```
# Importing Libararies
import pandas as pd
import spacy
from tensorflow.keras.preprocessing.text import one_hot
from tensorflow.keras.preprocessing.sequence import pad_sequences
import numpy as np
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow import keras
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
```

```
df = pd.read_csv('labeled_data.csv')
```

```
df.shape
```

```
(24783, 7)
```

```
df.head()
```

	Unnamed: 0	count	hate_speech	offensive_language	neither	class	tweet
0	0	3	0	0	3	2	!!! RT @mayasolovely: As a woman you shouldn't...
1	1	3	0	3	0	1	!!!!! RT @mleew17: boy

Next steps: Generate code with df View recommended plots

```
df.columns
```

```
Index(['Unnamed: 0', 'count', 'hate_speech', 'offensive_language', 'neither',
       'class', 'tweet'],
      dtype='object')
```

```
# Deleting unwanted columns
df.drop(columns = ['Unnamed: 0','count','hate_speech','offensive_language','neither'], inplace = True)
```

```
df.head()
```

	class	tweet
0	2	!!! RT @mayasolovely: As a woman you shouldn't...
1	1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...
2	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...
3	1	!!!!!!!!! RT @C_G_Anderson: @viva_based she lo...
4	1	!!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...

Next steps: Generate code with df View recommended plots

```
# checking null values
df.isnull().sum()
```

```
class    0
tweet    0
dtype: int64
```

```
df['tweet'].iloc[0]
```

```
'!!! RT @mayasolovely: As a woman you shouldn't complain about cleaning up your hous
e. &amp; as a man you should always take the trash out...'
```

```
df['tweet'].iloc[100]
```

```
'"@ClicquotSuave: LMAOOOOOOOOOO this nigga @Krillz_Nuh_Care http://t.co/AAnpSUjmYI"
&lt:bitch want likes for some depressing shit..foh'
```

```
df['tweet'].iloc[1000]
```

```
'&#128514;&#128514;&#128514;&#128514;&#128514;&#128514;&#128514;"@betysweetcocker: T
hat pussy is just....&#128561; imma assume she just had a baby like..the day befor
e"'
```

```
# Deleting unwanted symbols and numeric data
df['processed_tweet']=df['tweet'].str.replace(r'[^a-zA-Z]',' ', regex=True)
```

```
df.head()
```

	class	tweet	processed_tweet
0	2	!!! RT @mayasolovely: As a woman you shouldn't...	RT mayasolovely As a woman you shouldn t...
1	1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...	RT mleew boy dats cold tyga dwn ba...
2	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...	RT UrKindOfBrand Dawg RT sbaby...
3	1	!!!!!!!!! RT @C_G_Anderson:	RT C G Anderson viva based she lo

Next steps: Generate code with df View recommended plots

```
# Handling unwanted spaces
df['processed_tweet_2']=df['processed_tweet'].str.replace(r'[\s]+',' ',regex = True)
```

```
df.head()
```

	class	tweet	processed_tweet	processed_tweet_2
0	2	!!! RT @mayasolovely: As a woman you shouldn't...	RT mayasolovely As a woman you shouldn t...	RT mayasolovely As a woman you shouldn t comp...
1	1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...	RT mleew boy dats cold tyga dwn ba...	RT mleew boy dats cold tyga dwn bad for cuffi...
2	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby	RT UrKindOfBrand Dawg RT sbaby...	RT UrKindOfBrand Dawg RT sbaby life You ever ...

Next steps: Generate code with df View recommended plots

```
df['processed_tweet_2'].iloc[1000]
```

```
' betysweetcocker That pussy is just imma assume she just had a baby like the day be
fore '
```

```
df.drop(columns = ['tweet','processed_tweet'], inplace = True)
```

```
df.head()
```

	class	processed_tweet_2
0	2	RT mayasolovely As a woman you shouldn t comp...
1	1	RT mleew boy dats cold tyga dwn bad for cuffi...
2	1	RT UrKindOfBrand Dawg RT sbaby life You ever ...
3	1	RT C G Anderson viva based she look like a tr...
4	1	RT ShenikaRoberts The shit you hear about me ...

Next steps: Generate code with df View recommended plots

```
# NLP
nlp = spacy.load('en_core_web_sm')
```

```
# lemmatization
def lemmatization(text):
  doc = nlp(text)
  lemmalist = [word.lemma_ for word in doc]
  return ' '.join(lemmalist)
```

```
df['lemma_tweet']=df['processed_tweet_2'].apply(lemmatization)
```

```
df.head()
```

	class	processed_tweet_2	lemma_tweet
0	2	RT mayasolovely As a woman you shouldn t comp...	RT mayasolovely as a woman you shouldn t com...
1	1	RT mleew boy dats cold tyga dwn bad for cuffi...	RT mleew boy dat cold tyga dwn bad for cuffi...
2	1	RT UrKindOfBrand Dawg RT sbaby life You ever ...	RT UrKindOfBrand Dawg RT sbaby life you ever...
3	1	RT C G Anderson viva based she look	RT C G Anderson viva base she look

Next steps: Generate code with df View recommended plots

```
# Removing the stopwords
def remove_stopwords(text):
  doc = nlp(text)
  no_stopwords_list = [word.text for word in doc if not word.is_stop]
  return ' '.join(no_stopwords_list)
```

```
df['final_tweet']=df['lemma_tweet'].apply(remove_stopwords)
```

```
df.head()
```

	class	processed_tweet_2	lemma_tweet	final_tweet
0	2	RT mayasolovely As a woman you shouldn t comp...	RT mayasolovely as a woman you shouldn t com...	RT mayasolovely woman shouldn t complain cl...
1	1	RT mleew boy dats cold tyga dwn bad for cuffi...	RT mleew boy dat cold tyga dwn bad for cuffi...	RT mleew boy dat cold tyga dwn bad cuffin d...
2	1	RT UrKindOfBrand Dawg RT sbaby life You ever ...	RT UrKindOfBrand Dawg RT sbaby life you ever	RT UrKindOfBrand Dawg RT sbaby life fuck bi

Next steps: Generate code with df View recommended plots

```
df.drop(columns = ['processed_tweet_2', 'lemma_tweet'], inplace = True)
```

```
df.head()
```

	class	final_tweet
0	2	RT mayasolovely woman shouldn t complain cl...
1	1	RT mleew boy dat cold tyga dwn bad cuffin d...
2	1	RT UrKindOfBrand Dawg RT sbaby life fuck bi...
3	1	RT C G Anderson viva base look like tranny
4	1	RT ShenikaRoberts shit hear true faker bitc...

Next steps: Generate code with df View recommended plots

```
# One - hot representation
vocab_size = 10000
one_hot_representation = [one_hot(words, vocab_size) for words in df['final_tweet']]
```

```
df['final_tweet'].iloc[0]
```

```
'  RT mayasolovely woman shouldn t complain clean house amp man trash'
```

```
one_hot_representation[0]
```

```
[490, 3873, 6677, 695, 9577, 1883, 4726, 6487, 1229, 4508, 205]
```

```
for i in range(0, 4):
  print(df['final_tweet'].iloc[i])
```

```
  RT mayasolovely woman shouldn t complain clean house amp man trash
  RT mleew boy dat cold tyga dwn bad cuffin dat hoe st place
  RT UrKindOfBrand Dawg RT sbaby life fuck bitch start cry confuse shit
  RT C G Anderson viva base look like tranny
```

```
for i in range(0, 4):
```

```
for i in range(0, 4):
  print(one_hot_representation[i])
```

```
[490, 3873, 6677, 695, 9577, 1883, 4726, 6487, 1229, 4508, 205]
[490, 8983, 8232, 2014, 4994, 7007, 8979, 1846, 8838, 2014, 1425, 2590, 5243]
[490, 7645, 5540, 490, 3233, 6634, 6861, 8840, 2326, 9388, 3207, 6512]
[490, 659, 8254, 447, 4676, 3590, 2345, 9773, 5254]
```

```
sentence_length = 20
embedded_tweet = pad_sequences(one_hot_representation, padding = 'pre', maxlen = sentence_length)
```

```
for i in range(0, 4):
  print(embedded_tweet[i])
```

```
[   0    0    0    0    0    0    0    0    0  490 3873 6677  695 9577
 1883 4726 6487 1229 4508  205]
[   0    0    0    0    0    0    0  490 8983 8232 2014 4994 7007 8979
 1846 8838 2014 1425 2590 5243]
[   0    0    0    0    0    0    0    0  490 7645 5540  490 3233 6634
 6861 8840 2326 9388 3207 6512]
[   0    0    0    0    0    0    0    0    0    0    0  490  659 8254
  447 4676 3590 2345 9773 5254]
```

```
X = np.array(embedded_tweet)
y = np.array(df['class'])
```

```
df['class'].value_counts()
```

```
class
1    19190
2     4163
0     1430
Name: count, dtype: int64
```

```
smote = SMOTE(sampling_strategy='minority')
X,y = smote.fit_resample(X,y)
```

```
# Train Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

