In [1]:

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
data=pd.read_csv(r"C:\Users\reshma_koduri\Downloads\Tweets airline.csv")
```

In [3]:

```
data
```

Out[3]:

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	
0	5.700000e+17	neutral	1.0000	NaN	NaN	/
1	5.700000e+17	positive	0.3486	NaN	0.0000	/
2	5.700000e+17	neutral	0.6837	NaN	NaN	/
3	5.700000e+17	negative	1.0000	Bad Flight	0.7033	/
4	5.700000e+17	negative	1.0000	Can't Tell	1.0000	/
...	...	...	...	...	...	
14635	5.700000e+17	positive	0.3487	NaN	0.0000	A
14636	5.700000e+17	negative	1.0000	Customer Service Issue	1.0000	A
14637	5.700000e+17	neutral	1.0000	NaN	NaN	A
14638	5.700000e+17	negative	1.0000	Customer Service Issue	0.6659	A
14639	5.700000e+17	neutral	0.6771	NaN	0.0000	A

14640 rows × 12 columns

In [4]:
```
data.head(10)
```

Out[4]:

	tweet_id	airline_sentiment	airline_sentiment_confidence	negative reason	negative reason_confidence	airline
0	5.700000e+17	neutral	1.0000	NaN	NaN	Virgin America
1	5.700000e+17	positive	0.3486	NaN	0.0000	Virgin America
2	5.700000e+17	neutral	0.6837	NaN	NaN	Virgin America
3	5.700000e+17	negative	1.0000	Bad Flight	0.7033	Virgin America
4	5.700000e+17	negative	1.0000	Can't Tell	1.0000	Virgin America
5	5.700000e+17	negative	1.0000	Can't Tell	0.6842	Virgin America
6	5.700000e+17	positive	0.6745	NaN	0.0000	Virgin America
7	5.700000e+17	neutral	0.6340	NaN	NaN	Virgin America
8	5.700000e+17	positive	0.6559	NaN	NaN	Virgin America
9	5.700000e+17	positive	1.0000	NaN	NaN	Virgin America

In [5]:
```
data.tail(10)
```

Out[5]:

	tweet_id	airline_sentiment	airline_sentiment_confidence	negative reason	negative reason_confidence	
14630	5.700000e+17	positive	1.0000	NaN	NaN	A

	tweet_id	airline_sentiment	airline_sentiment_confidence	negative reason	negative reason_confidence	
14631	5.700000e+17	negative	1.0000	Bad Flight	1.0000	A
14632	5.700000e+17	neutral	0.6760	NaN	0.0000	A
14633	5.700000e+17	negative	1.0000	Cancelled Flight	1.0000	A
14634	5.700000e+17	negative	0.6684	Late Flight	0.6684	A
14635	5.700000e+17	positive	0.3487	NaN	0.0000	A
14636	5.700000e+17	negative	1.0000	Customer Service Issue	1.0000	A
14637	5.700000e+17	neutral	1.0000	NaN	NaN	A
14638	5.700000e+17	negative	1.0000	Customer Service Issue	0.6659	A
14639	5.700000e+17	neutral	0.6771	NaN	0.0000	A

In [6]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14640 entries, 0 to 14639
Data columns (total 12 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   tweet_id                      14640 non-null  float64
 1   airline_sentiment             14640 non-null  object
 2   airline_sentiment_confidence  14640 non-null  float64
 3   negative reason               9178 non-null   object
 4   negative reason_confidence    10522 non-null  float64
 5   airline                       14640 non-null  object
 6   name                          14640 non-null  object
 7   retweet_count                 14640 non-null  int64
 8   text                          14640 non-null  object
```

```
 9   tweet_created                 14640 non-null  object
 10  tweet_location                9907 non-null   object
 11  user_timezone                 9820 non-null   object
dtypes: float64(3), int64(1), object(8)
memory usage: 1.3+ MB
```

