**Topic: Sentiment Analysis**

**Aim:** Write a code for Sentiment Analysis.

**Introduction:**

Sentiment Analysis is the process of classifying whether a block of text is positive, negative, or, neutral. Sentiment analysis is contextual mining of words which indicates the social sentiment of a brand and also helps the business to determine whether the product which they are manufacturing is going to make a demand in the market or not.

**Source code:**

# ## Sentiment Analysis on US Airline Reviews

# In[1]:

import pandas as pd

import matplotlib.pyplot as plt

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad\_sequences

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM,Dense, Dropout, SpatialDropout1D

from tensorflow.keras.layers import Embedding

df = pd.read\_csv("Desktop/Tweets.csv")

# In[2]:

df.head()

# In[23]:

df.columns

# In[4]:

tweet\_df = df[['text','airline\_sentiment']]

print(tweet\_df.shape)

tweet\_df.head(5)

# In[22]

tweet\_df = tweet\_df[tweet\_df['airline\_sentiment'] != 'neutral']

print(tweet\_df.shape)

tweet\_df.head(5)

# In[21]:

tweet\_df["airline\_sentiment"].value\_counts()

# In[6]:

sentiment\_label = tweet\_df.airline\_sentiment.factorize()

sentiment\_label

# In[7]:

tweet = tweet\_df.text.values

tokenizer = Tokenizer(num\_words=5000)

tokenizer.fit\_on\_texts(tweet)

vocab\_size = len(tokenizer.word\_index) + 1

encoded\_docs = tokenizer.texts\_to\_sequences(tweet)

padded\_sequence = pad\_sequences(encoded\_docs, maxlen=200)

# In[8]:

print(tokenizer.word\_index)

# In[9]:

print(tweet[0])

print(encoded\_docs[0])

# In[10]:

print(padded\_sequence[0])

# In[11]:

embedding\_vector\_length = 32

model = Sequential()

model.add(Embedding(vocab\_size, embedding\_vector\_length, input\_length=200) )

model.add(SpatialDropout1D(0.25))

model.add(LSTM(50, dropout=0.5, recurrent\_dropout=0.5))

model.add(Dropout(0.2))

model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary\_crossentropy',optimizer='adam', metrics=['accuracy'])

print(model.summary())

# In[12]:

history = model.fit(padded\_sequence,sentiment\_label[0],validation\_split=0.2, epochs=5, batch\_size=32)

# In[16]:

plt.plot(history.history['accuracy'], label='acc')

plt.plot(history.history['val\_accuracy'], label='val\_acc')

plt.legend()

plt.show()

plt.savefig("Accuracy plot.jpg")

# In[25]:

plt.plot(history.history['loss'], label='loss')

plt.plot(history.history['val\_loss'], label='val\_loss')

plt.legend()

plt.show()

plt.savefig("Loss plot.jpg")

# In[18]:

def predict\_sentiment(text):

tw = tokenizer.texts\_to\_sequences([text])

tw = pad\_sequences(tw,maxlen=200)

prediction = int(model.predict(tw).round().item())

print("Predicted label: ", sentiment\_label[1][prediction])

# In[19]:

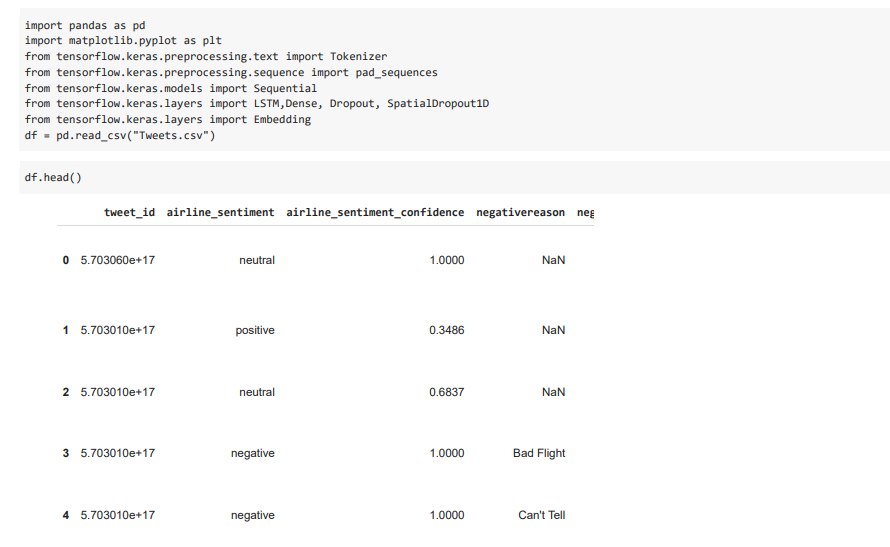
test\_sentence1 = "I enjoyed my journey on this flight."

predict\_sentiment(test\_sentence1)

test\_sentence2 = "This is the worst flight experience of my life!"

