IST 664: Natural Language Processing
Final Project

Fake News Classification

Punami Chowdary: Visualizations, PoS features, Deep Learning
Anneka Herre: Bag of Words N-grams, Sentiment, Quantitative Linguistic Features
Dhruv Prashant Shah: Visualizations, TF-IDF Vectorization

Link to Full Results Summary in Appendix

Results Highlights

Features	Classifier	Prec	Recall	F1	Spec.	Accuracy
unigrams + bigrams + POS	Naive Bayes (NLTK)	0.85	0.96	0.90	0.82	89%
title unigrams + quant. linguistic	Logistic Regression	0.91	0.90	0.91	0.91	91%
TF-IDF + quant. linguistic	Random Forest	0.939	0.951	0.945	0.936	94.4%
BERT	Logistic Regression	0.90	0.91	0.91	0.82	91%

Description of Dataset

Source: https://www.kaggle.com/datasets/saurabhshahane/fake-news-classification

Research Paper: https://ieeexplore.ieee.org/document/9395133[1]

[1] P. K. Verma, P. Agrawal, I. Amorim and R. Prodan, "WELFake: Word Embedding Over Linguistic Features for Fake News Detection," in IEEE Transactions on Computational Social Systems, vol. 8, no. 4, pp. 881-893, Aug. 2021, doi: 10.1109/TCSS.2021.3068519.

Dataset & Featureset CSVs:
https://drive.google.com/drive/folders/1k70WbKghlwaTiIhZI44F4DKcFFaTefQr?usp=drive_link

Notebooks:
https://drive.google.com/drive/folders/1zqWDNhXwBE2cLbcOn7C23sPRlr8YmZYo?usp=sharing

For our project, we selected a dataset of labeled news items classified as **"fake" (1)** or **"real" (0)** created by a team of researchers who were developing their own model for fake news detection with a focus on social media postings. The dataset consists of **72,134** news articles with 35,028 real and 37,106 fake news items. These were structured as 'title' (headline), 'text' (article body), and 'label' columns.

This dataset is larger than past labeled fake news datasets referenced in the literature and merges four popular sets—**Kaggle, McIntire, Reuters, and BuzzFeed Political**—to create a more generic set and prevent over-fitting.

Limitations of the Dataset
The researchers acknowledge some of the limitations of the dataset sources in their paper (emphasis added):

- **BuzzFeed News** is a *very small* data set of 101 news articles
- **Reuters** consists of real and fake news articles from a *single source that increases the chances of biased data*
- **McIntire** consists of real and fake news categories *without authentic confirmation* from any individual.
- **Kaggle** consists of real and fake news data *without source information*

In working with the dataset closely, we came up against some of these limitations. Some of these could be filtered out in our pre-processing (see Methods), but we also discovered some questionable classifications that were built into the dataset. For instance, the unigram "Breitbart" is strongly associated with "real" news, indicating that items from this news service are being classified as real news, despite the fact that Breitbart's content has been described as "misogynistic, xenophobic, and racist by academics and journalists" and is known for its regular publication of "conspiracy theories and intentionally misleading stories" according to Wikipedia.

Similarly, the unigram "Onion", as well as the n-grams that make up *The Onion*'s tagline, "America's Finest News Source", are strongly associated with the fake news label. But this classification is dubious, because *The Onion* is a *satirical* news source. Satire would not normally be defined as "fake news" —fake news is functionally defined by a deceptive intent to be received by readers as "real" news, whereas satire depends on the readers' understanding that the content is not "real" to perform its function of humorous commentary.

Beyond this, we observed biases in our models that point to the effects of these limitations (e.g. toward the classification of items as “real news”), but, because we do not have a specific application in mind, we focused on general reliability of the models, rather than minimizing particular outcomes, such as false negatives. Of course, there are certainly use cases in which minimizing false negatives at the cost of more false positives might be a legitimate aim of the model—e.g. use of the model on a social media platform to flag dangerous or misleading health information or detection of fake news items that include hate speech (both of which were present in the dataset).

Methods Overview

Our goal was to score as high on mean accuracy in Cross Validation, as well as standard Evaluation Measures (Precision, Recall, Specificity, and especially F1) as possible. We approached the task agnostically and experimented with many different featuresets, classifiers, and vectorization methods:

Visualizations with the Raw Data Set

- word cloud
- dimensionality reduction (GLoVE)
- histograms

Vectorization and Features

- Bag of Words (Boolean)
    - unigram
    - bigram
    - trigram
- Sentiment Analysis (Opinion Lexicon) (Boolean)
- Quantitative Linguistic Features (continuous values)
    - average words per sentence
    - average word length
    - percent tokens in all caps
    - percent punctuation
- TF-IDF
- Part of Speech Tagging
- BERT embeddings

Classifiers

- Naive Bayes (NLTK - multinomial)
- Linear SVC (SciKit Learn)
- Naive Bayes (SciKit Learn - Gaussian)
- Logistic Regression (SciKit Learn)
- Random Forest (SciKit Learn)

Deep Learning Classifiers

- LSTM (RNN)

Unsupervised Learning

- BERT (pre-trained model)

We each tackled different elements of the project. We describe our contributions in our individual Methods Reports below.

METHODS

Preprocessing

Before working with the data, we performed some minimal processing. First, we removed 597 rows with 'NaN' values. Then, we assessed the number of both labels in the dataset and found that it was slightly unbalanced, with more positive label (36,509) items than negative labels (35,028). We balanced the labels and reshuffled the dataset.

Before proceeding, we ran both the 'title', or headline text and the 'text' or body text the dataset through a customized version of standard NLP preprocessing which entailed:

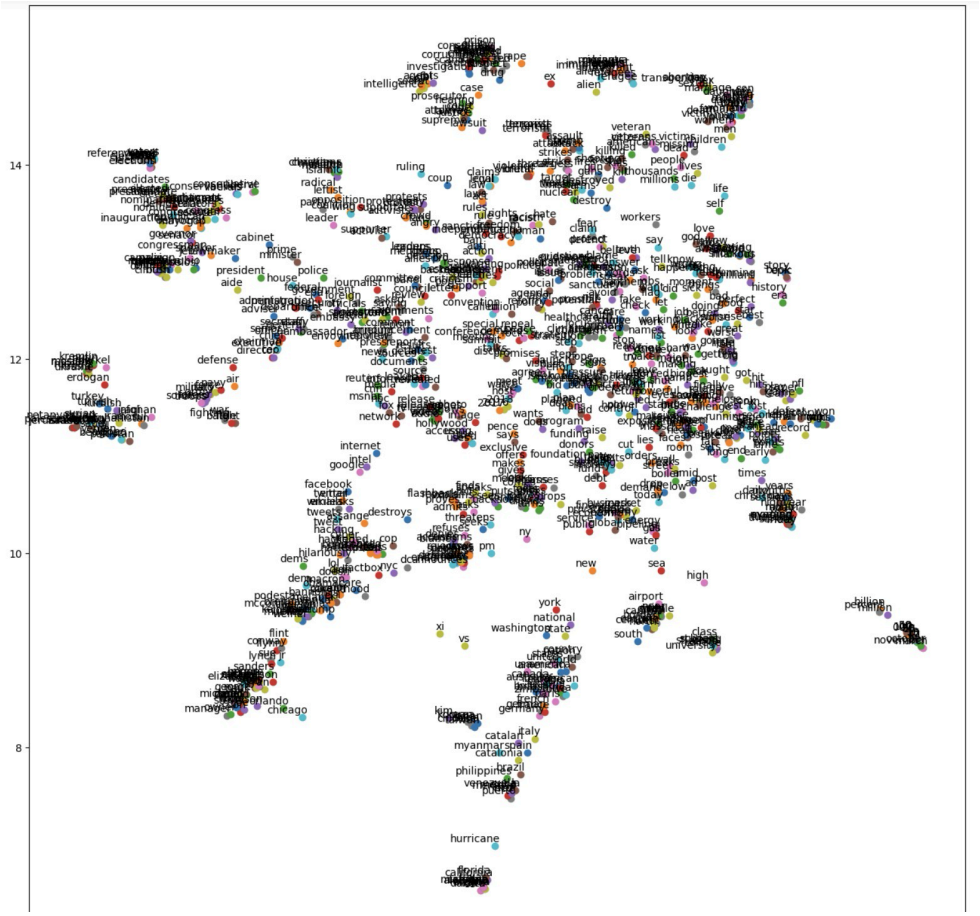
- lowercasing all alphabetic characters
- removal of hyphens and periods within words
- filtering out standard stopwords (excluding words for negation detection)
- removal of punctuation except for stylistically significant punctuation ('!', '#', '*')
    - removed punctuation included several special punctuation marks found in the dataset but not covered by string.punctuation
- use of a custom regex tokenizer to keep hashes attached to their hashtags and asterisks within obscured expletives (both common items in the dataset)
- stemming of all tokens using the NLTK Snowball Stemmer

Visualization:

We used a couple of different visualization techniques to better understand our data and detect any underlying patterns

Dimensionality Reduction:

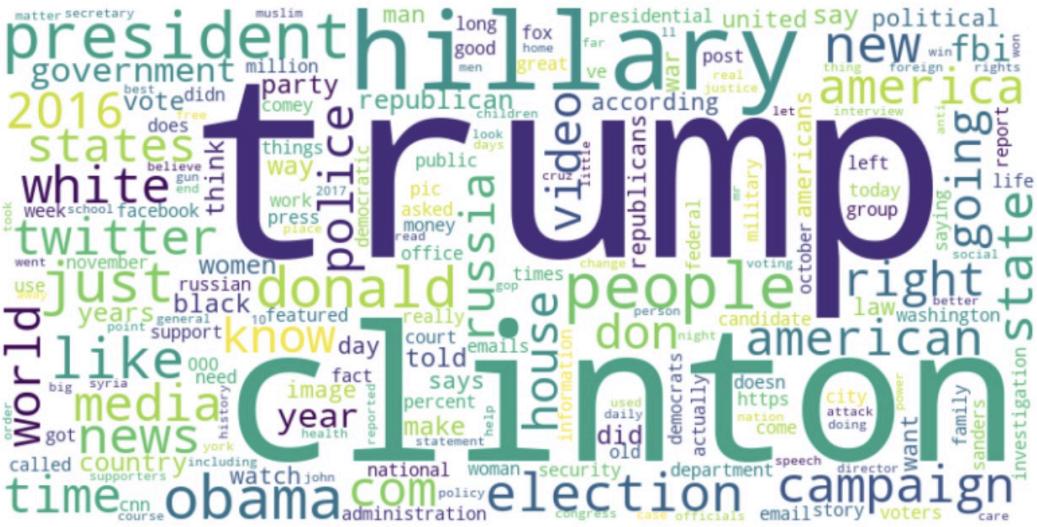
We used dimensionality reduction techniques to project our data on a 2D graph. This dimensionality reduction was performed to help us identify any clusters or groupings within our data. As you can see in the visualization below, our data revealed a few distinct clusters. Similar words, like countries and numbers are grouped together.



Word Cloud:

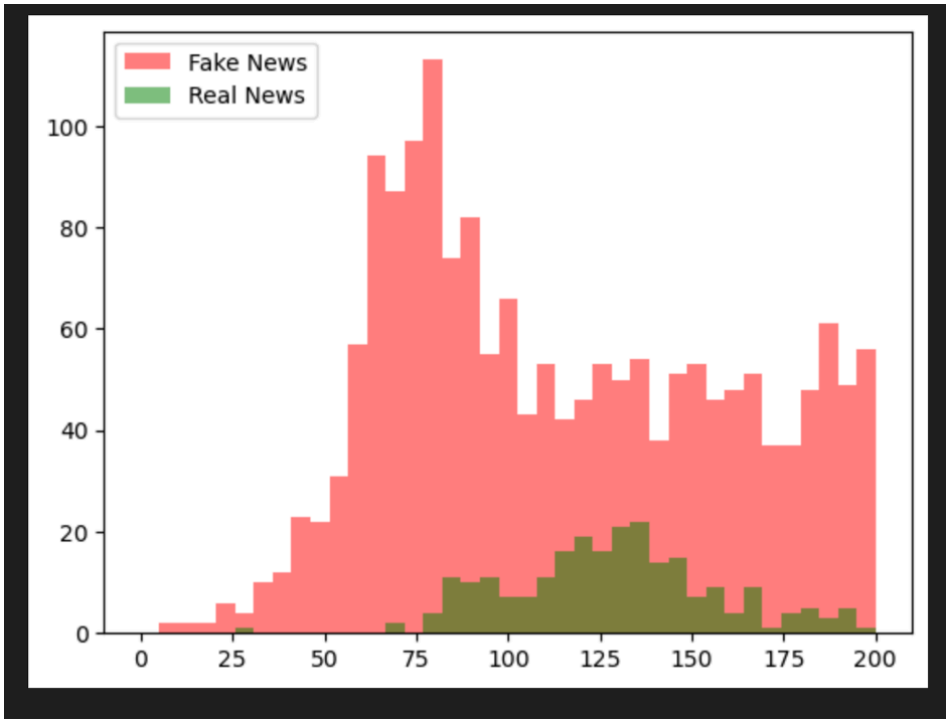
We used Word Clouds to visually represent the most frequent words in our data. We separated our text into two categories: fake news and real news. We generated a word cloud for each category and compared the two. The intention with this was to see if there are certain words that are more commonly used in fake news when compared to real news. As you can see in the two word clouds below, the words in both word clouds are more or less the same.

Real News Word Cloud:



Fake News Word Cloud:

Word Length Distribution: Histograms showing the distribution of word lengths for both genuine and deceptive news articles.



Bag of Words Features

Test 1 Discussion: Minimal Processing

Naive Bayes Classifier on Unigram Features of the Minimally Processed Dataset ('Title' field only)

Before working with the data, we performed some minimal processing. First, we removed 597 rows with 'NaN' values. Then, we assessed the number of both labels in the dataset and found that it was slightly unbalanced, with more positive label (36,509) items than negative labels (35,028). We balanced the labels and reshuffled the dataset.

We started by working with the minimally processed, label-balanced dataset, extracting unigram features and working at first with the "title" field only to create a baseline. The 'title' field was simply lowercase and tokenized using the NLTK word_tokenizer function. No further filtering or customization was done at this stage.

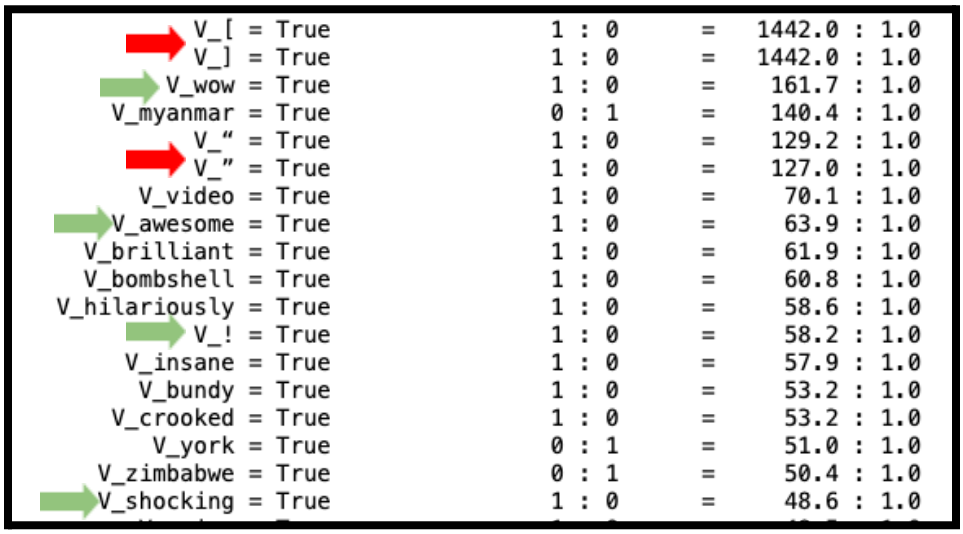
We used a **70:30 train/test split** for this classification task, as we did for the remaining tests in this section.

This model achieved surprisingly impressive results using the NLTK Naive Bayes Classifier, scoring **88.1% mean accuracy** with 10-fold cross evaluation.

Looking at the detailed evaluation metrics, the **F1 Score for both the positive and negative label are close to each other and to the mean accuracy**, suggesting that the model is generally well-balanced.

Test 1 **Featureset:** Unigrams of Minimally Processed Dataset (‘Title’ Only) **Classifier:** Naive Bayes (NLTK)					
	Precision	**Recall**	**F1**	**Specificity**	**Mean Accuracy**
0	0.850	0.930	0.888	0.838	**88.1%**
1	**0.924**	**0.838**	**0.879**	**0.930**	

However, looking at the **Most Informative Features**, we see some irregularities. Some of the features, such as "wow", "awesome", "shocking", and even "!" are plausible stylistic distinguishing features for fake news, indicative of a more sensationalistic rhetorical style.



But the two **Most Informative Features are open and closed square brackets**, and in the top ten we also have double quotation marks. These seem arbitrary and are less plausible as truly informative features. One possible explanation is that we are seeing **over-fitting due to specific sources in the dataset**, e.g. headline labels such as "[VIDEO]" (‘video’ is also in the top 10) might have come from a single fake news source that happens to have a lot of video content and consistently formats video titles this way.

Interestingly, though these outliers in Most Informative Features predict positive labels, in the detailed evaluation metrics, we see that **Recall**, or sensitivity, is higher for the negative label assignment, indicating a bias toward negative classification. And because the data is label-balanced, we know that this is not simply due to an imbalance in favor of negative labels in the dataset. We also see that **Precision and Specificity for positive (fake) label prediction is quite high at 92.4% and 93%** respectively, indicating that there are unigram features that are highly predictive of a positive label, with relatively few false positives. This may be related to the observation that a few features, most likely not very common across the dataset, are accounting for much of the accuracy in assigning the positive (fake) label.

Though this minimally processed featureset gives us a reasonably high level of accuracy, the model will not generalize well in a dataset that does not have these particular, arbitrary features. To correct this, we will perform some custom text pre-processing.

