# 516 FINAL PROJECT

AJ Acacio, Padmapriya Eunny, Isha Jain, Sita Sanjivini Sawhney, Riya Sirdeshmukh

In [2]:
```
#importing modules

#import praw - commented out as scraping no longer needed
import pandas as pd
import requests
from bs4 import BeautifulSoup
import nltk
from nltk.corpus import stopwords
from nltk.corpus import brown
from collections import Counter
import numpy as np
from pprint import pprint
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import os
import string, collections
import re
from string import punctuation

from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.decomposition import LatentDirichletAllocation as LDA

from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
from scipy.sparse import hstack

from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier, BaggingClassifier
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.svm import LinearSVC
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.linear_model import LogisticRegression, PassiveAggressiveClassifier
from sklearn.neighbors import RadiusNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay
import statsmodels.api as sm

import matplotlib.pyplot as plt
from sklearn.svm import SVC
from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
```

Here is the code for scraping from Reddit. Commenting it out as we froze out dataset.

In [3]:
```
# reddit = praw.Reddit(client_id= 'RjdDRvb0SRsZ6L-F4RFaAg',
#                      client_secret= '6PNorggimRykIL16HaoIttzdEYGbgg',
#                      user_agent ='516_scraper_practice')

# posts = []
# cmv = reddit.subreddit('ChangeMyView')
# for post in cmv.hot(limit=100):
#     posts.append([post.title, post.id, post.url, post.selftext, post.link_flair_text, post.score, post.num_comments])
# posts = pd.DataFrame(posts,columns=['title', 'id', 'url', 'body', 'flair', 'score', 'num_comments'])
# display(posts)
```

In [4]:
```
#Importing the .csv that was downloaded the afternoon of May 1st
posts = pd.read_csv('frozen_516_reddit_data.csv')
posts['flair'] = posts['flair'].fillna("None")
posts.head()
```

Out[4]:

| | title | body | flair | score | num_comments | id | url |
|---|---|---|---|---|---|---|---|
| 0 | CMV: Student Loans Should Not be Guaranteed by... | I was surprised to see that this topic has not... | None | 168 | 123 | ug3oa7 | https://www.reddit.com/r/changemyview/comments... |
| 1 | CMV: *IF* you believe isolation from lockdown ... | edit note: \nWhile I've already awarded delta... | Delta(s) from OP | 440 | 201 | ufqwxc | https://www.reddit.com/r/changemyview/comments... |
| 2 | CMV: US Colleges should not waste student's ti... | I went to a very competitive college in the US... | Delta(s) from OP | 1892 | 638 | ufdhhz | https://www.reddit.com/r/changemyview/comments... |
| 3 | CMV: There is Nothing Wrong with Abortion What... | We do not treat life as something that is sacr... | None | 15 | 85 | ug4uhk | https://www.reddit.com/r/changemyview/comments... |
| 4 | CMV: Western countries should invest in and su... | Admittedly, I do not know much about the compl... | Delta(s) from OP | 1270 | 140 | uf9cyg | https://www.reddit.com/r/changemyview/comments... |

In [5]:
```
#cleaning the text, creating tokenized versions
def preprocess_text(text):
    text = text.lower()
    text = re.sub(f"[{re.escape(punctuation)}]", "", text)  # Remove punctuation
    text = " ".join(text.split())  # Remove extra spaces, tabs, and new lines
    return text

posts["title_cleaned"] = posts["title"].apply(preprocess_text)
posts["body_cleaned"] = posts["body"].apply(preprocess_text)

#removing stopwords
sw = nltk.corpus.stopwords.words('english') + ['cmv:', 'cmv', 'people','us','need','give','would','dont','better','think',
                                               'shouldnt','better','life','someone','nothing','increase','good','least',
                                               'top','like','best','new','much','without','also','day','bad','one',
                                               'etc','word','get','less','really','theyre','cannot','used','make',
                                               'instead','given','say','become','isnt','take','needs','thing','never',
                                               'top','always','ive','ways','way','use','becomes','come','many','lives',
                                               'mods','alive','far','age','across']

#separate stopwords list for sentiment analysis - used later in the code
sw_senti = nltk.corpus.stopwords.words('english') + ['cmv:', 'cmv', 'people','us','need','give','would','think','life',
                                                     'someone','top','new','much','also','day','one','etc',
                                                     'instead','given','say','become','becomes','come','many','lives','states'
                                                     'mods','alive','far','age','across','believe','go','programs','time','times','states']

posts['title_withoutstop'] = posts['title_cleaned'].apply(lambda x: ' '.join([word for word in x.split() if word not in (sw)]))
posts['body_withoutstop'] = posts['body_cleaned'].apply(lambda x: ' '.join([word for word in x.split() if word not in (sw)]))

posts['tokenized_title'] = posts.apply(lambda column: nltk.word_tokenize(column['title_withoutstop']), axis=1)
posts['tokenized_body'] = posts.apply(lambda column: nltk.word_tokenize(column['body_withoutstop']), axis=1)

posts.drop(posts[posts['flair'] == 'Removed - Submission Rule E'].index, inplace = True)
posts.drop(posts[posts['flair'] == 'Removed - Submission Rule A'].index, inplace = True)
posts.drop(posts[posts['flair'] == 'Removed - Submission Rule B'].index, inplace = True)
posts.drop(posts[posts['flair'] == 'META'].index, inplace = True)

#Checking values for flair
```

```
final = pd.DataFrame(posts,columns=['flair','score','num_comments','title_withoutstop','body_withoutstop','tokenized_title','tokenized_body'])

final['flair'].unique()
mapping = {'Delta(s) from OP': 1, 'None': 0,'Delta(s) from OP - Fresh Topic Friday': 1,'Fresh Topic Friday':0}

final["flair"] = [mapping[item] for item in final.flair]

no_delta = final[final.flair == 0]
with_delta = final[final.flair == 1]
final.head()
```