In [7]:

```
data.describe()
```

Out[7]:

	tweet_id	airline_sentiment_confidence	negative reason_confidence	retweet_count
count	1.464000e+04	14640.000000	10522.000000	14640.000000
mean	5.692605e+17	0.900169	0.638298	0.082650
std	8.098842e+14	0.162830	0.330440	0.745778
min	5.680000e+17	0.335000	0.000000	0.000000
25%	5.690000e+17	0.692300	0.360600	0.000000
50%	5.690000e+17	1.000000	0.670600	0.000000
75%	5.700000e+17	1.000000	1.000000	0.000000
max	5.700000e+17	1.000000	1.000000	44.000000

In [8]:

```
data.shape
```

Out[8]:

```
(14640, 12)
```

In [9]:

```
data.isna().sum()
```

Out[9]:

```
tweet_id                          0
airline_sentiment                 0
airline_sentiment_confidence      0
negative reason                5462
negative reason_confidence     4118
airline                           0
name                              0
retweet_count                     0
text                              0
tweet_created                     0
tweet_location                 4733
user_timezone                  4820
dtype: int64
```

In [10]:

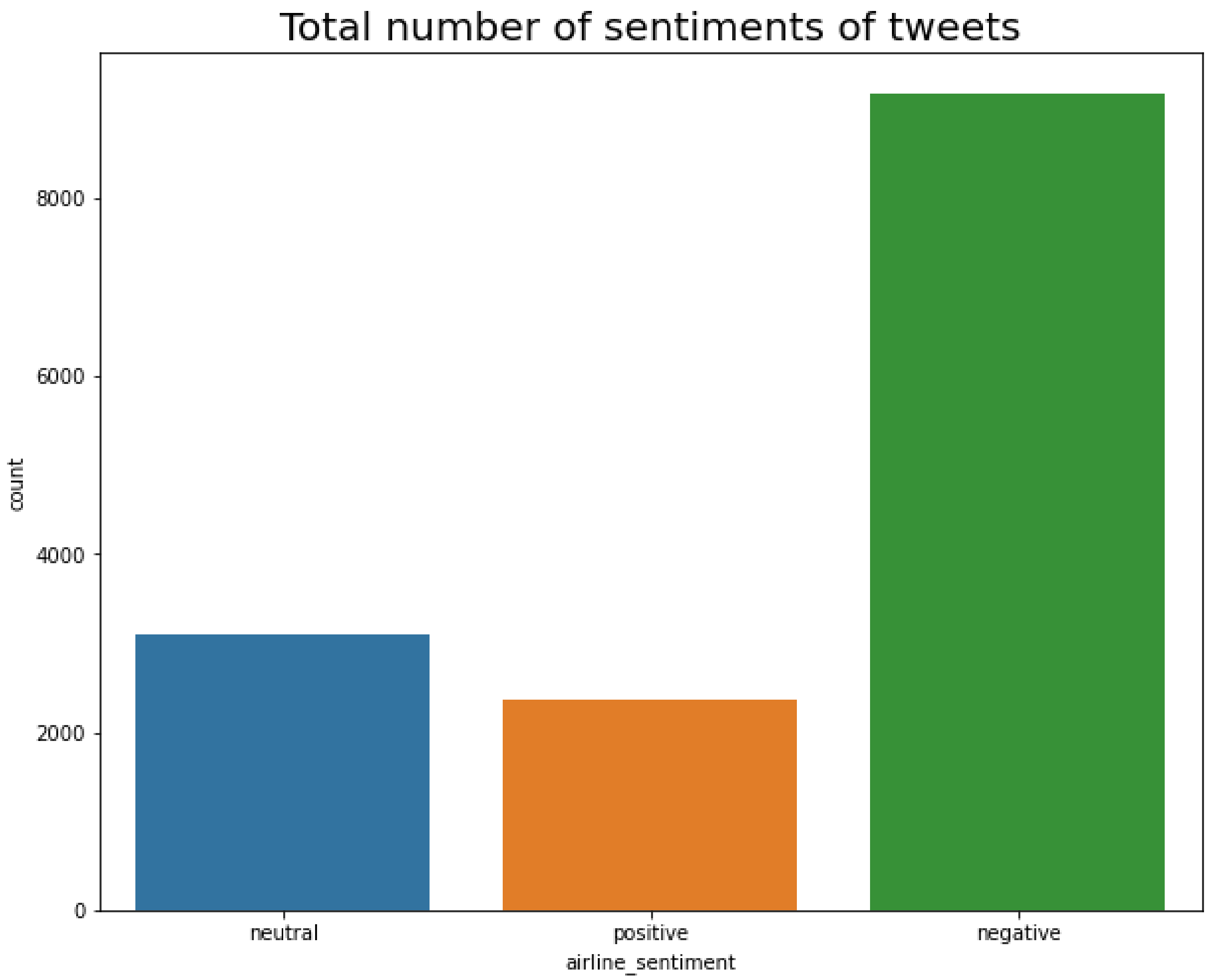
```
data.groupby(['airline_sentiment']).count()
```

