Final Report: Fake News Detection Using Naive Bayes’

Literature Review/Description of the Problem

“Now fake news is more popular and widely spread through social media than mainstream media (Balmas, 2014). Being extensively used for confusing and persuading online users with biased facts, fake news has become the major concern for both industry and academia.” [5]. With the results of the recent 2024 election passing Americans by, the focus on identifying fake news has increased exponentially. While fake political materials are a main focus of industries at the moment, health-related content, financial ploys, and reputation pieces regarding innocent civilians are all of primary concern [4]. As of September 2024, Pew Research Center found that 86% of U.S. adults say they at least sometimes get news from a smartphone, computer or tablet, including 57% who say they do so often [1]. Considering the rising rate of Americans gaining news via the internet and the real world effects of fake news on the results of, notably, the 2016, 2020, and 2024 elections, it’s imperative that industries begin looking for solutions.

Using artificial intelligence (AI) for such an issue has proven to be effective as multiple market solutions exist for this exact issue including Logically, Full Fact, Fabula AI, etc. [3]. However, the underlying machine learning methods in each of these products differs. Professor at University of Bangladesh, M. F. Mridha and their colleagues, tried an extensive list of machine learning and deep learning models on a dataset of about 20,000 fake and real news articles and headlines. Their list included, but was not limited to, Support Vector Machine, Naive Bayes’, K-Nearest Neighbor, Random Forest, Convolutional Neural Network, and Recurrent Neural Network. Through their trials, they found ~98% accuracy with CNN and Deep CNN making their overall conclusion that deep learning models are better than machine learning models at fake news detection [4]. However, this motivated me to see if I could make a machine learning model like Naive Bayes’ compare to some degree to the 98% Mridha found. My goal was to get a better classification accuracy than researchers Mykhailo Granik and Volodymyr Mesyura at Vinnytsia National Technical University who achieved 74% on their test set containing

Facebook news posts. Within their work, they loaded the articles, filtered them based on presence of content/relevant labels, separated data in the training, validation, and test datasets, trained the classic Naive Bayes' classifier, tested it, and outputted the accuracy evaluation. Ultimately, Granik and Mesyura concluded that even a simple Naive Bayes' classifier does a decent job at fake news detection and should be further explored [2].

Methods

Firstly, I downloaded my desired fake news dataset from:
https://www.kaggle.com/datasets/saurabhshahane/fake-news-classification

This specific dataset contains 72,134 labelled real (1) and fake (0) headlines and articles.

Examining the Data

Before starting with building the model, I wanted to examine the quality of the data and whether or not it had to be "cleaned". Upon examination, it did have extra spaces, special characters, and some words that weren't part of the English language. I also saw in other fake news detection work the concept of lemmatizing, or reducing the words in a large body of text to its root while maintaining it's meaning for easier comprehension. So I created a clean_text() function that could at least remove extra spaces and special characters. At one point I also added lemmatization and Spanish, French, and German languages for processing, but these two additions made the accuracy of the model worse somehow.

Figuring out Which Naive Bayes' Model to use

I narrowed the models down to the ones that sklearn.naive_bayes offers which are: Bernoulli, Multinomial, Complement, Gaussian, _Base, Categorical, _DiscreteBase. I further narrowed it down to Bernoulli, Multinomial, Complement, and Gaussian because these four are the most realistic models to use for this application.

I created a method called find_best_nb_model(X, y) that trained the four models on a small sample of the original dataset (110 records instead of the original 72,134 records). It performed cross-validation via StratifiedKFold(n_splits=10, shuffle=True, random_state=42) and hyperparameterized each model:

Bernoulli Naive Bayes’

- 'alpha': [0.01, 0.1, 1, 10] - I chose to try different alphas as this prevents probabilities from becoming zero, especially for features not observed in the training set. I tested different values because it could help balance bias-variance trade-off and optimize model performance.
- 'fit_prior': [True, False] - I chose to try different fit_priors as it helps determine whether the class prior probabilities are learned from the data (True) or set to uniform (False). In datasets with imbalanced classes, allowing the model to learn priors can improve predictions.

