Sentimental Analysis

Dataset Tweets Airlines

- Listing all the column names

In [2]:

```
import pandas as pd
tweets = pd.read_csv("Tweets.csv")
list(tweets.columns.values)
```

Out[2]:

```
['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']
```

The first five data-set

In [3]:

```
tweets.head()
```

Out[3]:

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativ
0	570306133677760513	neutral	1.0000	NaN	
1	570301130888122368	positive	0.3486	NaN	
2	570301083672813571	neutral	0.6837	NaN	
3	570301031407624196	negative	1.0000	Bad Flight	
4	570300817074462722	negative	1.0000	Can't Tell	

Listed the number of positive, negative and neutral tweets

In [4]:

```
sentiment_counts = tweets.airline_sentiment.value_counts()
print(sentiment_counts)
```

```
negative    9178
neutral     3099
positive    2363
Name: airline_sentiment, dtype: int64
```

All the different airlines with all the labels count

In [5]:

```
dff = tweets.groupby(["airline", "airline_sentiment" ]).count()["name"]
df_companySentiment = dff.to_frame().reset_index()
df_companySentiment.columns = ["airline", "airline_sentiment", "count"]
df_companySentiment
```

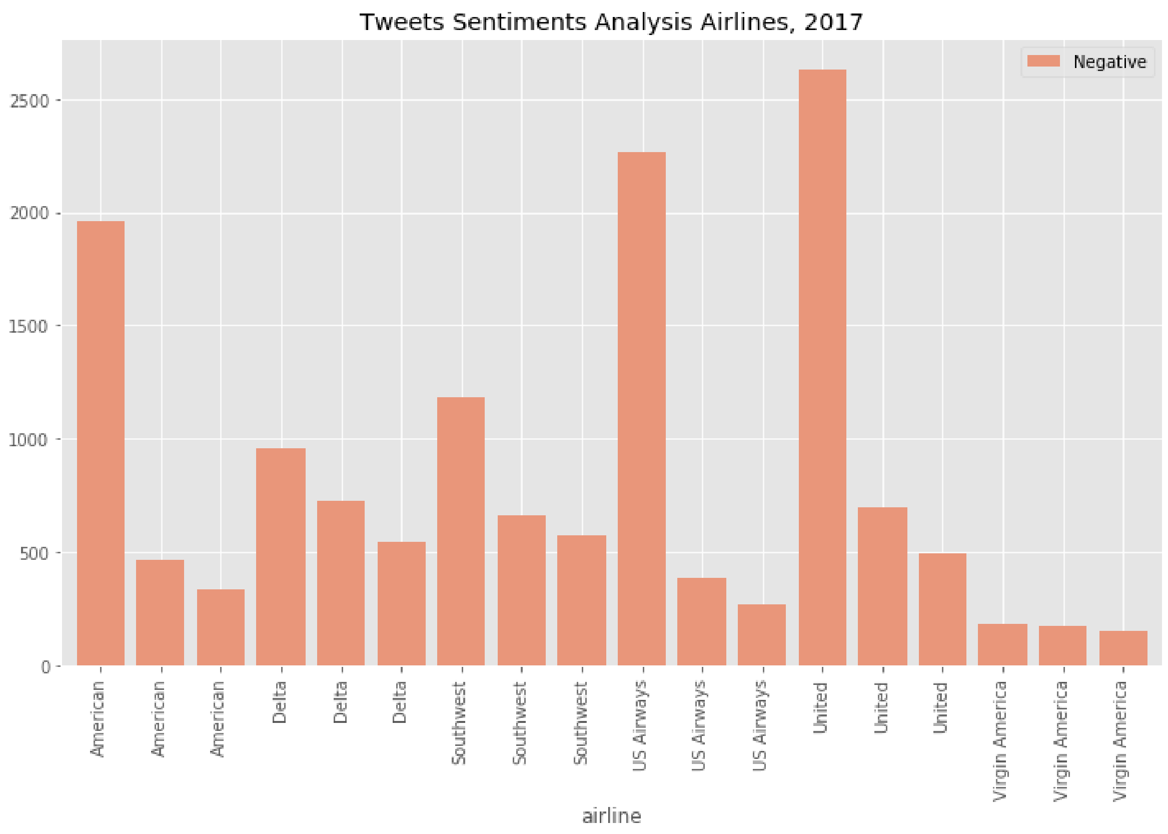
Out[5]:

	airline	airline_sentiment	count
0	American	negative	1960
1	American	neutral	463
2	American	positive	336
3	Delta	negative	955
4	Delta	neutral	723
5	Delta	positive	544
6	Southwest	negative	1186
7	Southwest	neutral	664
8	Southwest	positive	570
9	US Airways	negative	2263
10	US Airways	neutral	381
11	US Airways	positive	269
12	United	negative	2633
13	United	neutral	697
14	United	positive	492
15	Virgin America	negative	181
16	Virgin America	neutral	171
17	Virgin America	positive	152

A plot showing the negativeness in tweets by different airlines

In [6]:

```
import matplotlib.pyplot as plt
import matplotlib.style
%matplotlib inline
import matplotlib.style
from matplotlib.pyplot import subplots
matplotlib.style.use('ggplot')

df2 = df_companySentiment
df2.index = df2['airline']
del df2['airline']
df2
fig, ax = subplots()
my_colors =['darksalmon', 'papayawhip', 'cornflowerblue']
df2.plot(kind='bar', stacked=True, ax=ax, color=my_colors, figsize=(12, 7), width=0
ax.legend(["Negative", "Neutral", "Positive"])
plt.title("Tweets Sentiments Analysis Airlines, 2017")
plt.show()
```



WordNetLemmatizer- lemmatizing can often create actual words.

stopwords- Stopwords are the English words which does not add much meaning to a sentence.

In [13]:

```
import re, nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
wordnet_lemmatizer = WordNetLemmatizer()

def normalizer(tweet):
    only_letters = re.sub("[^a-zA-Z]", " ",tweet)
    tokens = nltk.word_tokenize(only_letters)[2:]
    lower_case = [l.lower() for l in tokens]
    filtered_result = list(filter(lambda l: l not in stop_words, lower_case))
    lemmas = [wordnet_lemmatizer.lemmatize(t) for t in filtered_result]
    return lemmas
```

In [15]:

```
normalizer("I recently wrote some texts.")
```

Out[15]:

```
['wrote', 'text']
```

In [16]:

```
pd.set_option('display.max_colwidth', -1) # Setting this so we can see the full cor
tweets['normalized_tweet'] = tweets.text.apply(normalizer)
tweets[['text','normalized_tweet']].head()
```

Out[16]:

	text	normalized_tweet
0	@VirginAmerica What @dhepburn said.	[dhepburn, said]
1	@VirginAmerica plus you've added commercials to the experience... tacky.	[added, commercial, experience, tacky]
2	@VirginAmerica I didn't today... Must mean I need to take another trip!	[today, must, mean, need, take, another, trip]
3	@VirginAmerica it's really aggressive to blast obnoxious "entertainment" in your guests' faces &amp; they have little recourse	[really, aggressive, blast, obnoxious, entertainment, guest, face, amp, little, recourse]
4	@VirginAmerica and it's a really big bad thing about it	[really, big, bad, thing]

In [17]:

```
from nltk import ngrams
def ngrams(input_list):
    #onegrams = input_list
    bigrams = [' '.join(t) for t in list(zip(input_list, input_list[1:]))]
    trigrams = [' '.join(t) for t in list(zip(input_list, input_list[1:], input_lis
    return bigrams+trigrams
tweets['grams'] = tweets.normalized_tweet.apply(ngrams)
tweets[['grams']].head()
```

