Mark Snyder
Treyson Grange
CS 5830
Project 5
Slides
Repo

Part I: Introduction

In this project, we explored the Naive Bayes Classifier, and used it to predict phishing URLs based on the Web Page Phishing dataset. This model and project is important, as we progress in the digital age, we need to protect ourselves and our data online, and defend against ransomware.

Part II: Dataset

The dataset comprises features extracted from URLs along with their corresponding labels indicating whether the URL is associated with phishing (labeled as 1) or legitimate (labeled as 0). All features in this dataset follow the same format, n_item, or number of items. The dataset tracks 17 different characters, and counts how often they occur inside the URL. The dataset also reports the length of the URL. For reasons of safety and in an attempt to not spread these links, the dataset does not include any of the actual URLs. Out of 100077 entries, 63715 of the URLs were not phishing, and 36362 were, giving us a ratio of about 20:11. This ratio is decent, however it indicates a slight class imbalance between the two.

Part III: Analysis Technique

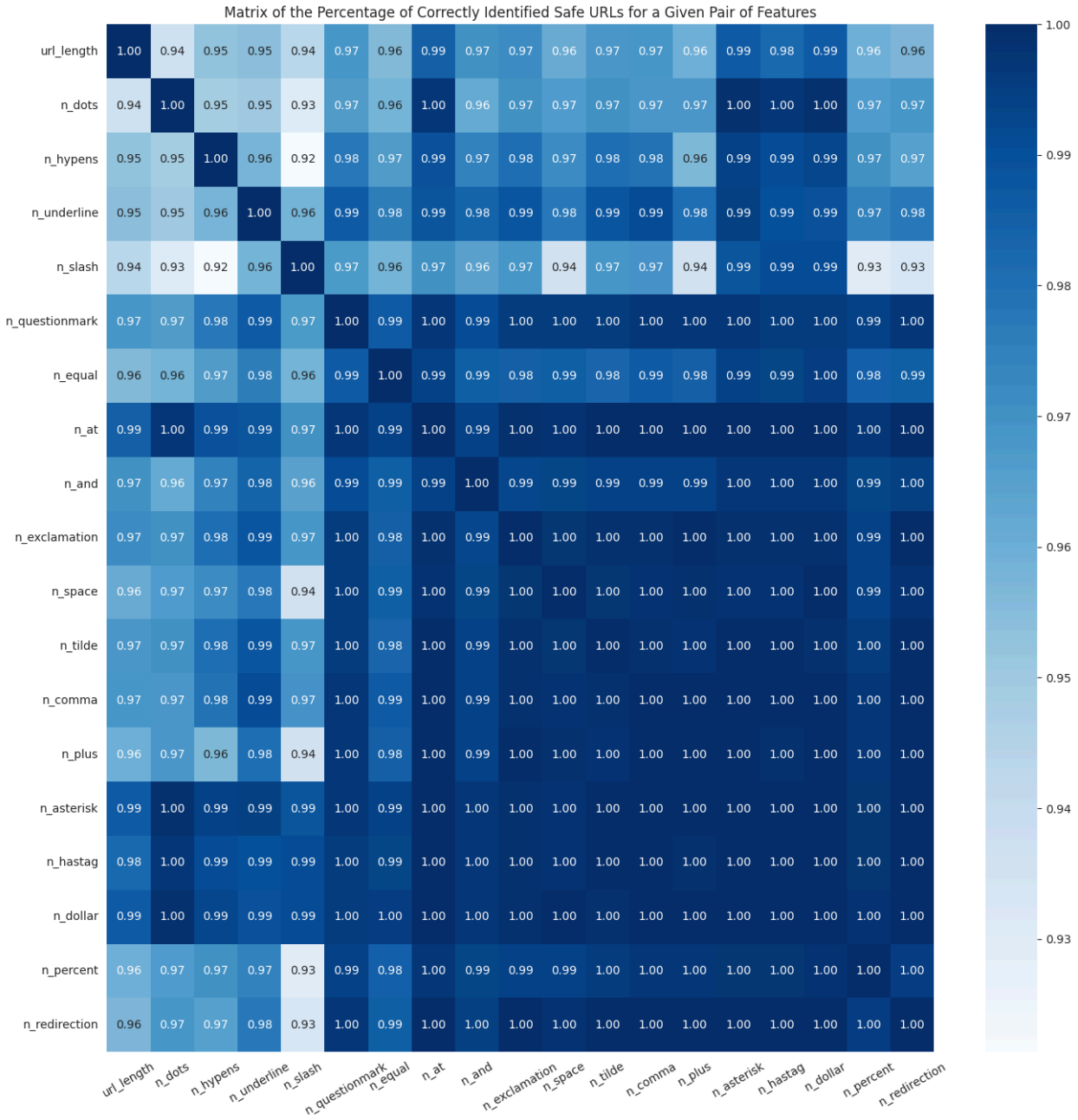
In this project we started by creating our dataframe, and splitting the features and our target. Obviously our target will be whether or not the URL is associated with a phishing scheme. To best test our data, we would define a function that runs the Naive Bayes Gaussian classifier given certain attributes. Within this function we would set testing and training data, and fit our model with the former data. Using our model, we would then predict on the `X_Test` . Given our new predictions, we would calculate precision, recall and f score.

