In [1]:
```
cd G:/
```

```
G:\
```

In [2]:
```
cd smart_bridge
```

```
G:\smart_bridge
```

In [3]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

In [4]:
```
df = pd.read_csv("SENTIMENTAL ANALYSIS BASED ON FLIGHT REVIEWS.csv")
df.head()
```

Out[4]:

	airline_sentiment	airline	text
0	positive	Virgin America	@VirginAmerica What @dhepburn said.
1	positive	Virgin America	@VirginAmerica plus you've added commercials t...
2	positive	Virgin America	@VirginAmerica I didn't today... Must mean I n...
3	negative	Virgin America	@VirginAmerica it's really aggressive to blast...
4	negative	Virgin America	@VirginAmerica and it's a really big bad thing...

In [5]:
```
df.airline_sentiment = df.airline_sentiment.map({"positive":1, "negative":0})

df.airline = df.airline.map({"Virgin America":2, "United":1})
```

In [6]:
```
from nltk.stem.snowball import SnowballStemmer
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
import re
import nltk
nltk.download('stopwords')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out[6]:
```
True
```

In [7]:
```
my_stopwords = ["is", "am","an", "are", "of","can", "were", "he", "she", "him",
                "also", "to", "for", "the", "we", "you", "by", "during", "on", "
                "that", "would", "this", "there", "youd", "who","then", "they",
def clean_review(review):
    review=re.sub("(n\'t)|(\w+nt)"," not",review)
    review=re.sub("(\@\w+)","",review)
    review=re.sub("[^a-zA-Z]"," ",review)
    review=re.sub("\s\s"," ",review)

    review=review.lower()
    review=review.split()
    ps=PorterStemmer()
    lemma = nltk.wordnet.WordNetLemmatizer()

    review=[ps.stem(lemma.lemmatize(word)) for word in review if ((word not in my
    review = [word for word in review if len(word) > 1]
    review=' '.join(review)
    return review
```

In [8]:
```
df["processed_review"] = df.text.apply(clean_review)
df.head()
```

Out[8]:

	airline_sentiment	airline	text	processed_review
0	1	2	@VirginAmerica What @dhepburn said.	what said
1	1	2	@VirginAmerica plus you've added commercials t...	plu ve ad commerci experi tacki
2	1	2	@VirginAmerica I didn't today... Must mean I n...	did not today must mean need take anoth trip
3	0	2	@VirginAmerica it's really aggressive to blast...	realli aggress blast obnoxi not your guest fac...
4	0	2	@VirginAmerica and it's a really big bad thing...	and realli big bad thing about

In [9]:
```
X = df.processed_review
y = df.airline_sentiment
```

In [10]:
```
df.processed_review.sample(5).tolist()
```

Out[10]:
```
['worst nonrefund first class ticket oh becaus when select global fc their syst
em auto select economi upgrad',
 'did start claim but week unrealist realli suppos go long with out car seat ch
ild ridicul',
 'just promot product all problem with southwest and recommend noneoth than bes
t http co tfanxbh cf',
 'fli from love austin now most not news',
 'realli aggress blast obnoxi not your guest face amp littl recours']
```

In [11]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25, shuffle=True,
```

In [12]:
```
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer

cv=TfidfVectorizer(max_features=1200)

X_train=cv.fit_transform(X_train)
X_test=cv.transform(X_test)
```

In [13]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier

from keras.layers import InputLayer, Dense, Activation, Dropout
from keras.models import Sequential

model = Sequential()
model.add(Dense(1200, input_shape=X_train.shape[1:], activation="sigmoid"))

model.add(Dense(1,activation="sigmoid"))
model.compile(optimizer="adam", loss="binary_crossentropy", metrics=["accuracy"])
model.fit(X_train, y_train, epochs=25)
```

```
Using TensorFlow backend.

WARNING:tensorflow:From C:\Users\Admin\Anaconda3\lib\site-packages\keras\back
end\tensorflow_backend.py:74: The name tf.get_default_graph is deprecated. Pl
ease use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From C:\Users\Admin\Anaconda3\lib\site-packages\keras\back
end\tensorflow_backend.py:517: The name tf.placeholder is deprecated. Please
use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From C:\Users\Admin\Anaconda3\lib\site-packages\keras\back
end\tensorflow_backend.py:4138: The name tf.random_uniform is deprecated. Ple
ase use tf.random.uniform instead.

WARNING:tensorflow:From C:\Users\Admin\Anaconda3\lib\site-packages\keras\opti
mizers.py:790: The name tf.train.Optimizer is deprecated. Please use tf.compa
t.v1.train.Optimizer instead.

WARNING:tensorflow:From C:\Users\Admin\Anaconda3\lib\site-packages\keras\back
end\tensorflow_backend.py:3376: The name tf.log is deprecated. Please use tf.
math.log instead.

WARNING:tensorflow:From C:\Users\Admin\Anaconda3\lib\site-packages\tensorflow
\python\ops\nn_impl.py:180: add_dispatch_support.<locals>.wrapper (from tenso
rflow.python.ops.array_ops) is deprecated and will be removed in a future ver
sion.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From C:\Users\Admin\Anaconda3\lib\site-packages\keras\back
end\tensorflow_backend.py:986: The name tf.assign_add is deprecated. Please u
se tf.compat.v1.assign_add instead.