Out[17]:

	grams
0	[dhepburn said]
1	[added commercial, commercial experience, experience tacky, added commercial experience, commercial experience tacky]
2	[today must, must mean, mean need, need take, take another, another trip, today must mean, must mean need, mean need take, need take another, take another trip]
3	[really aggressive, aggressive blast, blast obnoxious, obnoxious entertainment, entertainment guest, guest face, face amp, amp little, little recourse, really aggressive blast, aggressive blast obnoxious, blast obnoxious entertainment, obnoxious entertainment guest, entertainment guest face, guest face amp, face amp little, amp little recourse]
4	[really big, big bad, bad thing, really big bad, big bad thing]

In [18]:

```
import collections
def count_words(input):
    cnt = collections.Counter()
    for row in input:
        for word in row:
            cnt[word] += 1
    return cnt
```

In [19]:

```
tweets[(tweets.airline_sentiment == 'negative')][['grams']].apply(count_words)['gra
```

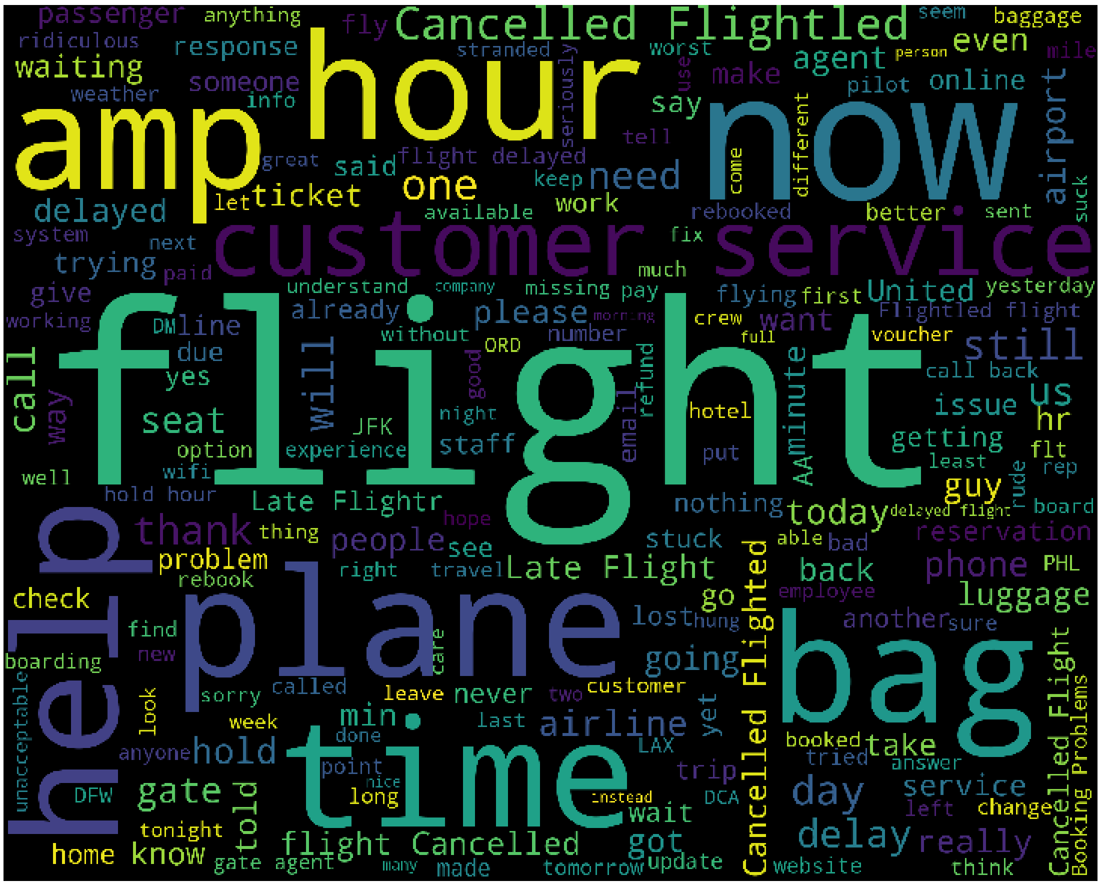
Out[19]:

```
[('http co', 449),
 ('customer service', 438),
 ('cancelled flightled', 425),
 ('late flight', 215),
 ('cancelled flighted', 196),
 ('flight cancelled', 185),
 ('late flightr', 144),
 ('cancelled flight', 131),
 ('hold hour', 128),
 ('flightled flight', 123),
 ('flight cancelled flightled', 117),
 ('flight delayed', 115),
 ('cancelled flightled flight', 107),
 ('call back', 106),
 ('booking problem', 98),
 ('gate agent', 83),
 ('flight flight', 74),
 ('hour late', 69),
 ('delayed flight', 69),
 ('flight attendant', 60)]
```

In [21]:

```
train_neg = tweets[(tweets.airline_sentiment == 'negative')]
train_neg = train_neg['text']
train_pos = tweets[(tweets.airline_sentiment == 'positive')]
train_pos = train_pos['text']

from wordcloud import WordCloud,STOPWORDS

def wordcloud_draw(data, color = 'black'):
    words = ' '.join(data)
    cleaned_word = " ".join([word for word in words.split()
                            if 'http' not in word
                                and not word.startswith('@')
                                and not word.startswith('#')
                                and word != 'RT'
                            ])
    wordcloud = WordCloud(stopwords=STOPWORDS,
                      background_color=color,
                      width=2500,
                      height=2000
                     ).generate(cleaned_word)
    plt.figure(1,figsize=(13, 13))
    plt.imshow(wordcloud)
    plt.axis('off')
    plt.show()

print("Negative words")
wordcloud_draw(train_neg)
```

```
Negative words
```



In [22]:

```
tweets[(tweets.airline_sentiment == 'positive')][['grams']].apply(count_words)['gra
```

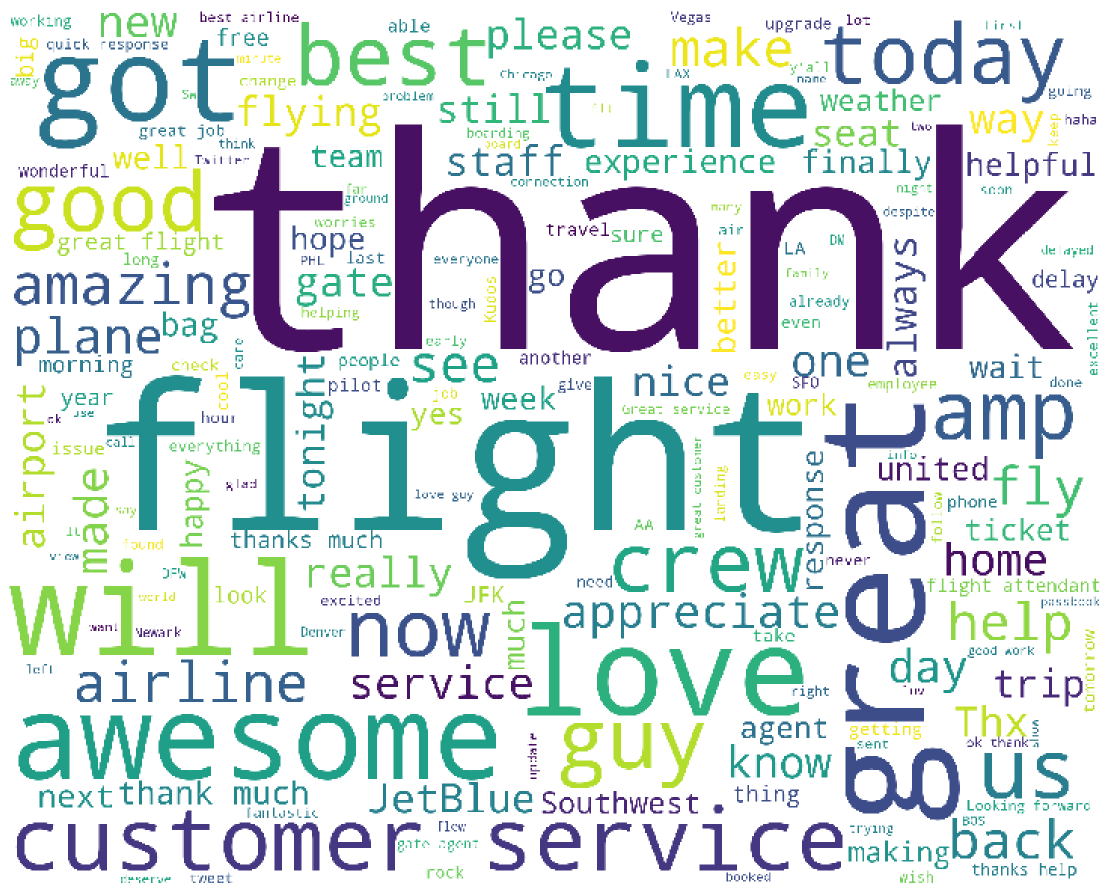
Out[22]:

```
[('http co', 233),
 ('customer service', 91),
 ('flight attendant', 25),
 ('quick response', 19),
 ('great flight', 17),
 ('best airline', 16),
 ('great job', 16),
 ('great service', 16),
 ('gate agent', 16),
 ('booking problem', 15),
 ('thanks help', 15),
 ('thank much', 15),
 ('good work', 14),
 ('fleet fleek', 14),
 ('fleek http', 14),
 ('fleet fleek http', 14),
 ('fleek http co', 14),
 ('guy rock', 13),
 ('looking forward', 13),
 ('great customer', 12)]
```

