**Project Proposal:**

As great as social media and the Internet is today as a means to get information fast to the public, the threat of *disinformation* is very real and is only getting worse. The problem of identifying disinformation/misinformation (the two will be used interchangeably throughout this paper) is one that this project intends to solve. My clients for this project is any website that deals with text data, especially those websites that share news articles such as Twitter and Facebook. Being able to train a machine learning or deep learning model to identify which news is fake or not is important to maintaining the integrity of those platforms. With enough and the right kind of data, this model could also even perhaps generalize to websites that contain user comments (e.g., Reddit, Instagram) or user reviews (e.g., Amazon, Goodreads). Thus, identifying which text data is actually written by a person as opposed to a computerized/Russian bot or factual versus satirical (or straight-up false) is of paramount importance to saving ourselves from entering a post-truth world.

The data that I am using is a public data set that is available on Kaggle. It consists of two CSV files: “truth” and “fake.” Both data files contain columns dedicated to the text of a news article. However, the difference between the two files is that one column is from an actual news source while the other article is entirely made up. The other columns in the data sets are the title of the article, the subject of the article, and the date of the article.

My approach to this project’s problem is to pre-process the text data for both the titles and the article text. I will keep the title, subject, and article text as features, with the main feature being the text of the article. I will keep the previously mentioned

features for the model, with the main predictor variable being the text of the article. Since I want the model to analyze one data set only, I will have to combine the two datasets “truth” and “fake” into one data file. Before the merge, I will create a single column in each: both named “real/fake.” That column in the file *“truth*” will contain all values of “0,” and that column in the file *“fake*” contain all values of “1.” In this way, after I combine them, all of the rows that contain real news articles will have a label of “0” and all the rows that contain fake news articles will have a label of “1.” Then I’ll randomize the observations so that the rows are a mixture of “real” and “fake” news.

As far as modeling is concerned, I will test out four competing models. Two of which are machine learning and the other two are deep learning: logistic regression, random forest, basic deep learning neural networks, and convolutional neural networks. I can use the Jupyter Notebook software for the ML models, but I will require the usage of Google Collab for the deep learning models so that I can use Tensor Flow, as I have found that Tensor Flow and the Windows operating system do not get along.

I will present the results of my project on my Github. There we will find the dataset (or link if it’s too big), slide deck presentation, and code.

**Data Wrangling:**

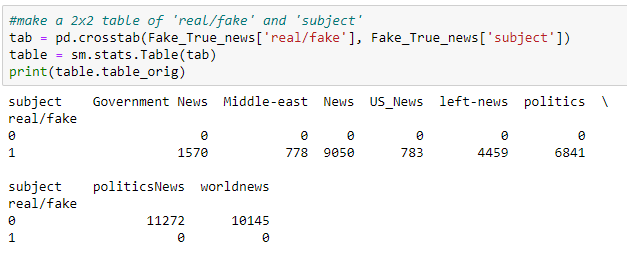
The cleaning steps that I need to perform are pre-processing the article text data. To do this, I will use the packages “spacy,” “NLTK,” and “contractions,” the latter of which comes from DJ Sarkar’s Github repository: <https://github.com/dipanjanS>. For pre-processing, I will use a catch-all function that will remove stopwords, remove

accented characters, lower case words (except those that occur in the middle of a sentence), expand contractions, remove extra white space, and remove special characters. “Stopwords” are commonly used words in a text, such as “the,” “a,” “an,” and “in.” Removing accented characters, removing extra white space, and making words lower case is self-explanatory. Expanding contractions means expanding a word such as “isn’t” into the two words that make it: “is not.” Removing special characters includes removing characters such as brackets and exclamation points. I did make a slight alteration by adding a snippet to the “lower case” function. Instead of lower-casing all words, if a word appeared mid-sentence in all capitalized letters (CAPS), then I kept that word in all CAPS. This is because when eyeballing the text data, it was clear that the “fake news” dataset contained words that were in all CAPS for emphasis. Seeing that this could be an important predictor, I think it’s important to keep these capitalized within the text.

There were fortunately no missing values within the data set.

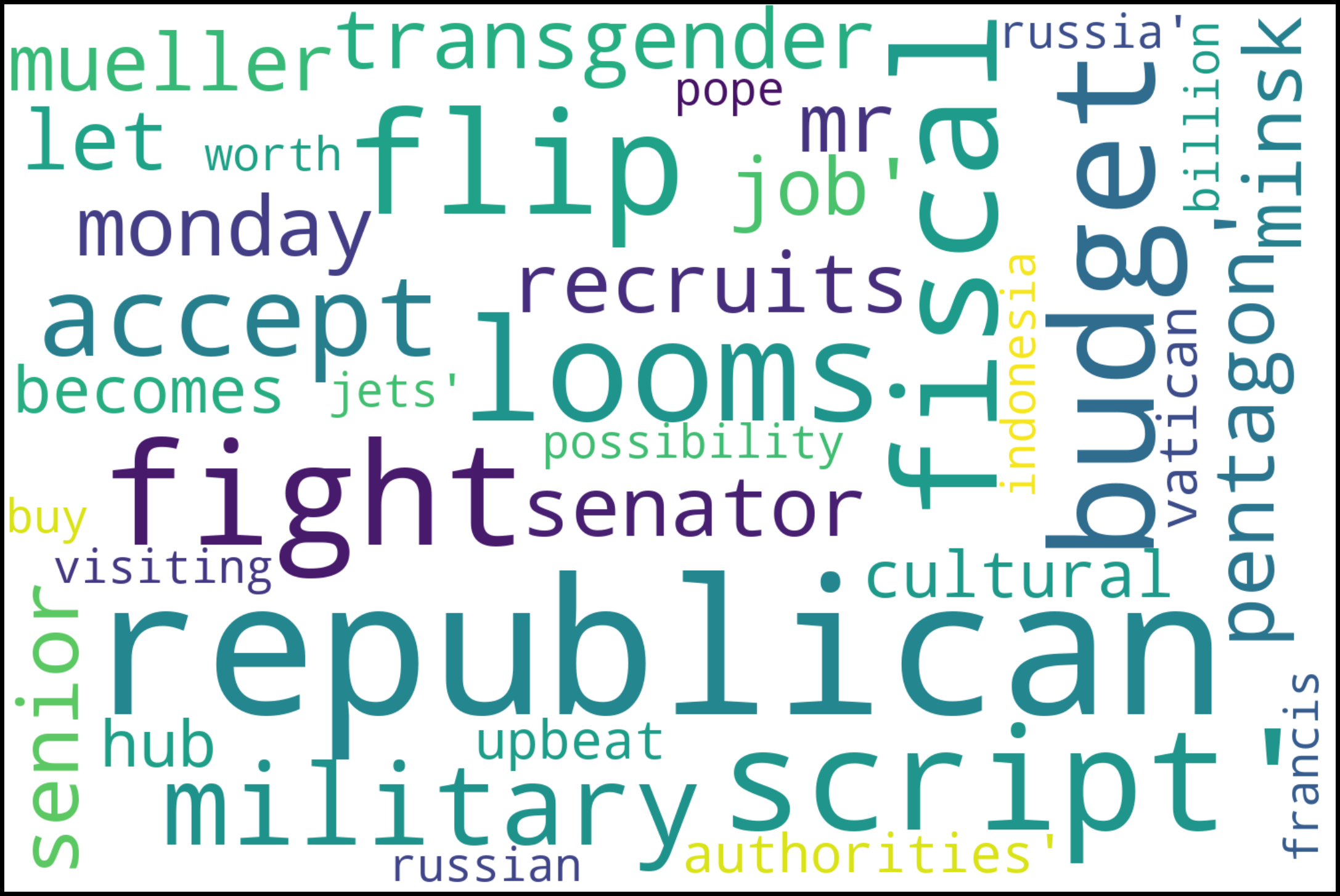
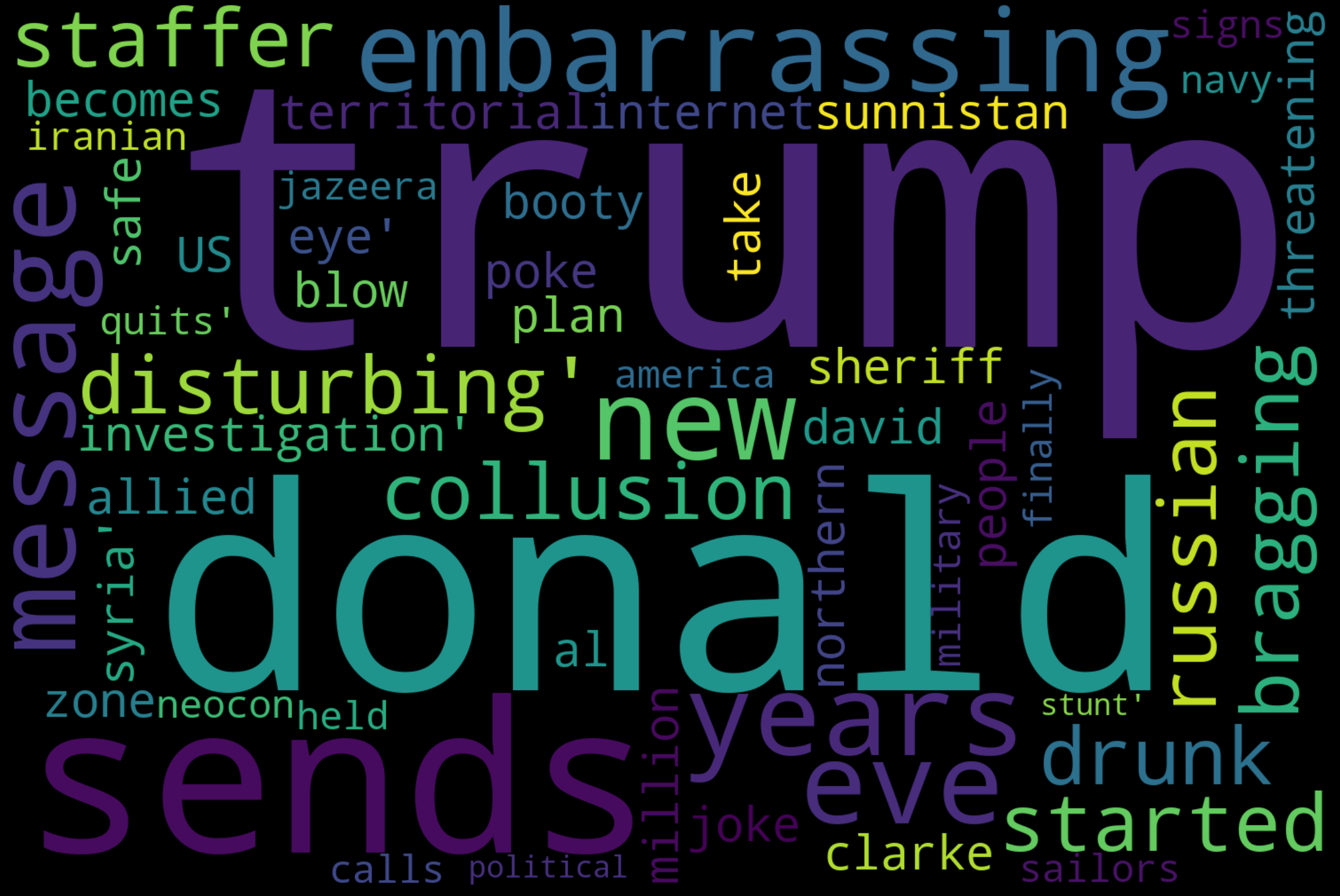
**Exploration**:

I proceeded to explore the dataset. I created the following contingency table on the “subject” variable and the “real/fake” variable:



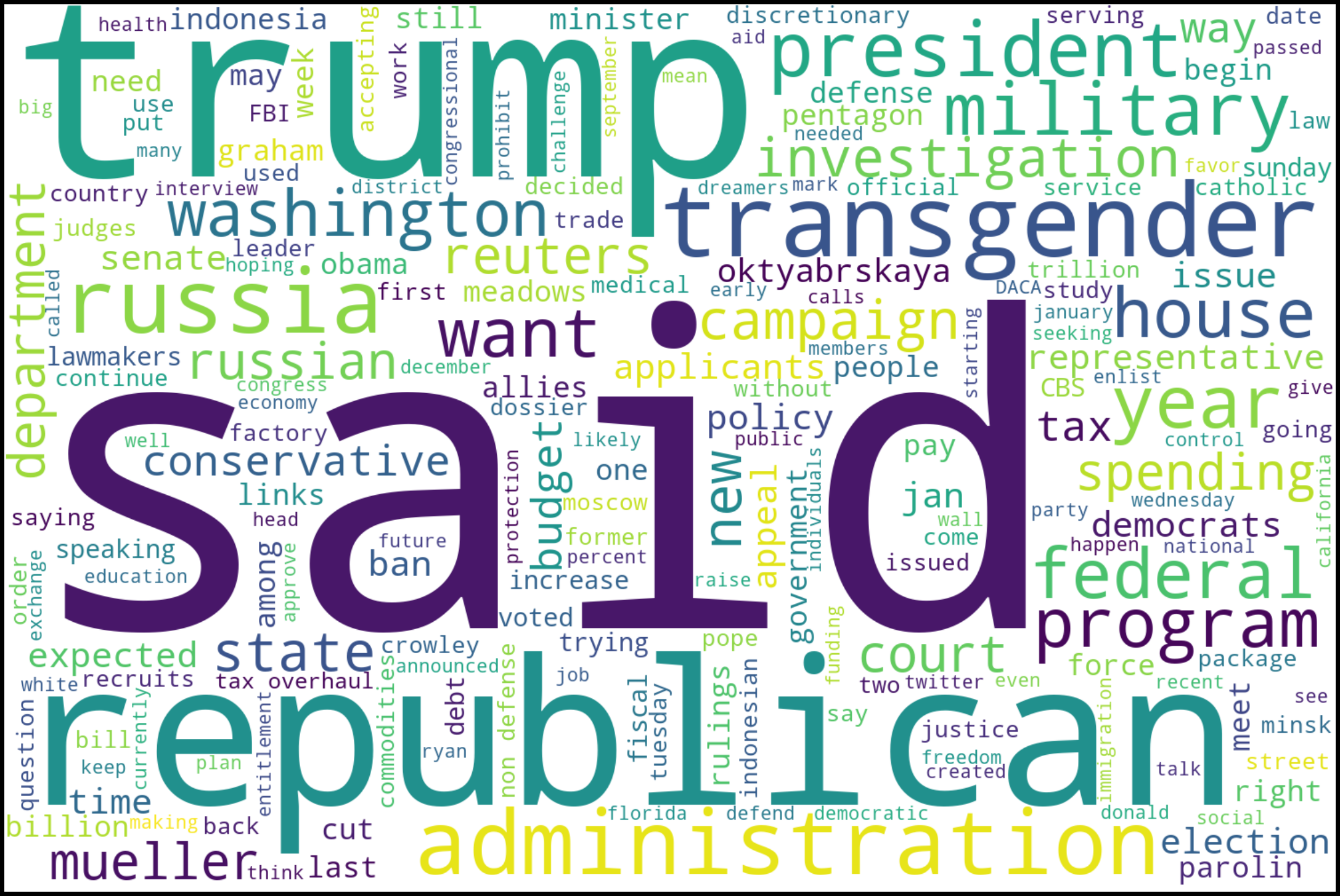
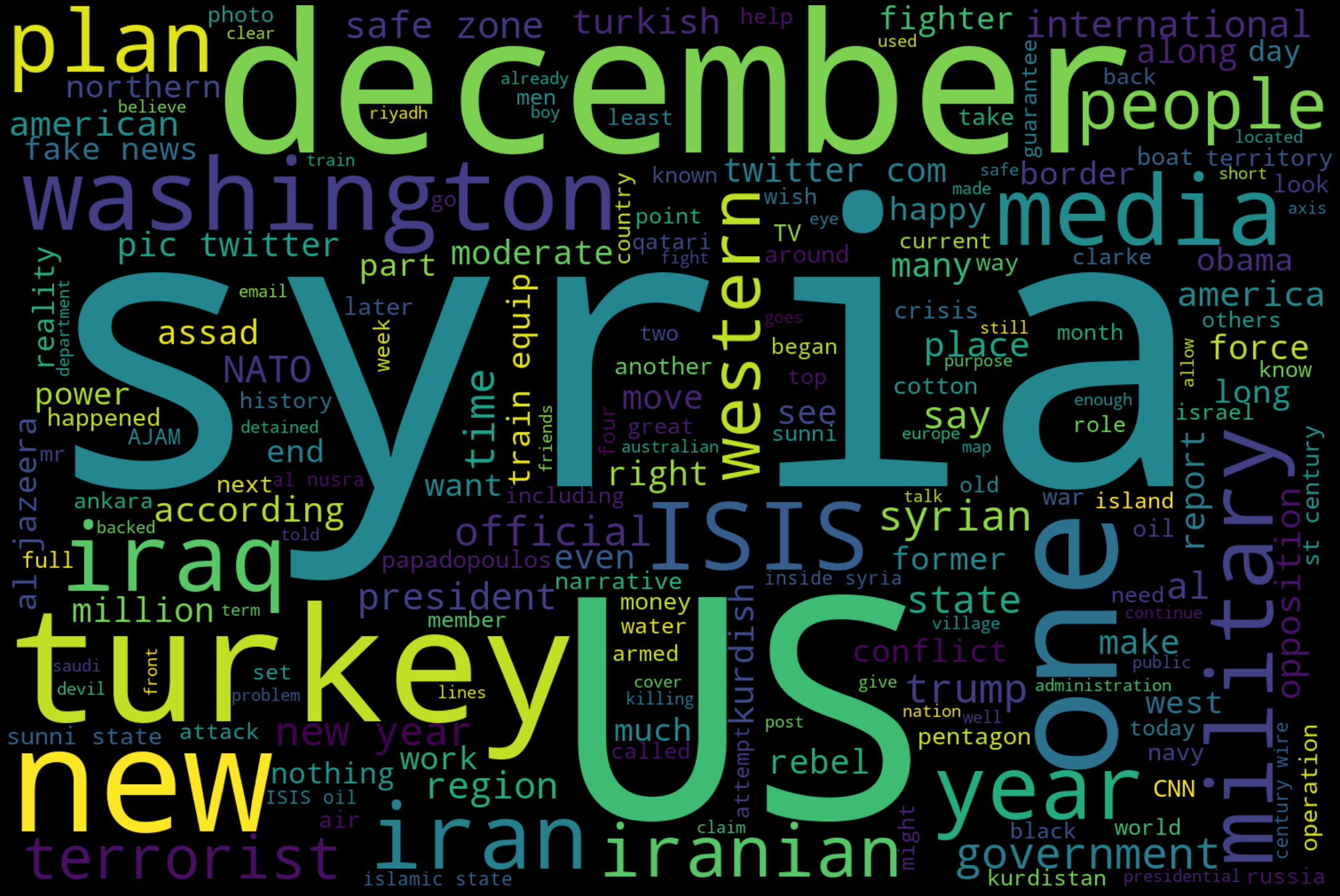
Certain subjects were only within the “real news” dataset and other subjects were within the “fake news” dataset. This means that it is not a good predictor and will not be included in the model.

