This Jupyther Notebook is prepaired by Brantley Deines

Load Data and Perform Basic EDA

I - Import Libraries

```
import pandas as pd
import seaborn as sns
import numpy as np
import nltk
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem import LancasterStemmer
from nltk.corpus import stopwords
import string
```

```
nltk.download('wordnet') # we will use wordnet lemmatizer
nltk.download('averaged_perceptron_tagger') #to be used for tagger
nltk.download('punkt')
nltk.download('stopwords')
```

```
[nltk_data] Error loading wordnet: <urlopen error [Errno -2] Name or
[nltk_data]     service not known>
[nltk_data] Error loading averaged_perceptron_tagger: <urlopen error
[nltk_data]     [Errno -2] Name or service not known>
[nltk_data] Error loading punkt: <urlopen error [Errno -2] Name or
[nltk_data]     service not known>
[nltk_data] Error loading stopwords: <urlopen error [Errno -2] Name or
[nltk_data]     service not known>
False
```

```
stemmer = LancasterStemmer()
stopwordList = stopwords.words('english')
```

```
def text_process(mess):
    """
    Takes in a string of text, then performs the following:
    1. Remove all punctuation
```

```
    3. convert them to lower case
    4. Remove all stopwords
    3. Perform stemming
    4. Returns a list of the cleaned text
    """



    # Check characters to see if they are in punctuation
    mess = [char for char in mess if char not in string.punctuation]
     # Join the characters again to form the string.
    mess = ''.join(mess)

    words = nltk.word_tokenize(mess)
    words = [t for t in words if t not in stopwordList]
    words = [stemmer.stem(w.lower()) for w in words]

    return words
```

II - Read tho File, Create List, and Show First 10 Items

```
texts = [line.rstrip() for line in open('news.csv')]
for text in texts[1:11]:
    print(text + '\n')
```

```
As U.S. budget fight looms, Republicans flip their fiscal script        WASHI

U.S. military to accept transgender recruits on Monday: Pentagon        WASHI

Senior U.S. Republican senator: 'Let Mr. Mueller do his job'    WASHINGTON (R

FBI Russia probe helped by Australian diplomat tip-off: NYT     WASHINGTON (R

Trump wants Postal Service to charge 'much more' for Amazon shipments   SEATT

White House, Congress prepare for talks on spending, immigration        WEST

Trump says Russia probe will be fair, but timeline unclear: NYT WEST PALM BEA

Factbox: Trump on Twitter (Dec 29) - Approval rating, Amazon    The following

Trump on Twitter (Dec 28) - Global Warming      The following statements were

Alabama official to certify Senator-elect Jones today despite challenge: CNN
```

III - Show How the Data is Sepereated, and Load Data into DataFrame

```
data = pd.read_csv('news.csv', sep = '\t')
data.head()
```

	title	text	subject	date	target
0	As U.S. budget fight looms, Republicans flip t...	WASHINGTON (Reuters) - The head of a conservat...	politicsNews	December 31, 2017	1
1	U.S. military to accept transgender recruits o...	WASHINGTON (Reuters) - Transgender people will...	politicsNews	December 29, 2017	1

Senior U.S.

IV - Check for Null Values and Remove any Colums with Null Values

```
data.isnull().sum().sort_values(ascending = False)
```

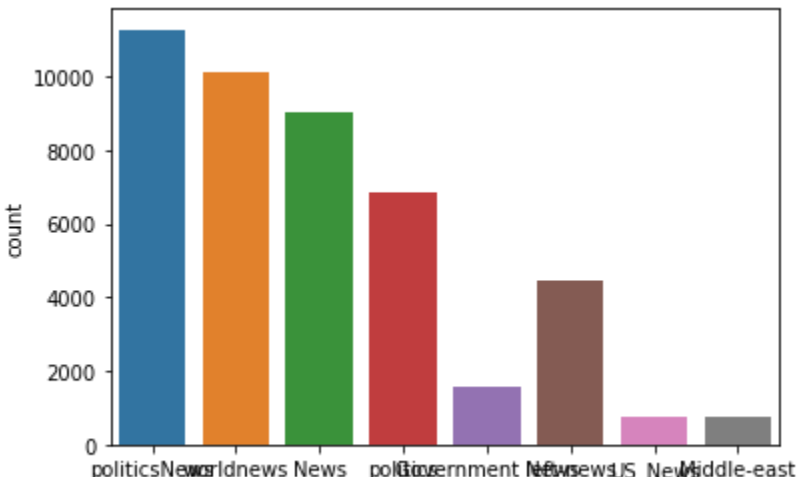
```
title      0
text       0
subject    0
date       0
target     0
dtype: int64
```

no null values to remove

V - Generate Counterplot to Show Number of News in Each Subject

```
sns.countplot(x = 'subject', data = data)
```

```
<AxesSubplot:xlabel='subject', ylabel='count'>
```



subject

VI - Generate Counterplot to Show Nu ber of News in Each Category

[ ] ↳ *1 cell hidden*

VII - Generate 2 Word Clouds , Write th Most Common Words in Each Category

[ ] ↳ *3 cells hidden*

VIII - Create "AllText" Column that contains the Concatenation of 'subject' , 'title' , 'text'

[ ] ↳ *1 cell hidden*

IX - Copy DataFrame to a New DataFrame

[ ] ↳ *1 cell hidden*

X - Drop the Columns Used to Create 'AllText'

[ ] ↳ *1 cell hidden*

XI - Calculate Length of 'AllText' and put it in a New Column 'length'

[ ] ↳ *1 cell hidden*

XII - Plot a Histrogram for Each Category

[ ] ↳ *3 cells hidden*

XIII - What is TFIDF, How do you use SKLearn to create a Bag of Words, and how do you generate TFIDF for the Bag of Words?

↳ *1 cell hidden*

2 - Train Test Split

I - Import Related Libraries and Perform Split With 20% in Test

[ ] *4 cells hidden*

II - Use Count Plot to Show Distribution of Real Vs. Fake News are in each Train and Test

[ ] *2 cells hidden*

3 - Training and Testing Classifier Useing MultinominalNB

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.feature_extraction.text import TfidfTransformer
```

I - Create a pipeline that will use countVectorizer with the function you have created earlier for data preprocessing, then use Tftransformer and then use the NaiveBayes classifier

[ ] *1 cell hidden*

II - Fit the Pipeline and Perform Prediction

[ ] *2 cells hidden*

III - Generate Classification Report and Cofusion Matrix

[ ] *2 cells hidden*

IV - Discuss Performance

1 cell hidden

V - Copy News From a Website and use the Model to Predict it

[ ] ↳ *3 cells hidden*

4 - Training and Testing a Deep Nural Network

I - Import Related Libraries

[ ] ↳ *1 cell hidden*

II - Create a pipeline like 3i, for MLPClassfier you should use at least two layers and also should verbose = 2

[ ] ↳ *1 cell hidden*

III - Fit the Pipeline and Predict

[ ] ↳ *2 cells hidden*

IV - Generate Classification Report and Confusion Matrix

[ ] ↳ *2 cells hidden*

V - Discus Performance

↳ *1 cell hidden*

VI - Use the Same News From 3V and Run it Through This Model

[ ] ↳ *2 cells hidden*

VII - Discuss Performance Diffecences Between this and NB Models

↳ *1 cell hidden*

5 - Extra Credit

```
target = cData['subject']
cData = cData.drop(['subject', 'AllText'], axis = 1)
cols = ['title', 'text']
cData['AllText'] = cData[cols].apply(lambda row: '_'.join(row.values.astype(str)),
cData = cData.drop(cols, axis = 1)
```

```
ctext_train, ctext_test, clabel_train, clabel_test = train_test_split(cData['AllTe
```

```
cpipeline = Pipeline([
    ('bow', CountVectorizer(analyzer=text_process)),  # strings to token integer co
    ('tfidf', TfidfTransformer()),  # integer counts to weighted TF-IDF scores
    ('classifier', MLPClassifier(hidden_layer_sizes=(100,50), random_state=0, earl
])
```

```
cpipeline.fit(ctext_train, clabel_train)
```

```
     This problem is unconstrained.
    RUNNING THE L-BFGS-B CODE

               * * *

    Machine precision = 2.220D-16
     N =     17997658     M =           10

    At X0         0 variables are exactly at the bounds

    At iterate    0    f=  2.09569D+00    |proj g|=  1.38949D-01

    At iterate    1    f=  1.82194D+00    |proj g|=  4.49171D-02

    At iterate    2    f=  1.79004D+00    |proj g|=  3.05059D-02

    At iterate    3    f=  1.77449D+00    |proj g|=  1.38607D-02

    At iterate    4    f=  1.77003D+00    |proj g|=  9.87100D-03

    At iterate    5    f=  1.75734D+00    |proj g|=  2.72810D-02

    At iterate    6    f=  1.68615D+00    |proj g|=  9.61442D-02

    At iterate    7    f=  1.63513D+00    |proj g|=  1.39238D-01

    At iterate    8    f=  1.46958D+00    |proj g|=  1.78277D-01

    At iterate    9    f=  1.29675D+00    |proj g|=  1.25346D-01

    At iterate   10    f=  1.17846D+00    |proj g|=  1.89861D-01
```

```
At iterate   11    f=  1.04521D+00    |proj g|=  6.63906D-02

At iterate   12    f=  9.85721D-01    |proj g|=  1.01726D-01

At iterate   13    f=  9.40625D-01    |proj g|=  4.51589D-02

At iterate   14    f=  9.14563D-01    |proj g|=  1.31128D-01

At iterate   15    f=  8.84143D-01    |proj g|=  7.77328D-02

At iterate   16    f=  8.57835D-01    |proj g|=  7.80861D-02

At iterate   17    f=  8.48008D-01    |proj g|=  4.50666D-02

At iterate   18    f=  8.31621D-01    |proj g|=  3.29933D-02

At iterate   19    f=  8.17996D-01    |proj g|=  5.27520D-02

At iterate   20    f=  8.02182D-01    |proj g|=  6.27798D-02

At iterate   21    f=  7.85700D-01    |proj g|=  4.89266D-02

At iterate   22    f=  7.79577D-01    |proj g|=  3.84905D-02

At iterate   23    f=  7.73525D-01    |proj g|=  2.23859D-02
```

```
cpredict = cpipeline.predict(ctext_test)
```

```
cpredict
```

```
array(['worldnews', 'worldnews', 'politicsNews', ..., 'worldnews',
       'worldnews', 'politics'], dtype='<U15')
```

```
classification_report(clabel_test, cpredict)
```

```
/home/brantley/.local/lib/python3.8/site-packages/sklearn/metrics/_classifica
  _warn_prf(average, modifier, msg_start, len(result))
/home/brantley/.local/lib/python3.8/site-packages/sklearn/metrics/_classifica
  _warn_prf(average, modifier, msg_start, len(result))
/home/brantley/.local/lib/python3.8/site-packages/sklearn/metrics/_classifica
  _warn_prf(average, modifier, msg_start, len(result))
'                 precision    recall  f1-score   support\n\nGovernment News
```

```
confusion_matrix(clabel_test, cpredict)
```

```
array([[   8,    0,   18,    4,   68,  213,    6,   11],
       [  11,    0,   10,  111,   10,    0,    2,    7],
       [   1,    0, 1667,    6,   36,   51,    0,    2],
       [  16,    0,   10,  106,    7,    0,    2,   10],
       [  12,    0,   49,    4,  206,  600,    6,    8],
       [  11,    0,   74,    6,  309,  963,   19,   10],
```

```
[  11,    0,   74,    0,  309,  903,   19,   10],
[   0,    0,    0,    1,    0,   30, 2250,   16],
[   1,    0,    1,    3,    1,   11,   28, 1968]])
```