# Importing the Data

In [2]:
```
import json
import pandas as pd

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('fivethirtyeight')
import nltk # importing Natural Landuage Processing Toolit

from nltk.corpus import stopwords #importing stop words to remove non-necissary
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import f1_score, confusion_matrix, get_scorer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import make_scorer, f1_score
from sklearn.tree import DecisionTreeClassifier
from imblearn.over_sampling import SMOTE
from imblearn.pipeline import make_pipeline


import re
import nltk
import string
import pickle


raw_data = pd.read_json('../../../data/data.json', lines=True)

raw_data.head(5)
```

Out[2]:

| | category | headline | authors | link | short_desc |
|---|---|---|---|---|---|
| **0** | CRIME | There Were 2 Mass Shootings In Texas Last Week... | Melissa Jeltsen | https://www.huffingtonpost.com/entry/texas-ama... | She husband. H their chi |
| **1** | ENTERTAINMENT | Will Smith Joins Diplo And Nicky Jam For The 2... | Andy McDonald | https://www.huffingtonpost.com/entry/will-smit... | Of course |
| **2** | ENTERTAINMENT | Hugh Grant Marries For The First Time At Age 57 | Ron Dicker | https://www.huffingtonpost.com/entry/hugh-gran... | The actor longtime gi Ann |

| | category | headline | authors | link | short_desc |
|---|---|---|---|---|---|
| **3** | ENTERTAINMENT | Jim Carrey Blasts 'Castrato' Adam Schiff And D... | Ron Dicker | https://www.huffingtonpost.com/entry/jim-carre... | The acto Dems kicking for |
| **4** | ENTERTAINMENT | Julianna Margulies Uses Donald Trump Poop Bags... | Ron Dicker | https://www.huffingtonpost.com/entry/julianna-... | The "Di actress sai the ba |

# Inspecting the data

In [3]:
```
#inspecting the data types - Noticed that they're all objects
raw_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200853 entries, 0 to 200852
Data columns (total 6 columns):
 #   Column             Non-Null Count   Dtype         
---  ------             --------------   -----         
 0   category           200853 non-null  object        
 1   headline           200853 non-null  object        
 2   authors            200853 non-null  object        
 3   link               200853 non-null  object        
 4   short_description  200853 non-null  object        
 5   date               200853 non-null  datetime64[ns]
dtypes: datetime64[ns](1), object(5)
memory usage: 9.2+ MB
```

In [4]:
```
#checking For Null Values
raw_data.isna().sum()
```

Out[4]:
```
category             0
headline             0
authors              0
link                 0
short_description    0
date                 0
dtype: int64
```

In [5]:
```
#reassigning dataframe to df to uphold the original structure of the data - used
df = raw_data.copy()

df.head()
```

Out[5]:

| | category | headline | authors | link | short_desc |
|---|---|---|---|---|---|

| | category | headline | authors | link | short_desc |
|---|---|---|---|---|---|
| **0** | CRIME | There Were 2 Mass Shootings In Texas Last Week... | Melissa Jeltsen | https://www.huffingtonpost.com/entry/texas-ama... | She husband. H their chi |
| **1** | ENTERTAINMENT | Will Smith Joins Diplo And Nicky Jam For The 2... | Andy McDonald | https://www.huffingtonpost.com/entry/will-smit... | Of course |
| **2** | ENTERTAINMENT | Hugh Grant Marries For The First Time At Age 57 | Ron Dicker | https://www.huffingtonpost.com/entry/hugh-gran... | The actor longtime gi Ann |
| **3** | ENTERTAINMENT | Jim Carrey Blasts 'Castrato' Adam Schiff And D... | Ron Dicker | https://www.huffingtonpost.com/entry/jim-carre... | The acto Dems kicking for |
| **4** | ENTERTAINMENT | Julianna Margulies Uses Donald Trump Poop Bags... | Ron Dicker | https://www.huffingtonpost.com/entry/julianna-... | The "Di actress sai the ba |

---

# Preprocessing the Data for Feature Engineering

## Removing Punctuation and cleaning rows

### Short Description Feature

In [6]:
```
df.short_description
```

Out[6]:
```
0         She left her husband. He killed their children...
1                               Of course it has a song.
2         The actor and his longtime girlfriend Anna Ebe...
3         The actor gives Dems an ass-kicking for not fi...
4         The "Dietland" actress said using the bags is ...
                              ...
200848    Verizon Wireless and AT&T are already promotin...
200849    Afterward, Azarenka, more effusive with the pr...
200850    Leading up to Super Bowl XLVI, the most talked...
200851    CORRECTION: An earlier version of this story i...
200852    The five-time all-star center tore into his te...
Name: short_description, Length: 200853, dtype: object
```

