In [3]:
```
# # Sentiment Analysis on US Airline Reviews

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(r"C:\Users\Kuhal Soni\OneDrive\Desktop\Sentiment Analysis project\Tw
```

In [4]:
```
df.head()
```

Out[4]:

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confiden
0	5.700000e+17	neutral	1.0000	NaN	Na
1	5.700000e+17	positive	0.3486	NaN	0.00
2	5.700000e+17	neutral	0.6837	NaN	Na
3	5.700000e+17	negative	1.0000	Bad Flight	0.70
4	5.700000e+17	negative	1.0000	Can't Tell	1.00

In [5]:
```
df.columns
```

Out[5]:
```
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')
```

In [6]:
```
tweet_df = df[['text','airline_sentiment']]
print(tweet_df.shape)
tweet_df.head(5)
```

```
(14640, 2)
```

Out[6]:

	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

In [7]:
```
tweet_df = tweet_df[tweet_df['airline_sentiment'] != 'neutral']
print(tweet_df.shape)
tweet_df.head(5)
```

```
(11541, 2)
```

Out[7]:

	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

In [8]:
```
tweet_df["airline_sentiment"].value_counts()
```

Out[8]:
```
negative    9178
positive    2363
Name: airline_sentiment, dtype: int64
```

In [9]:
```
sentiment_label = tweet_df.airline_sentiment.factorize()
sentiment_label
```

Out[9]:
```
(array([0, 1, 1, ..., 0, 1, 1], dtype=int64),
 Index(['positive', 'negative'], dtype='object'))
```

In [10]:
```
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 [11]:
```
print(tokenizer.word_index)
```

```
'tonight': 262, 'fit': 263, 'airlines': 264, "i'll": 265, 'rebooked': 266, 'say
s': 267, 'miles': 268, 'put': 269, 'mins': 270, 'morning': 271, 'reservation': 27
2, 'dfw': 273, 'he': 274, 'issues': 275, 'awesome': 276, 'info': 277, 'think': 27
8, 'hope': 279, 'nice': 280, 'pay': 281, 'air': 282, 'down': 283, 'working': 284,
'booking': 285, 'finally': 286, 'tell': 287, 'ago': 288, 'rebook': 289, 'use': 29
0, 'lax': 291, 'anything': 292, 'also': 293, 'wifi': 294, 'its': 295, '20': 296,
'every': 297, 'anyone': 298, 'until': 299, "what's": 300, 'appreciate': 301, 'fre
e': 302, 'done': 303, 'missing': 304, 'week': 305, 'having': 306, 'business': 30
7, 'helpful': 308, 'terrible': 309, 'answer': 310, 'dca': 311, 'phl': 312, 'makin
g': 313, 'always': 314, 'person': 315, 'say': 316, 'sfo': 317, 'able': 318, 'prob
lem': 319, '8': 320, 'fail': 321, 'checked': 322, 'board': 323, 'team': 324, 'dis
appointed': 325, 'without': 326, 'fix': 327, '7': 328, 'credit': 329, 'delta': 33
0, 'class': 331, 'rep': 332, 'voucher': 333, 'speak': 334, 'which': 335, 'updat
e': 336, 'amazing': 337, 'ord': 338, 'yesterday': 339, 'ridiculous': 340, 'almos
t': 341, 'thx': 342, 'paid': 343, 'understand': 344, 'unacceptable': 345, 'attend
ant': 346, 'early': 347, "couldn't": 348, 'come': 349, 'app': 350, 'leave': 351,
'extra': 352, 'hung': 353, 'happy': 354, 'money': 355, 'available': 356, 'sorry':
357, 'many': 358, 'claim': 359, "isn't": 360, '15': 361, 'look': 362, 'pilot': 36
3, 'poor': 364, "haven't": 365, 'ewr': 366, 'talk': 367, 'horrible': 368, 'stop':
369, 'full': 370, 'planes': 371, 'tarmac': 372, 'employees': 373, '45': 374, 'lea
```

In [12]:
```
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]
```

In [13]:
```
print(padded_sequence[0])
```

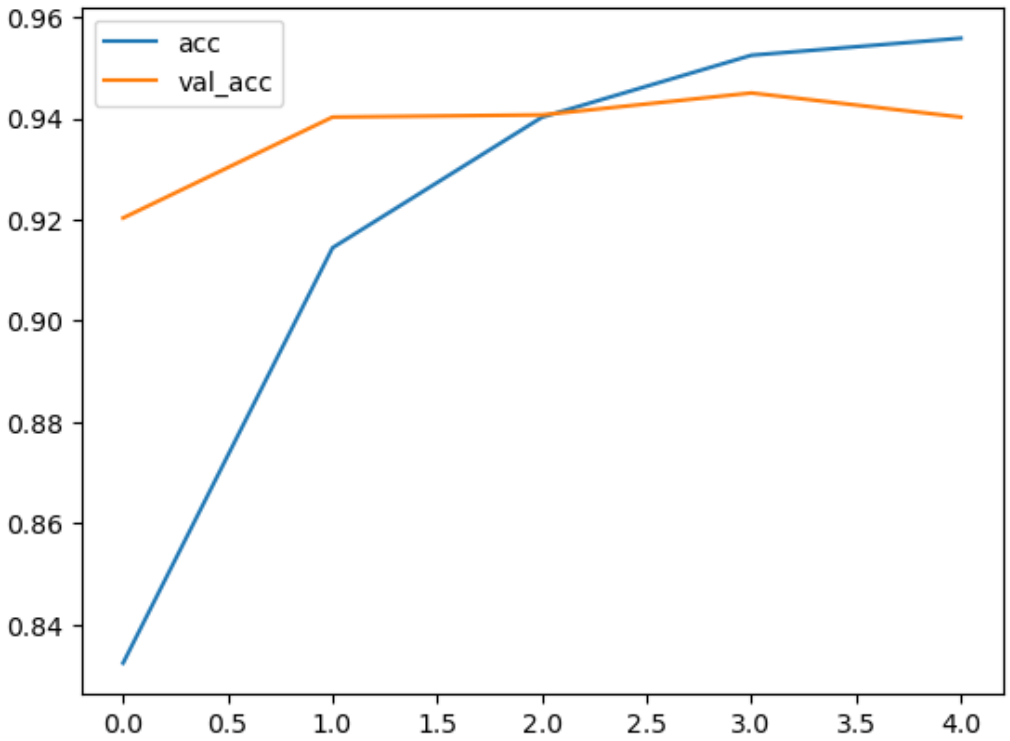
```
[   0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0  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

