DAT305

November 25, 2024

[1]:
```
import sys,os
import pandas as pd
pd.options.mode.chained_assignment = None
import matplotlib.pyplot as plt
from matplotlib.patches import ConnectionPatch
import seaborn as sns
import numpy as np
import sklearn
import string
import re
import nltk
import tensorflow as tf
from collections import Counter
from tensorflow import keras
from sklearn.feature_extraction.text import TfidfVectorizer
from keras.utils import pad_sequences,to_categorical
from sklearn.feature_selection import SelectKBest,chi2
from sklearn.utils.class_weight import compute_class_weight
from sklearn.metrics import␣
↪accuracy_score,f1_score,roc_auc_score,confusion_matrix,precision_score,recall_score,classif
from datetime import datetime
nltk.download('punkt')
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
from nltk.tokenize import TweetTokenizer
from google.colab import files,drive
from wordcloud import WordCloud, STOPWORDS
from sklearn.svm import LinearSVC
from sklearn.model_selection import GridSearchCV
import warnings
warnings.filterwarnings(action="ignore", message="^internal gelsd")
print("Running Panda Version:"+pd.__version__)
print("Running TensorFlow Version:"+ tf.__version__)
#print("Running Keras API Version:"+ keras.__version__)
```

```
print("Running Python {0}.{1}".format(sys.version_info[:2][0],sys.version_info[:
↪2][1]))
```

```
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
Running Panda Version:2.2.2
Running TensorFlow Version:2.17.1
Running Python 3.10
```

[2]:
```
seed = 0
tf.keras.utils.set_random_seed(seed)
```

[3]:
```
uploaded = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving Tweets.csv to Tweets.csv
```

1 *Exploratory Data Analysis*

[4]:
```
dataset = pd.read_csv("Tweets.csv",na_values=['NA'], low_memory=False)
```

1.0.1 Dataset shapes

[5]:
```
print('Dataset structure: rows =',dataset.shape[0], ' - columns =',dataset.
↪shape[1])
```

```
Dataset structure: rows = 14640  - columns = 15
```

Some random rows

[6]:
```
dataset.sample(3)
```

[6]:
```
                  tweet_id airline_sentiment  airline_sentiment_confidence  \
13983  569682010270101504          negative                        0.6163
14484  569608307184242688          negative                        0.7039
6403   567879304593408001          negative                        1.0000

         negativereason  negativereason_confidence    airline  \
13983       Late Flight                     0.6163   American
14484        Bad Flight                     0.3587   American
6403   Cancelled Flight                     1.0000  Southwest

      airline_sentiment_gold          name negativereason_gold  retweet_count  \
13983                    NaN      zsalim03                 NaN              0
14484                    NaN      sa_craig                 NaN              0
6403                     NaN  DanaChristos                 NaN              1
```

```
                                                    text tweet_coord  \
13983  @AmericanAir In car gng to DFW. Pulled over 1h…         NaN
14484  @AmericanAir after all, the plane didn’t land …         NaN
6403   @SouthwestAir can't believe how many paying cu…         NaN

                   tweet_created       tweet_location  \
13983  2015-02-22 18:15:50 -0800                Texas
14484  2015-02-22 13:22:57 -0800  College Station, TX
6403   2015-02-17 18:52:31 -0800                   CT

                    user_timezone
13983  Central Time (US & Canada)
14484  Central Time (US & Canada)
6403   Eastern Time (US & Canada)
```

1.0.2 Descriptive statistics for the dataset

[7]:
```python
print('Dataset Features types:')
dataset.dtypes
```

```
Dataset Features types:
```

[7]:
```
tweet_id                          int64
airline_sentiment                object
airline_sentiment_confidence    float64
negativereason                   object
negativereason_confidence       float64
airline                          object
airline_sentiment_gold           object
name                             object
negativereason_gold              object
retweet_count                     int64
text                             object
tweet_coord                      object
tweet_created                    object
tweet_location                   object
user_timezone                    object
dtype: object
```

[8]:
```python
print("List of names of columns:\n")
print('-'*40)
dataset.columns.tolist()
```

```
List of names of columns:

----------------------------------------
```

[8]:
```
['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']
```

[9]:
```
print('Descriptive Statistics for numeric features on Dataset')
dataset.describe(include=np.number).T
```

```
Descriptive Statistics for numeric features on Dataset
```

[9]:
```
                                count          mean           std  \
tweet_id                      14640.0  5.692184e+17  7.791112e+14
airline_sentiment_confidence  14640.0  9.001689e-01  1.628300e-01
negativereason_confidence     10522.0  6.382983e-01  3.304398e-01
retweet_count                 14640.0  8.265027e-02  7.457782e-01

                                       min           25%           50%  \
tweet_id                      5.675883e+17  5.685592e+17  5.694779e+17
airline_sentiment_confidence  3.350000e-01  6.923000e-01  1.000000e+00
negativereason_confidence     0.000000e+00  3.606000e-01  6.706000e-01
retweet_count                 0.000000e+00  0.000000e+00  0.000000e+00

                                       75%           max
tweet_id                      5.698905e+17  5.703106e+17
airline_sentiment_confidence  1.000000e+00  1.000000e+00
negativereason_confidence     1.000000e+00  1.000000e+00
retweet_count                 0.000000e+00  4.400000e+01
```

[10]:
```
print('Range for numeric features on Dataset')
print('-'*40)
dataset.max(numeric_only=True) - dataset.min(numeric_only=True)
```

```
Range for numeric features on Dataset
----------------------------------------
```

[10]:
```
tweet_id                        2.722322e+15
airline_sentiment_confidence    6.650000e-01
```

```
negativereason_confidence       1.000000e+00
retweet_count                   4.400000e+01
dtype: float64
```

[11]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14640 entries, 0 to 14639
Data columns (total 15 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   tweet_id                      14640 non-null  int64
 1   airline_sentiment             14640 non-null  object
 2   airline_sentiment_confidence  14640 non-null  float64
 3   negativereason                9178 non-null   object
 4   negativereason_confidence     10522 non-null  float64
 5   airline                       14640 non-null  object
 6   airline_sentiment_gold        40 non-null     object
 7   name                          14640 non-null  object
 8   negativereason_gold           32 non-null     object
 9   retweet_count                 14640 non-null  int64
 10  text                          14640 non-null  object
 11  tweet_coord                   1019 non-null   object
 12  tweet_created                 14640 non-null  object
 13  tweet_location                9907 non-null   object
 14  user_timezone                 9820 non-null   object
dtypes: float64(2), int64(2), object(11)
memory usage: 1.7+ MB
```

[12]:
```
print('Descriptive Statistics for categorical features on Dataset')
dataset.describe(include='O').T
```

```
Descriptive Statistics for categorical features on Dataset
```

[12]:

	count	unique	top	freq
airline_sentiment	14640	3	negative	9178
negativereason	9178	10	Customer Service Issue	2910
airline	14640	6	United	3822
airline_sentiment_gold	40	3	negative	32
name	14640	7701	JetBlueNews	63
negativereason_gold	32	13	Customer Service Issue	12
text	14640	14427	@united thanks	6
tweet_coord	1019	832	[0.0, 0.0]	164
tweet_created	14640	14247	2015-02-24 09:54:34 -0800	5
tweet_location	9907	3081	Boston, MA	157
user_timezone	9820	85	Eastern Time (US & Canada)	3744

[13]:
```
print('Number of classes: ',len(dataset.airline_sentiment.unique().tolist()))
```

```
Number of classes:  3
```

[14]:
```python
print('Name of classes (y): ',dataset.airline_sentiment.unique().tolist())
```

```
Name of classes (y):  ['neutral', 'positive', 'negative']
```

[15]:
```python
print('Check for duplicates?',dataset.duplicated().any())
print('-'*70)
print('Sum of duplicated rows :',dataset.duplicated().sum())
```

```
Check for duplicates? True
----------------------------------------------------------------------
Sum of duplicated rows : 36
```

[16]:
```python
print('Check for missing values ',dataset.isnull().any().any())
print('-'*40)
print('Sum of Missing values accross columns\n',dataset.isnull().sum())
print('-'*40)
print("Sum of missing values",sum(dataset.isnull().sum()))
```

```
Check for missing values  True
----------------------------------------
Sum of Missing values accross columns
 tweet_id                            0
airline_sentiment                   0
airline_sentiment_confidence        0
negativereason                   5462
negativereason_confidence        4118
airline                             0
airline_sentiment_gold          14600
name                                0
negativereason_gold             14608
retweet_count                       0
text                                0
tweet_coord                     13621
tweet_created                       0
tweet_location                   4733
user_timezone                    4820
dtype: int64
----------------------------------------
Sum of missing values 61962
```

[17]:
```python
missing_data = dataset[dataset.isnull().any(axis=1)]
missing_data.head(3)
```

[17]:
```
             tweet_id airline_sentiment  airline_sentiment_confidence  \
0  570306133677760513           neutral                        1.0000
1  570301130888122368          positive                        0.3486
2  570301083672813571           neutral                        0.6837
```

```
  negativereason  negativereason_confidence         airline  \
0            NaN                        NaN  Virgin America
1            NaN                        0.0  Virgin America
2            NaN                        NaN  Virgin America

  airline_sentiment_gold        name negativereason_gold  retweet_count  \
0                    NaN     cairdin                 NaN              0
1                    NaN    jnardino                 NaN              0
2                    NaN  yvonnalynn                 NaN              0

                                                text tweet_coord  \
0                @VirginAmerica What @dhepburn said.         NaN
1  @VirginAmerica plus you've added commercials t…         NaN
2  @VirginAmerica I didn't today… Must mean I n…         NaN

               tweet_created tweet_location               user_timezone
0  2015-02-24 11:35:52 -0800            NaN  Eastern Time (US & Canada)
1  2015-02-24 11:15:59 -0800            NaN  Pacific Time (US & Canada)
2  2015-02-24 11:15:48 -0800      Lets Play  Central Time (US & Canada)
```

[18]:
```
dataset.airline_sentiment.value_counts()
```

[18]:
```
airline_sentiment
negative    9178
neutral     3099
positive    2363
Name: count, dtype: int64
```

Data imbalance, more negative and neutral sentiment than positive.

[19]:
```
list_of_airlines =dataset.airline.unique().tolist()
print('list of airlines: ',list_of_airlines)
```

```
list of airlines:  ['Virgin America', 'United', 'Southwest', 'Delta', 'US
Airways', 'American']
```

[20]:
```
print("Time of first tweet in the dataset:",dataset.tweet_created.min())
print('-'*65)
print("Time of last tweet in the dataset:",dataset.tweet_created.max())
```

```
Time of first tweet in the dataset: 2015-02-16 23:36:05 -0800
-----------------------------------------------------------------
Time of last tweet in the dataset: 2015-02-24 11:53:37 -0800
```

[21]:
```
print("Airlines with tweet count\n", dataset.airline.value_counts())
```

```
Airlines with tweet count
 airline
United            3822
```

```
US Airways        2913
American          2759
Southwest         2420
Delta             2222
Virgin America     504
Name: count, dtype: int64
```

[22]:
```
print("Tweets frequencies grouped by sentiments for the airlines")
print('-'*60)
airlines_sentiments_groups = dataset.groupby("airline",␣
 ↪group_keys=True)[['airline_sentiment']].value_counts()
airlines_sentiments_groups
```

```
Tweets frequencies grouped by sentiments for the airlines
------------------------------------------------------------
```

[22]:
```
airline         airline_sentiment
American        negative             1960
                neutral               463
                positive              336
Delta           negative              955
                neutral               723
                positive              544
Southwest       negative             1186
                neutral               664
                positive              570
US Airways      negative             2263
                neutral               381
                positive              269
United          negative             2633
                neutral               697
                positive              492
Virgin America  negative              181
                neutral               171
                positive              152
Name: count, dtype: int64
```

[23]:
```
print("Maximum tweet confidence",dataset.airline_sentiment_confidence.max())
print('-'*30)
print("Minimum tweet confidence",dataset.airline_sentiment_confidence.min())
```

```
Maximum tweet confidence 1.0
------------------------------
Minimum tweet confidence 0.335
```

[24]:
```
print(dataset.airline_sentiment_confidence.quantile([0,0.25,0.50,0.75,1]))
```

```
0.00    0.3350
0.25    0.6923
```

```
0.50    1.0000
0.75    1.0000
1.00    1.0000
Name: airline_sentiment_confidence, dtype: float64
```

####25 percent quantile for sentiment confidence is 0.65, which means that 25 percent of the dataset values for this measure is less than 0.65.

[25]:
```
dataset['negativereason'] =  dataset['negativereason'].fillna('N/A')
```

[26]:
```
dataset[dataset['negativereason'] != 'N/A'].groupby("airline",␣
↪group_keys=True)[['negativereason']].value_counts()
```

[26]:
```
airline         negativereason
American        Customer Service Issue         768
                Late Flight                    249
                Cancelled Flight               246
                Can't Tell                     198
                Lost Luggage                   149
                Flight Booking Problems        130
                Bad Flight                      87
                Flight Attendant Complaints     87
                longlines                       34
                Damaged Luggage                 12
Delta           Late Flight                    269
                Customer Service Issue         199
                Can't Tell                     186
                Bad Flight                      64
                Flight Attendant Complaints     60
                Lost Luggage                    57
                Cancelled Flight                51
                Flight Booking Problems         44
                longlines                       14
                Damaged Luggage                 11
Southwest       Customer Service Issue         391
                Cancelled Flight               162
                Can't Tell                     159
                Late Flight                    152
                Bad Flight                      90
                Lost Luggage                    90
                Flight Booking Problems         61
                Flight Attendant Complaints     38
                longlines                       29
                Damaged Luggage                 14
US Airways      Customer Service Issue         811
                Late Flight                    453
                Can't Tell                     246
                Cancelled Flight               189
```

```
                Lost Luggage                   154
                Flight Attendant Complaints    123
                Flight Booking Problems        122
                Bad Flight                     104
                longlines                       50
                Damaged Luggage                 11
United          Customer Service Issue         681
                Late Flight                    525
                Can't Tell                     379
                Lost Luggage                   269
                Bad Flight                     216
                Cancelled Flight               181
                Flight Attendant Complaints    168
                Flight Booking Problems        144
                longlines                       48
                Damaged Luggage                 22
Virgin America  Customer Service Issue          60
                Flight Booking Problems         28
                Can't Tell                      22
                Bad Flight                      19
                Cancelled Flight                18
                Late Flight                     17
                Flight Attendant Complaints      5
                Lost Luggage                     5
                Damaged Luggage                  4
                longlines                        3
Name: count, dtype: int64
```

Highest retweet

[27]:
```
dataset[['text','airline','name','airline_sentiment']].
  ↪loc[dataset['retweet_count'].max()]
```

[27]:
```
text                     @VirginAmerica are flights leaving Dallas for …
airline                                                    Virgin America
name                                                            papamurat
airline_sentiment                                                 neutral
Name: 44, dtype: object
```

[27]:
```
```

1.1 *Data visualization (Multivariate analysis).*

[28]:
```
target_classes,tweet_freq = np.unique(dataset.
  ↪airline_sentiment,return_counts=True)
print(target_classes,tweet_freq)
```

