Airline Tweets Data Project / Yeoman Yoon

In [1]:
```
# conda install -c conda-forge spacy
# $ conda update -n base -c defaults conda
```

In [2]:
```
import re
import numpy as np                                    #for large and multi-dimen
sional arrays
import pandas as pd                                   #for data manipulation and
analysis
import nltk                                           #Natural language processi
ng tool-kit

# nltk.download('stopwords')    #download stopwords
# nltk.download('punkt')

from nltk.corpus import stopwords                     #Stopwords corpus
from nltk.stem import PorterStemmer                   # Stemmer
from nltk.tokenize import word_tokenize

from sklearn.feature_extraction.text import CountVectorizer          #For Bag
of words
from sklearn.feature_extraction.text import TfidfVectorizer          #For TF-I
DF

# !pip install vaderSentiment    # Install vader sentiment package
# !pip install textblob   # Install textblob package

# install and import necessary libraries.

# !pip install contractions

import string, unicodedata                            # Import Regex, string and
unicodedata.
import contractions                                       # Import contractions
library.
from bs4 import BeautifulSoup                             # Import BeautifulSou
p.

from nltk.corpus import stopwords                         # Import stopwords.
import nltk
nltk.download('stopwords')

from nltk.tokenize import word_tokenize, sent_tokenize  # Import Tokenizer.
from nltk.stem.wordnet import WordNetLemmatizer         # Import Lemmatizer.
import matplotlib.pyplot as plt
from sklearn.model_selection import KFold , StratifiedKFold, cross_val_score
from sklearn.metrics import accuracy_score
import seaborn as sns


from sklearn.ensemble import RandomForestClassifier       # Import Random fore
st Classifier
from sklearn.metrics import classification_report         # Import Classificat
ion report
from sklearn.model_selection import cross_val_score

from wordcloud import WordCloud,STOPWORDS
```

```
# Ignore the warnings
import warnings
warnings.filterwarnings("ignore")
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\yeoma\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Opinion on problem statement:

There are many ways to enhance products and service. Yet, it is most critial to hear customers' voice. Although the platform tweeter may not represent the whole customers, it is a good start to evaluate on the customer reviews and see what can be improved in the product or service.

Import Data

In [3]:
```
tweets = pd.read_csv("tweets.csv") # use copy to store the original data
data = tweets.copy()
```

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

Out[4]:

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativere
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	

In [5]:
```
print(f"The data has {data.shape[0]} rows and {data.shape[1]} columns")
```

```
The data has 14640 rows and 15 columns
```

EDA

In [6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14640 entries, 0 to 14639
Data columns (total 15 columns):
tweet_id                      14640 non-null int64
airline_sentiment             14640 non-null object
airline_sentiment_confidence  14640 non-null float64
negativereason                9178 non-null object
negativereason_confidence     10522 non-null float64
airline                       14640 non-null object
airline_sentiment_gold        40 non-null object
name                          14640 non-null object
negativereason_gold           32 non-null object
retweet_count                 14640 non-null int64
text                          14640 non-null object
tweet_coord                   1019 non-null object
tweet_created                 14640 non-null object
tweet_location                9907 non-null object
user_timezone                 9820 non-null object
dtypes: float64(2), int64(2), object(11)
memory usage: 1.7+ MB
```

In [7]:
```
data.describe()
```

Out[7]:

	tweet_id	airline_sentiment_confidence	negativereason_confidence	retweet_count
count	1.464000e+04	14640.000000	10522.000000	14640.000000
mean	5.692184e+17	0.900169	0.638298	0.082650
std	7.791112e+14	0.162830	0.330440	0.745778
min	5.675883e+17	0.335000	0.000000	0.000000
25%	5.685592e+17	0.692300	0.360600	0.000000
50%	5.694779e+17	1.000000	0.670600	0.000000
75%	5.698905e+17	1.000000	1.000000	0.000000
max	5.703106e+17	1.000000	1.000000	44.000000

In [8]:
```
data.nunique().sort_values(ascending=False)
```

Out[8]:
```
tweet_id                          14485
text                              14427
tweet_created                     14247
name                               7701
tweet_location                     3081
negativereason_confidence          1410
airline_sentiment_confidence       1023
tweet_coord                         832
user_timezone                        85
retweet_count                        18
negativereason_gold                  13
negativereason                       10
airline                               6
airline_sentiment_gold                3
airline_sentiment                     3
dtype: int64
```

Missing Value Treatment

In [9]:
```
data.isnull().sum()
```

Out[9]:
```
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
```

We have significant amount of missing variables. gold, coord, location , and timezone are irrelatvent in our research. However, negativereason and negativereason_confidence may need more study.

In [10]:
```
data[data['airline_sentiment']=='negative'].shape
```

Out[10]:
```
(9178, 15)
```

In [11]:
```
data[data['airline_sentiment']=='negative'].isnull().sum()
```

Out[11]:
```
tweet_id                          0
airline_sentiment                 0
airline_sentiment_confidence      0
negativereason                    0
negativereason_confidence         0
airline                           0
airline_sentiment_gold         9146
name                              0
negativereason_gold            9146
retweet_count                     0
text                              0
tweet_coord                    8515
tweet_created                     0
tweet_location                 3142
user_timezone                  3170
dtype: int64
```

In [12]:
```
data[data['airline_sentiment']=='positive'].shape
```

Out[12]:
```
(2363, 15)
```

In [13]:
```
data[data['airline_sentiment']=='positive'].isnull().sum()
```

Out[13]:
```
tweet_id                          0
airline_sentiment                 0
airline_sentiment_confidence      0
negativereason                 2363
negativereason_confidence      2033
airline                           0
airline_sentiment_gold         2358
name                              0
negativereason_gold            2363
retweet_count                     0
text                              0
tweet_coord                    2188
tweet_created                     0
tweet_location                  629
user_timezone                   679
dtype: int64
```