Out[10]:

	tweet_id	airline_sentiment_confidence	negative reason	negative reason_confidence	airline	name
airline_sentiment						
negative	9178	9178	9178	9178	9178	9178
neutral	3099	3099	0	1014	3099	3099
positive	2363	2363	0	330	2363	2363

In [11]:

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize = (10, 8))
ax = sns.countplot(x = 'airline_sentiment', data = data)
ax.set_title(label = 'Total number of sentiments of tweets', fontsize = 20)
plt.show()
```

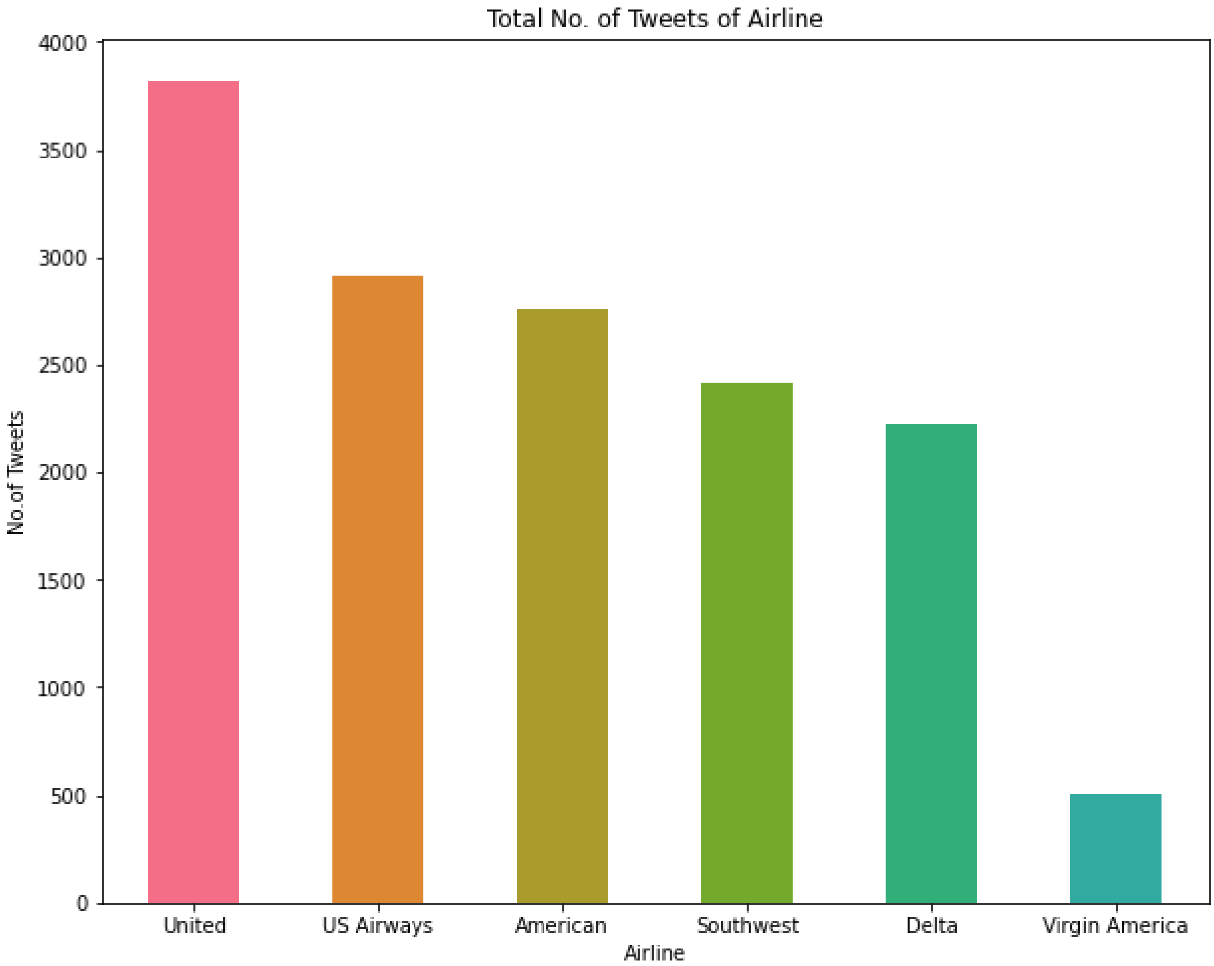


In [12]:

```
colors=sns.color_palette('husl',10)
pd.Series(data['airline']).value_counts().plot(kind="bar",color=colors,figsize=(10,8
plt.xlabel('Airline',fontsize=10)
plt.ylabel('No.of Tweets',fontsize=10)
```

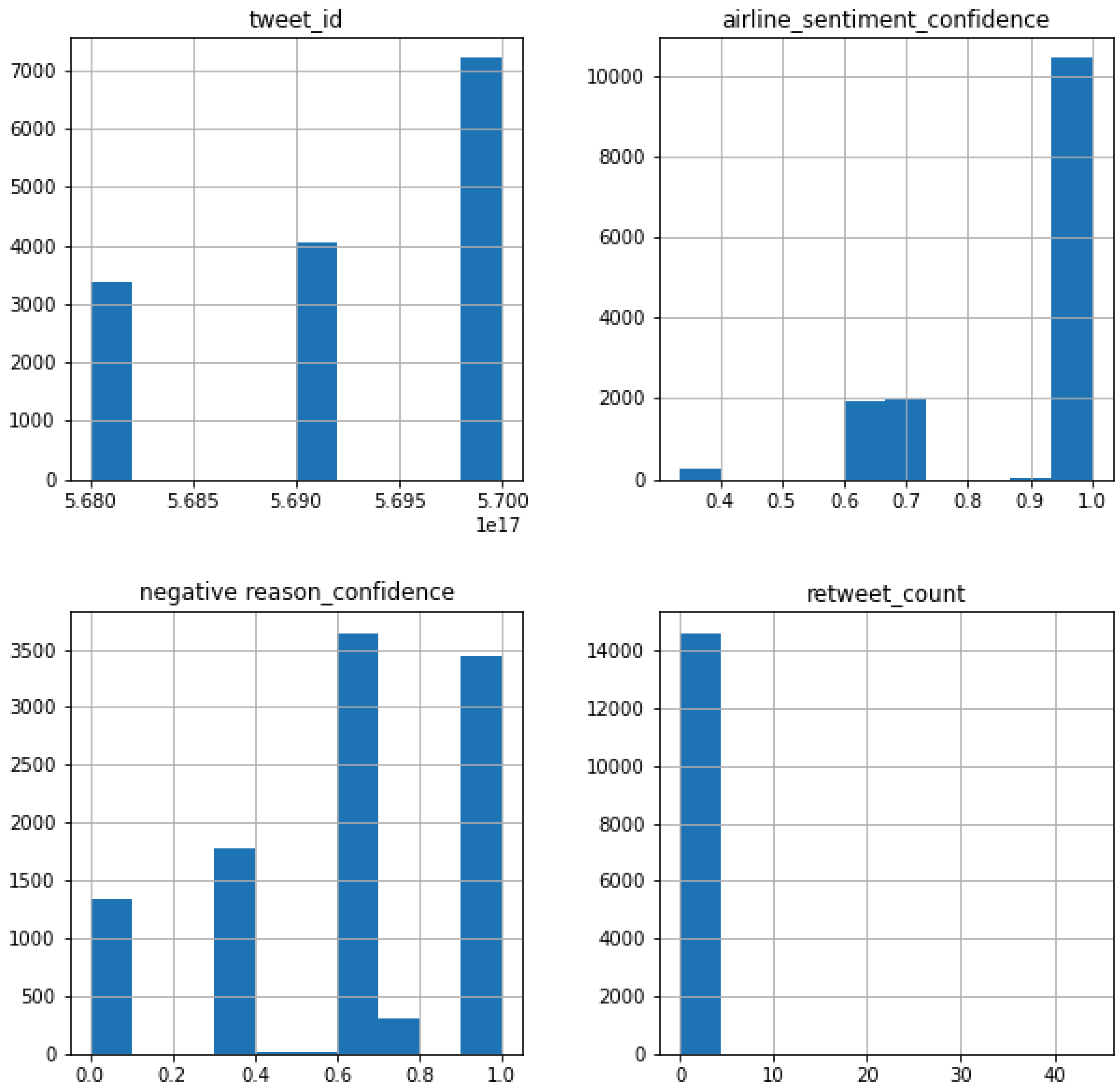
Out[12]:

