INFX 574: Assignment 04

Casey Pham, Mike Stepanovic

Overview

In this assignment, we examined three machine learning algorithms for spam detection in 495 emails (142 ham, 353 spam). The emails were collected from an ancient email account on Yahoo.com, and split into six features: Subject, Body, From: (Name), From: (Address), From: (Type), and Importance. Each case was labeled with spam or not spam based on the email folder it was collected from.

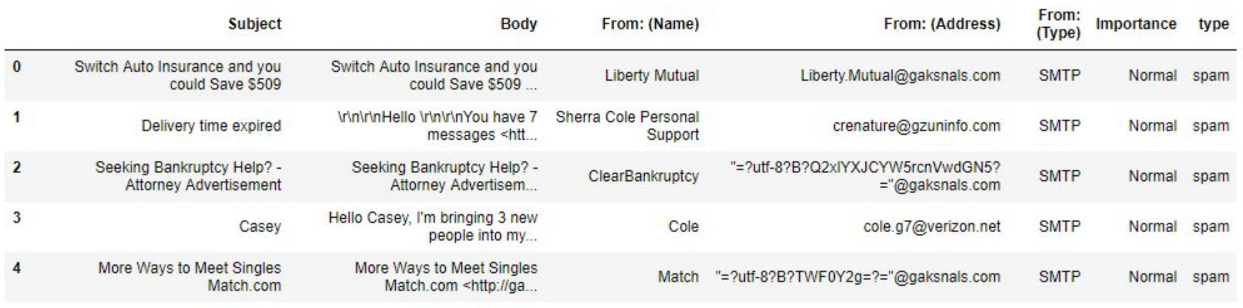
To build the classifier, the feature set was first tokenized into 1-grams and then vectorized for input into the model. Naive Bayes, Random Forest, and Logistic Regression algorithms were then trained according to a 2:1 train-test split. These algorithms were chosen based on their contrasting approaches toward classification. After fitting each model, ROC curves were produced and each model was evaluated according to AUC. The final AUCs were 0.99 for Naive Bayes, 0.97 for the Random Forest algorithm, and 0.99 for Logistic Regression.

Classification Task

Spam detection is messy because email data is unstructured. A binary classification (spam vs. ham) is being predicted based on inherent qualities of the text in the body or header of an email. However, spam is often easy for humans to detect. It either mercilessly advertises for superficial or eccentric products, is intended to steal information, or to bring awareness to political initiatives. Because spam contains unusual words (either product names or lots of URL HTML code), it is a good test for classification models that are able to estimate these differences.

Dataset Sample

The dataset contained 7 columns and 495 rows:



	Subject	Body	From: (Name)	From: (Address)	From: (Type)	Importance	type
0	Switch Auto Insurance and you could Save $509	Switch Auto Insurance and you could Save $509 ...	Liberty Mutual	Liberty.Mutual@gaksnals.com	SMTP	Normal	spam
1	Delivery time expired	\r\n\r\nHello \r\n\r\nYou have 7 messages <htt...	Sherra Cole Personal Support	crenature@gzuninfo.com	SMTP	Normal	spam
2	Seeking Bankruptcy Help? - Attorney Advertisement	Seeking Bankruptcy Help? - Attorney Advertisem...	ClearBankruptcy	"=?utf-8?B?Q2xlYXJCYW5rcnVwdGN5?="@gaksnals.com	SMTP	Normal	spam
3	Casey	Hello Casey, I'm bringing 3 new people into my...	Cole	cole.g7@verizon.net	SMTP	Normal	spam
4	More Ways to Meet Singles Match.com	More Ways to Meet Singles Match.com <http://ga...	Match	"=?utf-8?B?TWF0Y2g=?="@gaksnals.com	SMTP	Normal	spam

Figure 01: Head of email data

Procedure

Email were obtained from a Yahoo! Email account. The account was linked to Microsoft Outlook, and then exported via Outlook, as Yahoo! does not have a native tool to export emails. Exported emails were split into the following columns: Subject, Body, From: (Name), From: (Address), From: (Type), To: (Name), To: (Address), To: (Type), CC: (Name), CC: (Address), CC: (Type), BCC: (Name), BCC: (Address), BCC: (Type), Billing Information, Categories, Importance, Mileage, and Sensitivity. Of these, only columns with content were kept and the To address removed, thus keeping only the original columns Subject, Body, From: (Name), From: (Address), From: (Type), and Importance. Spam and ham were obtained from separate export operations. Each set was then given the column Type, in which the emails were labeled spam and ham respectively. The two sets were then combined into a single CSV file "email.csv", which is used as the base dataset.

As the models are unable to handle strings as their input features, these needed to be converted into numerical values. We utilized Bags of Words to count and vectorize email strings, and did so for each feature. Each feature then had to be split into separate dataframes. This is because each feature, when converted to bags of words, is a matrix of the features and their bags of words. Creating a dataframe where each element is a matrix is impossible, so each desired feature into its own dataframe, and as necessary concatenating dataframes to combine features for use in the model.

