Neural Networks & Deep Learning Assignment 07 →ICP 07

Name: Azharuddin Mohammad

700#: 700756035

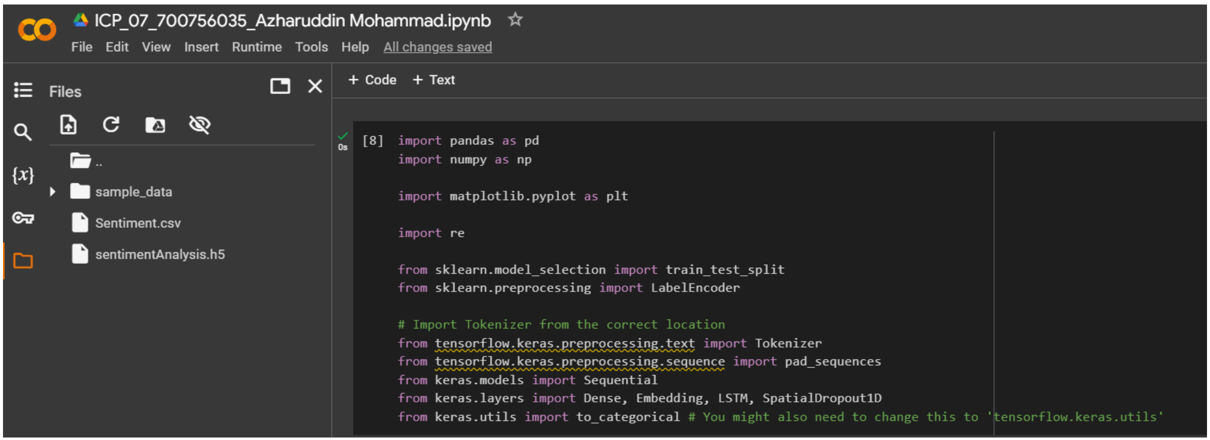
GitHub Link: https://github.com/mohammadazharuddin982/NN-DL_ICP_07_700756035_Azharuddin_Mohammad

Video Link: https://vimeo.com/1023119949/006db408a1?share=copy

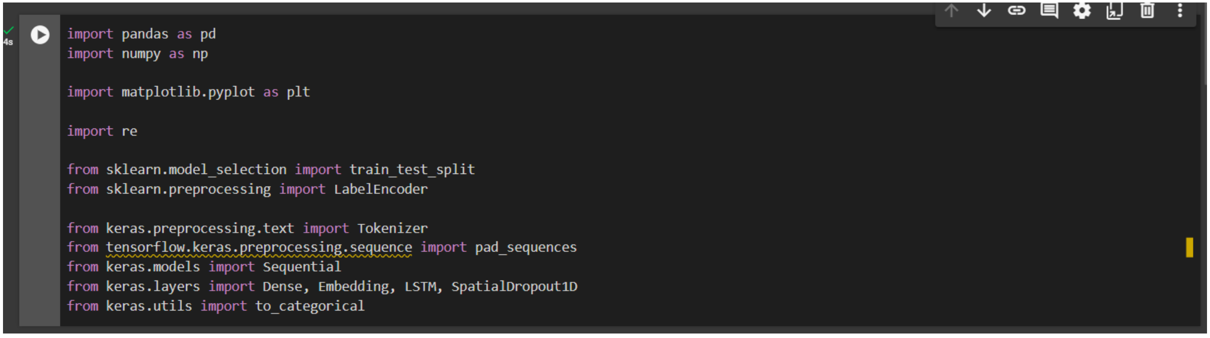
Programming elements:

1. Basics of LSTM
2. Types of RNN
3. Use case: Sentiment Analysis on the Twitter data set

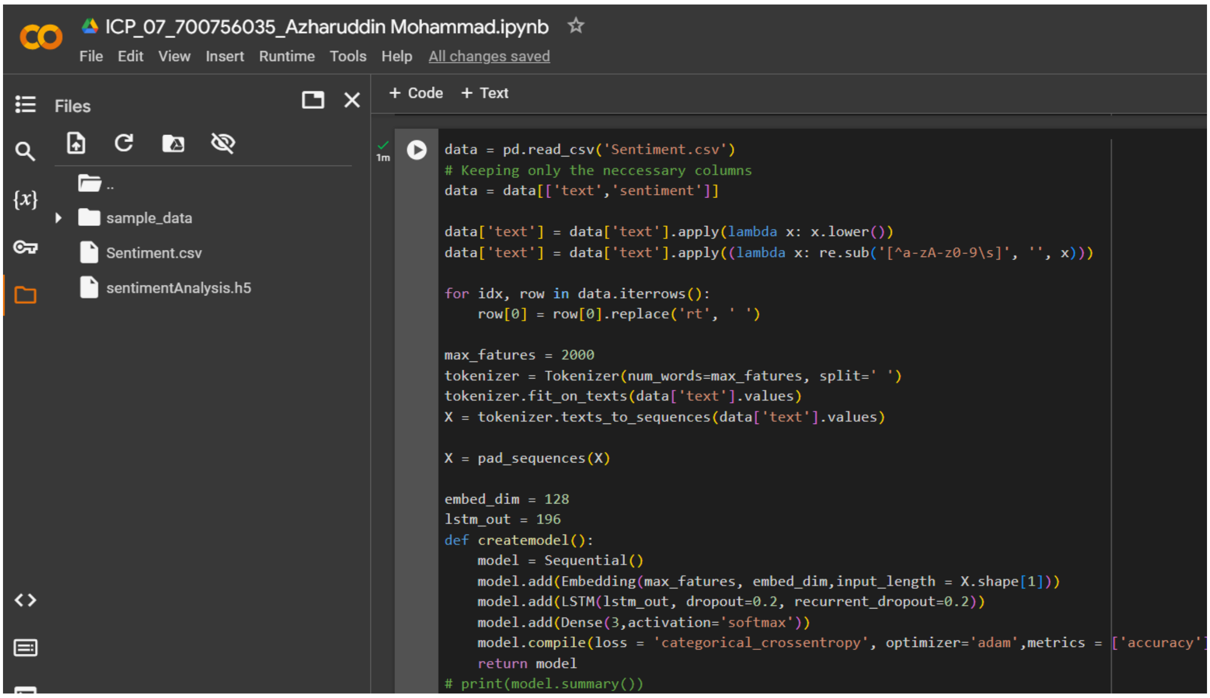
Question 01: Save the model and use the saved model to predict on new text data (ex, "A lot of good things are happening. We are respected again throughout the world, and that's a great thing.@realDonaldTrump")



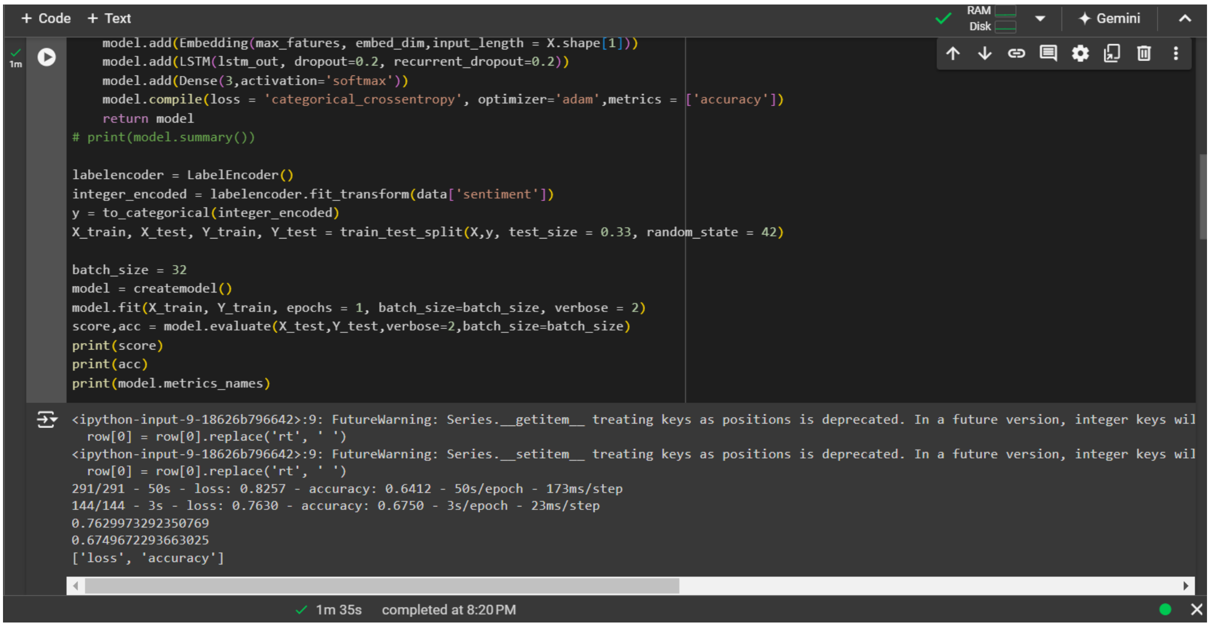
```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt

import re

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

# Import Tokenizer from the correct location
from tensorflow.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 # You might also need to change this to 'tensorflow.keras.utils'
```



```
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
```

