Sentiment Analysis using US-Airlines dataset

In [1]:

```
import numpy as np
import pandas as pd
import string,re
import nltk
#nltk.download('all')
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Downloading Dataset

In [2]:

```
data_source_url = "https://raw.githubusercontent.com/satyajeetkrjha/kaggle-Twitter-US-Airline-Sentiment-/master/T
airline_tweets = pd.read_csv(data_source_url)
airline_tweets.head()
```

Out[2]:

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	airline_sentiment_gold
0	570306133677760513	neutral	1.0000	NaN	NaN	Virgin America	NaN
1	570301130888122368	positive	0.3486	NaN	0.0000	Virgin America	NaN
2	570301083672813571	neutral	0.6837	NaN	NaN	Virgin America	NaN
3	570301031407624196	negative	1.0000	Bad Flight	0.7033	Virgin America	NaN
4	570300817074462722	negative	1.0000	Can't Tell	1.0000	Virgin America	NaN

In [3]:

```
airline_tweets.shape
```

Out[3]: (14640, 15)

Converting target labels into numerical form for ease

In [3]:

```
airline_tweets['sentiment_class'] =
airline_tweets['airline_sentiment'].map({'negative' : 0,
                                          'neutral' : 3,
                                          'positive' : 4})
airline_tweets.head()
```

Out[3]:

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	airline_sentiment_gold
0	570306133677760513	neutral	1.0000	NaN	NaN	Virgin America	NaN
1	570301130888122368	positive	0.3486	NaN	0.0000	Virgin America	NaN
2	570301083672813571	neutral	0.6837	NaN	NaN	Virgin America	NaN
3	570301031407624196	negative	1.0000	Bad Flight	0.7033	Virgin America	NaN
4	570300817074462722	negative	1.0000	Can't Tell	1.0000	Virgin America	NaN

In [4]:

```
airline_tweets.shape
```

Out[4]:

```
(14640, 16)
```

In [5]:
```
airline_tweets.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14640 entries, 0 to 14639
Data columns (total 16 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 
 15  sentiment_class               14640 non-null  int64  
dtypes: float64(2), int64(3), object(11)
memory usage: 1.8+ MB
```

In [6]:
```
airline_tweets.isnull().sum()
```

Out[6]:
```
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
sentiment_class                     0
dtype: int64
```

Pre-defining fix Fig size

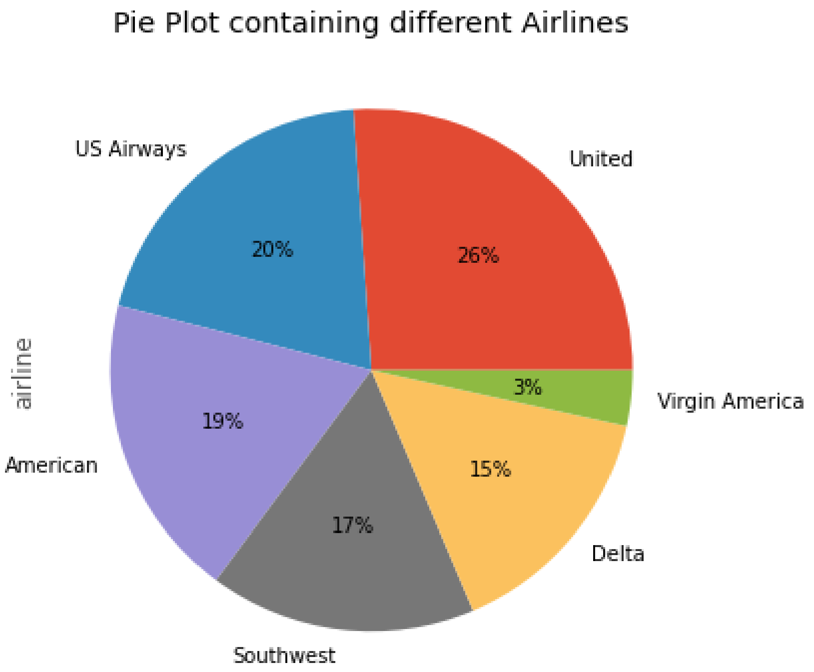
In [7]:
```
plt.rcParams["font.family"] = 'DejaVu Sans'
plt.style.use('ggplot')
plot_size = plt.rcParams["figure.figsize"]
# print(plot_size[0])
# print(plot_size[1])