In running the model, the desired feature sets were train test split using sklearn.model_selection.train_test_split, with a set random_state = 0, and test_size = 0.33 for consistency between models.

Models were fit and predicted using built-in functions of sklearn: sklearn.naive_bayes.MultinomialNB, sklearn.ensemble.RandomForestClassifier, and sklearn.linear_model.LogisticRegression. Mean accuracy scores were extraction from each model's score method, and ROC curves based off of each model's predict probability values of the test set, and then applied to a ROC curve using sklearn.metrics.roc_curve.

Results

Naive Bayes performed with 95.12% accuracy in predicting the testing data when trained based on Subject, Body, From: (Name), and From: (Address). The resulting Receiver Operating Characteristic (ROC) curve has an Area Under the Curve (AUC) of 0.99, shown in figure 02.

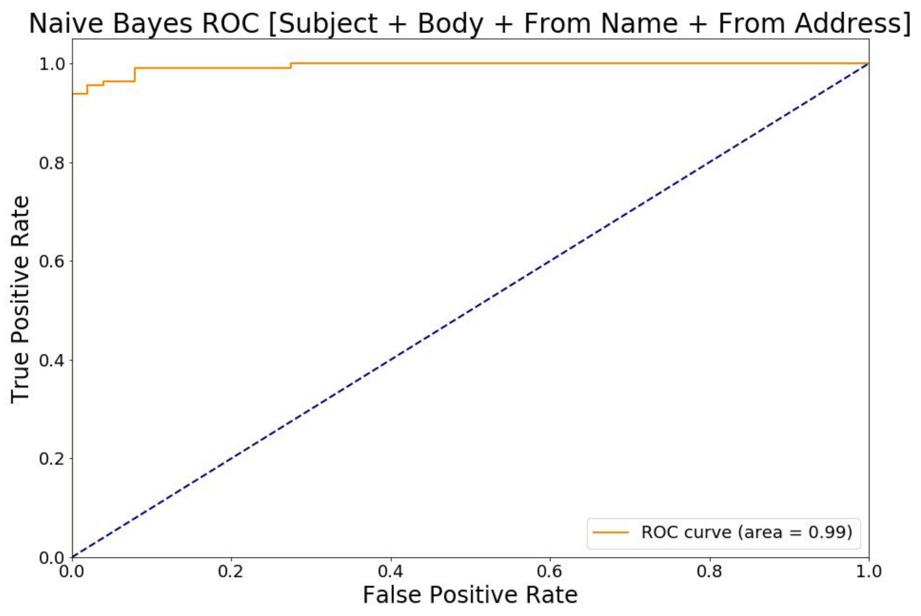


Figure 02: Naive Bayes ROC Curve

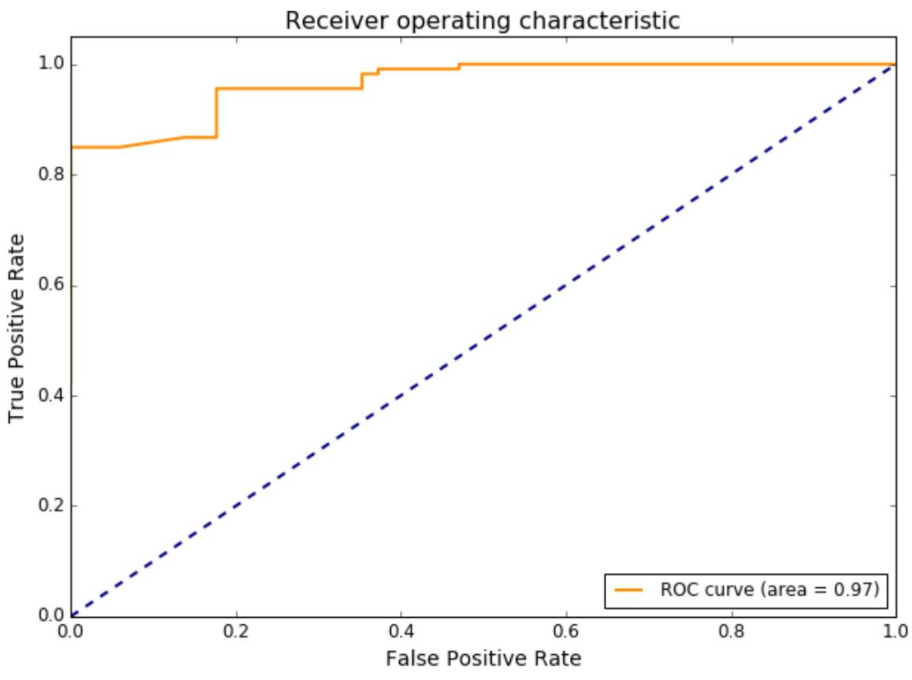


Figure 03: Random Forest ROC Curve

Random Forest Classification performed with an accuracy of 78.65% in predicting the testing data, trained based on the Subject and Body. The resulting ROC has an AUC of 0.97, shown in Figure 03.

