APANK 4335 Machine Learning: Concepts & Application
Dr. Jinjun Xiong
March 17, 2017

Machine Learning in Spam Email Classifications

Group WAD: Furong Bai, Shuangqiu Liu, Danlu Pan,Xiangxi Kong

Introduction and Problem Statement

In a more general way, spam could be defined as any unsolicited email. The statistics shows that the share of global spam volume as percentage of total email traffic is 61.33 percent in the most recently reported period (“Spam statistics”, n.d.). Applying machine learning techniques on filtering email would distinguish legitimate emails from spam which has been considered one of the most effective approaches for countering spam at this moment.

While a wealth of documentations examined spam detectors by using different machine learning techniques, most of them focus on implementing classification. In this project, we perform different the linear classification algorithms including: Adaline, Logistic regression, and SVM. For Adaline and Logistic regression, we code concrete steps and write our own algorithms to train the model. For SVM, we used an efficient R. package, “doMC”.

Then we followed by comparing the strength and weakness of each classifier, their performance on detecting spam, and impact of different ways of splitting dataset on model accuracy .

Data Cleaning and Exploration

The source of data is UCI dataset repository. The original dataset contains 4601 observations of 58 variables, including 1 nominal class label denoted whether the email was considered spam or not, and 57 features represented frequency of a particular word or character occurring in the email.

There is no missing value found in entire data set. We also tracked various characteristics of all numeric variables such as mean,median,mode, variance, standard deviation, quartiles, and etc. Appendix I shows the summary table of these statistics.

Correlation plot (Fig 1.) below shows the linear correlation between each pair of continuous variables which quantifies the strength of a linear relationship between two numerical features. In spite of the values sitting on the diagonal, dots in darker blue are usually considered as positively correlated while those in darker red will be regarded as negatively correlated.

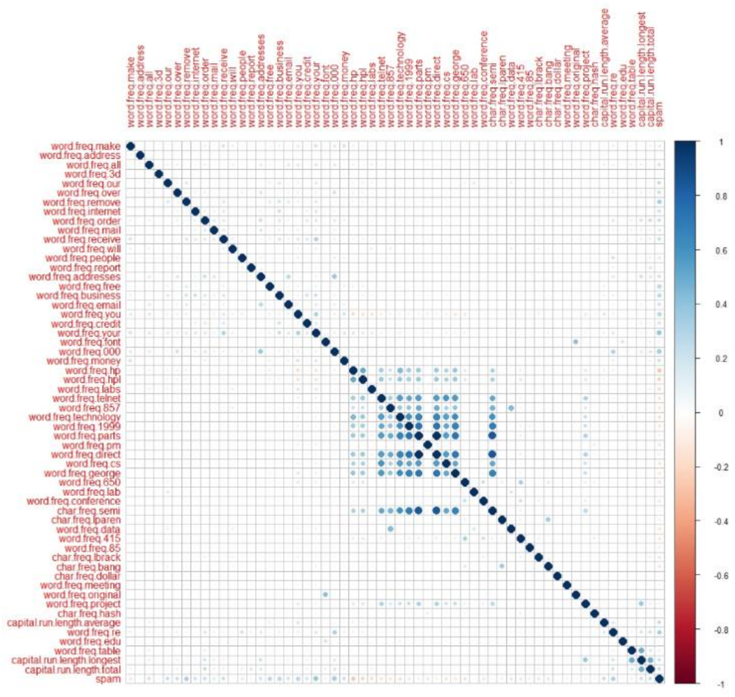


Fig.1 correlation plot for all features in pairs

Typically, in our model of Adaline and logistic regression, we assume that all predictors are independent. Just as what we can see from the *Fig.1*, the majority of variables are independent from each other. However, there are few predictors that generate higher correlations, which may bring disturbs or errors when we apply linear model to the datasets.Fortunately, the RBF kernel we use in this project could effectively deal with nonlinear problems (Scikit-learn line 1). In a nutshell, we will pay more attention to these latent factors and come up with more practical algorithm or model in future research.

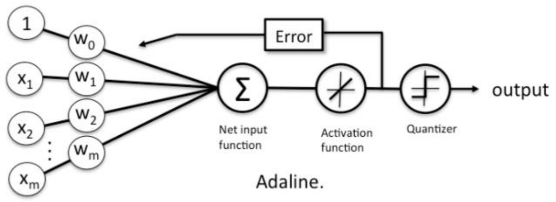
Boxplots (Appendix II) display the distributions of each predictor against label feature, namely spam or not spam.

