Program Name: B.Sc. Hons. Computer Science
Semester: VI
Title of Paper: Core XIII: Artificial Intelligence
Name of Student: Tarun Verma
Class Roll No.: 17HCS4142
Exam Roll No.: 17015570029

Fake News Detection

Fake news is a term that has been used to describe very different issues, from satirical articles to completely fabricated news and plain government propaganda in some outlets. Fake news, information bubbles, news manipulation and the lack of trust in the media are growing problems with huge ramifications in our society. However, in order to start addressing this problem, we need to have an understanding on what Fake News is. Only then can we look into the different techniques and fields of machine learning (ML), natural language processing (NLP) and artificial intelligence (AI) that could help us fight this situation.

What is Fake News ?

“Fake news” has been used in a multitude of ways in the last half a year and multiple definitions have been given. For instance, the New York times defines it as “a made-up story with an intention to deceive”. This definition focuses on two dimensions: the intentionality (very difficult to prove) and the fact that the story is made up.

First Draft News, an organisation dedicated to improving skills and standards in the reporting and sharing of online information, has published a great article that explains the fake news environment and proposes 7 types of fake content:

1. **False Connection: Headlines, visuals or captions don’t support the content**
2. **False Context: Genuine content is shared with false contextual information**
3. **Manipulated content: Genuine information or imagery is manipulated**
4. **Satire or Parody: No intention to cause harm but potential to fool**
5. **Misleading Content: Misleading use of information to frame an issue/individual**
6. **Imposter Content: Impersonation of genuine sources**
7. **Fabricated content: New content that is 100% false**

In this notebook, we'll build models to for classification of fake news dataset which is available in kaggle librabry.

In [0]:

```
import pandas as pd
import numpy as np
import itertools
import unicodedata
```

```
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
import matplotlib.pyplot as plt
```

In [2]:

```
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

Data Exploration

To begin, we should take a quick look at the data and get to know its contents. To do so, use a Pandas DataFrame and check the shape, head and apply any necessary transformations.

DataSource:
https://www.kaggle.com/rchitic17/real-or-fake

In [0]:

```
# from google.colab import drive
# drive.mount("/content/drive")
# df = pd.read_csv('/content/drive/My Drive/DataSets/fake_or_real_news.csv')

df = pd.read_csv('https://raw.githubusercontent.com/ta-verma/AI-PROJECT/master/fake_or_real_news.csv')
```

In [5]:

```
df.shape
```

Out[5]:

```
(6335, 4)
```

In [6]:

```
df.head()
```

Out[6]:

	Unnamed: 0	title	text	label
0	8476	You Can Smell Hillary's Fear	Daniel Greenfield, a Shillman Journalism Fello...	FAKE
1	10294	Watch The Exact Moment Paul Ryan Committed Pol...	Google Pinterest Digg Linkedin Reddit Stumbleu...	FAKE
2	3608	Kerry to go to Paris in gesture of sympathy	U.S. Secretary of State John F. Kerry said Mon...	REAL
3	10142	Bernie supporters on Twitter erupt in anger ag...	— Kaydee King (@KaydeeKing) November 9, 2016 T...	FAKE
4	875	The Battle of New York: Why This Primary Matters	It's primary day in New York and front-runners...	REAL

In [0]:

```
df = df.set_index('Unnamed: 0')
```

In [8]:

```
df.head()
```

Out[8]:

Unnamed: 0	title	text	label
8476	You Can Smell Hillary's Fear	Daniel Greenfield, a Shillman Journalism Fello...	FAKE
10294	Watch The Exact Moment Paul Ryan Committed Pol...	Google Pinterest Digg Linkedin Reddit Stumbleu...	FAKE
3608	Kerry to go to Paris in gesture of sympathy	U.S. Secretary of State John F. Kerry said Mon...	REAL
10142	Bernie supporters on Twitter erupt in anger ag...	— Kaydee King (@KaydeeKing) November 9, 2016 T...	FAKE
875	The Battle of New York: Why This Primary Matters	It's primary day in New York and front-runners...	REAL

Extracting the training data

In [0]:

```
y = df.label
```

In [0]:

```
df = df.drop('label', axis=1)
```

In [0]:

```
X_train, X_test, y_train, y_test = train_test_split(df['text'], y, test_size=0.33, random_state=53)
```

Building Vectorizer Classifiers

Now that we have our training and testing data, we can build our classifiers. To get a good idea if the words and tokens in the articles had a significant impact on whether the news was fake or real, we begin by using `CountVectorizer` and `TfidfVectorizer`.

This code has a max threshhold set at `.7` for the TF-IDF vectorizer `tfidf_vectorizer` using the `max_df` argument. This removes words which appear in more than 70% of the articles. Also, the built-in `stop_words` parameter will remove English stop words from the data before making vectors.

In [0]:

```
count_vectorizer = CountVectorizer(stop_words='english')
count_train = count_vectorizer.fit_transform(X_train)
```

```
count_test = count_vectorizer.transform(X_test)
```

In [0]:

```
tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)
tfidf_train = tfidf_vectorizer.fit_transform(X_train)
tfidf_test = tfidf_vectorizer.transform(X_test)
```

Now that we have vectors, we can then take a look at the vector features, stored in `count_vectorizer` and `tfidf_vectorizer`.

In [14]:

```
tfidf_vectorizer.get_feature_names()[-10:]
```

Out[14]:

['والمرضى', 'هذا', 'من', 'محاولات', 'ما', 'لم', 'عن', 'عربي', 'حلب', 'ﻟﻪade']

In [15]:

```
count_vectorizer.get_feature_names()[:10]
```

Out[15]:

```
['00',
 '000',
 '0000',
 '00000031',
 '000035',
 '00006',
 '0001',
 '0001pt',
 '000ft',
 '000km']
```

Count versus TF-IDF Features

As we can see by running the cells below, both vectorizers extracted the same tokens, but obviously have different weights. Likely, changing the `max_df` and `min_df` of the TF-IDF vectorizer could alter the result and lead to different features in each.

In [0]:

```
count_df = pd.DataFrame(count_train.A,
columns=count_vectorizer.get_feature_names())
```

In [0]:

```
tfidf_df = pd.DataFrame(tfidf_train.A,
columns=tfidf_vectorizer.get_feature_names())
```

In [18]:

```
set(count_df.columns).symmetric_difference(tfidf_df.columns)
```

Out[18]:

```
set()
```

In [19]:

```
print(count_df.equals(tfidf_df))
```

```
False
```

In [20]:

```
count_df.head()
```

Out[20]:

	00	000	0000	00000031	000035	00006	0001	0001pt	000ft	000km	001	0011	002	003	004	006	00
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	
2	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	
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

5 rows × 56922 columns

In [21]:

```
tfidf_df.head()
```

Out[21]:

	00	000	0000	00000031	000035	00006	0001	0001pt	000ft	000km	001	0011	002	003	004	006	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	
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	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	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	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	0.0	

5 rows × 56922 columns

Comparing Models

In [0]:

```
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() / 2.0
    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')
```

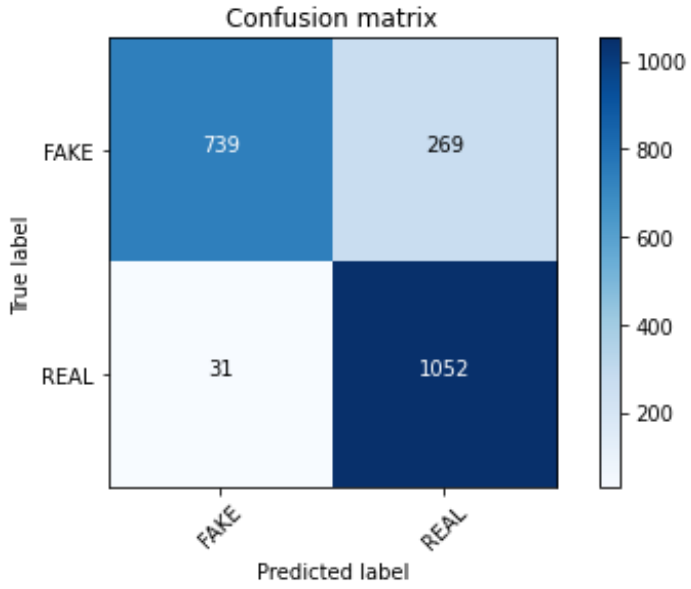
In [0]:

```
clf = MultinomialNB()
```

In [24]:

```
clf.fit(tfidf_train, y_train)
pred = clf.predict(tfidf_test)
score = metrics.accuracy_score(y_test, pred)
print("accuracy:   ", score)
cm = metrics.confusion_matrix(y_test, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
```

```
accuracy:   0.8565279770444764
Confusion matrix, without normalization
```



In [0]:

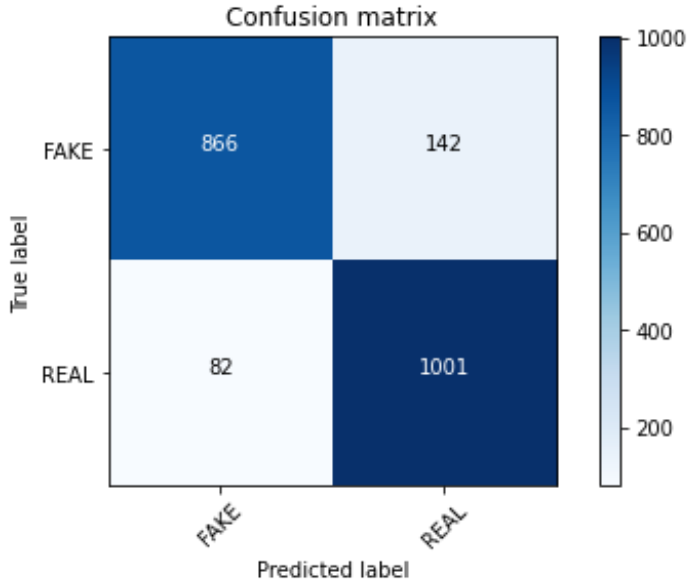
```
clf = MultinomialNB()
```

In [36]:

```
clf.fit(count_train, y_train)
pred = clf.predict(count_test)
score = metrics.accuracy_score(y_test, pred)
print("accuracy:    " ,score)
cm = metrics.confusion_matrix(y_test, pred, labels=['FAKE', 'REAL'])
print('\n\n')
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
```

```
accuracy:    0.8928742228598756
```

```
Confusion matrix, without normalization
```



Linear Model

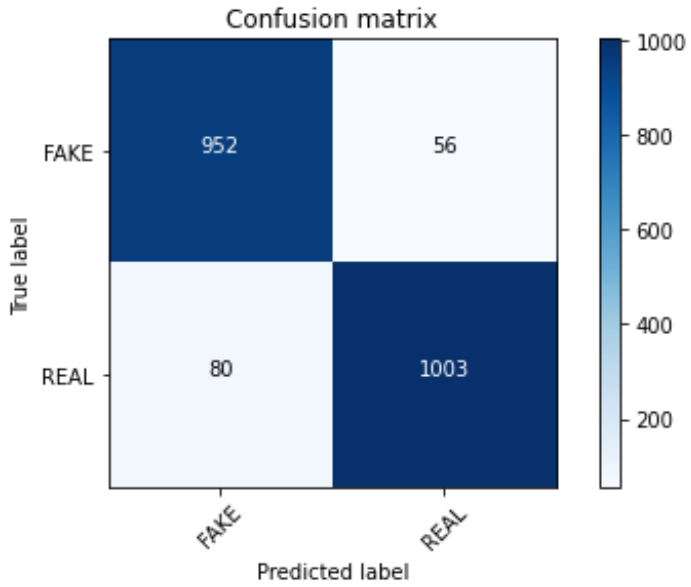
In [0]:

```
linear_clf = PassiveAggressiveClassifier(n_iter_no_change=50)
```

In [28]:

```
linear_clf.fit(tfidf_train, y_train)
pred = linear_clf.predict(tfidf_test)
score = metrics.accuracy_score(y_test, pred)
print("accuracy:   " , score)
cm = metrics.confusion_matrix(y_test, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
```

```
accuracy:    0.9349593495934959
Confusion matrix, without normalization
```



Testing MultinomialNB

In [0]:

```
clf = MultinomialNB(alpha=0.1)
```

In [0]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [31]:

```
last_score = 0
for alpha in np.arange(0,1,.1):
    nb_classifier = MultinomialNB(alpha=alpha)
    nb_classifier.fit(tfidf_train, y_train)
    pred = nb_classifier.predict(tfidf_test)
    score = metrics.accuracy_score(y_test, pred)
    if score > last_score:
        clf = nb_classifier
    print("Alpha: {:.2f} Score: {:.5f}".format(alpha, score))
```

```
Alpha: 0.00 Score: 0.88140
Alpha: 0.10 Score: 0.89766
Alpha: 0.20 Score: 0.89383
Alpha: 0.30 Score: 0.89000
Alpha: 0.40 Score: 0.88570
Alpha: 0.50 Score: 0.88427
Alpha: 0.60 Score: 0.87470
Alpha: 0.70 Score: 0.87040
Alpha: 0.80 Score: 0.86609
Alpha: 0.90 Score: 0.85892
```

Introspecting models

In [0]:

```
feature_names = tfidf_vectorizer.get_feature_names()
```

In [33]:

```
### Most real
sorted(zip(clf.coef_[0], feature_names), reverse=True)[:20]
```

Out[33]:

```
[(-6.257361214701583, 'trump'),
 (-6.494453094312678, 'said'),
 (-6.6539784739838845, 'clinton'),
 (-7.037944662867073, 'obama'),
 (-7.146539983381228, 'sanders'),
 (-7.215376008647511, 'president'),
 (-7.266562805741618, 'campaign'),
 (-7.2875931446681514, 'republican'),
 (-7.341118458599064, 'state'),
 (-7.341357110247905, 'cruz'),
 (-7.378312441985425, 'party'),
 (-7.44688067245789, 'new'),
 (-7.476288801154588, 'people'),
 (-7.547225599514773, 'percent'),
 (-7.5553074094582335, 'bush'),
 (-7.580150633909893, 'republicans'),
 (-7.585540501265243, 'house')]
```

```
 (-7.5655405012652435, 'house'),
 (-7.634478172520314, 'voters'),
 (-7.648482443695299, 'rubio'),
 (-7.6734836186463795, 'states')]
```

In [34]:

```
### Most fake
sorted(zip(clf.coef_[0], feature_names))[:20]
```

Out[34]:

```
[(-11.349866225220305, '0000'),
 (-11.349866225220305, '000035'),
 (-11.349866225220305, '0001'),
 (-11.349866225220305, '0001pt'),
 (-11.349866225220305, '000km'),
 (-11.349866225220305, '0011'),
 (-11.349866225220305, '006s'),
 (-11.349866225220305, '007'),
 (-11.349866225220305, '007s'),
 (-11.349866225220305, '008s'),
 (-11.349866225220305, '0099'),
 (-11.349866225220305, '00am'),
 (-11.349866225220305, '00p'),
 (-11.349866225220305, '00pm'),
 (-11.349866225220305, '014'),
 (-11.349866225220305, '015'),
 (-11.349866225220305, '018'),
 (-11.349866225220305, '01am'),
 (-11.349866225220305, '020'),
 (-11.349866225220305, '023')]
```

In [35]:

```
my_news = [input()]
print(shape(my_news))
my_vector = tfidf_vectorizer.transform(my_news)
pred = linear_clf.predict(my_vector)
print(pred)
```

```
 Former U.S. Ambassador to China Gary Locke is denouncing U.S. President Donald
Trump for a new campaign ad that seems to falsely imply Locke was a Chinese off
icial.   Trump’s Republican reelection campaign released an ad Thursday that ac
cused former Vice-President Joe Biden, the presumptive Democratic presidential
nominee, of being too cozy with China. It featured an image of Biden and Locke
on a stage with U.S. and Chinese flags in the background.Locke, an Asian
American, said Friday that Trump and his team are “fanning hatred” at a time wh
en hate crimes and discrimination against Asian Americans are on the rise. He s
aid in a statement that “the Trump team is making it worse” and that “Asian Ame
ricans are Americans. Period.”  Locke was an ambassador during the Obama
administration and also served as U.S. commerce secretary. He served as
governor of Washington from 1997 to 2005.
(1,)
['REAL']
```

Conclusion

Our accuracies for the models are as follows.

- **MultinomialNB - 89.33%**
- **Linear Model - 93.63%**

Thank You!