Email Spam Classifier

MA4270 Term Project

Anika Lee Ting Hui, Wang Yang Yi
National University of Singapore

1. Introduction

In MA4270, we have learnt several classification techniques such as the perceptron algorithm, Support Vector Machines (SVM), AdaBoost and logistic regression etc. These algorithms mainly focus on binary classification and hence in this project, we would explore and compare how good some of these algorithms are in doing binary classification. On top of that, we will be using Random Forest Classifier to see how it fares in this classification problem.

The classification problem we will be solving is to determine whether an email is considered spam or not. We chose this problem as email spam filtering is widely used by email providers to filter out phishing emails and scams. It is also easy to understand which allows us to dive deeper into how these algorithms work on this problem and their effectiveness.

2. Problem Definition

2.1 Dataset

For this email spam filtering problem, we have obtained a dataset with class labels from kaggle to train our model. The dataset consists of 5172 emails and 3001 columns including the class labels. The columns are different words that appear in emails and each observation consists of the number of times these words appear in the email. Some examples of the more common words found in the emails are “the”, “to”, “and”, etc. The last column has the labels for prediction: 1 for spam email and 0 for non-spam.
In this dataset, there were no null values and all entries have numerical values. Hence, data need not be further pre-processed.

2.2 Training and testing sets

To be able to evaluate our models’ performances, we have split the dataset into 2 subgroups -- training set (70%) and testing set (30%). Splitting the dataset ensures that we are building more accurate models that are relevant to future data and not just accuracy on the data that we are training the model with as it prevents overfitting.

3. Methods

We have explored 3 different methods for this classification problem:

1. Support Vector Machine (SVM)
2. AdaBoost
3. Random Forest
4. Logistic Regression

3.1 Support Vector Machine (SVM)

As described in MA4270, the SVM algorithm finds the optimal hyperplane that separates the samples into different groups. In the case of data that are not 'centered', an offset will be incorporated and this is called the affine classifier. While some data are not linearly or affinely separable, 'slack' variables are introduced to allow some errors.

Recall the equation for primal-SVM with slack:

$$\min_{(\boldsymbol{\theta},\theta_0,\boldsymbol{\xi})\in\mathbb{R}^d\times\mathbb{R}\times\mathbb{R}^n_+} \frac{1}{2}\|\boldsymbol{\theta}\|^2 + C\sum_{t=1}^{n}\xi_t \quad \text{s.t.} \quad y_t\left(\langle\boldsymbol{\theta},\mathbf{x}_t\rangle+\theta_0\right)\geq 1-\xi_t, \quad \xi_t\geq 0 \qquad \forall t=1,\dots,n.$$

Figure 1. Equation taken from MA4270 lecture notes

In python's 'sklearn' package, the SVM model consists of several tuning parameters - kernel, C, gamma, where C corresponds to the penalty of the error term in the equation. In class, we learnt that as penalty C increases, the margin becomes harder and less error is allowed. We experimented and fitted the model with different C values and we see that as C increases, the number of correct predictions and accuracy indeed decreases.

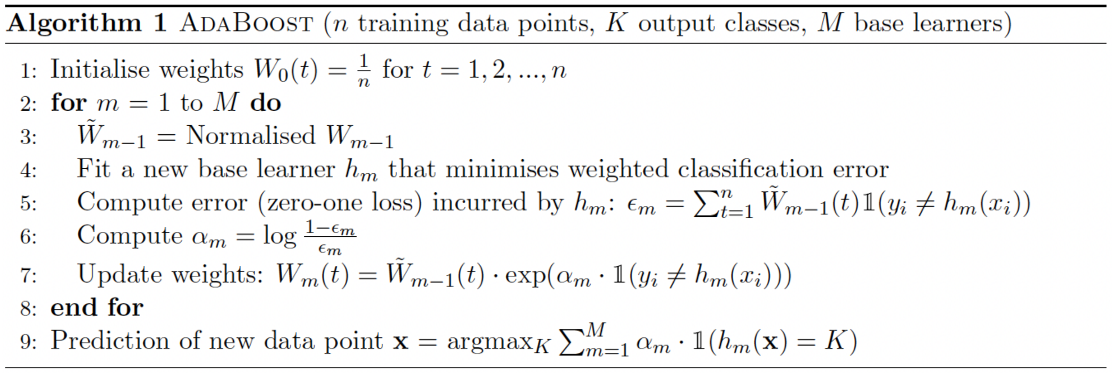
C	**No. of correct predictions**	**Accuracy (3 s.f.)**	**Precision (3 s.f.)**
1.0 (default)	1500	96.6%	93.1%
5.0	1489	95.9%	91.9%
10.0	1487	95.8%	91.9%

Table 1. Evaluation of SVM model with different penalty C

In this project, we will fit a SVM model with linear kernel and default values of C and gamma.

