Spam Email Classifier using Machine Learning Models

Mason Leung, Wen Tao Lin

Introduction:

Our goal for this case study was to develop an algorithm that would be able to classify emails based on the relative frequency of 30 different words. Because we were given both the features and their respective classifications, we were able to choose a supervised learning method among the ones that we learned in class. The final classifier that we decided upon was the *random forest classifier(RF*) as it had the highest combination of AUC as well as true positive rate at 1% false positive rate (TPR at 1% FPR).

Final Choice:

Hyperparameters*:*	Value*:*	Description*:*
criterion	'gini'	It is the criteria used as a function in order to measure the quality of the split
n_estimators	2000	Number of trees generated
min_samples_split	4	Minimum number of samples needed to split an internal node
min_samples_leaf	1	Minimum number of samples required to be at a leaf
max_depth	15	Maximum depth of the tree
bootstrap	FALSE	Samples are not drawn with replacement

Pre-processing:

Initially, we split the data set into 80% training and 20% test set in order to perform data exploration and data processing without data leakage.

Figure 1 below shows the frequency of each word in the total number of spam emails and not spam emails. We divided the training set into two sets of data: one with all the spam emails, and the other with all the legitimate emails. This was to see if there were any trends apparent in one but not the other. However, in terms of the overall shape, nothing statistically significant can be determined visually from the graph.

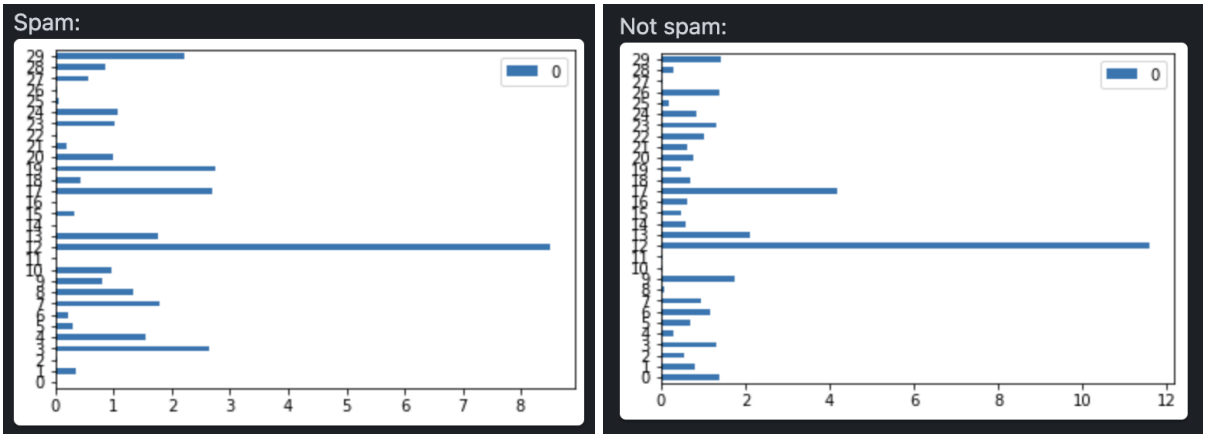

Figure 1: Word Frequencies in Spam and Non-spam Emails

We also decided to plot the data points after performing PCA dimensionality reduction to see if there could be any noticeable trend in the plot. **Figure 2** below represents the data set being reduced to 2 principal components. However, this graph as well shows no significant correlation between the features and the two classes.

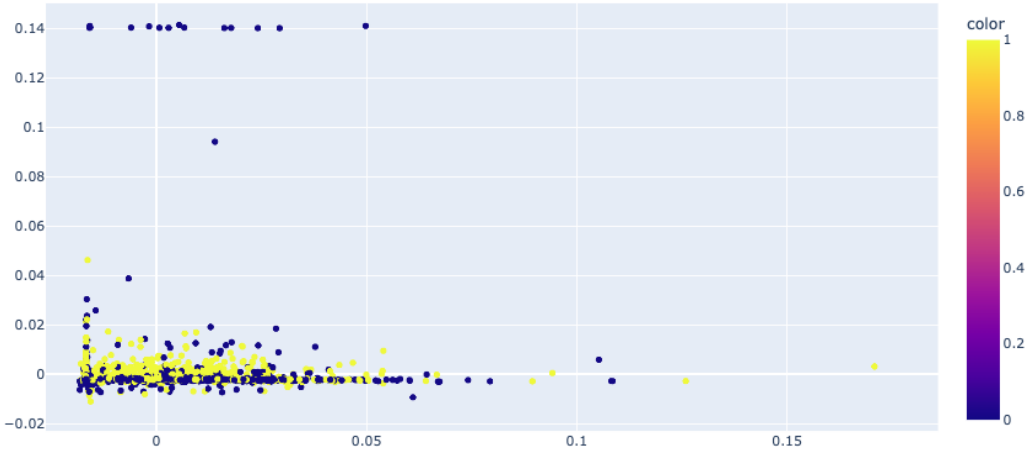

Figure 2: PCA Transformed Scatter Plot

When we initially performed a brief round of algorithm selection as described in the section below, a quick observation was made that standardization and normalization affected the accuracy negatively. Feature selection was done at a later stage in combination with hyperparameter tuning.

Classification Algorithm Selection:

Our plan for classification algorithm selection involved testing a variety of linear and non-linear classifiers and seeing how well they performed with the default parameters without any tuning. **Figure 3** below shows the 10-fold cross-validation mean AUC for each of the classification algorithms we tested. From there, we decided to move on with *random forest* and *support vector machine with the gaussian kernel* (*SVM RBF)* as they produced two of the highest mean AUC using the given spam dataset. We also

noticed that *logistic regression(LR)* has high accuracy but we decided to drop the three *LR* algorithms because they produce low TPR at 1% FPR when we further investigated. Furthermore, we also don't expect *logistic regression* and *KNN* to perform well due to the curse of dimensionality, given that there are 30 features.

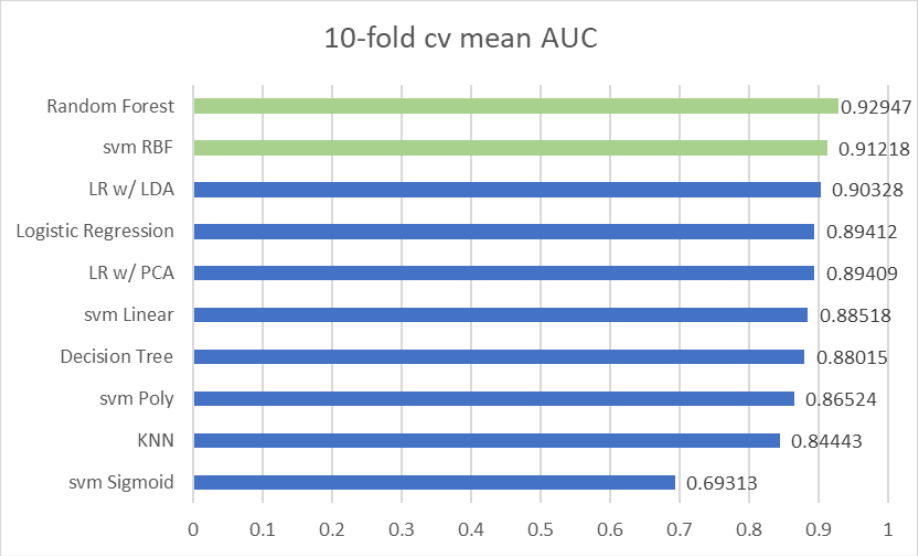

Figure 3: Mean AUC for Algorithms of Interest

Our next step in algorithm selection was to perform feature selection and compare the accuracy for *random forest* and *SVM RBF* using the optimal hyperparameters that were returned from GridSearchCV. Hoping to improve the true positive rate, we also tried out the *voting classifier* using the two algorithms we chose above. The test set AUC for the three classifiers are near each other with a range of ±0.20, but the TPRs differ by a decent margin (~0.64-*RF* vs. ~0.55-*Voting* vs. ~0.50-*RBF*). As a result, we decided to proceed with *random forest* as it produced a consistent high test set AUC and a higher TPR at 1% FPR. The classification algorithms were evaluated using spamTrain1 as train data and spamTrain2 as test data, and also tested again using the 80/20 split after combining spamTrain1 and spamTrain2.

Random Forest Classifier:

After selecting the *random forest classifier*, we decided to stagger feature selection and hyperparameter tuning, so that the result would lead to the optimal hyperparameters paired with the ideal features. Using the 80% train split from the 3000 given samples, we first performed randomized search cv, instead of the more computationally demanding grid search, to narrow in on the range of optimal values for the hyperparameters. Then we used grid search cv to exhaustively search through the potential values, recording the best performing hyperparameter values.