Out[5]:

| | flair | score | num_comments | title_withoutstop | body_withoutstop | tokenized_title | tokenized_body |
|---|---|---|---|---|---|---|---|
| **0** | 0 | 168 | 123 | student loans guaranteed government | surprised see topic debated sub recently light... | [student, loans, guaranteed, government] | [surprised, see, topic, debated, sub, recently... |
| **1** | 1 | 440 | 201 | believe isolation lockdown mentally unhealthy ... | edit note already awarded deltas recommended f... | [believe, isolation, lockdown, mentally, unhea... | [edit, note, already, awarded, deltas, recomme... |
| **2** | 1 | 1892 | 638 | colleges waste students time useless mandatory... | went competitive college astounded number abso... | [colleges, waste, students, time, useless, man... | [went, competitive, college, astounded, number... |
| **3** | 0 | 15 | 85 | wrong abortion whatsoever | treat something sacred countries certain crimi... | [wrong, abortion, whatsoever] | [treat, something, sacred, countries, certain,... |
| **4** | 1 | 1270 | 140 | western countries invest subsidize semiconduct... | admittedly know complexities matters mind coul... | [western, countries, invest, subsidize, semico... | [admittedly, know, complexities, matters, mind... |

Descriptive Stats

In [43]:
```
#number of delta vs. non-delta posts in dataset
final.sum(axis=None) #344 out of 754
```

Out[43]:
```
flair                                                    344
score                                                  95312
num_comments                                           83415
title_withoutstop    student loans guaranteed governmentbelieve iso...
body_withoutstop     surprised see topic debated sub recently light...
tokenized_title      [student, loans, guaranteed, government, belie...
tokenized_body       [surprised, see, topic, debated, sub, recently...
dtype: object
```

In [49]:
```
#average reddit score, number of comments for delta and non-delta posts
print(with_delta["score"].mean())
print(with_delta["num_comments"].mean())
print(no_delta["score"].mean())
print(no_delta["num_comments"].mean())
```

```
223.42732558139534
179.53488372093022
80.2304347826087
94.15217391304348
```

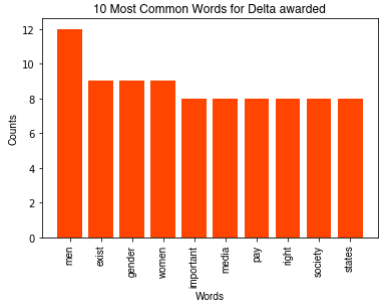
# Topic Analysis

In [6]:
```
#creating a vectorizer instance for topic analysis and the models
count_vectorizer = CountVectorizer(stop_words = sw)
```

## Dataset with Delta Awarded

Top ten words and topic analysis

In [7]:
```
    count_data1 = count_vectorizer.fit_transform(with_delta['title_withoutstop'])

    def plot_10_most_common_words1(count_data1, count_vectorizer):
        import matplotlib.pyplot as plt
        words = count_vectorizer.get_feature_names()
        total_counts = np.zeros(len(words))
        for t in count_data1:
            total_counts+=t.toarray()[0]

        count_dict = (zip(words, total_counts))
        count_dict = sorted(count_dict, key=lambda x:x[1], reverse=True)[0:10]
        words = [w[0] for w in count_dict]
        counts = [w[1] for w in count_dict]
        x_pos = np.arange(len(words))

        plt.bar(x_pos, counts, align = 'center', color = '#ff4500')
        plt.xticks(x_pos, words, rotation = 90, fontname = 'helvetica')
        plt.xlabel('Words', fontname = 'helvetica')
        plt.ylabel('Counts', fontname = 'helvetica')
        plt.title('10 Most Common Words for Delta awarded', fontname = 'helvetica')
        plt.show()

    plot_10_most_common_words1(count_data1, count_vectorizer)
```




In [8]:
```
def print_topics(model, count_vectorizer, n_top_words):
    words = count_vectorizer.get_feature_names()
    for topic_idx, topic in enumerate(model.components_):
        print("\nTopic #%d:" % topic_idx)
        print(" ".join([words[i]
                        for i in topic.argsort()[:-n_top_words - 1:-1]]))
```

In [9]:
```
number_topics = 3
number_words = 8
```

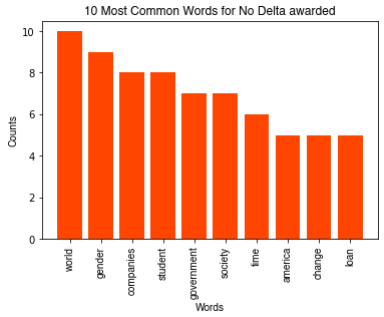
```
lda = LDA(n_components = number_topics, random_state = 42)
lda.fit(count_data1)

print("Topics found via LDA:")
print_topics(lda, count_vectorizer, number_words)
```

```
Topics found via LDA:

Topic #0:
pay exist women states companies right ukraine elon

Topic #1:
gender reason useless media time wing death trans

Topic #2:
men american important believe society sports champion body
```