```
['negative' 'neutral' 'positive'] [9178 3099 2363]
```

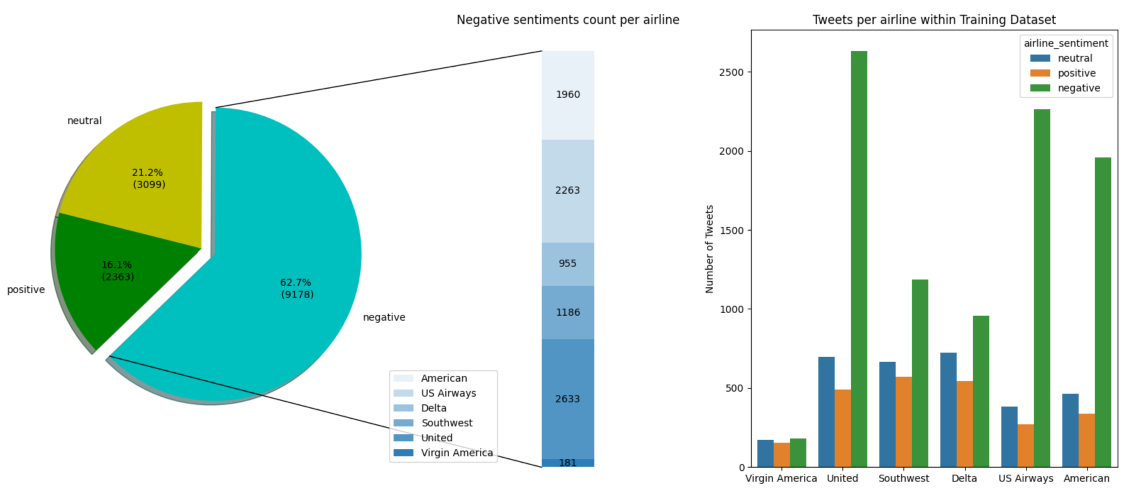
[29]:

```python
def func(pct, allvals)->str:
    absolute = int(np.round(pct/100.*np.sum(allvals)))
    return f"{pct:.1f}%\n ({absolute:d})"
```

[30]:

```python
# make figure and assign axis objects
fig, (ax1,ax2,ax3) = plt.subplots(1, 3, figsize=(20, 8))
fig.subplots_adjust(wspace=0)

# pie chart parameters
overall_ratios = tweet_freq
labels = target_classes
explode = [0.1, 0, 0]
colors =['c','y','g']
# rotate so that first wedge is split by the x-axis
angle = -272 * overall_ratios[2]
wedges, *_ = ax1.pie(overall_ratios, autopct=lambda pct: func(pct,␣
 ↪overall_ratios),shadow=True,labels=labels,␣
 ↪explode=explode,colors=colors,startangle=angle)

# bar chart parameters
negative_sentiment_ratios =[airlines_sentiments_groups[x]['negative'] for x in␣
 ↪list_of_airlines]
bottom = 5
width = .2

# Adding from the top matches the legend.
for j, (height, label) in enumerate(reversed([*zip(negative_sentiment_ratios,␣
 ↪list_of_airlines)])):
    bottom -= height
    bc = ax2.bar(0, height, width, bottom=bottom, color='C0',␣
 ↪label=label,alpha=0.1 + 0.17 * j)
    ax2.bar_label(bc, labels=[height], label_type='center')

ax2.legend(loc=3)
ax2.set_title('Negative sentiments count per airline')
ax2.axis('off')
ax2.set_xlim(- 3.5 * width, 3.5 * width)

# use ConnectionPatch to draw lines between the two plots
theta1, theta2 = wedges[0].theta1, wedges[0].theta2
center, r = wedges[0].center, wedges[0].r
bar_height = sum(negative_sentiment_ratios)

# draw top connecting line
x = r * np.cos(np.pi / 180 * theta2) + center[0]
y = r * np.sin(np.pi / 180 * theta2) + center[1]
con = ConnectionPatch(xyA=(-width / 2, 0), coordsA=ax2.transData,
```

```
                      xyB=(x, y), coordsB=ax1.transData)
con.set_color([0, 0, 0])
con.set_linewidth(1)
ax2.add_artist(con)

# draw bottom connecting line
x = r * np.cos(np.pi / 180 * theta1) + center[0]
y = r * np.sin(np.pi / 180 * theta1) + center[1]
con = ConnectionPatch(xyA=(-width / 2, -bar_height), coordsA=ax2.
 ↪transData,xyB=(x, y), coordsB=ax1.transData)
con.set_color([0, 0, 0])
ax2.add_artist(con)
con.set_linewidth(1)

ax3 = sns.countplot(x="airline", hue="airline_sentiment", data=dataset)
ax3.set_xlabel('')
ax3.set_ylabel('Number of Tweets')
ax3.set_title('Tweets per airline within Training Dataset')


plt.show()
del ax1,ax2,ax3
```



Date decomposition (feature creation).

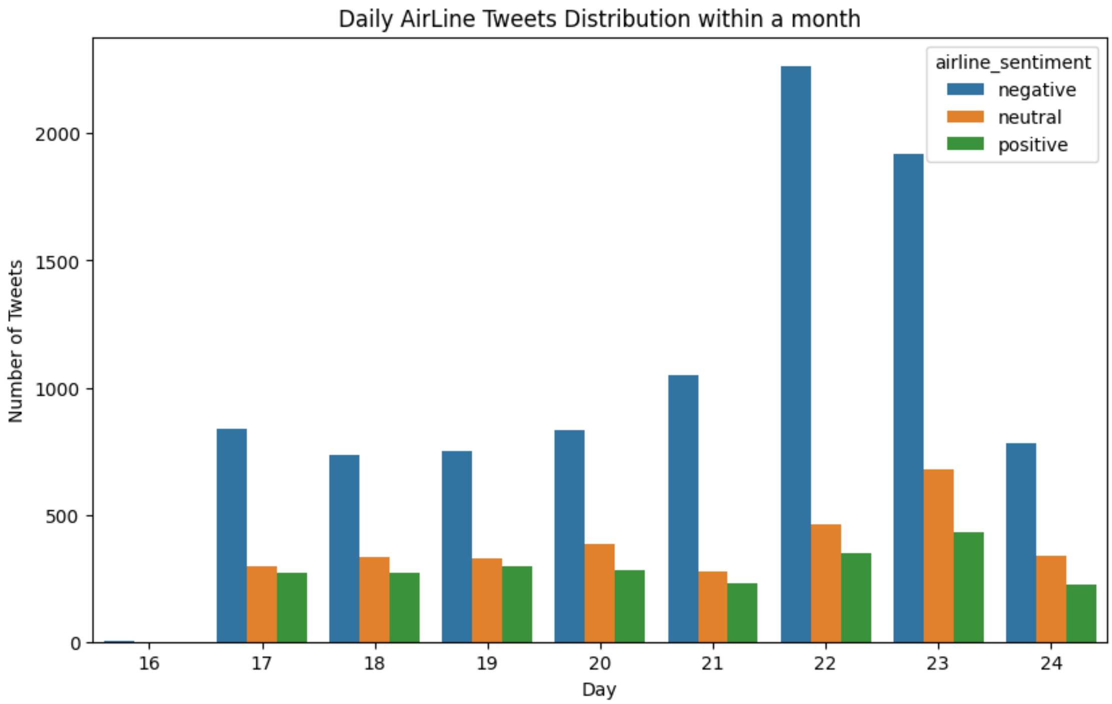
[31]:
```
def decode_date(date_str)-> pd.Series:
  aux = date_str.replace(' -0800','')
  date_utc = datetime.strptime(aux,"%Y-%m-%d %H:%M:%S")
  return pd.Series([date_utc.weekday(),date_utc.day,date_utc.hour])
```

[32]:
```
dataset[['week_day','day','hour']] = dataset.pop('tweet_created').
↪apply(decode_date)
```

[33]:
```
dataset.head(3)
```

[33]:
```
             tweet_id airline_sentiment  airline_sentiment_confidence  \
0  570306133677760513           neutral                        1.0000
1  570301130888122368          positive                        0.3486
2  570301083672813571           neutral                        0.6837

  negativereason  negativereason_confidence         airline  \
0            N/A                        NaN  Virgin America
1            N/A                        0.0  Virgin America
2            N/A                        NaN  Virgin America

  airline_sentiment_gold        name negativereason_gold  retweet_count  \
0                    NaN     cairdin                 NaN              0
1                    NaN    jnardino                 NaN              0
2                    NaN  yvonnalynn                 NaN              0

                                                text tweet_coord  \
0                @VirginAmerica What @dhepburn said.         NaN
1  @VirginAmerica plus you've added commercials t…         NaN
2  @VirginAmerica I didn't today… Must mean I n…         NaN

  tweet_location               user_timezone  week_day  day  hour
0            NaN  Eastern Time (US & Canada)         1   24    11
1            NaN  Pacific Time (US & Canada)         1   24    11
2      Lets Play  Central Time (US & Canada)         1   24    11
```

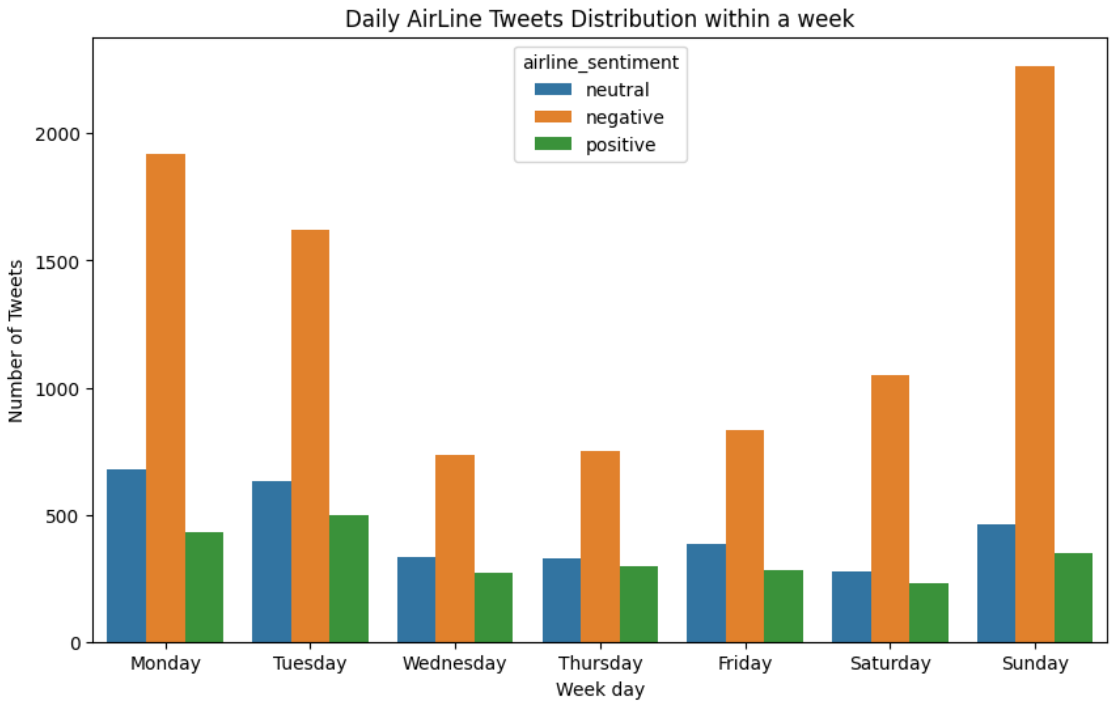
[34]:
```
fig, axes = plt.subplots(figsize=(10,6))
ax = sns.countplot(x="day", hue="airline_sentiment", data=dataset)
ax.set_xlabel('Day')
ax.set_ylabel('Number of Tweets')
ax.set_title('Daily AirLine Tweets Distribution within a month')
del ax,fig,axes
```



[35]:

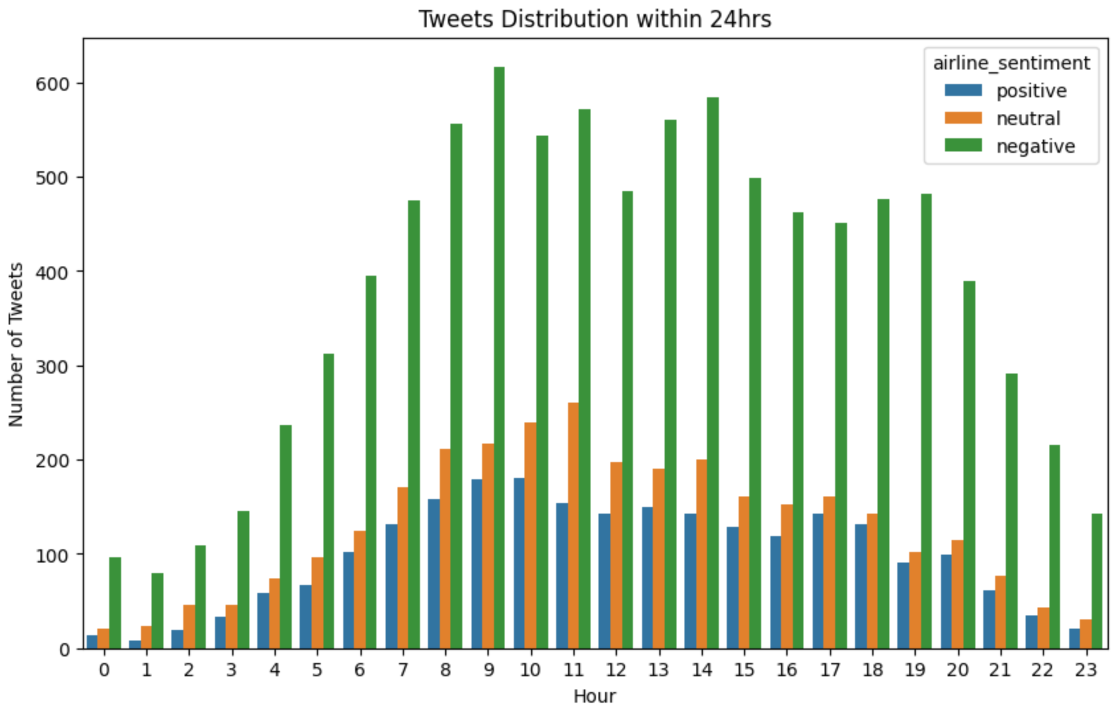
```
fig, axes = plt.subplots(figsize=(10,6))
ax = sns.countplot(x="week_day", hue="airline_sentiment", data=dataset)
ax.set_xlabel('Week day')
ax.
 ↪set(xticklabels=['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunday'])
ax.set_ylabel('Number of Tweets')
ax.set_title('Daily AirLine Tweets Distribution within a week')
del ax,fig,axes
```

