import html2text

import re

import pandas as pd

import spacy as sp

from datetime import datetime

from sklearn.feature\_selection import SelectKBest, f\_classif

df = pd.read\_csv('2018.csv') # skiprows=(1, 20000),

df.shape

# Checking and removing null values

df[df['headline'].isnull()] # No null values

df[df['pageviews'].isnull()]

df.dropna(subset=['pageviews'], inplace=True)

df.dropna(subset=['headline'], inplace=True)

print(df.shape)

# Discarding rows with less than 100 words in df['body']

df['headline'] = df['headline'].astype('str')

#df = df[df['body'].str.len() > 100]

#df['pageviews'] = df['pageviews'].astype('float')

#df[df['pageviews'].apply(lambda x: str(x).isdigit())]

import numpy as np

# df = df[df['editors\_pick'] == True] # df will have only rows with True in c3

print(df.shape)

body\_raw = df['headline']

#body\_prepro= df['headline']

body\_prepro = []

pv = []

# for pages in df['pageviews']: #written\_by\_forbes\_staff

# if pages < 2000:

# pv.append('low')

# elif pages > 50000:

# pv.append('high')

# else:

# pv.append('med')

for val in df['editors\_pick']:

if val == True:

pv.append(1)

else:

pv.append(0)

for line in body\_raw:

line = re.sub(r'[?]', " h\_question ", line)

line = re.sub(r'[?]', " h\_question ", line)

line = re.sub(r'[!]', " h\_exclamation ", line)

line = re.sub(r'[:]', " h\_colon ", line)

line = re.sub(r'[%]', " h\_percent ", line)

line = re.sub(r'[0-9]+', " h\_integer ", line)

line = re.sub(r'[Bb]attle [Rr]oyal', " BattleRoyale ", line)

line = re.sub(r'[Aa]rtificial [Ii]ntelligence', " artificial\_intelligence ", line)

line = re.sub(r'[Nn][Ee][Ww] [Yy][Oo][Rr][Kk]', "NewYork", line)

line = re.sub(r'[Aa][Ii]', " artificial\_intelligence ", line)

body\_prepro.append(line)

from nltk import word\_tokenize

from nltk.stem import WordNetLemmatizer

class LemmaTokenizer(object):

def \_\_init\_\_(self):

self.wnl = WordNetLemmatizer()

def \_\_call\_\_(self, body\_prepro):

return [self.wnl.lemmatize(t) for t in word\_tokenize(body\_prepro)]

from sklearn.feature\_extraction.text import TfidfVectorizer, CountVectorizer

max\_features = 500

vectorizer = TfidfVectorizer(max\_features=max\_features)

y = vectorizer.fit\_transform(body\_prepro).toarray()

tfidf = vectorizer.get\_feature\_names()

indices = np.argsort(vectorizer.idf\_)[::-1]

#y = SelectKBest(k=10000).fit\_transform(df\_t,binnumber)

# top\_n = 20

# dividing data into train and test set

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(y, pv, test\_size=0.20, random\_state=0)

from sklearn.linear\_model import LogisticRegression, LinearRegression

from sklearn import metrics

clf = LogisticRegression(multi\_class='auto').fit(X\_train, y\_train)

log\_pred = clf.predict(X\_test)

accu\_log = metrics.accuracy\_score(y\_test, log\_pred)

print(' Logistic Regression Accuracy :', accu\_log \* 100)

coefs = clf.coef\_[0]

top\_20 = np.argpartition(coefs, -40)[-40:] # For positive features

#top\_20 = np.argpartition(coefs,30)[:30] #For negative features

top\_20\_sorted = top\_20[np.argsort(coefs[top\_20])]

top\_20\_desc = top\_20\_sorted[::-1]

for i in top\_20\_desc:

print(int(coefs[i]\*100), tfidf[i])

d = {}

for a, x in zip(coefs, tfidf):

d[x] = a

import matplotlib.pyplot as plt

from wordcloud import WordCloud

wordcloud = WordCloud(background\_color='white', width=2400, height=1800)

wordcloud.generate\_from\_frequencies(frequencies=d)

plt.figure()

plt.tight\_layout(pad=0)

plt.imshow(wordcloud, interpolation="bilinear")

plt.axis("off")

plt.show()