## Dataset with No Delta Awarded

In [10]:

```
count_data2 = count_vectorizer.fit_transform(no_delta['title_withoutstop'])

def plot_10_most_common_words2(count_data2, count_vectorizer):
    import matplotlib.pyplot as plt
    words = count_vectorizer.get_feature_names()
    total_counts = np.zeros(len(words))
    for t in count_data2:
        total_counts+=t.toarray()[0]

    count_dict = (zip(words, total_counts))
    count_dict = sorted(count_dict, key = lambda x:x[1], reverse = True)[0:10]
    words = [w[0] for w in count_dict]
    counts = [w[1] for w in count_dict]
    x_pos = np.arange(len(words))

    plt.bar(x_pos, counts, align = 'center', color = '#ff4500')
    plt.xticks(x_pos, words, rotation = 90, fontname = 'helvetica')
    plt.xlabel('Words',fontname = 'helvetica')
    plt.ylabel('Counts',fontname = 'helvetica')
    plt.title('10 Most Common Words for No Delta awarded', fontname = 'helvetica')
    plt.show()

plot_10_most_common_words2(count_data2, count_vectorizer)
```




In [11]:

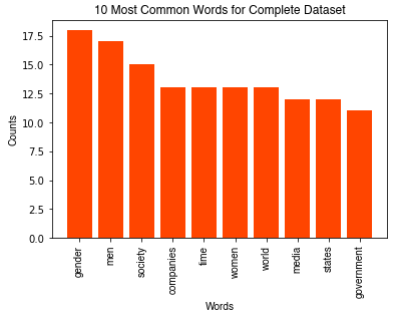
```
number_topic = 3
number_words = 8

lda = LDA(n_components = number_topics, random_state = 42)
lda.fit(count_data2)

print("Topics found via LDA:")
print_topics(lda, count_vectorizer, number_words)
```

```
Topics found via LDA:

Topic #0:
companies student government reason russia society loan human

Topic #1:
world state america sexual time student western wrong

Topic #2:
men gender death change women body ukraine society
```

## Complete dataset (with and without delta)

In [12]:

```
count_data3 = count_vectorizer.fit_transform(final['title_withoutstop'])

def plot_10_most_common_words3(count_data3, count_vectorizer):
    import matplotlib.pyplot as plt
    words = count_vectorizer.get_feature_names()
    total_counts = np.zeros(len(words))
    for t in count_data3:
        total_counts+=t.toarray()[0]

    count_dict = (zip(words, total_counts))
    count_dict = sorted(count_dict, key=lambda x:x[1], reverse=True)[0:10]
    words = [w[0] for w in count_dict]
    counts = [w[1] for w in count_dict]
    x_pos = np.arange(len(words))

    plt.bar(x_pos, counts, align = 'center', color = '#ff4500')
    plt.xticks(x_pos, words, rotation = 90, fontname = 'helvetica')
    plt.xlabel('Words', fontname = 'helvetica')
    plt.ylabel('Counts', fontname = 'helvetica')
    plt.title('10 Most Common Words for Complete Dataset', fontname = 'helvetica')
    plt.show()

plot_10_most_common_words3(count_data3, count_vectorizer)
```




In [13]:

```
number_topic = 3
number_words = 8

lda = LDA(n_components = number_topics, random_state = 42)
lda.fit(count_data3)

print("Topics found via LDA:")
print_topics(lda, count_vectorizer, number_words)
```

```
Topics found via LDA:

Topic #0:
time states companies american pay men exist women

Topic #1:
death media ukraine state wrong person world speech

Topic #2:
gender student important society trans body government believe
```

## Sentiment Analysis

In [14]:

```
#creating a new datasent for sentiment analysis that is cleaned using different stopword
senti = posts
senti = senti.drop(columns =["tokenized_title", "tokenized_body", "title_withoutstop", "body_withoutstop","id","url"], axis = 1)
senti['title_withoutstopsenti'] = senti['title_cleaned'].apply(lambda x: ' '.join([word for word in x.split() if word not in (sw_senti)]))
senti['body_withoutstopsenti'] = senti['body_cleaned'].apply(lambda x: ' '.join([word for word in x.split() if word not in (sw_senti)]))
senti['tokenized_title_senti'] = senti.apply(lambda column: nltk.word_tokenize(column['title_withoutstopsenti']), axis=1)
senti['tokenized_body_senti'] = senti.apply(lambda column: nltk.word_tokenize(column['body_withoutstopsenti']), axis=1)
senti.drop(senti[senti["flair"] == "Removed - Submission Rule E"].index, inplace = True)
senti.drop(senti[senti["flair"] == "Removed - Submission Rule A"].index, inplace = True)
senti.drop(senti[senti["flair"] == "Removed - Submission Rule B"].index, inplace = True)
senti.drop(senti[senti["flair"] == "META"].index, inplace = True)
mapping = {"Delta(s) from OP": 1, "None": 0,"Delta(s) from OP - Fresh Topic Friday": 1,"Fresh Topic Friday":0}
senti["flair"] = [mapping[item] for item in senti.flair]
senti.head()
senti
```

Out[14]:

| | title | body | flair | score | num_comments | title_cleaned | body_cleaned | title_withoutstopsenti | body_withoutstopsenti | tokenized_title_senti | tokenized_body_senti |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | CMV: Student Loans Should Not be Guaranteed by... | I was surprised to see that this topic has not... | 0 | 168 | 123 | cmv student loans should not be guaranteed by ... | i was surprised to see that this topic has not... | student loans guaranteed government | surprised see topic debated sub recently light... | [student, loans, guaranteed, government] | [surprised, see, topic, debated, sub, recently... |
| **1** | CMV: *IF* you believe isolation from lockdown ... | edit note: \nWhile I've already awarded delta... | 1 | 440 | 201 | cmv if you believe isolation from lockdown was... | edit note while ive already awarded deltas its... | isolation lockdown mentally unhealthy must log... | edit note ive already awarded deltas recommend... | [isolation, lockdown, mentally, unhealthy, mus... | [edit, note, ive, already, awarded, deltas, re... |
| **2** | CMV: US Colleges should not waste student's ti... | I went to a very competitive college in the US... | 1 | 1892 | 638 | cmv us colleges should not waste students time... | i went to a very competitive college in the us... | colleges waste students useless mandatory classes | went competitive college astounded number abso... | [colleges, waste, students, useless, mandatory... | [went, competitive, college, astounded, number... |
| **3** | CMV: There is Nothing Wrong with Abortion What... | We do not treat life as something that is sacr... | 0 | 15 | 85 | cmv there is nothing wrong with abortion whats... | we do not treat life as something that is sacr... | nothing wrong abortion whatsoever | treat something sacred countries certain crimi... | [nothing, wrong, abortion, whatsoever] | [treat, something, sacred, countries, certain,... |
| **4** | CMV: Western countries should invest in and su... | Admittedly, I do not know much about the compl... | 1 | 1270 | 140 | cmv western countries should invest in and sub... | admittedly i do not know much about the comple... | western countries invest subsidize semiconduct... | admittedly know complexities matters mind coul... | [western, countries, invest, subsidize, semico... | [admittedly, know, complexities, matters, mind... |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **576** | CMV: Financially, things would get better if m... | One big aspect a lot of people seem to skip wh... | 1 | 0 | 31 | cmv financially things would get better if mor... | one big aspect a lot of people seem to skip wh... | financially things get better saved money | big aspect lot seem skip discussing financial ... | [financially, things, get, better, saved, money] | [big, aspect, lot, seem, skip, discussing, fin... |
| **577** | CMV: The United States will, at some level, ou... | Let me put my bias and world view on the table... | 1 | 0 | 46 | cmv the united states will at some level outla... | let me put my bias and world view on the table... | united level outlaw abortion gay marriage inte... | let put bias world view table upfront im strai... | [united, level, outlaw, abortion, gay, marriag... | [let, put, bias, world, view, table, upfront, ... |
| **578** | CMV: It is only a matter of time until Russia ... | The view I wish to have changed is that Ukrain... | 1 | 1511 | 862 | cmv it is only a matter of time until russia w... | the view i wish to have changed is that ukrain... | matter russia wins ukraine | view wish changed ukraine cannot win war curre... | [matter, russia, wins, ukraine] | [view, wish, changed, ukraine, can, not, win, ... |
| **579** | CMV: Philosophy is the most important academic... | I think that philosophy is an incredibly impor... | 1 | 21 | 99 | cmv philosophy is the most important academic ... | i think that philosophy is an incredibly impor... | philosophy important academic subject mandator... | philosophy incredibly important subject core c... | [philosophy, important, academic, subject, man... | [philosophy, incredibly, important, subject, c... |
| **580** | CMV: The abuse of Reddit's revamped block feat... | For those who aren't aware, a few weeks ago Re... | 0 | 80 | 214 | cmv the abuse of reddits revamped block featur... | for those who arent aware a few weeks ago redd... | abuse reddits revamped block feature outweigh ... | arent aware weeks ago reddit revamped block fe... | [abuse, reddits, revamped, block, feature, out... | [arent, aware, weeks, ago, reddit, revamped, b... |

574 rows × 11 columns

In [15]:

```
sentiments = senti.drop(columns = ["tokenized_title_senti", "tokenized_body_senti"], axis = 1)

vader = SentimentIntensityAnalyzer()
scores = [vader.polarity_scores(body_withoutstopsenti) for body_withoutstopsenti in sentiments.body_withoutstopsenti]

# Convert the list of dicts into a DataFrame
body_feelings = pd.DataFrame(scores)

pd.set_option('display.max_rows', 100)

#Appending the dataframe with just the sentiments to the data.frame (final2).
sentiments = sentiments.join(body_feelings)

#Creating a column that displays a binary valence column
sentiments['valence']  = sentiments['compound'].apply(lambda c: 'pos' if c >=0 else 'neg')
```

```
sentiments = sentiments.dropna()
sentiments = sentiments.drop(columns = ['body_withoutstopsenti'])

sentiments.head()
```

Out[15]:

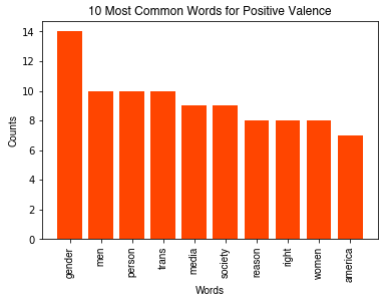
| | title | body | flair | score | num_comments | title_cleaned | body_cleaned | title_withoutstopsenti | neg | neu | pos | compound | valence |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | CMV: Student Loans Should Not be Guaranteed by... | I was surprised to see that this topic has not... | 0 | 168 | 123 | cmv student loans should not be guaranteed by ... | i was surprised to see that this topic has not... | student loans guaranteed government | 0.125 | 0.589 | 0.286 | 0.9684 | pos |
| 1 | CMV: *IF* you believe isolation from lockdown ... | edit note: \nWhile I've already awarded delta... | 1 | 440 | 201 | cmv if you believe isolation from lockdown was... | edit note while ive already awarded deltas its... | isolation lockdown mentally unhealthy must log... | 0.352 | 0.422 | 0.225 | -0.9902 | neg |
| 2 | CMV: US Colleges should not waste student's ti... | I went to a very competitive college in the US... | 1 | 1892 | 638 | cmv us colleges should not waste students time... | i went to a very competitive college in the us... | colleges waste students useless mandatory classes | 0.114 | 0.678 | 0.208 | 0.9934 | pos |
| 3 | CMV: There is Nothing Wrong with Abortion What... | We do not treat life as something that is sacr... | 0 | 15 | 85 | cmv there is nothing wrong with abortion whats... | we do not treat life as something that is sacr... | nothing wrong abortion whatsoever | 0.384 | 0.409 | 0.207 | -0.9834 | neg |
| 4 | CMV: Western countries should invest in and su... | Admittedly, I do not know much about the compl... | 1 | 1270 | 140 | cmv western countries should invest in and sub... | admittedly i do not know much about the comple... | western countries invest subsidize semiconduct... | 0.051 | 0.725 | 0.224 | 0.9844 | pos |

# Topic Analysis based on Valence

In [16]:
```
#splitting dataset based on valence
possentiment = sentiments[sentiments.valence == 'pos']
negsentiment = sentiments[sentiments.valence == 'neg']
```

## Positive Sentiment

In [17]:
```
count_data4 = count_vectorizer.fit_transform(possentiment['title_withoutstopsenti'])

def plot_10_most_common_words4(count_data4, count_vectorizer):
    import matplotlib.pyplot as plt
    words = count_vectorizer.get_feature_names()
    total_counts = np.zeros(len(words))
    for t in count_data4:
        total_counts+=t.toarray()[0]

    count_dict = (zip(words, total_counts))
    count_dict = sorted(count_dict, key=lambda x:x[1], reverse=True)[0:10]
    words = [w[0] for w in count_dict]
    counts = [w[1] for w in count_dict]
    x_pos = np.arange(len(words))

    plt.bar(x_pos, counts, align = 'center', color = '#ff4500')
    plt.xticks(x_pos, words, rotation = 90, fontname = 'helvetica')
    plt.xlabel('Words', fontname = 'helvetica')
    plt.ylabel('Counts', fontname = 'helvetica')
    plt.title('10 Most Common Words for Positive Valence', fontname = 'helvetica')
    plt.show()

plot_10_most_common_words4(count_data4, count_vectorizer)
```



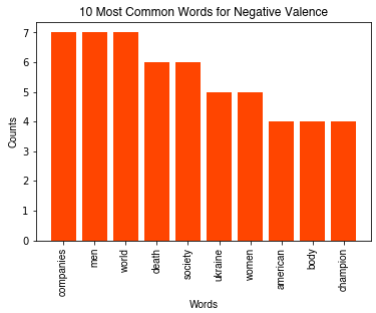

In [18]:
```
number_topics = 3
number_words = 8

lda = LDA(n_components = number_topics, random_state = 42)
lda.fit(count_data4)

print("Topics found via LDA:")
print_topics(lda, count_vectorizer, number_words)
```

```
Topics found via LDA:

Topic #0:
rights russia media gender care ukraine government student

Topic #1:
society person exist problems useless state work important

Topic #2:
trans reason argument elon gender valid america wrong
```

## Negative Sentiment

In [19]:
```
count_data5 = count_vectorizer.fit_transform(negsentiment['title_withoutstopsenti'])

def plot_10_most_common_words5(count_data5, count_vectorizer):
    import matplotlib.pyplot as plt
    words = count_vectorizer.get_feature_names()
    total_counts = np.zeros(len(words))
    for t in count_data5:
        total_counts+=t.toarray()[0]

    count_dict = (zip(words, total_counts))
    count_dict = sorted(count_dict, key=lambda x:x[1], reverse=True)[0:10]
    words = [w[0] for w in count_dict]
    counts = [w[1] for w in count_dict]
    x_pos = np.arange(len(words))

    plt.bar(x_pos, counts, align = 'center', color = '#ff4500')
    plt.xticks(x_pos, words, rotation = 90, fontname = 'helvetica')
    plt.xlabel('Words', fontname = 'helvetica')
    plt.ylabel('Counts', fontname = 'helvetica')
```

```
    plt.title('10 Most Common Words for Negative Valence', fontname = 'helvetica')
    plt.show()

plot_10_most_common_words5(count_data5, count_vectorizer)
```




