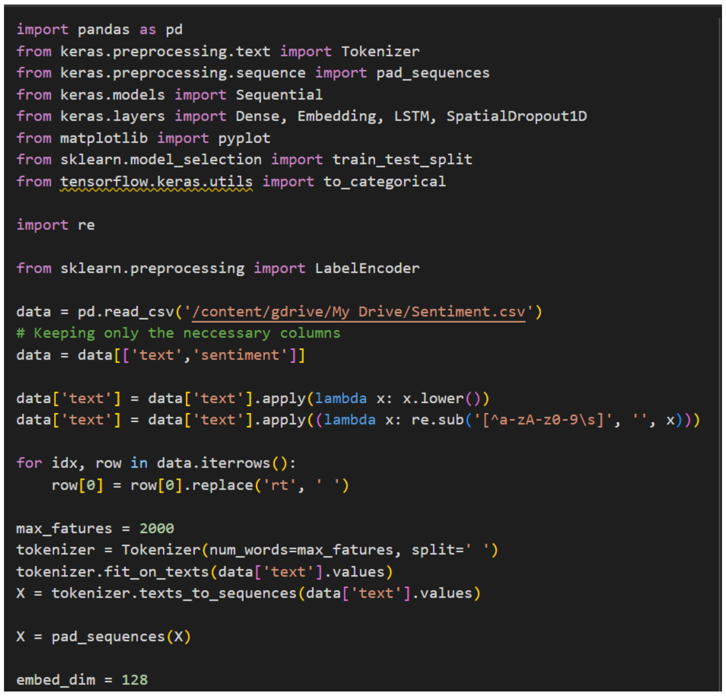
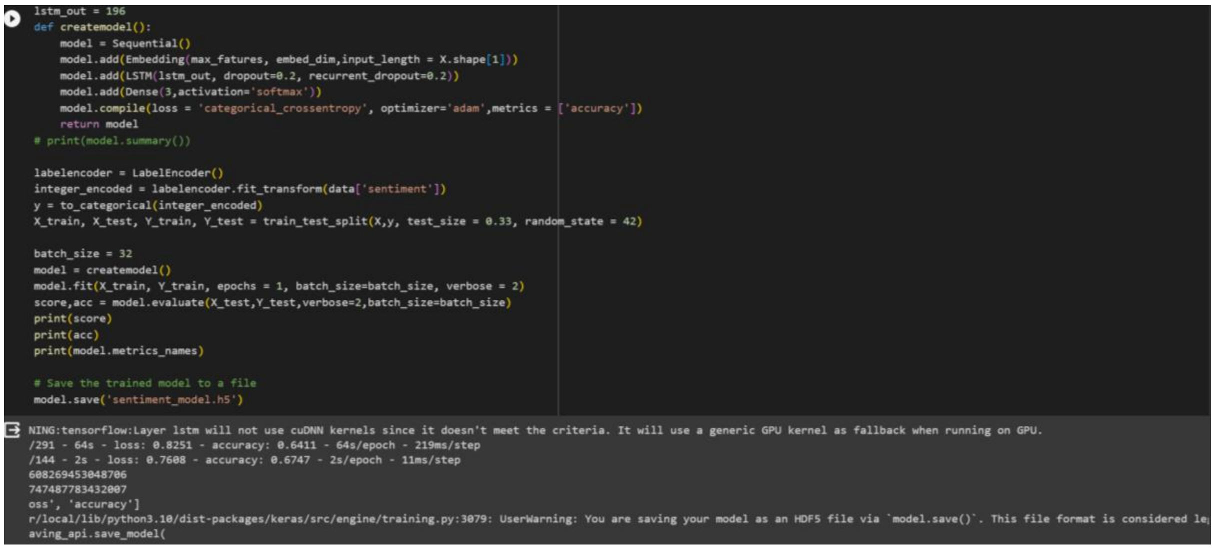
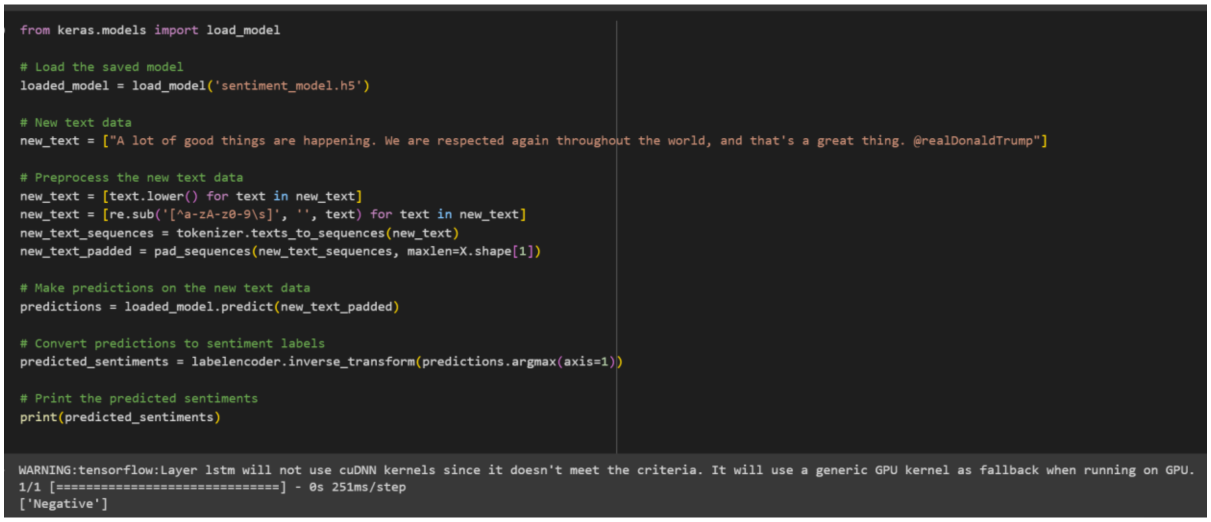


```python
import pandas as pd
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical

import re

from sklearn.preprocessing import LabelEncoder

data = pd.read_csv('/content/gdrive/My Drive/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
```