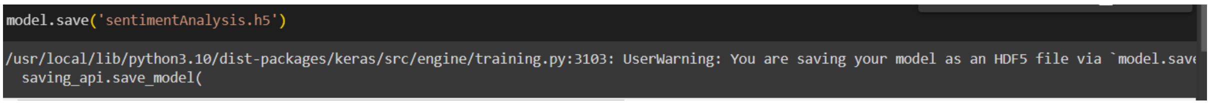

ICP_07_700756035_Azharuddin Mohammad.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Files
+ Code + Text
sample_data
Sentiment.csv
sentimentAnalysis.h5
1m

```
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())
```



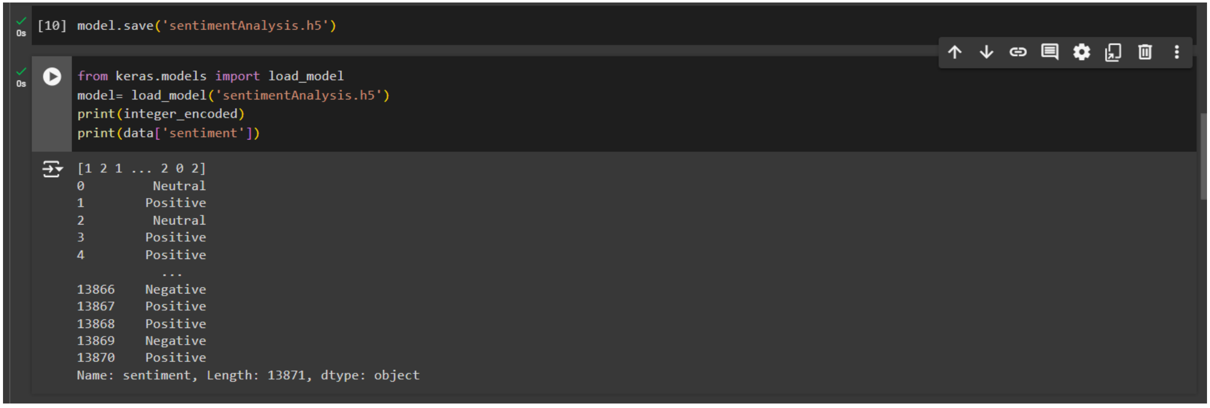
```
    model.add(Embedding(max_fatures, embed_dim,input_length = X.shape[1]))
    model.add(LSTM(lstm_out, dropout=0.2, recurrent_dropout=0.2))
    model.add(Dense(3,activation='softmax'))
    model.compile(loss = 'categorical_crossentropy', optimizer='adam',metrics = ['accuracy'])
    return model
# 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)
```

```
<ipython-input-9-18626b796642>:9: FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a future version, integer keys wil
  row[0] = row[0].replace('rt', ' ')
<ipython-input-9-18626b796642>:9: FutureWarning: Series.__setitem__ treating keys as positions is deprecated. In a future version, integer keys wil
  row[0] = row[0].replace('rt', ' ')
291/291 - 50s - loss: 0.8257 - accuracy: 0.6412 - 50s/epoch - 173ms/step
144/144 - 3s - loss: 0.7630 - accuracy: 0.6750 - 3s/epoch - 23ms/step
0.7629973292350769
0.6749672293663025
['loss', 'accuracy']
```