In [20]:

```
number_topics = 3
number_words = 8

lda = LDA(n_components = number_topics, random_state = 42)
lda.fit(count_data5)

print("Topics found via LDA:")
print_topics(lda, count_vectorizer, number_words)
```

```
Topics found via LDA:

Topic #0:
men companies world anything school religion home change

Topic #1:
society american wrong sports women ukraine wild problem

Topic #2:
death champion twitter gender penalty worst date white
```

## Creating a Model Using Sentiment Scores as Features (Model 1)

In [21]:

```
sentiments.head()
```

Out[21]:

| | title | body | flair | score | num_comments | title_cleaned | body_cleaned | title_withoutstopsenti | neg | neu | pos | compound | valence |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | CMV: Student Loans Should Not be Guaranteed by... | I was surprised to see that this topic has not... | 0 | 168 | 123 | cmv student loans should not be guaranteed by ... | i was surprised to see that this topic has not... | student loans guaranteed government | 0.125 | 0.589 | 0.286 | 0.9684 | pos |
| 1 | CMV: *IF* you believe isolation from lockdown ... | edit note: \nWhile I've already awarded delta... | 1 | 440 | 201 | cmv if you believe isolation from lockdown was... | edit note while ive already awarded deltas its... | isolation lockdown mentally unhealthy must log... | 0.352 | 0.422 | 0.225 | -0.9902 | neg |
| 2 | CMV: US Colleges should not waste student's ti... | I went to a very competitive college in the US... | 1 | 1892 | 638 | cmv us colleges should not waste students time... | i went to a very competitive college in the us... | colleges waste students useless mandatory classes | 0.114 | 0.678 | 0.208 | 0.9934 | pos |
| 3 | CMV: There is Nothing Wrong with Abortion What... | We do not treat life as something that is sacr... | 0 | 15 | 85 | cmv there is nothing wrong with abortion whats... | we do not treat life as something that is sacr... | nothing wrong abortion whatsoever | 0.384 | 0.409 | 0.207 | -0.9834 | neg |
| 4 | CMV: Western countries should invest in and su... | Admittedly, I do not know much about the compl... | 1 | 1270 | 140 | cmv western countries should invest in and sub... | admittedly i do not know much about the comple... | western countries invest subsidize semiconduct... | 0.051 | 0.725 | 0.224 | 0.9844 | pos |

In [22]:

```
#Using score, num_comments, and compound as predictors of Delta

#dropping columns from sentiments to form dataset for Model 1
numericmodel = sentiments.drop(columns = ['title_withoutstopsenti',"title_cleaned","body_cleaned"])
numericmodel.head()
```

Out[22]:

| | title | body | flair | score | num_comments | neg | neu | pos | compound | valence |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | CMV: Student Loans Should Not be Guaranteed by... | I was surprised to see that this topic has not... | 0 | 168 | 123 | 0.125 | 0.589 | 0.286 | 0.9684 | pos |
| 1 | CMV: *IF* you believe isolation from lockdown ... | edit note: \nWhile I've already awarded delta... | 1 | 440 | 201 | 0.352 | 0.422 | 0.225 | -0.9902 | neg |
| 2 | CMV: US Colleges should not waste student's ti... | I went to a very competitive college in the US... | 1 | 1892 | 638 | 0.114 | 0.678 | 0.208 | 0.9934 | pos |
| 3 | CMV: There is Nothing Wrong with Abortion What... | We do not treat life as something that is sacr... | 0 | 15 | 85 | 0.384 | 0.409 | 0.207 | -0.9834 | neg |
| 4 | CMV: Western countries should invest in and su... | Admittedly, I do not know much about the compl... | 1 | 1270 | 140 | 0.051 | 0.725 | 0.224 | 0.9844 | pos |

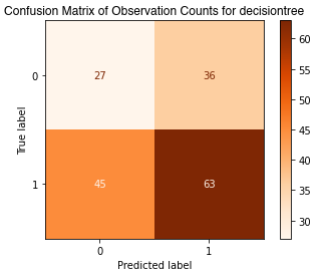
In [23]:

```
def evaluate_model(data, model, test_size = 0.3, normalize_cf = False, prediction_data = None):
    if model not in  ['knn','randomforest','boost','svm','decisiontree','bagging','logistic',
                      'passive_aggressive','radius_neighbors' ]:
        raise ValueError('Model {} not recognized. Model must be one of the following: {}'.format(model, ['knn','randomforest','boost','svm','decisiontree','bagging','logistic
                      'passive_aggressive','radius_neighbors' ]))

    scaler = MinMaxScaler()

    features = data.loc[:,'score':'compound']
    features = scaler.fit_transform(features)

    target = data['flair']

    x_train, x_test, y_train, y_test = train_test_split(features, target, test_size = test_size, random_state = 808)

    init_model = None

    if model == 'knn':
        init_model = knn(n_neighbors = 1)
    elif model == 'randomforest':
        init_model = RandomForestClassifier(random_state = 808)
    elif model == 'boost':
        init_model = GradientBoostingClassifier(random_state = 808)
    elif model == 'svm':
        init_model = SVC(random_state = 808)
    elif model == 'decisiontree':
        init_model = DecisionTreeClassifier(random_state = 808)
    elif model == 'bagging':
        init_model = BaggingClassifier(random_state = 808)
    elif model == 'logistic':
        init_model = LogisticRegression(random_state = 808)
    elif model == 'passive_aggressive':
```

