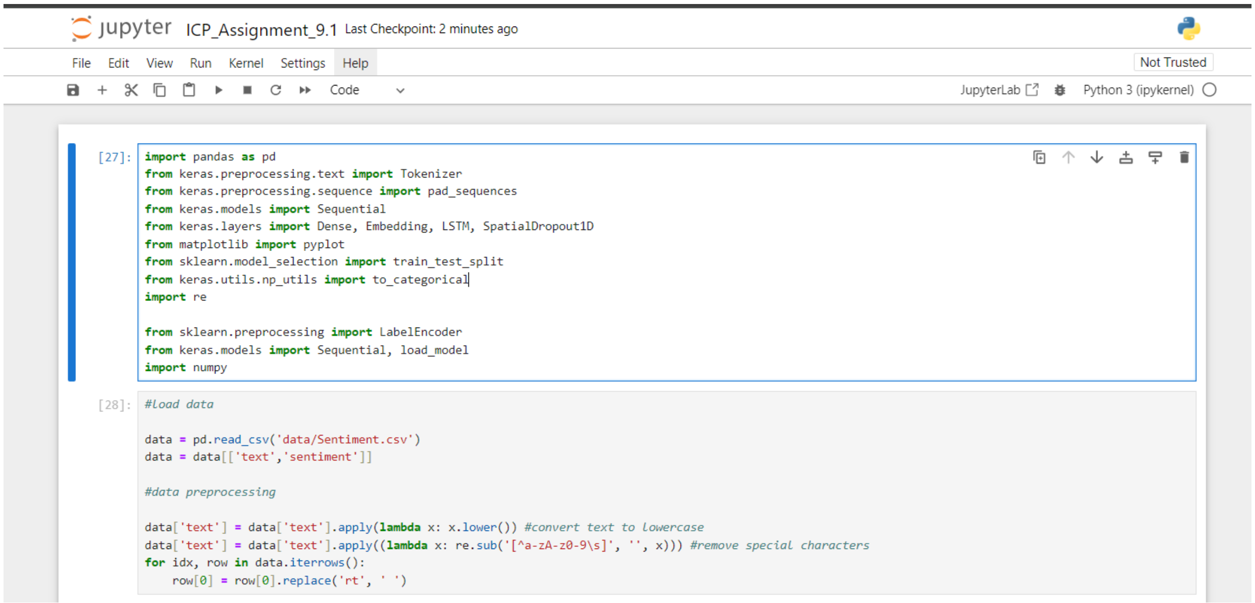
https://github.com/HemanthLakkimsetti76/NNDL_Assignment9

NNDL ASSIGNMENT 9

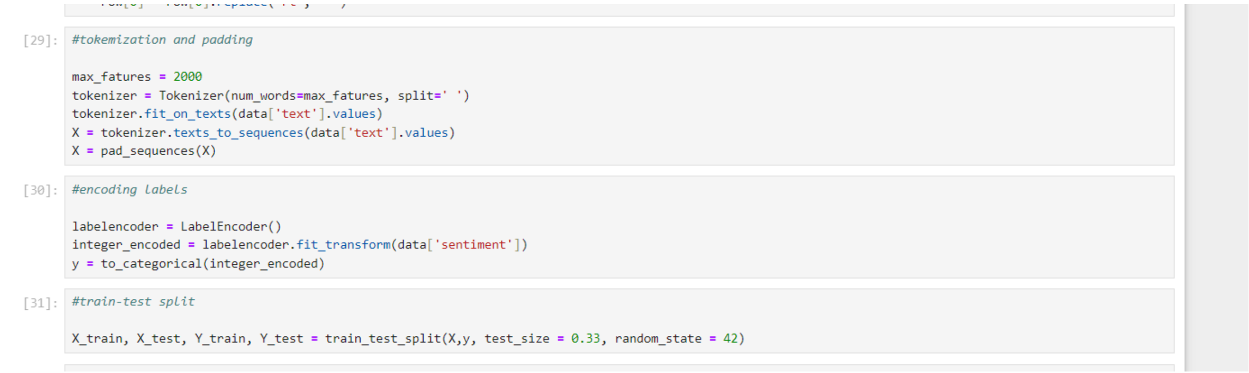
Hemanth Lakkimsetti – 700747872

Task 9.1:


jupyter
ICP_Assignment_9.1 Last Checkpoint: 2 minutes ago
File Edit View Run Kernel Settings Help
Not Trusted
Code
JupyterLab
Python 3 (ipykernel)

```
[27]: 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
```

```
[28]: #load data

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

      #data preprocessing

      data['text'] = data['text'].apply(lambda x: x.lower()) #convert text to lowercase
      data['text'] = data['text'].apply((lambda x: re.sub('[^a-zA-z0-9\s]', '', x))) #remove special characters
      for idx, row in data.iterrows():
          row[0] = row[0].replace('rt', ' ')
```

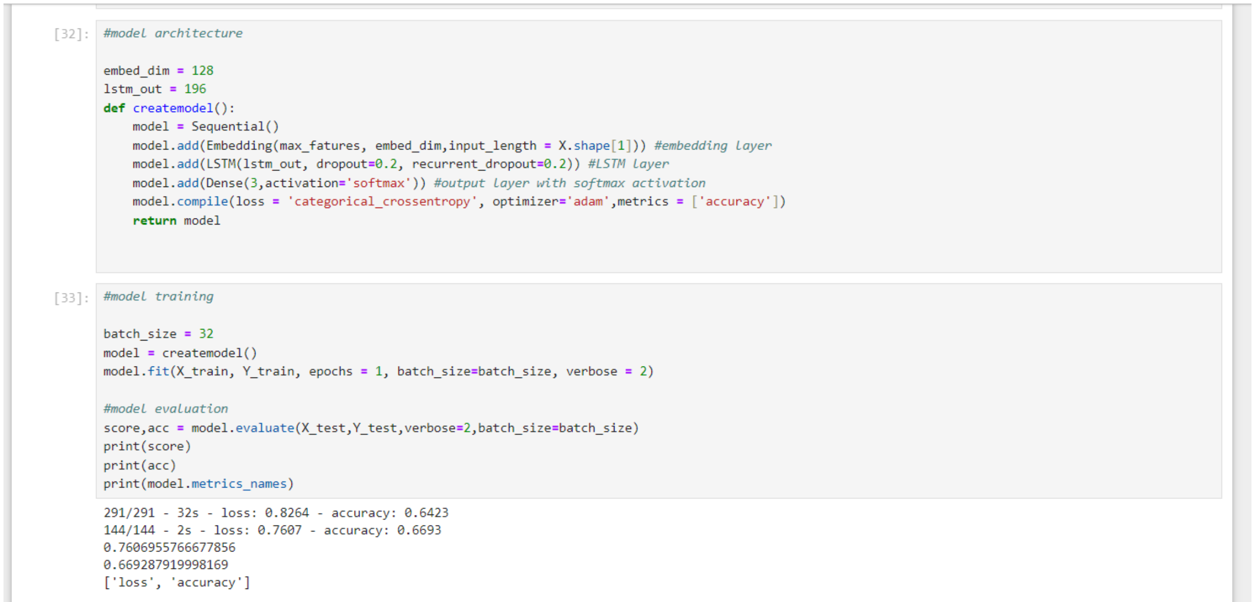


```
[29]: #tokemization and padding

      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)
```

```
[30]: #encoding labels

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

```
[31]: #train-test split

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

https://github.com/HemanthLakkimsetti76/NNDL_Assignment9



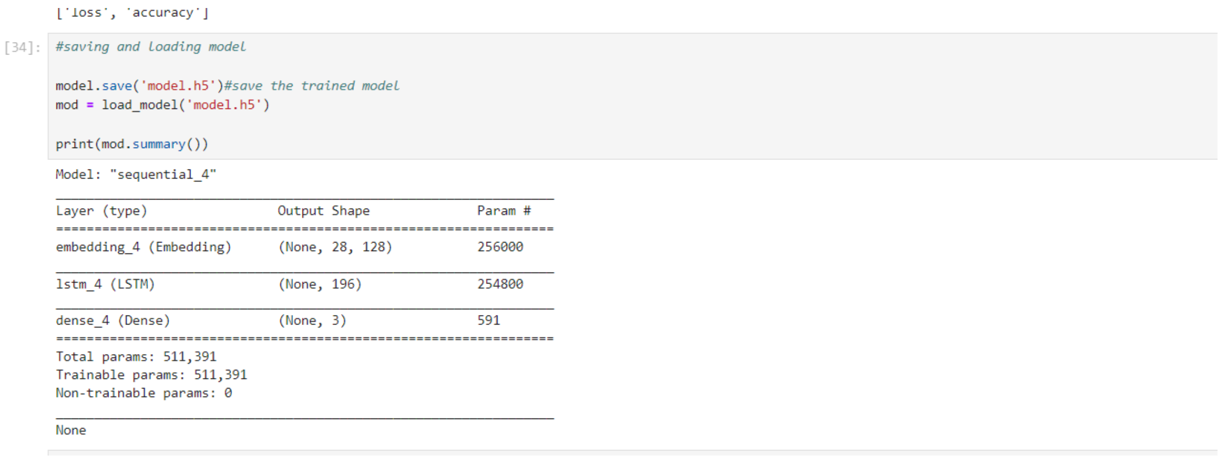
[32]:

```
#model architecture

embed_dim = 128
lstm_out = 196
def createmodel():
    model = Sequential()
    model.add(Embedding(max_fatures, embed_dim,input_length = X.shape[1])) #embedding layer
    model.add(LSTM(lstm_out, dropout=0.2, recurrent_dropout=0.2)) #LSTM layer
    model.add(Dense(3,activation='softmax')) #output layer with softmax activation
    model.compile(loss = 'categorical_crossentropy', optimizer='adam',metrics = ['accuracy'])
    return model
```

[33]:

```
#model training

batch_size = 32
model = createmodel()
model.fit(X_train, Y_train, epochs = 1, batch_size=batch_size, verbose = 2)

#model evaluation
score,acc = model.evaluate(X_test,Y_test,verbose=2,batch_size=batch_size)
print(score)
print(acc)
print(model.metrics_names)
```

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



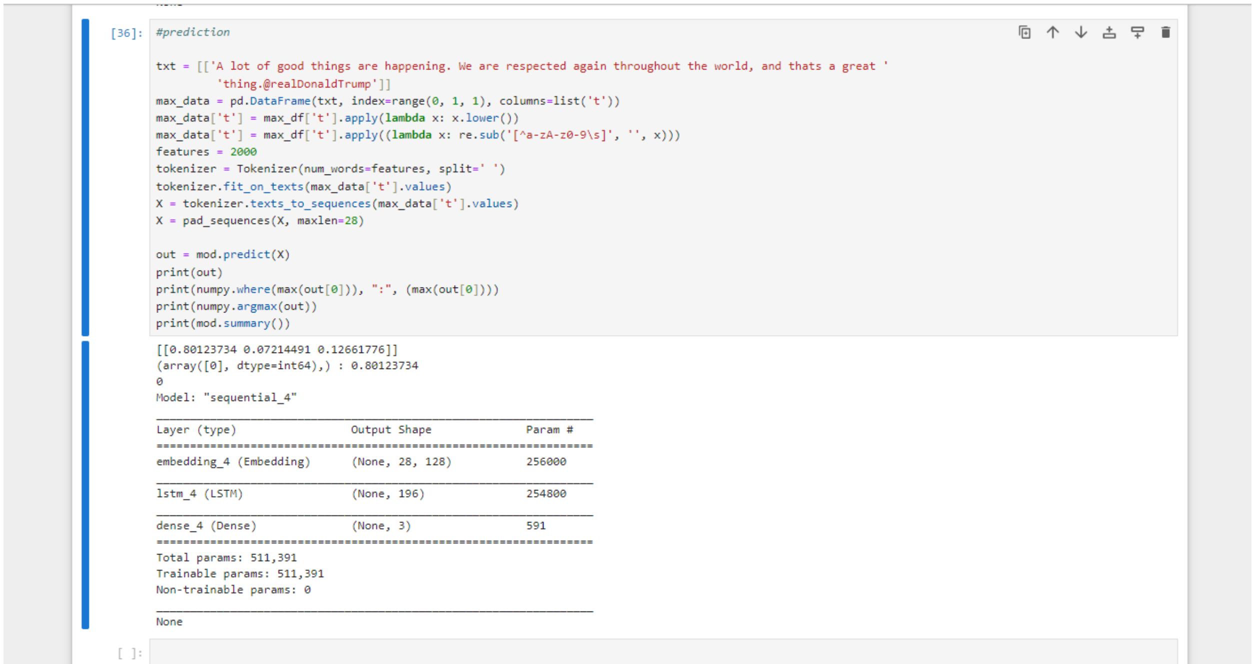
```
['loss', 'accuracy']
```

[34]:

```
#saving and loading model

model.save('model.h5')#save the trained model
mod = load_model('model.h5')

print(mod.summary())
```

```
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
```