In [23]:

```
print("Positive words")
wordcloud_draw(train_pos, 'white')
```

```
Positive words
```



CountVectorizer - implements both tokenization and occurrence counting in a single class

In [26]:

```
import numpy as np
from scipy.sparse import hstack
from sklearn.feature_extraction.text import CountVectorizer
count_vectorizer = CountVectorizer(ngram_range=(1,2))
```

In [27]:

```
vectorized_data = count_vectorizer.fit_transform(tweets.text)
indexed_data = hstack((np.array(range(0,vectorized_data.shape[0]))[:,None], vectori
```

In [28]:

```
def sentiment2target(sentiment):
    return {
        'negative': 0,
        'neutral': 1,
        'positive' : 2
    }[sentiment]
targets = tweets.airline_sentiment.apply(sentiment2target)
```

In [29]:

```
from sklearn.model_selection import train_test_split
data_train, data_test, targets_train, targets_test = train_test_split(indexed_data,
data_train_index = data_train[:,0]
data_train = data_train[:,1:]
data_test_index = data_test[:,0]
data_test = data_test[:,1:]
```

OneVsRestClassifier- this strategy consists in fitting one classifier per class. For each classifier, the class is fitted against all the other classes.

In [31]:

```
from sklearn import svm
from sklearn.multiclass import OneVsRestClassifier
from sklearn.model_selection import cross_val_score

clf = OneVsRestClassifier(svm.SVC(gamma=0.01, C=100., probability=True, class_weigh
# scores = cross_val_score(clf, indexed_data, targets, cv=1)
# scores
clf_output = clf.fit(data_train, targets_train).decision_function(data_test)
```

In [32]:

```
clf.score(data_test, targets_test)
```

Out[32]:

```
0.7851775956284153
```

In [33]:

```
sentences = count_vectorizer.transform([
    "What a great airline, the trip was a pleasure!",
    "My issue was quickly resolved after calling customer support. Thanks!",
    "What the hell! My flight was cancelled again. This sucks!",
    "Service was awful. I'll never fly with you again.",
    "You fuckers lost my luggage. Never again!",
    "I have mixed feelings about airlines. I don't know what I think.",
    ""
])
clf.predict_proba(sentences)
```