Logistic Regression performed with an accuracy of 90.85% in predicting the testing data, trained based on the Subject and Body. The resulting ROC has an AUC of 0.99, shown in *Figure 04*.

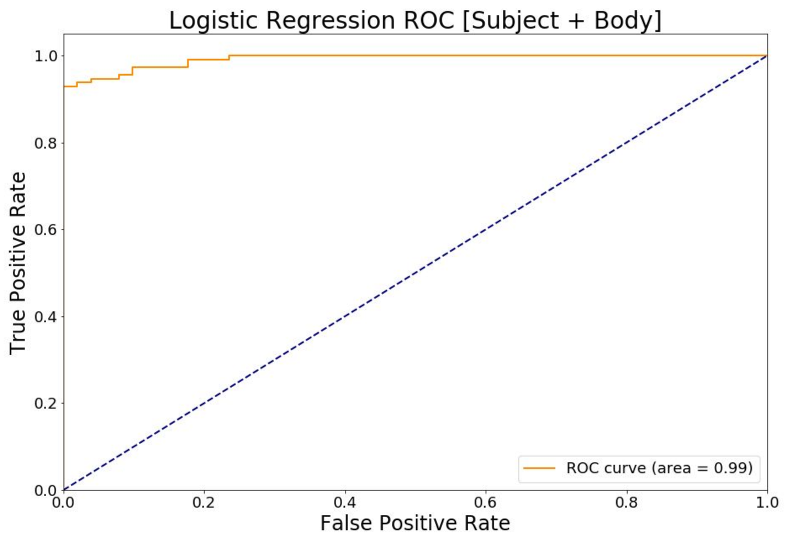


Figure 04: Logistic Regression ROC Curve

Comparing Classification Algorithms

Naive Bayes is a probabilistic classifier, applying Bayes' theorem with a naive assumption of independence between features[1]. This algorithm was chosen due to its simplicity, high performance, and popularity as a baseline algorithm for categorization. In this case we used Multinomial Naive Bayes, where feature vectors represent frequencies of events, in this case words, based on the bags of words extracted from emails.

Random Forest is an ensemble model that relies on bootstrap aggregation, which uses a different random subset of the original dataset for each model in the ensemble[2]. Random Forests grows many classification trees, each of which represents an independent classification model. Each tree's predictions represents a kind of "vote" for a particular class. On the whole, the "forest" ultimately decides on the class that receives the most total votes.

[1] Naive Bayes classifier. (n.d.). *Wikipedia*
[2] Breiman, L, & Cutler, A. (n.d.). Random Forests

Logistic Regression is a regression model, in this case used for a binary dependent variable[3]. Similar to Naive Bayes, it utilizes probabilities in prediction, estimated using a logistic function. Logistic Regression makes two assumptions: the conditional distribution y | x is a Bernoulli distribution, not Gaussian, as the dependent variable is binary, and predicted probabilities are restricted between (0, 1).

We chose this particular selection of algorithms mainly due to intellectual curiosity. We wanted to test a variety of families of classification algorithms, including linear models (logistic regression), ensemble decision trees (random forest), and Bayesian classifiers (naive bayes). However, all of these models have different strengths and weaknesses.

Logistic regression is a simpler model than random forests or naive bayes, and it also has low variance (0, 1). Like random forests, it can provide variable importance metrics in the form of parameter coefficients. However, the overall classification model is much easier to interpret (by ordering absolute values of coefficients) versus random forests (which can have many non-linear predictions). However, plain logistic regression is prone to overfitting when hundreds of levels of variables are included (as is the case in this text-based classification data). Random forests, as an ensemble classifier, are essentially impervious to this. Naive bayes has advantages over both classifiers. For one, it is computationally fast because it assumes independence of each parameter. But this can make it perform inaccurately when parameters are dependent. Each parameter in this spam detection dataset is mostly independent. Finally, because naive bayes works well with high dimensions (partially because it assumes they are all independent), it is easier to scale to text-based problems such as spam detection. Logistic regression, in contrast, might overfit a test sample.

Based on the results of this spam detection test (total AUC), the naive bayes performed the best, followed by the random forest, and finally the logistic classifier. Despite having a small sample size, the naive bayes model was able to detect specific text that distinguishes between spam and ham. We can assume that our features were independent and likely non-linear. We can also assume that levels within our features had very high variance, which stumped the wisdom of the random forest.

[3] Jurafsky, Daniel, & Martin, J. H. (2016). *Speech and Language Processing*

References

Naive Bayes classifier. (n.d.). *Wikipedia.* Retrieved from https://en.wikipedia.org/wiki/Naive_Bayes_classifier

Breiman, L, & Cutler, A. (n.d.). Random Forests. Retrieved from https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

Jurafsky, Daniel, & Martin, J. H. (2016). *Speech and Language Processing*. Retrieved from https://web.stanford.edu/~jurafsky/slp3/7.pdf