```
Text(0, 0.5, 'No.of Tweets')
```



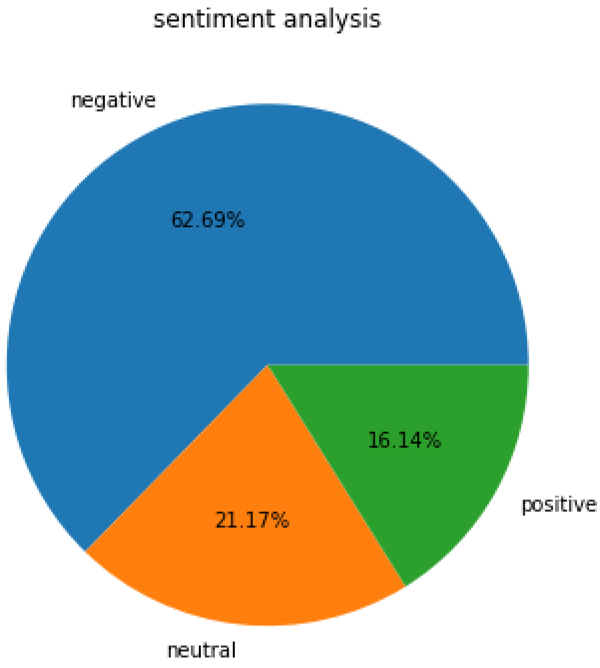
In [13]:
```
data.hist(figsize=(10,10))
```

Out[13]:
```
array([[<AxesSubplot:title={'center':'tweet_id'}>,
        <AxesSubplot:title={'center':'airline_sentiment_confidence'}>],
       [<AxesSubplot:title={'center':'negative reason_confidence'}>,
        <AxesSubplot:title={'center':'retweet_count'}>]], dtype=object)
```



In [14]:

```
plt.figure(figsize=(8, 6))
data['airline_sentiment'].value_counts().plot.pie(autopct='%2.2f%%')
plt.title('sentiment analysis')
plt.ylabel('')
plt.show()
```



In [15]:
```
data['final_text'] = data['negative reason'].fillna('') + ' ' + data['text']
```

In [16]:
```
#import nltk
#nltk.download('stopwords')
```

In [17]:
```
data['airline_sentiment']=data['airline_sentiment'].map({'neutral':0,'positive':1,'n
data
```

Out[17]:

	tweet_id	airline_sentiment	airline_sentiment_confidence	negative reason	negative reason_confidence	
0	5.700000e+17	0	1.0000	NaN	NaN	/
1	5.700000e+17	1	0.3486	NaN	0.0000	/
2	5.700000e+17	0	0.6837	NaN	NaN	/
3	5.700000e+17	-1	1.0000	Bad Flight	0.7033	/
4	5.700000e+17	-1	1.0000	Can't Tell	1.0000	/
...	...	...	...	...	...	
14635	5.700000e+17	1	0.3487	NaN	0.0000	Ai

	tweet_id	airline_sentiment	airline_sentiment_confidence	negative reason	negative reason_confidence	
14636	5.700000e+17	-1	1.0000	Customer Service Issue	1.0000	Ai
14637	5.700000e+17	0	1.0000	NaN	NaN	Ai
14638	5.700000e+17	-1	1.0000	Customer Service Issue	0.6659	Ai
14639	5.700000e+17	0	0.6771	NaN	0.0000	Ai

14640 rows × 13 columns

In [18]:

```
data['final_text']
```

Out[18]:

```
0                       @VirginAmerica What @dhepburn said.
1             @VirginAmerica plus you've added commercials ...
2             @VirginAmerica I didn't today... Must mean I ...
3            Bad Flight @VirginAmerica it's really aggressi...
4            Can't Tell @VirginAmerica and it's a really bi...
                                ...
14635         @AmericanAir thank you we got on a different ...
14636        Customer Service Issue @AmericanAir leaving ov...
14637         @AmericanAir Please bring American Airlines t...
14638        Customer Service Issue @AmericanAir you have m...
14639         @AmericanAir we have 8 ppl so we need 2 know ...
Name: final_text, Length: 14640, dtype: object
```

In [19]:

```
cor=data.corr()
cor
```

Out[19]:

	tweet_id	airline_sentiment	airline_sentiment_confidence	nega reason_confide
tweet_id	1.000000	-0.087910	0.037667	0.034
airline_sentiment	-0.087910	1.000000	-0.205936	-0.693
airline_sentiment_confidence	0.037667	-0.205936	1.000000	0.685
negative reason_confidence	0.034864	-0.693294	0.685879	1.000
retweet_count	-0.006689	-0.015717	0.012581	0.021

In [20]:
```
import seaborn as sb
sb.heatmap(cor,vmax=0,vmin=-2,annot=True,linewidth=-5,cmap="rocket")
```

Out[20]: <AxesSubplot:>

In [21]:
```
x=data['final_text']
y=data['airline_sentiment']
```

In [22]:
```
x
```

Out[22]:
```
0                        @VirginAmerica What @dhepburn said.
1          @VirginAmerica plus you've added commercials ...
2          @VirginAmerica I didn't today... Must mean I ...
3         Bad Flight @VirginAmerica it's really aggressi...
4         Can't Tell @VirginAmerica and it's a really bi...
                                ...
14635      @AmericanAir thank you we got on a different ...
14636     Customer Service Issue @AmericanAir leaving ov...
14637      @AmericanAir Please bring American Airlines t...
14638     Customer Service Issue @AmericanAir you have m...
14639      @AmericanAir we have 8 ppl so we need 2 know ...
Name: final_text, Length: 14640, dtype: object
```

In [23]:
```
y
```

Out[23]:
```
0         0
1         1
2         0
3        -1
4        -1
         ..
14635     1
14636    -1
14637     0
14638    -1
```

```
14639    0
Name: airline_sentiment, Length: 14640, dtype: int64
```

In [24]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
tfid = TfidfVectorizer()
```

In [25]:
```
x_final=tfid.fit_transform(x)
```

In [26]:
```
x_final
```

Out[26]:
```
<14640x15052 sparse matrix of type '<class 'numpy.float64'>'
	with 252187 stored elements in Compressed Sparse Row format>
```

In [27]:
```
from imblearn.over_sampling import SMOTE
smote = SMOTE()
x_sm,y_sm = smote.fit_resample(x_final,y)
```

In [28]:
```
from sklearn.model_selection import train_test_split
```

In [29]:
```
x_train,x_test,y_train,y_test=train_test_split(x_sm,y_sm,test_size=0.33,random_state
```

