Fake News Classifier:

Pinky Chauhan
University of Illinois at Urbana Champaign

Overview:

- This objective of this project is to build a classifier system based on machine learning that is able to identify fake news from real/reliable news given a news title and/or news text content as the input. Such a tool can be integrated with social media platforms to flag potentially fake articles or filter those out.
- This is essentially a data categorization problem where I have trained several classifier models on the following dataset from Kaggle: https://www.kaggle.com/c/fake-news/data
- The dataset comprises of csv format files for training and testing with each file containing id, news title, text and author fields.
- The train.csv also has label field to categorize data as Reliable (label value 0) and Fake (label value 1)
- After evaluation based on various performance metrics, one of the models (in this case, Linear SVC over unigram bag-of-words/TF-IDF representation) is integrated in the final tester notebook to test with news data.
- The classifier takes a news article (title and text) as input and provides a prediction for the news article as either of the 2 categories:
  - Fake News
  - Reliable News

Software Implementation Details:

- Data Analysis:
  - Comprises of checking several attributes to evaluate their contribution towards classification. For such a classifier, the text and title of the news make obvious choices as features.
  - I also analyzed authors' distribution using pandas and polarity/sentiment differences using NLTK vader sentiment intensity analyzer library on the dataset.

- Preprocessing of data:
  - Handling missing values by removing any rows with no text and title, preprocess data to remove any punctuations, remove any words with length 3 or less, stop words removal, tokenization and lemmatization using NLTK libraries

- Feature selection:
  - Concatenated news title and text into article field and preprocessed it. Article comprises the feature to train the model

- Vectorization

- Different vector forms listed below have been used using NLTK vectorization/transformation libraries:
    - Term frequency (TF) based vector over unigrams bag of words representation
    - Term frequency/inverse document frequency (TF-IDF) based vector over unigrams
    - Term frequency (TF) based vector over unigrams and bigrams
    - Term frequency/inverse document frequency (TF-IDF) based vector over unigrams and bigrams
    - Term frequency/inverse document frequency (TF-IDF) based vector over unigrams, bigrams and trigrams

- Training/hyperparameter tuning/validation using classification models:
    - Models used (sklearn libraries):
        - Naïve bayes (With/without smoothing, TF vs TF-IDF vectors, Unigram/N-gram)
        - Logistic Regression (TF-IDF vectors using Unigrams/N-grams)
        - SVM using Linear SVC (TF-IDF vectors using Unigrams/N-grams, Regularization)
        - SGDC classifier (TF-IDF vectors using Unigrams/N-grams)
        - Decision Tree (TF-IDF vectors using Unigrams/N-grams)

- Performance evaluation:
    - Compute and analyze metrics using sklearn metrics libraries
        - Precision (macro/micro), recall (macro/micro), F1 (macro/micro)
        - Classification Accuracy
        - Confusion matrix to see distribution of true/false positives/negatives
    - Select the best performing model based on evaluation results (SVM using Linear SVC using TF-IDF vector over unigrams)
    - Results:

	accuracy	precision(macro)	precision(micro)	recall(macro)	recall(micro)	f1_score(macro)	f1_score(micro)
Decision Tree (TFIDF/Uni-bi-trigram)	95.92%	0.959	0.959	0.959	0.959	0.959	0.959
Decision Tree (TFIDF/Uni-bigram)	95.60%	0.956	0.956	0.956	0.956	0.956	0.956
Decision Tree (TFIDF/Unigram)	92.81%	0.928	0.928	0.928	0.928	0.928	0.928
Linear SVC (TFIDF/Uni-bi-trigram)	95.87%	0.959	0.959	0.959	0.959	0.959	0.959
Linear SVC (TFIDF/Uni-bigram)	96.04%	0.96	0.96	0.96	0.96	0.96	0.96
Linear SVC (TFIDF/Unigram)	96.04%	0.96	0.96	0.96	0.96	0.96	0.96
Linear SVC (TFIDF/Unigram/Regularization)	87.27%	0.873	0.873	0.873	0.873	0.873	0.873
Logistic Regression (TFIDF/Uni-bi-trigram)	93.73%	0.937	0.937	0.937	0.937	0.937	0.937
Logistic Regression (TFIDF/Uni-bigram)	93.62%	0.936	0.936	0.936	0.936	0.936	0.936
Logistic Regression (TFIDF/Unigram)	94.27%	0.943	0.943	0.943	0.943	0.943	0.943
Multinomial naive bayes (TF/Uni-bigram/Smoothing)	92.44%	0.931	0.924	0.923	0.924	0.924	0.924
Multinomial naive bayes (TF/Unigram/NoSmoothing)	92.06%	0.923	0.921	0.92	0.921	0.92	0.921
Multinomial naive bayes (TF/Unigram/Smoothing)	89.75%	0.907	0.897	0.897	0.897	0.897	0.897
Multinomial naive bayes (TFIDF/Uni-bi-trigram/Smoothing)	75.46%	0.831	0.755	0.761	0.755	0.742	0.755
Multinomial naive bayes (TFIDF/Uni-bigram/Smoothing)	80.94%	0.861	0.809	0.808	0.809	0.802	0.809
Multinomial naive bayes (TFIDF/Unigram/NoSmoothing)	91.83%	0.921	0.918	0.918	0.918	0.918	0.918
Multinomial naive bayes (TFIDF/Unigram/Smoothing)	81.98%	0.865	0.82	0.818	0.82	0.813	0.82
SGDC (TFIDF/Uni-bi-trigram)	95.46%	0.955	0.955	0.954	0.955	0.955	0.955
SGDC (TFIDF/Uni-bigram)	95.65%	0.957	0.957	0.957	0.957	0.957	0.957
SGDC (TFIDF/Unigram)	95.56%	0.956	0.956	0.956	0.956	0.956	0.956

- Save/export trained model:
    - Using pipeline to specify all steps (vectorizer/classifier), fit training data and exporting model using joblib library

- Kaggle submission:
  - Predicted results for data in test.csv and submitted notebook/results to Kaggle (https://www.kaggle.com/pinkychauhan/fakenewsclassifierusingnltk-sklearn)
  - Accuracy: 94%

- Create script (Jupyter notebook) that will take news text as input and generate classification as reliable news or fake news.

Installation/Execution Details:
Code is written using Jupyter notebook and python 3

Code structure:

- data: This directory contains the dataset from Kaggle (https://www.kaggle.com/c/fake-news/data). There are 3 files:
  - train.csv: To use for analysis, training, validation
  - test.csv: Test dataset for submission of results to Kaggle competition
  - submit.csv: File containing results/predictions for data in test.csv
- notebooks: This directory contains 2 notebooks:
  - FakeNewsClassifierTraining.ipynb: Jupyter notebook containing code/results for data analysis, cleanup, features set up, vectorization, training using various classifier algorithms, tuning and performance evaluation/comparison, model pipeline creation/export, prediction of results for test.csv for Kaggle submission
  - Tester.ipynb: This notebook loads the pretrained/exported model and predicts the category for a given news article. Use this notebook to test the classifier.
- model: This directory contains the pretrained model exported by FakeNewsClassifierTraining.ipynb notebook and loaded by Tester.ipynb
- results: This directory contains the summarized performance metrics from different models used for training and a copy of the submit.csv file generated from predictions for data in data/test.csv

Code Setup:

- Install python 3 and Jupyter notebook
- Install the following python/machine learning libraries:
  - re: For regular expression matching
  - itertools: To iterate over data
  - pandas: For Data analysis/representation as Dataframes
  - nltk: Natural language toolkit
  - sklearn: For model selection, training, evaluation, export using pipeline
  - matplotlib: For visualization
  - joblib: For model export and load
- Checkout the project from main branch in Github
- Launch Jupyter notebook and navigate to the directory where project is checked out

- Tester.ipynb located in notebooks folder can be used for testing the classifier by providing values for title and text
- FakeNewsClassificationTraining.ipynb can also be executed to see all stages entailed in bulding the classifier and training/evaluation of different models

Note: In case you see an issue around missing packages stopwords, punkt, vader_lexicon or wordnet, download them one time using below commands:
nltk.download('vader_lexicon')
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

References:

- https://www.kaggle.com/c/fake-news/data
- https://scikit-learn.org/stable/user_guide.html
- https://medium.com/datadriveninvestor/python-data-science-getting-started-tutorial-nltk-2d8842fedfdd
- https://matplotlib.org/tutorials/introductory/pyplot.html
- https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623