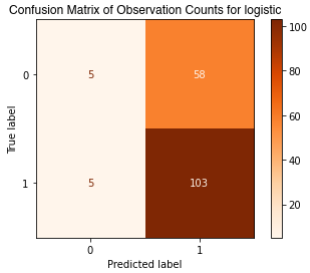
```
        init_model = PassiveAggressiveClassifier(random_state = 808)
    elif model == 'radius_neighbors':
        init_model = RadiusNeighborsClassifier(radius = 100, outlier_label = "most_frequent")

    fitted_model = init_model.fit(x_train ,y_train)
    test_predictions = fitted_model.predict(x_test)
    accuracy_score = fitted_model.score(x_test,y_test)

    cm = confusion_matrix(y_test, test_predictions, labels = fitted_model.classes_)
    disp = ConfusionMatrixDisplay(confusion_matrix = cm,display_labels = fitted_model.classes_)
    disp.plot(cmap = 'Oranges')
    plt.title('Confusion Matrix of Observation Counts for {}'.format(model), fontname = 'helvetica')
    plt.show()

    print("Accuracy Score for {}: {:.2%}".format(model, accuracy_score))
```

In [24]:

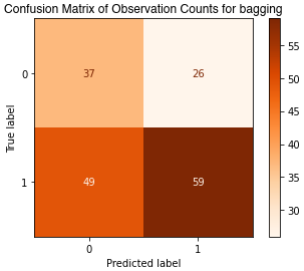
```
evaluate_model(numericmodel, model = 'decisiontree')
evaluate_model(numericmodel, model = 'logistic')
evaluate_model(numericmodel, model = 'bagging')
evaluate_model(numericmodel, model = 'boost')
evaluate_model(numericmodel, model = 'randomforest')
evaluate_model(numericmodel, model = 'svm')
evaluate_model(numericmodel, model = 'passive_aggressive')
evaluate_model(numericmodel, model = 'radius_neighbors')
```



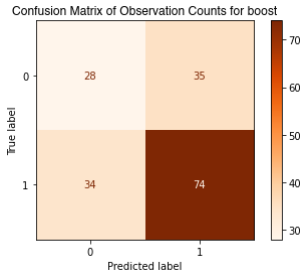

```
Accuracy Score for decisiontree: 52.63%
```



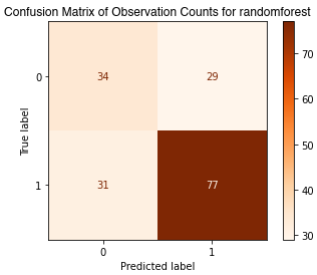

```
Accuracy Score for logistic: 63.16%
```



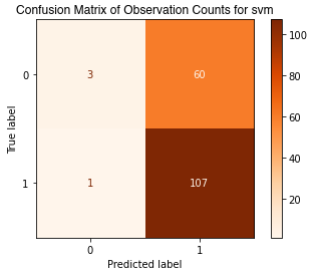

```
Accuracy Score for bagging: 56.14%
```



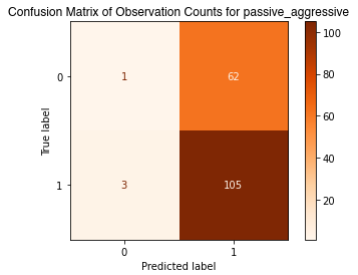

```
Accuracy Score for boost: 59.65%
```



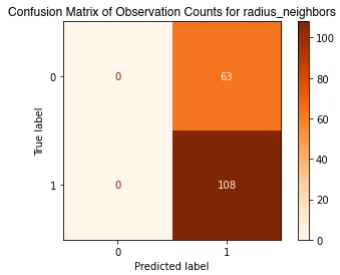

```
Accuracy Score for randomforest: 64.91%
```




```
Accuracy Score for svm: 64.33%
```




```
Accuracy Score for passive_aggressive: 61.99%
```




```
Accuracy Score for radius_neighbors: 63.16%
```

Obtaining Coefficients for Logistic Regression

In [25]:

```
scaler = MinMaxScaler()
features1 = numericmodel[['score', 'num_comments', 'compound']]
features1 = scaler.fit_transform(features1)
target = numericmodel['flair']
test_size = 0.3

x_train, x_test, y_train, y_test = train_test_split(features1, target, test_size = test_size, random_state = 808)

logreg = LogisticRegression().fit(x_train, y_train)
logreg

print("Training set score: {:.3f}".format(logreg.score(x_train,y_train)))
print("Test set score: {:.3f}".format(logreg.score(x_test,y_test)))

logit_model = sm.Logit(target, features1)
result = logit_model.fit()
print(result.summary())
```

