Spring 2024: CS5720

Neural Networks & Deep Learning – ICP10

NAME:NANDIMANDALAM VENKATA VINAY VARMA

STUDENT ID:700745193

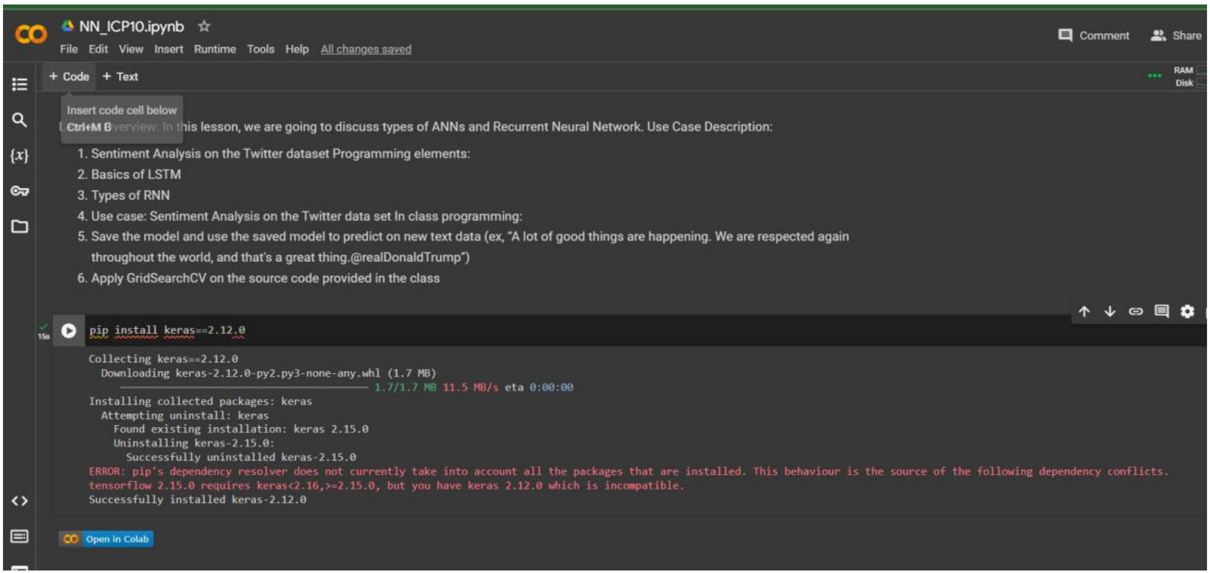
GitHub Link:

https://github.com/venkatavinayvarma/NeuralNetworks_ICP10.git

GitHub Link:

Video Link: https://drive.google.com/drive/folders/1B0X1eq38WGeVXGh2-kyPpdM1e71SFWM5?usp=sharing

Lesson Overview: In this lesson, we are going to discuss types of ANNs and Recurrent Neural Network. Use Case Description: 1. Sentiment Analysis on the Twitter dataset Programming elements: 1. Basics of LSTM 2. Types of RNN 3. Use case: Sentiment Analysis on the Twitter data set In class programming: 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”) 2. Apply GridSearchCV on the source code provided in the class