Multinomial Naive Bayes’

- 'alpha': [0.01, 0.1, 1, 10] - I chose to try different alphas as this prevents probabilities from becoming zero, especially for features not observed in the training set. I tested different values because it could help balance bias-variance trade-off and optimize model performance.
- 'fit_prior': [True, False] - I chose to try different fit_priors as it helps determine whether the class prior probabilities are learned from the data (True) or set to uniform (False). In datasets with imbalanced classes, allowing the model to learn priors can improve predictions.

Complement Naive Bayes’

- alpha': [0.01, 0.1, 1, 10] - I chose to try different alphas as this prevents probabilities from becoming zero, especially for features not observed in the training set. I tested different values because it could help balance bias-variance trade-off and optimize model performance.
- 'fit_prior': [True, False] - I chose to try different fit_priors as it helps determine whether the class prior probabilities are learned from the data (True) or set to uniform (False). In datasets with imbalanced classes, allowing the model to learn priors can improve predictions.
- 'norm': [True, False] - I chose to try different norms as it normalizes the feature weights to ensure balanced contributions across features. I think it’s

particularly helpful for this dataset since it could contain highly variable feature magnitudes.

Gaussian Naive Bayes'

- 'var_smoothing': [1e-10, 1e-9, 1e-8, 1e-7] - I chose to try different var_smoothings as it helps avoid overfitting and ensures numerical stability. Overall, this was kind of a last ditch effort to make Gaussian NB work better

For each model, I output mean AUROC, standard deviation, best parameters, best cross-validated accuracy, precision, recall, confusion matrix, and classification report.

Moving Forward With the Ideal Naive Bayes' Model

After finding the ideal model, I trained it using its' ideal parameters on 80% of the entire dataset so I could save the remaining 20% as a test set. I did this so that I could see the true accuracy of my model on data it hasn't seen yet.

For both the training and testing I output mean AUROC, accuracy, precision, recall, confusion matrix, and classification report.

Results

Finding the Best Naive Bayes' Model

Gaussian Naive Bayes' performed significantly worse than Multinomial, Complement, or Bernoulli at a mean AUROC of .69. Meanwhile, Multinomial and Complement Naive Bayes' performed similarly with a mean AUROC of .81. Bernoulli performed the best with a mean AUROC of .82. I found it important to note the standard deviation in each of the models where Gaussian had the least deviation at .07 and Bernoulli had the most at .1. Furthermore, Gaussian had a low precision and recall (both at .72) compared to the other three which landed at .79 for both metrics, except Bernoulli which had a .87 recall. This points to Bernoulli Naive Bayes' being an ideal model for fake news detection out of the four I tested.

```
Gaussian NB:
Best Parameters: {'var_smoothing': 1e-10}
Best Cross-Validated Accuracy: 0.6947
Mean AUROC: 0.6894
Precision Score: 0.7258
Recall Score: 0.7258
Accuracy Standard Deviation: 0.0657
Confusion Matrix:
[[32 17]
 [17 45]]
Classification Report:
              precision    recall  f1-score   support

           0       0.65      0.65      0.65        49
           1       0.73      0.73      0.73        62

    accuracy                           0.69       111
   macro avg       0.69      0.69      0.69       111
weighted avg       0.69      0.69      0.69       111
```

```
Complement NB:
Best Parameters: {'alpha': 1, 'fit_prior': True, 'norm': True}
Best Cross-Validated Accuracy: 0.7667
Mean AUROC: 0.8147
Precision Score: 0.7903
Recall Score: 0.7903
Accuracy Standard Deviation: 0.0889
Confusion Matrix:
[[36 13]
 [13 49]]
Classification Report:
              precision    recall  f1-score   support

           0       0.73      0.73      0.73        49
           1       0.79      0.79      0.79        62

    accuracy                           0.77       111
   macro avg       0.76      0.76      0.76       111
weighted avg       0.77      0.77      0.77       111
```

```
Multinomial NB:
Best Parameters: {'alpha': 1, 'fit_prior': False}
Best Cross-Validated Accuracy: 0.7667
Mean AUROC: 0.8140
Precision Score: 0.7903
Recall Score: 0.7903
Accuracy Standard Deviation: 0.0889
Confusion Matrix:
[[36 13]
 [13 49]]
Classification Report:
              precision    recall  f1-score   support

           0       0.73      0.73      0.73        49
           1       0.79      0.79      0.79        62

    accuracy                           0.77       111
   macro avg       0.76      0.76      0.76       111
weighted avg       0.77      0.77      0.77       111
```

```
Bernoulli NB:
Best Parameters: {'alpha': 0.01, 'binarize': 0.0, 'fit_prior': True}
Best Cross-Validated Accuracy: 0.8008
Mean AUROC: 0.8203
Precision Score: 0.7941
Recall Score: 0.8710
Accuracy Standard Deviation: 0.1068
Confusion Matrix:
[[35 14]
 [ 8 54]]
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.71      0.76        49
           1       0.79      0.87      0.83        62

    accuracy                           0.80       111
   macro avg       0.80      0.79      0.80       111
weighted avg       0.80      0.80      0.80       111
```