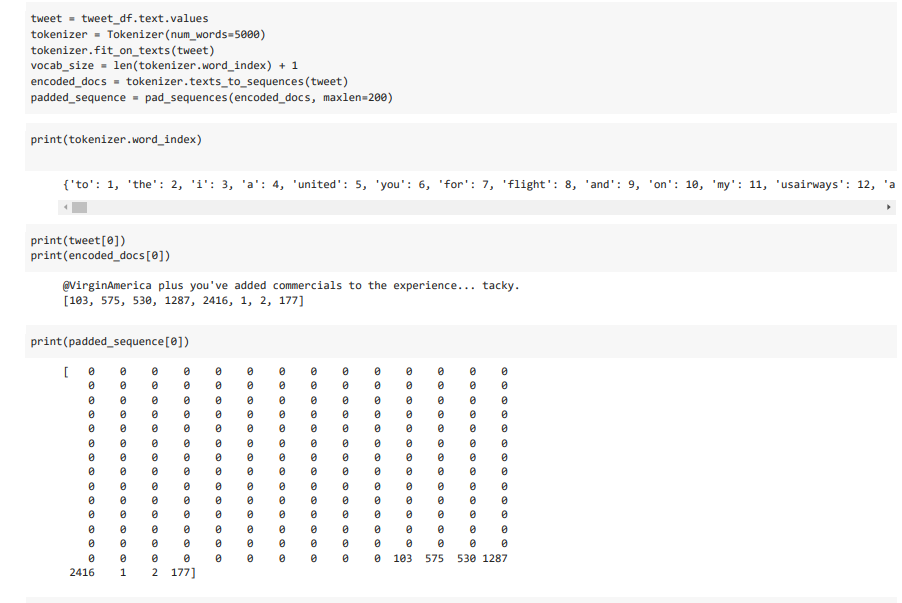
predict\_sentiment(test\_sentence2)

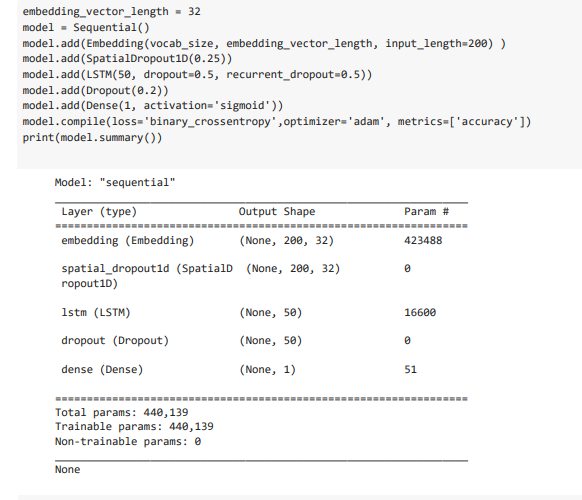
**OUTPUT:**

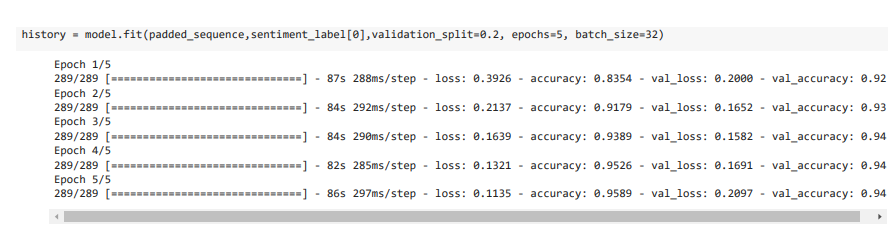
****

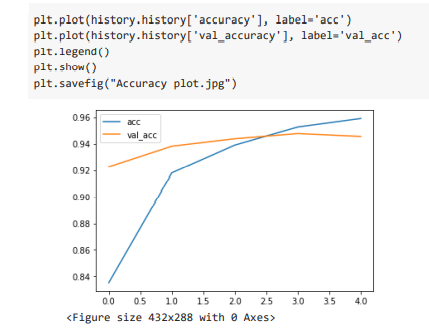
****

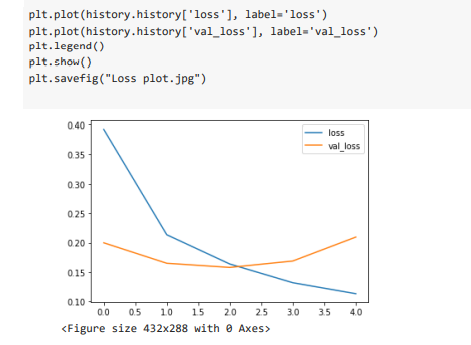
****

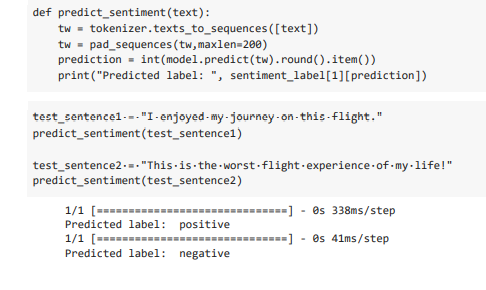
****

****

****

****

****

****