In [14]:
```
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
```

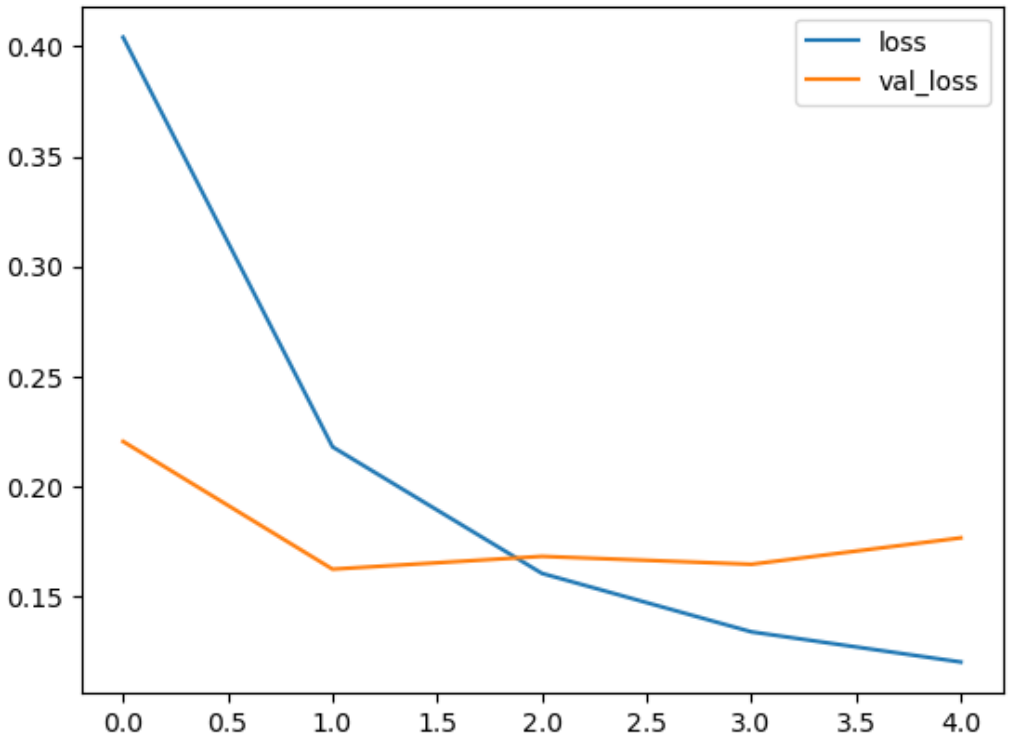
In [16]:
```
history = model.fit(padded_sequence,sentiment_label[0],validation_split=0.2, epochs=
plt.plot(history.history['accuracy'], label='acc')
plt.plot(history.history['val_accuracy'], label='val_acc')
plt.legend()
plt.show()
plt.savefig("Accuracy plot.jpg")
```

```
Epoch 1/5
289/289 [==============================] - 56s 181ms/step - loss: 0.4042 - accurac
y: 0.8324 - val_loss: 0.2203 - val_accuracy: 0.9203
Epoch 2/5
289/289 [==============================] - 57s 197ms/step - loss: 0.2178 - accurac
y: 0.9144 - val_loss: 0.1622 - val_accuracy: 0.9402
Epoch 3/5
289/289 [==============================] - 51s 176ms/step - loss: 0.1603 - accurac
y: 0.9402 - val_loss: 0.1680 - val_accuracy: 0.9407
Epoch 4/5
289/289 [==============================] - 53s 184ms/step - loss: 0.1337 - accurac
y: 0.9524 - val_loss: 0.1644 - val_accuracy: 0.9450
Epoch 5/5
289/289 [==============================] - 51s 175ms/step - loss: 0.1200 - accurac
y: 0.9558 - val_loss: 0.1764 - val_accuracy: 0.9402
```




```
<Figure size 640x480 with 0 Axes>
```

In [17]:
```
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 640x480 with 0 Axes>
```

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 [20]:
```
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 363ms/step
Predicted label:  positive
1/1 [==============================] - 0s 40ms/step
Predicted label:  negative
```

In [ ]:
```
```