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

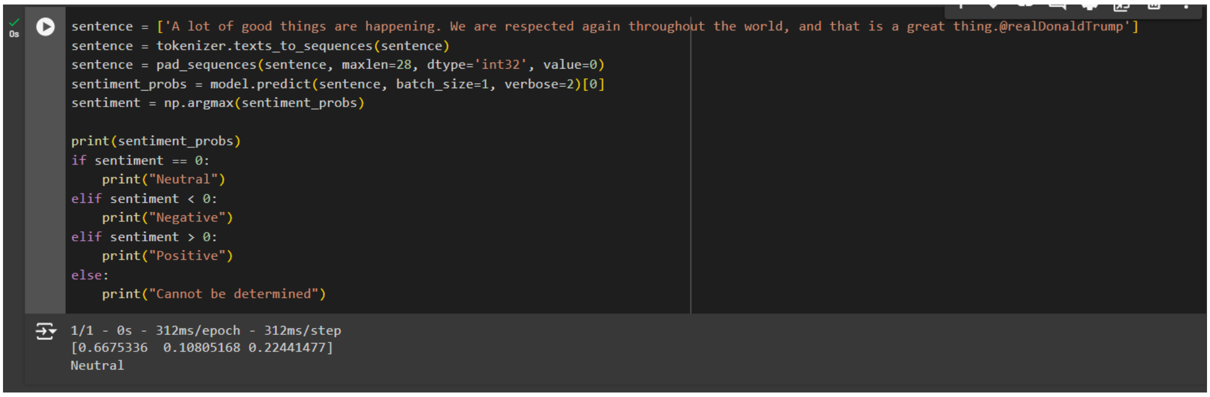
```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `model.save
  saving_api.save_model(
```



```
[10] model.save('sentimentAnalysis.h5')
```

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

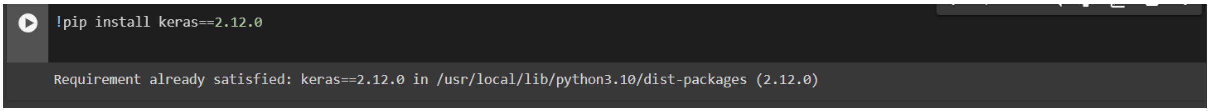
```
[1 2 1 ... 2 0 2]
0         Neutral
1        Positive
2         Neutral
3        Positive
4        Positive
           ...
13866    Negative
13867    Positive
13868    Positive
13869    Negative
13870    Positive
Name: sentiment, Length: 13871, dtype: object
```



```
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")
```

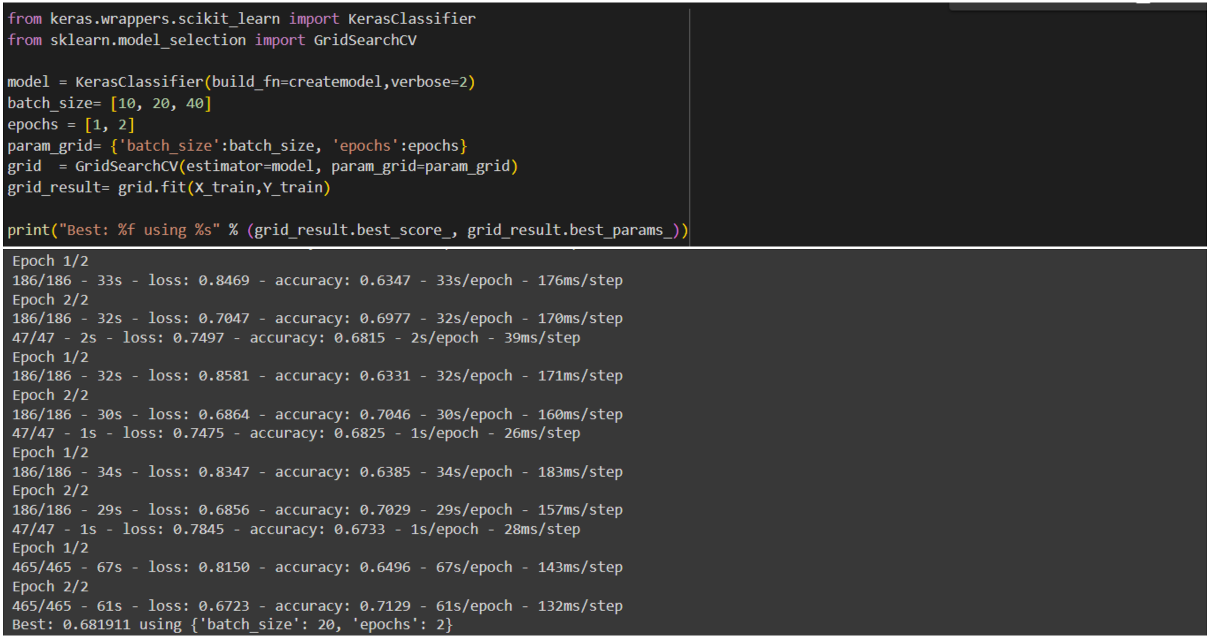
```
1/1 - 0s - 312ms/epoch - 312ms/step
[0.6675336  0.10805168 0.22441477]
Neutral
```

