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:

```
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("Tweets.csv")
```

```
df.head()
```

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	neg
0	5.703060e+17	neutral	1.0000	NaN	
1	5.703010e+17	positive	0.3486	NaN	
2	5.703010e+17	neutral	0.6837	NaN	
3	5.703010e+17	negative	1.0000	Bad Flight	
4	5.703010e+17	negative	1.0000	Can't Tell	

```
df.columns
```

```
Index(['tweet_id', 'airline_sentiment', 'airline_sentiment_confidence',
       'negativereason', 'negativereason_confidence', 'airline',
       'airline_sentiment_gold', 'name', 'negativereason_gold',
       'retweet_count', 'text', 'tweet_coord', 'tweet_created',
       'tweet_location', 'user_timezone'],
      dtype='object')
```

```
tweet_df = df[['text','airline_sentiment']]
print(tweet_df.shape)
tweet_df.head(5)
```

```
(14640, 2)
```

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

```
tweet_df = tweet_df[tweet_df['airline_sentiment'] != 'neutral']
print(tweet_df.shape)
tweet_df.head(5)
```

```
(11541, 2)
```

	text	airline_sentiment
1	@VirginAmerica plus you've added commercials t...	positive
3	@VirginAmerica it's really aggressive to blast...	negative
4	@VirginAmerica and it's a really big bad thing...	negative
5	@VirginAmerica seriously would pay $30 a fligh...	negative
6	@VirginAmerica yes, nearly every time I fly VX...	positive

```
tweet_df["airline_sentiment"].value_counts()
```

```
negative    9178
positive    2363
Name: airline_sentiment, dtype: int64
```

```
sentiment_label = tweet_df.airline_sentiment.factorize()
sentiment_label
```

```
(array([0, 1, 1, ..., 0, 1, 1]),
 Index(['positive', 'negative'], dtype='object'))
```

```
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)
```

```
print(tokenizer.word_index)
```

```
{'to': 1, 'the': 2, 'i': 3, 'a': 4, 'united': 5, 'you': 6, 'for': 7, 'flight': 8, 'and': 9, 'on': 10, 'my': 11, 'usairways': 12, 'a
```

```
print(tweet[0])
print(encoded_docs[0])
```

```
@VirginAmerica plus you've added commercials to the experience... tacky.
[103, 575, 530, 1287, 2416, 1, 2, 177]
```

```
print(padded_sequence[0])
```

```
[   0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0  103  575  530 1287
 2416    1    2  177]
```

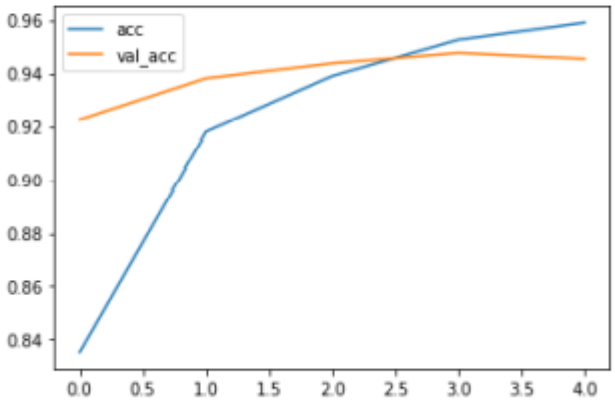
```
embedding_vector_length = 32
model = Sequential()
model.add(Embedding(vocab_size, embedding_vector_length, input_length=200) )
model.add(SpatialDropout1D(0.25))
model.add(LSTM(50, dropout=0.5, recurrent_dropout=0.5))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam', metrics=['accuracy'])
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 200, 32)           423488

 spatial_dropout1d (SpatialD  (None, 200, 32)          0
 ropout1D)

 lstm (LSTM)                 (None, 50)                16600

 dropout (Dropout)           (None, 50)                0

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 440,139
Trainable params: 440,139
Non-trainable params: 0
_________________________________________________________________
None
```

```
history = model.fit(padded_sequence,sentiment_label[0],validation_split=0.2, epochs=5, batch_size=32)
```

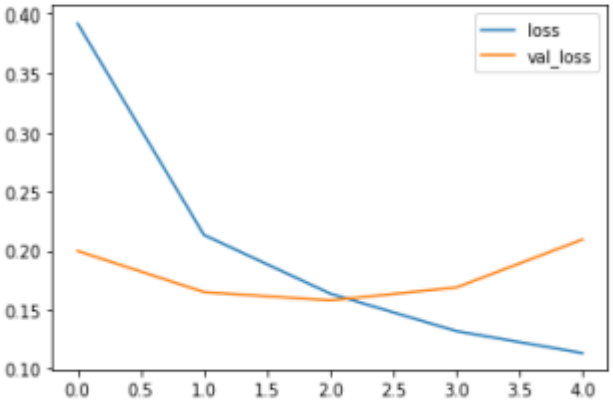
```
Epoch 1/5
289/289 [==============================] - 87s 288ms/step - loss: 0.3926 - accuracy: 0.8354 - val_loss: 0.2000 - val_accuracy: 0.92
Epoch 2/5
289/289 [==============================] - 84s 292ms/step - loss: 0.2137 - accuracy: 0.9179 - val_loss: 0.1652 - val_accuracy: 0.93
Epoch 3/5
289/289 [==============================] - 84s 290ms/step - loss: 0.1639 - accuracy: 0.9389 - val_loss: 0.1582 - val_accuracy: 0.94
Epoch 4/5
289/289 [==============================] - 82s 285ms/step - loss: 0.1321 - accuracy: 0.9526 - val_loss: 0.1691 - val_accuracy: 0.94
Epoch 5/5
289/289 [==============================] - 86s 297ms/step - loss: 0.1135 - accuracy: 0.9589 - val_loss: 0.2097 - val_accuracy: 0.94
```

```
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")
```




```
<Figure size 432x288 with 0 Axes>
```

```
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")
```




```
<Figure size 432x288 with 0 Axes>
```

```
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])
```

```
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)
```

```
1/1 [==============================] - 0s 338ms/step
Predicted label:  positive
1/1 [==============================] - 0s 41ms/step
Predicted label:  negative
```