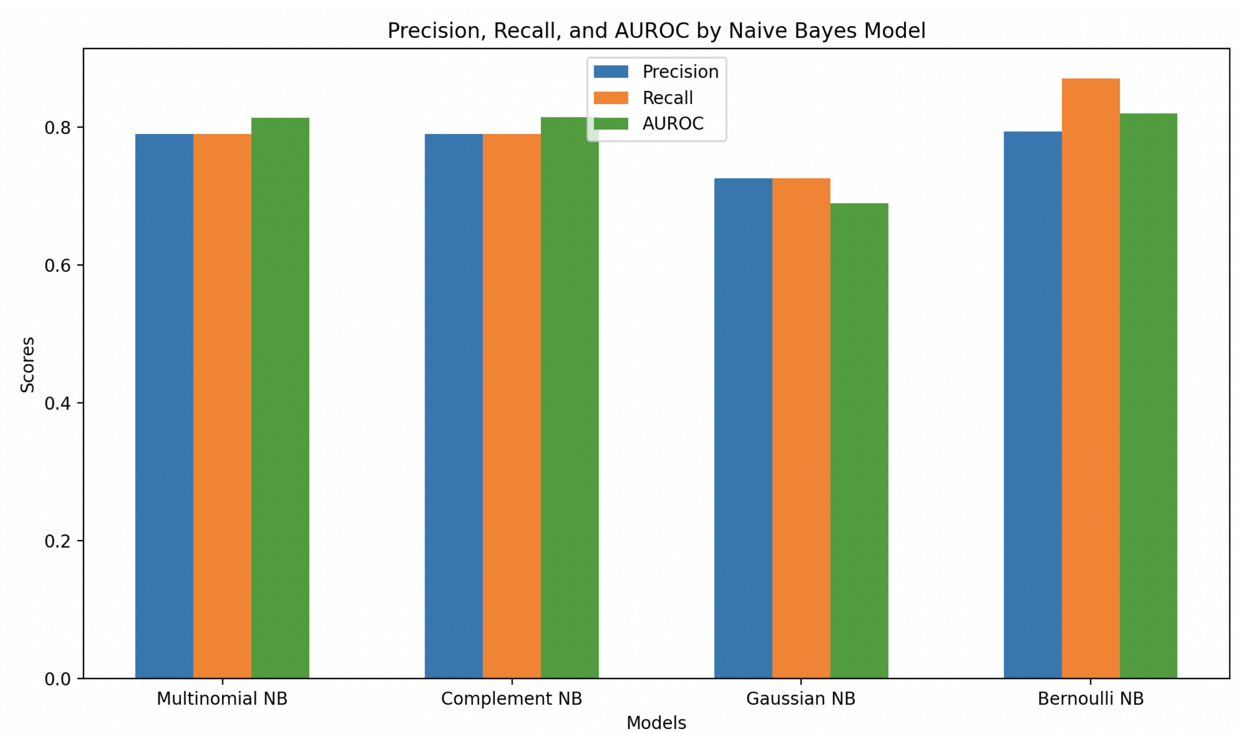
Training and Testing Bernoulli Naive Bayes’