In [7]:
```
df.headline
```

Out[7]:
```
0         There Were 2 Mass Shootings In Texas Last Week...
1         Will Smith Joins Diplo And Nicky Jam For The 2...
2           Hugh Grant Marries For The First Time At Age 57
3         Jim Carrey Blasts 'Castrato' Adam Schiff And D...
4         Julianna Margulies Uses Donald Trump Poop Bags...
                              ...
200848    RIM CEO Thorsten Heins' 'Significant' Plans Fo...
200849    Maria Sharapova Stunned By Victoria Azarenka I...
200850    Giants Over Patriots, Jets Over Colts Among  M...
200851    Aldon Smith Arrested: 49ers Linebacker Busted ...
200852    Dwight Howard Rips Teammates After Magic Loss ...
Name: headline, Length: 200853, dtype: object
```

### Removing Casing and punctuaton

In [8]:
```
#Removing punctuation. It helps us reduce the size of the data
df['short_description'] = df['short_description'].str.lower().str.replace('[^\w\
print(df['short_description'].head(5))

print('\n============================================================')

df['headline'] = df['headline'].str.lower().str.replace('[^\w\s]','')
print('\n', df['headline'].head(5))
```

```
0    she left her husband he killed their children ...
1                            of course it has a song
2    the actor and his longtime girlfriend anna ebe...
3    the actor gives dems an asskicking for not fig...
4    the dietland actress said using the bags is a ...
Name: short_description, dtype: object

============================================================

 0    there were 2 mass shootings in texas last week...
1    will smith joins diplo and nicky jam for the 2...
2      hugh grant marries for the first time at age 57
3    jim carrey blasts castrato adam schiff and dem...
4    julianna margulies uses donald trump poop bags...
Name: headline, dtype: object
```

### Merging Description and Headline together

- checking lengths

In [9]:
```
len(df.short_description[1])
```

Out[9]:
```
23
```

In [10]:
```
len(df.headline[1])
```

Out[10]:
```
74
```

In [11]:
```
#merging
df['head_description'] = df.headline + ' ' + df.short_description
len(df.head_description[1])
```

Out[11]:
```
98
```

## Removing Non Contributional Words from the Dataset

In [12]:
```
#downloading stopwords to use
nltk.download('stopwords')

stop = stopwords.words('english')

#inspecting stop words
print(stop)
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're",
"you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'h
im', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's",
'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'whic
h', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'ar
e', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do',
'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because',
'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'be
tween', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to',
'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'furth
er', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'an
y', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor',
'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'wil
l', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o',
're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "did
n't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "have
n't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn',
"needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren',
"weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/chandler_oneal/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [13]:
```
#how many stop words do we have?
df['stopwords'] = df['head_description'].apply(lambda x: len([x for x in x.split
df[['head_description','stopwords']].head()
```

Out[13]:

| | head_description | stopwords |
|---|---|---|
| **0** | there were 2 mass shootings in texas last week... | 12 |
| **1** | will smith joins diplo and nicky jam for the 2... | 8 |
| **2** | hugh grant marries for the first time at age 5... | 9 |
| **3** | jim carrey blasts castrato adam schiff and dem... | 7 |
| **4** | julianna margulies uses donald trump poop bags... | 8 |

In [14]:
```
# inspecting stop words column to identify the number of unnecissary words withi
df.head(2)
```

Out[14]:

| | category | headline | authors | link | short_descripti |
|---|---|---|---|---|---|
| **0** | CRIME | there were 2 mass shootings in texas last week... | Melissa Jeltsen | https://www.huffingtonpost.com/entry/texas-ama... | she left h husband he kil their childrer |

| | category | headline | authors | link | short_descripti |
|---|---|---|---|---|---|
| 1 | ENTERTAINMENT | will smith joins diplo and nicky jam for the 2... | Andy McDonald | https://www.huffingtonpost.com/entry/will-smit... | of course it ha sc |

### Removing Stop Words

In [15]:
```
#removing stopwords
df['clean_head_description'] = df['head_description'].apply(lambda x: " ".join(x
```

In [16]:
```
df['clean_head_description'].head()
```

Out[16]:
```
0    2 mass shootings texas last week 1 tv left hus...
1    smith joins diplo nicky jam 2018 world cups of...
2    hugh grant marries first time age 57 actor lon...
3    jim carrey blasts castrato adam schiff democra...
4    julianna margulies uses donald trump poop bags...
Name: clean_head_description, dtype: object
```

## Lemmitizing the data to reduce repitition by removing the affixes

In [17]:
```
#instantiating a lemmatizer
lemma = WordNetLemmatizer() #instantiate
```

## Tokenizing Data to Break Each Sentence Into Individual Strings

- **splitting a phrase, sentence, paragraph, or an entire text document into smaller units, such as individual words or terms. Each of these smaller units are called tokens.**

In [18]:
```
#removing affixes from all words
def lemm_words(data):
    tokens = word_tokenize(data)   #Tokenizing data
    lemmed_tokens = [lemma.lemmatize(word) for word in tokens]   #lemmatizing da
    return ' '.join(lemmed_tokens)
```

In [19]:
```
lemmed = df.head_description.apply(lemm_words)
```

In [20]:
```
#resetting lemmed to l to avoid overwriting issues - Do not want to re-lemmatize
l = lemmed                                  #there were 2 mass shooting in texas las
```

In [21]:
```
l
```

Out[21]:
```
0         there were 2 mass shooting in texas last week ...
1         will smith join diplo and nicky jam for the 20...
2         hugh grant marries for the first time at age 5...
3         jim carrey blast castrato adam schiff and demo...
4         julianna margulies us donald trump poop bag to...
                              ...
200848    rim ceo thorsten heins significant plan for bl...
200849    maria sharapova stunned by victoria azarenka i...
200850    giant over patriot jet over colt among most im...
200851    aldon smith arrested 49ers linebacker busted f...
```

```
200852    dwight howard rip teammate after magic loss to...
Name: head_description, Length: 200853, dtype: object
```

In [22]:
```
#changing series into a list of lists
df['unique_word'] = l.str.split(' ')   # [there, were, 2, mass, shooting, in, te
```

In [23]:
```
df[['unique_word']]
```

Out[23]:

| | unique_word |
|---|---|
| **0** | [there, were, 2, mass, shooting, in, texas, la... |
| **1** | [will, smith, join, diplo, and, nicky, jam, fo... |
| **2** | [hugh, grant, marries, for, the, first, time, ... |
| **3** | [jim, carrey, blast, castrato, adam, schiff, a... |
| **4** | [julianna, margulies, us, donald, trump, poop,... |
| **...** | ... |
| **200848** | [rim, ceo, thorsten, heins, significant, plan,... |
| **200849** | [maria, sharapova, stunned, by, victoria, azar... |
| **200850** | [giant, over, patriot, jet, over, colt, among,... |
| **200851** | [aldon, smith, arrested, 49ers, linebacker, bu... |
| **200852** | [dwight, howard, rip, teammate, after, magic, ... |

200853 rows × 1 columns

## Creating a Bag of Words for Vectorization

In [24]:
```
# list containing all words

unique_lemmed_list = []                 #the  (appears ->)  261830
for x in df['unique_word']:
    unique_lemmed_list += x  # append elements of lists to full list
```

- **Removing Single Characters**

In [25]:
```
#inspecting the original number of words
len(unique_lemmed_list)
```

Out[25]:
```
5855301
```

In [26]:
```
#looping through and adding words > len(1) to a new list
full_words = [] # creating quotes
index = 0
for words in unique_lemmed_list:
    if len(words) > 1:
        full_words.append(words)
```

In [27]:
```
#checking the NEW number of words after removal
len(full_words)
```

Out[27]:
```
5621546
```

- **Changing New Bag of Words into a Series for TF-IDF**

In [28]:
```
word_value_counts = pd.Series(full_words).value_counts()
word_value_counts
```

Out[28]:
```
the                261830
to                 163957
of                 128119
and                119623
in                  95541
                     ...
oseary                  1
bermudez                1
hotelrestaurant         1
waster                  1
asmr                    1
Length: 102078, dtype: int64
```

In [29]:
```
plot = pd.DataFrame(data = word_value_counts).reset_index()
plot.columns = ['word','count']
plot.head()
```

Out[29]:

| | word | count |
|---|---|---|
| **0** | the | 261830 |
| **1** | to | 163957 |
| **2** | of | 128119 |
| **3** | and | 119623 |
| **4** | in | 95541 |

# Feature Engineering

In [30]:
```
len(df.short_description[0])
```

Out[30]:
```
73
```

In [31]:
```
df['len_description_head'] = [len(df.short_description[x]) for x in range(len(df
```

## Merging Similar Columns:

In [32]:
```
df.head(1)
```

Out[32]:

| | category | headline | authors | link | short_description | dat |
|---|---|---|---|---|---|---|
| **0** | CRIME | there were 2 mass shootings in texas last week... | Melissa Jeltsen | https://www.huffingtonpost.com/entry/texas-ama... | she left her husband he killed their children ... | 2018 05 2 |

- **Visualizing current columns**

In [33]:
```
#total number of categories
print('# of categories:', df.category.value_counts().nunique())

print('\n\nListing Categories:', df.category.value_counts())
```

```
# of categories: 41


Listing Categories: POLITICS          32739
WELLNESS          17827
ENTERTAINMENT     16058
TRAVEL             9887
STYLE & BEAUTY     9649
PARENTING          8677
HEALTHY LIVING     6694
QUEER VOICES       6314
FOOD & DRINK       6226
BUSINESS           5937
COMEDY             5175
SPORTS             4884
BLACK VOICES       4528
HOME & LIVING      4195
PARENTS            3955
THE WORLDPOST      3664
WEDDINGS           3651
WOMEN              3490
IMPACT             3459
DIVORCE            3426
CRIME              3405
MEDIA              2815
WEIRD NEWS         2670
GREEN              2622
WORLDPOST          2579
RELIGION           2556
STYLE              2254
SCIENCE            2178
WORLD NEWS         2177
TASTE              2096
TECH               2082
MONEY              1707
ARTS               1509
FIFTY              1401
GOOD NEWS          1398
ARTS & CULTURE     1339
ENVIRONMENT        1323
COLLEGE            1144
LATINO VOICES      1129
CULTURE & ARTS     1030
EDUCATION          1004
Name: category, dtype: int64
```

- **Combining columns**

In [34]:
```
categories = df['category'].value_counts().index
categories
```

Out[34]:
```
Index(['POLITICS', 'WELLNESS', 'ENTERTAINMENT', 'TRAVEL', 'STYLE & BEAUTY',
       'PARENTING', 'HEALTHY LIVING', 'QUEER VOICES', 'FOOD & DRINK',
       'BUSINESS', 'COMEDY', 'SPORTS', 'BLACK VOICES', 'HOME & LIVING',
       'PARENTS', 'THE WORLDPOST', 'WEDDINGS', 'WOMEN', 'IMPACT', 'DIVORCE',
       'CRIME', 'MEDIA', 'WEIRD NEWS', 'GREEN', 'WORLDPOST', 'RELIGION',
```

```
       'STYLE', 'SCIENCE', 'WORLD NEWS', 'TASTE', 'TECH', 'MONEY', 'ARTS',
       'FIFTY', 'GOOD NEWS', 'ARTS & CULTURE', 'ENVIRONMENT', 'COLLEGE',
       'LATINO VOICES', 'CULTURE & ARTS', 'EDUCATION'],
      dtype='object')
```

In [35]:

```
def groupper(grouplist,name):
    for idx in categories: #for each category name
        if idx in grouplist: #if the category name appears in the grouplist belo
            df.loc[df['category'] == idx, 'category'] = name  #assign it to name



groupper( grouplist= ['WELLNESS', 'HEALTHY LIVING','HOME & LIVING','STYLE & BEAU
groupper( grouplist= [ 'PARENTING', 'PARENTS' ,'EDUCATION' ,'COLLEGE'] , name =
groupper( grouplist= ['SPORTS','ENTERTAINMENT' , 'COMEDY','WEIRD NEWS','ARTS'] ,
groupper( grouplist= ['TRAVEL', 'ARTS & CULTURE','CULTURE & ARTS','FOOD & DRINK'
groupper( grouplist= ['WOMEN','QUEER VOICES', 'LATINO VOICES', 'BLACK VOICES'] ,
groupper( grouplist= ['BUSINESS' ,  'MONEY'] , name =  'BUSINESS-MONEY')
groupper( grouplist= ['THE WORLDPOST' , 'WORLDPOST' , 'WORLD NEWS'] , name =  'W
groupper( grouplist= ['ENVIRONMENT' ,'GREEN'] , name =  'ENVIRONMENT')
groupper( grouplist= ['TECH', 'SCIENCE'] , name =  'SCIENCE AND TECH')
groupper( grouplist= ['FIFTY' , 'IMPACT' ,'GOOD NEWS','CRIME'] , name =  'GENERA
groupper( grouplist= ['WEDDINGS', 'DIVORCE',  'RELIGION','MEDIA'] , name =  'MIS
```

- **Inspecting new columns**

In [36]:

```
#inspecting newly combined categories
print('# of categories:', df.category.value_counts().nunique())
print('\n\nListing Categories:', df.category.value_counts())
```

```
# of categories: 12


Listing Categories: LIFESTYLE AND WELLNESS          40619
POLITICS                          32739
SPORTS AND ENTERTAINMENT          30296
TRAVEL-TOURISM & ART-CULTURE      20578
EMPOWERED VOICES                  15461
PARENTING AND EDUCATION           14780
MISC                              12448
GENERAL                            9663
WORLDNEWS                          8420
BUSINESS-MONEY                     7644
SCIENCE AND TECH                   4260
ENVIRONMENT                        3945
Name: category, dtype: int64
```

**Sorting Words Using Parts of Speech Sepparator Function**

## TF IDF

**Term Frequency — Inverse Document Frequency** - Derermining word importance to documents

- **Vectorizing the (title description + full description) column (converting it into numeric data).**

Final Model TF-IDF

In [55]:
```
tfidf_vectorizer= TfidfVectorizer(max_features=3000)
X_tfidf = tfidf_vectorizer.fit_transform(df['head_description'])
features = (tfidf_vectorizer.get_feature_names())
```

In [57]:
```
features[:10]
```

Out[57]:
```
['10', '100', '11', '12', '13', '14', '15', '16', '17', '18']
```

In [58]:
```
print("\n\nShape of tfidf : \n", X_tfidf.shape)


#printing out the features
print("\n\nFeatures : \n", features[:200])
print("\n\nX1 : \n", X_tfidf.toarray())
```

```
Shape of tfidf :
 (200853, 3000)


Features :
 ['10', '100', '11', '12', '13', '14', '15', '16', '17', '18', '19', '20', '20
0', '2008', '2010', '2011', '2012', '2013', '2014', '2015', '2016', '2017', '201
8', '21', '22', '23', '24', '247', '25', '26', '27', '30', '31', '35', '40', '5
0', '60', '90s', '911', 'ability', 'able', 'abortion', 'about', 'above', 'abroa
d', 'absolutely', 'abuse', 'academy', 'accept', 'access', 'according', 'accoun
t', 'accounts', 'accused', 'achieve', 'across', 'act', 'acting', 'action', 'acti
ons', 'active', 'activist', 'activists', 'activities', 'activity', 'actor', 'act
ors', 'actress', 'acts', 'actual', 'actually', 'ad', 'adam', 'add', 'added', 'ad
diction', 'address', 'administration', 'admit', 'admits', 'adorable', 'ads', 'ad
ult', 'adults', 'advantage', 'adventure', 'advice', 'advocates', 'affair', 'affe
ct', 'affected', 'affects', 'afford', 'affordable', 'afghanistan', 'afraid', 'af
rica', 'african', 'after', 'afternoon', 'again', 'against', 'age', 'agency', 'ag
enda', 'aging', 'ago', 'agree', 'agreement', 'ahead', 'aid', 'aids', 'aims', 'ai
r', 'airline', 'airlines', 'airport', 'al', 'alabama', 'album', 'alcohol', 'aliv
e', 'all', 'allegations', 'alleged', 'allegedly', 'allies', 'allow', 'allowed',
'allows', 'almost', 'alone', 'along', 'already', 'also', 'alternative', 'althoug
h', 'always', 'alzheimers', 'am', 'amazing', 'amazon', 'amendment', 'america',
'american', 'americans', 'americas', 'amid', 'among', 'amount', 'amy', 'an', 'an
cient', 'and', 'andrew', 'angeles', 'anger', 'angry', 'animal', 'animals', 'ann
a', 'anniversary', 'announced', 'announcement', 'announces', 'annual', 'anothe
r', 'answer', 'answers', 'anthony', 'anxiety', 'any', 'anymore', 'anyone', 'anyt
hing', 'anyway', 'anywhere', 'ap', 'apart', 'apartment', 'app', 'apparently', 'a
ppeal', 'appear', 'appearance', 'appeared', 'appears', 'apple', 'appreciate', 'a
pproach', 'apps', 'april', 'are', 'area', 'areas', 'arent', 'arizona', 'armed',
'arms', 'army']