In [30]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [31]:
```
cls=RandomForestClassifier()
```

In [32]:
```
cls.fit(x_train,y_train)
```

Out[32]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [33]:
```
ypred=cls.predict(x_test)
ypred
```

Out[33]:
```
array([-1, -1,  1, ...,  1,  1,  1], dtype=int64)
```

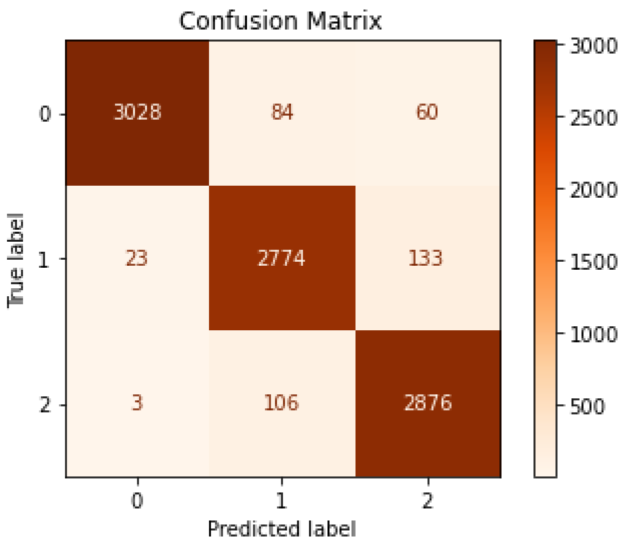
In [34]:
```
from sklearn.metrics import accuracy_score,confusion_matrix
```

In [35]:
```
confusion_matrix(ypred,y_test)
```

Out[35]:
```
array([[3028,   84,   60],
       [  23, 2774,  133],
       [   3,  106, 2876]], dtype=int64)
```

In [36]:
```
from sklearn.metrics import ConfusionMatrixDisplay
cm1=confusion_matrix(ypred,y_test)
ConfusionMatrixDisplay(cm1).plot(cmap='Oranges')
plt.title('Confusion Matrix')
```

Out[36]:
```
Text(0.5, 1.0, 'Confusion Matrix')
```



In [37]:
```
accuracy_score(ypred,y_test)
```

Out[37]:
```
0.9549906459777704
```

In [38]:
```
from sklearn.metrics import classification_report
report1=classification_report(ypred,y_test)
print("Classification Report:\n", report1)
```

```
Classification Report:
               precision    recall  f1-score   support

          -1       0.99      0.95      0.97      3172
           0       0.94      0.95      0.94      2930
           1       0.94      0.96      0.95      2985

    accuracy                           0.95      9087
   macro avg       0.95      0.95      0.95      9087
weighted avg       0.96      0.95      0.96      9087
```

In [39]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [40]:
```
tree=DecisionTreeClassifier()
```

In [41]:
```
tree.fit(x_train,y_train)
```

Out[41]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [42]:
```
ypred1=tree.predict(x_test)
ypred1
```

Out[42]:
```
array([-1, -1,  1, ...,  1,  1,  1], dtype=int64)
```

In [43]:
```
confusion_matrix(ypred1,y_test)
```

Out[43]:
```
array([[3000,   43,   25],
       [  36, 2663,  269],
       [  18,  258, 2775]], dtype=int64)
```

In [44]:
```
cm2=confusion_matrix(ypred1,y_test)
#ConfusionMatrixDisplay(cm2).plot(cmap='Oranges')
#plt.title('Confusion Matrix')
```

In [45]:
```
accuracy_score(ypred1,y_test)
```

Out[45]:
```
0.9285792890943105
```

In [46]:
```
from sklearn.metrics import classification_report
report2=classification_report(ypred1,y_test)
print("Classification Report:\n", report2)
```

```
Classification Report:
               precision    recall  f1-score   support

          -1       0.98      0.98      0.98      3068
           0       0.90      0.90      0.90      2968
           1       0.90      0.91      0.91      3051

    accuracy                           0.93      9087
   macro avg       0.93      0.93      0.93      9087
weighted avg       0.93      0.93      0.93      9087
```