```
<ipython-input-35-64147deda210>:4: UserWarning: set_ticklabels() should only be
used with a fixed number of ticks, i.e. after set_ticks() or using a
FixedLocator.
  ax.set(xticklabels=['Monday','Tuesday','Wednesday','Thursday','Friday','Saturd
ay','Sunday'])
```



[36]:

```
fig, axes = plt.subplots(figsize=(10,6))
ax = sns.countplot(x="hour", hue="airline_sentiment", data=dataset)
ax.set_xlabel('Hour')
ax.set_ylabel('Number of Tweets')
ax.set_title('Tweets Distribution within 24hrs')
del ax,fig,axes
```



1.1.1 *Top 10 tweet authors*

[37]:
```
top_tweeter = Counter([name for name in dataset['name']])
top_tweeter_df = pd.DataFrame(top_tweeter.most_common(10))
top_tweeter_df.columns = ['top_tweet_authors','count']
print("Top 10 Authors",'-'*15)
top_tweeter_df.style.background_gradient(cmap='inferno')
```

```
Top 10 Authors ---------------
```

[37]: <pandas.io.formats.style.Styler at 0x7824f30f3c10>

1.1.2 *Top reasons for complaints.*

[38]:
```
top_complain = Counter([reason for reason in dataset[dataset['negativereason'] !
 ↪= 'N/A']['negativereason']])
top_complain_df = pd.DataFrame(top_complain.most_common(10))
top_complain_df.columns = ['top_complain_reason','count']
print("Top 10 complain",'-'*15)
top_complain_df.style.background_gradient(cmap='inferno')
```

```
Top 10 complain ---------------
```

[38]: <pandas.io.formats.style.Styler at 0x7824f3144c70>

A closer look at the two top authors

[39]:
```
dataset[dataset['name'] ==␣
↪'JetBlueNews'][['airline','negativereason','airline_sentiment','retweet_count']].
↪value_counts()
```

[39]:
```
airline         negativereason  airline_sentiment  retweet_count
Delta           N/A             neutral            0                56
                                positive           0                 5
                Can't Tell      negative           0                 1
Virgin America  N/A             neutral            0                 1
Name: count, dtype: int64
```

[40]:
```
dataset[dataset['name'] ==␣
↪'kbosspotter'][['airline','negativereason','airline_sentiment','retweet_count']].
↪value_counts()
```

[40]:
```
airline  negativereason          airline_sentiment  retweet_count
Delta    N/A                     neutral            0                22
                                 positive           0                 6
         Cancelled Flight        negative           0                 1
         Customer Service Issue  negative           0                 1
         Flight Booking Problems negative           0                 1
         Late Flight             negative           0                 1
Name: count, dtype: int64
```

1.1.3 *Data preprocessing, feature engineering.*

[41]:
```
#For removing user tags(@user)
def remove_user_tags(tweet:str)->tuple[str,int]:
    user = re.compile(r'@\S+')
    initial = len(tweet)
    text = user.sub(r'',tweet)
    result = ''.join([i for i in text if not i.isdigit()])
    final = len(result)
    number = 0
    for i in text:
      if i.isdigit():
        number+=1
    if( final == initial - number):
      return (result,0)
    else:
      return (result,1)

#For removing Url links
def remove_url(tweet:str)->tuple[str,int]:
    url = re.compile(r'https?://\S+|www\.\S+')
    initial = len(tweet)
```

```python
    text = url.sub(r'',tweet,re.IGNORECASE)
    result = ''.join([i for i in text if not i.isdigit()])
    final = len(result)
    number = 0
    for i in text:
      if i.isdigit():
        number += 1
    if( final == initial - number):
      return (result,0)
    else:
      return (result,1)
```

[42]:
```python
def clean_and_create_feature(tweet:str)-> pd.Series:
    (str_without_tags,tag_flag) = remove_user_tags(tweet)
    (str_without_links,url_flag) = remove_url(str_without_tags)
    return pd.Series([str_without_links,tag_flag,url_flag])
```

[43]:
```python
dataset[['text','user_tag','url_flag']] = dataset.apply(lambda x:␣
 ↪clean_and_create_feature(x.text),axis=1)
```

[44]:
```python
dataset.head(3)
```

[44]:
```
             tweet_id airline_sentiment  airline_sentiment_confidence  \
0  570306133677760513           neutral                        1.0000
1  570301130888122368          positive                        0.3486
2  570301083672813571           neutral                        0.6837

  negativereason  negativereason_confidence         airline  \
0            N/A                        NaN  Virgin America
1            N/A                        0.0  Virgin America
2            N/A                        NaN  Virgin America

  airline_sentiment_gold        name negativereason_gold  retweet_count  \
0                    NaN     cairdin                 NaN              0
1                    NaN    jnardino                 NaN              0
2                    NaN  yvonnalynn                 NaN              0

                                                text tweet_coord  \
0                                        What  said.         NaN
1   plus you've added commercials to the experien…         NaN
2   I didn't today… Must mean I need to take an…         NaN

  tweet_location               user_timezone  week_day  day  hour  user_tag  \
0            NaN  Eastern Time (US & Canada)         1   24    11         1
1            NaN  Pacific Time (US & Canada)         1   24    11         1
2      Lets Play  Central Time (US & Canada)         1   24    11         1
```

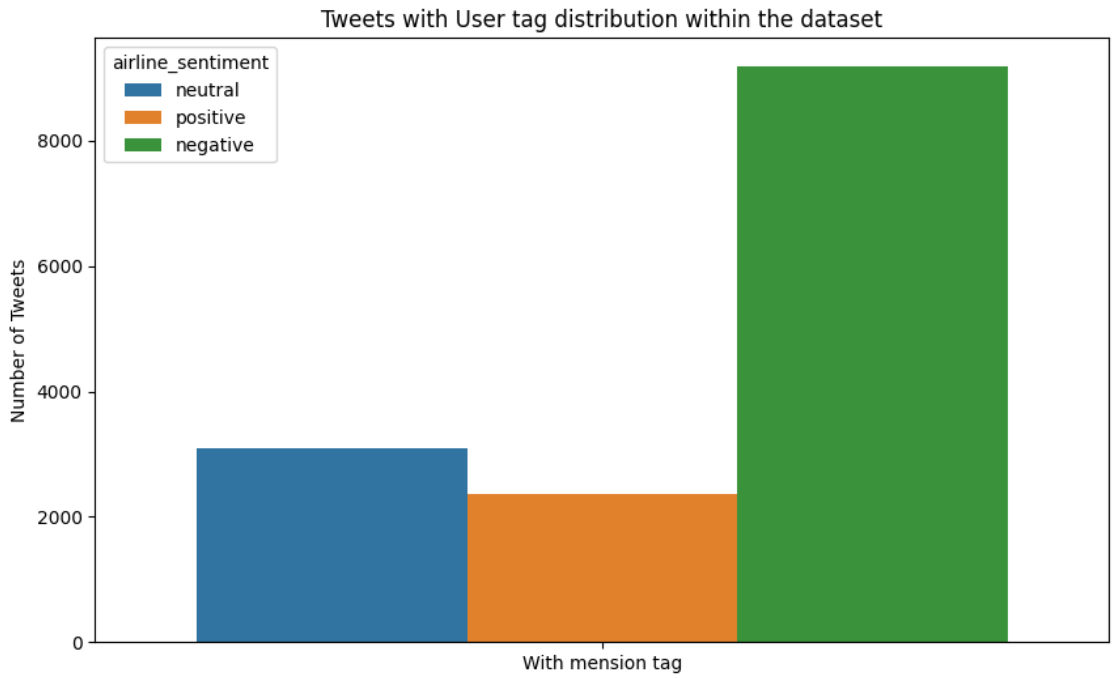
```
   url_flag
0         0
1         0
2         0
```

[45]:
```
print(dataset.groupby("airline_sentiment", group_keys=True)[['user_tag']].
 ↪value_counts())
print('-'*35)
print(dataset.groupby("airline_sentiment", group_keys=True)[['url_flag']].
 ↪value_counts())
```

```
airline_sentiment  user_tag
negative           1           9178
neutral            1           3099
positive           1           2363
Name: count, dtype: int64
-----------------------------------
airline_sentiment  url_flag
negative           0           8730
                   1            448
neutral            0           2603
                   1            496
positive           0           2134
                   1            229
Name: count, dtype: int64
```

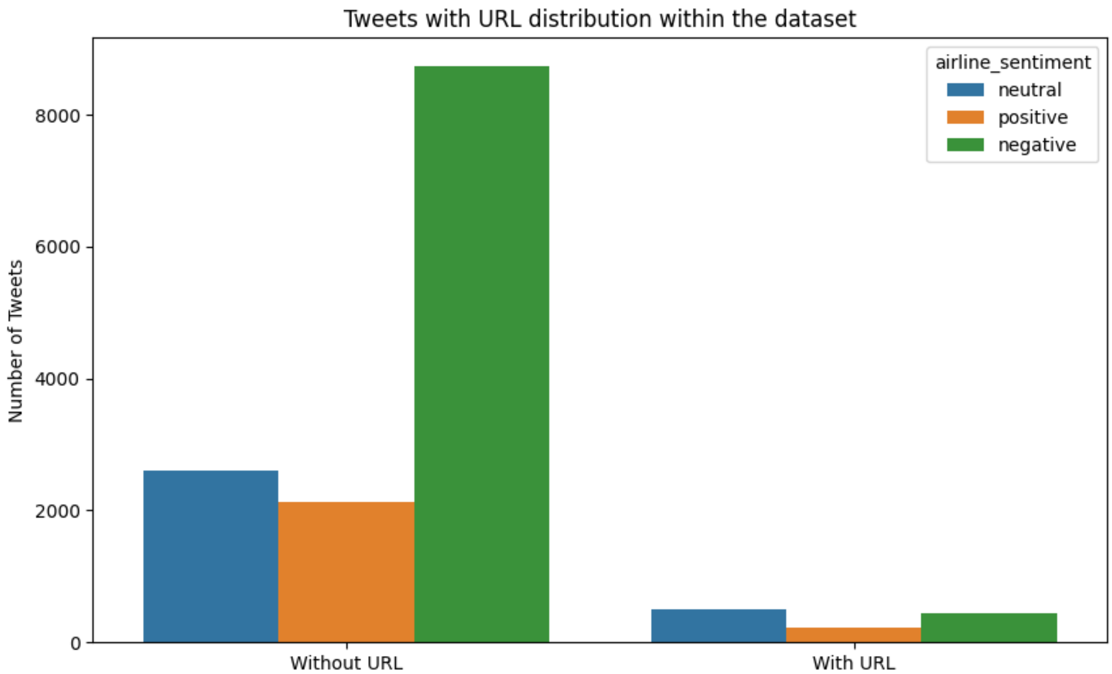
[46]:
```
f, axes = plt.subplots(figsize=(10,6))
ax = sns.countplot(x="user_tag", hue="airline_sentiment", data=dataset)
ax.set(xticklabels=['With mension tag'])
ax.set_xlabel('')
ax.set_ylabel('Number of Tweets')
ax.set_title('Tweets with User tag distribution within the dataset')
del ax,f,axes
```

```
<ipython-input-46-05fe470843a3>:3: UserWarning: set_ticklabels() should only be
used with a fixed number of ticks, i.e. after set_ticks() or using a
FixedLocator.
  ax.set(xticklabels=['With mension tag'])
```



[47]:
```
f, axes = plt.subplots(figsize=(10, 6))
ax = sns.countplot(x="url_flag",hue="airline_sentiment", data=dataset)
ax.set(xticklabels=['Without URL','With URL'])
ax.set_xlabel('')
ax.set_ylabel('Number of Tweets')
ax.set_title('Tweets with URL distribution within the dataset')
del ax,f,axes
```

```
<ipython-input-47-5d468f3940bb>:3: UserWarning: set_ticklabels() should only be
used with a fixed number of ticks, i.e. after set_ticks() or using a
FixedLocator.
  ax.set(xticklabels=['Without URL','With URL'])
```



[48]:
```python
def happy_emoticons_removal(tweet:str) ->tuple[str,int]:
    happy = re.compile(r"([xX;:]-?[dDpP)])")
    initial = len(tweet)
    text = happy.sub(r'',tweet)
    result = ''.join([i for i in text if not i.isdigit()])
    final = len(result)
    number = 0
    for i in text:
      if i.isdigit():
        number += 1
    if( final != initial - number):
      return (result,1)
    return (result,0)

