Separation & Exploratory Analysis Of Data

Bring in the data

imports

In [161]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.utils import shuffle
%matplotlib inline
```

data

In [162]:
```
fake_data = pd.read_csv('Fake.csv.nosync.csv')
real_data = pd.read_csv('Real.csv.nosync.csv')
```

In [163]:
```
fake_data.head()
```

Out[163]:

	title	text	subject	date
0	Donald Trump Sends Out Embarrassing New Year'...	Donald Trump just couldn t wish all Americans ...	News	December 31, 2017
1	Drunk Bragging Trump Staffer Started Russian ...	House Intelligence Committee Chairman Devin Nu...	News	December 31, 2017
2	Sheriff David Clarke Becomes An Internet Joke...	On Friday, it was revealed that former Milwauk...	News	December 30, 2017
3	Trump Is So Obsessed He Even Has Obama's Name...	On Christmas day, Donald Trump announced that ...	News	December 29, 2017
4	Pope Francis Just Called Out Donald Trump Dur...	Pope Francis used his annual Christmas Day mes...	News	December 25, 2017

In [164]:
```
real_data.head()
```

Out[164]:

	title	text	subject	date
0	As U.S. budget fight looms, Republicans flip t...	WASHINGTON (Reuters) - The head of a conservat...	politicsNews	December 31, 2017
1	U.S. military to accept transgender recruits o...	WASHINGTON (Reuters) - Transgender people will...	politicsNews	December 29, 2017
2	Senior U.S. Republican senator: 'Let Mr. Muell...	WASHINGTON (Reuters) - The special counsel inv...	politicsNews	December 31, 2017
3	FBI Russia probe helped by Australian diplomat...	WASHINGTON (Reuters) - Trump campaign adviser ...	politicsNews	December 30, 2017
4	Trump wants Postal Service to charge 'much mor...	SEATTLE/WASHINGTON (Reuters) - President Donal...	politicsNews	December 29, 2017

Adding columns

In [165]:
```
fake_data['real?'] = 'Fake'
```

In [166]:
```
real_data['real?'] = 'Real'
```

Combining Data

In [167]:
```
all_data = pd.concat([real_data, fake_data])
```

In [168]:
```
all_data
```

Out[168]:

	title	text	subject	date	real?
0	As U.S. budget fight looms, Republicans flip t...	WASHINGTON (Reuters) - The head of a conservat...	politicsNews	December 31, 2017	Real
1	U.S. military to accept transgender recruits o...	WASHINGTON (Reuters) - Transgender people will...	politicsNews	December 29, 2017	Real
2	Senior U.S. Republican senator: 'Let Mr. Muell...	WASHINGTON (Reuters) - The special counsel inv...	politicsNews	December 31, 2017	Real
3	FBI Russia probe helped by Australian diplomat...	WASHINGTON (Reuters) - Trump campaign adviser ...	politicsNews	December 30, 2017	Real
4	Trump wants Postal Service to charge 'much mor...	SEATTLE/WASHINGTON (Reuters) - President Donal...	politicsNews	December 29, 2017	Real
...	...	...	...	...	...
23476	McPain: John McCain Furious That Iran Treated ...	21st Century Wire says As 21WIRE reported earl...	Middle-east	January 16, 2016	Fake
23477	JUSTICE? Yahoo Settles E-mail Privacy Class-ac...	21st Century Wire says It s a familiar theme. ...	Middle-east	January 16, 2016	Fake
23478	Sunnistan: US and Allied ‘Safe Zone’ Plan to T...	Patrick Henningsen 21st Century WireRemember ...	Middle-east	January 15, 2016	Fake
23479	How to Blow $700 Million: Al Jazeera America F...	21st Century Wire says Al Jazeera America will...	Middle-east	January 14, 2016	Fake
23480	10 U.S. Navy Sailors Held by Iranian Military ...	21st Century Wire says As 21WIRE predicted in ...	Middle-east	January 12, 2016	Fake

44898 rows × 5 columns

Adding Numericals, Dropping, Shuffling

Adding Some Numerical Columns

In [169]:
```
# Gather numerical features of the data
all_data['title_len'] = all_data.title.apply(len)
all_data['text_len'] = all_data.text.apply(len)
```

Label Encoding Binary Column

In [170]:
```
# Need to convert the Real? column to numerical prior to separating all
 of the numerical columns
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
all_data['true'] = le.fit_transform(all_data['real?'].values)
```

In [171]:
```
all_data
```

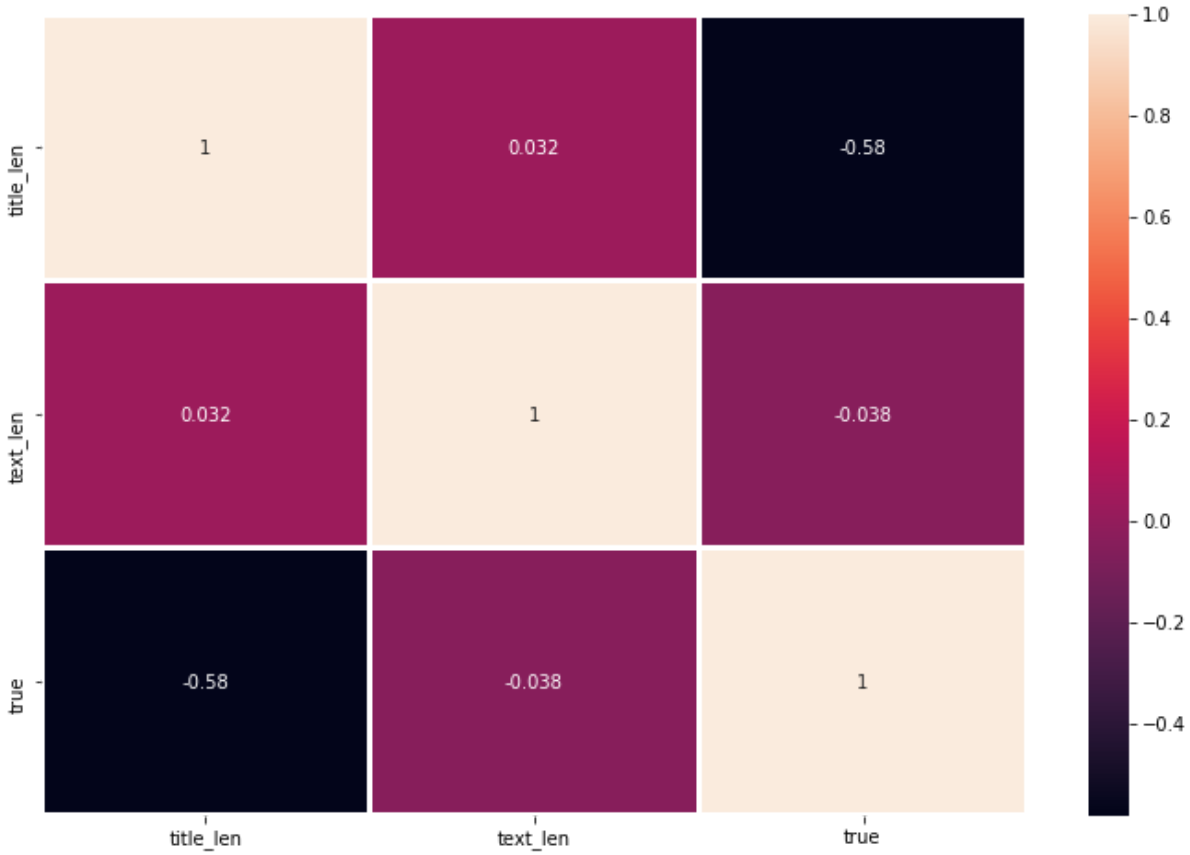
Out[171]:

	title	text	subject	date	real?	title_len	text_len	true
0	As U.S. budget fight looms, Republicans flip t...	WASHINGTON (Reuters) - The head of a conservat...	politicsNews	December 31, 2017	Real	64	4659	1
1	U.S. military to accept transgender recruits o...	WASHINGTON (Reuters) - Transgender people will...	politicsNews	December 29, 2017	Real	64	4077	1
2	Senior U.S. Republican senator: 'Let Mr. Muell...	WASHINGTON (Reuters) - The special counsel inv...	politicsNews	December 31, 2017	Real	60	2789	1
3	FBI Russia probe helped by Australian diplomat...	WASHINGTON (Reuters) - Trump campaign adviser ...	politicsNews	December 30, 2017	Real	59	2461	1
4	Trump wants Postal Service to charge 'much mor...	SEATTLE/WASHINGTON (Reuters) - President Donal...	politicsNews	December 29, 2017	Real	69	5204	1
...	...	...	...	...	...	...	...	...
23476	McPain: John McCain Furious That Iran Treated ...	21st Century Wire says As 21WIRE reported earl...	Middle-east	January 16, 2016	Fake	61	3237	0
23477	JUSTICE? Yahoo Settles E-mail Privacy Class-ac...	21st Century Wire says It s a familiar theme. ...	Middle-east	January 16, 2016	Fake	81	1684	0
23478	Sunnistan: US and Allied ‘Safe Zone’ Plan to T...	Patrick Henningsen 21st Century WireRemember ...	Middle-east	January 15, 2016	Fake	85	25065	0
23479	How to Blow $700 Million: Al Jazeera America F...	21st Century Wire says Al Jazeera America will...	Middle-east	January 14, 2016	Fake	67	2685	0
23480	10 U.S. Navy Sailors Held by Iranian Military ...	21st Century Wire says As 21WIRE predicted in ...	Middle-east	January 12, 2016	Fake	81	5251	0

44898 rows × 8 columns

Quick Heat Map of the Data

In [172]:
```
corr = all_data.corr()
plt.figure(figsize=(12,8))
ax = sns.heatmap(corr, annot=True, cbar=True,linewidths=2.0)
```



In [173]:
```
# Combine the two text columns in order to allow NB analysis later
all_data['full_text'] = all_data['title'] + all_data['text']
# Remove extraneous columns
all_data.drop(['subject','date','text','title'],axis=1,inplace=True)
# Shuffle data so that when a subset is used for model training we don't
bias toward one outcome or another
all_data = shuffle(all_data)
all_data.reset_index(inplace=True, drop=True)
```

In [174]:
```
all_data.sample(10)
```

Out[174]:

	real?	title_len	text_len	true	full_text
34977	Real	74	4007	1	Trump Cabinet's First World problem: omitting ...
44378	Fake	95	2120	0	JUDGE GIVES ‘LEGAL PERSONHOOD’ TO MONKEYS: SHO...
29572	Fake	49	4138	0	While We Were Sleeping, Trump Declared Civil ...
20930	Real	60	630	1	Britain will honour commitments made to EU - M...
44455	Real	48	530	1	Israel's Netanyahu to speak with Trump on Sund...
20776	Fake	97	3453	0	Trump Meets With Russian Officials One Day Af...
17898	Real	53	400	1	Saudi Arabia names Nabeel al-Amudi transport m...
6477	Fake	84	3084	0	New Ad Proves Hillary’s Been Fighting For Inc...
6379	Fake	78	2373	0	EXPLOSIVE New Report PROVES Trump Hotel Linke...
36504	Real	73	2058	1	Having nuclear weapons 'matter of life and dea...

Separating Numerical & Text Data into Separate Frames

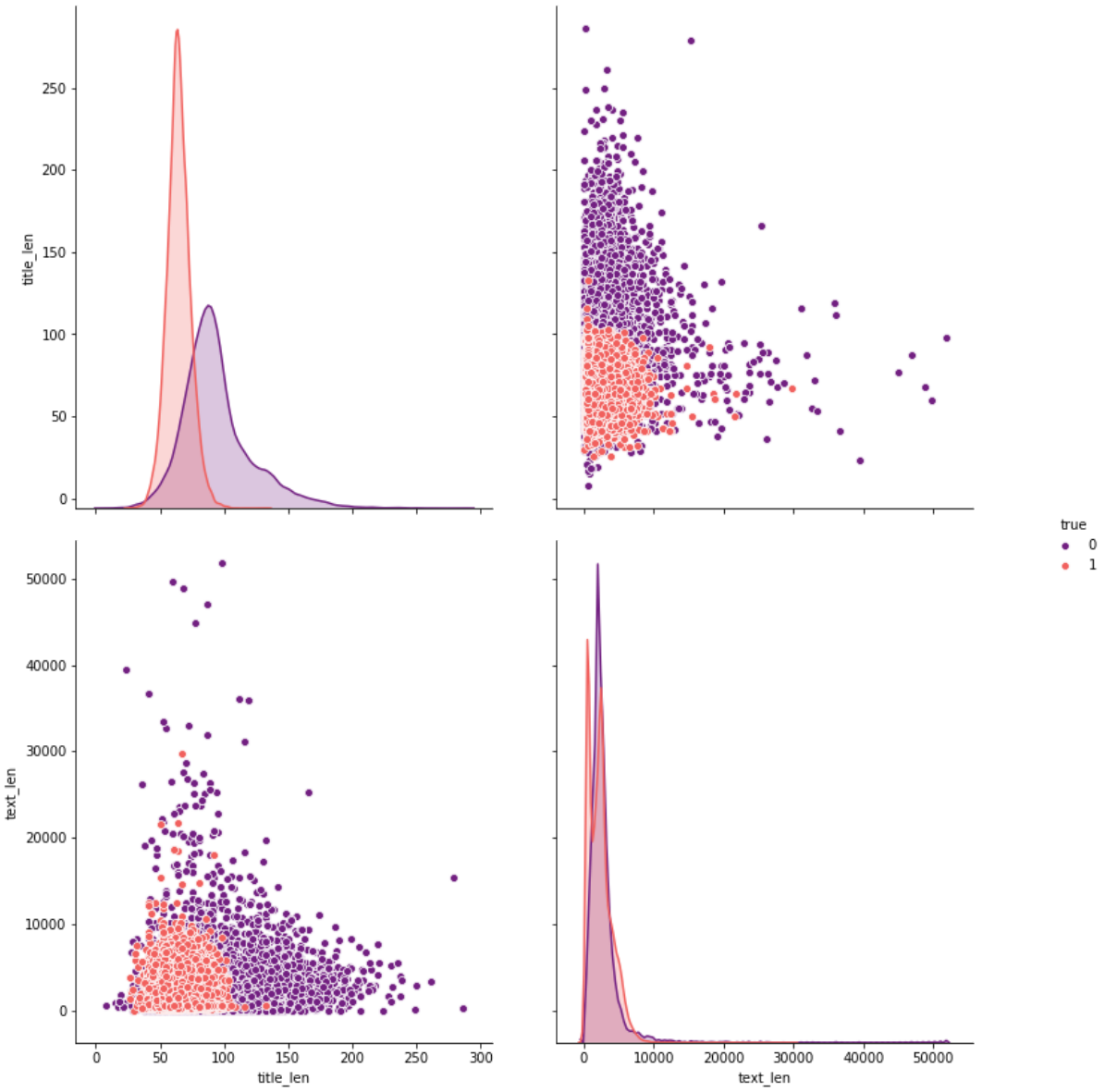
In [175]:
```
# Separating out numerical data into its own dataframe
num_data = all_data.loc[:][['title_len', 'text_len', 'true']]
```

In [176]:
```
text_data = all_data.loc[:][['full_text','real?']]
```

Visualize The Numerical Data

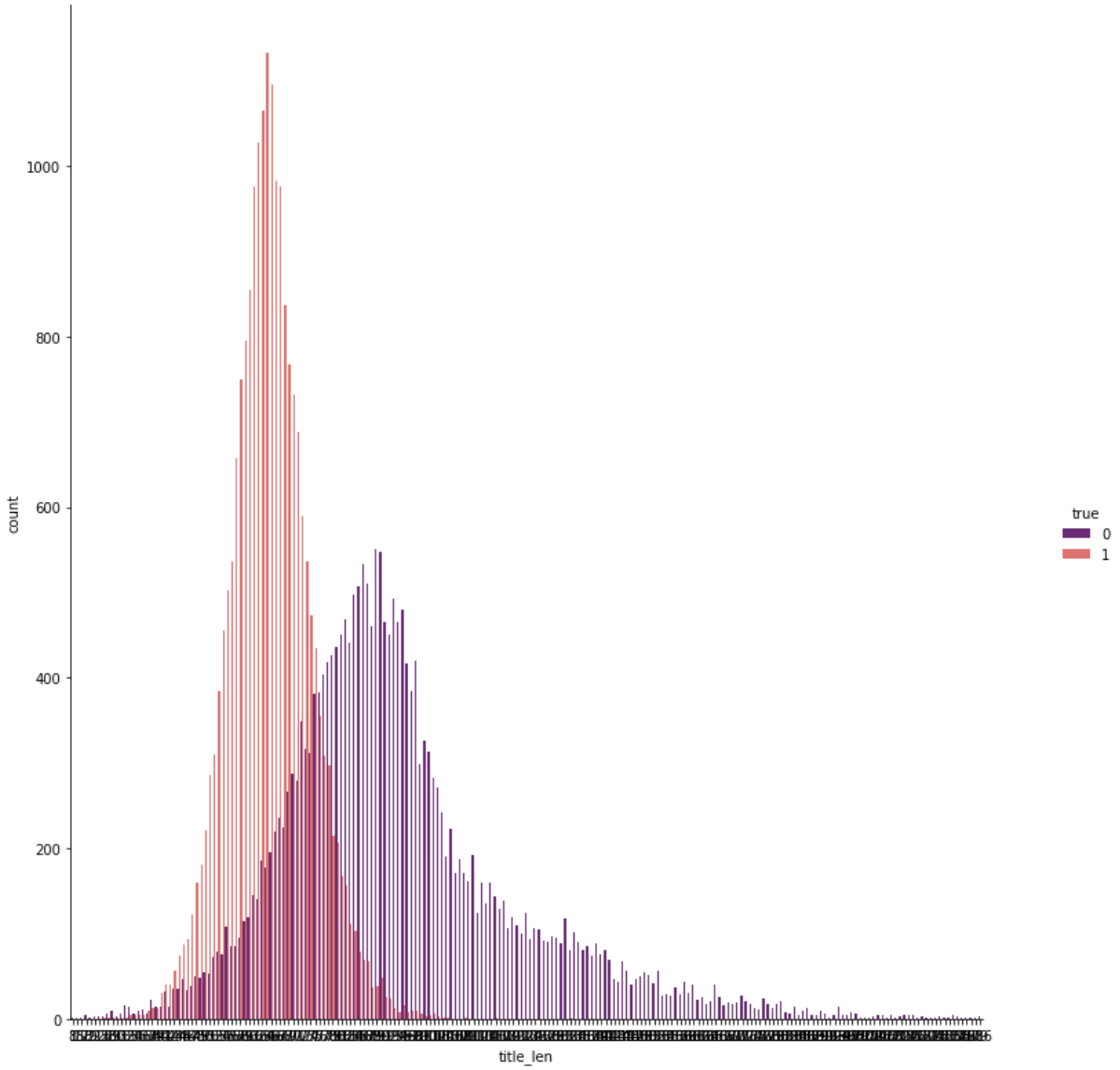
Pairplots To Start...

In [177]:
```
g = sns.pairplot(num_data, hue='true', palette='magma')
g.fig.set_size_inches(12,12)
```



Veracity vs. Title Length

In [178]:
```
h = sns.catplot('title_len',data=num_data, hue='true',kind='count',palette='magma')
h.fig.set_size_inches(12,12)
```

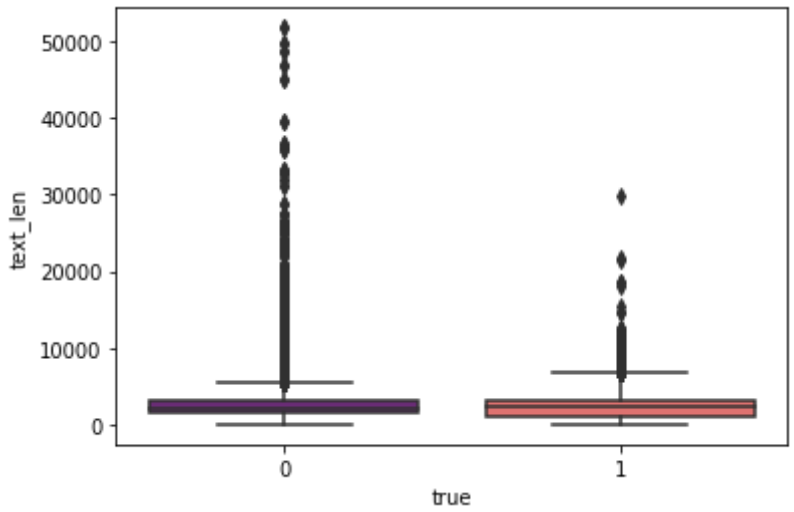


It seems that most of the fake news articles contain article titles which are longer than the true news articles.

Veracity vs. Article Length

In [179]:
```
sns.boxplot(x='true',y='text_len',data=num_data, palette='magma')
```

Out[179]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff56cc338d0>
```



There doesn't seem to be much connection between article length and article veracity, though the fake news stories certainly have a longer tail in terms of word count.

Working With The Text Features

In [180]:
```
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransf
ormer
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from nltk.corpus import stopwords
```

Extracting a sample of six articles for comparison with a fully trained model and use in my web application

In [182]:
```
sample_of_six = text_data[0:6][:]
sample_of_six.to_csv('sample_data',index=False)
sample_of_six
```

Out[182]:

	full_text	real?
0	LOL! CROWD CHANTS “CNN Sucks” At Trump Rally W...	Fake
1	UK Diplomat Says He’s Met With DNC Leaker…They...	Fake
2	Trump attacks Clinton on gender, risking backl...	Real
3	Trump signs repeal of U.S. broadband privacy r...	Real
4	Trump taps Michigan Republican DeVos for educa...	Real
5	Exclusive: Trump supporters more likely to vie...	Real

Dropping the first 6 indexes to be used as samples once the models are trained

In [183]:
```
text_data.drop(text_data.index[0:6], inplace=True)
text_data
```

Out[183]:

	full_text	real?
6	In 2017, America’s liberal abortion agenda loo...	Fake
7	Trump Hesitant To Negotiate With Schumer On H...	Fake
8	Merkel points to grand coalition with Social D...	Real
9	Trump Just Proved He Has The Foreign Diplomac...	Fake
10	Denmark passes law that could ban Russian pipe...	Real
...	...	...
44893	James Comey Just Called Trump A Liar, And The...	Fake
44894	VA Middle School Student Accused Of Stealing ...	Fake
44895	Trump travel restrictions hit demand for visit...	Real
44896	A MUST WATCH! Mark Steyn Calls Out Political V...	Fake
44897	Czech president plans to appoint Babis's cabin...	Real

44892 rows × 2 columns

The standard string.punctuation object contains an apostrophe which can screw up some of the stopword removals, so I am creating my own collection of punctuation for removal later

In [184]:
```
my_punc = ['!','"','#','$','%','&','\\','(',')','*','+',',','-','.','/',
':',';','<','=','>','?','@','\[','\]','^','_','`','{','|','}','~']
```

Creating my own string processor. I'm sure there is a tokenizer available that would do this for me somehow, but I couldn't find it and got tired of looking.

In [185]:
```
def clean_string(string):
    """
    1. Tokenize Words
    2. Remove Punctuation
    3. Remove Stop Words
    """
    nopunc_list = (''.join(char for char in string if char not in my_pun
c)).split()
    del_stopwords = [word for word in nopunc_list if word.lower() not in
stopwords.words('english')]
    return del_stopwords
```

In [186]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
# Splitting Data With Just the Titles
text_X = text_data['full_text'][:].values
text_y = text_data['real?'][:].values
text_X_train, text_X_test, text_y_train, text_y_test = train_test_split(
text_X,text_y,test_size=0.2,random_state=1)
```

Creating the pipeline for the Random Forest Classifier

In [187]:
```
RF_pipeline = Pipeline([
    ('bow', CountVectorizer(analyzer=clean_string)),
    ('tfidf', TfidfTransformer()),
    ('classifier', RandomForestClassifier())
])
```

In [189]:
```
RF_pipeline.fit(text_X_train,text_y_train)
```

Out[189]:
```
Pipeline(memory=None,
         steps=[('bow',
                 CountVectorizer(analyzer=<function clean_string at 0x7
ff560a84dd0>,
                                 binary=False, decode_error='strict',
                                 dtype=<class 'numpy.int64'>, encoding
='utf-8',
                                 input='content', lowercase=True, max_d
f=1.0,
                                 max_features=None, min_df=1,
                                 ngram_range=(1, 1), preprocessor=None,
                                 stop_words=None, strip_accents=None,
                                 token_pattern='(?u)\\b\\w\\w+\\b...
                 RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
                                        class_weight=None, criterion='g
ini',
                                        max_depth=None, max_features='a
uto',
                                        max_leaf_nodes=None, max_sample
s=None,
                                        min_impurity_decrease=0.0,
                                        min_impurity_split=None,
                                        min_samples_leaf=1, min_samples
_split=2,
                                        min_weight_fraction_leaf=0.0,
                                        n_estimators=100, n_jobs=None,
                                        oob_score=False, random_state=N
one,
                                        verbose=0, warm_start=False))],
         verbose=False)
```

In [190]:
```
rf_preds = RF_pipeline.predict(text_X_test)
```

In [191]:
```
rf_class_rep = classification_report(text_y_test, rf_preds)
print(rf_class_rep)
```

```
              precision    recall  f1-score   support

        Fake       1.00      0.99      0.99      4730
        Real       0.99      1.00      0.99      4249

    accuracy                           0.99      8979
   macro avg       0.99      0.99      0.99      8979
weighted avg       0.99      0.99      0.99      8979
```

Pretty impressive, but those numbers are a little *too* good. Let's do a couple of sample tests from the data which weren't included in the training data.

In [212]:
```
art_1 = sample_of_six['full_text'][0]
actual_1 = sample_of_six['real?'][0]
print(f'Predicted: {RF_pipeline.predict([art_1])}')
print(f'Actual: {actual_1}')
print(f'Prediction probability: {np.max(RF_pipeline.predict_proba([art_
1])*100)}')
```

```
Predicted: ['Fake']
Actual: Fake
Prediction probability: 98.0
```

Correct determination, great confidence...

In [214]:
```
art_2 = sample_of_six['full_text'][2]
actual_2 = sample_of_six['real?'][2]
print(f'Predicted: {RF_pipeline.predict([art_2])}')
print(f'Actual: {actual_2}')
print(f'Prediction probability: {np.max(RF_pipeline.predict_proba([art_
2])*100)}')
```

```
Predicted: ['Real']
Actual: Real
Prediction probability: 76.0
```

Another correct determination and less confidence. Let's see what a Naive Bayes classifier can achieve.

Creating the pipeline for the Naive Bayes Classifier

In [194]:
```
NB_pipeline = Pipeline([
    ('bow', CountVectorizer(analyzer=clean_string)),
    ('tfidf', TfidfTransformer()),
    ('classifier', MultinomialNB())
])
```

In [195]:
```
NB_pipeline.fit(text_X_train, text_y_train)
```

Out[195]:
```
Pipeline(memory=None,
         steps=[('bow',
                 CountVectorizer(analyzer=<function clean_string at 0x7
ff560a84dd0>,
                                 binary=False, decode_error='strict',
                                 dtype=<class 'numpy.int64'>, encoding
='utf-8',
                                 input='content', lowercase=True, max_d
f=1.0,
                                 max_features=None, min_df=1,
                                 ngram_range=(1, 1), preprocessor=None,
                                 stop_words=None, strip_accents=None,
                                 token_pattern='(?u)\\b\\w\\w+\\b',
                                 tokenizer=None, vocabulary=None)),
                ('tfidf',
                 TfidfTransformer(norm='l2', smooth_idf=True,
                                  sublinear_tf=False, use_idf=True)),
                ('classifier',
                 MultinomialNB(alpha=1.0, class_prior=None, fit_prior=T
rue))],
         verbose=False)
```

Running the test data through the model

In [196]:
```
text_preds = NB_pipeline.predict(text_X_test)
```

In [197]:
```
nb_class_rep = classification_report(text_y_test, text_preds)
print(nb_class_rep)
```

```
              precision    recall  f1-score   support

        Fake       0.98      0.96      0.97      4730
        Real       0.95      0.98      0.97      4249

    accuracy                           0.97      8979
   macro avg       0.97      0.97      0.97      8979
weighted avg       0.97      0.97      0.97      8979
```

That looks pretty good. Let's do the same couple of sample tests from the data as we did above for the RFC

In [215]:
```
print(f'Predicted: {NB_pipeline.predict([art_1])}')
print(f'Actual: {actual_1}')
print(f'Prediction probability: {np.max(NB_pipeline.predict_proba([art_
1])*100)}')
```

```
Predicted: ['Fake']
Actual: Fake
Prediction probability: 99.16431194506887
```

Nice! Correct answer and fantastic probability. That's a slight improvement from the RFC confidence of 98%.

In [216]:
```
# art_2 = text_data['full_text'][44897]
# actual_2 = text_data['real?'][44897]
print(f'Predicted: {NB_pipeline.predict([art_2])}')
print(f'Actual: {actual_2}')
print(f'Prediction probability: {np.max(NB_pipeline.predict_proba([art_
2])*100)}')
```

```
Predicted: ['Real']
Actual: Real
Prediction probability: 80.66595345331345
```

Another correct determination, and a bit better than the RFC confidence of 76%.

Perhaps a joint effort would be worthwhile...

Creating A Voting Classifier To Combine RFC & NB

In [200]:
```
from sklearn.ensemble import VotingClassifier

model_combo = VotingClassifier(estimators=[('RandFor', RF_pipeline),
                                           ('NaiveBayes', NB_pipeline)],
                               voting='soft',weights=(1,2))
model_combo.fit(text_X_train,text_y_train)
```

Out[200]:
```
VotingClassifier(estimators=[('RandFor',
                              Pipeline(memory=None,
                                       steps=[('bow',
                                               CountVectorizer(analyzer
=<function clean_string at 0x7ff560a84dd0>,
                                                               binary=F
alse,
                                                               decode_e
rror='strict',
                                                               dtype=<c
lass 'numpy.int64'>,
                                                               encoding
='utf-8',
                                                               input='c
ontent',
                                                               lowercas
e=True,
                                                               max_df=
1.0,
                                                               max_feat
ures=None,
                                                               min_df=
1,
                                                               ngram_ra
nge=(1,
1),
                                                               preproce
ssor=None,
                                                               stop_wor
ds=None,
                                                               strip_ac
cen...
                                                               preproce
ssor=None,
                                                               stop_wor
ds=None,
                                                               strip_ac
cents=None,
                                                               token_pa
ttern='(?u)\\b\\w\\w+\\b',
                                                               tokenize
r=None,
                                                               vocabula
ry=None)),
                                              ('tfidf',
                                               TfidfTransformer(norm='l
2',
                                                                smooth_
idf=True,
                                                                subline
ar_tf=False,
                                                                use_idf
=True)),
                                              ('classifier',
                                               MultinomialNB(alpha=1.0,
                                                             class_prio
```

```
r=None,
                                                                    fit_prior=
True))],
                                          verbose=False))],
                   flatten_transform=True, n_jobs=None, voting='soft',
                   weights=(1, 2))
```

In [201]:
```
combo_pred = model_combo.predict(text_X_test)
```

In [202]:
```
combo_class_rep = classification_report(text_y_test, combo_pred)
print(combo_class_rep)
```

```
              precision    recall  f1-score   support

        Fake       0.99      0.97      0.98      4730
        Real       0.97      0.99      0.98      4249

    accuracy                           0.98      8979
   macro avg       0.98      0.98      0.98      8979
weighted avg       0.98      0.98      0.98      8979
```

Similarly nice numbers here, now let's check the same samples from the previous two checks.

In [217]:
```
print(f'Predicted: {model_combo.predict([art_1])}')
print(f'Actual: {actual_1}')
print(f'Prediction probability: {np.max(model_combo.predict_proba([art_
1])*100)}')
```

```
Predicted: ['Fake']
Actual: Fake
Prediction probability: 98.10954129671258
```

Right around the middle of the two models individually on the same sample.

In [218]:
```
print(f'Predicted: {model_combo.predict([art_2])}')
print(f'Actual: {actual_2}')
print(f'Prediction probability: {np.max(model_combo.predict_proba([art_
2])*100)}')
```

```
Predicted: ['Real']
Actual: Real
Prediction probability: 79.44396896887565
```

Once again a good balance between the two previous models. Previous RFC confidence of 76% and the NB confidence of 80%.

Preparing to export and save the model

In [205]:
```
import pickle

model_combo_pkl = "model_combo.pkl"
with open(model_combo_pkl, 'wb') as file:
    pickle.dump(model_combo, file)
```