In [14]:
```
data[data['airline_sentiment']=='neutral'].shape
```

Out[14]:
```
(3099, 15)
```

In [15]:
```
data[data['airline_sentiment']=='neutral'].isnull().sum()
```

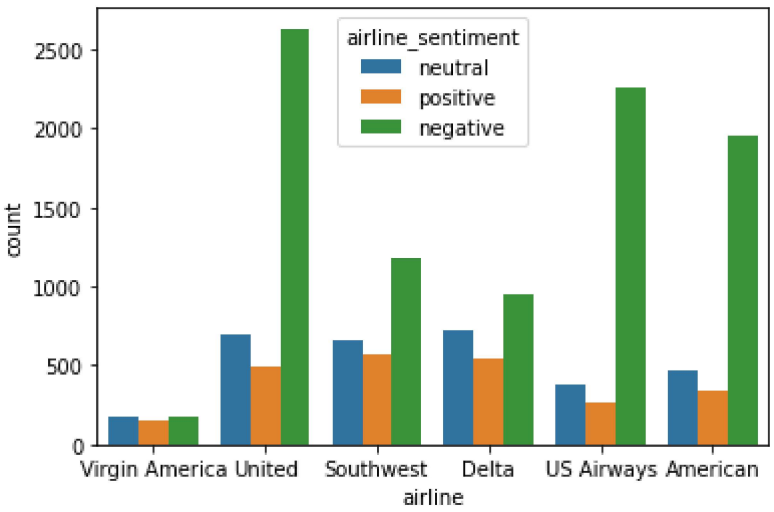
Out[15]:
```
tweet_id                          0
airline_sentiment                 0
airline_sentiment_confidence      0
negativereason                 3099
negativereason_confidence      2085
airline                           0
airline_sentiment_gold         3096
name                              0
negativereason_gold            3099
retweet_count                     0
text                              0
tweet_coord                    2918
tweet_created                     0
tweet_location                  962
user_timezone                   971
dtype: int64
```

In [16]:
```
# data[data['negativereason_confidence']==0]['airline_sentiment'].nunique() #
if negative reason is 0 then it's not negative
# data[data['negativereason_confidence']==0]['airline_sentiment'].shape # 5462
-4118 = 1344
```

As expected, negativereason is present only when sentiment is negative. 0s and NaN means the same in negativereason_confidence

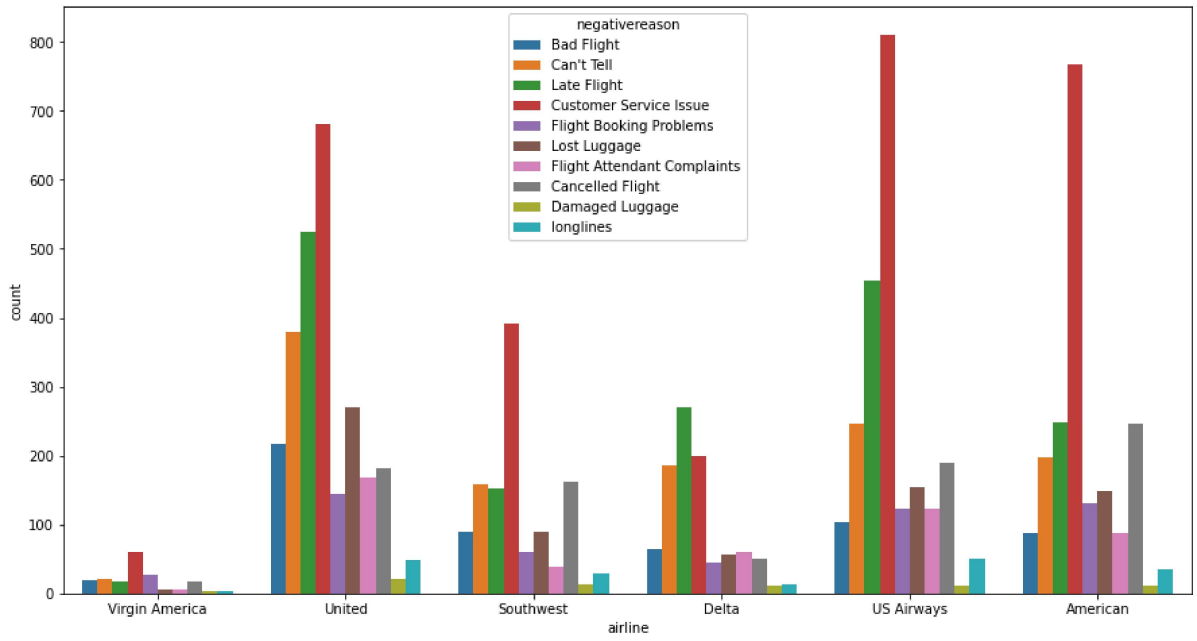
a. Plot the distribution of all tweets among each airline & plot the distribution of sentiment across all the tweets.

In [17]:
```
sns.countplot(data['airline'], hue = data['airline_sentiment'])
plt.show()
```



b. Plot the distribution of Sentiment of tweets for each airline & plot the distribution of all the negative reasons.

In [18]:
```
plt.figure(figsize = (15,8))
sns.countplot(data['airline'], hue = data['negativereason'])
plt.show()
```