The training and testing sets of Bernoulli Naive Bayes’ performed well with both having a mean AUROC of ~.9 and precision and recall in the ranges of .77-.87.

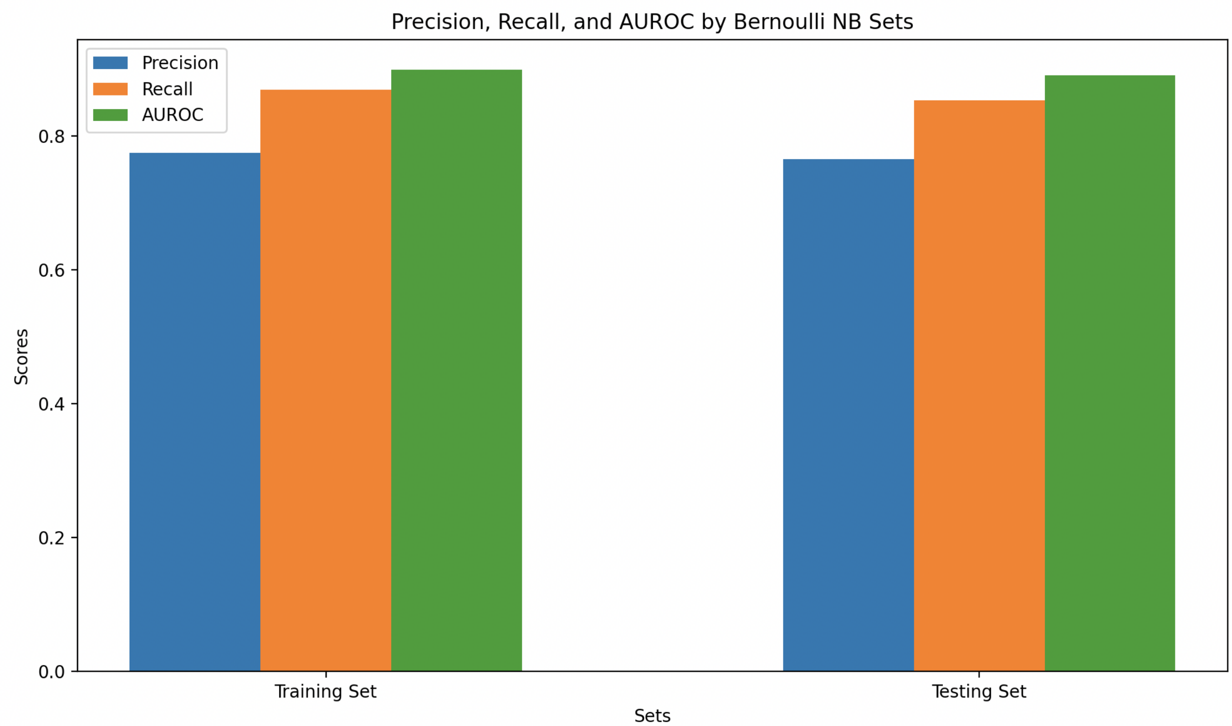
```
Bernoulli NB Training set:
Mean AUROC: 0.8994
Accuracy: 0.8031
Precision: 0.7752
Recall: 0.8692
Confusion Matrix:
[[20541  7481]
 [ 3884 25801]]
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.73      0.78     28022
           1       0.78      0.87      0.82     29685

    accuracy                           0.80     57707
   macro avg       0.81      0.80      0.80     57707
weighted avg       0.81      0.80      0.80     57707
```

```
Bernoulli NB Test set:
Mean AUROC: 0.8908
Accuracy: 0.7900
Precision: 0.7653
Recall: 0.8535
Confusion Matrix:
[[5063 1943]
 [1087 6334]]
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.72      0.77      7006
           1       0.77      0.85      0.81      7421

    accuracy                           0.79     14427
   macro avg       0.79      0.79      0.79     14427
weighted avg       0.79      0.79      0.79     14427
```