def sad_emoticons_removal(tweet:str) ->tuple[str,int]:
    sad = re.compile(r"[:;](['\"]?[-~]?[/(\|C<>{}\[]+)")
    initial = len(tweet)
    text = sad.sub(r'',tweet)
    result = ''.join([i for i in text if not i.isdigit()])
    final = len(result)
    number = 0
    for i in text:
      if i.isdigit():
        number += 1
    if( final != initial - number):
```

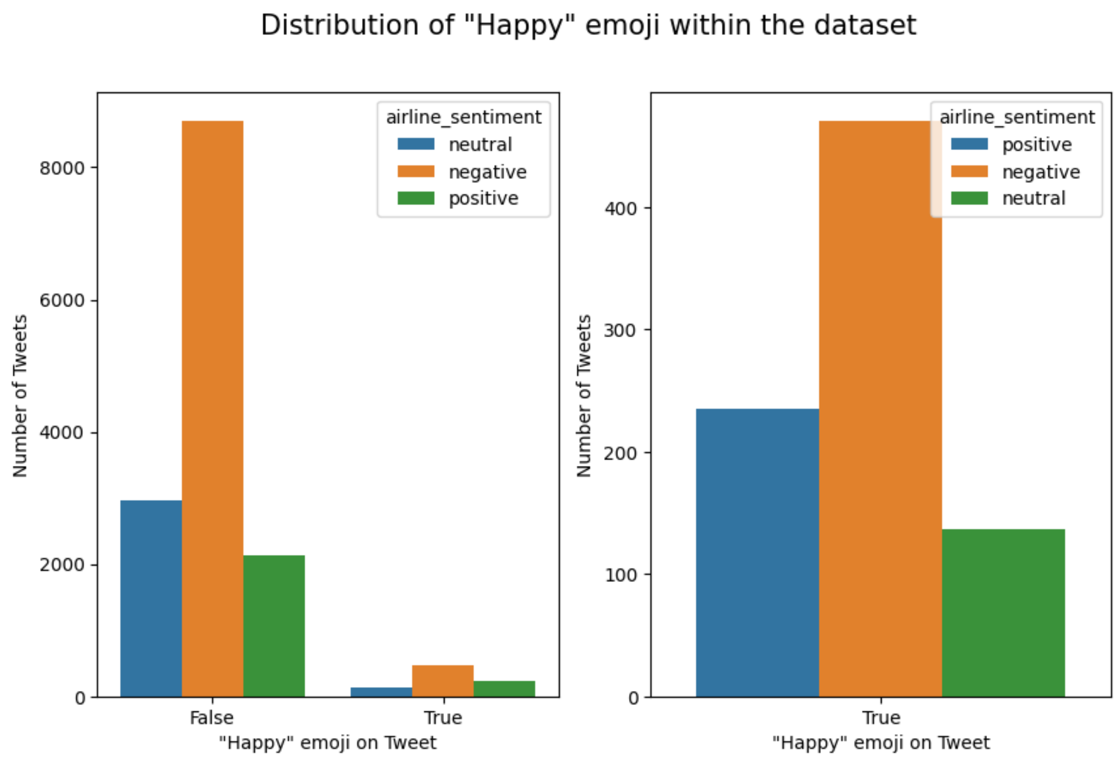
```
        return (result,1)
    return (result,0)
```

[49]:
```
def emoticon_removal_and_feature_creation(tweet:str)-> pd.Series:
    (str_without_happy_emoji,happy_emoji_flag) = happy_emoticons_removal(tweet)
    (str_without_emoji,sad_emoji_flag) =␣
 ↪sad_emoticons_removal(str_without_happy_emoji)
    return pd.Series([str_without_emoji,happy_emoji_flag,sad_emoji_flag])
```

[50]:
```
dataset[['text','happy_emoji','sad_emoji']] = dataset.apply(lambda x:␣
 ↪emoticon_removal_and_feature_creation(x.text),axis=1)
```

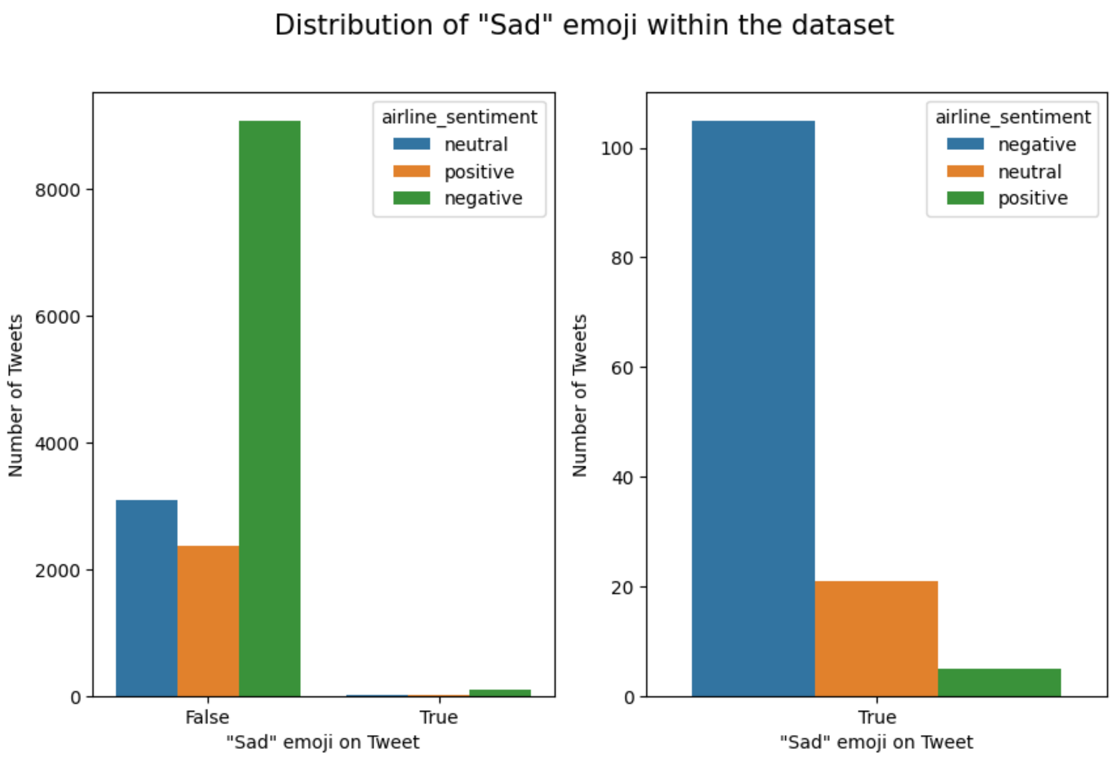
[51]:
```
f, axes = plt.subplots(1,2,figsize=(10, 6))
ax = sns.countplot(x="happy_emoji", hue="airline_sentiment",␣
 ↪data=dataset,ax=axes[0])
ax.set(xticklabels=['False','True'])
ax.set_xlabel('"Happy" emoji on Tweet')
ax.set_ylabel('Number of Tweets')
ax = sns.countplot(x="happy_emoji", hue="airline_sentiment",␣
 ↪data=dataset[dataset.happy_emoji==1],ax=axes[1])
ax.set(xticklabels=['True'])
ax.set_xlabel('"Happy" emoji on Tweet')
ax.set_ylabel('Number of Tweets')
f.suptitle('Distribution of "Happy" emoji within the dataset',fontsize=15)
del ax,f,axes
```

```
<ipython-input-51-91fc4191489b>:3: UserWarning: set_ticklabels() should only be
used with a fixed number of ticks, i.e. after set_ticks() or using a
FixedLocator.
  ax.set(xticklabels=['False','True'])
<ipython-input-51-91fc4191489b>:7: UserWarning: set_ticklabels() should only be
used with a fixed number of ticks, i.e. after set_ticks() or using a
FixedLocator.
  ax.set(xticklabels=['True'])
```



[52]:
```
f, axes = plt.subplots(1,2,figsize=(10, 6))
ax = sns.countplot(x="sad_emoji", hue="airline_sentiment",␣
↪data=dataset,ax=axes[0])
ax.set(xticklabels=['False','True'])
ax.set_xlabel('"Sad" emoji on Tweet')
ax.set_ylabel('Number of Tweets')
ax = sns.countplot(x="sad_emoji", hue="airline_sentiment", data=dataset[dataset.
↪sad_emoji==1],ax=axes[1])
ax.set(xticklabels=['True'])
ax.set_xlabel('"Sad" emoji on Tweet')
ax.set_ylabel('Number of Tweets')
f.suptitle('Distribution of "Sad" emoji within the dataset',fontsize=15)
del ax,f,axes
```

```
<ipython-input-52-52f20b644c5b>:3: UserWarning: set_ticklabels() should only be
used with a fixed number of ticks, i.e. after set_ticks() or using a
FixedLocator.
  ax.set(xticklabels=['False','True'])
<ipython-input-52-52f20b644c5b>:7: UserWarning: set_ticklabels() should only be
used with a fixed number of ticks, i.e. after set_ticks() or using a
FixedLocator.
  ax.set(xticklabels=['True'])
```



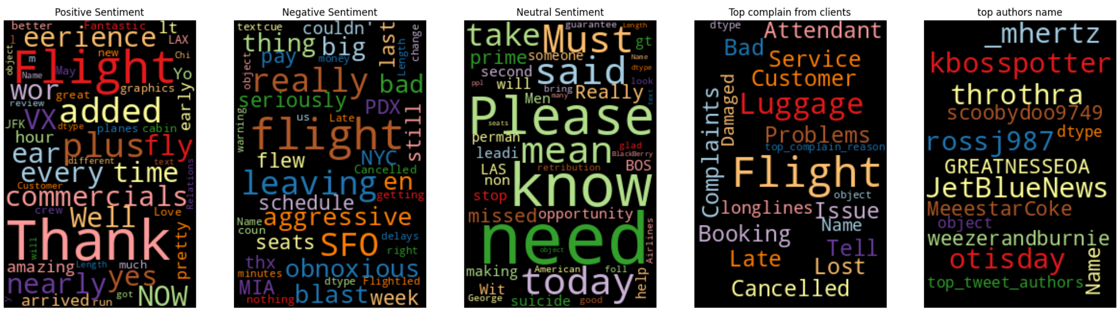
[53]:
```python
print(dataset.groupby("airline_sentiment", group_keys=True)[['happy_emoji']].
    value_counts())
print('-'*40)
print(dataset.groupby("airline_sentiment", group_keys=True)[['sad_emoji']].
    value_counts())
```

```
airline_sentiment  happy_emoji
negative           0              8707
                   1               471
neutral            0              2962
                   1               137
positive           0              2128
                   1               235
Name: count, dtype: int64
----------------------------------------
airline_sentiment  sad_emoji
negative           0            9073
                   1             105
neutral            0            3078
                   1              21
positive           0            2358
                   1               5
Name: count, dtype: int64
```

[54]:
```
positive_sentiment = dataset[dataset.airline_sentiment == "positive"]
positive_text=positive_sentiment['text']
negative_sentiment = dataset[dataset.airline_sentiment == 'negative']
negative_text=negative_sentiment['text']
neutral_sentiment = dataset[dataset.airline_sentiment == 'neutral']
neutral_text=neutral_sentiment['text']
complain_text = top_complain_df['top_complain_reason']
top_authors = top_tweeter_df["top_tweet_authors"]
```

[55]:
```
# Create and generate a word
fig, ax = plt.subplots(1, 5, figsize=(25, 8),edgecolor = 'k')
positive_tweet = WordCloud(width = 200,height =␣
 ↪300,colormap="Paired",background_color = 'black',max_words = 90,stopwords =␣
 ↪STOPWORDS).generate(str(positive_text))
negative_tweet = WordCloud(width = 200,height =␣
 ↪300,colormap="Paired",background_color = 'black',max_words = 90,stopwords =␣
 ↪STOPWORDS).generate(str(negative_text))
neutral_tweet = WordCloud(width = 200,height =␣
 ↪300,colormap="Paired",background_color = 'black',max_words = 90,stopwords =␣
 ↪STOPWORDS).generate(str(neutral_text))
complain_tweet = WordCloud(width = 200,height =␣
 ↪300,colormap="Paired",background_color = 'black',max_words = 90,stopwords =␣
 ↪STOPWORDS).generate(str(complain_text))
top_authors_tweet = WordCloud(width = 200,height =␣
 ↪300,colormap="Paired",background_color = 'black',max_words = 90,stopwords =␣
 ↪STOPWORDS).generate(str(top_authors))

ax[0].imshow(positive_tweet)
ax[0].axis('off')
ax[0].set_title('Positive Sentiment')

ax[1].imshow(negative_tweet)
ax[1].axis('off')
ax[1].set_title('Negative Sentiment')

ax[2].imshow(neutral_tweet)
ax[2].axis('off')
ax[2].set_title('Neutral Sentiment')



ax[3].imshow(complain_tweet)
ax[3].axis('off')
ax[3].set_title('Top complain from clients')


ax[4].imshow(top_authors_tweet)
```

```
ax[4].axis('off')
ax[4].set_title('top authors name')

plt.show()
del␣
↪top_tweeter_df,top_complain_df,fig,ax,positive_tweet,negative_tweet,neutral_tweet,complain_
```




```
[56]: def stop_word_and_stemming(tweet:str)->str:
          tknzr = TweetTokenizer(preserve_case=False,strip_handles=True,␣
      ↪reduce_len=True)
          myStemmer = PorterStemmer()
          tweet = tknzr.tokenize(tweet)
          stop_words = set(stopwords.words('english'))
          new_list = [myStemmer.stem(word) for word in tweet if word not in␣
      ↪stop_words]
          tweet = ' '.join(new_list)
          return tweet

      #For removing punctuation
      def remove_punctuations(text:str):
          for punctuation in string.punctuation:
              text = text.replace(punctuation, '')
          return text
```

```
[57]: dataset['text'] = dataset['text'].apply(stop_word_and_stemming)
```

```
[58]: dataset['text'] = dataset['text'].apply(remove_punctuations)
```

```
[59]: airline_dict = dict(zip(list_of_airlines,range(len(list_of_airlines))))
      dataset['airline_code'] = dataset.pop('airline').map(airline_dict)
      print(airline_dict)
      del airline_dict
```

```
{'Virgin America': 0, 'United': 1, 'Southwest': 2, 'Delta': 3, 'US Airways': 4,
'American': 5}
```

1.2 Drop less beneficial columns

[60]:
```
dataset.
↪drop(['user_tag','tweet_id','name','negativereason','negativereason_confidence','airline_se
```

[61]:
```
print("Sum of missing values",sum(dataset.isnull().sum()))
```

```
Sum of missing values 0
```

1.2.1 y_i can take values between 0 to $N-1$ categorories

[62]:
```
target_dict = {'positive':1,'negative': 0,'neutral': 2}
print(target_dict)
dataset['target'] = dataset['airline_sentiment'].map(target_dict)
```

```
{'positive': 1, 'negative': 0, 'neutral': 2}
```

[63]:
```
dataset.head(3)
```

[63]:
```
  airline_sentiment  airline_sentiment_confidence  retweet_count  \
0           neutral                        1.0000              0
1          positive                        0.3486              0
2           neutral                        0.6837              0