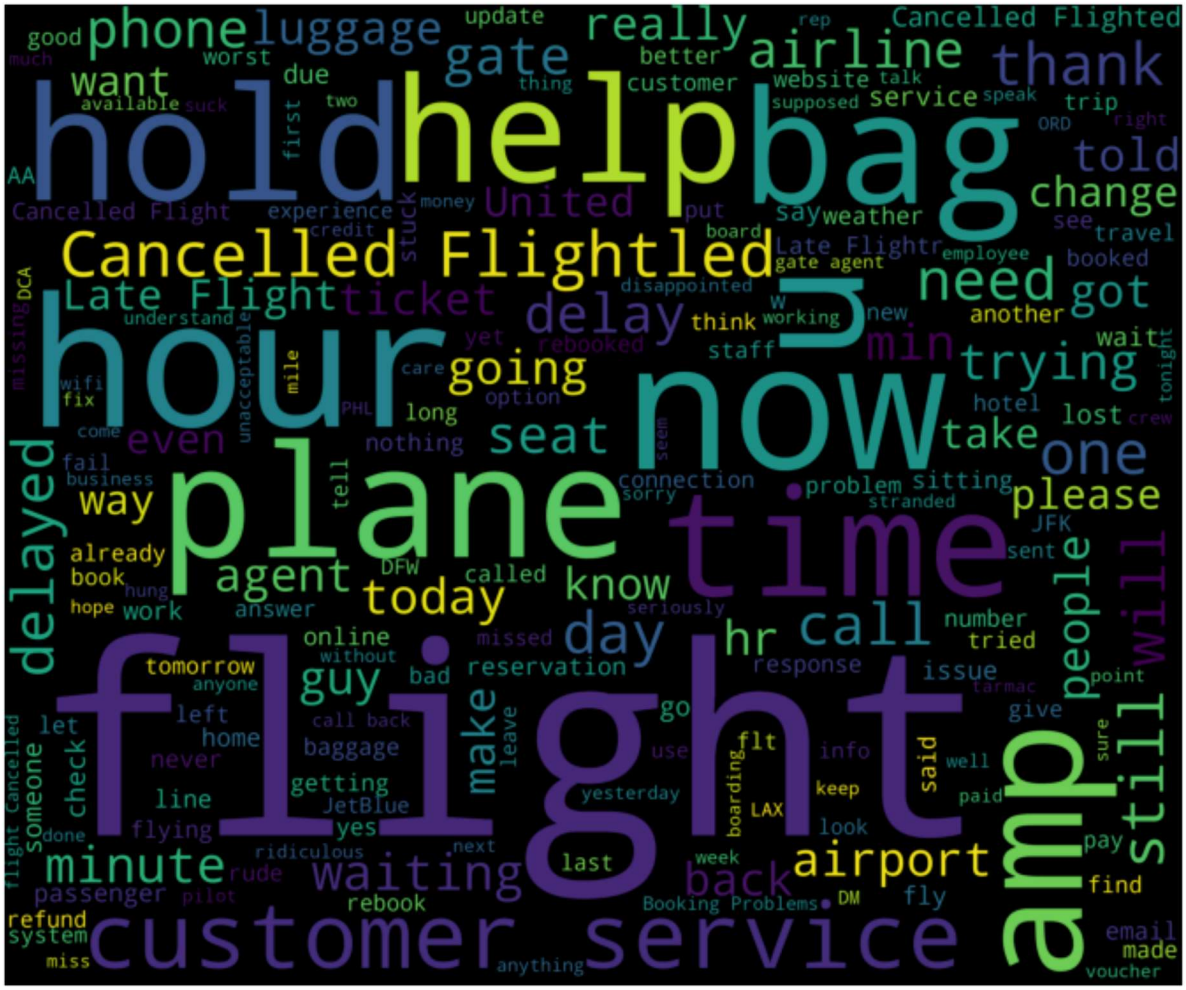


c. Plot the word cloud graph of tweets for positive and negative sentiment separately.

In [19]:
```
negative_only=data[data['airline_sentiment']=='negative']
negative_words = ' '.join(negative_only['text'])
separated_negative_word = ' '.join([word for word in negative_words.split()
                            if 'http' not in word
                                and not word.startswith('@')
                                and word != 'RT'
                            ])
```

In [20]:
```
negative_wordcloud = WordCloud(stopwords=STOPWORDS,
                        background_color='black',
                        width=3000,
                        height=2500
                       ).generate(separated_negative_word)
```

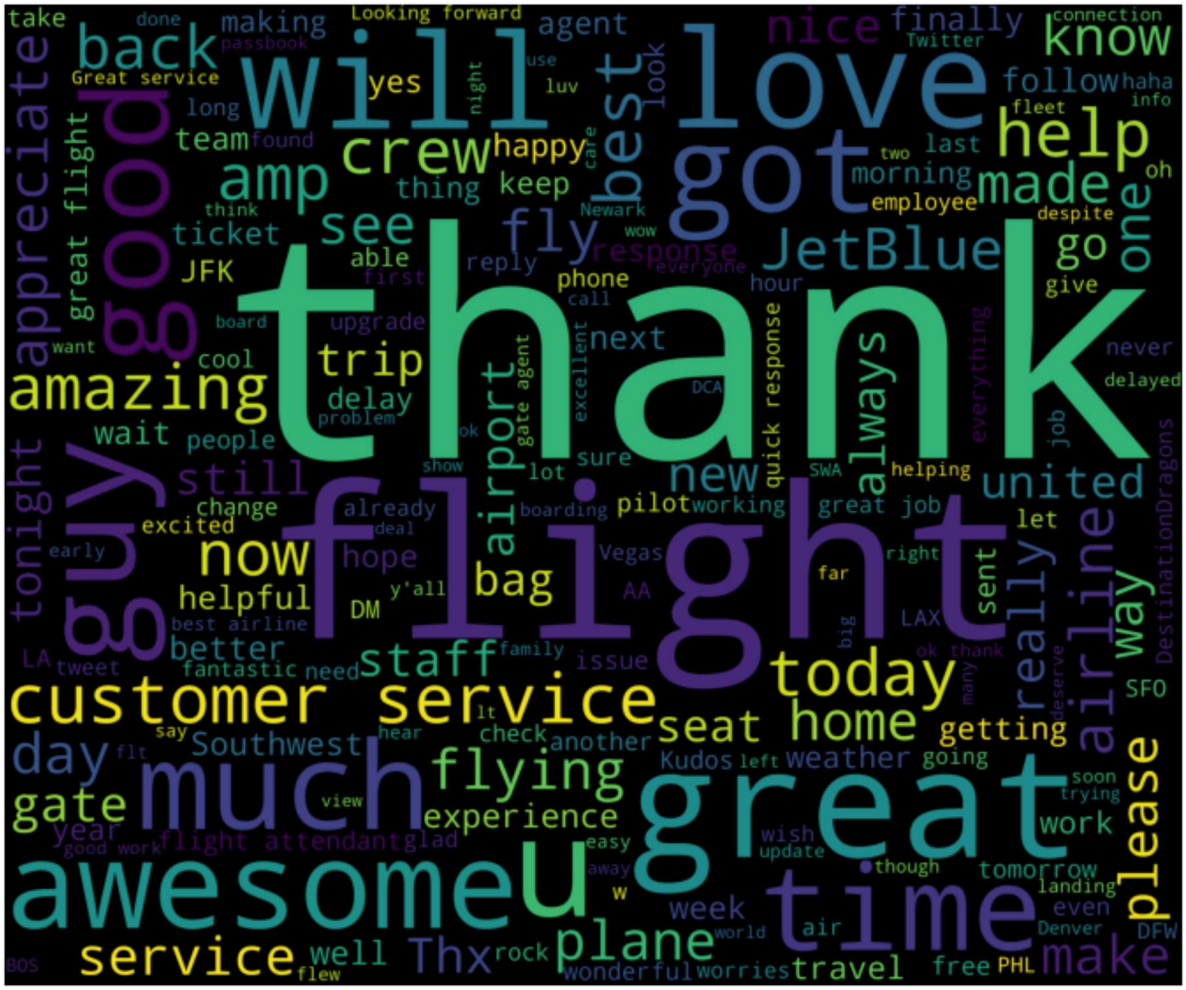
In [21]:
```
plt.figure(1,figsize=(12, 12))
plt.imshow(negative_wordcloud)
plt.axis('off')
plt.show()
```



