Classifying Real or Fake News Articles with Naïve Bayes Classifiers and Perceptrons in Python

CSE 415 Assignment 7 Part B

University of Washington, Seattle, Autumn 2017

Nicholas Nagy

Introduction:

Because the internet is so readily available to both passive and assertive users, it's very easy for an idea to be published and then heard and accepted by millions of people, even if the idea isn't rooted in any evidence. Fake news mimics real news to appear credible, and the intent is to manipulate its audience in a way that they won't question the reliability of who is saying it. Fake news sites take advantage of the fact that most people do not fact-check articles, and the confirmation bias, which is the trend that human beings are more likely to search for "facts" that support their ideologies than facts that refute them. Without doing additional research, it's not always easy to tell if an article is reliable or not.

Formulation:

My goal was to find a labeled dataset or article sources large enough to train over and test over, where each sample included the article text as an attribute. I was willing to test over other features, but to me text was essential. There are a lot of different attributes that may or may not help determine if a source is honest or not, but the text body itself is sure to always reveal this.

Techniques Used:

I wanted to see if a Naïve Bayes classifier or a Perceptron could classify articles as real or fake based on text alone. If they could, how much of the article would they need (for example, can they accurately predict based just on the article titles)? And which classifier was more accurate?

Training and Testing Data Used:

I was able to find multiple CSV files for potential datasets.

https://www.kaggle.com/mrisdal/fake-news - contained a very large CSV file with a wide variety of metadata (title, text, author, URL, shares, etc.) and labels (mostly true, conspiracy, bias, etc.). I ended up not using this for the basis of my classifier because the labels were not explicitly different from one another, and there was no absolute "real" news classification, but I was nonetheless impressed with the collection of data.

https://github.com/BuzzFeedNews/2016-12-fake-news-survey/blob/master/data/raw-data.csv - another CSV file I considered, but the samples did not include the text bodies.

https://github.com/GeorgeMcIntire/fake_real_news_dataset - this is the source I ended up using. The metadata is much more basic (title and text), but the data was labeled either "real" or "fake", and because of the way I wanted to approach the problem this dataset seemed most appropriate. This is the dataset that I ended up using.

There are 6335 samples.

Experiments:

I did one test on classifying based on article titles, on test based on article text, and one based on both (sending titles and text to one vector). Each test, I fit two third of the sample data to both a Multinomial Naïve Bayes classifier and a Perceptron (4244 training samples, 2091 test samples). The Perceptron was set to run a maximum of 5 epochs (weight updates) over the training data.

There were about a dozen runs of this algorithm that I used: the data below is just an example of one of these runs, but the data across all runs was fairly consistent (the normalized matrices attached to the end of this report are also from a separate run).

Results:

Training over article titles

2091 test samples	True Positives	True negatives	False positives	False negatives	F score
Naïve Bayes	884	753	314	140	0.795679568
Perceptron	771	820	247	253	0.755142018

Training over article text (bodies)

2091 test samples	True Positives	True negatives	False positives	False negatives	F score
Naïve Bayes	1019	664	386	22	0.833197056
Perceptron	950	983	67	91	0.923226433

Training over both

2091 test samples	True Positives	True negatives	False positives	False negatives	F score
Naïve Bayes	1024	632	418	17	0.824808699
Perceptron	932	987	63	109	0.915520629

Discussion:

The difficulty with classifying articles is that just because we can fit a function to calculate an accurate guess does not mean there aren't exceptions. In all above cases, both classifiers still classified some samples incorrectly. People can at least research the information elsewhere. At the end of the day, these algorithms are an *educated* guess, but still a guess, based only on the articles themselves.

There are a few things to note from the above tables. One is that the Naïve Bayes classifier has a greater F score than the Perceptron when both classifiers were looking at titles only. However, in the other two

tests the F scores for both classifiers are noticeably higher, and the Perceptron had significantly better precision than the Naïve Bayes in these tests. These facts were consistent for all runs.

Another thing to note is that training over body text, and training over titles and body text had almost identical F scores (in the above case, the F scores are greatest when title is ignored, but this was not true for every run). This information, combined with the F scores for the title test, suggest that article titles are trivial compared to the body text itself when trying to classify articles. This is not a terribly surprising fact considering the small portion of the training and test vectors the title took up – if I were to do future tests, I would try and weight the title data and text data differently when combining.

Another feature I think would be a strong determinant is the news site of the article (the home page URL). As I mentioned, I did find CSV files that included this as a feature, but their labels were not as relevant to the assignment.

Personal Retrospective:

One thing this taught me is that one classifier might work better or worse than other classifiers depending on the datasets. I assumed that a Perceptron is a far superior algorithm to Naïve Bayes, but Naïve Bayes better predicted my title data in all my tests. Nevertheless, my data shows me the impressive accuracy of the Perceptron's fit after very few epochs.

References:

5.6.2. The 20 newsgroups text dataset¶. (n.d.). Retrieved December 06, 2017, from http://scikit-learn.org/stable/datasets/twenty_newsgroups.html

Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures. (2016, November 11). Retrieved December 06, 2017, from http://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/

Confusion matrix¶. (n.d.). Retrieved December 06, 2017, from http://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html#sphx-glr-auto-examples-model-selection-plot-confusion-matrix-py

G. (2017, February 24). GeorgeMcIntire/fake_real_news_dataset. Retrieved December 06, 2017, from https://github.com/GeorgeMcIntire/fake_real_news_dataset

Sklearn.linear_model.Perceptron¶. (n.d.). Retrieved December 06, 2017, from http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Perceptron.html

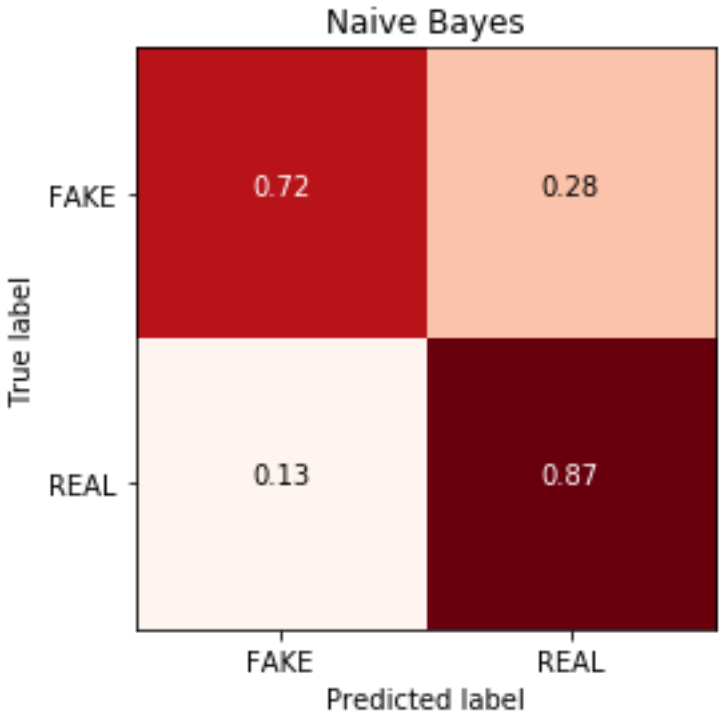
Appendix A

```
# Nick Nagy
# CSE 415 Assignment 7 Part B
# Classifying articles as REAL or FAKE using a Naive Bayes classifier
# versus a Perceptron

import numpy as np
import pandas as pd
import zipfile
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import Perceptron
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import itertools

# plots a (normalized) confusion matrix
def plot(cm, classifier):
    plt.figure()
    if classifier == 'Naive Bayes':
        colormap = plt.cm.Reds
    elif classifier == 'Perceptron':
        colormap = plt.cm.Blues
    plt.imshow(cm, interpolation = 'nearest', cmap = colormap)
    cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
    ticks = np.arange(2)
    plt.xticks(ticks, ['FAKE','REAL'])
    plt.yticks(ticks, ['FAKE','REAL'])
    plt.title(classifier)
    for i,j in itertools.product(range(cm.shape[0]),range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], '.2f'), horizontalalignment = 'cen
ter',
                 color = 'white' if cm[i,j] > cm.max()/2 else 'black')
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.show()

# computes and plots two confusion matrices from the data and y using the
two classifiers
def compute(train_set, test_set, y_train, y_test, vectorizer, title):
    train_vectors = vectorizer.fit_transform(train_set)
    test_vectors = vectorizer.transform(test_set)
    # Naive Bayes
    nb_classifier.fit(train_vectors, y_train)
    nb_y_pred = nb_classifier.predict(test_vectors)
    nb_cm = confusion_matrix(y_test, nb_y_pred)
    # Perceptron
    p.fit(train_vectors, y_train)
    p_y_pred = p.predict(test_vectors)
    p_cm = confusion_matrix(y_test, p_y_pred)
    print(title)
    plot(nb_cm, 'Naive Bayes')
```

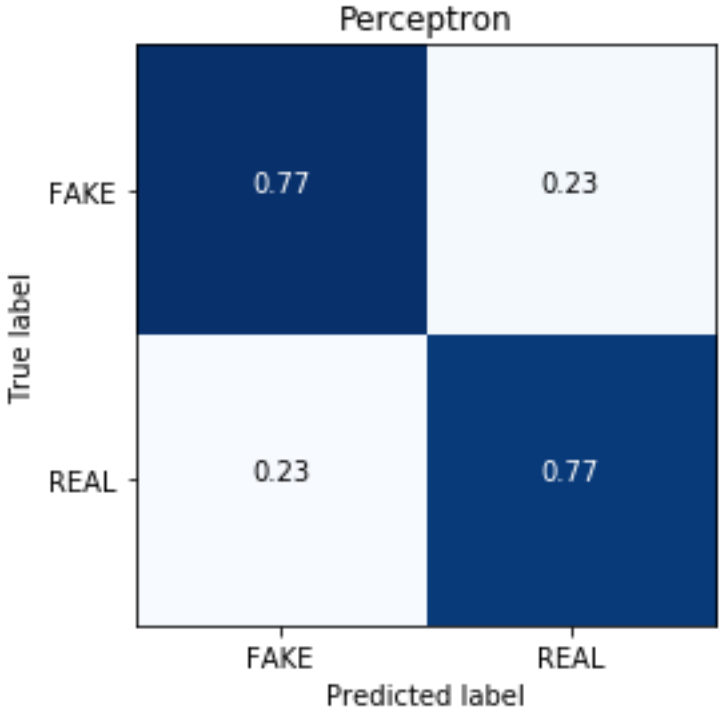
```
    nb_f1 = f(nb_cm)
    print("F1 Score: " + str(nb_f1))
    plot(p_cm, 'Perceptron')
    p_f1 = f(p_cm)
    print("F1 Score: " + str(p_f1))
    print()

# computes and returns F1 score from a confusion matrix
def f(cm):
    precision = cm[1,1] / (cm[1,1] + cm[1,0])
    recall = cm[1,1] / (cm[1,1] + cm[0,1])
    return 2*(precision*recall)/(precision + recall)

##########################################################################
####################

# prepare data

zf = zipfile.ZipFile('fake_or_real_news.csv.zip')
df = pd.read_csv(zf.open('fake_or_real_news.csv'))

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

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

tfvectorizer = TfidfVectorizer(max_df = 0.7)

nb_classifier = MultinomialNB()
p = Perceptron(max_iter = 5, tol = None)

##########################################################################
###################

# train over article titles

df_title_train, df_title_test, y_train, y_test = train_test_split(df.title
, y, test_size = 0.33)
compute(df_title_train, df_title_test, y_train, y_test, tfvectorizer,
        "Confusion matrix classifying news articles as real or fake based
on title:")

# train over article texts

df_text_train, df_text_test, y_train, y_test = train_test_split(df.text, y
, test_size = 0.33)
compute(df_text_train, df_text_test, y_train, y_test, tfvectorizer,
        "Confusion matrix classifying news articles as real or fake based
on text:")

# train over both

df_train = np.add(df_title_train, df_text_train)
df_test = np.add(df_title_test, df_text_test)
```

```
compute(df_train, df_test, y_train, y_test, tfvectorizer,
        "Confusion matrix classifying news articles as real or fake based
on title and text:")
```

```
Confusion matrix classifying news articles as real or fake based on title:
```

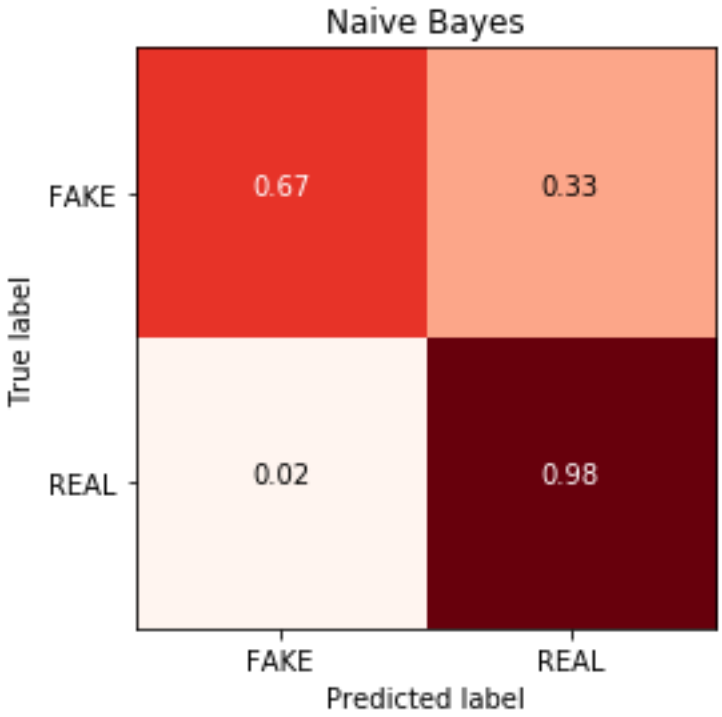



```
F1 Score: 0.807881773399
```

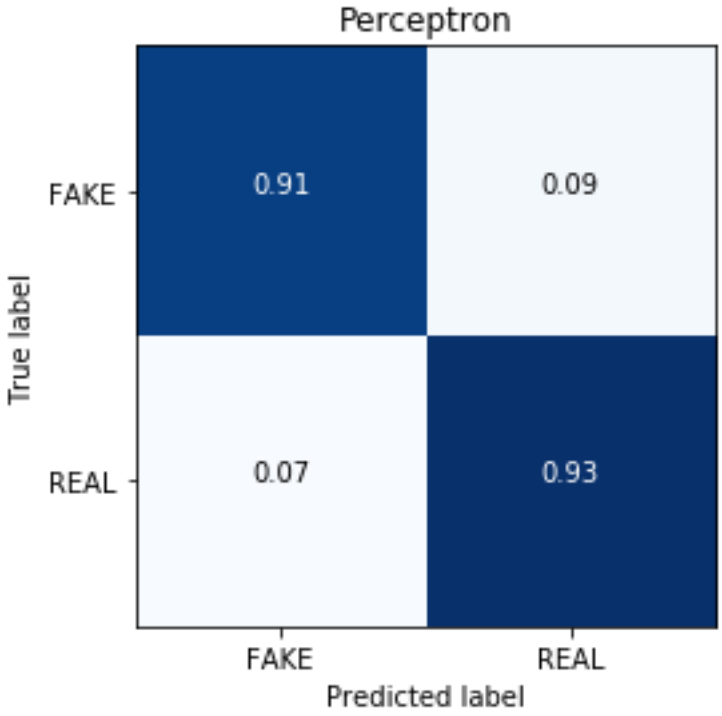


F1 Score: 0.770420492992

Confusion matrix classifying news articles as real or fake based on text:

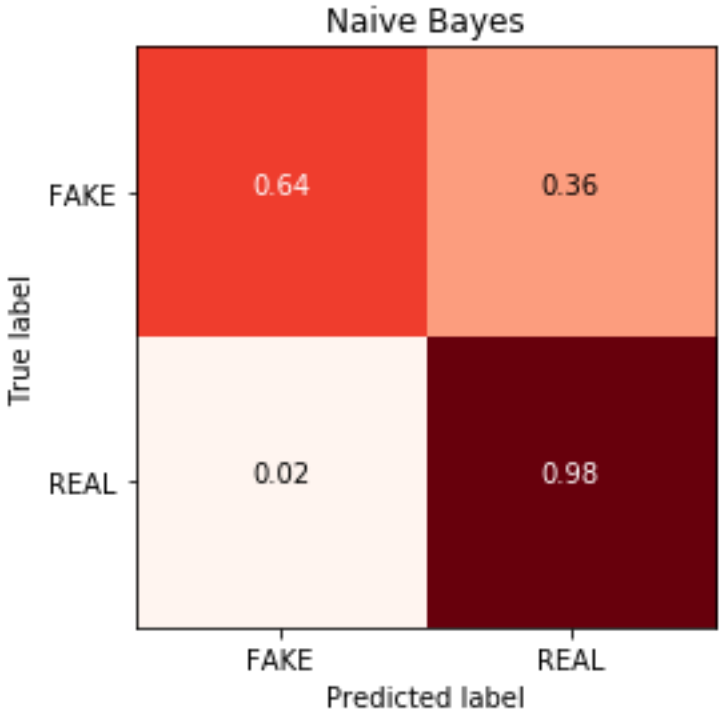


F1 Score: 0.851307189542

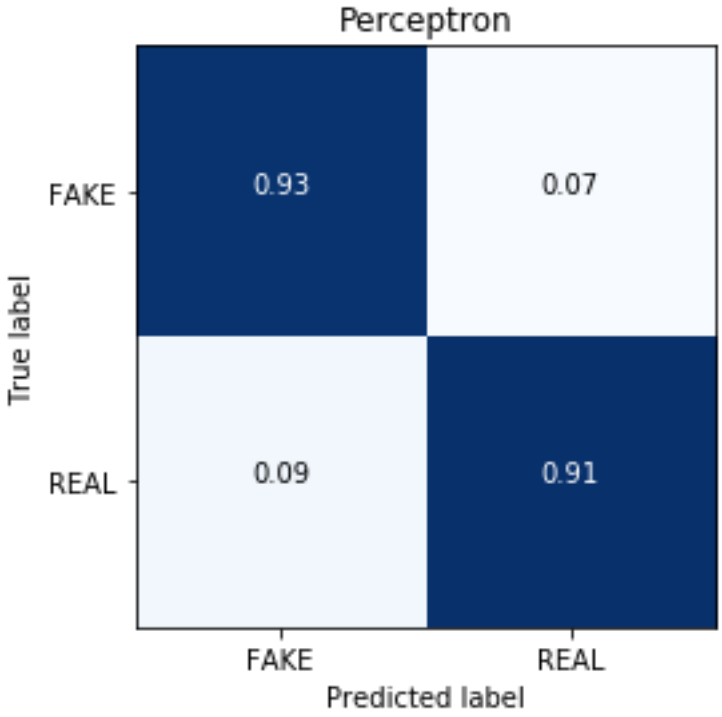


F1 Score: 0.924510717614

Confusion matrix classifying news articles as real or fake based on title and text:



F1 Score: 0.841000807103



F1 Score: 0.920589633856