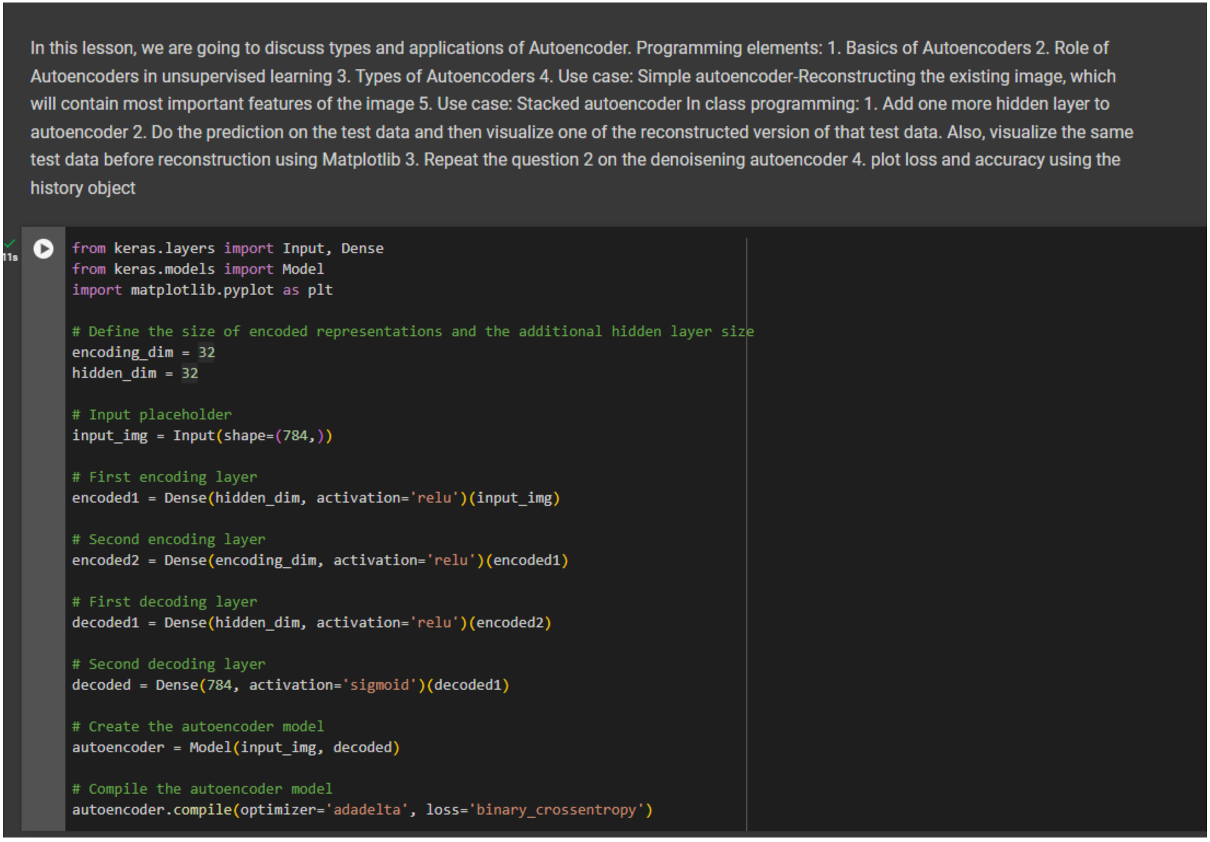


Lesson Overview: In this lesson, we are going to discuss types of ANNs and Recurrent Neural Network. Use Case Description:

1. Sentiment Analysis on the Twitter dataset Programming elements:
2. Basics of LSTM
3. Types of RNN
4. Use case: Sentiment Analysis on the Twitter data set In class programming:
5. 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")
6. Apply GridSearchCV on the source code provided in the class