Given the selected hyperparameter values, we used two different feature selection techniques: sequential backward selection (SBS) and recursive feature elimination with cross validation (RFECV). The two yielded two different results: SBS kept features: [0, 2, 3, 4, 6, 7, 8, 12, 16, 19, 21, 22, 24, 28, 29], whereas RFECV kept all features but [11]. Then, we performed another grid search in order to have the most

optimal hyperparameters for the newly selected features. The result can be summarized down below with three different experiments:

1. Training the classifier using spamTrain1 and testing it against spamTrain 2.
2. Training the classifier using 80 percent of the data from spamTrain1 + spamTrain2 and then testing it against the 20 percent left over data.
3. Training the classifier using 90 percent of the data from spamTrain1 + spamTrain2 and then testing it against the 10 percent left over data.

Experiments

Different Test Sets:	**Performance Metric**	*Original Random Forest*	*RFECV*	*SBS*
Train: Spam 1 Test: Spam 2	AUC	0.9078	0.908	0.9044
	TPR at 1% FPR	0.5096	0.5079	0.4869
Train: 80% of total Test: 20% of total	AUC	0.9325	0.9325	0.9132
	TPR at 1% FPR	0.6402	0.6318	0.6619
Train: 90% of total Test: 10% of total	AUC	0.9377	0.9388	0.9249
	TPR at 1% FPR	0.6746	0.6746	0.6825

From the experiments, we observed that the *random forest classifier* with the selected features from SBS has the highest TPR at FPR of 1% where there was more data, but the classifier also suffered a performance loss in terms of AUC. TPR also has a high variance, so there is a risk in using the SBS selected features for the random forest classifier, as there is no guarantee that the TPR would perform to the same degree on the actual test set.

We settled on training the *random forest classifier* without feature 11 as it performed better than the original classifier when we trained both algorithms on 90% of the total data and tested it against the remaining 10% data. Both the AUC and TPR at 1% FPR metrics were greater, as shown above. We are assuming that this shows a trend in the classifier when more data is provided. Taking this into account, we chose to drop feature 11 because we would have an additional 600 train samples since the accuracy in the chart above was trained using 2400 and 2700 samples for 80% and 90% respectively.

Figure 4 below shows the confusion matrix to compare the predicted labels vs. the actual labels using the random forest classifier. 343 represents the number of correctly classified non-spam emails. 190 represents the number of correctly classified spam emails, which leaves the last two numbers for the misclassification rate. 18 emails were misclassified as spam (false positive), and 49 emails were misclassified to be non-spam (false negative).

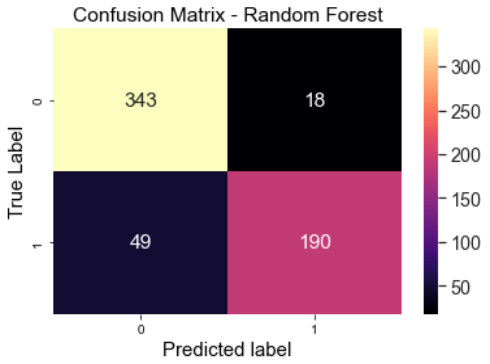

Figure 4: Confusion Matrix for Random Forest Classifier

Application:

The evaluation of the algorithm's efficacy in general could be calculated using the total number of correct classifications divided by the total number of examples. In this case, the total number of correctly classified emails, whether spam or not spam, should be divided by the total number of emails being classified to compute the algorithm's accuracy. However, an emphasis of the algorithm's accuracy should also be placed on the classifier's false positive rate, as there could be severe consequences if an emergent email was to be classified as spam and never read by the user.

This *random forest classifier* first trains decision trees on the sub datasets of the original dataset using different sub features, and then aggregates the outcomes into a majority vote. Since our hyperparameter tuning algorithm indicated that 2000 individual decision trees would be the optimal number, the user has to note that this model can require a lot of memory as a spam filter due to the excessive number of trees. However, the vast number of trees also increases the classifier's efficiency in large datasets. For example, because spam emails are so common and accessible, the classifier would have an abundance of training examples. We also expect that real world data is extremely noisy and unprocessed, so the *random forest classifier* also does a superb job at preprocessing the data.

One of the most important things for a classifier is having a sufficient amount of data to be able to train the classifier. The more training samples, the better the classifier will perform in testing as demonstrated by the increased performance in both AUC and TPR after training the classifier on the entire dataset rather than a subset of it. A way to be able to gather more data is by having users manually submit examples of where their email is being misclassified. There are two cases of where an email could be misclassified:

1. False positive, where the email was misclassified as a spam and placed into the spam folder
2. False negative, where the spam email was misclassified as a non-spam email and placed into the normal inbox

Users have options of reporting emails in the normal inbox as spam, as well as being able to go into their spam inbox and reporting an email as not spam. Then there can be an algorithm that parses through the email in order to convert the email into a viable data point with their respective true class. Both actions of reporting misclassification will send back vital sample points to retrain the classifier on in order to reduce the false positive rate and the false negative rate.