```
X.shape, X_train.shape, X_test.shape
```

```
((42543, 20), (34034, 20), (8509, 20))
```

```
# Creating Model

dimension = 50

model = keras.Sequential([
    # embedding layer
    keras.layers.Embedding(vocab_size, dimension, input_length = sentence_length),
    # LSTM layers (stacked)
    keras.layers.LSTM(100, return_sequences = True),
    keras.layers.LSTM(50, return_sequences = True),
    keras.layers.LSTM(50),
    # Output Layer
    keras.layers.Dense(3, activation = 'softmax')
])

model.compile(optimizer = 'adam',
              loss = 'sparse_categorical_crossentropy',
              metrics = ['accuracy'])
```

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding_1 (Embedding)     (None, 20, 50)            500000    
                                                                 
 lstm_3 (LSTM)               (None, 20, 100)           60400     
                                                                 
 lstm_4 (LSTM)               (None, 20, 50)            30200     
                                                                 
 lstm_5 (LSTM)               (None, 50)                20200     
                                                                 
 dense_1 (Dense)             (None, 3)                 153       
```

```
=================================================================
Total params: 610953 (2.33 MB)
Trainable params: 610953 (2.33 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
model.fit(X_train, y_train, epochs = 10, batch_size = 32)
```

```
Epoch 1/10
1064/1064 [==============================] - 39s 28ms/step - loss: 0.3413 - accuracy: 0.8703
Epoch 2/10
1064/1064 [==============================] - 13s 12ms/step - loss: 0.1679 - accuracy: 0.9456
Epoch 3/10
1064/1064 [==============================] - 11s 10ms/step - loss: 0.1220 - accuracy: 0.9618
Epoch 4/10
1064/1064 [==============================] - 11s 10ms/step - loss: 0.0948 - accuracy: 0.9713
Epoch 5/10
1064/1064 [==============================] - 10s 9ms/step - loss: 0.0744 - accuracy: 0.9778
Epoch 6/10
1064/1064 [==============================] - 9s 8ms/step - loss: 0.0572 - accuracy: 0.9824
Epoch 7/10
1064/1064 [==============================] - 10s 10ms/step - loss: 0.0429 - accuracy: 0.9859
Epoch 8/10
1064/1064 [==============================] - 14s 13ms/step - loss: 0.0299 - accuracy: 0.9906
Epoch 9/10
1064/1064 [==============================] - 16s 15ms/step - loss: 0.0236 - accuracy: 0.9927
Epoch 10/10
1064/1064 [==============================] - 15s 14ms/step - loss: 0.0190 - accuracy: 0.9942
<keras.src.callbacks.History at 0x79e800188f70>
```

```
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Model Accuracy : {accuracy * 100}')
```

```
266/266 [==============================] - 3s 4ms/step - loss: 0.5636 - accuracy: 0.8951
Model Accuracy : 89.50523138046265
```

```
pred = np.argmax(model.predict(X_test), axis = -1)
```

```
266/266 [==============================] - 2s 8ms/step
```

```
y_test[:5]
```

```
array([1, 1, 0, 1, 2])
```

```
pred[:5]
```

```
array([1, 1, 0, 1, 2])
```

```
print(classification_report(y_test, pred))
```

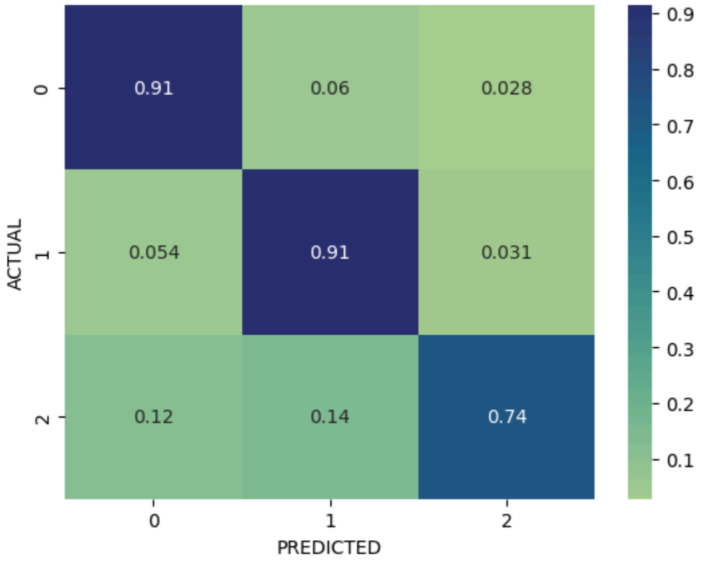
```
              precision    recall  f1-score   support

           0       0.92      0.91      0.91      3812
           1       0.91      0.91      0.91      3807
           2       0.75      0.74      0.74       890

    accuracy                           0.90      8509
   macro avg       0.86      0.86      0.86      8509
weighted avg       0.89      0.90      0.89      8509
```

Suggested code may be subject to a license | hjs90911/1804_Learning_MachineLearning

```
cf = confusion_matrix(y_test, pred, normalize = 'true')
sns.heatmap(cf, annot = True, cmap = 'crest')
plt.xlabel('PREDICTED')
plt.ylabel('ACTUAL')
plt.show()
```



Start coding or generate with AI.