In class Programming Assignment - 7

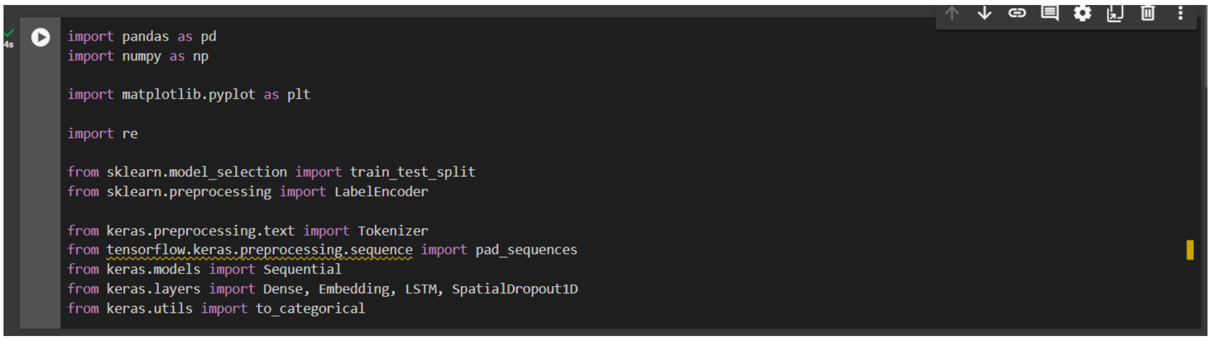
GitHub Link: https://github.com/HarshithaBoyapati2002/nn_icp7

Video:https://drive.google.com/file/d/1eYoYEjKHoG_Rk-adsMh5Qkv3OPVYoEpd/view?usp=sharing

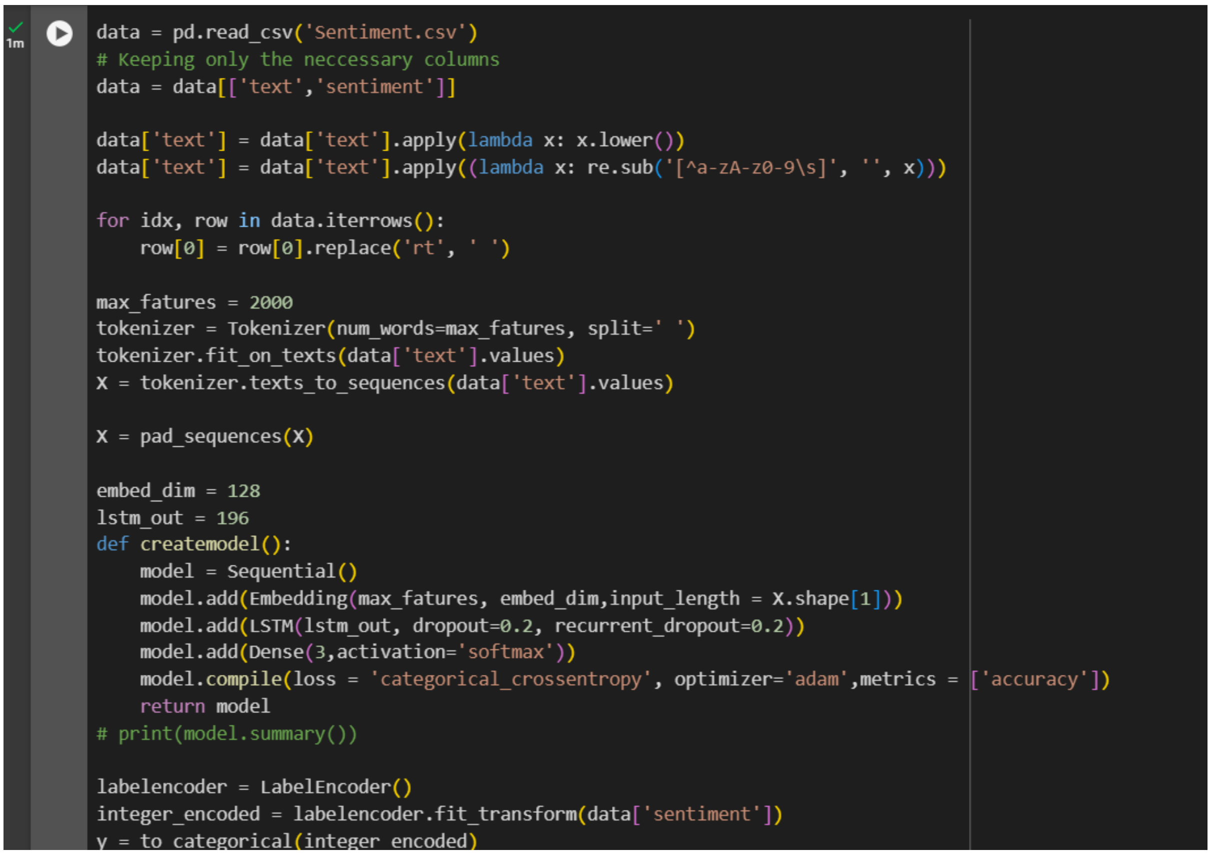
Problem 1: Basics of LSTM, Types of RNN, Sentiment Analysis on the Twitter data set

