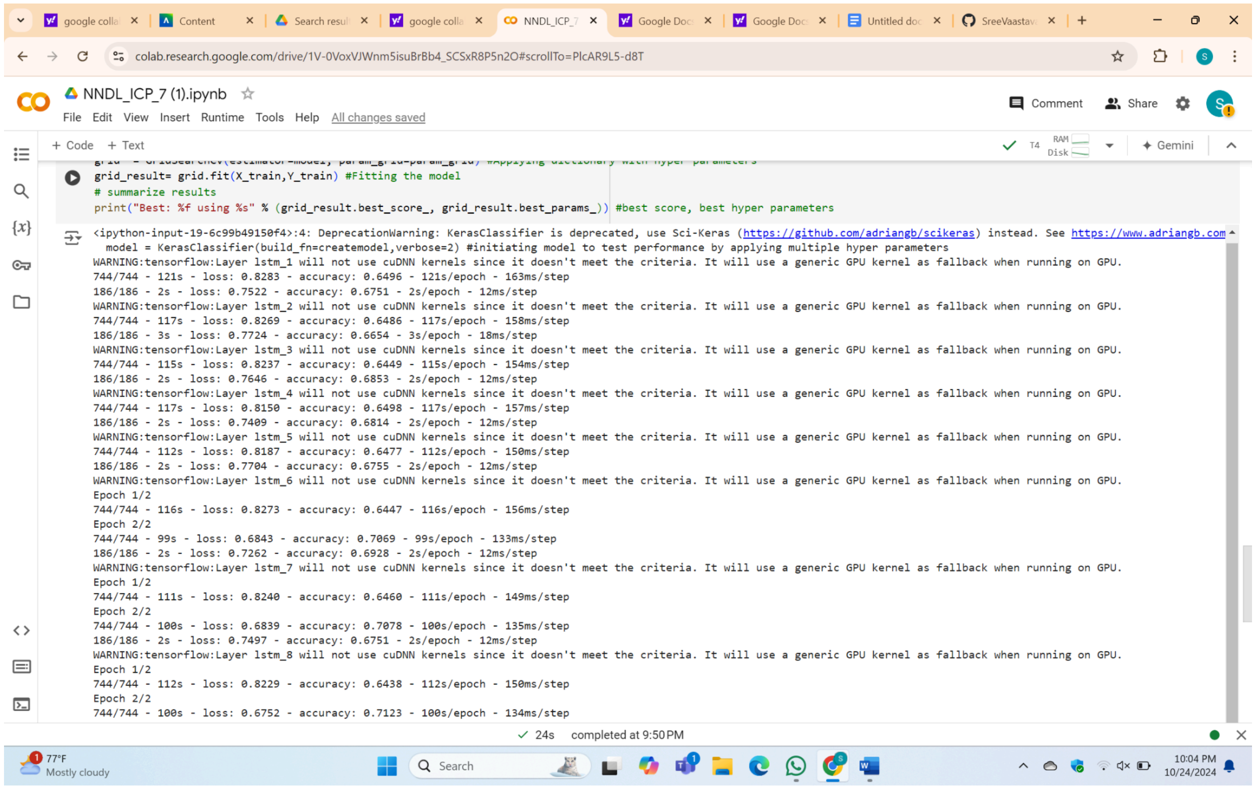
```
[15] # Predicting on the text data
     sentence = ['A lot of good things are happening. We are respected again throughout the world, and that is a great thing.@realDonaldTrump']
     sentence = tokenizer.texts_to_sequences(sentence) # Tokenizing the sentence
     sentence = pad_sequences(sentence, 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 - 214ms/step
[0.6660256  0.1046136  0.22936083]
Neutral
```

Apply GridSearchCV on the source code provided in the class

```
[ ] 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 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
```

✓ 24s completed at 9:50 PM


colab.research.google.com/drive/1V-0VoxVJWnm5isuBrBb4_SCSxR8P5n2O#scrollTo=PlcAR9L5-d8T
NNDL_ICP_7 (1).ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Gemini

```
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-19-6c99b49150f4>:4: DeprecationWarning: KerasClassifier is deprecated, use Sci-Keras (https://github.com/adriangb/scikeras) instead. See https://www.adriangb.com
  model = KerasClassifier(build_fn=createmodel,verbose=2) #initiating model to test performance by applying multiple hyper 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 - 121s - loss: 0.8283 - accuracy: 0.6496 - 121s/epoch - 163ms/step
186/186 - 2s - loss: 0.7522 - accuracy: 0.6751 - 2s/epoch - 12ms/step
WARNING:tensorflow:Layer lstm_2 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 - 117s - loss: 0.8269 - accuracy: 0.6486 - 117s/epoch - 158ms/step
186/186 - 3s - loss: 0.7724 - accuracy: 0.6654 - 3s/epoch - 18ms/step
WARNING:tensorflow:Layer lstm_3 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 - 115s - loss: 0.8237 - accuracy: 0.6449 - 115s/epoch - 154ms/step
186/186 - 2s - loss: 0.7646 - accuracy: 0.6853 - 2s/epoch - 12ms/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 - 117s - loss: 0.8150 - accuracy: 0.6498 - 117s/epoch - 157ms/step
186/186 - 2s - loss: 0.7409 - accuracy: 0.6814 - 2s/epoch - 12ms/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 - 112s - loss: 0.8187 - accuracy: 0.6477 - 112s/epoch - 150ms/step
186/186 - 2s - loss: 0.7704 - accuracy: 0.6755 - 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 - 116s - loss: 0.8273 - accuracy: 0.6447 - 116s/epoch - 156ms/step
Epoch 2/2
744/744 - 99s - loss: 0.6843 - accuracy: 0.7069 - 99s/epoch - 133ms/step
186/186 - 2s - loss: 0.7262 - accuracy: 0.6928 - 2s/epoch - 12ms/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 - 111s - loss: 0.8240 - accuracy: 0.6460 - 111s/epoch - 149ms/step
Epoch 2/2
744/744 - 100s - loss: 0.6839 - accuracy: 0.7078 - 100s/epoch - 135ms/step
186/186 - 2s - loss: 0.7497 - accuracy: 0.6751 - 2s/epoch - 12ms/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 - 112s - loss: 0.8229 - accuracy: 0.6438 - 112s/epoch - 150ms/step
Epoch 2/2
744/744 - 100s - loss: 0.6752 - accuracy: 0.7123 - 100s/epoch - 134ms/step
```