https://github.com/HemanthLakkimsetti76/NNDL_Assignment9



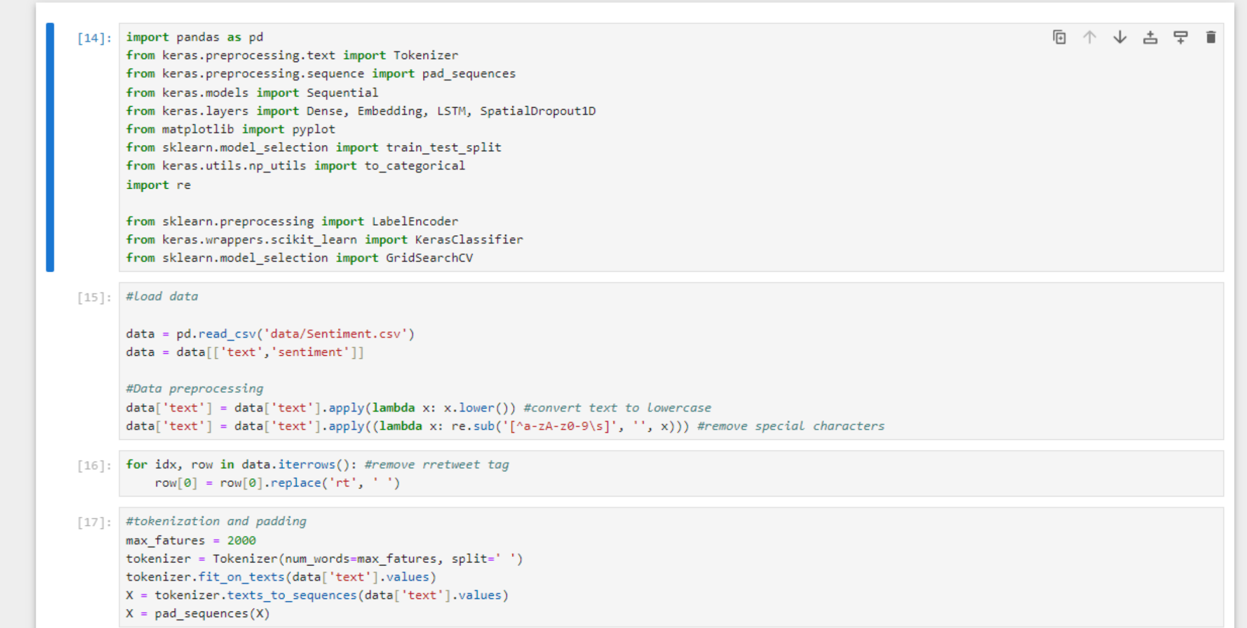
[36]:

```
#prediction

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_df['t'].apply(lambda x: x.lower())
max_data['t'] = max_df['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())
```

```
[[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
```

[ ]:

Task 9.2:



[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]:

```
#load data

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

#Data preprocessing
data['text'] = data['text'].apply(lambda x: x.lower()) #convert text to lowercase
data['text'] = data['text'].apply((lambda x: re.sub('[^a-zA-z0-9\s]', '', x))) #remove special characters
```