                                      text  week_day  day  hour  url_flag  \
0                                     said         1   24    11         0
1         plu ad commerci eerienc  tacki         1   24    11         0
2   today  must mean need take anoth trip         1   24    11         0

   happy_emoji  sad_emoji  airline_code  target
0            0          0             0       2
1            1          0             0       1
2            0          0             0       2
```

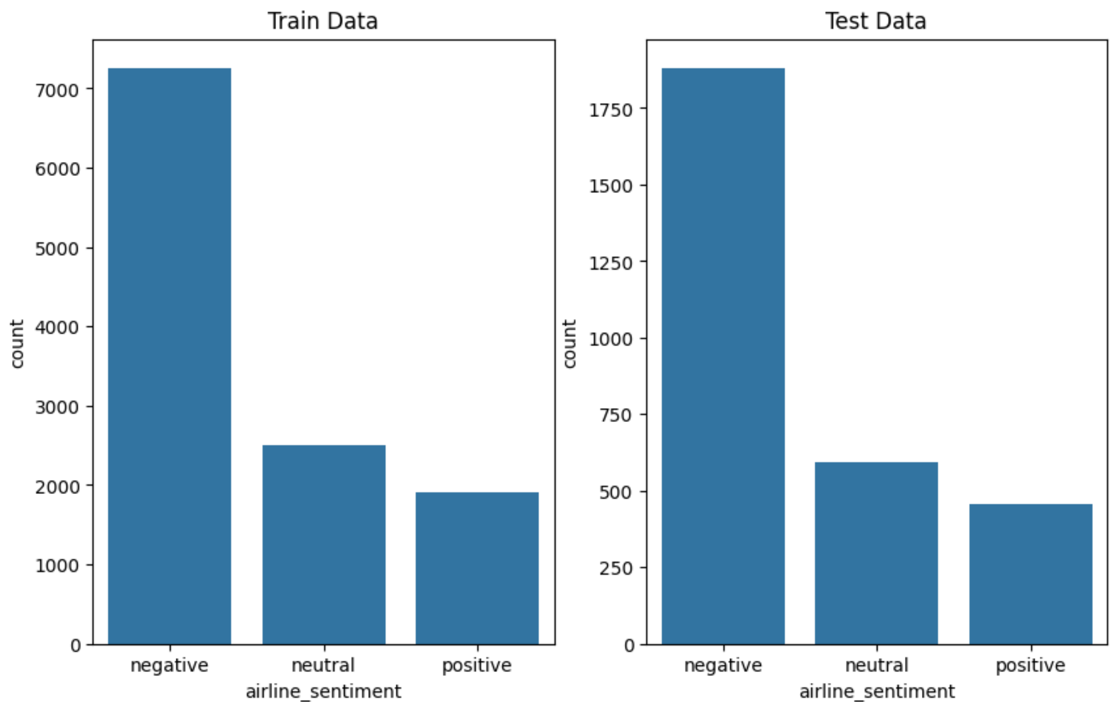
[64]:
```
base_line_df = dataset.copy()
```

[65]:
```
def split_dataset(df:pd.DataFrame,test_percentage:float)-> tuple[pd.
↪DataFrame,pd.DataFrame]:
  shuffle = np.random.permutation(len(df))
  test_size = int(len(df) * test_percentage)
  test_aux = shuffle[:test_size]
  train_aux = shuffle[test_size:]
  return (df.iloc[train_aux],df.iloc[test_aux])
```

[66]:
```
train, test = split_dataset(base_line_df,0.2)
```

[67]:
```
train.drop_duplicates(inplace=True)
```

[68]:
```
fig, ax = plt.subplots(1, 2, figsize=(10, 6))
sns.countplot(x='airline_sentiment', data=train, ax=ax[0])
sns.countplot(x='airline_sentiment', data=test, ax=ax[1])
ax[0].set_title('Train Data')
ax[1].set_title('Test Data')
plt.show()
del ax,fig
```



[69]:
```
print(train.shape,test.shape)
```

```
(11659, 12) (2928, 12)
```

[70]:
```
print(train[['target','airline_sentiment']].value_counts())
print("test samples",'-'*20)
print(test[['target','airline_sentiment']].value_counts())
```

```
target  airline_sentiment
0       negative             7260
2       neutral              2498
1       positive             1901
Name: count, dtype: int64
test samples --------------------
target  airline_sentiment
0       negative             1882
2       neutral               592
```

```
1       positive               454
Name: count, dtype: int64
```

[71]:
```python
train.head(3)
```

[71]:
```
       airline_sentiment  airline_sentiment_confidence  retweet_count  \
4219            negative                        1.0000              0
13090           negative                        0.6326              0
9033            negative                        1.0000              0

                                                 text  week_day  day  hour  \
4219   flight alreadi cancel flightl  tri get home tw…         1   17    11
13090  realli appreci great custom servic  one servic…         0   23    11
9033   spoke someon told breast pump medic equip  pla…         1   24     9

       url_flag  happy_emoji  sad_emoji  airline_code  target
4219          0            0          0             1       0
13090         0            0          0             5       0
9033          0            0          0             4       0
```

1.2.2 Baseline Model Implementation

[72]:
```python
y = train['target'].to_numpy()
```

[73]:
```python
y_t = test['target'].to_numpy()
```

1.2.3 *Bag of words (BoW).*

[74]:
```python
vec = TfidfVectorizer(ngram_range=(1,2),stop_words='english').fit(train['text'])
```

[75]:
```python
X = vec.transform(train['text'])
```

[76]:
```python
X_t =vec.transform(test['text'])
```

[77]:
```python
vocab_size = len(vec.vocabulary_) + 1
print("Vocabulary Size :", vocab_size)
```

```
Vocabulary Size : 61691
```

Linear support vector machine (With class weight to compensate for the imbalance).

[78]:
```python
param_grid = {'C': [0.01, 0.1, 1.0, 10.0, 100.0]}
clf = LinearSVC(loss='hinge',class_weight="balanced")
```

[79]:
```python
grids = GridSearchCV(clf, param_grid,verbose=1,n_jobs=-1)
grids = grids.fit(X, y)
print ("Best parameters: %s" % grids.best_params_)
```

```
Fitting 5 folds for each of 5 candidates, totalling 25 fits
Best parameters: {'C': 1.0}
```

[80]:
```
base_line_predictions = grids.predict(X_t)
```

[81]:
```
def score_board_df(y_test,predictions,model_name,averaging_method='macro')-> pd.
↪DataFrame:
  f1 = "%0.3f" % f1_score(y_test, predictions,average=averaging_method)
  prec = "%0.3f" % precision_score(y_test, predictions,average=averaging_method)
  rec ="%0.3f" % recall_score(y_test, predictions,average=averaging_method)
  acc = "%0.3f" % accuracy_score(y_test, predictions)
  return pd.DataFrame({"classifier-name":model_name,"f1_score":[f1],"precision":
↪[prec],"recall":[rec],"accuracy":[acc],"Average-method":averaging_method})
```

[262]:
```
score_board = score_board_df(y_t, base_line_predictions,"LinearSVC +␣
↪Weights","weighted")
```

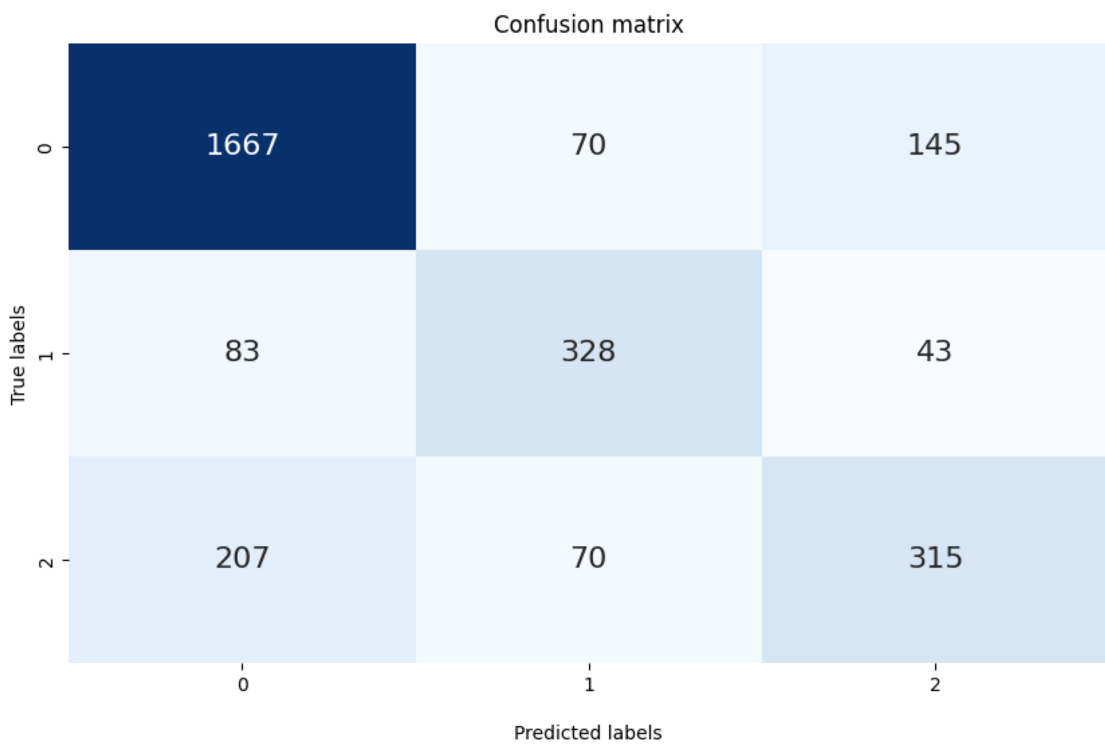
[263]:
```
score_board
```

[263]:

	classifier-name	f1_score	precision	recall	accuracy	Average-method
0	LinearSVC + Weights	0.785	0.783	0.789	0.789	weighted

[264]:
```
def conf_matrix(y_true, y_pred)-> pd.DataFrame:
    # Creating a confusion matrix
    cm = confusion_matrix(y_true, y_pred)
    con_mat = pd.DataFrame(cm, index=np.unique(y_true),columns=np.
↪unique(y_true))
    #Ploting the confusion matrix
    plt.figure(figsize=(10,6))
    ax = sns.heatmap(con_mat, annot=True, annot_kws={"size": 16}, fmt='g',␣
↪cmap=plt.cm.Blues, cbar=False)
    ax.set_title('Confusion matrix')
    ax.set_xlabel('\nPredicted labels')
    ax.set_ylabel('True labels')
    plt.show()
    del ax
    return con_mat
```

[265]:
```
_= conf_matrix(y_t, base_line_predictions)
```



[266]:
```
#print(test[y_t!=base_line_predictions]) #misclassified texts
```

1.2.4 *Word Embedding*

1.2.5 *Model 1 (Word2vec Embedding)*

[267]:
```
def create_word_vector(text):
  tknzr = TweetTokenizer()
  tweet = tknzr.tokenize(text)
  aux = [int(vec.vocabulary_[k]) for k in tweet if k in vec.vocabulary_.keys()]
  return aux
```

[268]:
```
model_1_train = train.copy()
model_1_test = test.copy()
```

[269]:
```
model_1_train['text'] = model_1_train.text.apply(create_word_vector)
model_1_test['text'] = model_1_test.text.apply(create_word_vector)
```

[270]:
```
MAX_LENGTH = len(max(model_1_train.text, key=len))
print(MAX_LENGTH)
```

20

[271]:
```python
# plot the distribution of review lengths
#sns.distplot([len(x) for x in model_1_train.text])
#plt.xlim([0, 256]);
#plt.xlabel('Token count')
```

[272]:
```python
X_train= pad_sequences(model_1_train.text, maxlen=MAX_LENGTH,padding='post')
```

[273]:
```python
X_test = pad_sequences(model_1_test.text, maxlen=MAX_LENGTH,padding='post')
```

[274]:
```python
Y_train = keras.utils.to_categorical(y, num_classes=3)
```

[275]:
```python
Y_test = keras.utils.to_categorical(y_t, num_classes=3)
```

[276]:
```python
Y_train[27:30]
```

[276]:
```
array([[1., 0., 0.],
       [0., 0., 1.],
       [0., 1., 0.]])
```

[277]:
```python
model_1_train.airline_sentiment[27:30]
```

[277]:
```
14026    negative
9324      neutral
11389    positive
Name: airline_sentiment, dtype: object
```

1.2.6 *Model 1 RNN*

[278]:
```python
METRICS = [keras.metrics.CategoricalAccuracy(name='accuracy'),keras.metrics.
 ↪Precision(name='precision'),keras.metrics.Recall(name='recall'),keras.
 ↪metrics.AUC(name='prc', curve='PR')]
embedding_vector_length= 32
```

[279]:
```python
def create_model(vocab_size:int,embedding_vector_length:int,weights:
 ↪list=None,metrics:list=METRICS,trainable:bool=False):
  model = keras.models.Sequential()
  model.add(keras.layers.
 ↪Embedding(vocab_size,embedding_vector_length,weights=weights,trainable=trainable))
  model.add(keras.layers.Bidirectional(keras.layers.
 ↪LSTM(64,return_sequences=True,dropout=0.2, recurrent_dropout=0.2)))
  model.add(keras.layers.GlobalAveragePooling1D())
  model.add(keras.layers.Dense(200, activation='relu'))
  model.add(keras.layers.Dense(3, activation='softmax'))
  model.compile(loss=keras.losses.CategoricalCrossentropy(),optimizer=tf.keras.
 ↪optimizers.Adam(learning_rate=0.005),metrics=METRICS)
  return model
```

[280]:
```
model1 = create_model(vocab_size,embedding_vector_length)
print(model1.summary())
```

Model: "sequential_11"

```
 Layer (type)                         Output Shape                          ␣
↪Param #

 embedding_13 (Embedding)             ?                                   0␣
↪(unbuilt)

 bidirectional_11 (Bidirectional)     ?                                   0␣
↪(unbuilt)

 global_average_pooling1d_11          ?                                   0␣
↪(unbuilt)
 (GlobalAveragePooling1D)                                                    ␣
↪

 dense_26 (Dense)                     ?                                   0␣
↪(unbuilt)

 dense_27 (Dense)                     ?                                   0␣
↪(unbuilt)
```

Total params: 0 (0.00 B)

Trainable params: 0 (0.00 B)

Non-trainable params: 0 (0.00 B)

None

[281]:
```
EPOCHS = 100
BATCH_SIZE =  2048
```

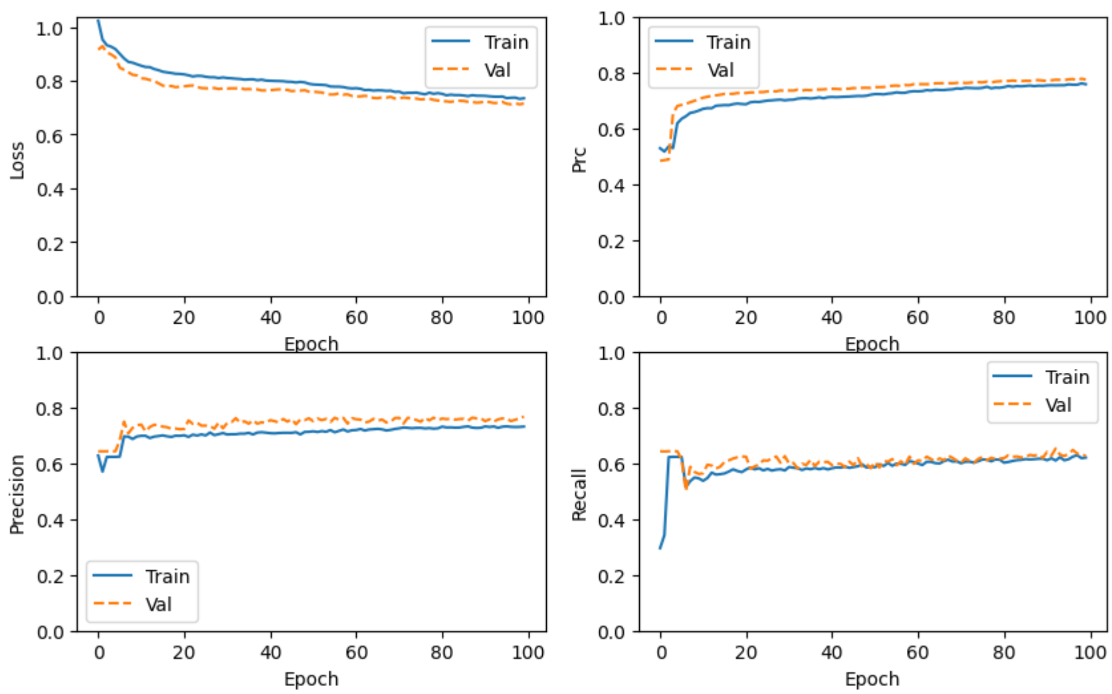
[282]:
```
steps_per_epoch = int(np.ceil(3.0*tweet_freq[0]/BATCH_SIZE))
```

[283]:
```
lr = tf.keras.callbacks.ReduceLROnPlateau(factor = 0.1, min_lr = 0.01,␣
 ↪monitor='val_prc')
early_stopping = tf.keras.callbacks.
 ↪EarlyStopping(monitor='val_prc',mode='max',patience=10,restore_best_weights=True)
```

