In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn import feature_extraction, linear_model, model_selection, preprocessing
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
```

In [2]:
```
fake = pd.read_csv("Fake(1).csv")
true = pd.read_csv("True(1).csv")
```

In [4]:
```
fake.shape
```
Out[4]: (23481, 4)

In [5]:
```
true.shape
```
Out[5]: (21417, 4)

In [6]:
```
# Add flag to track fake and real
fake['target'] = 'fake'
true['target'] = 'true'
```

In [7]:
```
# Concatenate dataFrames
data = pd.concat([fake, true]).reset_index(drop = True)
data.shape
```
Out[7]: (44898, 5)

In [8]:
```
# Shuffle the data
from sklearn.utils import shuffle
data = shuffle(data)
data = data.reset_index(drop=True)
```

In [9]:
```
# Check the data
data.head()
```
Out[9]:

	title	text	subject	date	target
0	WATCH: Trump Loves The GOP Healthcare Bill Bu...	If Trump isn t willing to attach his name to s...	News	March 8, 2017	fake
1	New 9/11 Trailer – Featuring Charlie Sheen and...	21st Century Wire says Everything changed on 9...	US_News	July 29, 2017	fake
2	RAW VIDEO: HILLARY AND BILL CLINTON DANCIN' AN...	Hillary has to be the absolute worst dancer ev...	politics	Aug 17, 2015	fake
3	EU ready for climate leadership in case of U.S. ...	BRUSSELS (Reuters) - A withdrawal by the Unite...	politicsNews	May 31, 2017	true
4	U.S. urges authorities to review Honduras elec...	WASHINGTON (Reuters) - The United States urged...	worldnews	November 29, 2017	true

In [10]:
```
# Remove the date (we won't use it for the analysis)
data.drop(["date"],axis=1,inplace=True)
data.head()
```
Out[10]:

	title	text	subject	target
0	WATCH: Trump Loves The GOP Healthcare Bill Bu...	If Trump isn t willing to attach his name to s...	News	fake
1	New 9/11 Trailer – Featuring Charlie Sheen and...	21st Century Wire says Everything changed on 9...	US_News	fake
2	RAW VIDEO: HILLARY AND BILL CLINTON DANCIN' AN...	Hillary has to be the absolute worst dancer ev...	politics	fake
3	EU ready for climate leadership in case of U.S. ...	BRUSSELS (Reuters) - A withdrawal by the Unite...	politicsNews	true
4	U.S. urges authorities to review Honduras elec...	WASHINGTON (Reuters) - The United States urged...	worldnews	true

In [11]:
```
# Removing the title (we will only use the text)
data.drop(["title"],axis=1, inplace=True)
data.head()
```
Out[11]:

	text	subject	target
0	If Trump isn t willing to attach his name to s...	News	fake
1	21st Century Wire says Everything changed on 9...	US_News	fake
2	Hillary has to be the absolute worst dancer ev...	politics	fake
3	BRUSSELS (Reuters) - A withdrawal by the Unite...	politicsNews	true
4	WASHINGTON (Reuters) - The United States urged...	worldnews	true

In [12]:
```
# Convert to lowercase
data['text'] = data['text'].apply(lambda x: x.lower())
data.head()
```
Out[12]:

	text	subject	target
0	if trump isn t willing to attach his name to s...	News	fake
1	21st century wire says everything changed on 9...	US_News	fake
2	hillary has to be the absolute worst dancer ev...	politics	fake
3	brussels (reuters) - a withdrawal by the unite...	politicsNews	true
4	washington (reuters) - the united states urged...	worldnews	true

In [13]:
```
# Remove punctuation
import string

def punctuation_removal(text):
    all_list = [char for char in text if char not in string.punctuation]
    clean_str = ''.join(all_list)
    return clean_str

data['text'] = data['text'].apply(punctuation_removal)
```

In [14]:
```
# Check
data.head()
```
Out[14]:

	text	subject	target
0	if trump isn t willing to attach his name to s...	News	fake
1	21st century wire says everything changed on 9...	US_News	fake
2	hillary has to be the absolute worst dancer ev...	politics	fake
3	brussels reuters a withdrawal by the unite s...	politicsNews	true
4	washington reuters the united states urged wi...	worldnews	true

In [15]:
```
#Removing stopwords
import nltk
from nltk.corpus import stopwords
stop = stopwords.words('english')

data['text'] = data['text'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop)]))
```
```
[nltk_data] Error loading stopwords: <urlopen error [Errno 11001]
[nltk_data]     getaddrinfo failed>
```
```
data.head()
```
Out[15]:

	text	subject	target
0	trump willing attach name something know must ...	News	fake
1	21st century wire says everything changed 911 ...	US_News	fake
2	hillary absolute worst dancer ever hands air t...	politics	fake
3	brussels reuters withdrawal united states pain...	politicsNews	true
4	washington reuters united states urged electio...	worldnews	true

In [17]:
```
# How many articles per subject?
print(data.groupby(['subject'])['text'].count())
data.groupby(['subject'])['text'].count().plot(kind="bar")
plt.show()
```
```
subject
Government News     1570
Middle-east          778
News                9050
US_News              783
left-news           4459
politics            6841
politicsNews       11272
worldnews          10145
Name: text, dtype: int64
```


In [18]:
```
# How many fake and real articles?
print(data.groupby(['target'])['text'].count())
data.groupby(['target'])['text'].count().plot(kind="bar")
plt.show()
```
```
target
fake    23481
true    21417
Name: text, dtype: int64
```


In [19]:
```
# Most frequent words counter (Code adapted from https://www.kaggle.com/rodolfoluna/fake-new
# s-detector)
from nltk import tokenize

token_space = tokenize.WhitespaceTokenizer()

def counter(text, column_text, quantity):
    all_words = ' '.join([text for text in text[column_text]])
    token_phrase = token_space.tokenize(all_words)
    frequency = nltk.FreqDist(token_phrase)
    df_frequency = pd.DataFrame({"Word": list(frequency.keys()),
                                 "Frequency": list(frequency.values())})
    df_frequency = df_frequency.nlargest(columns = "Frequency", n = quantity)
    plt.figure(figsize=(12,8))
    ax = sns.barplot(data=df_frequency, x= "Word", y = "Frequency", color = 'blue')
    ax.set(ylabel = "Count")
    plt.xticks(rotation='vertical')
    plt.show()
```

In [21]:
```
#Most frequent words in fake news
counter(data[data["target"] == "fake"], "text", 20)
```


In [22]:
```
# Most frequent words in real news
counter(data[data["target"] == "true"], "text", 20)
```


In [23]:
```
# Function to plot the confusion matrix (code from https://scikit-learn.org/stable/auto_exam
# ples/model_selection/plot_confusion_matrix.html)
from sklearn.import metrics
import itertools

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

In [24]:
```
# Split the data
X_train,X_test,y_train,y_test = train_test_split(data['text'], data.target, test_size=0.2, r
andom_state=42)
```

In [25]:
```
# Vectorizing and applying TF-IDF
from sklearn.linear_model import LogisticRegression

pipe = Pipeline([('vect', CountVectorizer()),
                 ('tfidf', TfidfTransformer()),
                 ('model', LogisticRegression())])

# Fitting the model
model = pipe.fit(X_train, y_train)

# Accuracy
prediction = model.predict(X_test)
print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))
```
```
accuracy: 98.88%
```

In [26]:
```
cm = metrics.confusion_matrix(y_test, prediction)
plot_confusion_matrix(cm, classes=['Fake', 'Real'])
```
```
Confusion matrix, without normalization
```


In [28]:
```
from sklearn.tree import DecisionTreeClassifier

# Vectorizing and applying TF-IDF
pipe = Pipeline([('vect', CountVectorizer()),
                 ('tfidf', TfidfTransformer()),
                 ('model', DecisionTreeClassifier(criterion= 'entropy',
                                                  max_depth = 20,
                                                  splitter='best',
                                                  random_state=42))])

# Fitting the model
model = pipe.fit(X_train, y_train)

# Accuracy
prediction = model.predict(X_test)
print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))
```
```
accuracy: 99.61%
```

In [29]:
```
cm = metrics.confusion_matrix(y_test, prediction)
plot_confusion_matrix(cm, classes=['Fake', 'Real'])
```
```
Confusion matrix, without normalization
```


In [31]:
```
from sklearn.ensemble import RandomForestClassifier

pipe = Pipeline([('vect', CountVectorizer()),
                 ('tfidf', TfidfTransformer()),
                 ('model', RandomForestClassifier(n_estimators=50, criterion="entropy"))])

model = pipe.fit(X_train, y_train)
prediction = model.predict(X_test)
print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))
```
```
accuracy: 99.25%
```

In [32]:
```
cm = metrics.confusion_matrix(y_test, prediction)
plot_confusion_matrix(cm, classes=['Fake', 'Real'])
```
```
Confusion matrix, without normalization
```


In [ ]:
```
```