X1 :
 [[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
```

In [59]:
```
X_tfidf_df = pd.DataFrame(X_tfidf.toarray(), columns=features)
```

In [60]:
```
#inspecting values for imbalances to determine whether or not to SMOTE the data
X_tfidf_df
```

Out[60]:

| | 10 | 100 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | ... | youll | young | younger | your | youre |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **1** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **2** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **3** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **4** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **200848** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **200849** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **200850** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **200851** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **200852** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

200853 rows × 3000 columns

# Modeling

## Model Preparatition:

**Creating a function to shorten the modeling process**

In [63]:
```
#creating a function to pass in our model that will then return an F-1 Score

def make_train_scorer(model, X_train, y_train, X_val, y_val):
    model.fit(X_train, y_train)
    scorer = make_scorer(f1_score, average='weighted')

    train_f1 = scorer(model, X_train, y_train)
    val_f1 = scorer(model, X_val, y_val)

    print('Train Score:', train_f1)
    print('Validation Score:', val_f1)
```

**Splitting the vecorized data into the variable X and the Categories into the variable y**

In [65]:
```
X = X_tfidf_df
y = df.category
```

In [66]:
```
X_tr, X_test, y_tr, y_test = train_test_split(X, y, random_state=3, test_size=.2
```

In [67]:
```
X_train, X_val, y_train, y_val = train_test_split(X_tr, y_tr, random_state=3, te
```

**Rebalancing Data**

In [68]:
```
sm = SMOTE(random_state=3)

X_train_res, y_train_res = sm.fit_resample(X_train, y_train)
```

```
X_val_res, y_val_res = sm.fit_resample(X_val, y_val)
```

In [69]:
```
print('X_train resampled:', y_train_res)

print('=========================')
```

```
X_train resampled: 0                 BUSINESS-MONEY
1          LIFESTYLE AND WELLNESS
2                        POLITICS
3                            MISC
4                     ENVIRONMENT
                     ...
311575                   WORLDNEWS
311576                   WORLDNEWS
311577                   WORLDNEWS
311578                   WORLDNEWS
311579                   WORLDNEWS
Name: category, Length: 311580, dtype: object
=========================
```

## BASIC MODELS: Fitting and Returning Scores for Models

- **Random Forest - Basic**

In [ ]:
```
#instantiating a random forest model


#NEXT -> increase features and run through this model

# max_features='auto'
# randf1_model1 = RandomForestClassifier(random_state=3, max_features='auto')
```

In [ ]:
```
# make_train_scorer(randf1_model1, X_train_res, y_train_res, X_val_res, y_val_re
```

### Final Model Logistic Regression - Basic (PICKELED)

In [ ]:
```
# instantiating a logistic regression model

# logreg1_model2 = LogisticRegression(random_state=3, max_iter=1000)
```

In [ ]:
```
# make_train_scorer(logreg1_model2, X_train_res, y_train_res, X_val_res, y_val_r
```

- **Pickeling Model**

In [ ]:
```
# filename = 'finalized_model.sav'
# pickle.dump(logreg1_model2, open(filename, 'wb'))
```

- **Testing Pickeled Model**

In [82]:
```
loaded_model = pickle.load(open('finalized_model.sav', 'rb'))
```

In [87]:
```
scorer = make_scorer(f1_score, average='weighted')
train_f1 = scorer(loaded_model, X_train, y_train)
val_f1 = scorer(loaded_model, X_val, y_val)
```