Test 2 Discussion: Preprocessing

Naive Bayes Classifier on Unigram Features of the Preprocessed Unigram Features —Headlines ('Title' field) only

Preprocessing the Text
Before proceeding, we ran both the 'title', or headline text and the 'text' or body text the dataset through a customized version of standard NLP preprocessing which entailed:

- lowercasing all alphabetic characters
- removal of hyphens and periods within words
- filtering out standard stopwords (excluding words for negation detection)
- removal of punctuation except for stylistically significant punctuation ('!', '#', '*')
    - removed punctuation included several special punctuation marks found in the dataset but not covered by string.punctuation
- use of a custom regex tokenizer to keep hashes attached to their hashtags and asterisks within obscured expletives (both common items in the dataset)
- stemming of all tokens using the NLTK Snowball Stemmer

Having taken the text through a customized version of standard pre-processing, **accuracy has dropped from 88 to 84%**.However, while we may have sacrificed some accuracy, we appear to have a more generalizable model.

Looking at the **Most Informative Features** we have fewer items that seem arbitrary and more words with a plausible relationship to the categorization of real vs. fake news.

```
       V_wow = True                1 : 0      =    162.3 : 1.0
   V_myanmar = True                0 : 1      =    141.1 : 1.0
         V_… = True                1 : 0      =     95.7 : 1.0
         V_! = True                1 : 0      =     56.9 : 1.0
      V_york = True                0 : 1      =     50.5 : 1.0
 V_brilliant = True                1 : 0      =     49.6 : 1.0
     V_onion = True                1 : 0      =     49.2 : 1.0
    V_finest = True                1 : 0      =     47.2 : 1.0
 V_breitbart = True                0 : 1      =     45.3 : 1.0
     V_video = True                1 : 0      =     38.8 : 1.0
         V_– = True                1 : 0      =     37.6 : 1.0
    V_antifa = True                1 : 0      =     35.8 : 1.0
   V_mnuchin = True                0 : 1      =     33.5 : 1.0
     V_epic = True                 1 : 0      =     32.8 : 1.0
    V_laugh = True                 1 : 0      =     32.6 : 1.0
     V_thug = True                 1 : 0      =     30.3 : 1.0
   V_furious = True                1 : 0      =     28.7 : 1.0
    V_unreal = True                1 : 0      =     27.8 : 1.0
        V_xi = True                0 : 1      =     26.6 : 1.0

                          *      *      *

   V_tantrum = True                1 : 0      =     22.4 : 1.0
   V_somalia = True                0 : 1      =     22.2 : 1.0
  V_cambodia = True                0 : 1      =     21.6 : 1.0
       V_rio = True                0 : 1      =     21.6 : 1.0
     V_viral = True                1 : 0      =     21.4 : 1.0
     V_idiot = True                1 : 0      =     20.3 : 1.0
     V_czech = True                0 : 1      =     19.6 : 1.0
    V_soccer = True                0 : 1      =     19.6 : 1.0
   V_kremlin = True                0 : 1      =     19.1 : 1.0
    V_tucker = True                1 : 0      =     19.1 : 1.0
   V_ireland = True                0 : 1      =     18.9 : 1.0
       V_kkk = True                1 : 0      =     18.3 : 1.0
 V_tillerson = True                0 : 1      =     18.2 : 1.0
     V_egypt = True                0 : 1      =     17.4 : 1.0
```

Drilling down into the list a bit, we can see **four broad categories of unigrams** that are correlated with fake news categorization:

- **sensationalistic** language (e.g. "brutal", "epic", "unreal")
- **informal**, esp. derogatory or dismissive language (e.g. "cop", "stupid", "idiot")
- **"hot-button"** topics (e.g. "antifa", "kkk", "rapist")
- **special punctuation** typical of online language use (i.e. "!", "...")

At the same time, names of **non-controversial political figures, places, or roles** (e.g. "Macron", "Myanmar", "envoy") are strongly associated with **real news**.

Looking at the **Evaluation Metrics** we see that **Precision, Recall, and Specificity are all much closer** for both the positive and negative labels and relative to each other. Specificity and Precision have both gone down for the positive label, and Recall has gone down for the

negative label. This suggests that the preprocessing we have done has gone a long way to correcting the biases of the minimally processed dataset.

Test 2 **Featureset:** Preprocessed Unigrams (‘Title’ Only) **Classifier:** Naive Bayes (NLTK)					
	Precision	**Recall**	**F1**	**Specificity**	**Mean Accuracy**
0	0.835	0.855	0.845	0.833	**84.0%**
1	**0.853**	**0.833**	**0.842**	**0.855**	

The difference that remains between the scores shows the same pattern, with the model being slightly more sensitive to negative label items and having a slightly higher precision and specificity for the positive label.

Having found a good recipe for pre-processing, next we will take a look at expanding the corpus from title-only to a concatenated title-and-text.

Test 3: Naive Bayes Classifier on Combined Title + Text Unigram

Surprisingly, adding the additional text from the 'text' column to the dataset did not improve accuracy. In fact, it **significantly reduced accuracy from 84%** looking only at the titles, **to 78.16%** mean accuracy on a 5-fold cross validation.

Interestingly, **Most Informative Features** for the combined text included several characters that appear to be from a **Cryllic characterset** (e.g. 'и', 'на','в'), and these are all strongly associated with the **fake news** category. However, there were more features strongly associated with “real” news in this featureset. We can also see that individually, the unigram features are less powerfully informative in this featureset than they were in the title-only set.

25 Most Informative Features: Combined Title + Text Unigram

```
     V_getti = True                1 : 0      =    101.9 : 1.0
   V_catalan = True                0 : 1      =     90.6 : 1.0
  V_rohingya = True                0 : 1      =     75.8 : 1.0
         V_и = True                1 : 0      =     71.3 : 1.0
 V_catalonia = True                0 : 1      =     66.2 : 1.0
       V_000 = True                0 : 1      =     53.4 : 1.0
   V_myanmar = True                0 : 1      =     43.8 : 1.0
         V_в = True                1 : 0      =     43.2 : 1.0
        V_на = True                1 : 0      =     41.6 : 1.0
    V_reuter = True                0 : 1      =     37.7 : 1.0
V_screenshot = True                1 : 0      =     32.2 : 1.0
     V_bundi = True                1 : 0      =     23.9 : 1.0
       V_com = True                0 : 1      =     22.7 : 1.0
    V_macron = True                0 : 1      =     14.4 : 1.0
    V_hilari = True                1 : 0      =     12.0 : 1.0
       V_que = True                1 : 0      =     11.3 : 1.0
  V_overhaul = True                0 : 1      =     11.3 : 1.0
       V_via = True                1 : 0      =     10.3 : 1.0
     V_mugab = True                0 : 1      =     10.0 : 1.0
        V_rt = True                1 : 0      =      9.5 : 1.0
      V_beij = True                0 : 1      =      9.5 : 1.0
 V_pyongyang = True                0 : 1      =      9.3 : 1.0
V_bangladesh = True                0 : 1      =      8.8 : 1.0
        V_xi = True                0 : 1      =      8.7 : 1.0
       V_doj = True                1 : 0      =      8.5 : 1.0
```

Looking at the **confusion matrix**, we can see there has been some **increase in True Positives**, but it is offset by more than **twice the False Positives** and **significantly fewer True Negatives**.

Title Unigram (baseline)		Combined Unigram (Title + Text)	
8968	1590	**8992**	**1566**
1226	**9233**	**3000**	**7456**

Rows = Actual
Columns = Predicted

Per the **evaluation metrics**, we see that with the combined text and title, there is an inversion of the relationship between the positive and negative labels that we have been seeing in the 'title only' dataset. In the combined title and text dataset, **Precision** was significantly **higher for the negative label** than for the positive label, indicating that we are getting **more false positives**. Correspondingly, we see that Recall for the positive label is also significantly higher,

indicating a bias toward predicting the postive fake news label. Conversely, **Specificity** is lower, indicating that the model is not identifying true negatives effectively.

Test 3 **Featureset:** Preprocessed Combined 'Title' + 'Text' Unigrams **Classifier:** Naive Bayes (NLTK)					
	Precision	**Recall**	**F1**	**Specificity**	**Mean Accuracy**
0	0.826	0.713	0.766	0.852	**78.16%**
1	0.750	0.852	0.798	0.713	

Because the additional data in the combined text both significantly reduces accuracy, introduces imbalances in Precision and Recall, and also slows processing down, we will be focusing on the headlines ('title') only and using it as a baseline in the remaining experiments. Next we will look at the effects of adjusting vocabulary length.

TEST 4: Comparing Effects of Vocabulary Size

Systematically testing different **vocabulary sizes** for the title unigram featuresets as a percentage of the normalized set of words gave improvements in accuracy as the size increased, **improving accuracy from 84% to 86.9%**.

However, there were diminishing returns even as execution time became significantly longer, increasing from just two minutes for the 10% vocabulary (2577 words) up to around six minutes for the 20% vocab size (5155 words) on simple accuracy.

Going forward, the **15% (word count: 3866) vocabulary with 86.6% accuracy for unigram features** is a reasonable compromise. This featureset will serve as the **baseline** for all further experiments.