Epoch 1/25
385/385 [==============================] - 1s 3ms/step - loss: 0.6788 - acc:
0.5818
Epoch 2/25
385/385 [==============================] - 1s 2ms/step - loss: 0.6458 - acc:
0.6286
Epoch 3/25
385/385 [==============================] - 1s 2ms/step - loss: 0.6542 - acc:
0.5584
Epoch 4/25
```

```
385/385 [==============================] - 1s 2ms/step - loss: 0.5904 - acc:
0.6494
Epoch 5/25
385/385 [==============================] - 1s 2ms/step - loss: 0.5213 - acc:
0.7403
Epoch 6/25
385/385 [==============================] - 1s 2ms/step - loss: 0.5095 - acc:
0.9117
Epoch 7/25
385/385 [==============================] - 1s 2ms/step - loss: 0.5107 - acc:
0.6727
Epoch 8/25
385/385 [==============================] - 1s 2ms/step - loss: 0.4792 - acc:
0.8519
Epoch 9/25
385/385 [==============================] - 1s 2ms/step - loss: 0.4176 - acc:
0.7974
Epoch 10/25
385/385 [==============================] - 1s 2ms/step - loss: 0.3851 - acc:
0.9273
Epoch 11/25
385/385 [==============================] - 1s 2ms/step - loss: 0.3615 - acc:
0.9481
Epoch 12/25
385/385 [==============================] - 1s 2ms/step - loss: 0.3540 - acc:
0.9636
Epoch 13/25
385/385 [==============================] - 1s 2ms/step - loss: 0.3224 - acc:
0.9844
Epoch 14/25
385/385 [==============================] - 1s 2ms/step - loss: 0.3042 - acc:
0.9506A: 0s - loss: 0.3246 - a
Epoch 15/25
385/385 [==============================] - 1s 2ms/step - loss: 0.3098 - acc:
0.9714
Epoch 16/25
385/385 [==============================] - 1s 2ms/step - loss: 0.2731 - acc:
0.9610
Epoch 17/25
385/385 [==============================] - 1s 2ms/step - loss: 0.2496 - acc:
0.9792
Epoch 18/25
385/385 [==============================] - 1s 2ms/step - loss: 0.2449 - acc:
0.9558
Epoch 19/25
385/385 [==============================] - 1s 2ms/step - loss: 0.2282 - acc:
0.9844
Epoch 20/25
385/385 [==============================] - 1s 2ms/step - loss: 0.2131 - acc:
0.9740
Epoch 21/25
385/385 [==============================] - 1s 2ms/step - loss: 0.2189 - acc:
0.9766
Epoch 22/25
385/385 [==============================] - 1s 2ms/step - loss: 0.1981 - acc:
0.9688
Epoch 23/25
```

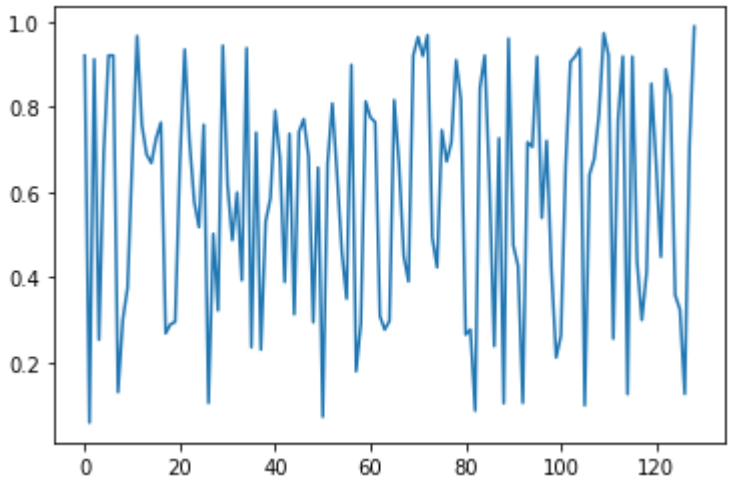
```
385/385 [==============================] - 1s 2ms/step - loss: 0.1867 - acc:
0.9766
Epoch 24/25
385/385 [==============================] - 1s 2ms/step - loss: 0.1593 - acc:
0.9948A: 0s - loss: 0.1592 - acc: 0.994
Epoch 25/25
385/385 [==============================] - 1s 2ms/step - loss: 0.1490 - acc:
0.9948
```

Out[13]: <keras.callbacks.History at 0x28b9c3481c8>

In [14]:
```
y_pred=model.predict(X_test)
```

In [15]:
```
plt.plot(y_pred)
```

Out[15]: [<matplotlib.lines.Line2D at 0x28b9c774848>]



In [16]:
```
y_pred_activated = y_pred >= 0.5
```

In [17]:
```
y_pred_activated.shape
```

Out[17]: (129, 1)

In [18]:
```
y_test.shape
```

Out[18]: (129,)

In [19]:
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred_activated)
cm
```

Out[19]:
```
array([[31, 12],
       [19, 67]], dtype=int64)
```

In [20]:
```
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
print(classification_report(y_test,y_pred_activated))
```

```
              precision    recall  f1-score   support

           0       0.62      0.72      0.67        43
           1       0.85      0.78      0.81        86

    accuracy                           0.76       129
   macro avg       0.73      0.75      0.74       129
weighted avg       0.77      0.76      0.76       129
```

In [21]:
```
print(accuracy_score(y_test,y_pred_activated))
```

```
0.7596899224806202
```

In [ ]:
```

```