Out[33]:

```
array([[0.20735092, 0.05926853, 0.73338055],
       [0.14006381, 0.0716648 , 0.78827139],
       [0.94247711, 0.04033469, 0.01718819],
       [0.88995503, 0.07318112, 0.03686385],
       [0.9733419 , 0.01754863, 0.00910946],
       [0.46776769, 0.50132269, 0.03090963],
       [0.26457433, 0.52210386, 0.21332181]])
```

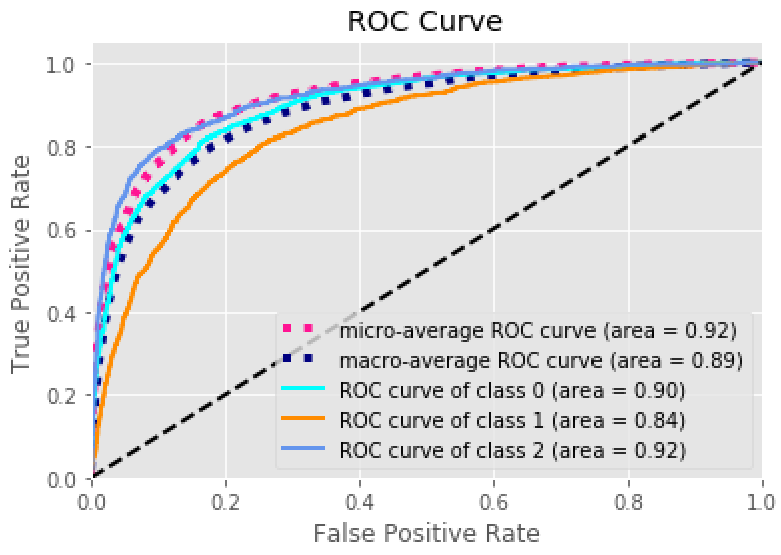
ROC Curve

In [34]:

```
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import roc_auc_score
from sklearn.preprocessing import label_binarize

# Compute macro-average ROC curve and ROC area
y = label_binarize(targets_test, classes=[0, 1, 2])
n_classes = y.shape[1]

fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y[:, i], clf_output[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Compute micro-average ROC curve and ROC area
fpr["micro"], tpr["micro"], _ = roc_curve(y.ravel(), clf_output.ravel())
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
```

In [35]:

```python
from scipy import interp
from itertools import cycle

lw = 2

# First aggregate all false positive rates
all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)]))

# Then interpolate all ROC curves at this points
mean_tpr = np.zeros_like(all_fpr)
for i in range(n_classes):
    mean_tpr += interp(all_fpr, fpr[i], tpr[i])

# Finally average it and compute AUC
mean_tpr /= n_classes

fpr["macro"] = all_fpr
tpr["macro"] = mean_tpr
roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])

# Plot all ROC curves
plt.figure()
plt.plot(fpr["micro"], tpr["micro"],
         label='micro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["micro"]),
         color='deeppink', linestyle=':', linewidth=4)

plt.plot(fpr["macro"], tpr["macro"],
         label='macro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["macro"]),
         color='navy', linestyle=':', linewidth=4)

colors = cycle(['aqua', 'darkorange', 'cornflowerblue'])
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=lw,
             label='ROC curve of class {0} (area = {1:0.2f})'
             ''.format(i, roc_auc[i]))

plt.plot([0, 1], [0, 1], 'k--', lw=lw)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc="lower right")
plt.show()
```


ROC Curve
True Positive Rate
False Positive Rate
micro-average ROC curve (area = 0.92)
macro-average ROC curve (area = 0.89)
ROC curve of class 0 (area = 0.90)
ROC curve of class 1 (area = 0.84)
ROC curve of class 2 (area = 0.92)
0.0
0.2
0.4
0.6
0.8
1.0

```
In [ ]:
```