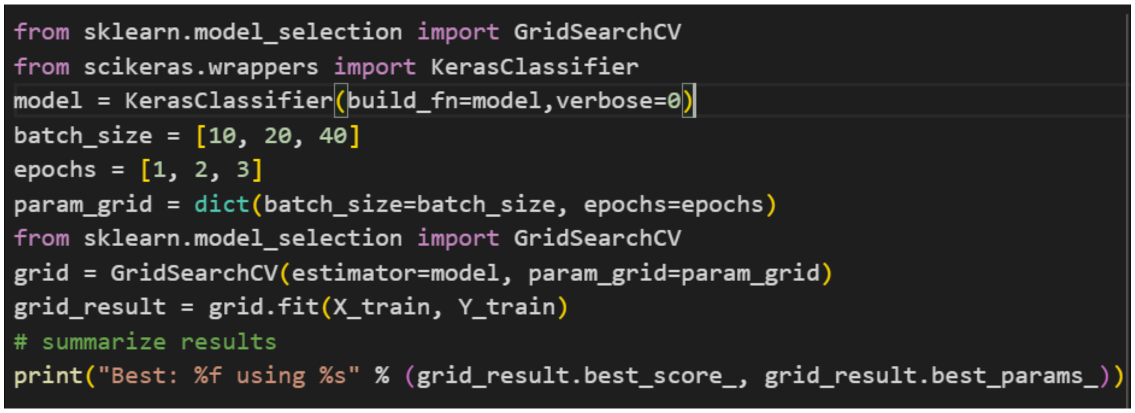
```python
lstm_out = 196
def createmodel():
    model = Sequential()
    model.add(Embedding(max_fatures, embed_dim,input_length = X.shape[1]))
    model.add(LSTM(lstm_out, dropout=0.2, recurrent_dropout=0.2))
    model.add(Dense(3,activation='softmax'))
    model.compile(loss = 'categorical_crossentropy', 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)

# Save the trained model to a file
model.save('sentiment_model.h5')
```

```
NING: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 - 64s - loss: 0.8251 - accuracy: 0.6411 - 64s/epoch - 219ms/step
/144 - 2s - loss: 0.7608 - accuracy: 0.6747 - 2s/epoch - 11ms/step
608269453048706
747487783432007
oss', 'accuracy']
r/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3079: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered le
aving_api.save_model(
```

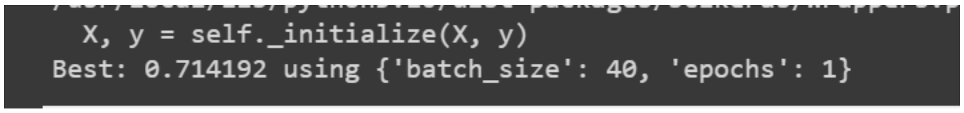


```
from keras.models import load_model

# Load the saved model
loaded_model = load_model('sentiment_model.h5')

# New text data
new_text = ["A lot of good things are happening. We are respected again throughout the world, and that's a great thing. @realDonaldTrump"]

# Preprocess the new text data
new_text = [text.lower() for text in new_text]
new_text = [re.sub('[^a-zA-z0-9\s]', '', text) for text in new_text]
new_text_sequences = tokenizer.texts_to_sequences(new_text)
new_text_padded = pad_sequences(new_text_sequences, maxlen=X.shape[1])

# Make predictions on the new text data
predictions = loaded_model.predict(new_text_padded)

# Convert predictions to sentiment labels
predicted_sentiments = labelencoder.inverse_transform(predictions.argmax(axis=1))

# Print the predicted sentiments
print(predicted_sentiments)
```

```
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.
1/1 [==============================] - 0s 251ms/step
['Negative']
```



```
from sklearn.model_selection import GridSearchCV
from scikeras.wrappers import KerasClassifier
model = KerasClassifier(build_fn=model,verbose=0)
batch_size = [10, 20, 40]
epochs = [1, 2, 3]
param_grid = dict(batch_size=batch_size, epochs=epochs)
from sklearn.model_selection import GridSearchCV
grid = GridSearchCV(estimator=model, param_grid=param_grid)
grid_result = grid.fit(X_train, Y_train)
# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
```



```
  X, y = self._initialize(X, y)
Best: 0.714192 using {'batch_size': 40, 'epochs': 1}
```

Github link: https://github.com/Harishwar-reddi/ICP_9