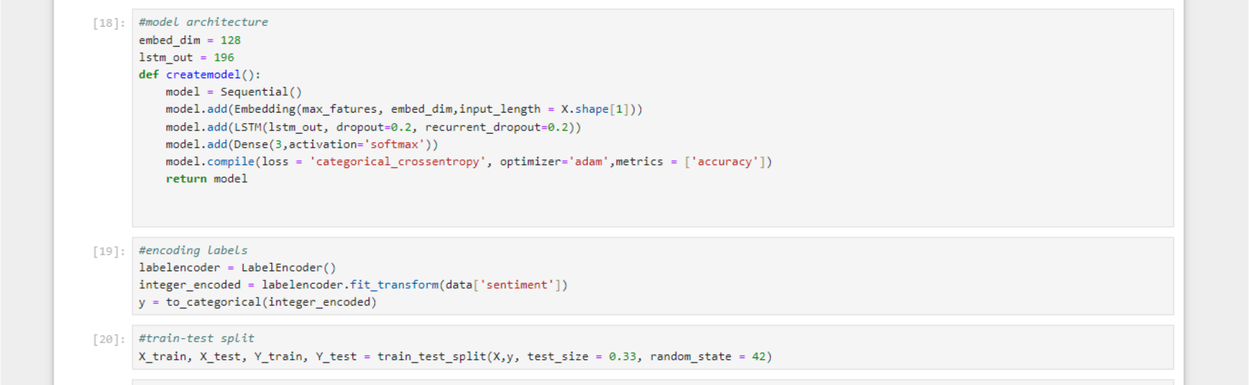
[16]:

```
for idx, row in data.iterrows(): #remove rretweet tag
    row[0] = row[0].replace('rt', ' ')
```

[17]:

```
#tokenization and padding
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)
```

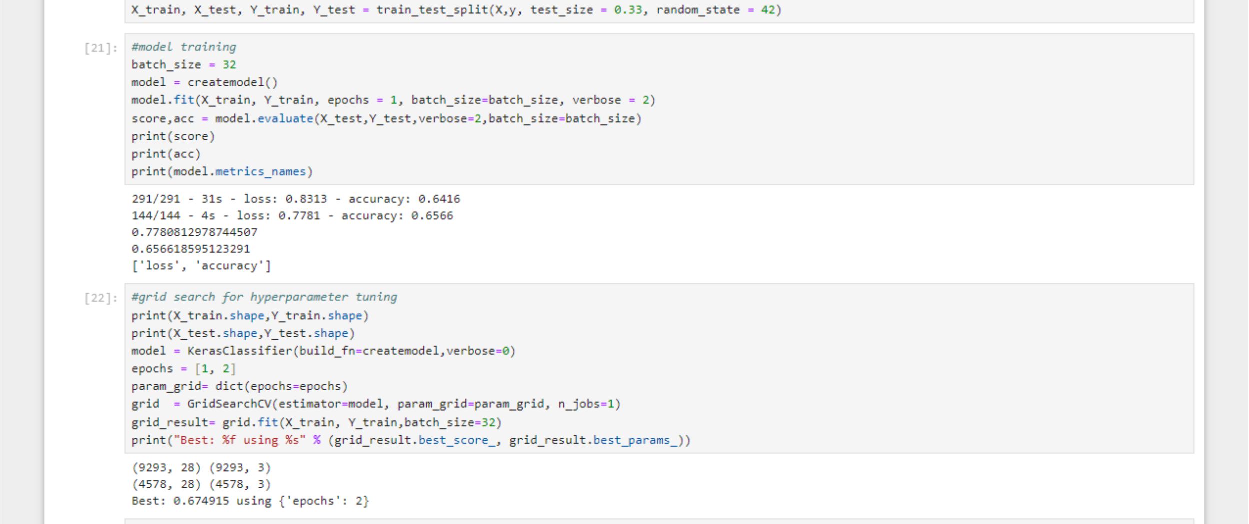
https://github.com/HemanthLakkimsetti76/NNDL_Assignment9



```
[18]: #model architecture
      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
```

```
[19]: #encoding labels
      labelencoder = LabelEncoder()
      integer_encoded = labelencoder.fit_transform(data['sentiment'])
      y = to_categorical(integer_encoded)
```

```
[20]: #train-test split
      X_train, X_test, Y_train, Y_test = train_test_split(X,y, test_size = 0.33, random_state = 42)
```



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

```
[21]: #model training
      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 - 31s - loss: 0.8313 - accuracy: 0.6416
144/144 - 4s - loss: 0.7781 - accuracy: 0.6566
0.7780812978744507
0.656618595123291
['loss', 'accuracy']
```

```
[22]: #grid search for hyperparameter tuning
      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_))
```

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