Using this function we can test every 2 column combination quickly. We double looped over our columns and we ran our function for each of them, being sure to note the precision, recall, and f1 score for both guessing positive predictions, and negative predictions, at each combination. This left us with 6 18x18 grids for scores. Using a custom heatmap function, we plot heatmaps for each matrix. See below.

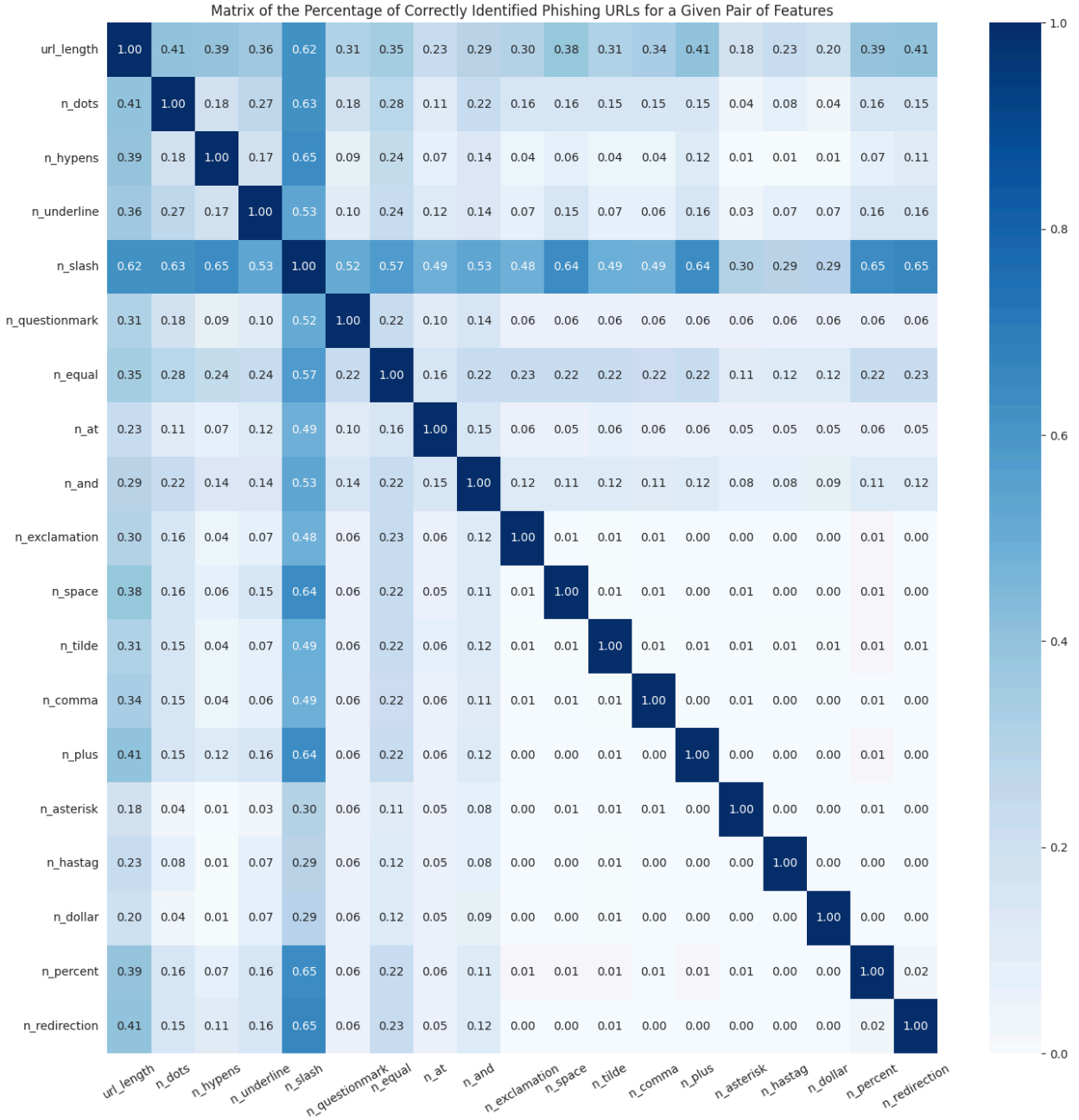
Part IV: Results

Part IV.a: Gaussian Naive Bayes Classifier

Using Gaussian Naive Bayes classifiers for every pair of attributes and analyzing the precision and recall of predicted results, we get the following two comparison matrices. The first shows the likelihood that a pair of attributes will accurately predict safe URLs, with displayed values at 1.00 yielding zero false positives. The second matrix shows the likelihood that a pair of attributes will accurately predict phishing URLs.