```
model_1_history  = model1.
  ↪fit(X_train,Y_train,batch_size=BATCH_SIZE,epochs=EPOCHS,callbacks=[lr,early_stopping],valid
  ↪,Y_test),verbose=0)
```

[284]:
```
def plot_metrics(history):
  metrics = ['loss', 'prc','precision', 'recall']
  for n, metric in enumerate(metrics):
    name = metric.replace("_"," ").capitalize()
    plt.subplot(2,2,n+1)
    plt.rcParams["figure.figsize"] = (10,6)
    plt.plot(history.epoch, history.history[metric], label='Train')
    plt.plot(history.epoch, history.history['val_'+metric],linestyle="--",␣
  ↪label='Val')
    plt.xlabel('Epoch')
    plt.ylabel(name)
    if metric == 'loss':
      plt.ylim([0, plt.ylim()[1]])
    elif metric == 'auc':
      plt.ylim([0.8,1])
    else:
      plt.ylim([0,1])

    plt.legend()
```

[285]:
```
_ = plot_metrics(model_1_history)
```



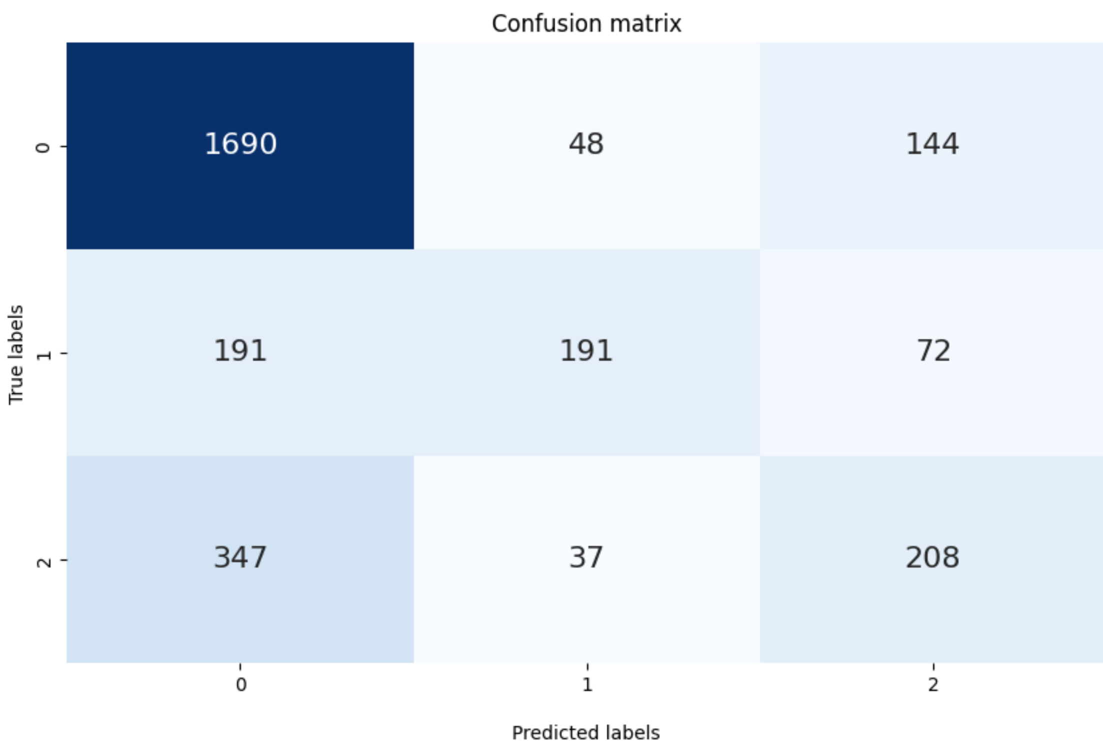
[286]:
```
model_1_results = model1.evaluate(X_test ,Y_test,verbose=0)
```

[287]:
```
for name, value in zip(model1.metrics_names, model_1_results):
  print(name, ': {:.3f}'.format(value))
print()
```

```
loss : 0.713
compile_metrics : 0.713
```

[288]:
```
model_1_predictions = np.argmax(model1.predict(X_test,verbose=0), axis=-1)
```

[289]:
```
_= conf_matrix(y_t, model_1_predictions)
```



[290]:
```
score_board2 = score_board_df(y_t, model_1_predictions,"RNN-Word2Vec")
```

[291]:
```
score_board = pd.concat([score_board,score_board2],ignore_index=True,␣
↪sort=False)
score_board
```

[291]:

	classifier-name	f1_score	precision	recall	accuracy	Average-method
0	LinearSVC + Weights	0.785	0.783	0.789	0.789	weighted

```
1          RNN-Word2Vec     0.585      0.647   0.557     0.713           macro
```

1.2.7 *Model 2 (With class Weights)*

Calculate class weights

[292]:
```
weight_for_0 = (1/tweet_freq[0]) * (sum(tweet_freq) / 3.0)
weight_for_1 = (1/tweet_freq[2]) * (sum(tweet_freq) / 3.0)
weight_for_2 = (1/tweet_freq[1]) * (sum(tweet_freq) / 3.0)
class_weights = {0: weight_for_0, 1: weight_for_1, 2: weight_for_2}
print("Weight for class negative: {:.2f}".format(weight_for_0))
print("Weight for class positive: {:.2f}".format(weight_for_1))
print("Weight for class neutral: {:.2f}".format(weight_for_2))
```

```
Weight for class negative: 0.53
Weight for class positive: 2.07
Weight for class neutral: 1.57
```

[293]:
```
model2 = create_model(vocab_size,32)
print(model2.summary())
```

```
Model: "sequential_12"


 Layer (type)                          Output Shape                         ␣
↪Param #

 embedding_14 (Embedding)              ?                                  0␣
↪(unbuilt)

 bidirectional_12 (Bidirectional)      ?                                  0␣
↪(unbuilt)

 global_average_pooling1d_12           ?                                  0␣
↪(unbuilt)
 (GlobalAveragePooling1D)                                                    ␣
↪

 dense_28 (Dense)                      ?                                  0␣
↪(unbuilt)

 dense_29 (Dense)                      ?                                  0␣
↪(unbuilt)


 Total params: 0 (0.00 B)
```

```
Trainable params: 0 (0.00 B)
```

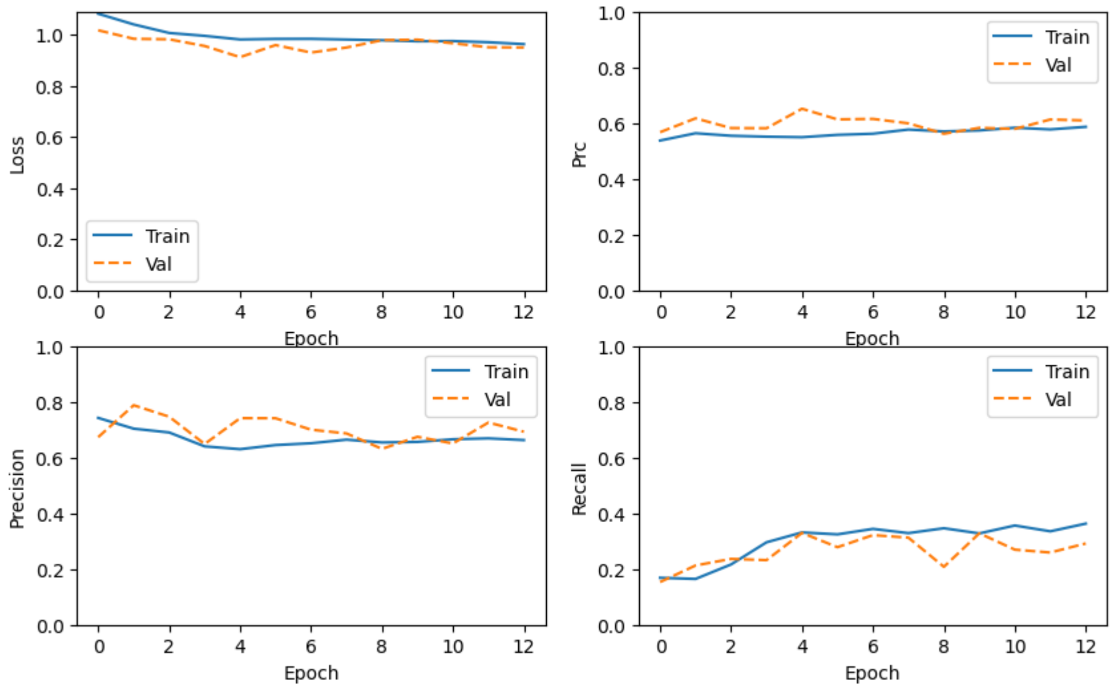
```
Non-trainable params: 0 (0.00 B)
```

```
None
```

[294]:
```
lr = tf.keras.callbacks.ReduceLROnPlateau(factor = 0.1, min_lr = 0.01, monitor='val_loss')
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_prc',patience=8,restore_best_weights=True)
model_2_history  = model2.fit(X_train,Y_train,batch_size=int(BATCH_SIZE*0.5),epochs=int(EPOCHS*0.5),callbacks=[lr,early_stopping],validation_data=(X_test ,Y_test),verbose=0,class_weight=class_weights)
```

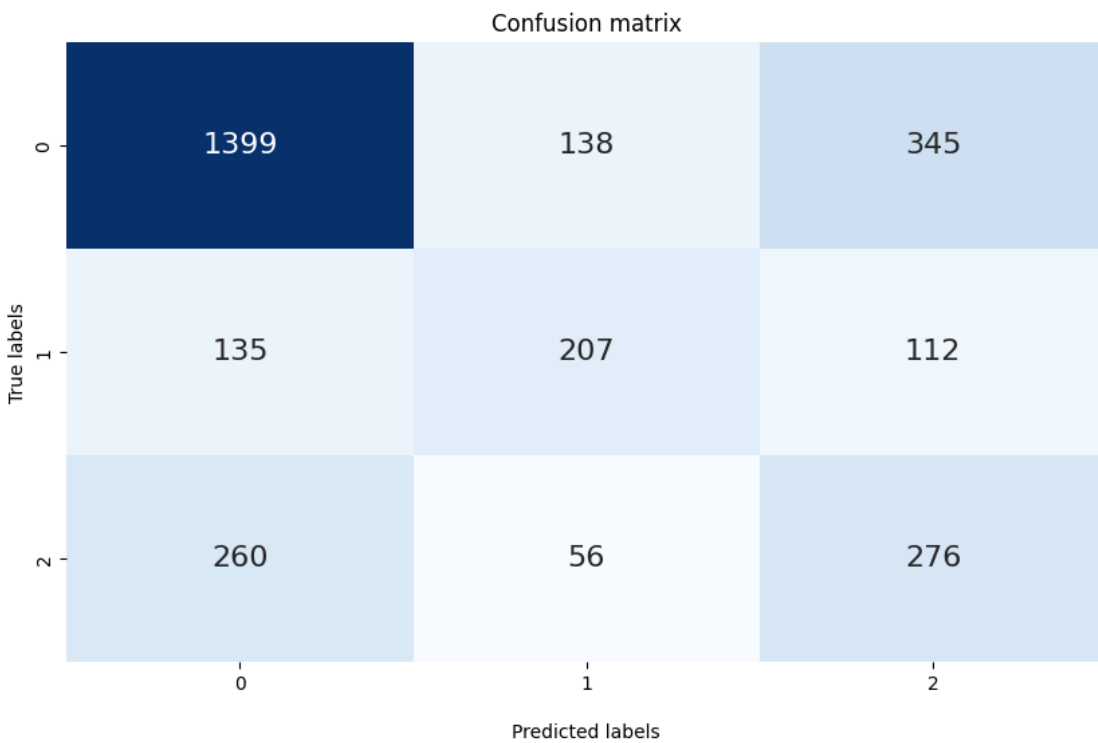
[295]:
```
model_2_results = model2.evaluate(X_test ,Y_test,verbose=0)
```

[296]:
```
_ = plot_metrics(model_2_history)
```



[297]:
```
for name, value in zip(model2.metrics_names, model_2_results):
  print(name, ': {:.3f}'.format(value))
print()
model_2_predictions = np.argmax(model2.predict(X_test,verbose=0), axis=1)
_= conf_matrix(y_t, model_2_predictions)
```

```
loss : 0.912
compile_metrics : 0.643
```



[298]:
```
score_board3 = score_board_df(y_t, model_2_predictions,"RNN-Word2Vec +␣
↪Weights","weighted")
score_board = pd.concat([score_board,score_board3],ignore_index=True,␣
↪sort=False)
score_board
```

[298]:

	classifier-name	f1_score	precision	recall	accuracy	Average-method
0	LinearSVC + Weights	0.785	0.783	0.789	0.789	weighted
1	RNN-Word2Vec	0.585	0.647	0.557	0.713	macro
2	RNN-Word2Vec + Weights	0.649	0.657	0.643	0.643	weighted

1.2.8 *Model 3 (Pre-Trained GloVe Embedding)*

[299]:
```
!wget http://nlp.stanford.edu/data/glove.6B.zip
!unzip glove.6B.zip
```

```
--2024-11-25 13:26:25--  http://nlp.stanford.edu/data/glove.6B.zip
Resolving nlp.stanford.edu (nlp.stanford.edu)… 171.64.67.140
Connecting to nlp.stanford.edu (nlp.stanford.edu)|171.64.67.140|:80…
connected.
```

```
HTTP request sent, awaiting response… 302 Found
Location: https://nlp.stanford.edu/data/glove.6B.zip [following]
--2024-11-25 13:26:25--  https://nlp.stanford.edu/data/glove.6B.zip
Connecting to nlp.stanford.edu (nlp.stanford.edu)|171.64.67.140|:443…
connected.
HTTP request sent, awaiting response… 301 Moved Permanently
Location: https://downloads.cs.stanford.edu/nlp/data/glove.6B.zip [following]
--2024-11-25 13:26:25--  https://downloads.cs.stanford.edu/nlp/data/glove.6B.zip
Resolving downloads.cs.stanford.edu (downloads.cs.stanford.edu)… 171.64.64.22
Connecting to downloads.cs.stanford.edu
(downloads.cs.stanford.edu)|171.64.64.22|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 862182613 (822M) [application/zip]
Saving to: ‘glove.6B.zip.3’

glove.6B.zip.3      100%[===================>] 822.24M  5.01MB/s    in 2m 39s

2024-11-25 13:29:04 (5.17 MB/s) - ‘glove.6B.zip.3’ saved [862182613/862182613]

