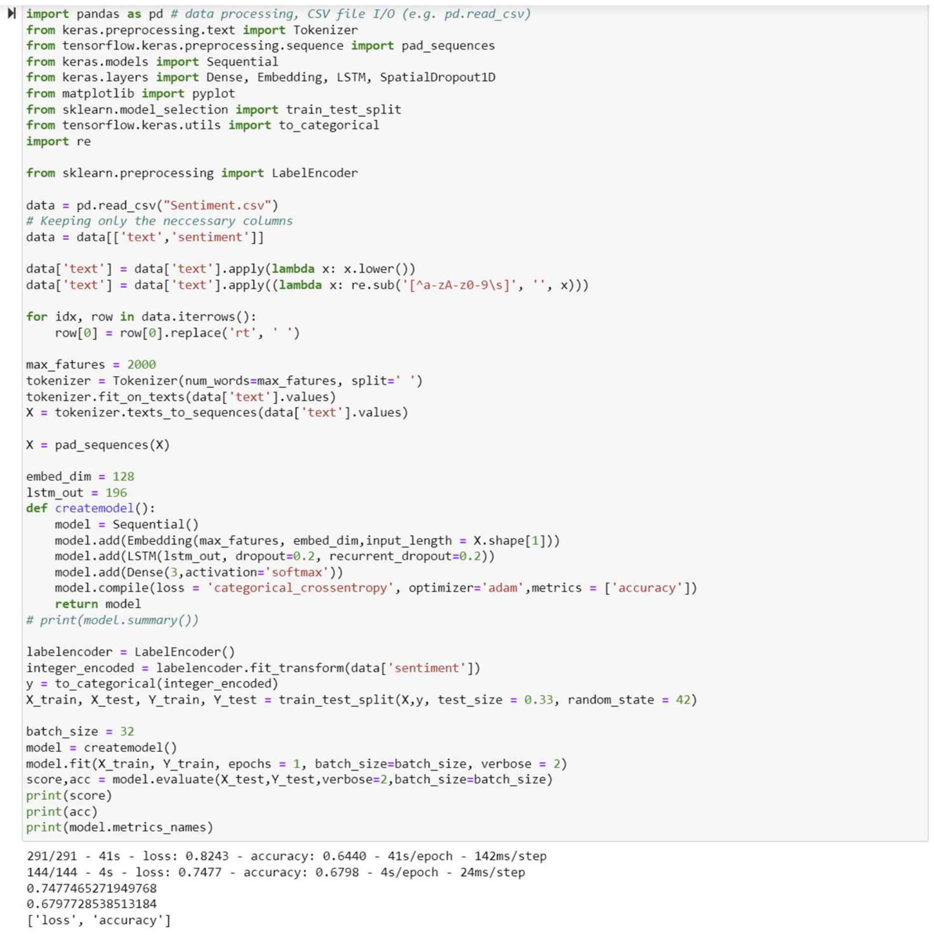
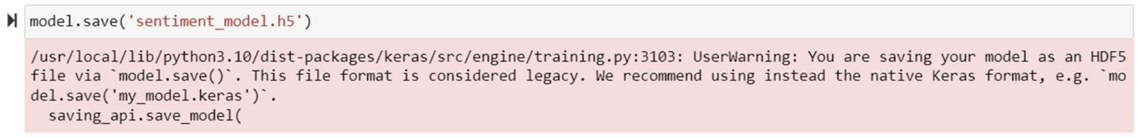
GitHub Link: https://github.com/AnishKoppula1/NeuralAssignment10

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”)



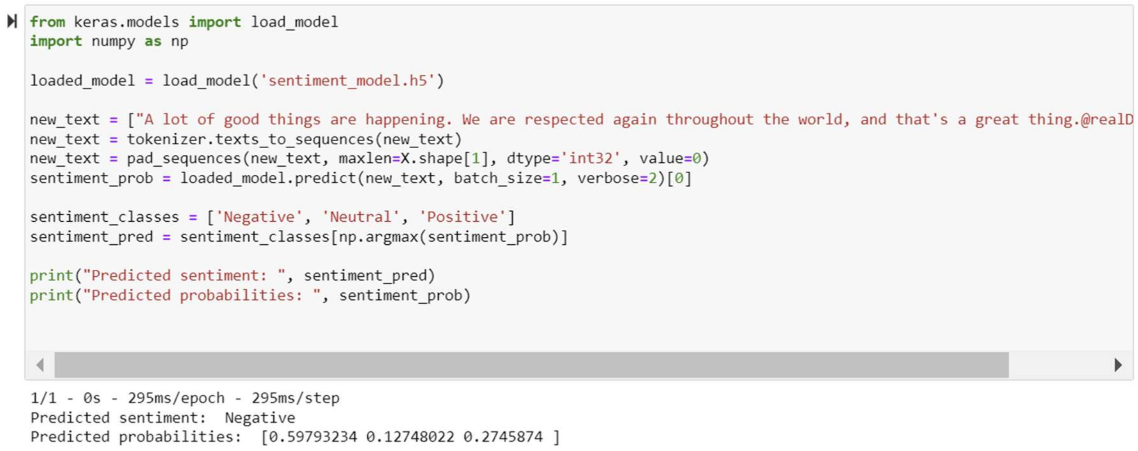
```
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
import re

from sklearn.preprocessing import LabelEncoder

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)
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 - 41s - loss: 0.8243 - accuracy: 0.6440 - 41s/epoch - 142ms/step
144/144 - 4s - loss: 0.7477 - accuracy: 0.6798 - 4s/epoch - 24ms/step
0.7477465271949768
0.6797728538513184
['loss', 'accuracy']
```



```
model.save('sentiment_model.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()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `mo
del.save('my_model.keras')`.
  saving_api.save_model(
```



```
from keras.models import load_model
import numpy as np

loaded_model = load_model('sentiment_model.h5')

new_text = ["A lot of good things are happening. We are respected again throughout the world, and that's a great thing.@realD
new_text = tokenizer.texts_to_sequences(new_text)
new_text = pad_sequences(new_text, maxlen=X.shape[1], dtype='int32', value=0)
sentiment_prob = loaded_model.predict(new_text, batch_size=1, verbose=2)[0]

sentiment_classes = ['Negative', 'Neutral', 'Positive']
sentiment_pred = sentiment_classes[np.argmax(sentiment_prob)]

print("Predicted sentiment: ", sentiment_pred)
print("Predicted probabilities: ", sentiment_prob)
```

```
1/1 - 0s - 295ms/epoch - 295ms/step
Predicted sentiment:  Negative
Predicted probabilities:  [0.59793234 0.12748022 0.2745874 ]
```

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

```python
from scikeras.wrappers import KerasClassifier
#from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import GridSearchCV
from keras.optimizers import Adam

def create_model(units=196, dropout=0.2, learning_rate=0.001):
    model = Sequential()
    model.add(Embedding(max_fatures, embed_dim,input_length = X.shape[1]))
    model.add(LSTM(units, dropout=dropout, recurrent_dropout=dropout))
    model.add(Dense(3, activation='softmax'))
    optimizer = Adam(lr=learning_rate)
    model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
    return model

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
```

```
/usr/local/lib/python3.10/dist-packages/scikeras/wrappers.py:915: UserWarning: ``build_fn`` will be renamed to ``model`` in
a future release, at which point use of ``build_fn`` will raise an Error instead.
  X, y = self._initialize(X, y)

744/744 - 101s - loss: 0.8249 - accuracy: 0.6439 - 101s/epoch - 136ms/step
186/186 - 3s - 3s/epoch - 18ms/step

/usr/local/lib/python3.10/dist-packages/scikeras/wrappers.py:915: UserWarning: ``build_fn`` will be renamed to ``model`` in
a future release, at which point use of ``build_fn`` will raise an Error instead.
  X, y = self._initialize(X, y)

744/744 - 87s - loss: 0.8256 - accuracy: 0.6497 - 87s/epoch - 117ms/step
186/186 - 3s - 3s/epoch - 14ms/step

/usr/local/lib/python3.10/dist-packages/scikeras/wrappers.py:915: UserWarning: ``build_fn`` will be renamed to ``model`` in
a future release, at which point use of ``build_fn`` will raise an Error instead.
  X, y = self._initialize(X, y)

744/744 - 86s - loss: 0.8283 - accuracy: 0.6461 - 86s/epoch - 116ms/step
186/186 - 2s - 2s/epoch - 13ms/step

/usr/local/lib/python3.10/dist-packages/scikeras/wrappers.py:915: UserWarning: ``build_fn`` will be renamed to ``model`` in
a future release, at which point use of ``build_fn`` will raise an Error instead.
  X, y = self._initialize(X, y)

744/744 - 84s - loss: 0.8289 - accuracy: 0.6457 - 84s/epoch - 114ms/step
186/186 - 2s - 2s/epoch - 12ms/step

/usr/local/lib/python3.10/dist-packages/scikeras/wrappers.py:915: UserWarning: ``build_fn`` will be renamed to ``model`` in
a future release, at which point use of ``build_fn`` will raise an Error instead.
  X, y = self._initialize(X, y)

744/744 - 93s - loss: 0.8215 - accuracy: 0.6459 - 93s/epoch - 125ms/step
186/186 - 3s - 3s/epoch - 13ms/step
```