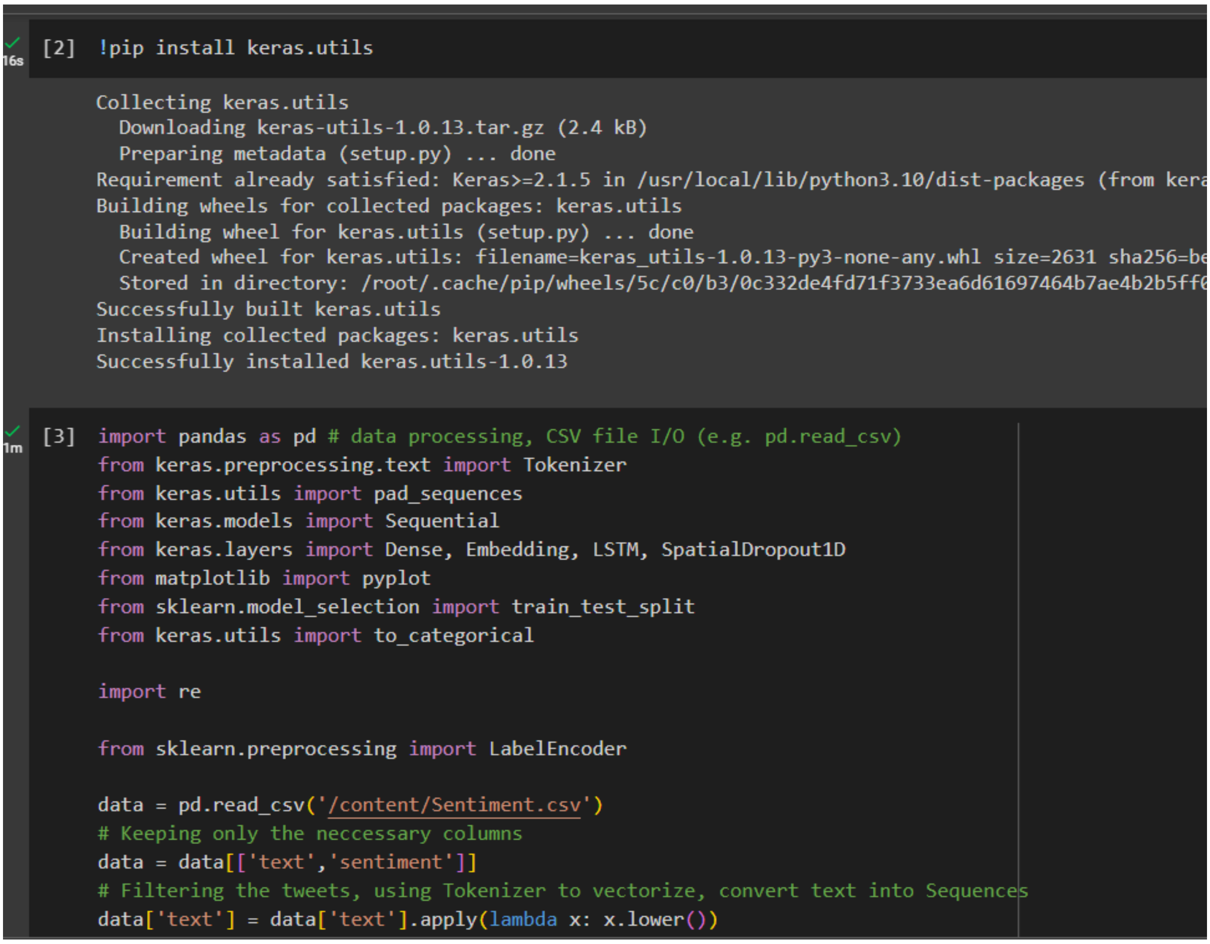
```
pip install keras==2.12.0
```

```
Collecting keras==2.12.0
  Downloading keras-2.12.0-py2.py3-none-any.whl (1.7 MB)
                                             ━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 11.5 MB/s eta 0:00:00
Installing collected packages: keras
  Attempting uninstall: keras
    Found existing installation: keras 2.15.0
    Uninstalling keras-2.15.0:
      Successfully uninstalled keras-2.15.0
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
tensorflow 2.15.0 requires keras<2.16,>=2.15.0, but you have keras 2.12.0 which is incompatible.
Successfully installed keras-2.12.0
```



