In [1]:

```
# This is a test for my final project coding algorithm
```

In [123…

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import cross_val_score
from sklearn import preprocessing
from sklearn.metrics import accuracy_score
from sklearn.feature_extraction import DictVectorizer
from sklearn.metrics import classification_report

import tensorflow.compat.v2 as tf

from tensorflow.python.platform import tf_logging as logging

from keras.models import Sequential
from keras import layers
```

In [ ]:

```

```

In [180…

```
## let's get a list together

# preprocessing
# Each sentence is put in to training data tuples. with the sentence and label


train_data = []
tags = []
with open('imdb_labelled.txt', 'r') as f:
    train_content = f.read()  # Raw Data without separation
    lines = train_content.split('\n')  # separates rows
    for line in lines:
        if line != "":
            tagged_words = line.split('\t')
            tup = (tagged_words[0],tagged_words[1])
            train_data.append(tup)

with open('amazon_cells_labelled.txt', 'r') as f:
    train_content = f.read()  # Raw Data without separation
    lines = train_content.split('\n')  # separates rows
    for line in lines:
        if line != "":
            tagged_words = line.split('\t')
            tup = (tagged_words[0],tagged_words[1])
            train_data.append(tup)

with open('yelp_labelled.txt', 'r') as f:
    train_content = f.read()  # Raw Data without separation
    lines = train_content.split('\n')  # separates rows
    for line in lines:
        if line != "":
            tagged_words = line.split('\t')
            tup = (tagged_words[0],tagged_words[1])
            train_data.append(tup)

#adds all 3 into training data!
```

In [181…

```
# train test split
X = []
for i in range(0, 3000):
    X.append(train_data[i][0])
y = []
for i in range(0, 3000):
    y.append(train_data[i][1])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=
```

In [182…

```
## import stuff for word embeddings
from keras.preprocessing.text import Tokenizer
```

In [183…

```
tk = Tokenizer(num_words=5000)
```

In [184…

```
tk.fit_on_texts(X_train)
```

In [185…

```
X2_train = tk.texts_to_sequences(X_train)
X_test = tk.texts_to_sequences(X_test)
```

In [186…

```
v_size = len(tk.word_index) + 1
```

In [187…

```
print(X2_train[0])
print(X_train[0])
```

```
[354, 355, 3, 388, 927, 11, 1, 30, 2, 49, 24, 1261, 8, 1262, 928]
There's barely a boring moment in the film and there are plenty of humorous parts.
```

In [188…

```
from keras_preprocessing.sequence import pad_sequences
X2_train = pad_sequences(X2_train, padding='post', maxlen=100)
X_test = pad_sequences(X_test, padding='post', maxlen=100)
```

In [189…

```
em_dim = 100
```

In [190…

```
md = Sequential()
```

In [191…

```
md.add(layers.Embedding(v_size, em_dim, input_length=100)) #Build layers of model
md.add(layers.Conv1D(128, 5, activation='relu'))
md.add(layers.GlobalAveragePooling1D())
md.add(layers.Dense(10, activation='relu'))
md.add(layers.Dense(1, activation='sigmoid'))
md.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

In [192…

```
# we have to edit y train and test to be proper
y2_train = []
y2_test = []
for i in range(0, 2400):
    if (y_train[i] == '1'):
        y2_train.append(1)
    else:
        y2_train.append(0)
```

```
for i in range(0, 600):
    if (y_test[i] == '1'):
        y2_test.append(1)
    else:
        y2_test.append(0)
```

In [193…

```
X2_train = np.asarray(X2_train)
y2_train = np.asarray(y2_train)
X_test = np.asarray(X_test)
y2_test = np.asarray(y2_test)
```

In [ ]:

```
fit = md.fit(X2_train, y2_train, epochs = 14, verbose = False, validation_data=(X_test
```

In [165…

```
md.summary()
```

```
Model: "sequential_12"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding_13 (Embedding)    (None, 100, 100)          471700    
                                                                 
 conv1d_5 (Conv1D)           (None, 96, 128)           64128     
                                                                 
 global_average_pooling1d_2   (None, 128)              0         
 (GlobalAveragePooling1D)                                        
                                                                 
 dense_26 (Dense)            (None, 10)                1290      
                                                                 
 dense_27 (Dense)            (None, 1)                 11        
                                                                 
=================================================================
Total params: 537,129
Trainable params: 537,129
Non-trainable params: 0
_________________________________________________________________
```