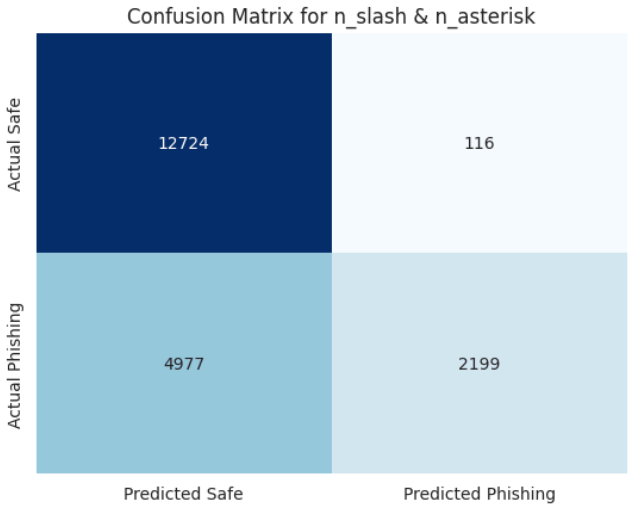


As you can see, the two matrices look inverted. This means that pairs of attributes that scored well for identifying safe URLs generally didn't perform well for identifying phishing URLs, and vice versa. This makes identifying a pair of "best" attributes tricky. Plenty of pairs rarely, if ever, incorrectly identify false positives. Whereas only a handful of pairs correctly identify phishing URLs, but also have the most false positives out of all the pairs. (The number of redirects paired with the number of slashes scored the best for identifying true positives; the number of slashes was the single best identifier in the set.)

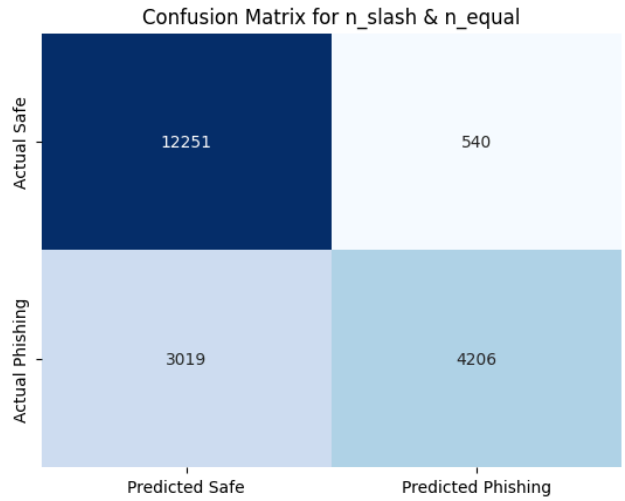


Which do you value more? If you aim to identify the maximum number of phishing URLs, which on paper sounds ideal, as many as 1/10 of safe URLs will be identified as phishing (you would catch 2/3 of all phishing links, however). If you falsely identify too many, people will become desensitized to the warnings and end up ignoring them all together, so you can't do that. Avoiding false positives in their entirety means you'll never catch a meaningful number of phishing URLs. A balance has to be struck.

There are two sets of pairs that we identified to score fairly well for both. Which of the two is better depends on your tolerance for false positives.



The first is more conservative for false positives. Taking the number of slashes and number of asterisks in a URL yields a correct negative (safe URL) identification rate of 0.985 and a correct positive (phishing URL) identification rate of 0.30. The first confusion matrix shows a random sampling of 20% of the dataset to see how accurately these attributes perform.



The second is the more aggressive at finding phishing URLs. Taking the number of slashes and number of equal signs in a URL yields a correct negative (safe URL) identification rate of 0.958 and a correct positive (phishing URL) identification rate of 0.568. The second confusion matrix shows a random sampling of 20% of the dataset to see how accurately these attributes perform. This is a massive increase in correctly identified phishing URLs and a moderate, but not too severe, decrease in the number of false positives.

Opinion time: If I had to choose between a more conservative and a more aggressive approach, I'd choose the latter. Less than 1/20 URLs are incorrectly identified as phishing, but more than 11/20 URLs are correctly identified as phishing, which is approaching the best we can get using this dataset and analysis of URLs alone. Considering how basic yet successful an approach this is for identifying phishing in the modern era of sophisticated phishing identification systems, I think this is quite good.