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



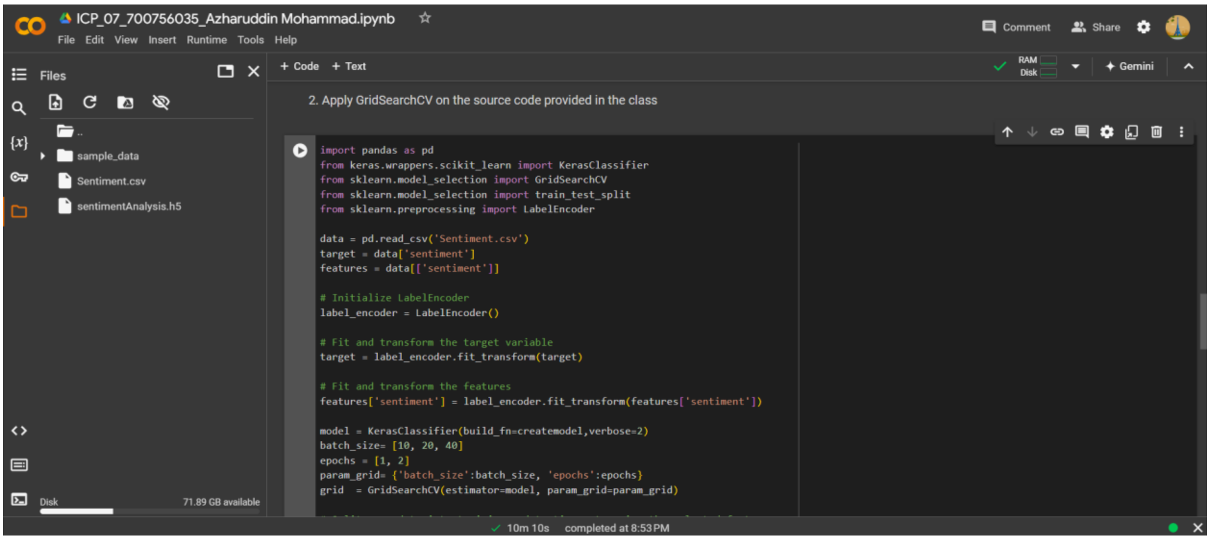
```
!pip install keras==2.12.0
```

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



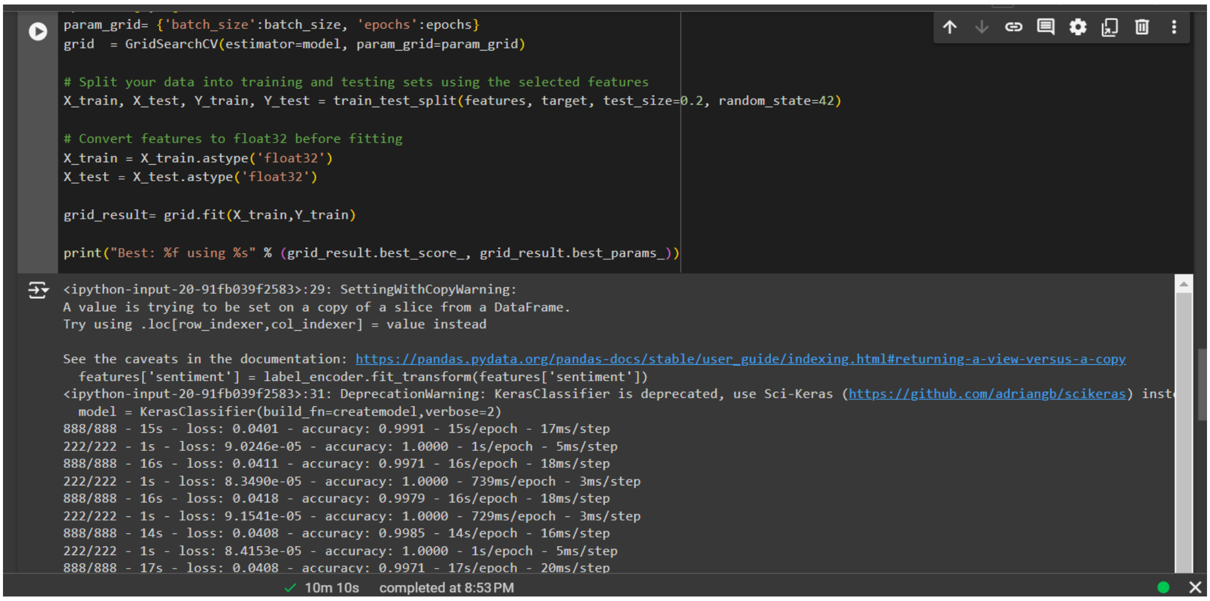
```
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import GridSearchCV

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}
```



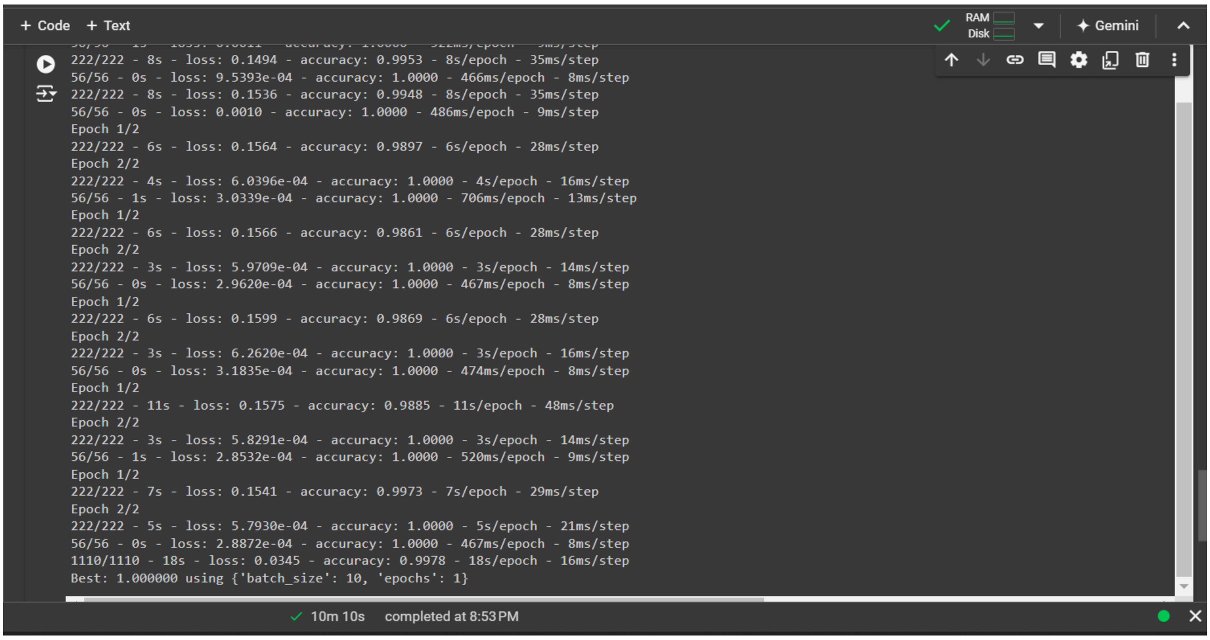
2. Apply GridSearchCV on the source code provided in the class

```
import pandas as pd
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

data = pd.read_csv('Sentiment.csv')
target = data['sentiment']
features = data[['sentiment']]

# Initialize LabelEncoder
label_encoder = LabelEncoder()

# Fit and transform the target variable
target = label_encoder.fit_transform(target)

# Fit and transform the features
features['sentiment'] = label_encoder.fit_transform(features['sentiment'])

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)
```



```
param_grid= {'batch_size':batch_size, 'epochs':epochs}
grid  = GridSearchCV(estimator=model, param_grid=param_grid)

# Split your data into training and testing sets using the selected features
X_train, X_test, Y_train, Y_test = train_test_split(features, target, test_size=0.2, random_state=42)

# Convert features to float32 before fitting
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')

grid_result= grid.fit(X_train,Y_train)

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

```
<ipython-input-20-91fb039f2583>:29: 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
  features['sentiment'] = label_encoder.fit_transform(features['sentiment'])
<ipython-input-20-91fb039f2583>:31: DeprecationWarning: KerasClassifier is deprecated, use Sci-Keras (https://github.com/adriangb/scikeras) inst
  model = KerasClassifier(build_fn=createmodel,verbose=2)
888/888 - 15s - loss: 0.0401 - accuracy: 0.9991 - 15s/epoch - 17ms/step
222/222 - 1s - loss: 9.0246e-05 - accuracy: 1.0000 - 1s/epoch - 5ms/step
888/888 - 16s - loss: 0.0411 - accuracy: 0.9971 - 16s/epoch - 18ms/step
222/222 - 1s - loss: 8.3490e-05 - accuracy: 1.0000 - 739ms/epoch - 3ms/step
888/888 - 16s - loss: 0.0418 - accuracy: 0.9979 - 16s/epoch - 18ms/step
222/222 - 1s - loss: 9.1541e-05 - accuracy: 1.0000 - 729ms/epoch - 3ms/step
888/888 - 14s - loss: 0.0408 - accuracy: 0.9985 - 14s/epoch - 16ms/step
222/222 - 1s - loss: 8.4153e-05 - accuracy: 1.0000 - 1s/epoch - 5ms/step
888/888 - 17s - loss: 0.0408 - accuracy: 0.9971 - 17s/epoch - 20ms/step
```



```
222/222 - 8s - loss: 0.1494 - accuracy: 0.9953 - 8s/epoch - 35ms/step
56/56 - 0s - loss: 9.5393e-04 - accuracy: 1.0000 - 466ms/epoch - 8ms/step
222/222 - 8s - loss: 0.1536 - accuracy: 0.9948 - 8s/epoch - 35ms/step
56/56 - 0s - loss: 0.0010 - accuracy: 1.0000 - 486ms/epoch - 9ms/step
Epoch 1/2
222/222 - 6s - loss: 0.1564 - accuracy: 0.9897 - 6s/epoch - 28ms/step
Epoch 2/2
222/222 - 4s - loss: 6.0396e-04 - accuracy: 1.0000 - 4s/epoch - 16ms/step
56/56 - 1s - loss: 3.0339e-04 - accuracy: 1.0000 - 706ms/epoch - 13ms/step
Epoch 1/2
222/222 - 6s - loss: 0.1566 - accuracy: 0.9861 - 6s/epoch - 28ms/step
Epoch 2/2
222/222 - 3s - loss: 5.9709e-04 - accuracy: 1.0000 - 3s/epoch - 14ms/step
56/56 - 0s - loss: 2.9620e-04 - accuracy: 1.0000 - 467ms/epoch - 8ms/step
Epoch 1/2
222/222 - 6s - loss: 0.1599 - accuracy: 0.9869 - 6s/epoch - 28ms/step
Epoch 2/2
222/222 - 3s - loss: 6.2620e-04 - accuracy: 1.0000 - 3s/epoch - 16ms/step
56/56 - 0s - loss: 3.1835e-04 - accuracy: 1.0000 - 474ms/epoch - 8ms/step
Epoch 1/2
222/222 - 11s - loss: 0.1575 - accuracy: 0.9885 - 11s/epoch - 48ms/step
Epoch 2/2
222/222 - 3s - loss: 5.8291e-04 - accuracy: 1.0000 - 3s/epoch - 14ms/step
56/56 - 1s - loss: 2.8532e-04 - accuracy: 1.0000 - 520ms/epoch - 9ms/step
Epoch 1/2
222/222 - 7s - loss: 0.1541 - accuracy: 0.9973 - 7s/epoch - 29ms/step
Epoch 2/2
222/222 - 5s - loss: 5.7930e-04 - accuracy: 1.0000 - 5s/epoch - 21ms/step
56/56 - 0s - loss: 2.8872e-04 - accuracy: 1.0000 - 467ms/epoch - 8ms/step
1110/1110 - 18s - loss: 0.0345 - accuracy: 0.9978 - 18s/epoch - 16ms/step
Best: 1.000000 using {'batch_size': 10, 'epochs': 1}
```