plot_size[0] = 8
plot_size[1] = 6
plt.rcParams["figure.figsize"] = plot_size
```

EDA part

In [8]:

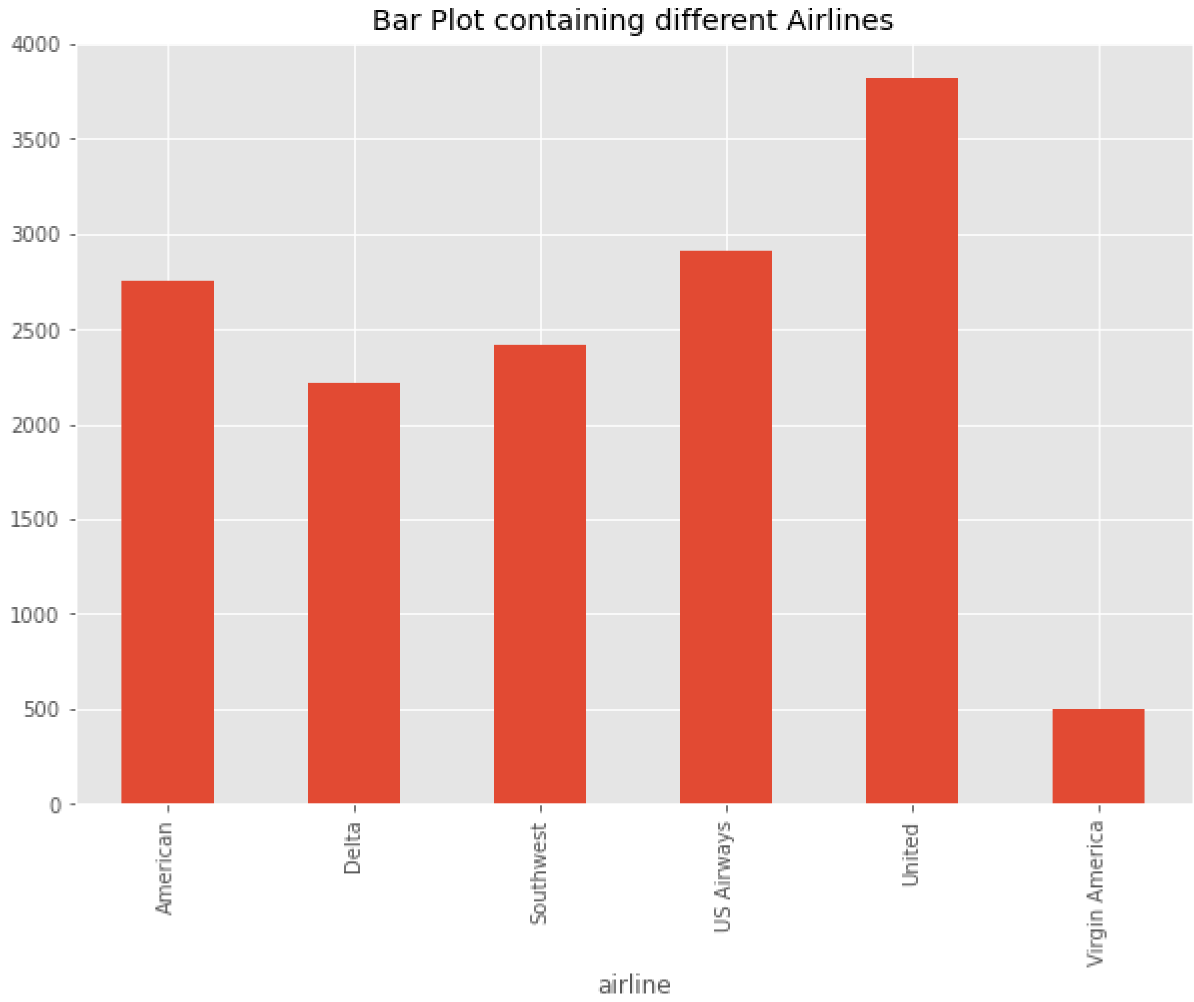
```
airline_tweets.airline.value_counts().plot(kind='pie',
                               title='Pie Plot containing different Airlines',
                               autopct='%1.0f%%')
```

Out[8]: <AxesSubplot:title={'center':'Pie Plot containing different Airlines'}, ylabel='airline'>



In [9]:
```
airline_tweets.groupby('airline')['airline'].count().plot(kind='bar',
                                title='Bar Plot containing different Airlines',
                                figsize=(10,7),grid=True)
```

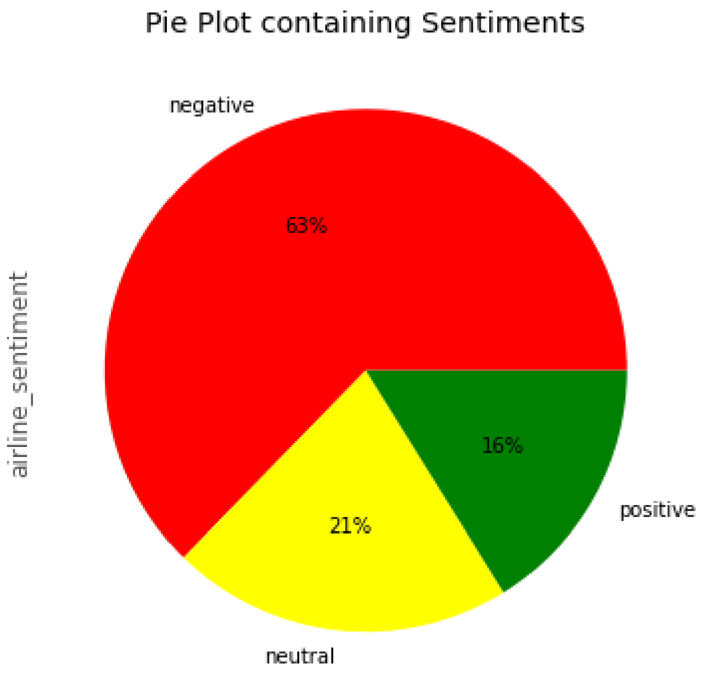
Out[9]:
```
<AxesSubplot:title={'center':'Bar Plot containing different Airlines'}, xlabel='airline'>
```



In [10]:

```
airline_tweets.airline_sentiment.value_counts().plot(kind='pie',
                                title='Pie Plot containing Sentiments',
                                autopct='%1.0f%%',
                                colors=["red", "yellow", "green"])
```

Out[10]: <AxesSubplot:title={'center':'Pie Plot containing Sentiments'}, ylabel='airline_sentiment'>



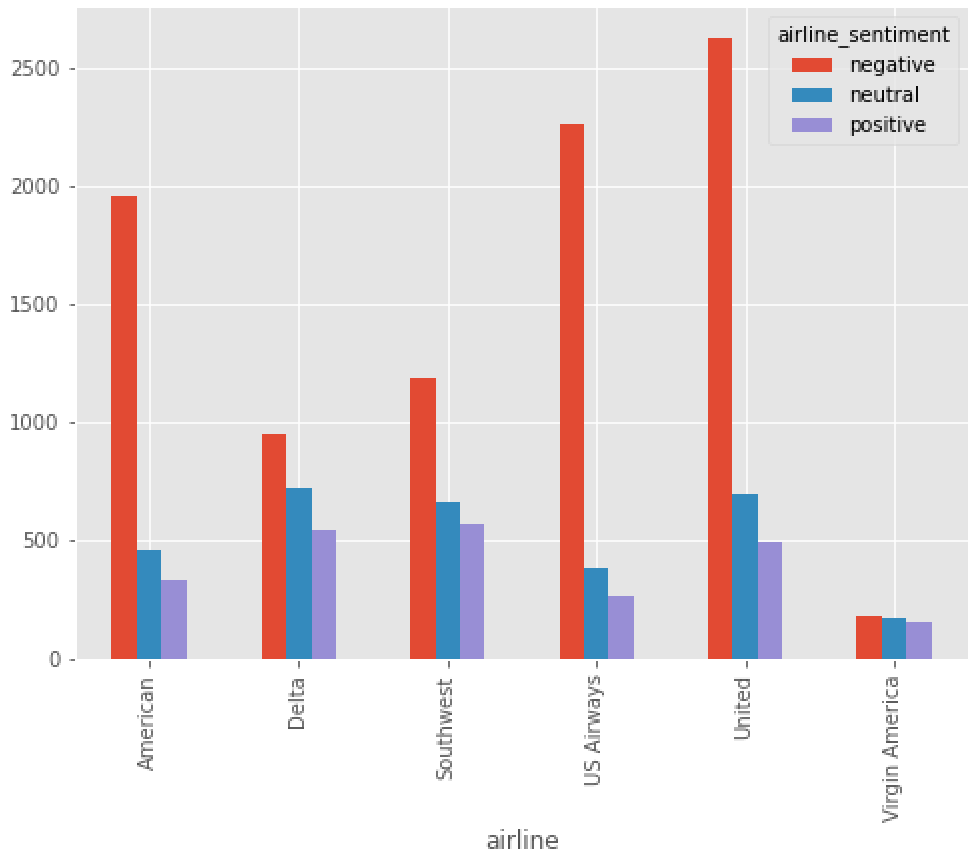
In [11]:

```
airline_tweets.groupby('airline_sentiment')['airline_sentiment'].count().plot(kind='bar',
                                                                        title='Target class',
                                                                        figsize=(10,7),
                                                                        grid=True)
```

Out[11]: <AxesSubplot:title={'center':'Target class'}, xlabel='airline_sentiment'>

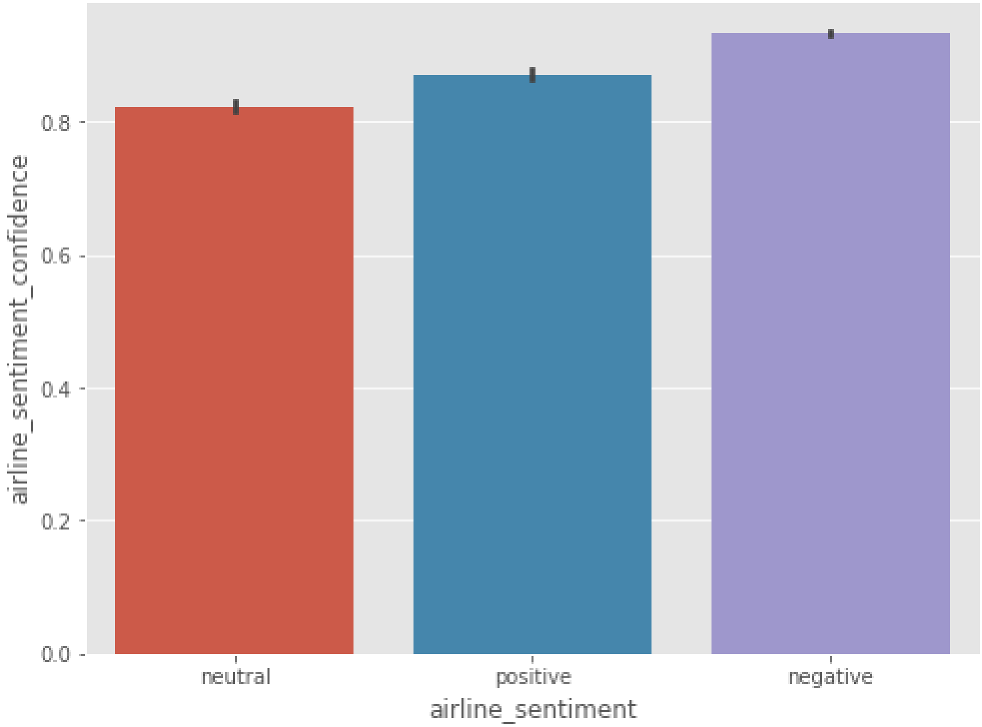
In [12]:
```
airline_sentiment = airline_tweets.groupby(['airline', 'airline_sentiment']).airline_sentiment.count().unstack()
airline_sentiment.plot(kind='bar')
```

Out[12]: <AxesSubplot:xlabel='airline'>



In [13]:
```
sns.barplot(x='airline_sentiment',
            y='airline_sentiment_confidence' ,
            data=airline_tweets)
```

Out[13]: <AxesSubplot:xlabel='airline_sentiment', ylabel='airline_sentiment_confidence'>



Data Pre-Processing

Removing Punctuation, Numbers, Special Characters and Short Words

In [14]:

```python
def remove_pattern(text,pattern):

    r = re.findall(pattern,text)

    for i in r:
        text = re.sub(i,"",text)

    return text
```

In [15]:

```python
airline_tweets['text'] = np.vectorize(remove_pattern)(airline_tweets['text'], "@[\w]*")

airline_tweets.head()
```

Out[15]:

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	airline_sentiment_gold
0	570306133677760513	neutral	1.0000	NaN	NaN	Virgin America	NaN
1	570301130888122368	positive	0.3486	NaN	0.0000	Virgin America	NaN
2	570301083672813571	neutral	0.6837	NaN	NaN	Virgin America	NaN
3	570301031407624196	negative	1.0000	Bad Flight	0.7033	Virgin America	NaN
4	570300817074462722	negative	1.0000	Can't Tell	1.0000	Virgin America	NaN

^[a-zA-Z] means any a-z or A-Z at the start of a line

[^a-zA-Z] means any character that IS NOT a-z OR A-Z

the first means "match all strings that start with a letter", the second means "match all strings that contain a non-letter". The caret ("^") is used in two different ways, one to signal the start of the text, one to negate a character match inside square brackets.

In [16]:

```
airline_tweets['text'] = airline_tweets['text'].str.replace("[^a-zA-Z#]", " ")

airline_tweets.head()
```

```
C:\Users\Eric\AppData\Roaming\Python\Python37\site-packages\ipykernel_launcher.py:1: FutureWarning: The default valu
e of regex will change from True to False in a future version.
  """Entry point for launching an IPython kernel.
```

Out[16]:

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	airline_sentiment_gold
0	570306133677760513	neutral	1.0000	NaN	NaN	Virgin America	NaN
1	570301130888122368	positive	0.3486	NaN	0.0000	Virgin America	NaN
2	570301083672813571	neutral	0.6837	NaN	NaN	Virgin America	NaN
3	570301031407624196	negative	1.0000	Bad Flight	0.7033	Virgin America	NaN
4	570300817074462722	negative	1.0000	Can't Tell	1.0000	Virgin America	NaN

In [17]:

```
airline_tweets['text'] = airline_tweets['text'].str.lower()

airline_tweets.head()
```

Out[17]:

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	airline_sentiment_gold
0	570306133677760513	neutral	1.0000	NaN	NaN	Virgin America	NaN
1	570301130888122368	positive	0.3486	NaN	0.0000	Virgin America	NaN
2	570301083672813571	neutral	0.6837	NaN	NaN	Virgin America	NaN
3	570301031407624196	negative	1.0000	Bad Flight	0.7033	Virgin America	NaN
4	570300817074462722	negative	1.0000	Can't Tell	1.0000	Virgin America	NaN

In [18]:

```
airline_tweets['text'] = airline_tweets['text'].apply(lambda x: ' '.join([w for w in x.split() if len(w)>3]))

airline_tweets.head()
```

Out[18]:

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	airline_sentiment_gold
0	570306133677760513	neutral	1.0000	NaN	NaN	Virgin America	NaN
1	570301130888122368	positive	0.3486	NaN	0.0000	Virgin America	NaN
2	570301083672813571	neutral	0.6837	NaN	NaN	Virgin America	NaN
3	570301031407624196	negative	1.0000	Bad Flight	0.7033	Virgin America	NaN
4	570300817074462722	negative	1.0000	Can't Tell	1.0000	Virgin America	NaN

Data Visualisation using WordCloud

In [19]:

```
from wordcloud import WordCloud,ImageColorGenerator
from PIL import Image
import urllib
import requests
```

Generating WordCloud for tweets with postive labels only

In [20]:

```python
all_positive_words = ' '.join(text for text in airline_tweets['text'][airline_tweets['airline_sentiment'] == 'pos

all_positive_words
```

Out[20]:

```
'plus added commercials experience tacky nearly every time this worm away well didn amazing arrived hour early goo
d pretty graphics much better than minimal iconography this such great deal already thinking about trip haven even
gone trip flying your #fabulous #seductive skies again take #stress away from travel http ahlxhhkiyn thanks excite
d first cross country flight heard nothing great things about virgin america daystogo flying know what would amazi
ngly awesome please want with only love this graphic http grrwaaa love hipster innovation feel good brand this gre
at news america could start flights hawaii year http moodlighting only best experience ever cool calming #moodlitm
onday done done best airline around hands down view downtown angeles hollywood sign beyond that rain mountains htt
p ibtr #elevategold good reason rock this just blew mind julie andrews though very impressive know need spotify st
at #guiltypleasures lady gaga amazing love three really beat classics congrats winning award best deals from airli
ne http iljaebv worried been great ride plane with great crew airlines should like this awesome flew yall morning 
correct bill applied more then once member #inflight crew team interested #flightattendant #dreampath amazing cust
omer service again raeann best #customerservice #virginamerica #flying getaway deals through from lots cool cities
http tzzjhuibch #cheapflights #farecompare getaway deals through from lots cool cities http #cheapflights #farecom
pare have great week come back #phl already need take this horrible cold #pleasecomeback http glxfwp best airline 
have flown easy change your reservation helpful representatives comfortable flying experience again another kicked
butt naelah represents your team beautifully thank your beautiful front design down right cool still book ticket y
our back secure love team running gate tonight waited delayed flight they kept things entertaining thanks your out
standing crew moved mountains home francisco tonight have absolute best team customer service ever every time with
delighted thank completely awesome experience last month nonstop thanks such awesome flight depart time #vabeatsjb
```

In [21]:

```python
Mask = np.array(Image.open(requests.get
                           ('http://clipart-library.com/image_gallery2/Twitter-PNG-Image.png', stream=True).raw))

image_colors = ImageColorGenerator(Mask)

wc = WordCloud(background_color='black',
               height=1500,
               width=4000,mask=Mask).generate(all_positive_words)
```

In [22]:

```
plt.figure(figsize=(10,20))

plt.imshow(wc.recolor(color_func=image_colors),interpolation="hamming")

plt.axis('off')
plt.show()
```

Generating WordCloud for tweets with negative labels only

In [23]:

```python
all_negative_words = ' '.join(text for text in airline_tweets['text'][airline_tweets['airline_sentiment'] == 'neg

all_negative_words
```

Out[23]:

```
'really aggressive blast obnoxious entertainment your guests faces they have little recourse really thing about se
riously would flight seats that didn have this playing really only thing about flying schedule still flew from las
t week couldn fully seat large gentleman either side help your first fares over three times more than other carrie
rs when seats available select guys messed seating reserved seating with friends guys gave seat away want free int
ernet status match program applied been three weeks called emailed with response what happened vegan food options
least site know able anything next #fail amazing that cold from vents #noair #worstflightever #roasted #sfotobos j
ust bked cool birthday trip with elevate cause entered middle name during flight booking problems help left expens
ive headphones flight today seat answering number awaiting return phone call just would prefer your online self se
rvice option your chat support working your site http gtdwpk first time flyer next week excited having hard time g
etting flights added elevate account help excited about your deal been trying book since last week page never load
s called weeks about adding flights from elevate they still haven shown help heyyyy guyyyys been trying through ho
ur someone call please virgin hold minutes there earlier flights from tonight earlier than everything fine until l
ost your airline awesome your loft needs step game dirty tables floors http vrfhjht what going with customer servi
ce there anyway speak human asap thank supp traveler like have customer service like #neverflyvirginforbusiness be
st whenever begrudgingly other airline delayed late flight have interesting flying with after this will cancelled
flight next four flights planned #neverflyvirginforbusiness disappointing experience which will shared with every
business traveler meet #neverflyvirgin having trouble adding this flight wife booked elevate account help http hqo
ks site down when will back like interesting video just disappointed cancelled flightled flight when other flights
went saturday just landed hour after should been here your late flight check business travel friendly #nomorevirgi
```

In [24]:

```python
Mask = np.array(Image.open(requests.get('http://clipart-library.com/image_gallery2/Twitter-PNG-Image.png', stream

image_colors = ImageColorGenerator(Mask)

wc = WordCloud(background_color='black',
               height=1500, width=4000,mask=Mask).generate(all_negative_words)
```

In [25]:
```
plt.figure(figsize=(10,20))

plt.imshow(wc.recolor(color_func=image_colors),interpolation="gaussian")

plt.axis('off')
plt.show()
```

Generating **TFIDF** and **CountVectorizer**

In [26]:
```python
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
```

max_df is used for removing terms that appear too frequently, also known as "corpus-specific stop words". For example:

max_df = 0.50 means "ignore terms that appear in more than 50% of the documents". max_df = 25 means "ignore terms that appear in more than 25 documents". The default max_df is 1.0, which means "ignore terms that appear in more than 100% of the documents". Thus, the default setting does not ignore any terms.

min_df is used for removing terms that appear too infrequently. For example:

min_df = 0.01 means "ignore terms that appear in less than 1% of the documents". min_df = 5 means "ignore terms that appear in less than 5 documents". The default min_df is 1, which means "ignore terms that appear in less than 1 document".

In [27]:
```python
vectorizer = CountVectorizer(max_df=0.90, min_df=2,
                             max_features=2500, stop_words = 'english')

airline_vector = vectorizer.fit_transform(airline_tweets['text'])

airline_vector_df = pd.DataFrame(airline_vector.todense())

airline_vector_df.shape
```

Out[27]: (14640, 2500)

In [28]:
```python
tfidf = TfidfVectorizer(max_df=0.90, min_df=2,
                        max_features=2500,stop_words='english')

airline_tfidf = tfidf.fit_transform(airline_tweets['text'])

airline_tfidf_df = pd.DataFrame(airline_tfidf.todense())

airline_tfidf_df.shape
```

Out[28]: (14640, 2500)

Importing necessary packages

In [29]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
```

Splitting our dataset into Training and Validation Set for both TFIDF and CountVectorizer

In [30]:
```
X_train,X_test,Y_train,Y_test = train_test_split(airline_vector,airline_tweets['sentiment_class'],test_size=0.3,

X_train.shape,Y_train.shape,X_test.shape,Y_test.shape
```

Out[30]: ((10248, 2500), (10248,), (4392, 2500), (4392,))

In [31]:
```
x_train,x_test,y_train,y_test = train_test_split(airline_tfidf,airline_tweets['sentiment_class'],test_size=0.3, r

x_train.shape,y_train.shape,x_test.shape,y_test.shape
```

Out[31]: ((10248, 2500), (10248,), (4392, 2500), (4392,))

LogisticRegression

In [32]:
```
Log_Reg = LogisticRegression(random_state=0,solver='lbfgs')
```

In [33]:

```
Log_Reg.fit(X_train,Y_train)
Log_Reg.fit(x_train,y_train)
```

```
C:\Users\Eric\anaconda3\envs\new_env\lib\site-packages\sklearn\linear_model\_logistic.py:765: ConvergenceWarning: lb
fgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessin
g.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/m
odules/linear_model.html#logistic-regression)
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
C:\Users\Eric\anaconda3\envs\new_env\lib\site-packages\sklearn\linear_model\_logistic.py:765: ConvergenceWarning: lb
fgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessin
g.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/m
odules/linear_model.html#logistic-regression)
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[33]:

```
LogisticRegression(random_state=0)
```

In [34]:

```
log_predict_vector = Log_Reg.predict(X_test)
log_predict_tfidf = Log_Reg.predict(x_test)
```

In [35]:

```python
print("########### Scores for CountVectorizer ###########","\n")
print("_______________________________________________________")
print(confusion_matrix(Y_test,log_predict_vector))
print(classification_report(Y_test,log_predict_vector))
print(accuracy_score(Y_test, log_predict_vector))
print("_______________________________________________________")
```

```
########### Scores for CountVectorizer ###########

_______________________________________________________
[[2427  163  141]
 [ 442  381  138]
 [ 131   52  517]]
              precision    recall  f1-score   support

           0       0.81      0.89      0.85      2731
           3       0.64      0.40      0.49       961
           4       0.65      0.74      0.69       700

    accuracy                           0.76      4392
   macro avg       0.70      0.67      0.68      4392
weighted avg       0.75      0.76      0.74      4392

0.7570582877959927
_______________________________________________________
```

In [36]:
```
print("########### Scores for TFIDF ###########","\n")
print("_______________________________________________________")
print(confusion_matrix(y_test,log_predict_tfidf))
print(classification_report(y_test,log_predict_tfidf))
print(accuracy_score(y_test, log_predict_tfidf))
print("_______________________________________________________")
```

```
########### Scores for TFIDF ###########

_______________________________________________________
[[2559  134   38]
 [ 513  395   53]
 [ 199   93  408]]
              precision    recall  f1-score   support

           0       0.78      0.94      0.85      2731
           3       0.64      0.41      0.50       961
           4       0.82      0.58      0.68       700

    accuracy                           0.77      4392
   macro avg       0.75      0.64      0.68      4392
weighted avg       0.76      0.77      0.75      4392

0.7654826958105647
_______________________________________________________
```

RandomForestClassifier

In [ ]:

```
text_classifier = RandomForestClassifier(n_estimators=50, random_state=0)
text_classifier.fit(X_train, Y_train)
text_classifier.fit(x_train, y_train)
```

Out[87]:

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=50,
                       n_jobs=None, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)
```

In [ ]:

```
ran_pred_vector = text_classifier.predict(X_test)
ran_pred_tfidf = text_classifier.predict(x_test)
```

In [ ]:

```
print("########### Scores for CountVectorizer ###########","\n")
print("_______________________________________________________")
print(confusion_matrix(Y_test,ran_pred_vector))
print(classification_report(Y_test,ran_pred_vector))
print(accuracy_score(Y_test, ran_pred_vector))
print("_______________________________________________________")
```

```
########### Scores for CountVectorizer ###########


_______________________________________________________
[[1916  667  148]
 [ 219  623  119]
 [  81  167  452]]
              precision    recall  f1-score   support

           0       0.86      0.70      0.77      2731
           3       0.43      0.65      0.52       961
           4       0.63      0.65      0.64       700

    accuracy                           0.68      4392
   macro avg       0.64      0.67      0.64      4392
weighted avg       0.73      0.68      0.70      4392

0.6810109289617486
_______________________________________________________
```

In [ ]:

```
print("########### Scores for TFIDF ###########","\n")
print("_______________________________________________________")
print(confusion_matrix(y_test,ran_pred_tfidf))
print(classification_report(y_test,ran_pred_tfidf))
print(accuracy_score(y_test, ran_pred_tfidf))
print("_______________________________________________________")
```

```
########### Scores for TFIDF ###########

_______________________________________________________
[[2433  215   83]
 [ 452  403  106]
 [ 193   87  420]]
              precision    recall  f1-score   support

           0       0.79      0.89      0.84      2731
           3       0.57      0.42      0.48       961
           4       0.69      0.60      0.64       700

    accuracy                           0.74      4392
   macro avg       0.68      0.64      0.65      4392
weighted avg       0.73      0.74      0.73      4392

0.7413479052823315
_______________________________________________________
```

LinearSVC

In [ ]:

```
clf_1 = LinearSVC()
```

In [ ]:

```
clf_1.fit(X_train, Y_train)
clf_1.fit(x_train, y_train)
```

Out[92]:

```
LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
```

In [ ]:

```
svc_pred_vector = clf_1.predict(X_test)
svc_pred_tfidf = clf_1.predict(x_test)
```

In [ ]:

```
print("########### Scores for CountVectorizer ###########","\n")
print("_______________________________________________________")
print(confusion_matrix(Y_test,svc_pred_vector))
print(classification_report(Y_test,svc_pred_vector))
print(accuracy_score(Y_test, svc_pred_vector))
print("_______________________________________________________")
```

```
########### Scores for CountVectorizer ###########


_______________________________________________________
[[2441  151  139]
 [ 400  403  158]
 [ 111   63  526]]
              precision    recall  f1-score   support

           0       0.83      0.89      0.86      2731
           3       0.65      0.42      0.51       961
           4       0.64      0.75      0.69       700

    accuracy                           0.77      4392
   macro avg       0.71      0.69      0.69      4392
weighted avg       0.76      0.77      0.76      4392

0.767304189435337
_______________________________________________________
```

In [ ]:

```
print("########### Scores for TFIDF ###########","\n")
print("_______________________________________________________")
print(confusion_matrix(y_test,svc_pred_tfidf))
print(classification_report(y_test,svc_pred_tfidf))
print(accuracy_score(y_test, svc_pred_tfidf))
print("_______________________________________________________")
```

```
########### Scores for TFIDF ###########


_______________________________________________________
[[2439  221   71]
 [ 384  493   84]
 [ 143  108  449]]
              precision    recall  f1-score   support

           0       0.82      0.89      0.86      2731
           3       0.60      0.51      0.55       961
           4       0.74      0.64      0.69       700

    accuracy                           0.77      4392
   macro avg       0.72      0.68      0.70      4392
weighted avg       0.76      0.77      0.76      4392

0.769808743169399
_______________________________________________________
```

SVC

In [ ]:

```
clf_2 = SVC(kernel = 'linear')
```

In [ ]:

```
clf_2.fit(X_train, Y_train)
clf_2.fit(x_train, y_train)
```

Out[97]:

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [ ]:

```
svm_pred_vector = clf_2.predict(X_test)
svm_pred_tfidf = clf_2.predict(x_test)
```

In [ ]:

```
print("########### Scores for CountVectorizer ###########","\n")
print("_______________________________________________________")
print(confusion_matrix(Y_test,svm_pred_vector))
print(classification_report(Y_test,svm_pred_vector))
print(accuracy_score(Y_test, svm_pred_vector))
print("_______________________________________________________")
```

```
########### Scores for CountVectorizer ###########

_______________________________________________________
[[2395  157  179]
 [ 462  346  153]
 [ 110   49  541]]
              precision    recall  f1-score   support

           0       0.81      0.88      0.84      2731
           3       0.63      0.36      0.46       961
           4       0.62      0.77      0.69       700

    accuracy                           0.75      4392
   macro avg       0.68      0.67      0.66      4392
weighted avg       0.74      0.75      0.73      4392

0.7472677595628415
_______________________________________________________
```

In [ ]:

```python
print("########### Scores for TFIDF ###########","\n")
print("_______________________________________________________")
print(confusion_matrix(y_test,svm_pred_tfidf))
print(classification_report(y_test,svm_pred_tfidf))
print(accuracy_score(y_test, svm_pred_tfidf))
print("_______________________________________________________")
```

```
########### Scores for TFIDF ###########

_______________________________________________________
[[2523  149   59]
 [ 517  384   60]
 [ 188   73  439]]
              precision    recall  f1-score   support

           0       0.78      0.92      0.85      2731
           3       0.63      0.40      0.49       961
           4       0.79      0.63      0.70       700

    accuracy                           0.76      4392
   macro avg       0.73      0.65      0.68      4392
weighted avg       0.75      0.76      0.75      4392

0.7618397085610202
_______________________________________________________
```