In this lesson, we are going to discuss types and applications of Autoencoder. Programming elements: 1. Basics of Autoencoders 2. Role of Autoencoders in unsupervised learning 3. Types of Autoencoders 4. Use case: Simple autoencoder-Reconstructing the existing image, which will contain most important features of the image 5. Use case: Stacked autoencoder In class programming: 1. Add one more hidden layer to autoencoder 2. Do the prediction on the test data and then visualize one of the reconstructed version of that test data. Also, visualize the same test data before reconstruction using Matplotlib 3. Repeat the question 2 on the denoisening autoencoder 4. plot loss and accuracy using the history object

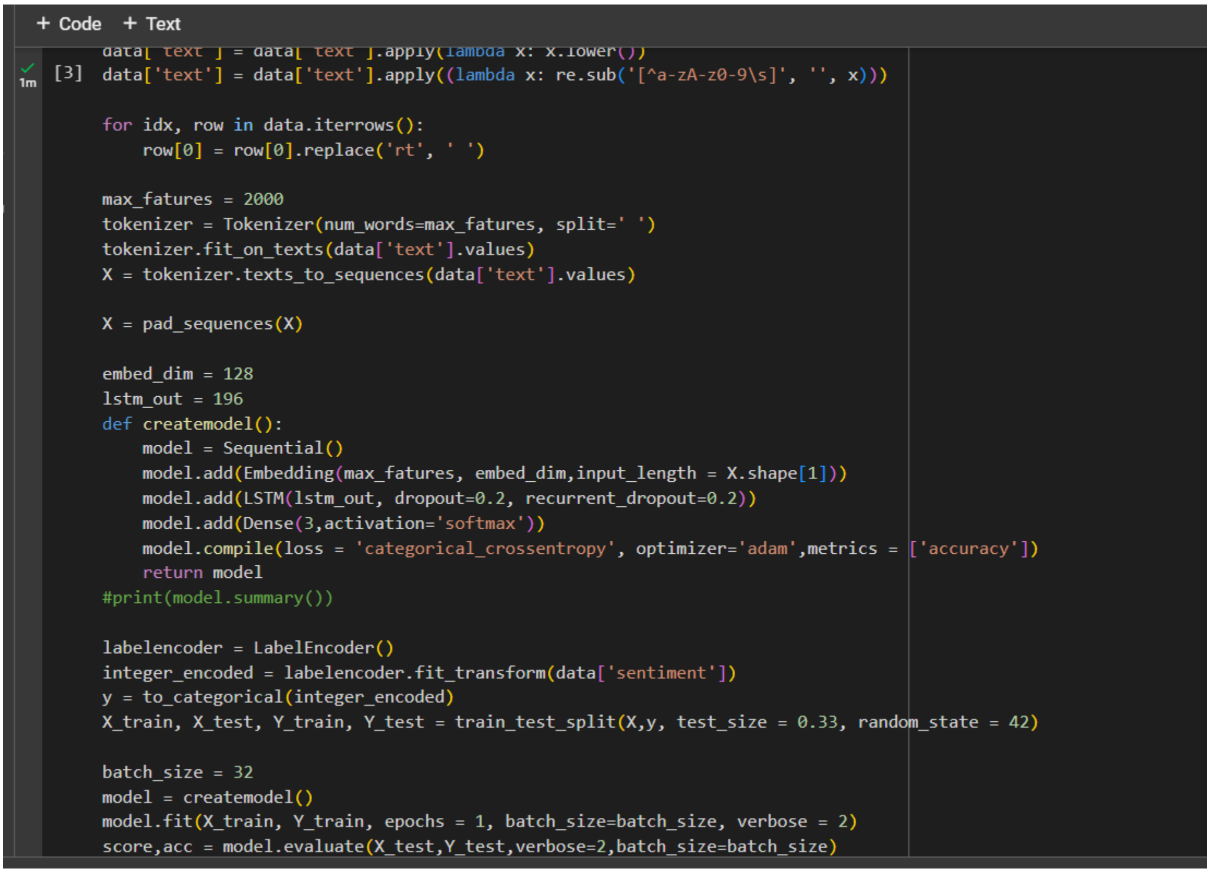
```python
from keras.layers import Input, Dense
from keras.models import Model
import matplotlib.pyplot as plt

# Define the size of encoded representations and the additional hidden layer size
encoding_dim = 32
hidden_dim = 32

# Input placeholder
input_img = Input(shape=(784,))

# First encoding layer
encoded1 = Dense(hidden_dim, activation='relu')(input_img)

# Second encoding layer
encoded2 = Dense(encoding_dim, activation='relu')(encoded1)

# First decoding layer
decoded1 = Dense(hidden_dim, activation='relu')(encoded2)

# Second decoding layer
decoded = Dense(784, activation='sigmoid')(decoded1)

# Create the autoencoder model
autoencoder = Model(input_img, decoded)

# Compile the autoencoder model
autoencoder.compile(optimizer='adadelta', loss='binary_crossentropy')
```