3.2 AdaBoost

Adaboost algorithm, which is short for Adaptive Boosting, got its name as the weights are re-assigned to each instance, hence “adaptive”. Higher weights are assigned to the incorrectly classified instances and smaller weights are assigned to the correctly classified instances.
As learnt in class, recall the steps of the algorithm:



Algorithm 1 AdaBoost (n training data points, K output classes, M base learners)

```
1: Initialise weights $W_0(t) = \frac{1}{n}$ for $t = 1, 2, ..., n$
2: for $m = 1$ to $M$ do
3:     $\tilde{W}_{m-1}$ = Normalised $W_{m-1}$
4:     Fit a new base learner $h_m$ that minimises weighted classification error
5:     Compute error (zero-one loss) incurred by $h_m$: $\epsilon_m = \sum_{t=1}^{n} \tilde{W}_{m-1}(t) \mathbb{1}(y_i \neq h_m(x_i))$
6:     Compute $\alpha_m = \log \frac{1-\epsilon_m}{\epsilon_m}$
7:     Update weights: $W_m(t) = \tilde{W}_{m-1}(t) \cdot \exp(\alpha_m \cdot \mathbb{1}(y_i \neq h_m(x_i)))$
8: end for
9: Prediction of new data point $\mathbf{x} = \text{argmax}_K \sum_{m=1}^{M} \alpha_m \cdot \mathbb{1}(h_m(\mathbf{x}) = K)$
```

Figure 2. Algorithm from lecture notes

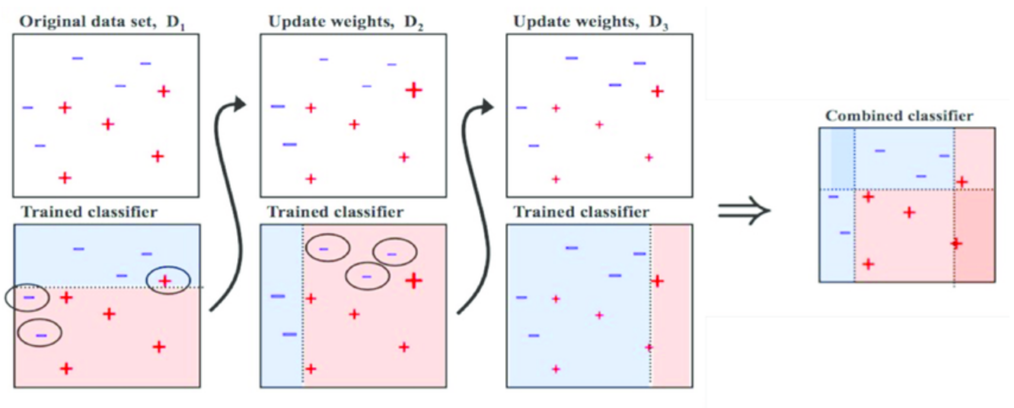


Figure 3. Visual Aid for Adaboost algorithm[1]

[1] Image from: https://towardsdatascience.com/understanding-adaboost-for-decision-tree-ff8f07d2851

3.3 Random Forest

Random forest is a supervised learning algorithm which builds an ensemble of decision trees for the classification problem. To understand the random forest algorithm, one has to first understand how decision trees work. A decision tree is made up of several decision nodes that splits the data. Eventually it reaches the leaf node that determines the final classification group. Random forest creates a forest of these decision trees and produces the majority vote - what majority of the decision trees predicted, as seen from the figure.

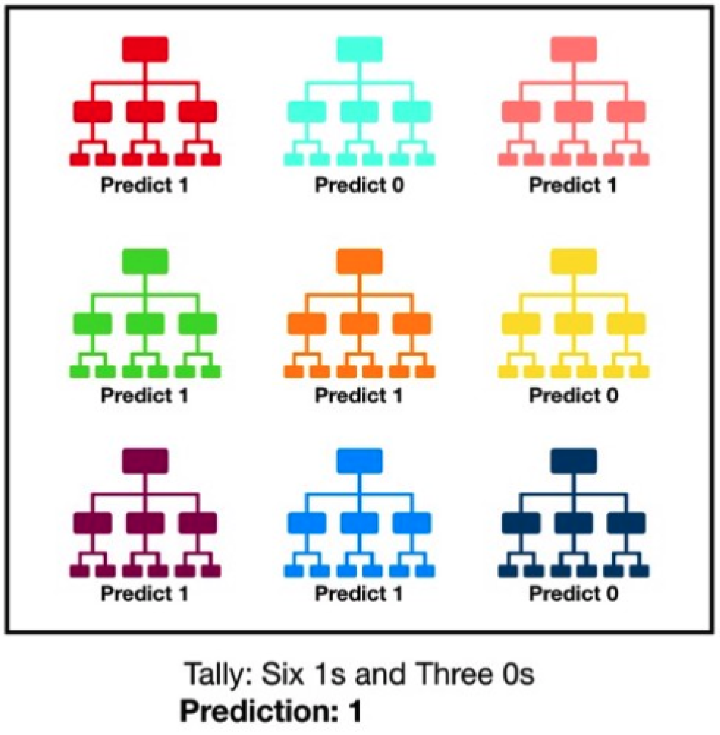


Figure 4. Random forest[2]

In our project, we have trained a random forest classifier that consists of 100 decision trees.

[2] Image from: https://towardsdatascience.com/understanding-random-forest-58381e0602d2

3.3 Logistic Regression

Logistic Regression is one of the most commonly used, if not the most, regression models for binary classification problems. Since our dataset is a binary classification problem, where the prediction is either spam email (1) or non-spam email (0), we decided to include this method as well.

Unlike in Linear Regression where we fit a straight line, for Logistic Regression, we fit an S-shaped curve, called Sigmoid, to our observations from the dataset, this is to help us map the predictions to probabilities.

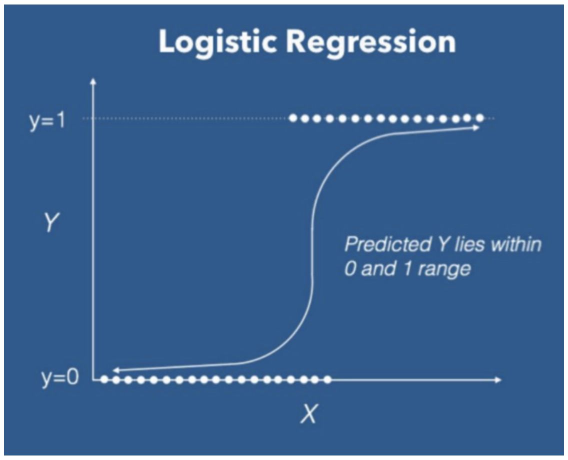


Figure 5. Sigmoid curve with data points[3]

$$sigmoid(x) = \frac{1}{1 + e^{-x}}$$

Figure 6. Sigmoid Equation

4. Experiments

For the experiments, we first use a 70-30 split for the training set and testing set. All the models are trained using the same training set and their performance is evaluated on the same testing set. We evaluated the models based on their accuracy, precision and training time.

[3] Image from: https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148

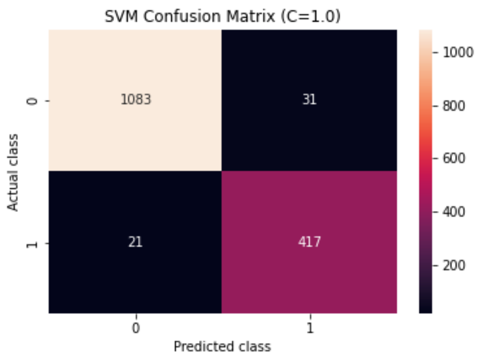
4.1 Algorithm performance

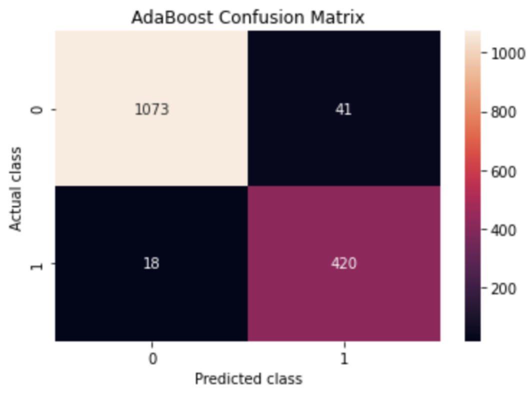
Using “autotime” in google colab and metrics from python’s “sklearn” package, we obtained the training time, accuracy and precision of the 3 models.

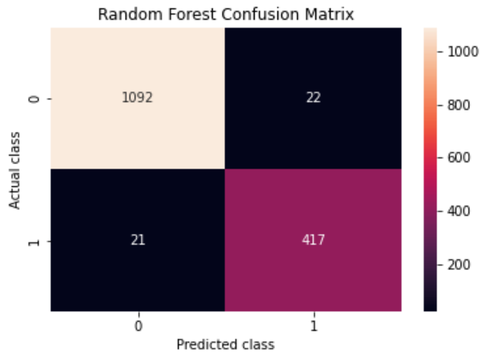
Algorithm	Time taken (s)	Accuracy (3 s.f.)	Precision (3 s.f.)
SVM	16.1	96.6%	93.1%
AdaBoost	11.6	96.2%	91.1%
Random Forest	2.79	97.2%	95.0%
Logistic Regression	1.32	97.2%	93.8%

Table 2. Evaluation of models

4.1.1 Confusion matrices







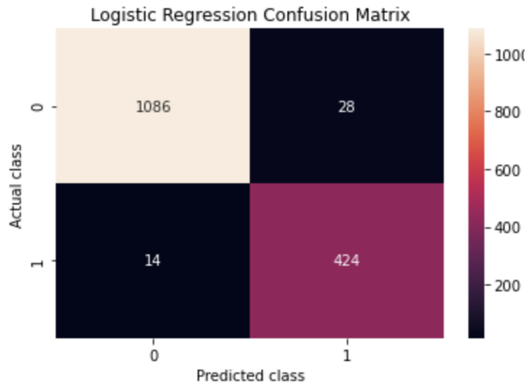
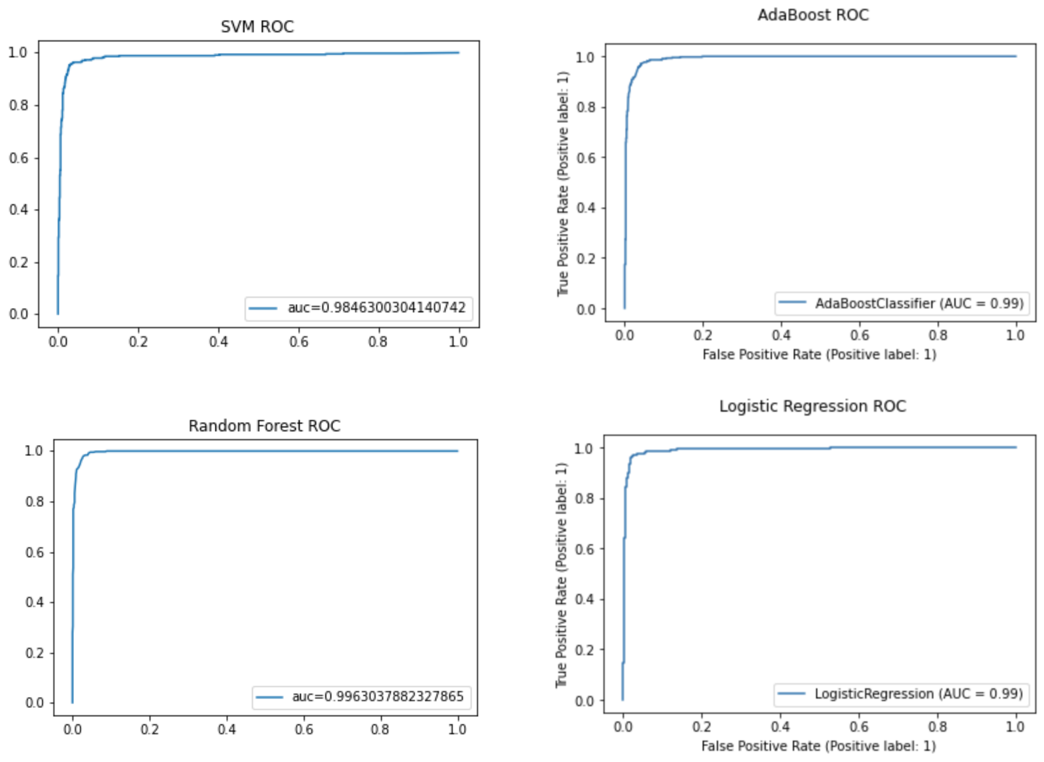


Figure 7: Confusion Matrix

The confusion matrices show the summary of the prediction results on the classification problem. It counts the number of predictions in each group which gives a clearer picture of the prediction accuracy of each model. The logistic regression and random forest can be seen to make the most number of correct predictions with 1510 and 1509 correct predictions respectively.

4.1.2 ROC curves



To evaluate the models, we also looked at the area under the ROC curve. The larger the area, the better the model. When the ROC curve is closer to the upper left corner, it also shows that the test is more efficient. From the ROC curve and area under the curve, we can see that random forest performed the best.

5. Conclusion

Based on the different metrics we have chosen, the random forest model proves to be the best performing model for this classification problem whereas AdaBoost performed the worst. Despite logistic regression having the shortest training time of only 1.32s, the random forest model outperformed the logistic regression model for the following reason. Both the random forest and logistic regression obtained a very similar accuracy score, with just a slight difference of one extra misclassified point by the random forest model. However, the random forest model performed better than logistic regression in terms of precision, by a significantly larger margin. Therefore, in general, the random forest model is the most suitable model for this particular data out of the four that we tested.