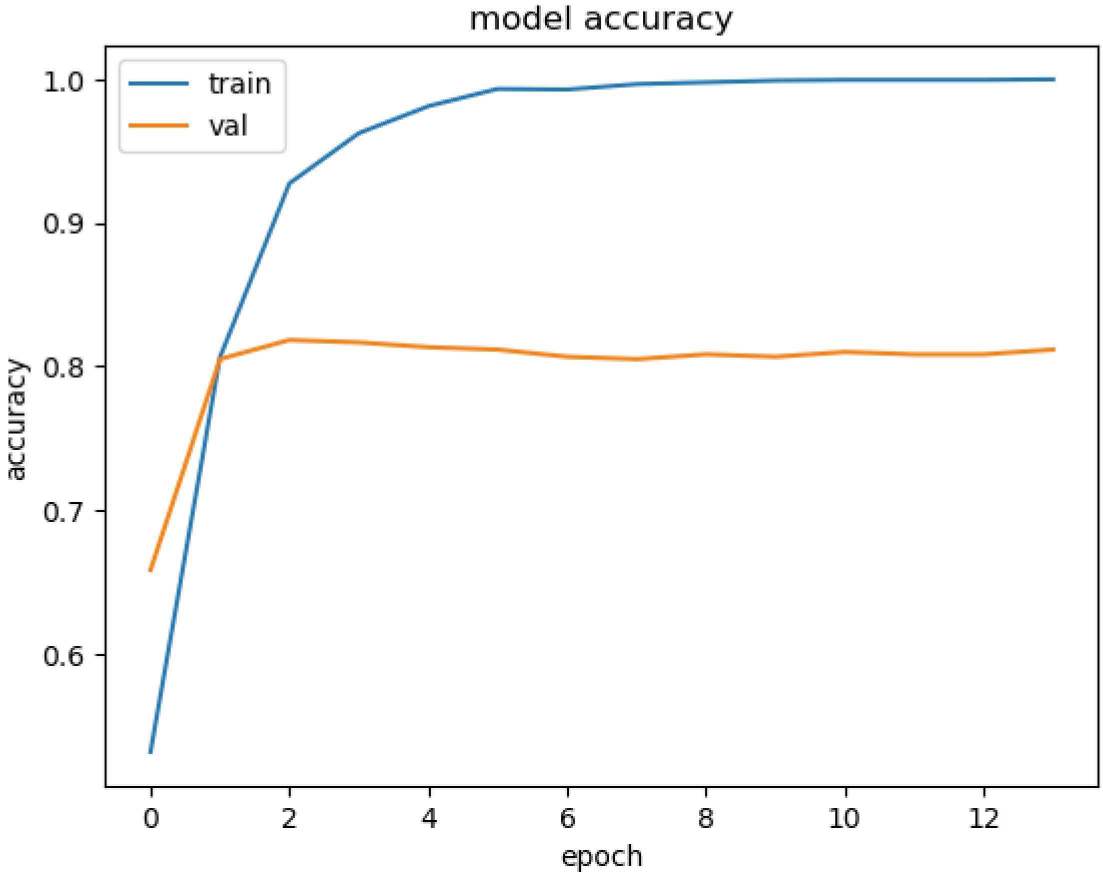
In [166…

```
loss, accuracy = md.evaluate(X2_train, y2_train, verbose=False)
print("Training Accuracy: {:.4f}".format(accuracy))
loss, accuracy = md.evaluate(X_test, y2_test, verbose=False)
print("Testing Accuracy:  {:.4f}".format(accuracy))
```

```
Training Accuracy: 1.0000
Testing Accuracy:  0.8117
```

In [167…

```
plt.plot(fit.history['accuracy'])
plt.plot(fit.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```


model accuracy
train
val
accuracy
1.0
0.9
0.8
0.7
0.6
epoch
0
2
4
6
8
10
12