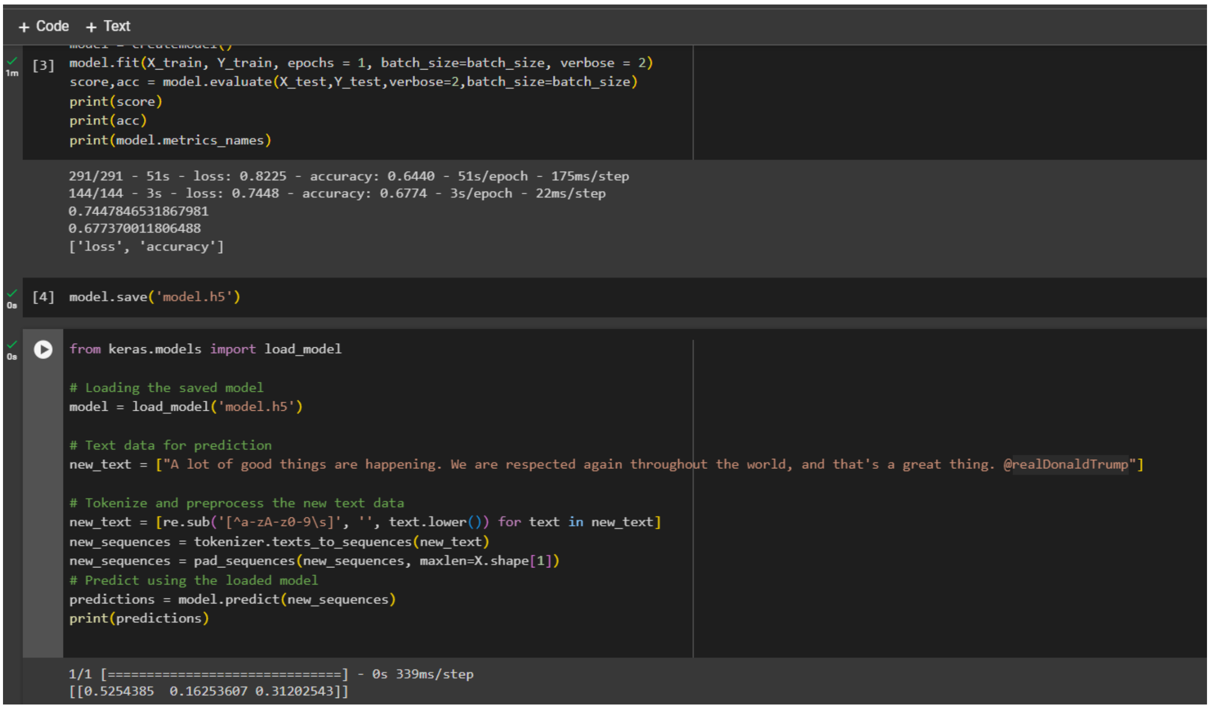
```
!pip install keras.utils
```

```
Collecting keras.utils
  Downloading keras-utils-1.0.13.tar.gz (2.4 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: Keras>=2.1.5 in /usr/local/lib/python3.10/dist-packages (from kera
Building wheels for collected packages: keras.utils
  Building wheel for keras.utils (setup.py) ... done
  Created wheel for keras.utils: filename=keras_utils-1.0.13-py3-none-any.whl size=2631 sha256=be
  Stored in directory: /root/.cache/pip/wheels/5c/c0/b3/0c332de4fd71f3733ea6d61697464b7ae4b2b5ff0
Successfully built keras.utils
Installing collected packages: keras.utils
Successfully installed keras.utils-1.0.13
```

```
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from keras.preprocessing.text import Tokenizer
from keras.utils 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 keras.utils import to_categorical

import re

from sklearn.preprocessing import LabelEncoder

data = pd.read_csv('/content/Sentiment.csv')
# Keeping only the neccessary columns
data = data[['text','sentiment']]
# Filtering the tweets, using Tokenizer to vectorize, convert text into Sequences
data['text'] = data['text'].apply(lambda x: x.lower())
```



```
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)
```