In [22]:
```
positive_only=data[data['airline_sentiment']=='positive']
positive_words = ' '.join(positive_only['text'])
separated_positive_word = ' '.join([word for word in positive_words.split()
                            if 'http' not in word
                                and not word.startswith('@')
                                and word != 'RT'
                            ])
```

In [23]:
```
positive_wordcloud = WordCloud(stopwords=STOPWORDS,
                       background_color='black',
                       width=3000,
                       height=2500
                      ).generate(separated_positive_word)
```

In [24]:
```
plt.figure(1,figsize=(12, 12))
plt.imshow(positive_wordcloud)
plt.axis('off')
plt.show()
```



Understand Data

In [25]:
```
data_original = data # run once
```

In [26]:
```
data = data[['airline_sentiment','text']]
```

In [27]:
```
print(f"The data has {data.shape[0]} rows and {data.shape[1]} columns")
```

```
The data has 14640 rows and 2 columns
```

In [28]:
```
# display all text
pd.set_option('display.max_colwidth', 0) # ?pd.set_option

data.head()
```

Out[28]:

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

Text pre-processing

Cleaning the Text

In [29]:
```
#remove the html tags
def remove_html(text):
    soup = BeautifulSoup(text, "html.parser")
    return soup.get_text()

#expand the contractions
def replace_contractions(text):
    return contractions.fix(text)

#remove the numericals present in the text
def remove_numbers(text):
  text = re.sub(r'\d+', '', text)
  return text

# remove the url's present in the text
def remove_url(text):
    text = re.sub('http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+','',text)
    return text

# remove the mentions in the tweets
def remove_mention(text):
    text = re.sub(r'@\w+','',text)
    return text

def clean_text(text):
    text = remove_html(text)
    text = replace_contractions(text)
    text = remove_numbers(text)
    text = remove_url(text)
    text = remove_mention(text)
```

```
    return text

data['text'] = data['text'].apply(lambda x: clean_text(x))
data.head()
```

Out[29]:

	airline_sentiment	text
0	neutral	What said.
1	positive	plus you have added commercials to the experience... tacky.
2	neutral	I did not today... Must mean I need to take another trip!
3	negative	it is really aggressive to blast obnoxious "entertainment" in your guests' faces & they have little recourse
4	negative	and it is a really big bad thing about it

Tokenization

In [30]:
```
data['text'] = data.apply(lambda row: nltk.word_tokenize(row['text']), axis=1)
```

normalization

In [31]:
```
#remove the non-ASCII characters
def remove_non_ascii(text):
    new_text = []
    for word in text:
        new_word = unicodedata.normalize('NFKD', word).encode('ascii', 'ignor
e').decode('utf-8', 'ignore')
        new_text.append(new_word)
    return new_text

# convert to lowercase
def to_lowercase(text):
    new_text = []
    for word in text:
        new_word = word.lower()
        new_text.append(new_word)
    return new_text

# Remove the hashtags
def remove_hash(text):
   new_text = []
   for word in text:
     new_word = re.sub(r'#\w+','',word)
     if new_word != '':
       new_text.append(new_word)
   return new_text

# Remove the punctuations
def remove_punctuation(text):
    new_text = []
    for word in text:
        new_word = re.sub(r'[^\w\s]', '', word)
        if new_word != '':
            new_text.append(new_word)
    return new_text

# Remove the stopword
stopwords = stopwords.words('english')
stopwords = list(set(stopwords))
def remove_stopwords(text):
    new_text = []
    for word in text:
        if word not in stopwords:
            new_text.append(word)
    return new_text

# Lemmatize the text
lemmatizer = WordNetLemmatizer()
def lemmatize_list(text):
    new_text = []
    for word in text:
      new_text.append(lemmatizer.lemmatize(word, pos='v'))
    return new_text
```

```
def normalize(text):
    text = remove_non_ascii(text)
    text = to_lowercase(text)
    text = remove_punctuation(text)
    text = remove_stopwords(text)
    text = lemmatize_list(text)
    return ' '.join(text)

data['text'] = data.apply(lambda row: normalize(row['text']), axis=1)
```

Print the first 5 rows of data after pre-processing.

In [32]:
```
data.head()
```

Out[32]:

	airline_sentiment	text
0	neutral	say
1	positive	plus add commercials experience tacky
2	neutral	today must mean need take another trip
3	negative	really aggressive blast obnoxious entertainment guests face little recourse
4	negative	really big bad thing

Vectorization

CountVectorizer

In [33]:
```
# Vectorization (Convert text data to numbers).
from sklearn.feature_extraction.text import CountVectorizer

Count_vec = CountVectorizer(max_features=500) # Keep only 500 features.
text_features = Count_vec.fit_transform(data['text'])

text_features = text_features.toarray() # Convert the data features to array.
```

In [34]:
```
text_features.shape
```

Out[34]: (14640, 500)

In [35]:
```
X = text_features

y = data['airline_sentiment']
```

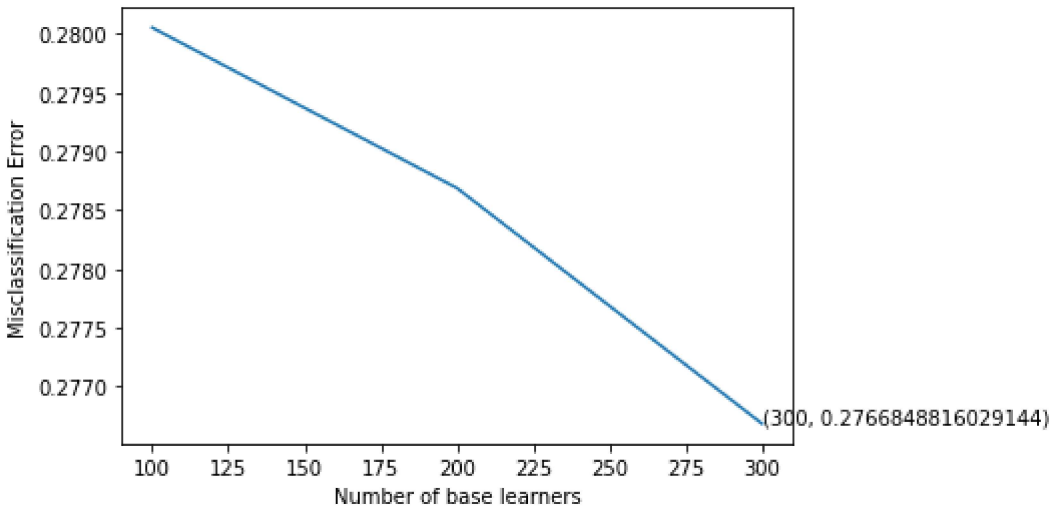
In [36]:
```
# Split data into training and testing set.

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, shuf
fle=False)
```