```
print('Train Score:', train_f1)
print('Validation Score:', val_f1)
```

```
Train Score: 0.6626628001821815
Validation Score: 0.6128748497328451
```

- **Desision Search Tree - Basic**

In [ ]:
```
# dectree1_model3 = DecisionTreeClassifier(random_state=3)
```

In [ ]:
```
# make_train_scorer(dectree1_model3, X_train, y_train, X_val, y_val)
```

- **Naive Bayes - Basic**

In [ ]:
```
# from sklearn.naive_bayes import MultinomialNB
# clf1_model4 = MultinomialNB()
```

In [ ]:
```
# make_train_scorer(clf1_model4, X_train, y_train, X_val, y_val)
```

## NON-BASIC MODEL: Fitting and Returning Scores for Models With Gridsearch CV and Pipeline

- **Logistic Regression - Pipeline & Gridsearch CV**

In [ ]:
```
# pipe = make_pipeline(LogisticRegression())
```

In [ ]:
```
# list(pipe.get_params().keys())
```

In [ ]:
```
# param_grid = {
#                 'logisticregression__C':[.5,.7,.8,.9,1.0],
#                 'logisticregression__class_weight':[None, "balanced" ],
#                 'logisticregression__penalty': ['l1', 'l2'],
#                 'logisticregression__solver': [ 'lbfgs', 'liblinear', 'sag'],
#                 'logisticregression__max_iter':[100, 1000, 2000]
#             }


# gs_logreg2_model5 = GridSearchCV(estimator=pipe, param_grid=param_grid, scorin
```

fitting to GS 1

In [ ]:
```
# make_train_scorer(gs_logreg2_model5, X_train_res, y_train_res, X_val_res, y_va
```

# Model Predictions

---

**STEPS: Preprocessing Testing Data:**

- `Returned Pickeled Model`
- `Save a Random Test Set as a CSV File`

- `Pickel the Transformed Data`

**Randomly selecting 5 descriptions from the cleaned data frame**

In [70]:
```
#pass into model
test_head_descr = l.sample(n=5, random_state=4)
```

In [71]:
```
# returning a greater portion from each head / description
pd.options.display.max_colwidth = 300
```

In [72]:
```
type(test_head_descr)
```

Out[72]:
```
pandas.core.series.Series
```

In [73]:
```
# dataframe to save to csv
test_head_descr_df = pd.DataFrame(test_head_descr)
test_head_descr_df
```

Out[73]:

| | head_description |
|---|---|
| **194912** | facebook tied to feeling fat eating disorder by leslie meredith published 03302012 0516 pm edt on technewsdaily do i look fat the answer is a resounding yes |
| **162594** | celebrating all that ireland ha to offer photo a jetblue flight attendant share her recent itinerary |
| **113442** | expert tip on getting girl into stem s no secret that girl lag behind boy when it come to an interest in science technology engineering and mathematics stem or that job relating to stem discipline are the fastestgrowing segment of the u economy |
| **46916** | hillary clinton and donald trump already rapped their debate who say there no rhyme or reason to the campaign |
| **51399** | the internet is having a field day comparing justin bieber and orlando bloom nude pic is it too late for twitter to say sorry |

In [74]:
```
#inspecting full rows of original dataframe to get unfiltered descriptions
```

In [75]:
```
raw_data.headline.iloc[51399]
```

Out[75]:
```
"The Internet Is Having A Field Day Comparing Justin Bieber And Orlando Bloom's Nude Pics"
```

In [76]:
```
raw_data.short_description.iloc[51399]
```

Out[76]:
```
'Is it too late for Twitter to say sorry?'
```

- **Saving Descriptions to CSV**

In [77]:
```
#saving Dataframe to a CSV file
# test_head_descr_df.to_csv('../../../data/test_df.csv',index=False)
```

# Predictions & Scoring

## Returning the Pickeled Model and Running Predictions

In [78]:
```
user_input = test_head_descr_df['head_description'].iloc[2]


vectorized = tfidf_vectorizer.transform([user_input])
loaded_model = pickle.load(open('finalized_model.sav', 'rb'))

#predicting on pickeled model
loaded_model.predict(vectorized)
```

Out[78]:
```
array(['PARENTING AND EDUCATION'], dtype=object)
```

- **Pickeling the Vectorizer for the New Test Data**

In [80]:
```
# filename = 'tfidf_vectorizer.sav'
# file = open(filename, 'wb')
# pickle.dump(tfidf_vectorizer, file)
# file.close()
```

---

## Scoring Test Data

- **Rebalancing the Testing Data**

In [86]:
```
X_test_res, y_test_res = sm.fit_resample(X_test, y_test)
```

- **Scoring the Test Data**

In [89]:
```
scorer = make_scorer(f1_score, average='weighted')
train_f1 = scorer(loaded_model, X_train, y_train)
test_f1 = scorer(loaded_model, X_test_res, y_test_res)



print('Train Score:', train_f1)
print('Test Score:', test_f1)
```

```
Train Score: 0.6626628001821815
Test Score: 0.622566468315379
```