Discussion



I didn’t project Gaussian doing super well in this application as it’s meant to do better probabilistic predictions on continuous data like income, age, etc. and that’s precisely what was reflected within the data. What’s interesting is Complement and Multinomial performing similarly. While Complement Naive Bayes’ is an offshoot of Multinomial, coming out with the exact same mean AUROC was definitely not expected on my end. Regardless, they performed very well and this was to be expected as they’re great at calculating probabilities on high dimensional spaces like blocks of text that have meaning.



Bernoulli Naive Bayes’ doing the best of the 4 models is not a surprise as it’s known for calculating probabilities for high dimensional spaces based on binary labels like fake (0) and real (1) in this case. What’s interesting is the disproportionately high recall in all three uses. While precision and recall are typically balanced like what we can observe in Multinomial, Complement, and Gaussian, Bernoulli’s precision was consistently in the .77-.78 range while recall was abnormally high between .85-.87 indicating that Bernoulli rarely missed true positive cases. I’m not quite sure why this is, but one possibility is that there were more real articles/headlines than fake ones in the dataset leading to more training/testing on real data than fake data. However, it is fair to say that this is a lofty assumption considering the recalls of the other models being very close to their precision.

One notable feature of the entire project is the cleaning of data. As seen in the results section, I made no discernment between unclean and clean results. This is because it made no difference in my trials. When I cleaned the smaller sample, it took little to no time, but cleaning the larger dataset took 1.5 hours. Once cleaned and stored in a separate .csv, I tested it and came out with the same results as the unclean version down to the decimal point. While I'm still not too sure why the cleaning of the data yielded the same results as the unclean version, I would guess its because the model is smart enough to ignore white spaces, words from other languages, complex vocabulary, and special characters.

Conclusion

Overall, while deep learning models like CNN might be better at fake news detection, I did achieve my goal of getting a higher classification accuracy than 74%. With Bernoulli Naive Bayes' I did achieve 79% (pulled from my final test set). Although this accuracy is lower than the 98% achieved by CNNs, it demonstrates the potential of simpler classifiers and highlights the inherent complexity of fake news detection.

Works Cited

[1] Sara Atske. 2024. News Platform Fact Sheet. *Pew Research Center*. Retrieved November 20, 2024 from https://www.pewresearch.org/journalism/fact-sheet/news-platform-fact-sheet/

[2] Mykhailo Granik and Volodymyr Mesyura. 2017. Fake News Detection Using Naive Bayes Classifier. *Ieee.org*. Retrieved November 20, 2024 from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8100379

[3] Sebastiaan Lans. 2021. 13 AI-Powered Tools for Fighting Fake News - The Trusted Web. *The Trusted Web*. Retrieved November 20, 2024 from https://thetrustedweb.org/ai-powered-tools-for-fighting-fake-news/

[4] M F Mridha, Ashfia Jannat Keya, Md Abdul Hamid, Muhammad Mostafa Monowar, and Md Saifur Rahman. 2021. A Comprehensive Review on Fake News

Detection With Deep Learning. *IEEE Access* 9, (January 2021), 156151–156170. DOI:https://doi.org/10.1109/access.2021.3129329

[5] Xichen Zhang and Ali A. Ghorbani. 2020. An overview of online fake news: Characterization, detection, and discussion. *Information Processing & Management* 57, 2 (March 2020), 102025. DOI:https://doi.org/10.1016/j.ipm.2019.03.004