Test 4 **Featureset:** Title Unigrams with Variable Vocab Sizes **Classifier:** Naive Bayes (NLTK)					
Vocab Size	**Precision**	**Recall**	**F1**	**Specificity**	**Mean Accuracy**
10%	0.875	0.842	0.858	0.879	85.9%
15%	**0.880**	**0.849**	**0.864**	**0.883**	**86.5%**
20%	0.884	0.851	0.8671	0.888	86.9%

TEST 5: Combining Unigram + Bigram Features (Title Only)

Bigram features were extracted with a filter applied to limit **bigrams to alphabetic tokens with a length longer than two characters and a minimum frequency of 50 in the corpus**. Bigrams were scored using PMI, but because the applied filters already yielded **only 123 new bigram features**, all of the bigrams were used, regardless of score.

Looking at the bigrams that we were able to extract from the dataset, we see that it is dominated by **proper nouns --person and place names** (e.g. Kellyann Conway, Puerto Rico). But there are also several that related to **potentially politically divisive issues** that might be predictive of real or fake news (e.g. 'Planned Parenthood', 'climate change', 'travel ban', ‘illegal alien’).

Top 30 Title Bigrams Scored Using PMI	
1. 'kellyann', 'conway' 2. 'elizabeth', 'warren' 3. 'ben', 'carson' 4. 'puerto', 'rico' 5. 'megyn', 'kelli' 6. 'sean', 'spicer' 7. 'boiler', 'room 8. 'saudi', 'arabia' 9. 'room', 'ep' 10. 'mike', 'penc' 11. 'attorney', 'general' 12. 'ted', 'cruz' 13. 'prime', 'minist' 14. 'sexual', 'assault' 15. 'paul', 'ryan'	16. 'illeg', 'alien' 17. 'berni', 'sander' 18. 'health', 'care' 19. 'even', 'brief' 20. 'live', 'matter' 21. 'wall', 'street' 22. 'suprem', 'court' 23. 'america', 'finest' 24. 'onion', 'america' 25. 'plan', 'parenthood' 26. 'climat', 'chang' 27. 'travel', 'ban' 28. 'north', 'korea' 29. 'finest', 'news' 30. 'north', 'carolina'

Despite this, **combining bigram and unigram features did not result in an increase in Mean Accuracy over unigram features alone.** In fact, Mean Accuracy was identical to the 5th decimal point. Precision, Recall, F1 and Specificity scores were also effectively identical.

Test 5 **Featureset:** Bigram + Unigram Features (Title Only) **Classifier:** Naive Bayes (NLTK)					
	Precision	**Recall**	**F1**	**Specificity**	**Mean Accuracy**
0	0.853	0.883	0.868	0.849	**86.5%**
1	**0.880**	**0.849**	**0.864**	**0.883**	

The reason for this becomes clear when looking at the **Most Informative Features**. All but one bigram feature is at the very bottom of the list, with only a False value and a probability ratio of 1.0:1.0. This suggests that the bigrams were too sparse to have any meaningful impact on the model's evaluation metrics.

Title Bigram Most Informative Features

```
  B_america_finest = False              0 : 1      =      1.0 : 1.0
B_attorney_general = False              0 : 1      =      1.0 : 1.0
    B_barack_obama = False              0 : 1      =      1.0 : 1.0
      B_ben_carson = False              0 : 1      =      1.0 : 1.0
    B_berni_sander = False              0 : 1      =      1.0 : 1.0
    B_bill_clinton = False              0 : 1      =      1.0 : 1.0
      B_black_live = False              0 : 1      =      1.0 : 1.0
     B_boiler_room = False              0 : 1      =      1.0 : 1.0
     B_border_wall = False              0 : 1      =      1.0 : 1.0
 B_breitbart_trump = False              0 : 1      =      1.0 : 1.0
       B_brief_new = False              0 : 1      =      1.0 : 1.0
      B_call_trump = False              0 : 1      =      1.0 : 1.0
    B_climat_chang = False              0 : 1      =      1.0 : 1.0
B_clinton_campaign = False              0 : 1      =      1.0 : 1.0
   B_clinton_email = False              0 : 1      =      1.0 : 1.0
```

Because of the small size of the bigram features in the the title portion of the dataset, and the extreme sparseness of the features, title bigrams are not useful in training a model. Next, we experimented with extracting bigrams from the 'text' body portion of the dataset, which has a significantly larger vocabulary and more data for each item to see if the more robust data would yield more informative bigram features.

TEST 6: Combining Title Unigram Features with Text Bigram Features

To see if a larger dataset and correspondingly larger featureset would improve bigram performance, we experimented with combining the title-only unigrams with bigram features extracted from the ‘text’, or body text, component of the dataset.

As before, we extracted bigrams with elements that were greater than two characters in length and only contained alphabetic characters.

Because the dataset is much larger in the ‘text’ body field than in the ‘title’, we were able to set the frequency filter significantly higher and still get a large number of features. With a frequency filter of 500, we got **1252 bigrams features**. Because we wanted a larger featureset, we kept all of these, though the top 30 scored by PMI can be seen below. Unsurprisingly, person names again dominate, moreso, in fact, than in the ‘title’ bigram features—probably because public figures are often referred to by only their last name in news headlines.

Top 30 Text Bigrams Scored Using PMI	
1. 'suu', 'kyi' 2. 'hong', 'kong' 3. 'shi', 'ite' 4. 'bashar', 'alassad' 5. 'silicon', 'valley' 6. 'reinc', 'priebus' 7. 'antonin', 'scalia' 8. 'goldman', 'sach' 9. 'lindsey', 'graham' 10. 'kellyann', 'conway' 11. 'emmanuel', 'macron' 12. 'asylum', 'seeker' 13. 'las', 'vega' 14. 'chancellor', 'angela' 15. 'nanci', 'pelosi' 16. 'puerto', 'rico'	17. 'julian', 'assang' 18. 'xi', 'jinp' 19. 'tayyip', 'erdogan' 20. 'loretta', 'lynch' 21. 'kim', 'jong' 22. huma', 'abedin' 23. 'palm', 'beach' 24. 'jare', 'kushner' 25. 'ronald', 'reagan 26. 'benjamin', 'netanyahu' 27. 'mitch', 'mcconnel' 28. 'mitt', 'romney' 29. 'tim', 'kain' 30. 'angela', 'merkel' 31. 'ethnic', 'cleans' 32. 'super', 'pac'

Surprisingly, this significant addition to the featureset still yielded nearly identical evaluation metric scores using the NLTK Naive Bayes classifier.

Test 6 **Featureset:** Title Unigram + Text Bigram Features **Classifier:** Naive Bayes (NLTK)					
	Precision	**Recall**	**F1**	**Specificity**	**Mean Accuracy**
0	0.853	0.883	0.868	0.849	**86.5%**
1	**0.880**	**0.849**	**0.864**	**0.883**	

When looking at the **Most Informative Features** for the model, the reason becomes evident. Only one bigram figured significantly in the index: “Suu Kyi” was highly predictive of the false (real news) label. The remainder of the bigram features were at the very bottom of the list with an odds ratio of 1.0: 1.0.

Top 10 Most Informative Features for Title Unigram + Text Bigram

```
    V_hilari = True                1 : 0      =    187.1 : 1.0
       V_wow = True                1 : 0      =    162.3 : 1.0
  V_myanmar = True                 0 : 1      =    141.1 : 1.0
        V_… = True                 1 : 0      =     95.7 : 1.0
  V_suu_kyi = True                 0 : 1      =     94.6 : 1.0
   V_awesom = True                 1 : 0      =     64.6 : 1.0
    V_bundi = True                 1 : 0      =     57.2 : 1.0
        V_! = True                 1 : 0      =     56.9 : 1.0
    V_crook = True                 1 : 0      =     53.9 : 1.0
     V_york = True                 0 : 1      =     50.5 : 1.0
```

TEST 7: Combining Title Unigram & Trigram Features

As a final n-gram experiment, we created a featureset that combined unigram and trigram features from the ‘title’ or headline component of the dataset. Trigrams were extracted applying a filter to limit them to alphabetic tokens with a length longer than two characters and a minimum frequency of 50 in the corpus. The were scored using PMI, but because the applied filters already yielded only 122 new, trigram features, all of the trigrams were used, regardless of score.

Looking at the trigrams that we were able to extract from the dataset, we are getting a bit more context than we were with bigrams. For instance, "Puerto Rico" becomes **"Puerto Rican Debt"** and "Lives Matter" becomes **"Black Lives Matter"**. Other newsworthy items also appear in the list, such as **"Dakota Access Pipeline"** and **"Voter ID Law"**.

It's interesting to note that the most common trigram is **'новое восточное обозрение'**, which translates to **"New Eastern Outlook"** (also in the top-scoring list of trigrams). This is the name of a Russia-based online journal that has come to be associated with fake news and was identified by the U.S. Dept. of State as part of "Russia's disinformation and propaganda system" in 2020.

Top 30 Title Trigrams Scored Using PMI

1. 'новое', 'восточное', 'обозрение' 2. 'dakota', 'access', 'pipelin' 3. 'patrick', 'henningsen', 'live 4. 'boiler', 'room', 'ep 5. 'onion', 'america', 'finest' 6. 'endingf', 'news', 'network' 7. 'puerto', 'rico', 'debt' 8. 'saturday', 'night', 'live' 9. 'border', 'patrol', 'agent' 10. 'wednesday', 'even', 'brief' 11. 'flint', 'water', 'crisi' 12. 'thursday', 'even', 'brief' 13. 'friday', 'even', 'brief' 14. 'tuesday', 'even', 'brief' 15. 'finest', 'news', 'sourc'	16. 'south', 'china', 'sea' 17. 'new', 'eastern', 'outlook' 18. 'director', 'jame', 'comey' 19. 'south', 'korea', 'moon' 20. 'black', 'live', 'matter' 21. 'voter', 'id', 'law' 22. 'america', 'finest', 'news' 23. 'fbi', 'director', 'jame' 24. 'suprem', 'court', 'nomine' 25. 'british', 'pm', 'may' 26. 'suprem', 'court', 'justic' 27. 'live', 'matter', 'terrorist' 28. 'fbi', 'director', 'comey' 29. 'hous', 'speaker', 'ryan' 30. 'nation', 'secur', 'advis'

Despite the additional context, **combining trigram features and unigram also did not result in an increase in accuracy or other evaluation metrics over unigram features alone.** Frustratingly, we again get nearly identical evaluation measures for the model:

Test 7 **Featureset:** Title Unigram + Trigram Features **Classifier:** Naive Bayes (NLTK)					
	Precision	**Recall**	**F1**	**Specificity**	**Mean Accuracy**
0	0.853	0.883	0.868	0.849	**86.5%**
1	**0.880**	**0.849**	**0.864**	**0.883**	

The reason for this again becomes clear when looking at the **Most Informative Features**. As with the bigram features, all trigram features are at the very bottom of the list and have a probability ratio of 1.0:1.0. In fact, **the first trigram does not even appear in the list until you go out ~5000 items**, and then its value is False: [('T_новое_восточное_обозрение', False)

To get a clearer picture of the effects of the n-grams feature variations, we have lined up the confusion matrices for all five n-gram experiments below. Each featureset yields a slightly different confusion matrix, but —with the exception of the combined text and title unigram featureset, which scores significantly lower — only a few items shift between quadrants with each iteration, and none of them decisively improves on the title-only unigram baseline.

It seems that for this dataset, the title-only unigram is so powerful that other n-gram featuresets yield no improvement in the model.

N-gram Featureset Confusion Matrices for NLTK Naive Bayes

Title Unigram (baseline)		Combined Unigram (Title + Text)		Unigram + Bigram (Title Only)		Unigram + Trigram (Title Only)		Title Unigram + Text Bigram	
8968	1590	**8992**	1566	**8965**	1593	**8967**	1591	**8968**	1590
1226	**9233**	3000	**7456**	1223	**9236**	1225	**9234**	1227	**9232**

Rows = Actual
Columns = Predicted

Because using a combined trigram/unigram featureset did not improve accuracy of the unigram baseline, we did not use it in subsequent models.

TEST 8: Unigram Features + Opinion Lexicon Features

Hypothesis: Fakenews will be more strongly associated with a higher use of emotionally charged language whether negative or positive.

To perform sentiment analysis, we used **Bing Liu's Opinion Lexicon**[2]. We selected this lexicon because it was developed for a similar classification task and is focused on the detection of strong opinions. It is also an unscored binary lexicon, so we were able to stem the lexicon entries to match our pre-processed dataset without having to handle different scores belonging to multiple words that share a common stem.

Count vs. Boolean Features
After experimenting with generating positive and negative counts and seeing a small decrease in accuracy for the NB classifier over the unigrams baseline, we modified the feature extractor to a **boolean "strong" positive or negative** based on a **>=2 count threshold**, which gave slightly better results. This finding makes sense in the context of fake news evaluation, because this allows the evaluation to focus simply on strongly emotionally charged language, rather than particular discrete counts.

---

[2] Citation: This lexicon was first published by Bing Liu in conjunction with: Minqing Hu and Bing Liu, ``Mining and summarizing customer reviews.'', Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD-2004), Seattle, Washington, USA, Aug 22-25, 2004.

There were small improvements in correct identification of negative items, and a small reduction in sensitivity to positives with the addition of the Boolean valued opinion lexicon features.

Test 8 **Featureset:** Title Unigram + Boolean Opinion Lexicon Features **Classifier:** Naive Bayes (NLTK)					
	Precision	**Recall**	**F1**	**Specificity**	**Mean Accuracy**
0	0.850	0.885	0.867	0.845	**86.5%**
1	**0.881**	**0.845**	**0.863**	**0.885**	

Title Unigram (baseline)		**Title Unigram + Opinion Lex.**	
8968	1590	**8925**	1633
1226	**9233**	1200	**9259**
Rows = Actual **Columns = Predicted**			

Further simplifying the opinion feature to reflect only whether the sum of the count of positive and negative for a given document was >= 2 slightly reduced accuracy.

TEST 9: Unigram Features + Quantitative Linguistic Features

Although the pre-processing steps that we undertook early in the process have been helpful in improving the Bag of Words (BoW) features, there is information lost in that process that might be useful to building a robust classification model.

We loaded the .csv of the minimally processed dataset created before pre-processing and extracted the following quantitative linguistic features:

- **Average word length**
  - best done on unstemmed data without stopword filtering
- **Percent of words in ALL CAPS**
  - cannot be done with lowercase tokens

- **Percent of each document that is punctuation**
    - cannot be done with data in which most or all punctuation has been removed

- **Average length in words for each sentence in the main "text" body**
    - cannot be done without sentence tokenization of the ‘text’

While looking at quantitative features in isolation did not perform as well as unigrams alone (75% vs. 85% accuracy), the addition of the quantitative linguistic features to unigrams yielded the most improvement in model performance of the baseline of any of the additional features.

Test 9 **Featureset:** Title Unigram + Quantitative Linguistic Features **Classifier:** Naive Bayes (NLTK)					
	Prec	**Recall**	**F1**	**Spec**	**Mean Acc**
0	0.853	0.918	0.884	0.843	**87.9%**
1	**0.912**	**0.843**	**0.876**	**0.918**	

Looking at the **Confusion Matrix**, we can see these shifts clearly—false positives have gone down significantly and true negatives have also gone up significantly. True positives have increased slightly. The bias toward a negative classification can be seen in the relatively large spread between Recall and Specificity and the corresponding increase in false negatives.

Title Unigram (baseline)		**Title Unigram + Quant Ling.**	
8968	1590	**8898**	**1660**
1226	**9233**	**859**	**9600**
Rows = Actual **Columns = Predicted**			

Looking at the **Most Informative Features** for the NB classifier reveals that several of the quantitative linguistic features appear in the top 10, and all four quantitative features appear in the top 100, with an odds ratio above 20:1, demonstrating a significant contribution to the model.

Top 10 Most Informative Features for Unigrams + Quantitative Linguistic Features

```
      all_caps = 1.0                1 : 0      =    347.1 : 1.0
avg_sen_length = 0.0                1 : 0      =    338.9 : 1.0
      V_hilari = True               1 : 0      =    187.1 : 1.0
      all_caps = 0.36               1 : 0      =    183.1 : 1.0
         V_wow = True               1 : 0      =    162.3 : 1.0
      all_caps = 0.24               1 : 0      =    144.9 : 1.0
     V_myanmar = True               0 : 1      =    141.1 : 1.0
           V_… = True               1 : 0      =     95.7 : 1.0
      all_caps = 0.43               1 : 0      =     93.4 : 1.0
      all_caps = 0.4                1 : 0      =     67.3 : 1.0
```

TEST 10: Large Composite Featureset

Title Unigrams/Text Bigrams/Opinion Lexicon/Quantitative Linguistic Features

As a final experiment with the unigram featureset, we combined several of the featuresets from past experiments to see if in aggregate they might be more than the sum of their parts. We included the largest additional featureset that performed reasonably well, the text bigrams features, as well as the Boolean opinion lexicon, and all of the quantitative linguistic features.

Although the **Most Informative Features** reported that at least one feature from each featureset had an odds ratio above 1.0:1.0, **mean accuracy and all evaluation metrics were slightly reduced**.

Test 10
Featureset: Title Unigram + Quantitative Linguistic Features
Classifier: Naive Bayes (NLTK)

	Prec	Recall	F1	Spec	Mean Acc
0	0.850	0.917	0.882	0.840	**87.79%**
1	**0.910**	**0.840**	**0.874**	**0.917**	

TEST 11: Unigram Features + Quant. Linguistic Features with SciKit Learn Classifiers

Because the combined title unigram and quantitative linguistic featuresets produced the best performing models using the NLTK Naive Bayes classifier, I next experimented with a few different SciKit Learn classifiers to see if any of them produced a better model.

Both the **Linear SVC** and the **Logistic Regression** classifiers performed better than the NLTK Naive Bayes classifier as shown below, with a **maximum mean accuracy coming from the Logistic Regression model at 91%**.

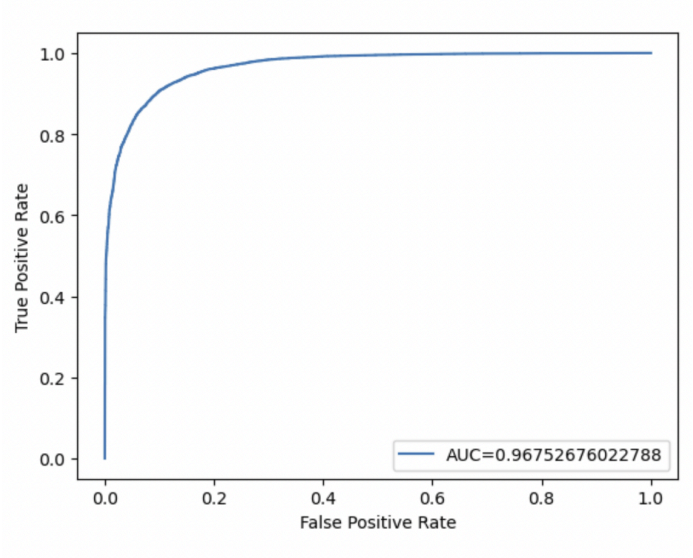
Test 11 **Featureset:** Title Unigram + Quantitative Linguistic Features **Classifiers:** Various SciKit Learn					
Classifier	**Prec**	**Recall**	**F1**	**Spec**	**Mean Acc**
Linear SVC	0.91	0.90	0.90	0.91	**90%**
Naive Bayes (Gaussian)	0.83	0.73	0.78	0.85	**79%**
Logistic Regression	**0.91**	**0.90**	**0.91**	**0.91**	**91%**

The **Gaussian Naive Bayes** classifier performed significantly more poorly than the NLTK Naive Bayes Classifier, which may be due to the fact that the NLTK Naive Bayes uses multinomial distribution, best suited for multinomial classification distributions, rather than Gaussian, which is best suited to distributing continuous values. While all of the quantitative linguistic features have continuous values and are dense matrices on their own, the unigram features they are combined with, which make up the vast majority of the features, are sparse and binary (Boolean), and therefore are not well-suited to a classifier applying Gaussian distribution.

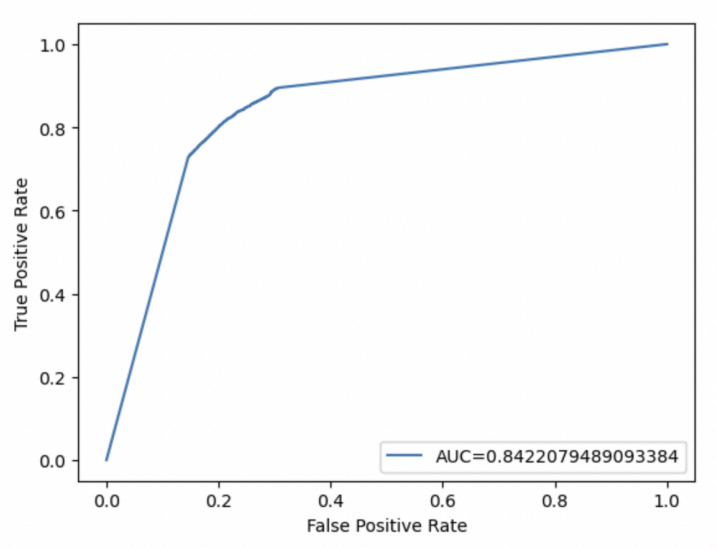
As an additional point of comparison between these three models, We plotted the Receiver Operating Characteristic (ROC) curve and generated an Area under the Curve (AUC) score.

Logistic Regression came out on top with an AUC of 0.97.

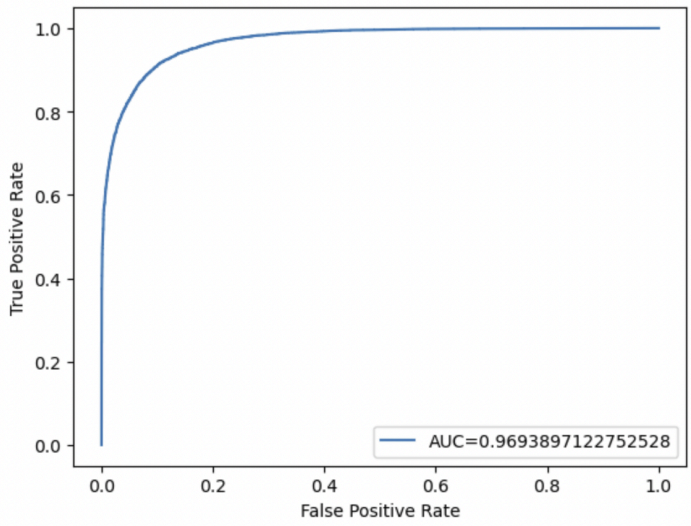
AUC ROC Curve for SVC Linear Classifier:



AUC ROC for Gaussian Naive Bayes Classifier:



AUC ROC for Logistic Regression:



Different types of vectorizations

In our project, we used 4 different types of word vectorizations or embeddings techniques tailored to specific tasks.

Bag of Words:
For models using feature sets, we used Bag of Words vectorization. The limitation of Bag of Words is that it doesn't measure the distance or similarity between words. Therefore, it doesn't capture the semantic relationships between words.

GLOVE:
As a result, when applying UMAP for dimensionality reduction we used GLOVE vectors. When GLOVE maps words onto a vector space, it keeps words that are similar in meaning closer together. This type of vectorization helps when we want to understand the nuances between very similar words. GLOVE vectors are ideal for quick visualizations such as our UMAP plot. However, where GLOVE falls short is understanding the context in which the word is used. The word 'bank' has the same word embedding in GLOVE regardless of whether it is used in the context of a financial bank or a river bank.

BERT:
To address the limitations of GloVe, we used BERT (Bidirectional Encoder Representations from Transformers) embeddings in our Deep Learning models. BERT considers the context surrounding a word in a sentence. More specifically, it accounts for context in both the left and right directions. In BERT embeddings, there are two different embeddings for the word 'bank', depending on the context, such as in 'river bank' versus 'bank account.'

TF-IDF:
Lastly, for our machine learning models, we used TF-IDF. TF-IDF is a word vectorization method which represents the relative importance of a word in the document as well as the whole corpus. TF-IDF is a more sophisticated vectorization method than Bag of Words while not being as computationally intensive as GLOVE or BERT embeddings. This made TF-IDF the ideal choice for the majority of our models.

Featuresets

- Unigrams
- Unigrams + Bigrams

- Unigrams + Bigrams + Subjectivity
- Unigrams + Bigrams + Quantitative linguistic features
- Bigrams + Bigrams + POS

	precision	recall	f1-score	support
0	0.94	0.82	0.88	474
1	0.85	0.96	0.90	526
accuracy			0.89	1000
macro avg	0.90	0.89	0.89	1000
weighted avg	0.90	0.89	0.89	1000

Machine Learning

In the fake news classification project, three machine learning models were employed to distinguish between genuine and deceptive news articles: Logistic Regression, Support Vector Machine (SVM), and Random Forest Classifier. Each model was evaluated based on its accuracy, precision, recall, and overall classification performance, as detailed in confusion matrices and classification reports. Here's a comparative analysis of these models:

1. **Logistic Regression**

- Accuracy: 78%

- Strengths:

  - Provides a good baseline with relatively simple implementation.

  - Effective in cases where the relationship between the data and the target is approximately linear.

- Weaknesses:

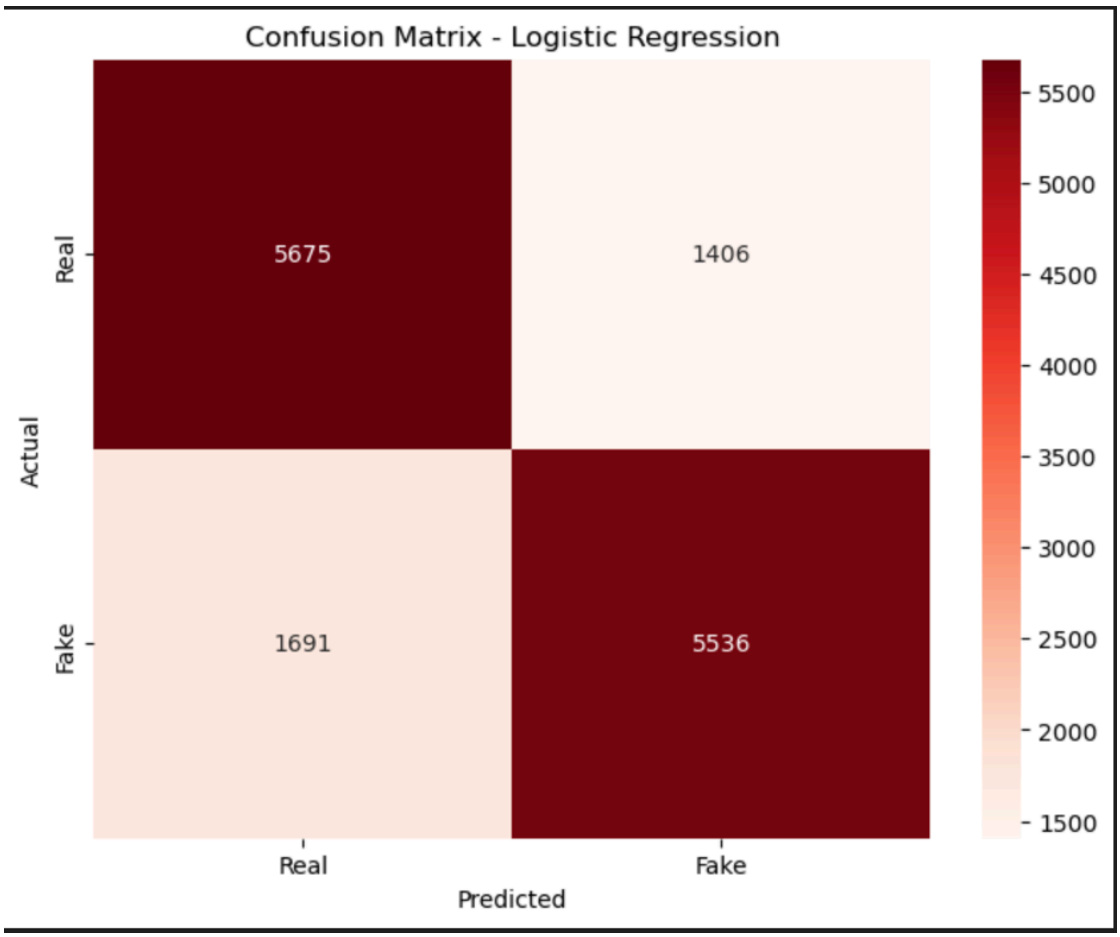
  - Might underperform if the relationship between features and the target is complex and nonlinear.

  - Equal precision and recall suggest it doesn't favor one class over the other, which might be a drawback in scenarios where one type of misclassification is costlier than the other.

```
Accuracy: 0.78
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.80      0.79      7081
           1       0.80      0.77      0.78      7227

    accuracy                           0.78     14308
   macro avg       0.78      0.78      0.78     14308
weighted avg       0.78      0.78      0.78     14308
```



2. **Support Vector Machine (SVM)**

- Accuracy: 79%

- Strengths:

  - Known for its effectiveness in high-dimensional spaces and when there is a clear margin of separation between classes.

  - Higher precision for fake news suggests it is more reliable for identifying deceptive content, reducing false positives.

- Weaknesses:

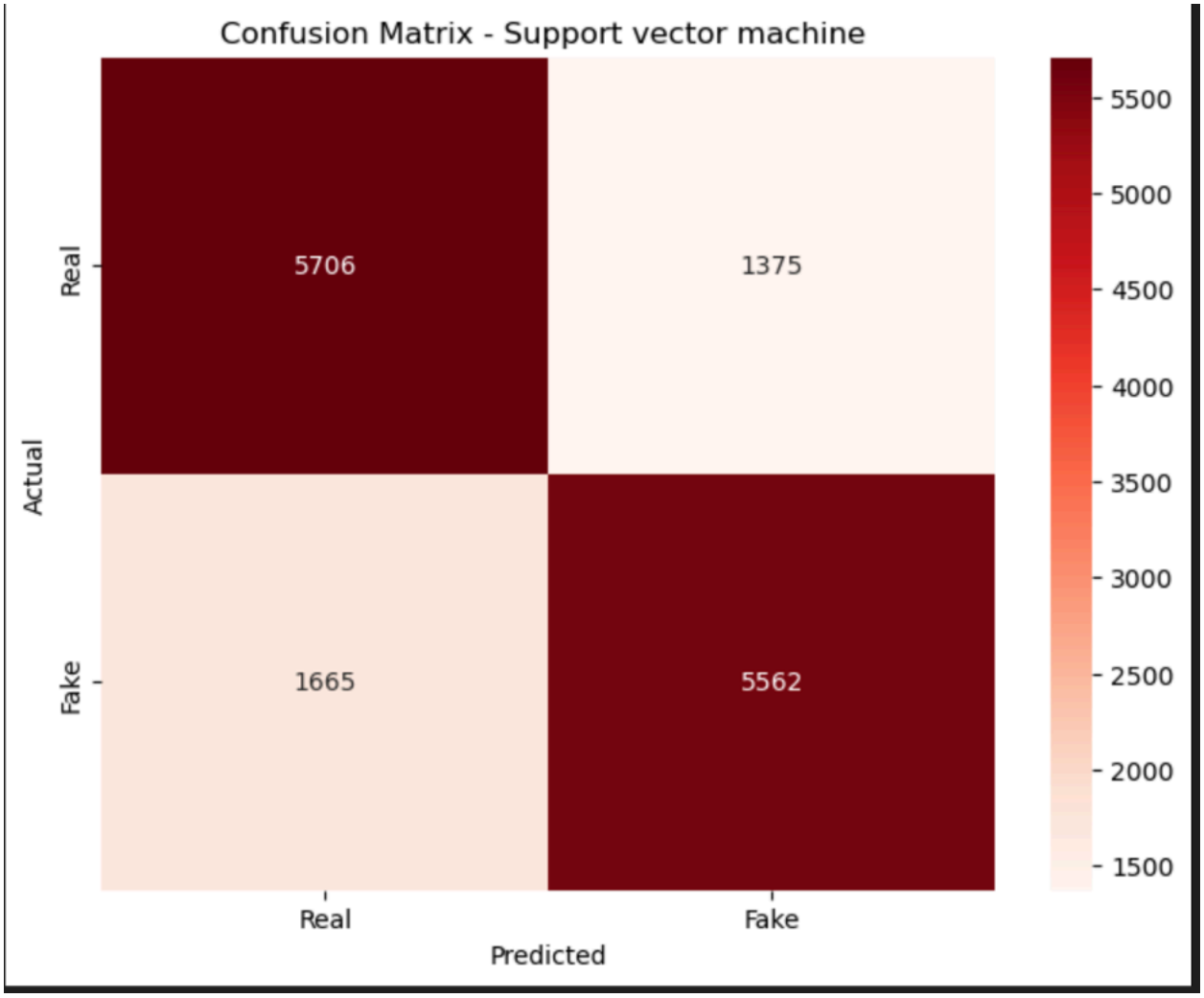
  - Computationally intensive, especially with large datasets.

  - Similar overall accuracy to Logistic Regression, indicating that linear kernel might not be capturing all the complexities of the data.

```
Accuracy: 0.79
Classification Report for SVM:
              precision    recall  f1-score   support

           0       0.77      0.81      0.79      7081
           1       0.80      0.77      0.79      7227

    accuracy                           0.79     14308
   macro avg       0.79      0.79      0.79     14308
weighted avg       0.79      0.79      0.79     14308
```



Random Forest Classifier

- Accuracy: 86%

- Strengths:

  - Best performance among the three, indicating a strong capability to handle non-linear relationships between features.

  - Robust against overfitting due to the ensemble method, which uses multiple decision trees to ensure generalizability.

  - Offers feature importance scores, which can be insightful for understanding which features most affect fake news detection.

- Weaknesses:

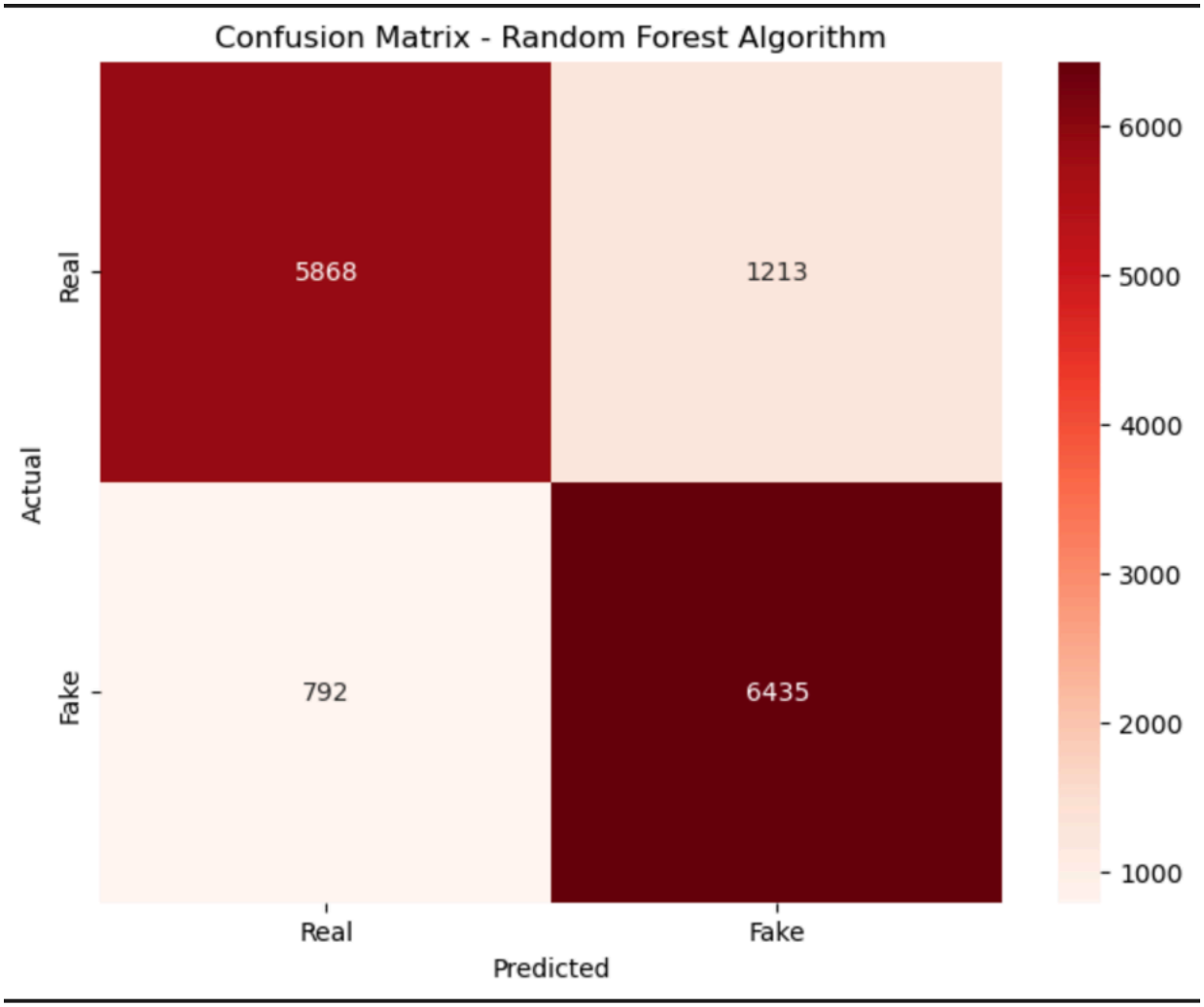
  - More complex and computationally demanding than Logistic Regression or linear SVM.

  - Requires careful tuning of parameters like the number of trees and depth of each tree to prevent underfitting or overfitting.

```
Accuracy: 0.86
Classification Report - Random Forest Algorithm:
              precision    recall  f1-score   support

           0       0.88      0.83      0.85      7081
           1       0.84      0.89      0.87      7227

    accuracy                           0.86     14308
   macro avg       0.86      0.86      0.86     14308
weighted avg       0.86      0.86      0.86     14308
```



As a quick test at the end, we tried adding several of the quantitative linguistic features used with the unigram featuresets to see how they would perform with TF-IDF vectorized features and founded that it actually gave us some of our highest scores :

```
Precision: 0.939 / Recall: 0.951 / Accuracy: 0.944
F1: 0.945
```

```
Neg_Precision: 0.949 / Neg_Recall: 0.936 / Neg_Accuracy: 0.944
Neg_F1: 0.942
```

Model Evaluation Metrics

- Precision: SVM showed slightly higher precision in identifying fake news compared to Logistic Regression, which is crucial for applications where falsely labeling genuine news as fake could be particularly damaging.

- Recall: Logistic Regression had balanced recall, making it a reliable choice if both false positives and false negatives carry similar cost.

- F1-Score: Random Forest had the highest F1-score, indicating a better balance between precision and recall, which is ideal for a balanced dataset.

Overall Comparison

- Performance: Random Forest outperformed the other models in accuracy and F1-score, making it the most effective model for this dataset. When some additional quantitative linguistic features were added, this improved to out-score all of our other models.

- Application Suitability: For scenarios where fast, real-time prediction is needed, Logistic Regression might be more suitable due to its simpler calculations. However, for critical applications requiring high accuracy and reliability, Random Forest is preferable despite its computational cost.

Deep Learning

In our project, we explored not only traditional feature sets and Machine Learning models but also delved into Deep Learning models. In particular, we worked with two different Deep Learning models: LSTM (Long Short-Term Memory) and distilBERT(a distilled version of BERT). Due to computational resource limitations, we only used the title column of our dataset for both the Deep Learning models.

For the LSTM model, we used TF-IDF to vectorize our words. We then passed these word vectors through the LSTM model with 3 layers and ran 3 epochs of this model. The details of the layers of our LSTM and the classification report can be seen in the screenshot below. The accuracy we achieved was 50.51%.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 1000, 64)          64000

 lstm (LSTM)                 (None, 100)               66000

 dense (Dense)               (None, 1)                 101

=================================================================
Total params: 130101 (508.21 KB)
Trainable params: 130101 (508.21 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

	precision	recall	f1-score	support
0	0.00	0.00	0.00	7081
1	0.51	1.00	0.67	7227
accuracy			0.51	14308
macro avg	0.25	0.50	0.34	14308
weighted avg	0.26	0.51	0.34	14308

Our second deep learning model was a pre-trained distilBERT, a version of the BERT model that retains most of BERT's predictive power but is faster and more computationally-efficient. This model was pretrained on a large corpus of data. This allows us to leverage existing knowledge without taking up too many computational resources. First, we created BERT word embeddings for the title column of our data and then passed these embeddings to a Logistic Regression Classifier. The classification report for this model is shown below.

	precision	recall	f1-score	support
0	0.91	0.90	0.90	7081
1	0.90	0.91	0.91	7227
accuracy			0.91	14308
macro avg	0.91	0.91	0.91	14308
weighted avg	0.91	0.91	0.91	14308

Our LSTM model resulted in much lower accuracy compared to our feature based and machine learning models. This lower accuracy can be attributed to the 'data-hungry' nature of Deep Learning models. Deep Learning models usually require large amounts of data to be able to learn complex patterns efficiently. Our dataset did not have a sufficient number of observations for the model to effectively learn and generalize.

To address this limitation of Deep Learning models, we chose to use distilBERT's embeddings to vectorize our text and then passed these embeddings through a Machine Learning model (Logistic Regression). This seems to give us the best results because BERT embeddings take into account the context and the semantic meaning of the words into account.

Final Results Summary

NLTK Naive Bayes Classifier

Test #	FEATURES	Prec	Recall	F1	Spec.	Mean Acc.
1	min. processed (title only)	0.924	0.838	0.879	0.930	88.1%
2	preprocessed unigram (title only)	0.853	0.833	0.842	0.855	84.0%
3	title unigram + text unigram (combined)	0.750	0.852	0.798	0.713	78.16%
4	* * Baseline * * title unigram (15% vocab)	0.880	0.849	0.864	0.883	**86.5%**
5	unigram + bigram (title only)	0.880	0.849	0.864	0.883	86.5%

6	title unigrams + text bigrams	0.880	0.849	0.864	0.883	86.5%
7	unigrams + trigrams (title only)	0.880	0.849	0.864	0.883	86.5%
8	title unigrams + opinion lexicon	0.881	0.845	0.863	0.885	86.5%
9	title unigrams + quant. linguistic features	0.912	0.843	0.876	0.918	87.9%
10	unigrams + bigrams + sent. features + quant. ling. features	0.910	0.840	0.874	0.917	87.8%
11	unigrams + bigrams + POS tag features	0.85	0.96	0.90	0.82	**89%**
SciKit Learn Classifiers **Features:** title unigrams + quant. linguistic						
12	Linear SVC	0.91	0.90	0.90	0.91	90%
	Gaussian Naive Bayes	0.83	0.73	0.78	0.85	79%
	Logistic Regression	0.91	0.90	0.91	0.91	**91%**
SciKit Learn Classifiers **Features:** TF-IDF Vectorization						
13	Logistic Regression	0.80	0.77	0.79	0.80	78%
14	SVM	0.80	0.77	0.79	0.81	79%
15	Random Forest	0.84	0.89	0.87	0.83	**86%**
	+ quantitative linguistic features	0.939	0.951	0.945	0.936	**94.4%**

Deep Learning Classifiers						
16	LSTM	0.51	1.00	0.67	0.00	**50.51%**
13	BERT + Logistic Regression	0.90	0.91	0.91	0.82	**91%**

Lessons Learned

We were surprised by how powerful a very basic feature like unigrams proved to be. We were also surprised by how little improvement adding bigram and trigram features yielded. This was presumably because these featuresets were always significantly more sparse than unigram features.

Simple quantitative features that produced a dense matrix of continuous values also proved very powerful, whereas qualitative feature counts, such as sentiment, were not as reliable. This may again be due to sparsity and might improve with a large enough lexicon.

TF-IDF vectorization was equally or more performant than Boolean Bag of Words unigram features, especially in combination with quantitative linguistic features.

We were somewhat puzzled by the fact that unigram features from the headline (‘title’ field) alone were so much more strongly predictive than the article body text (‘text’) with or without the headlines; intuitively it seems like more data and more features should yield better models. However, we speculate that this is because headlines are essentially human-coded “most important features” for an article. Keywords, as well as some stylistic features (e.g. writing words in all caps) that are strongly related to content will be concentrated in the headlines, whereas they will be more dilute in any dataset that includes the text body, creating noise and rendering the model less powerful.

This medium-sized dataset presented the opposite problem for data-hungry deep learning models (e.g. LSTM), which struggled to achieve scores better than chance.

This project gave all of us a much deeper understanding of Natural Language Processing tasks in a real-world classification scenario and gave us an opportunity to really use the tools we have been working with this semester, both in this class and outside of it.