Next, I made some visual images of the words within the ‘title’ and ‘text’ columns (now called ‘title\_nlp’ and ‘text\_nlp’ after pre-processing) using a Word Cloud. The following Word Cloud is from the ‘title\_nlp’ columns. The top is from the “fake news” data and the bottom is from the “real news” data:



The fake news data talked much more about Donald Trump in the headlines, while the real news had a much more varied word collage, with top words being “republican,” “fight,” and “budget.”

The next two word clouds are from the actual pre-processed text articles themselves. Once again, the top is the “fake” news and the bottom is the “real” news:



The top 3 words in the “real” news dataset appear to be “Trump,” “said,” and “Republican.” The “fake” news appears to have “Syria,” “US,” and “Turkey” has the most common words. So, there is a clear difference between the two.

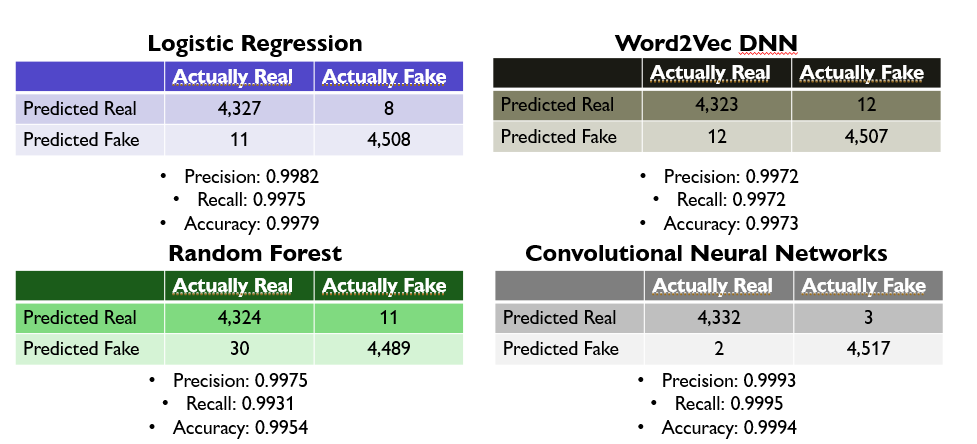
**Modeling:**

The purpose of this project was to determine whether I could accurately classify whether a news article was “real” or “fake.” Since I am dealing with text data here, I can use Natural Language Processing algorithms for this classification. I tested out four different models: 1) Logistic Regression, 2) Random Forest, 3) Deep Learning Neural Networks, and 4) Convolutional Neural Networks. The first two are machine learning (ML) models while the latter two are deep learning (DL) models. To evaluate the effectiveness of each model, I computed a confusion matrix. I’m able to see the counts of how many observations ended up being a true positive, false positive, true negative, or false negative. Through this confusion matrix, I can calculate the accuracy, precision, and recall rates of the models.

To begin, I first split the dataset into a training and test split. I used an 80:20 split, chosen arbitrarily. Then, I analyzed the text data using a “Bag of Words” modeling technique for the ML models (the DL models don’t use this technique). The concept behind this is that the algorithm grabs all of the words in the text and places them in a “bag” so that all we get is a numeric representation of the word, throwing out the word sequence and grammar rules. To get this numeric value, there are two main methods, which are the following algorithms *CountVectorizer* and *TF-IDF vectorizer*. The former obtains the counts of how many times each word occurs within the article’s text, assigning a numeric vector to each word of that count. The latter does the same but adds more weight to words that occur in a single document of text but not across all documents. In this project, our “document” is the text of the article. Instead of choosing

one or the other, I used both and reported the results for the Logistic Regression and Random Forest Models that had the best metrics.

After running the models on the dataset and evaluating each using the confusion matrix, I got the following results:



A quick glance at all of them shows that no matter which model one picks, *all* of them were very accurate: over 99% for each of them. For the Logistic Regression and Random Forest models, both of these confusion matrices came from the output generated by the *CountVectorizer* algorithm. I was surprised by how accurate they were compared to the DL models, especially the Logistic Regression model. That statistical model outperformed the ML model Random Forest *and* the DL model Word2Vec. However, the king was clearly the CNN model. For precision, recall, and accuracy, the rates were over 99.9% correct. It makes me want to look up the articles in which it got wrong. For a model to be that correct is astounding.

In conclusion, the CNN model was the best, as I expected it would be. And not only was it the best, it was amazingly accurate, precise, and sensitive (recall). Thus, we recommend the CNN model as a great starting point in identifying real and fake news articles. For future research, it’d be helpful to have a dataset that can also pinpoint the source of these news articles. Even though I got this dataset from Kaggle, which is a respected data source, I don’t know where exactly the articles came from. I’d like to know who wrote the real and fake news articles.