Archive:  glove.6B.zip
replace glove.6B.50d.txt? [y]es, [n]o, [A]ll, [N]one, [r]ename:
```

[300]:

```
embeddings_index = dict()
```

[301]:

```
f = open('glove.6B.100d.txt')
```

[302]:

```
EMBEDDING_DIM = 100
```

[303]:

```
for line in f:
        values = line.split()
        word = values[0]
        coefs = np.asarray(values[1:], dtype='float32')
        embeddings_index[word] = coefs
f.close()
print('Loaded %s word vectors.' % len(embeddings_index))
```

```
Loaded 400000 word vectors.
```

[304]:

```
embedding_matrix = np.zeros((vocab_size, EMBEDDING_DIM))
num_words_in_embedding = 0
for word, i in vec.vocabulary_.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        num_words_in_embedding += 1
        embedding_matrix[i] = embedding_vector
```

[305]:

```
model3 = create_model(embedding_matrix.shape[0], embedding_matrix.
 ↪shape[1],weights = [embedding_matrix])
```

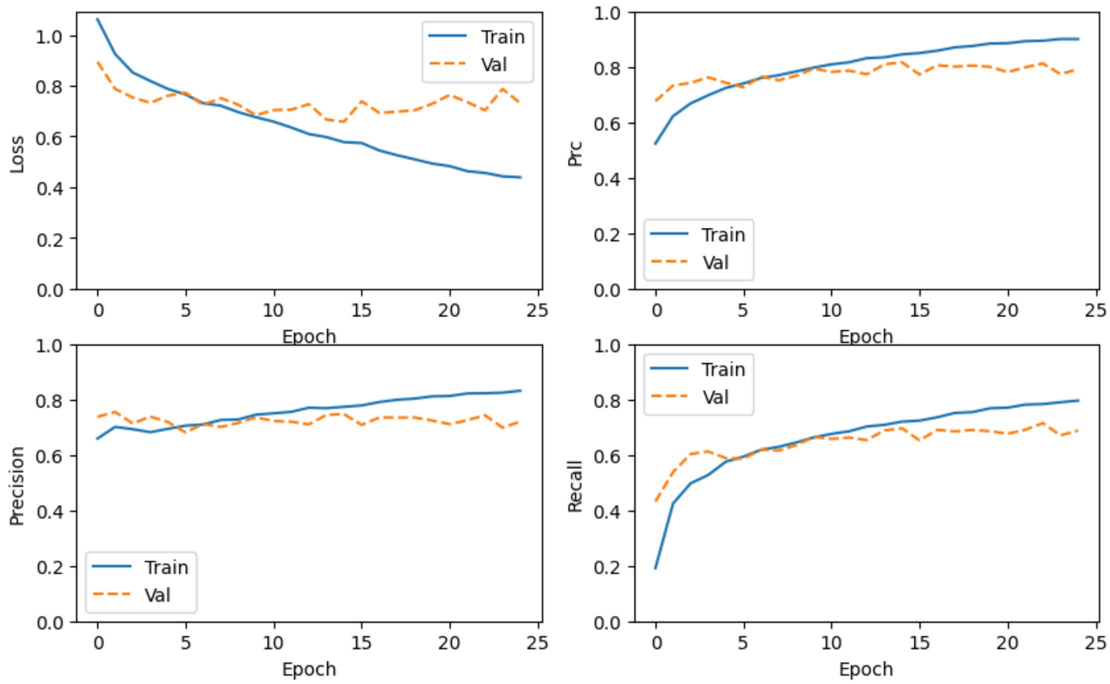
```
print(model2.summary())
```

```
Model: "sequential_12"


 Layer (type)                         Output Shape                        ␣
↪Param #

 embedding_14 (Embedding)             (None, 20, 32)                    ␣
↪1,974,112

 bidirectional_12 (Bidirectional)     (None, 20, 128)                       ␣
↪49,664

 global_average_pooling1d_12          (None, 128)                               ␣
↪  0
 (GlobalAveragePooling1D)                                                       ␣
↪

 dense_28 (Dense)                     (None, 200)                           ␣
↪25,800

 dense_29 (Dense)                     (None, 3)                                 ␣
↪603



 Total params: 2,202,315 (8.40 MB)


 Trainable params: 76,067 (297.14 KB)


 Non-trainable params: 1,974,112 (7.53 MB)


 Optimizer params: 152,136 (594.29 KB)


None
```

[306]:

```
lr = tf.keras.callbacks.ReduceLROnPlateau(factor = 0.1, min_lr = 0.01,␣
 ↪monitor='val_prc')
early_stopping = tf.keras.callbacks.
 ↪EarlyStopping(monitor='val_prc',mode='max',patience=10,restore_best_weights=True)
model_3_history  = model3.fit(X_train,Y_train,batch_size=int(BATCH_SIZE*0.
 ↪5),epochs=int(EPOCHS*0.
 ↪5),callbacks=[lr,early_stopping],validation_data=(X_test␣
 ↪,Y_test),verbose=0,class_weight=class_weights)
```

```
[307]: _ = plot_metrics(model_3_history)
```



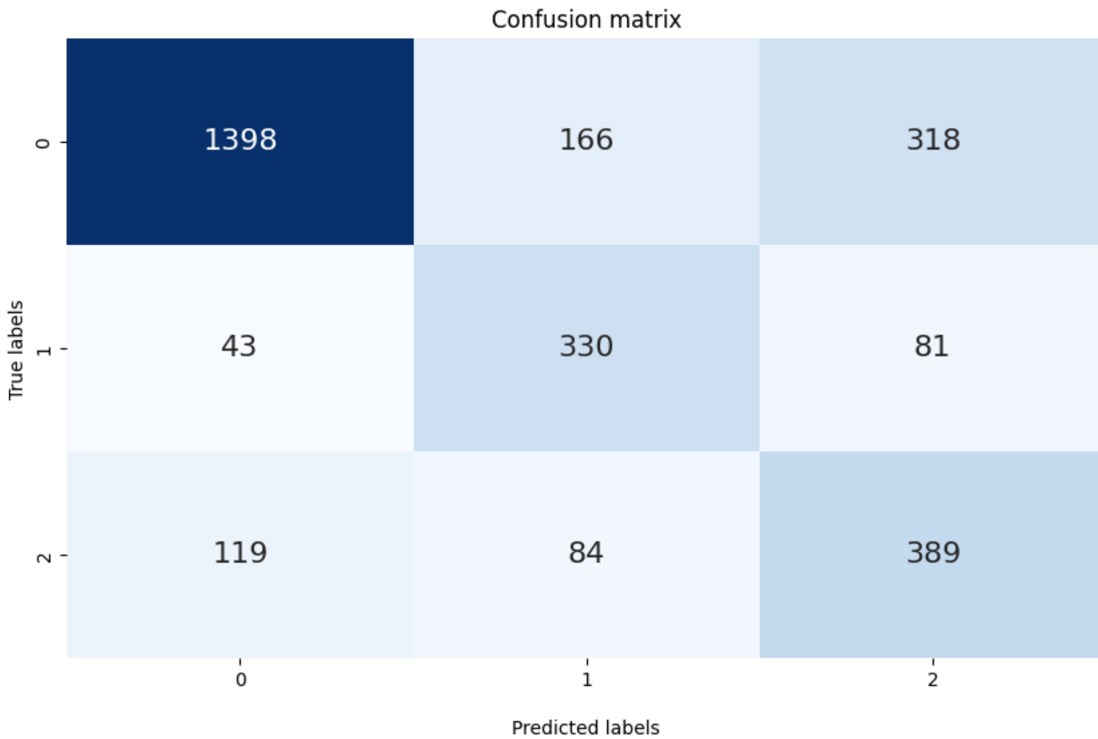

```
[308]: model_3_results = model3.evaluate(X_test ,Y_test,verbose=0)
```

```
[309]: for name, value in zip(model3.metrics_names, model_2_results):
         print(name, ': {:.3f}'.format(value))
       print()
```

```
loss : 0.912
compile_metrics : 0.643
```

```
[310]: model_3_predictions = np.argmax(model3.predict(X_test,verbose=0), axis=-1)
```

```
[311]: _= conf_matrix(y_t, model_3_predictions)
```



[312]:
```
score_board4 = score_board_df(y_t, model_3_predictions,"RNN-Glove +
↪Weights","weighted")
```

[313]:
```
score_board = pd.concat([score_board,score_board4],ignore_index=True,
↪sort=False)
score_board
```

[313]:

	classifier-name	f1_score	precision	recall	accuracy	Average-method
0	LinearSVC + Weights	0.785	0.783	0.789	0.789	weighted
1	RNN-Word2Vec	0.585	0.647	0.557	0.713	macro
2	RNN-Word2Vec + Weights	0.649	0.657	0.643	0.643	weighted
3	RNN-Glove + Weights	0.735	0.764	0.723	0.723	weighted

1.2.9 *Model 4 (With multiple features - x_n)*

[314]:
```
other_features_train = model_1_train.
↪drop(['target','text','airline_sentiment'],inplace=False,axis=1)
```

[315]:
```
other_features_test = model_1_test.
↪drop(['target','text','airline_sentiment'],inplace=False,axis=1)
```

[316]:
```
other_features_train.head(3)
```

[316]:
```
       airline_sentiment_confidence  retweet_count  week_day  day  hour  \
4219                         1.0000              0         1   17    11
13090                        0.6326              0         0   23    11
9033                         1.0000              0         1   24     9

       url_flag  happy_emoji  sad_emoji  airline_code
4219          0            0          0             1
13090         0            0          0             5
9033          0            0          0             4
```

[317]:
```
other_features_train.shape
```

[317]:
```
(11659, 9)
```

[318]:
```
X_train_2 = np.asarray(other_features_train.values,dtype=int)
```

[319]:
```
X_test_2 = np.asarray(other_features_test.values,dtype=int)
```

[320]:
```
print(X_train.shape,X_train_2.shape)
```

```
(11659, 20) (11659, 9)
```

[321]:
```
input_dense = keras.layers.Input(shape=(9,))
input_embedding = keras.layers.Input(shape=(MAX_LENGTH,))
embedding = keras.layers.Embedding(embedding_matrix.shape[0], embedding_matrix.
 ↪shape[1], weights=[embedding_matrix])(input_embedding)
lstm = keras.layers.LSTM(200)(embedding)
concat = keras.layers.Concatenate()([lstm,input_dense])
Dense1 = keras.layers.Dense(200,activation='relu')(concat)
Dropout =keras.layers.Dropout(0.2)(Dense1)
Output = keras.layers.Dense(3,activation='softmax')(Dropout)
model4 = keras.Model(inputs=[input_embedding, input_dense], outputs=[Output])
model4.compile(loss=tf.keras.losses.CategoricalCrossentropy(),optimizer=keras.
 ↪optimizers.Adam(learning_rate=0.005),metrics =METRICS)
print(model3.summary())
```

```
Model: "sequential_13"


 Layer (type)                          Output Shape                              ␣
↪Param #

 embedding_15 (Embedding)              (None, 20, 100)                         ␣
↪6,169,100

 bidirectional_13 (Bidirectional)      (None, 20, 128)                            ␣
↪84,480
```

```
 global_average_pooling1d_13           (None, 128)                             ␣
↪  0
 (GlobalAveragePooling1D)                                                      ␣
↪

 dense_30 (Dense)                      (None, 200)                          ␣
↪25,800

 dense_31 (Dense)                      (None, 3)                               ␣
↪603
```

Total params: 6,501,751 (24.80 MB)

Trainable params: 110,883 (433.14 KB)

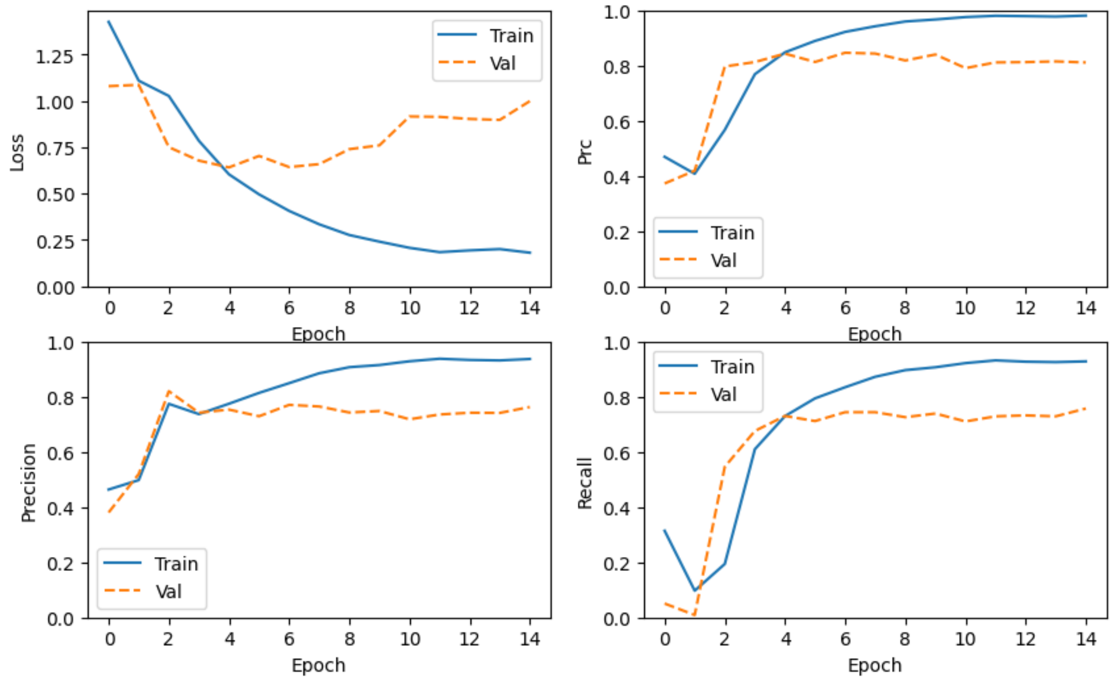
Non-trainable params: 6,169,100 (23.53 MB)

Optimizer params: 221,768 (866.29 KB)

None

[322]:
```
lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss',␣
 ↪patience=10,factor=0.5, min_lr=0.001)
early_stop = keras.callbacks.
 ↪EarlyStopping(monitor='val_prc',mode='max',patience=8,␣
 ↪restore_best_weights=True)
model_4_history = model4.
 ↪fit([X_train,X_train_2],Y_train,batch_size=int(BATCH_SIZE*0.
 ↪25),epochs=EPOCHS,validation_data=([X_test, X_test_2], Y_test),␣
 ↪callbacks=[lr,early_stop],verbose=0,class_weight=class_weights)
```

[323]:
```
plot_metrics(model_4_history)
```



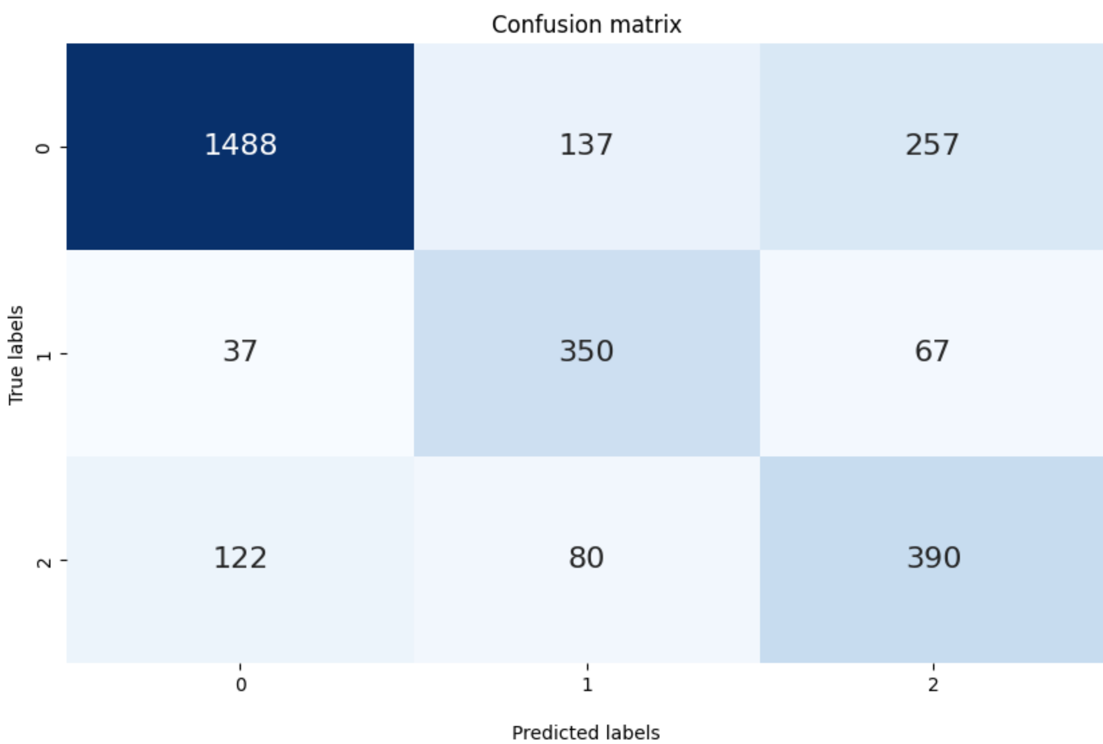
[324]:
```
model_4_results = model4.evaluate([X_test, X_test_2],Y_test,verbose=0)
```

[325]:
```
for name, value in zip(model4.metrics_names, model_4_results):
  print(name, ': {:.3f}'.format(value))
print()
```