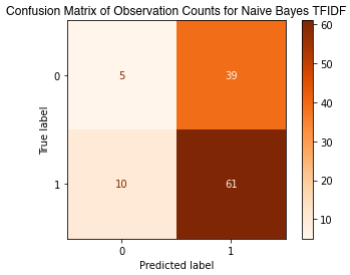
```
Training set score: 0.583
Test set score: 0.632
Optimization terminated successfully.
         Current function value: 0.652373
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:                  flair   No. Observations:                  567
Model:                          Logit   Df Residuals:                      564
Method:                           MLE   Df Model:                            2
Date:                Wed, 04 May 2022   Pseudo R-squ.:                0.03189
Time:                        11:25:06   Log-Likelihood:                -369.90
converged:                       True   LL-Null:                       -382.08
Covariance Type:            nonrobust   LLR p-value:                 5.115e-06
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
x1             0.1426      1.493      0.096      0.924      -2.784       3.069
x2             8.5578      2.661      3.216      0.001       3.341      13.774
x3             0.1381      0.145      0.951      0.342      -0.147       0.423
==============================================================================
```

## Creating a Model Using The Vectorized Text to Predict (Model 2)

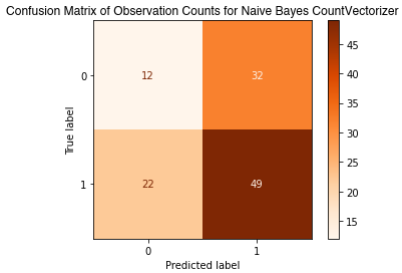
In [40]:

```
text_predictor = final

#Create a Target
flair = text_predictor.flair

X_train, X_test, y_train, y_test = train_test_split(text_predictor['title_withoutstop'], flair, test_size=0.2, random_state= 42)

#Intialize the Count Vectorizer
count_vectorizer = CountVectorizer(stop_words=sw)
count_train = count_vectorizer.fit_transform(X_train)
count_test = count_vectorizer.transform(X_test)

# Initialize tfidf vectorizer
tfidf_vectorizer = TfidfVectorizer(stop_words=sw)

# Create tfidf train and test variables
tfidf_train = tfidf_vectorizer.fit_transform(X_train)
tfidf_test = tfidf_vectorizer.transform(X_test)
```

```
#Creating Naive Bayes Model Using TFIDF
tfidf_nb = MultinomialNB()
tfidf_fitted_model = tfidf_nb.fit(tfidf_train, y_train)
tfidf_nb_pred = tfidf_nb.predict(tfidf_test)
tfidf_nb_score = metrics.accuracy_score(y_test, tfidf_nb_pred)

cm_tfidf = confusion_matrix(y_test, tfidf_nb_pred, labels = tfidf_fitted_model.classes_)
disp_tfidf = ConfusionMatrixDisplay(confusion_matrix = cm_tfidf, display_labels = tfidf_fitted_model.classes_)
disp_tfidf.plot(cmap = 'Oranges')
plt.title('Confusion Matrix of Observation Counts for Naive Bayes TFIDF', fontname = 'helvetica')
plt.show()

print('NaiveBayes Tfidf Score: ', tfidf_nb_score)

#Creating Naive Bayes Model Using Count Vectorizer
count_nb = MultinomialNB()
count_fitted_model = count_nb.fit(count_train, y_train)
count_nb_pred = count_nb.predict(count_test)
count_nb_score = metrics.accuracy_score(y_test, count_nb_pred)

cm_count = confusion_matrix(y_test, count_nb_pred, labels = count_fitted_model.classes_)
disp_count = ConfusionMatrixDisplay(confusion_matrix = cm_count, display_labels = count_fitted_model.classes_)
disp_count.plot(cmap = 'Oranges')
plt.title('Confusion Matrix of Observation Counts for Naive Bayes CountVectorizer', fontname = 'helvetica')
plt.show()

print('NaiveBayes Count Score: ', count_nb_score)
```




```
NaiveBayes Tfidf Score:  0.5739130434782609
```




```
NaiveBayes Count Score:  0.5304347826086957
```

In [41]:

```
tfidf_svc = LinearSVC()
tfidf_svc.fit(tfidf_train, y_train)
tfidf_svc_pred = tfidf_svc.predict(tfidf_test)
tfidf_svc_score = metrics.accuracy_score(y_test, tfidf_svc_pred)

print("LinearSVC Score:   %0.3f" % tfidf_svc_score)
```

```
LinearSVC Score:   0.539
```

In [42]:

```
# delta_post = "I do not feel very strongly about veganism, so please try to convince me"
# nondelta_post = "I will never change my view about veganism, but come at me"

# delta_post = "I do not feel very strongly about Alex winning Eurovision, so please try to convince me"
# nondelta_post = "I will never change my view about Alex winning Eurovision, but come at me"

delta_post = "I do not think Alex will win Eurovision, so convince me otherwise"
nondelta_post = "I will not change my view about Alex winning Eurovision, but come at me"

delta_post_vectorized = tfidf_vectorizer.transform([delta_post])
nondelta_post_vectorized = tfidf_vectorizer.transform([nondelta_post])

delta_post_pred = tfidf_svc.predict(delta_post_vectorized)
nondelta_post_pred = tfidf_svc.predict(nondelta_post_vectorized)

print('Predicted Post that Led to Changed View', delta_post_pred)
print('Predicted Post that Did Not Lead to View Change', nondelta_post_pred)
```

```
Predicted Post that Led to Changed View [1]
Predicted Post that Did Not Lead to View Change [0]
```

In [ ]: