Spam Detection Report

INFO 371
Eric Ma & Victoria Juan

Introduction

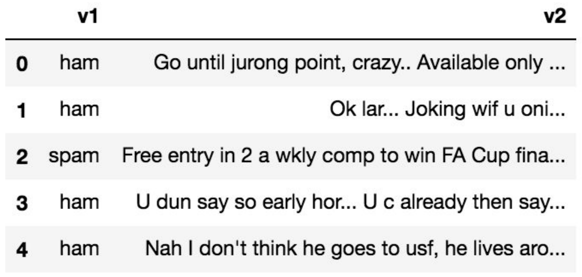
Spam detection is one of many applications of machine learning that benefits a vast majority of people nowadays. Without spam detectors, people will have to manually sort through emails in their inbox. That is both tedious and time consuming. The various algorithms used for spam detection is not 100% perfect. This means that there isn't any spam detectors out there that will 100% predict whether an email or message is a spam since there are always outliers. However, there are some that can get very close to perfect. In this paper, we will discuss three algorithms that we used as well as explaining which of the three is the best suited for spam detection.

Methods

For this experiment, the three algorithms used were: Multinomial Naive Bayes, Random Forests, and Logistic Regression. Before we used these algorithms, some data cleaning had to be done so that we can get an optimal result.

Data Preparation

We used a SMS Spam Collection dataset from Kaggle, which contains many messages that were already categorized into spam or ham (as shown in the picture below).

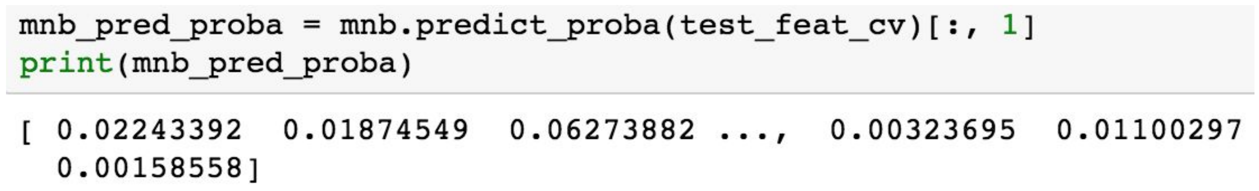


	v1	v2
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

Since we are dealing with binary outcomes, we converted the label to 0s and 1s where 0 corresponds to ham and 1 corresponds to spam. At the same time, we used a method called Term Frequency - Inverse Document Frequency (TF-IDF) to get the features for the dataset. TF-IDF is a method that associates a value or a weight with each individual unique word in the SMS message. The value of the weight ranges from 0 to 1, where values that are closer to 0 means that the word associated with the weight has lower frequency or is too common to distinguish between different documents. By contrast, words that are close to 1 have high frequency and are "significant" in that they only appear in a small number of documents. We also got rid of stop words since most stop words are not important when isolated.

Multinomial Naive Bayes

First, we used the Multinomial Naive Bayes classifier from the scikit-learn library. This is a specialized version of Naive Bayes that is designed for text classification and is typically used when the multiple occurrences of the words matter a lot in the classification problem. It is also a probabilistic classifier, therefore it can calculate the probability of each category using Bayes theorem. The classifier will output the highest probability from the different categories, in this case it’s either spam or non-spam email.



```python
mnb_pred_proba = mnb.predict_proba(test_feat_cv)[:, 1]
print(mnb_pred_proba)
```

```
[ 0.02243392  0.01874549  0.06273882 ...,  0.00323695  0.01100297
  0.00158558]
```

Random Forest

Next, we used the Random Forest Classifier. We thought that Random Forest Classifier would do well for this classification problem because Random Forests is just a combination of decisions trees merged together. Since a single decision tree have a tendency to overfit a dataset, by using a tree (or a forest) that consists of multiple decision trees, it will avoid overfitting and therefore, get a more accurate and stable prediction.

Logistic Regression

Finally, the last algorithm we used was logistic regression. We chose this algorithm because the dependent variable is binary and dichotomous, which fits perfectly for this problem since we are dealing with binary outcomes.

Performance Metrics

After cleaning up the data and identifying the features and labels of the dataset, we fit each algorithm to the training dataset. We then made predictions on the features of the testing dataset and create a confusion matrix which allows us to calculate AUC (Area Under Curve), accuracy, precision, and specificity.

The reason why we chose those metrics to evaluate the performance of our algorithms is as follows:

Area Under the Curve (AUC)

The AUC score is derived from the ROC curve where the curve represents a comparison between the false positive rate to the true positive rate for all possible thresholds. Since we want to minimize false-positive value, an algorithm that does well will have the true-positive rate to be

1 for all false-positive rate (AUC would thus be 1). This means that the algorithm correctly guessed that a spam message is spam. If it's lower than 1, it means that there are some messages that were falsely predicted to be spam, which is not what we want.

Accuracy

The accuracy score shows how accurate the predicted values were when compared to the actual values. In this experiment, it represents the percentage of SMS messages which the model accurately predicted to be spam or ham. This is important because for instance, if the model only made correct predictions about 50% of the time, then it is the same as just flipping a coin and therefore, making this investment would be a waste of money.

Precision

We chose precision because the goal of this experiment is to find a model that minimizes the False-Positive value in the confusion matrix. The False-Positive value represents messages that the are actually ham (not spam) but the model predicts spam. This is bad because we don't want potentially important emails to be placed in the spam box. The precision score formula is shown below:

$$\text{Precision} = \frac{tp}{tp + fp}$$

If the precision score for the algorithm is high, it indicates that the false-positive value is low and therefore, when the algorithm predicts spam, it will most likely to be spam.

Specificity

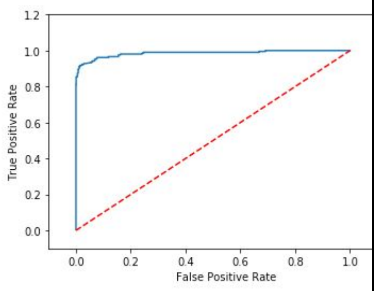
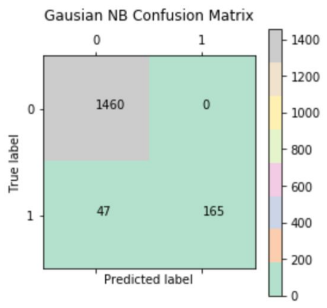
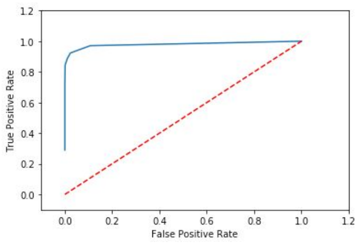
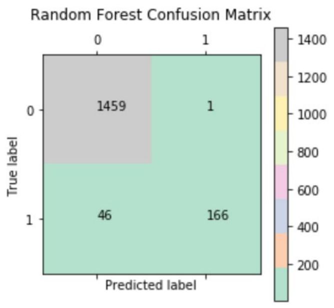
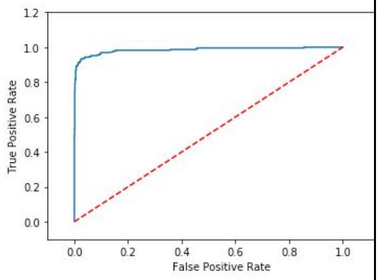
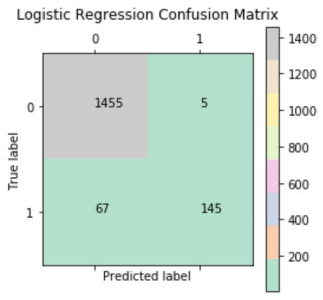
Similar to Precision, specificity was chosen because high specificity value means low false-positive value (as shown below).

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \text{X} 100$$

In the formula, True Negatives represent messages that the model accurately predicted to be ham (not spam). If the specificity score is high, we can assume that the false-positive value is low and therefore, the algorithm avoids mislabelling legitimate messages as spam.

Result

The table below shows the result of the metrics that were calculated for each of the algorithms:

	ROC Curve	AUC	Accuracy	Confusion Matrix	Precision	Specificit y
Multinomial Naive Bayes		~0.989	~0.971		~1.0	~1.0
Random Forest		~0.985	~0.971		~0.989	~0.999
Logistic Regression		~0.991	~0.957		~0.987	~0.999

Interpretation and Conclusion

From the table, we see that the calculated metrics were very similar between the three algorithms in that the values are just a little under 1. We also note that the ROC curves for all three algorithms are almost perfect. However among the three, Multinomial Naive-Bayes did the best since it has the highest value in all of the performance metrics (has lower false-positive values, and high true-positive and true-negatives). Therefore, if a company were to invest in a spam detection algorithm, they should use the multinomial naive-bayes algorithm.

Despite the fact that Naive-Bayes is the best choice in our example, both random forests and logistic regression also did very well in terms of performance since each of the performance metrics were close to 1.