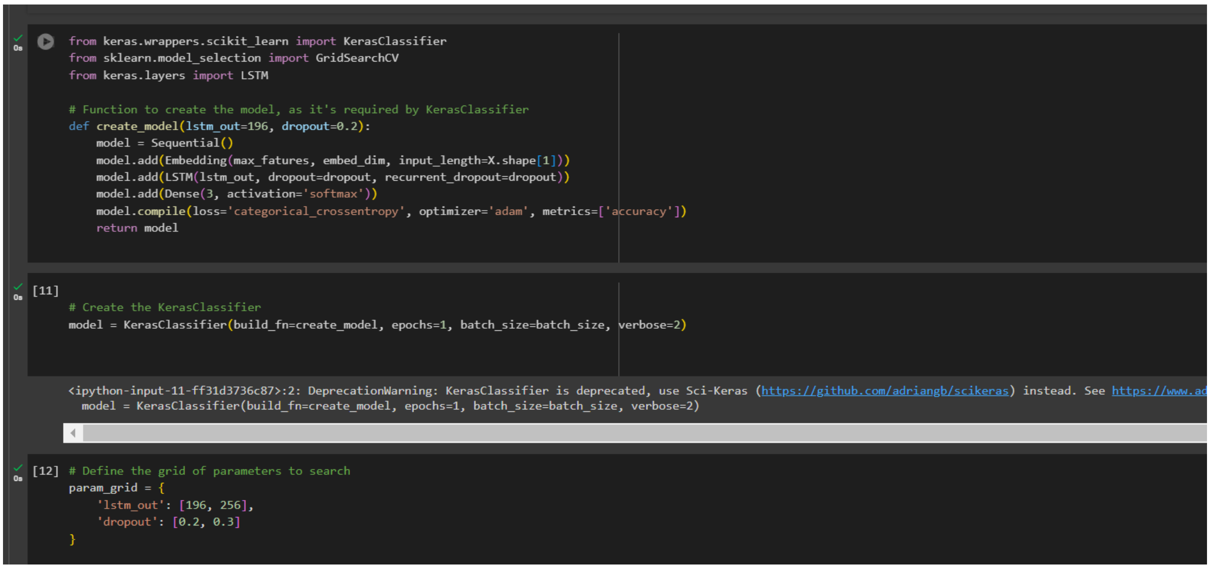
```
[3] model.fit(X_train, Y_train, epochs = 1, batch_size=batch_size, verbose = 2)
    score,acc = model.evaluate(X_test,Y_test,verbose=2,batch_size=batch_size)
    print(score)
    print(acc)
    print(model.metrics_names)
```

```
291/291 - 51s - loss: 0.8225 - accuracy: 0.6440 - 51s/epoch - 175ms/step
144/144 - 3s - loss: 0.7448 - accuracy: 0.6774 - 3s/epoch - 22ms/step
0.7447846531867981
0.677370011806488
['loss', 'accuracy']
```

```
[4] model.save('model.h5')
```

```
from keras.models import load_model

# Loading the saved model
model = load_model('model.h5')

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

# Tokenize and preprocess the new text data
new_text = [re.sub('[^a-zA-z0-9\s]', '', text.lower()) for text in new_text]
new_sequences = tokenizer.texts_to_sequences(new_text)
new_sequences = pad_sequences(new_sequences, maxlen=X.shape[1])
# Predict using the loaded model
predictions = model.predict(new_sequences)
print(predictions)
```

```
1/1 [==============================] - 0s 339ms/step
[[0.5254385  0.16253607 0.31202543]]
```

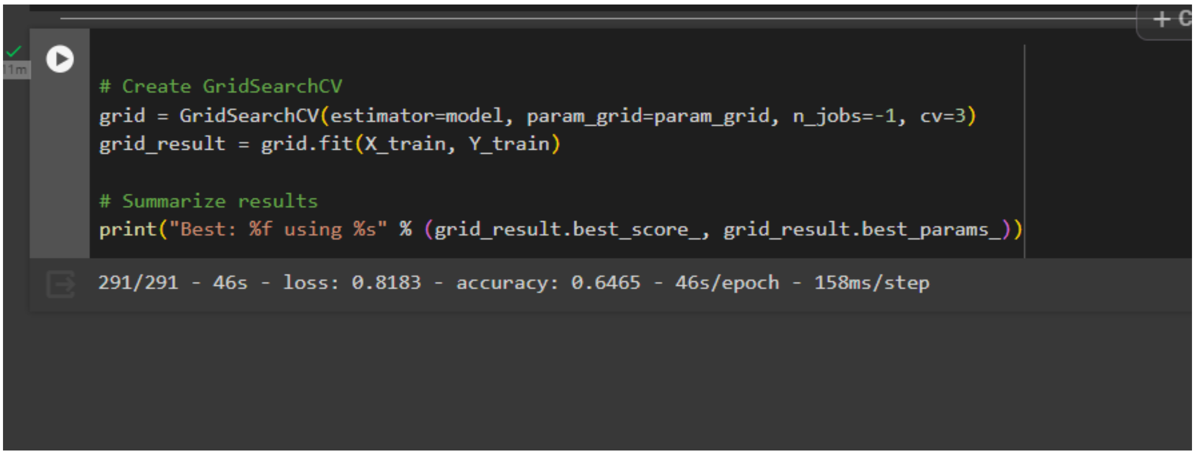


```
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import GridSearchCV
from keras.layers import LSTM

# Function to create the model, as it's required by KerasClassifier
def create_model(lstm_out=196, dropout=0.2):
    model = Sequential()
    model.add(Embedding(max_fatures, embed_dim, input_length=X.shape[1]))
    model.add(LSTM(lstm_out, dropout=dropout, recurrent_dropout=dropout))
    model.add(Dense(3, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model
```

```
[11]
# Create the KerasClassifier
model = KerasClassifier(build_fn=create_model, epochs=1, batch_size=batch_size, verbose=2)
```

```
<ipython-input-11-ff31d3736c87>:2: DeprecationWarning: KerasClassifier is deprecated, use Sci-Keras (https://github.com/adriangb/scikeras) instead. See https://www.ad
  model = KerasClassifier(build_fn=create_model, epochs=1, batch_size=batch_size, verbose=2)
```

```
[12] # Define the grid of parameters to search
param_grid = {
    'lstm_out': [196, 256],
    'dropout': [0.2, 0.3]
}
```



```
# Create GridSearchCV
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1, cv=3)
grid_result = grid.fit(X_train, Y_train)

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

```
291/291 - 46s - loss: 0.8183 - accuracy: 0.6465 - 46s/epoch - 158ms/step
```