```
loss : 0.643
compile_metrics : 0.761
```

[326]:
```
model_4_predictions = np.argmax(model4.predict([X_test, X_test_2],verbose=0),␣
  ↪axis=-1)
```

[327]:
```
_= conf_matrix(y_t, model_4_predictions)
```



[328]:
```
score_board5 = score_board_df(y_t, model_4_predictions,"RNN-Glove + Weights +␣
↪other features","weighted")
```

[329]:
```
score_board = pd.concat([score_board,score_board5],ignore_index=True,␣
↪sort=False)
score_board
```

[329]:
```
                      classifier-name f1_score precision recall accuracy  \
0                  LinearSVC + Weights    0.785     0.783  0.789    0.789
1                         RNN-Word2Vec    0.585     0.647  0.557    0.713
2               RNN-Word2Vec + Weights    0.649     0.657  0.643    0.643
3                  RNN-Glove + Weights    0.735     0.764  0.723    0.723
4  RNN-Glove + Weights + other features    0.769     0.787  0.761    0.761

  Average-method
0       weighted
1          macro
2       weighted
3       weighted
4       weighted
```

1.2.10 *Model 5 (Experiment-Upsampling of miniority classes using numpy)*

[330]:
```
train_labels =  y.reshape(-1, 1)
test_labels = y_t.reshape(-1, 1)
```

[331]:
```
train_labels[27:30]
```

[331]:
```
array([[0],
       [2],
       [1]])
```

[332]:
```
train_features = X_train
test_features = X_test
```

[333]:
```
bool_train_labels_pos =((train_labels[:, 0] != 0 ) & (train_labels[:, 0] != 2 ))
```

[334]:
```
bool_train_labels_neg = ((train_labels[:, 0] != 1 ) & (train_labels[:, 0] != 2␣
↪))
```

[335]:
```
bool_train_labels_neu = ((train_labels[:, 0] != 0 ) & (train_labels[:, 0] != 1␣
↪))
```

[336]:
```
#bool_train_labels_neu[27:30]
```

[337]:
```
pos_features = train_features[bool_train_labels_pos]
pos_labels = train_labels[bool_train_labels_pos]
```

[338]:
```
neg_features = train_features[bool_train_labels_neg]
```

[339]:
```
neg_labels = train_labels[bool_train_labels_neg]
```

[340]:
```
pos_ids = np.arange(len(pos_features))
```

[341]:
```
random_choices1 = np.random.choice(pos_ids, len(neg_features))
```

[342]:
```
res_pos_features = pos_features[random_choices1]
res_pos_labels = pos_labels[random_choices1]
```

[343]:
```
res_pos_features.shape
```

[343]:
```
(7260, 20)
```

[344]:
```
neu_features = train_features[bool_train_labels_neu]
neu_labels = train_labels[bool_train_labels_neu]
```

[345]:
```
neu_ids = np.arange(len(neu_features))
```

[346]:
```
random_choices2 = np.random.choice(neu_ids, len(neg_features))
```

[347]:
```
res_neu_features = neu_features[random_choices2]
res_neu_labels = neu_labels[random_choices2]
```

[348]:
```
res_neu_features.shape
```

[348]: (7260, 20)

[349]:
```
resampled_features = np.concatenate([res_pos_features,res_neu_features,␣
↪neg_features], axis=0)
```

[350]:
```
resampled_labels = np.concatenate([res_pos_labels,res_neu_labels, neg_labels],␣
↪axis=0)
```

[351]:
```
order = np.arange(len(resampled_labels))
np.random.shuffle(order)
resampled_features = resampled_features[order]
resampled_labels = resampled_labels[order]
```

[352]:
```
resampled_features.shape
```

[352]: (21780, 20)

[353]:
```
resampled_labels = keras.utils.to_categorical(resampled_labels, num_classes=3)
```

[354]:
```
model5 = create_model(embedding_matrix.shape[0], embedding_matrix.
↪shape[1],weights = [embedding_matrix])
print(model5.summary())
```

Model: "sequential_14"

```
 Layer (type)                          Output Shape                          ␣
↪Param #

 embedding_17 (Embedding)              ?                                   ␣
↪6,169,100

 bidirectional_14 (Bidirectional)      ?                                  0␣
↪(unbuilt)

 global_average_pooling1d_14           ?                                  0␣
↪(unbuilt)
 (GlobalAveragePooling1D)                                                      ␣
↪

 dense_34 (Dense)                      ?                                  0␣
↪(unbuilt)
```

```
 dense_35 (Dense)                      ?                                   0␣
↪(unbuilt)
```

Total params: 6,169,100 (23.53 MB)

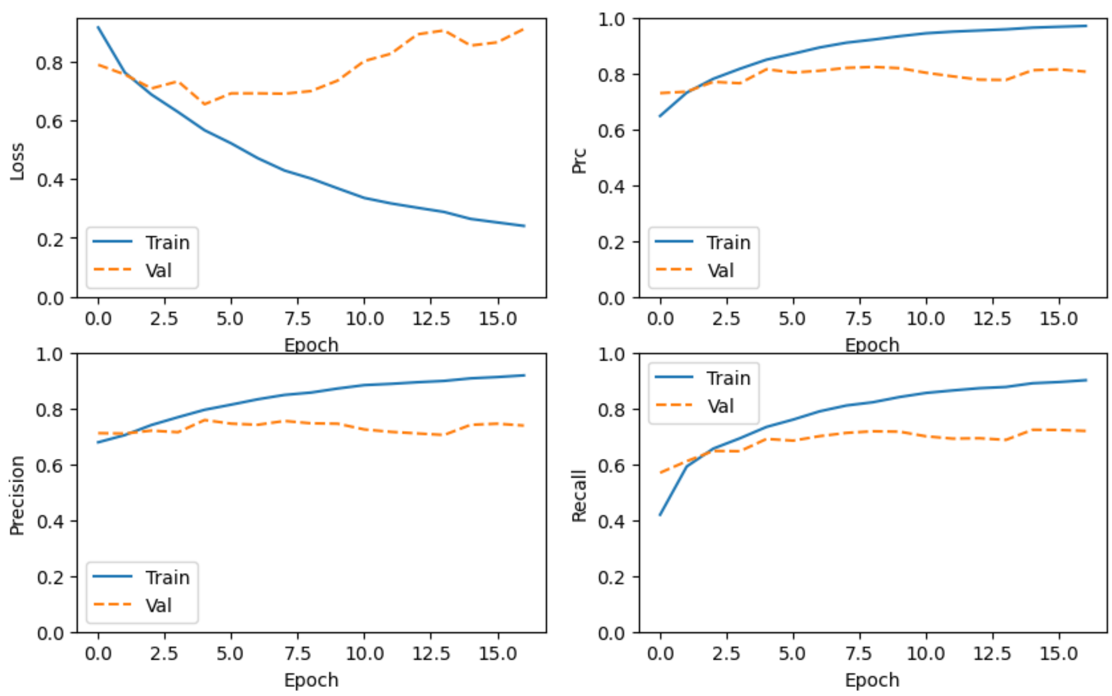
Trainable params: 0 (0.00 B)

Non-trainable params: 6,169,100 (23.53 MB)

None

[355]:
```
lr = tf.keras.callbacks.ReduceLROnPlateau(factor = 0.1, min_lr = 0.01,␣
 ↪monitor='val_prc')
early_stopping = tf.keras.callbacks.
 ↪EarlyStopping(monitor='val_prc',mode="max",patience=8,restore_best_weights=True)
model_5_history  = model5.fit(resampled_features,resampled_labels␣
 ↪,batch_size=int(BATCH_SIZE*0.
 ↪25),epochs=EPOCHS,callbacks=[lr,early_stopping],validation_data=(X_test␣
 ↪,Y_test),verbose=0)
```

[356]:
```
_ = plot_metrics(model_5_history)
```



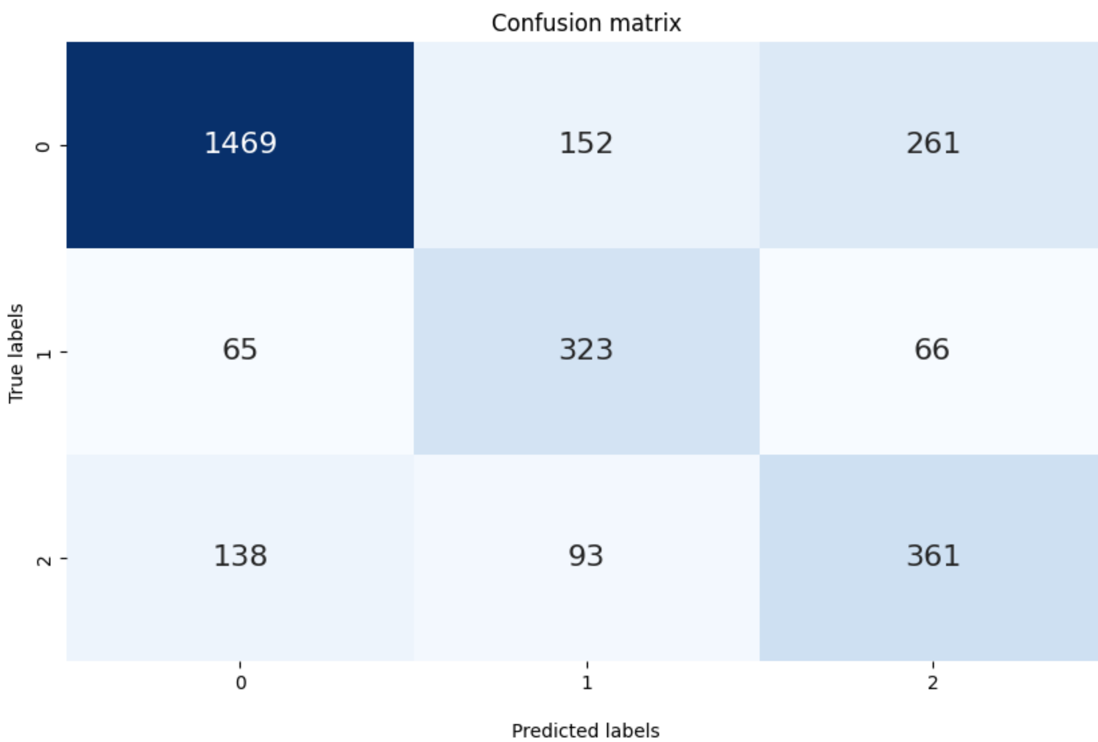
[357]:
```
model_5_results = model5.evaluate(X_test ,Y_test,verbose=0)
```

[358]:
```
for name, value in zip(model5.metrics_names, model_5_results):
  print(name, ': {:.3f}'.format(value))
print()
```

```
loss : 0.699
compile_metrics : 0.735
```

[359]:
```
model_5_predictions = np.argmax(model5.predict(X_test,verbose=0), axis=-1)
```

[360]:
```
_= conf_matrix(y_t, model_5_predictions)
```



[361]:
```
score_board6 = score_board_df(y_t, model_5_predictions,"RNN-Glove + Upsampling")
score_board = pd.concat([score_board,score_board6],ignore_index=True,␣
  ↪sort=False)
score_board
```

[361]:
```
                     classifier-name f1_score precision recall accuracy  \
0                 LinearSVC + Weights    0.785     0.783  0.789    0.789
1                        RNN-Word2Vec    0.585     0.647  0.557    0.713
2               RNN-Word2Vec + Weights    0.649     0.657  0.643    0.643
```

3	RNN-Glove + Weights	0.735	0.764	0.723	0.723
4	RNN-Glove + Weights + other features	0.769	0.787	0.761	0.761
5	RNN-Glove + Upsampling	0.674	0.657	0.701	0.735

	Average-method
0	weighted
1	macro
2	weighted
3	weighted
4	weighted
5	macro