In [47]:
```
from sklearn.linear_model import LogisticRegression
```

In [48]:
```
reg=LogisticRegression()
```

In [49]:
```
reg.fit(x_train,y_train)
```

Out[49]:
```
▾ LogisticRegression

LogisticRegression()
```

In [50]:
```
ypred2=reg.predict(x_test)
ypred2
```

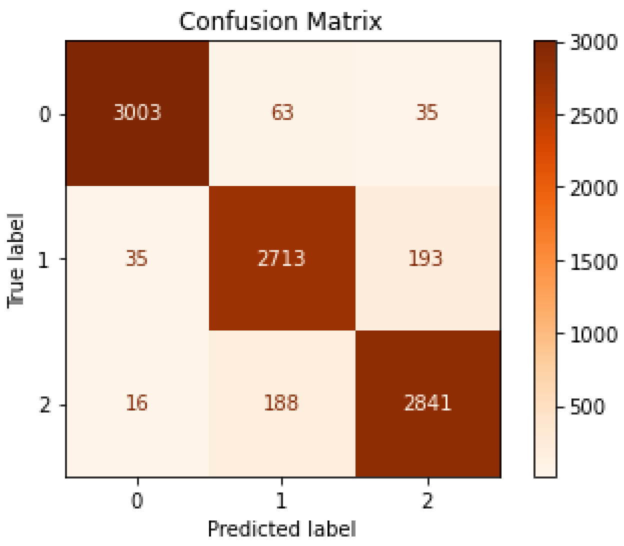
Out[50]:
```
array([-1, -1,  1, ...,  1,  1,  1], dtype=int64)
```

In [51]:
```
confusion_matrix(ypred2,y_test)
```

Out[51]:
```
array([[3003,   63,   35],
       [  35, 2713,  193],
       [  16,  188, 2841]], dtype=int64)
```

In [52]:
```
cm3=confusion_matrix(ypred2,y_test)
ConfusionMatrixDisplay(cm3).plot(cmap='Oranges')
plt.title('Confusion Matrix')
```

Out[52]: Text(0.5, 1.0, 'Confusion Matrix')



In [53]:
```
accuracy_score(ypred2,y_test)
```

Out[53]: 0.9416749202156928

In [54]:
```
from sklearn.metrics import classification_report
report3=classification_report(ypred2,y_test)
print("Classification Report:\n", report3)
```

```
Classification Report:
               precision    recall  f1-score   support

          -1       0.98      0.97      0.98      3101
           0       0.92      0.92      0.92      2941
           1       0.93      0.93      0.93      3045

    accuracy                           0.94      9087
   macro avg       0.94      0.94      0.94      9087
weighted avg       0.94      0.94      0.94      9087
```

In [1]:
```
import tkinter as tk
from tkinter import ttk
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

# Function to analyze sentiment
def analyze_sentiment():
    text = text_entry.get("1.0", "end-1c")  # Get text from the Text widget
    if text:
        analyzer = SentimentIntensityAnalyzer()
        sentiment_scores = analyzer.polarity_scores(text)
        sentiment = sentiment_scores['compound']
        if sentiment >= 0.05:
            result_label.config(text="Positive")
        elif sentiment <= -0.05:
            result_label.config(text="Negative")
        else:
```

```
            result_label.config(text="Neutral")
    else:
        result_label.config(text="Please enter text!")

# Create the main window
window = tk.Tk()
window.title('Sentiment Analysis')

# Create and place widgets
text_label = ttk.Label(window, text='Enter text:')
text_label.pack()

text_entry = tk.Text(window, height=5, width=50)
text_entry.pack()

analyze_button = ttk.Button(window, text='Analyze Sentiment', command=analyze_sentim
analyze_button.pack()

result_label = ttk.Label(window, text='')
result_label.pack()
# Start the GUI event loop
window.mainloop()
```

In [ ]:

```

```