FAKE NEWS

Project Done By: Anthony Chan, Wei Chen, Steven Chou

Every project starts with an idea. We were inspired to do this project on fake news after one of our teammates encountered an article published on NYC data science, where a few students did a project classifying fake news.

First, let's define fake news. Fake news is a type of propaganda (news with an agenda) that contains purposeful misinformation. In other words, it's false information that is engineered to achieve a goal, usually not a good one (take the 2016 US presidential election). Now, with social media, the spread of fake news has never been easier. (Side note: Buzzfeed did an analysis and people actually engaged with fake news stories more on Facebook than real ones. Why? Because fake news are engineered to be attention-grabbing. Explosive). Our project aims to classify news articles as Fake or Real. Fake meaning it contains false information, and real meaning it doesn't.

To do so, we need one thing: data. Our data would be in the form of news articles which we obtained from two places. Our fake news is a dataset of news articles collected during November 2016 from a news aggregation site called webhose.io. To get the real news from that same site, we had to pay money. As broke college students with creative minds, we instead aggregated our own news articles from a kaggle dataset called "All the news" (which had 143K news articles from 15 American publications collected over a span of a few years starting in 2015). We obtained only the news articles from the same time frame, November 2016 (with a margin of error of one month), and from diverse but reputable sources. Now, with two datasets about the same size (one with fake news and one with real news), we started the next step: data pre-processing.

To pre-process our data, we used Pandas and modified the data frame until we obtained the features we wanted (title, content, publication, label). Because the data came with feature spaces of different sizes and also different feature names, we had to do some dataframe magic to be able to successfully combine both datasets into one. With one dataset with all our news (over 28K articles and 4 features), we proceeded to the next stage of pre-processing: the text of our articles.

In order to feed our data into ML models, they need to be in an input form that can be understood by the models. Models like vectors with numbers. We proceeded to transform each article (each row) in our dataset into a word vector. Since there were many ways to do so, we decided on two ways to allow us to try for better results. We did TF-IDF and Word2Vec.

Word2Vec works by converting text to features while maintaining their original relationships between words in a body of text. To get the best results, we used pre-trained vectors trained on the Google News dataset (with about 100 billion words). With their model with 300-dimensional vectors for 3 million words and phrases, we were able to generate vectors with 300 features for each word in our article. Then, we took the average of them and that became the article vector we will use to represent each article. We then appended the fake/real label to each of the vectors. In the end, we have vectors of size 301 (300 numbers + one label).

TF- IDF works by looking through each word in the document, each unique word gets a weight based on its frequency in the document and in all documents. The problem with TF - IDF is that every unique word is now a feature, so the feature space got as big as 50,000, more than the number of samples we have. To overcome this challenge and come up with best feature space, we plotted the number of features vs. classifying accuracy score, where the set of features number was generated though exponential function, in range 2 to the sample size. And the classifying accuracy score was calculated using logistic regression. The performance increased as feature size increase, then flat out starting at 441, so we picked 441 to be the optimal feature size.

With the word vectors created, we now proceeded with ML models. We chose to use popular classifiers such as Logistic Regression and Random Forest . because they have shown giving good performance and have libraries available in python. But we ended up using 7 classifiers because of the availability. The results of our model give up to 85% accuracy and 87% ROC AUC, in particular, Logistic Regression, Gradient Descent, and Gradient Boosting perform the best, while Random Forest and Decision Tree perform very good on training data, but much poorer on testing data.

For next steps, we can always try various other models and explore what makes one model a better predictor than other. Furthermore, it’d be interesting if we can get hold of an even larger dataset of fake and real news from a bigger timeframe, though it’s difficult because good data is hard to find. There are always ways to better optimize each step our entire process from beginning to end. Overall, this project gave us a great introduction with machine learning using real data and allowed us to explore various aspects of data science including Python libraries, NLP, data pre-processing, and classification.