Solution:

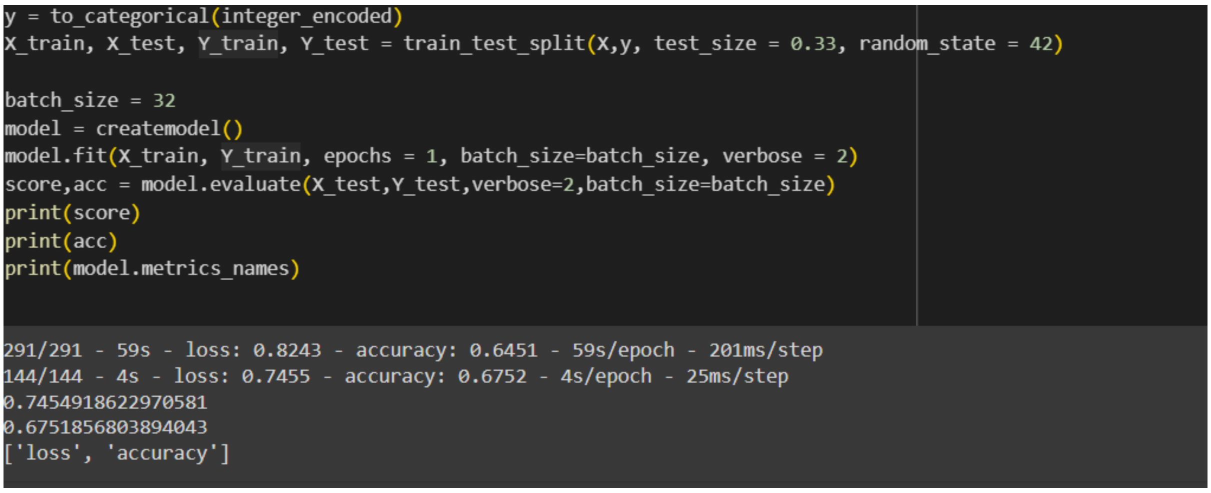
1. Saving the model and using the saved model to predict on new text data.



```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt

import re

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

from keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D
from keras.utils import to_categorical
```

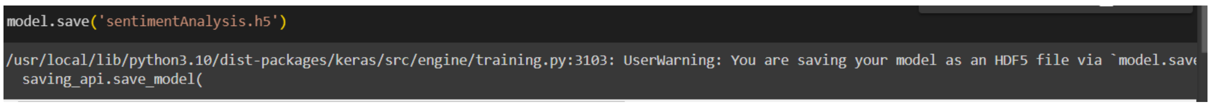


```
data = pd.read_csv('Sentiment.csv')
# Keeping only the neccessary columns
data = data[['text','sentiment']]

data['text'] = data['text'].apply(lambda x: x.lower())
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', ' ')

max_fatures = 2000
tokenizer = Tokenizer(num_words=max_fatures, split=' ')
tokenizer.fit_on_texts(data['text'].values)
X = tokenizer.texts_to_sequences(data['text'].values)

X = pad_sequences(X)

embed_dim = 128
lstm_out = 196
def createmodel():
    model = Sequential()
    model.add(Embedding(max_fatures, embed_dim,input_length = X.shape[1]))
    model.add(LSTM(lstm_out, dropout=0.2, recurrent_dropout=0.2))
    model.add(Dense(3,activation='softmax'))
    model.compile(loss = 'categorical_crossentropy', optimizer='adam',metrics = ['accuracy'])
    return model
# print(model.summary())

labelencoder = LabelEncoder()
integer_encoded = labelencoder.fit_transform(data['sentiment'])
y = to_categorical(integer_encoded)
```



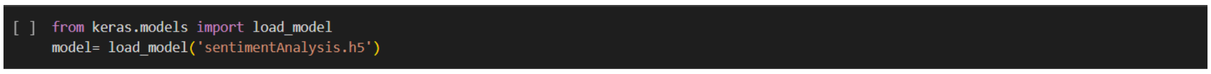
```
y = to_categorical(integer_encoded)
X_train, X_test, Y_train, Y_test = train_test_split(X,y, test_size = 0.33, random_state = 42)

batch_size = 32
model = createmodel()
model.fit(X_train, Y_train, epochs = 1, batch_size=batch_size, verbose = 2)
score,acc = model.evaluate(X_test,Y_test,verbose=2,batch_size=batch_size)
print(score)
print(acc)
print(model.metrics_names)
```

```
291/291 - 59s - loss: 0.8243 - accuracy: 0.6451 - 59s/epoch - 201ms/step
144/144 - 4s - loss: 0.7455 - accuracy: 0.6752 - 4s/epoch - 25ms/step
0.7454918622970581
0.6751856803894043
['loss', 'accuracy']
```

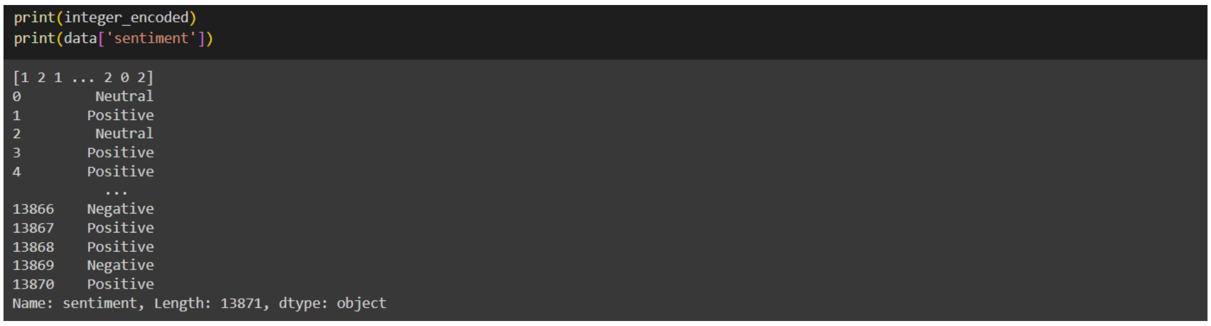


```
model.save('sentimentAnalysis.h5')
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `model.save
  saving_api.save_model(
```

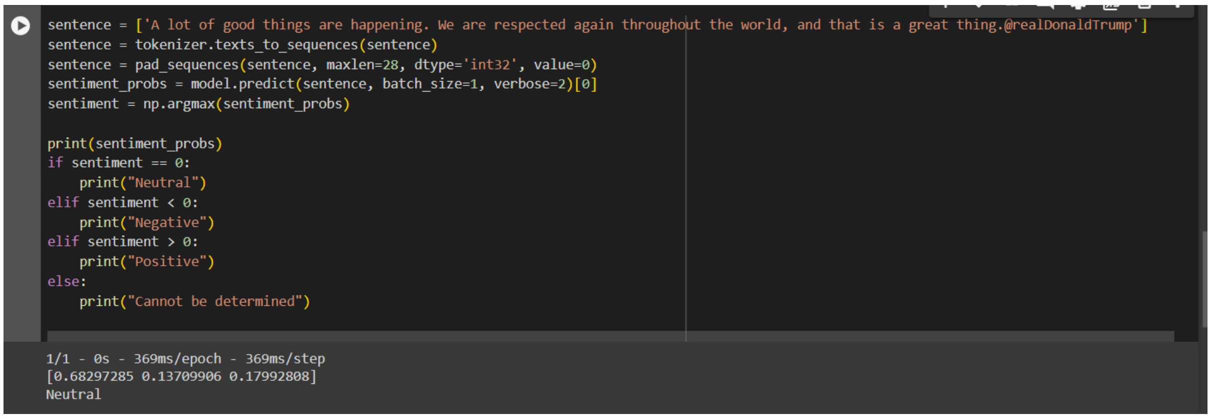


```
[ ]  from keras.models import load_model
     model= load_model('sentimentAnalysis.h5')
```



```
print(integer_encoded)
print(data['sentiment'])
```

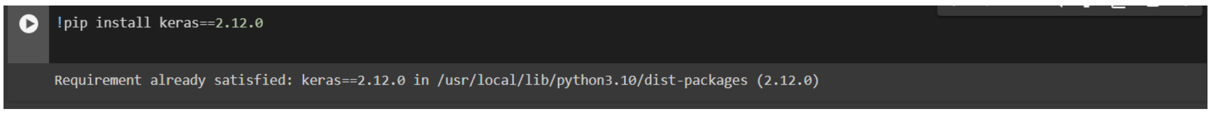
```
[1 2 1 ... 2 0 2]
0         Neutral
1        Positive
2         Neutral
3        Positive
4        Positive
           ...   
13866    Negative
13867    Positive
13868    Positive
13869    Negative
13870    Positive
Name: sentiment, Length: 13871, dtype: object
```



```
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)
sentence = pad_sequences(sentence, maxlen=28, dtype='int32', value=0)
sentiment_probs = model.predict(sentence, batch_size=1, verbose=2)[0]
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")
```

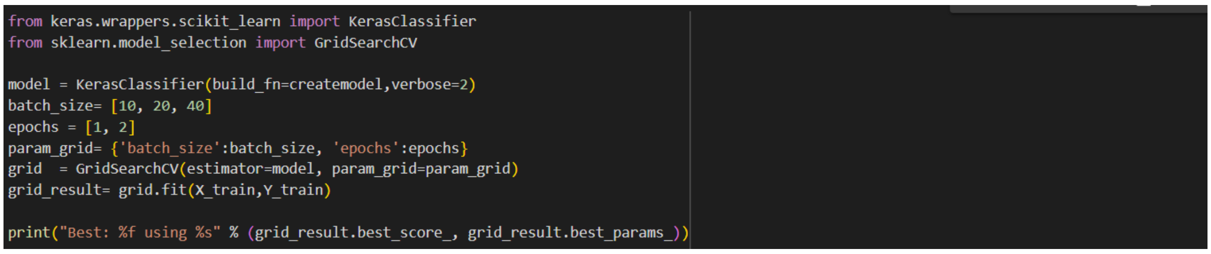
```
1/1 - 0s - 369ms/epoch - 369ms/step
[0.68297285 0.13709906 0.17992808]
Neutral
```

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

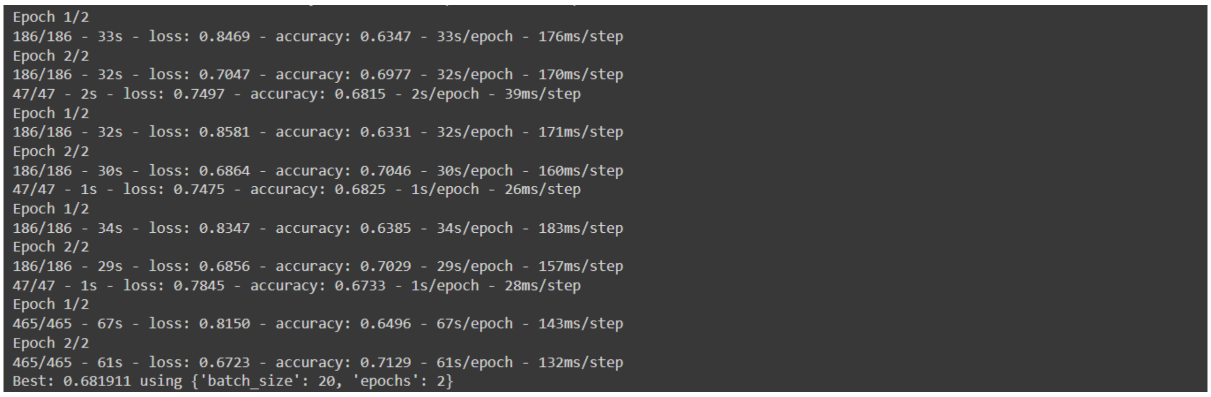


```
!pip install keras==2.12.0
```

```
Requirement already satisfied: keras==2.12.0 in /usr/local/lib/python3.10/dist-packages (2.12.0)
```



```python
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import GridSearchCV

model = KerasClassifier(build_fn=createmodel,verbose=2)
batch_size= [10, 20, 40]
epochs = [1, 2]
param_grid= {'batch_size':batch_size, 'epochs':epochs}
grid  = GridSearchCV(estimator=model, param_grid=param_grid)
grid_result= grid.fit(X_train,Y_train)

print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
```



```
Epoch 1/2
186/186 - 33s - loss: 0.8469 - accuracy: 0.6347 - 33s/epoch - 176ms/step
Epoch 2/2
186/186 - 32s - loss: 0.7047 - accuracy: 0.6977 - 32s/epoch - 170ms/step
47/47 - 2s - loss: 0.7497 - accuracy: 0.6815 - 2s/epoch - 39ms/step
Epoch 1/2
186/186 - 32s - loss: 0.8581 - accuracy: 0.6331 - 32s/epoch - 171ms/step
Epoch 2/2
186/186 - 30s - loss: 0.6864 - accuracy: 0.7046 - 30s/epoch - 160ms/step
47/47 - 1s - loss: 0.7475 - accuracy: 0.6825 - 1s/epoch - 26ms/step
Epoch 1/2
186/186 - 34s - loss: 0.8347 - accuracy: 0.6385 - 34s/epoch - 183ms/step
Epoch 2/2
186/186 - 29s - loss: 0.6856 - accuracy: 0.7029 - 29s/epoch - 157ms/step
47/47 - 1s - loss: 0.7845 - accuracy: 0.6733 - 1s/epoch - 28ms/step
Epoch 1/2
465/465 - 67s - loss: 0.8150 - accuracy: 0.6496 - 67s/epoch - 143ms/step
Epoch 2/2
465/465 - 61s - loss: 0.6723 - accuracy: 0.7129 - 61s/epoch - 132ms/step
Best: 0.681911 using {'batch_size': 20, 'epochs': 2}
```