In [37]:
```
# Finding optimal number of base learners using k-fold CV ->
base_ln = np.arange(100,400,100).tolist()
base_ln
```

Out[37]:
```
[100, 200, 300]
```

In [38]:
```
# K-Fold Cross - validation .
cv_scores = []
for b in base_ln:
    clf = RandomForestClassifier(n_estimators = b)
    scores = cross_val_score(clf, X_train, y_train, cv = 5, scoring = 'accurac
y')
    cv_scores.append(scores.mean())
```

In [39]:
```
# plotting the error as k increases
error = [1 - x for x in cv_scores]                                  #error corr
esponds to each nu of estimator
optimal_learners = base_ln[error.index(min(error))]                 #Selection
of optimal nu of n_estimator corresponds to minimum error.
plt.plot(base_ln, error)                                            #Plot betwe
en each nu of estimator and misclassification error
xy = (optimal_learners, min(error))
plt.annotate('(%s, %s)' % xy, xy = xy, textcoords='data')
plt.xlabel("Number of base learners")
plt.ylabel("Misclassification Error")
plt.show()
```



TfidfVectorizer

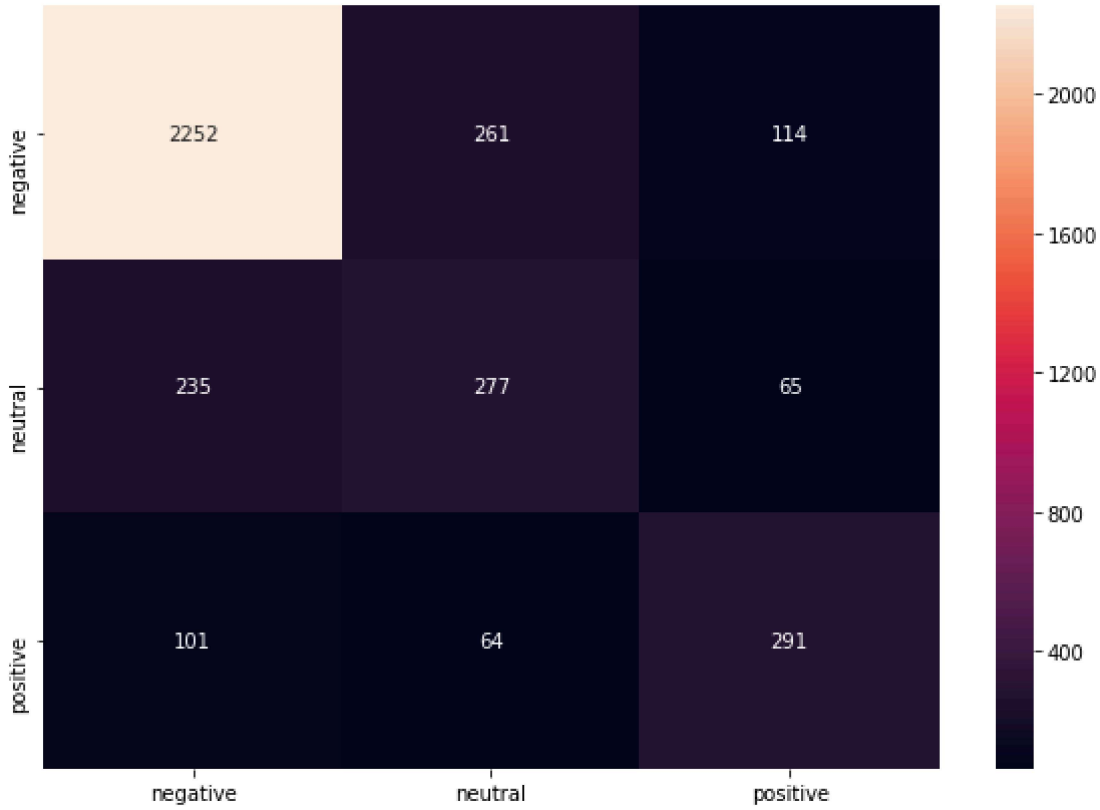
In [40]:
```
# Training the best model and calculating accuracy on test data .
clf = RandomForestClassifier(n_estimators = optimal_learners)
clf.fit(X_train, y_train)
clf.score(X_test, y_test)
count_vectorizer_predicted = clf.predict(X_test)
print(classification_report(y_test ,count_vectorizer_predicted , target_names
= ['positive' , 'neutral', 'negative']))
print("Accuracy of the model is : ",accuracy_score(y_test,count_vectorizer_pre
dicted))
```

```
              precision    recall  f1-score   support

    positive       0.87      0.86      0.86      2627
     neutral       0.46      0.48      0.47       577
    negative       0.62      0.64      0.63       456

    accuracy                           0.77      3660
   macro avg       0.65      0.66      0.65      3660
weighted avg       0.77      0.77      0.77      3660

Accuracy of the model is :  0.7704918032786885
```

In [41]:
```
# Print and plot Confusion matirx to get an idea of how the distribution of th
e prediction is, among all the classes.

import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

conf_mat = confusion_matrix(y_test, count_vectorizer_predicted)

print(conf_mat)

df_cm = pd.DataFrame(conf_mat, index = [i for i in ['negative', 'neutral', 'po
sitive']],
                     columns = [i for i in ['negative', 'neutral', 'positive']])
plt.figure(figsize = (10,7))
sns.heatmap(df_cm, annot=True, fmt='g')
```

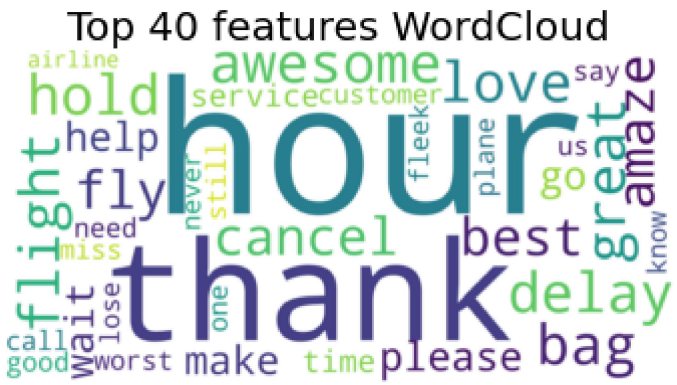
```
[[2252  261  114]
 [ 235  277   65]
 [ 101   64  291]]
```

Out[41]: <matplotlib.axes._subplots.AxesSubplot at 0x1738284af48>



In [42]:
```
all_features = Count_vec.get_feature_names()                    #Instantiate the fea
ture from the vectorizer
top_features=''                                                  # Addition of top 4
0 feature into top_feature after training the model
feat=clf.feature_importances_
features=np.argsort(feat)[::-1]
for i in features[0:40]:
    top_features+=all_features[i]
    top_features+=','

print(top_features)

print(" ")
print(" ")

from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white",colormap='viridis',width=2000,
                          height=1000).generate(top_features)

# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.figure(1, figsize=(14, 11), frameon='equal')
plt.title('Top 40 features WordCloud', fontsize=20)
plt.axis("off")
plt.show()
```

```
thank,delay,great,flight,love,get,hours,hold,bag,awesome,cancel,hour,fly,woul
d,amaze,best,please,go,wait,help,make,service,time,customer,worst,plane,call,
need,fleek,never,still,one,like,lose,us,say,good,know,miss,airline,
```



In [43]:
```
# Using TfidfVectorizer to convert text data to numbers.

from sklearn.feature_extraction.text import TfidfVectorizer

tfidf_vect = TfidfVectorizer(max_features=500)
text_features = tfidf_vect.fit_transform(data['text'])

text_features = text_features.toarray()

text_features.shape     #feature shape
```

Out[43]:
```
(14640, 500)
```

In [44]:
```
X = text_features

y = data['airline_sentiment']
```

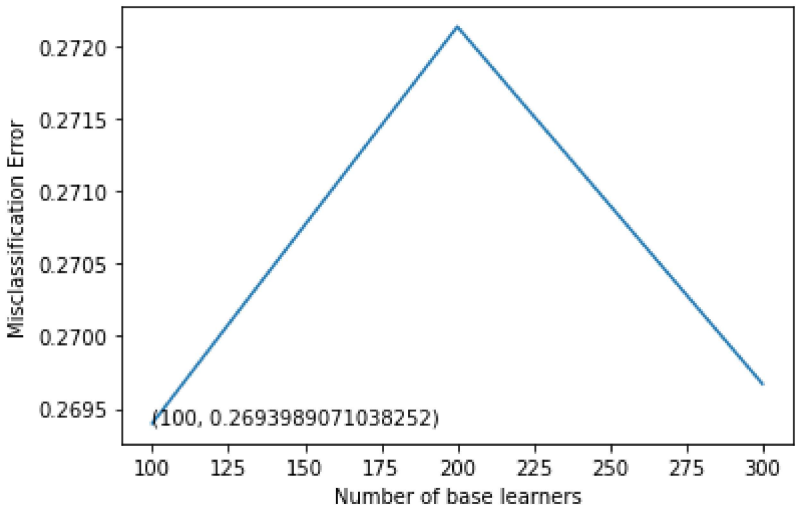
In [45]:
```
# Split data into training and testing set.

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, shuf
fle=False)
```

In [46]:
```
# Finding optimal number of base learners using k-fold CV ->
base_ln = np.arange(100,400,100).tolist()
base_ln
```

Out[46]:
```
[100, 200, 300]
```

In [47]:
```
# K-Fold Cross - validation .
cv_scores = []
for b in base_ln:
    clf = RandomForestClassifier(n_estimators = b)
    scores = cross_val_score(clf, X_train, y_train, cv = 5, scoring = 'accurac
y')
    cv_scores.append(scores.mean())
```

In [48]:
```
# plotting the error as k increases
error = [1 - x for x in cv_scores]                                  #error corr
esponds to each nu of estimator
optimal_learners = base_ln[error.index(min(error))]                 #Selection
of optimal nu of n_estimator corresponds to minimum error.
plt.plot(base_ln, error)                                            #Plot betwe
en each nu of estimator and misclassification error
xy = (optimal_learners, min(error))
plt.annotate('(%s, %s)' % xy, xy = xy, textcoords='data')
plt.xlabel("Number of base learners")
plt.ylabel("Misclassification Error")
plt.show()
```



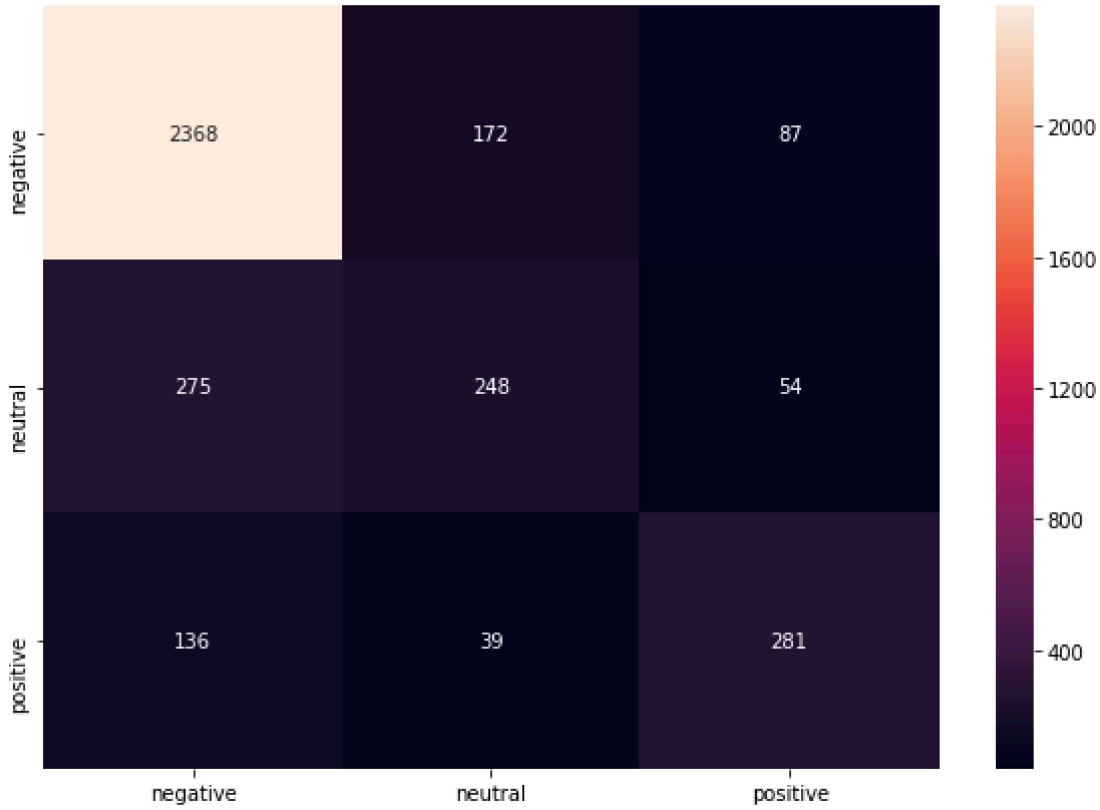
In [49]:
```
# Training the best model and calculating accuracy on test data .
clf = RandomForestClassifier(n_estimators = optimal_learners)
clf.fit(X_train, y_train)
clf.score(X_test, y_test)
tf_idf_predicted = clf.predict(X_test)
print(classification_report(y_test , tf_idf_predicted , target_names = ['negat
ive', 'neutral', 'positive']))
print("Accuracy of the model is : ",accuracy_score(y_test,tf_idf_predicted))
```

```
              precision    recall  f1-score   support

    negative       0.85      0.90      0.88      2627
     neutral       0.54      0.43      0.48       577
    positive       0.67      0.62      0.64       456

    accuracy                           0.79      3660
   macro avg       0.69      0.65      0.66      3660
weighted avg       0.78      0.79      0.78      3660

Accuracy of the model is :  0.7915300546448087
```

In [50]:
```
# Print and plot Confusion matirx to get an idea of how the distribution of th
e prediction is, among all the classes.

import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

conf_mat = confusion_matrix(y_test, tf_idf_predicted)

print(conf_mat)

df_cm = pd.DataFrame(conf_mat, index = [i for i in ['negative', 'neutral', 'po
sitive']],
                     columns = [i for i in ['negative', 'neutral', 'positive']])
plt.figure(figsize = (10,7))
sns.heatmap(df_cm, annot=True, fmt='g')
```

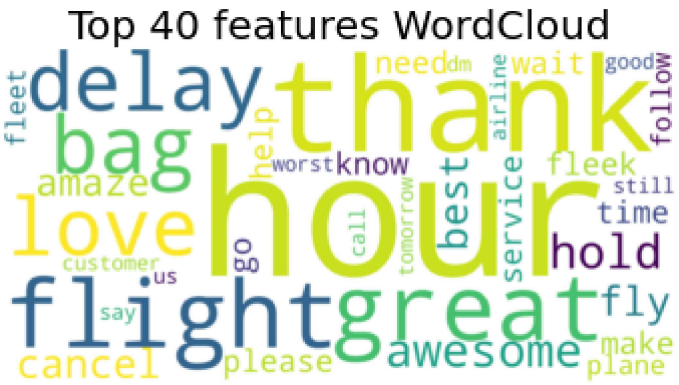
```
[[2368  172   87]
 [ 275  248   54]
 [ 136   39  281]]
```

Out[50]: <matplotlib.axes._subplots.AxesSubplot at 0x1738288f048>



In [51]:
```
all_features = tfidf_vect.get_feature_names()                    #Instantiate the fe
ature from the vectorizer
top_features=''                                                  # Addition of top 4
0 feature into top_feature after training the model
feat=clf.feature_importances_
features=np.argsort(feat)[::-1]
for i in features[0:40]:
    top_features+=all_features[i]
    top_features+=', '

print(top_features)

print(" ")
print(" ")

from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white",colormap='viridis',width=2000,
                          height=1000).generate(top_features)

# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.figure(1, figsize=(14, 11), frameon='equal')
plt.title('Top 40 features WordCloud', fontsize=20)
plt.axis("off")
plt.show()
```

thank, flight, great, delay, love, get, hours, bag, awesome, hold, cancel, fl
y, best, hour, amaze, please, time, make, service, need, would, fleek, help,
know, go, wait, plane, fleet, follow, customer, airline, call, good, dm, stil
l, tomorrow, worst, say, us, like,



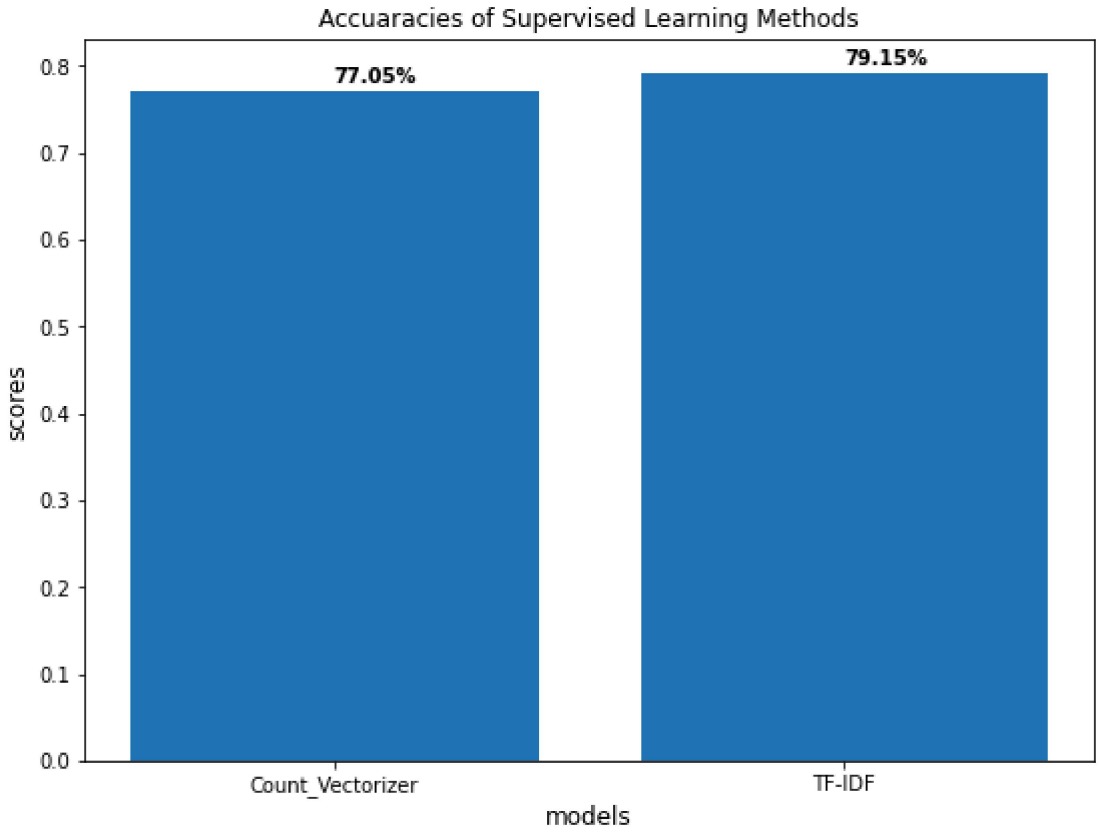
Compare

In [52]:
```
#convert the test samples
df = pd.DataFrame(y_test.tolist(),columns =['y_test'])
df['count_vectorizer_predicted'] = count_vectorizer_predicted
df['tf_idf_predicted'] = tf_idf_predicted
df.head()
```

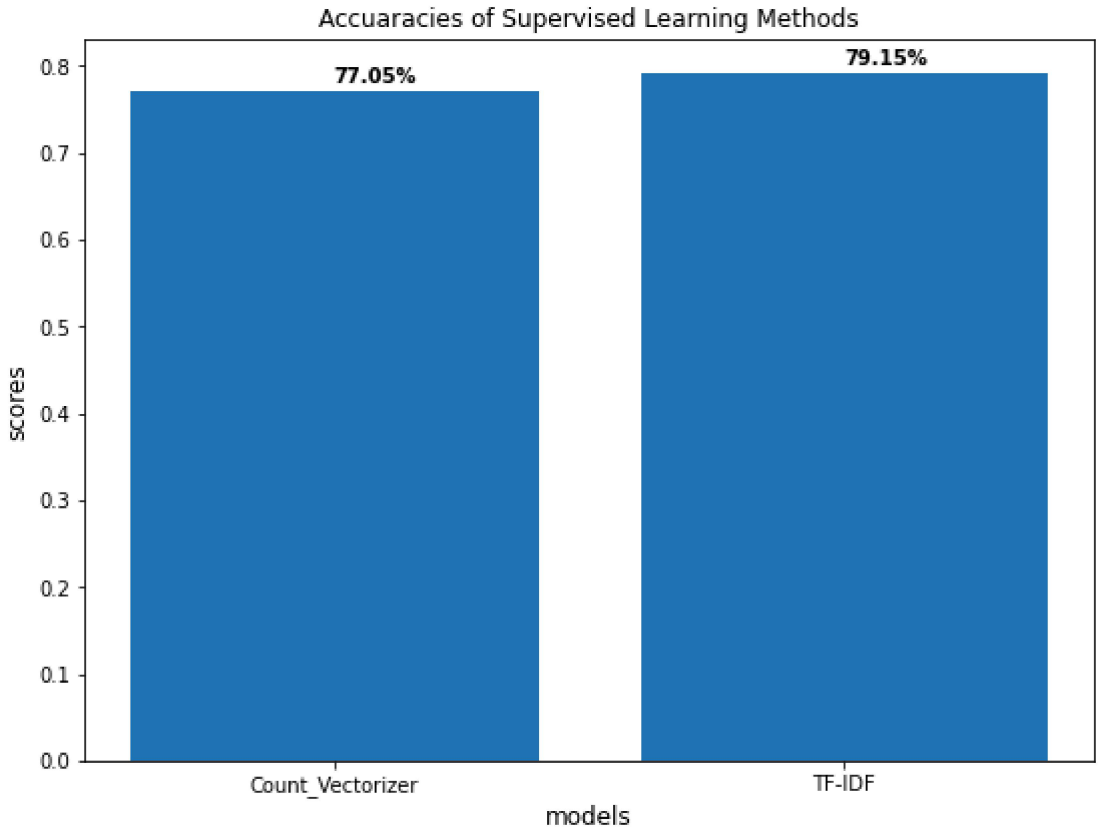
Out[52]:

	y_test	count_vectorizer_predicted	tf_idf_predicted
0	negative	negative	negative
1	negative	negative	negative
2	negative	negative	negative
3	negative	negative	negative
4	positive	negative	negative

In [53]:
```
#compare the accuaracies of Count Vectorizer and TF-IDF
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(7,5))
ax = fig.add_axes([0,0,1,1])
subjects = ['Count_Vectorizer', 'TF-IDF']

# Count Vectorizer and TF-IDF using accuracy_score metrics
scores = [accuracy_score(y_test,count_vectorizer_predicted),accuracy_score(y_t
est,tf_idf_predicted)]
ax.bar(subjects,scores)
ax.set_ylabel('scores',fontsize= 12)
ax.set_xlabel('models',fontsize= 12)
ax.set_title('Accuaracies of Supervised Learning Methods')  # title
for i, v in enumerate(scores):
    ax.text( i ,v+0.01, '{:.2f}%'.format(100*v), color='black', fontweight='bo
ld')
    plt.savefig('barplot_1.png',dpi=100, format='png', bbox_inches='tight')
plt.show()
```



In [54]:
```
# compare the accuaracies of Count Vectorizer and TF-IDF
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(7,5))
ax = fig.add_axes([0,0,1,1])
subjects = ['Count_Vectorizer', 'TF-IDF']

# Count Vectorizer and TF-IDF using accuracy_score metrics
scores = [accuracy_score(y_test,count_vectorizer_predicted),accuracy_score(y_t
est,tf_idf_predicted)]
ax.bar(subjects,scores)
ax.set_ylabel('scores',fontsize= 12)
ax.set_xlabel('models',fontsize= 12)
ax.set_title('Accuaracies of Supervised Learning Methods')  # title
for i, v in enumerate(scores):
    ax.text( i ,v+0.01, '{:.2f}%'.format(100*v), color='black', fontweight='bo
ld')
    plt.savefig('barplot_1.png',dpi=100, format='png', bbox_inches='tight')
plt.show()
```



Conclusion:

although the nlp may not be accurate as human manual reading, it is significantly required since review text is so huge. In order to improve, airlines must consider the @mention of other people agreeing the other users, sarcasm of users, emojis used, and all caps. tweets or other reviews outside of the original platform lack the rating level therefore must consider nlp to interpret customer needs.

Understading of NLP:

As a person who speaks Korean, I thought of how NLP could be applied to Korean language. The difficulty lies on slang, no punctuations and frequent error of space bar (For English every words are separated by space, yet in Korean space is very tricky that people may be wrong extremely frequently.). I think every language has its characters, tone and mood of the text may not be easily interpreted by machine (e.g good, not good, not bad, bad, good enough etc. may have different standards for people) finding numeric scales of the text must be enhanced with bigger data and patterns.

In [ ]: