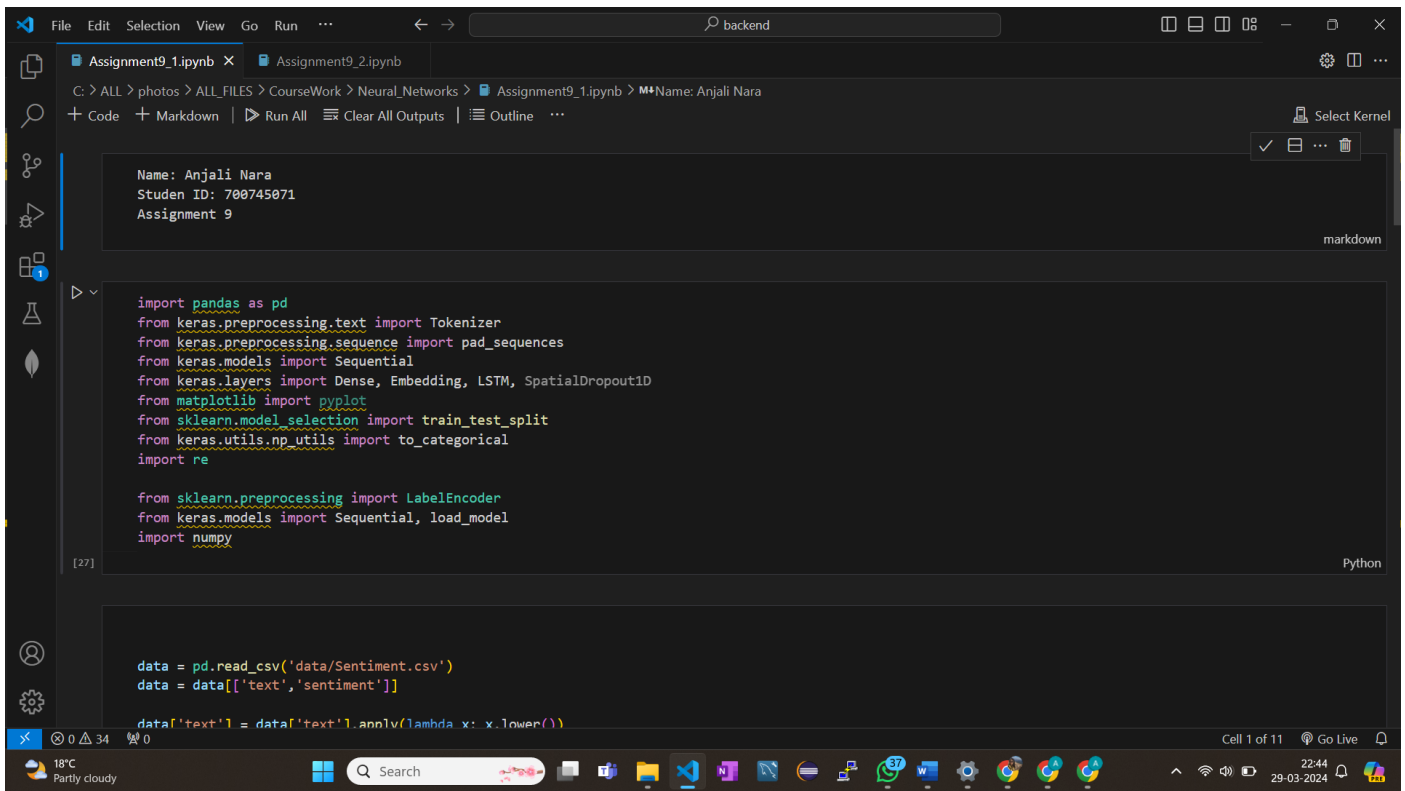


Github link: [https://github.com/naraanjali/Assignment\\_9](https://github.com/naraanjali/Assignment_9)

Video link: [https://drive.google.com/file/d/150q3K7\\_EFxcFk60k2B6NjAKdVwrfWNkh/view?usp=sharing](https://drive.google.com/file/d/150q3K7_EFxcFk60k2B6NjAKdVwrfWNkh/view?usp=sharing)

1.

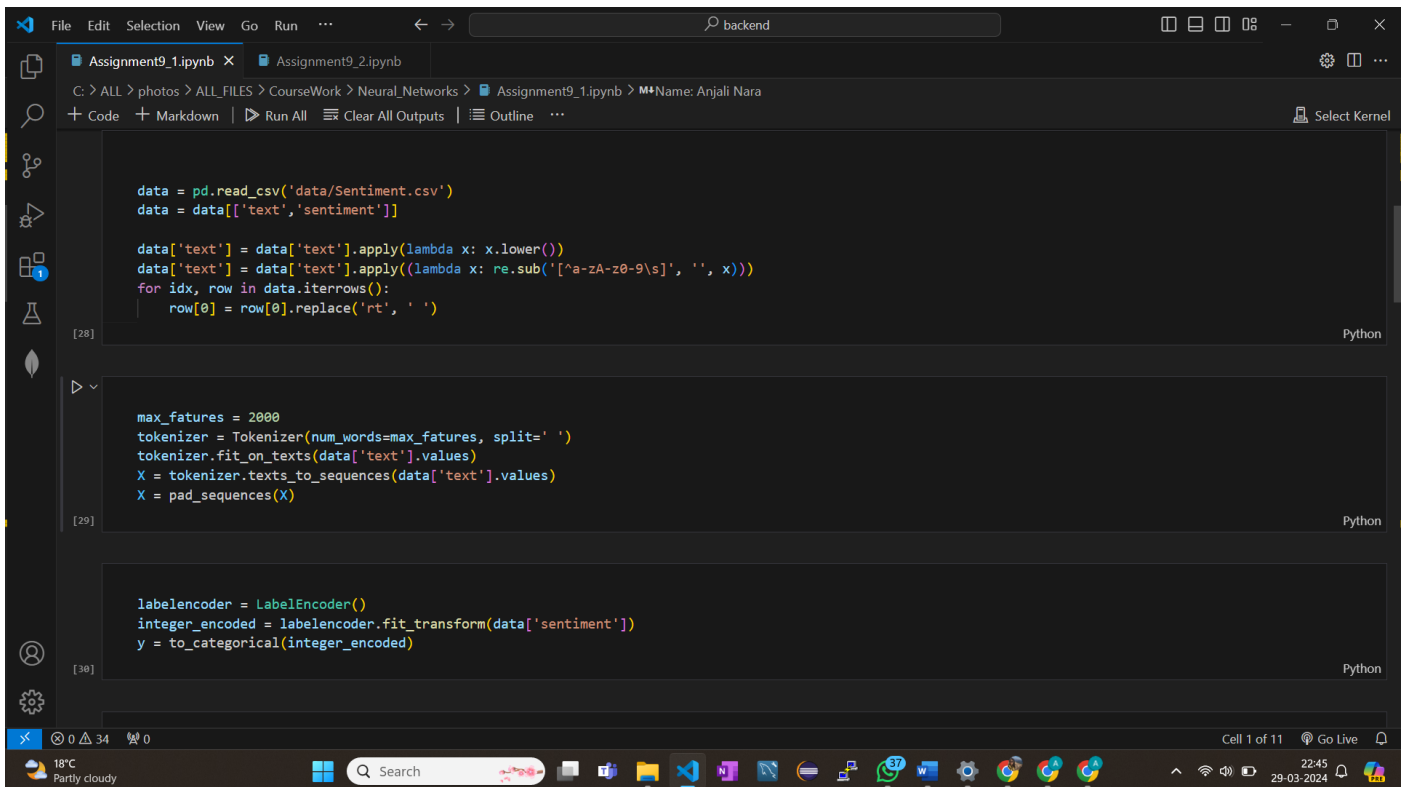


```
File Edit Selection View Go Run ... backend
Assignment9_1.ipynb Assignment9_2.ipynb
C: > ALL > photos > ALL_FILES > CourseWork > Neural_Networks > Assignment9_1.ipynb > M4Name: Anjali Nara
+ Code + Markdown ▶ Run All ⇄ Clear All Outputs | Outline ...
Select Kernel
Name: Anjali Nara
Studen ID: 700745071
Assignment 9
markdown
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 keras.utils.np_utils import to_categorical
import re

from sklearn.preprocessing import LabelEncoder
from keras.models import Sequential, load_model
import numpy

[27] Python
data = pd.read_csv('data/Sentiment.csv')
data = data[['text', 'sentiment']]

data['text'] = data['text'].apply(lambda x: x.lower())
Cell 1 of 11 Go Live
18°C Partly cloudy 22:44 29-03-2024
```



```
File Edit Selection View Go Run ... backend
Assignment9_1.ipynb Assignment9_2.ipynb
C: > ALL > photos > ALL_FILES > CourseWork > Neural_Networks > Assignment9_1.ipynb > M4Name: Anjali Nara
+ Code + Markdown ▶ Run All ⇄ Clear All Outputs | Outline ...
Select Kernel
data = pd.read_csv('data/Sentiment.csv')
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', ' ')
[28] Python

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)
[29] Python

labelencoder = LabelEncoder()
integer_encoded = labelencoder.fit_transform(data['sentiment'])
y = to_categorical(integer_encoded)
[30] Python
Cell 1 of 11 Go Live
18°C Partly cloudy 22:45 29-03-2024
```

File Edit Selection View Go Run ... backend

Assignment9\_1.ipynb Assignment9\_2.ipynb

C: > ALL > photos > ALL\_FILES > CourseWork > Neural\_Networks > Assignment9\_1.ipynb > M+Name: Anjali Nara

+ Code + Markdown | Run All | Clear All Outputs | Outline ... Select Kernel

```
[31] X_train, X_test, Y_train, Y_test = train_test_split(X,y, test_size = 0.33, random_state = 42)
```

Python

```
[32] 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
```

Python

```
[33] 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)
```

Python

Cell 1 of 11 Go Live

18°C Partly cloudy

Search

22:45 29-03-2024

File Edit Selection View Go Run ... backend

Assignment9\_1.ipynb Assignment9\_2.ipynb

C: > ALL > photos > ALL\_FILES > CourseWork > Neural\_Networks > Assignment9\_1.ipynb > M+Name: Anjali Nara

+ Code + Markdown | Run All | Clear All Outputs | Outline ... Select Kernel

```
[33] print(model.metrics_names)
```

Python

```
... 291/291 - 32s - loss: 0.8264 - accuracy: 0.6423
144/144 - 2s - loss: 0.7607 - accuracy: 0.6693
0.7606955766677856
0.669287919998169
['loss', 'accuracy']
```

```
[34] model.save('model.h5')
mod = load_model('model.h5')

print(mod.summary())
```

Python

```
... Model: "sequential_4"
```

Layer (type)	Output Shape	Param #
=====		
embedding_4 (Embedding)	(None, 28, 128)	256000
-----		
lstm_4 (LSTM)	(None, 196)	254800
-----		
dense_4 (Dense)	(None, 3)	591
=====		
Total params: 511,391		
Trainable params: 511,391		
Non-trainable params: 0		

Cell 1 of 11 Go Live

18°C Partly cloudy

Search

22:45 29-03-2024

File Edit Selection View Go Run

backend

Assignment9\_1.ipynb X Assignment9\_2.ipynb

C: > ALL > photos > ALL\_FILES > CourseWork > Neural\_Networks > Assignment9\_1.ipynb > M4Name: Anjali Nara

+ Code + Markdown ▶ Run All ⌵ Clear All Outputs | Outline ...

Select Kernel

None

```
txt = [['A lot of good things are happening. We are respected again throughout the world, and thats a great '
        'thing.@realDonaldTrump']]
max_data = pd.DataFrame(txt, index=range(0, 1, 1), columns=list('t'))
max_data['t'] = max_data['t'].apply(lambda x: x.lower())
max_data['t'] = max_data['t'].apply(lambda x: re.sub('[^a-zA-z0-9\s]', '', x))
features = 2000
tokenizer = Tokenizer(num_words=features, split=' ')
tokenizer.fit_on_texts(max_data['t'].values)
X = tokenizer.texts_to_sequences(max_data['t'].values)
X = pad_sequences(X, maxlen=28)

out = mod.predict(X)
print(out)
print(numpy.where(max(out[0])), ":", (max(out[0])))
print(numpy.argmax(out))
print(mod.summary())
```

[36]

Python

... [[0.80123734 0.07214491 0.12661776]]  
(array([0], dtype=int64),) : 0.80123734  
0  
Model: "sequential\_4"

Layer (type)	Output Shape	Param #
embedding_4 (Embedding)	(None, 28, 128)	256000

Cell 1 of 11 Go Live

18°C Partly cloudy Search

File Edit Selection View Go Run

backend

Assignment9\_1.ipynb X Assignment9\_2.ipynb

C: > ALL > photos > ALL\_FILES > CourseWork > Neural\_Networks > Assignment9\_1.ipynb > M4Name: Anjali Nara

+ Code + Markdown ▶ Run All ⌵ Clear All Outputs | Outline ...

Select Kernel

```
tokenizer.fit_on_texts(max_data['t'].values)
X = tokenizer.texts_to_sequences(max_data['t'].values)
X = pad_sequences(X, maxlen=28)

out = mod.predict(X)
print(out)
print(numpy.where(max(out[0])), ":", (max(out[0])))
print(numpy.argmax(out))
print(mod.summary())
```

[36]

Python

... [[0.80123734 0.07214491 0.12661776]]  
(array([0], dtype=int64),) : 0.80123734  
0  
Model: "sequential\_4"

Layer (type)	Output Shape	Param #
embedding_4 (Embedding)	(None, 28, 128)	256000
lstm_4 (LSTM)	(None, 196)	254800
dense_4 (Dense)	(None, 3)	591

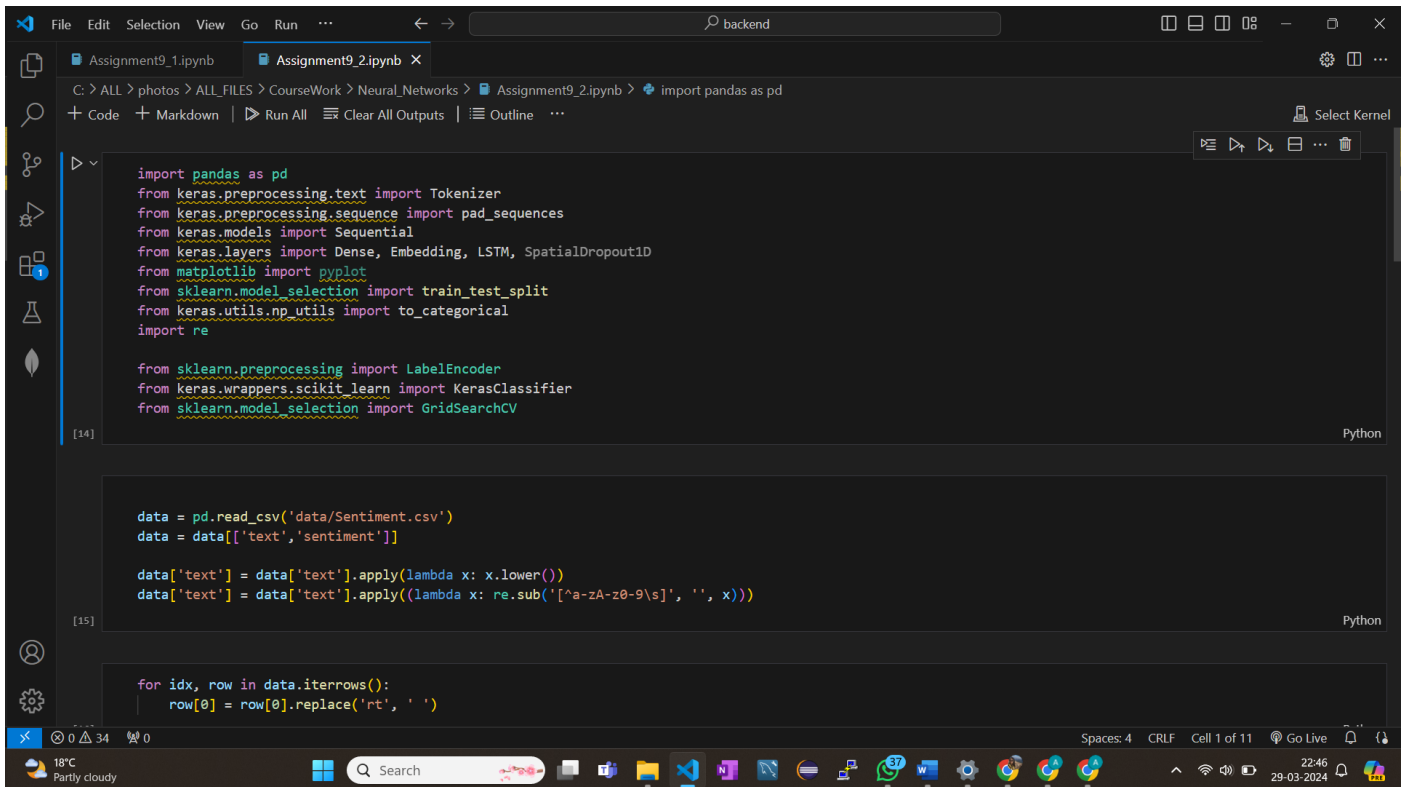
Total params: 511,391  
Trainable params: 511,391  
Non-trainable params: 0

None

Cell 1 of 11 Go Live

18°C Partly cloudy Search

2.



The screenshot shows a Jupyter Notebook interface with the file 'Assignment9\_2.ipynb' open. The notebook is running on a Python kernel. The first three code cells are visible:

```
[14]: 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 keras.utils.np_utils import to_categorical
      import re

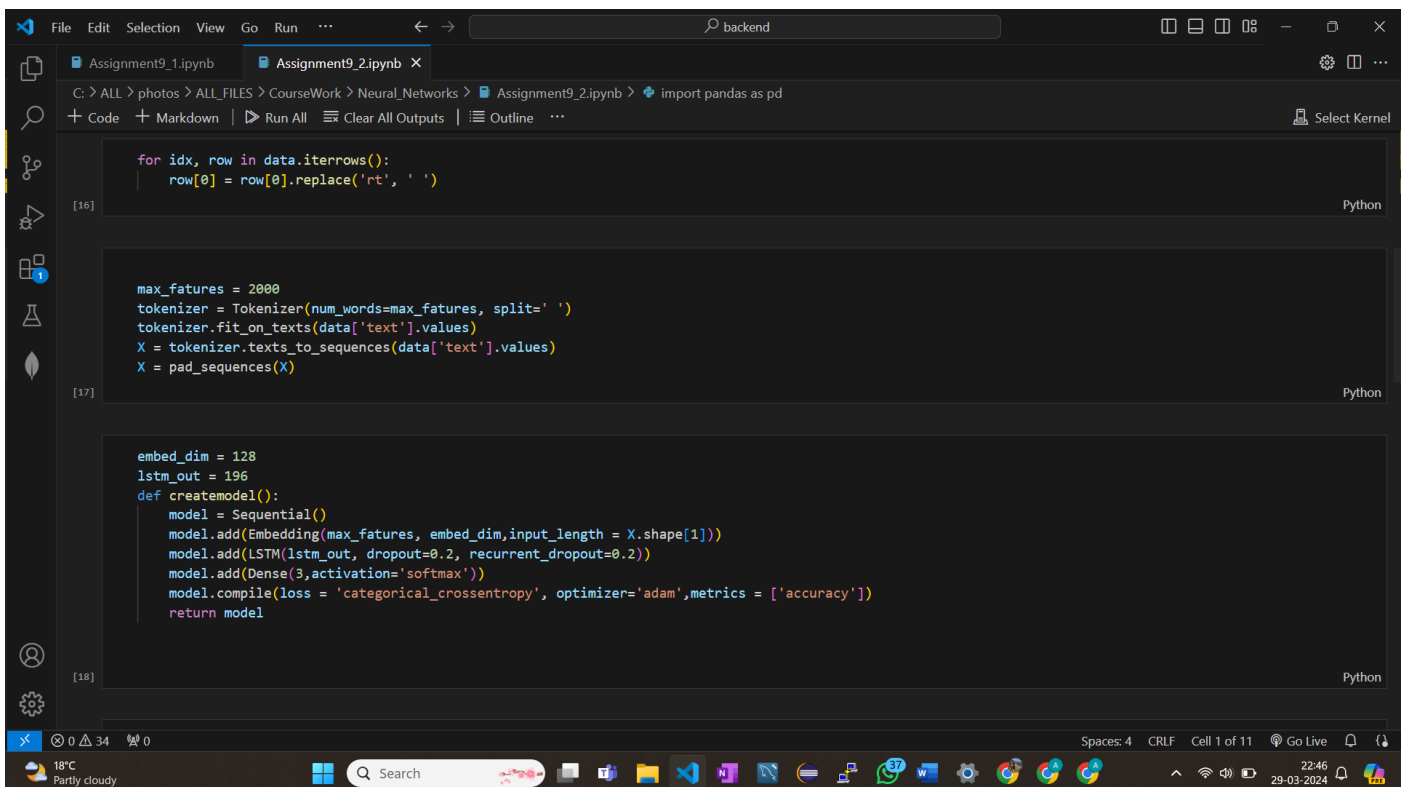
      from sklearn.preprocessing import LabelEncoder
      from keras.wrappers.scikit_learn import KerasClassifier
      from sklearn.model_selection import GridSearchCV
```

```
[15]: data = pd.read_csv('data/Sentiment.csv')
      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', ' ')
```

The bottom status bar shows 'Spaces: 4', 'CRLF', 'Cell 1 of 11', 'Go Live', and the time '22:46 29-03-2024'.



The screenshot shows the next three code cells of the Jupyter Notebook:

```
[16]: for idx, row in data.iterrows():
      row[0] = row[0].replace('rt', ' ')
```

```
[17]: max_features = 2000
      tokenizer = Tokenizer(num_words=max_features, split=' ')
      tokenizer.fit_on_texts(data['text'].values)
      X = tokenizer.texts_to_sequences(data['text'].values)
      X = pad_sequences(X)
```

```
[18]: embed_dim = 128
      lstm_out = 196
      def createmodel():
          model = Sequential()
          model.add(Embedding(max_features, 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
```

The bottom status bar shows 'Spaces: 4', 'CRLF', 'Cell 1 of 11', 'Go Live', and the time '22:46 29-03-2024'.

```
File Edit Selection View Go Run ... backend
Assignment9_1.ipynb Assignment9_2.ipynb X
C:\> ALL > photos > ALL_FILES > CourseWork > Neural_Networks > Assignment9_2.ipynb > import pandas as pd
+ Code + Markdown | Run All | Clear All Outputs | Outline ... Select Kernel

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

[20] X_train, X_test, Y_train, Y_test = train_test_split(X,y, test_size = 0.33, random_state = 42) Python

[21] 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) Python

... 291/291 - 31s - loss: 0.8313 - accuracy: 0.6416
144/144 - 4s - loss: 0.7781 - accuracy: 0.6566
0.7780812978744507
0.656618595123291
['loss', 'accuracy']

print(X_train.shape, Y_train.shape)
```

```
File Edit Selection View Go Run ... backend
Assignment9_1.ipynb Assignment9_2.ipynb X
C:\> ALL > photos > ALL_FILES > CourseWork > Neural_Networks > Assignment9_2.ipynb > empty cell
+ Code + Markdown | Run All | Clear All Outputs | Outline ... Select Kernel

0.656618595123291
['loss', 'accuracy']

[22] print(X_train.shape,Y_train.shape)
print(X_test.shape,Y_test.shape)
model = KerasClassifier(build_fn=createmodel,verbose=0)
epochs = [1, 2]
param_grid= dict(epochs=epochs)
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=1)
grid_result= grid.fit(X_train, Y_train,batch_size=32)
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_)) Python

... (9293, 28) (9293, 3)
(4578, 28) (4578, 3)
Best: 0.674915 using {'epochs': 2}
```