Building Classifiers

Before doing classification on data set, we split the original data set into training set and testing set as training data is used for building and tuning models while testing data provides an indication on how well the trained model would generalize to new data. We use 80:20 randomly sampled subset for training and testing sets as initial step. We also standardize all the predictors to obtain unit variance since it will help to find an appropriate learning rate if the predictors are on the same scale. Also it will prevent the weights from becoming too small and lead to faster convergence of global optimization.

ADALINE - Adaptive Linear Neuron Network

The Adaptive Linear Neuron Network(Adaline) is used to solve a supervised binary classification problem given a set of training data with known binary labels. It aims to find a classifier in a specific structure shown as below, such that the Sum of Squared Errors (SSE) between the net-input value and the target is minimized. Therefore it tries to move the decision boundary as far from the training patterns as possible(Drucker, 1999).



Adline updates the weights based on a linear activation function. In this case, we can define the activation function g(z) as the identity function of the net input $\mathbf{g(w^T x)=w^T x}$ where w is model parameters. The cost function which is SSE and can be calculated by formula below, where y(i) is i-th observation in the data set (Xiong class note 7, 2017):

$$f(w) = \frac{1}{2}\sum_{i=1}^{N}\left(y^{(i)} - w^T x^{(i)}\right)^2$$

One of the advantages of using linear activation function is that it is differentiable so that allows to tune our weights via minimizing cost function. The minimization problem is described as below (Xiong class note 7, 2017):

$$\min_w: f(w) = \frac{1}{2}\sum_{i=1}^{N}\left(y^{(i)} - w^T x^{(i)}\right)^2$$

In order to minimize the problem, we apply global optimization via Gradient Descent. It is the most popular learning algorithm for Adaline despite being the simplest form of optimization approaches. The search direction we used is (Xiong class note 7, 2017):

$$p(k)_j = -\frac{\partial f}{\partial w_j} = -\sum_{i=1}^{N}\left(y^{(i)} - w^T x^{(i)}\right)\left(-x_j^{(i)}\right) = \sum_{i=1}^{N}\left(y^{(i)} - w^T x^{(i)}\right)x_j^{(i)}$$

Evaluation of Adaline Classifier
In practice, it took us a number of experiments to find a good learning rate for optimal convergence. Fig. 2. plots log(SSE) along learning process for two different learning rates. They both converge to a stable low cost around 6.64. We choose to use 0.00001 learning rate in analyzing performance of Adaline classifier.

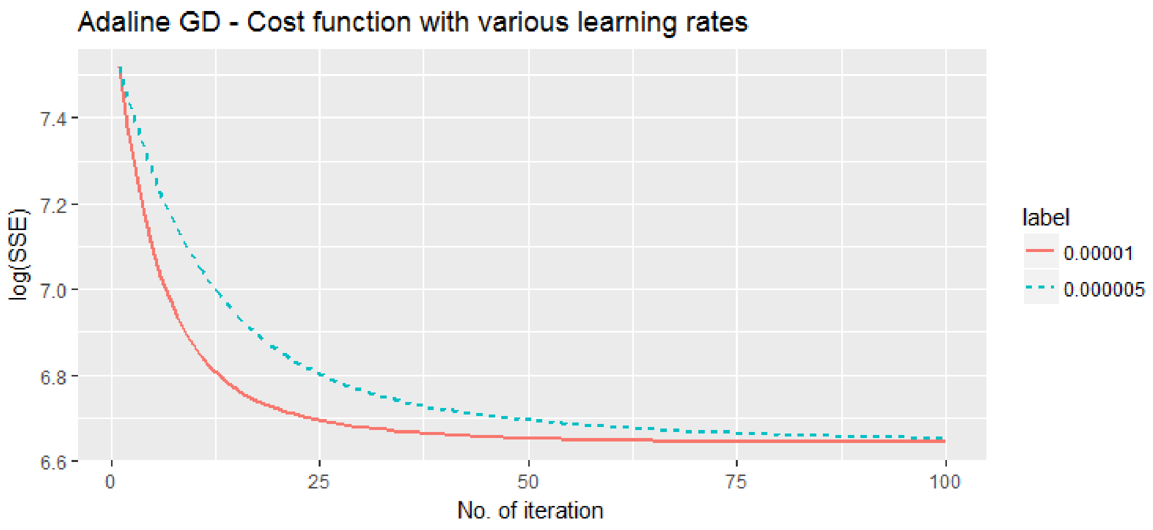


Fig. 2 Cost function for Adaline

Following below (Fig. 3 & Fig. 4) are two commonly used tools when selecting classification models based on relevant metrics: ROC (Receiver Operating Characteristic) curve and Precision-recall curve. ROC curve of Adaline classifier is above the diagonal line which indicates it is possible a good classification model. AUC (area under the curve) in this plot also presents the relative high performance of the model.

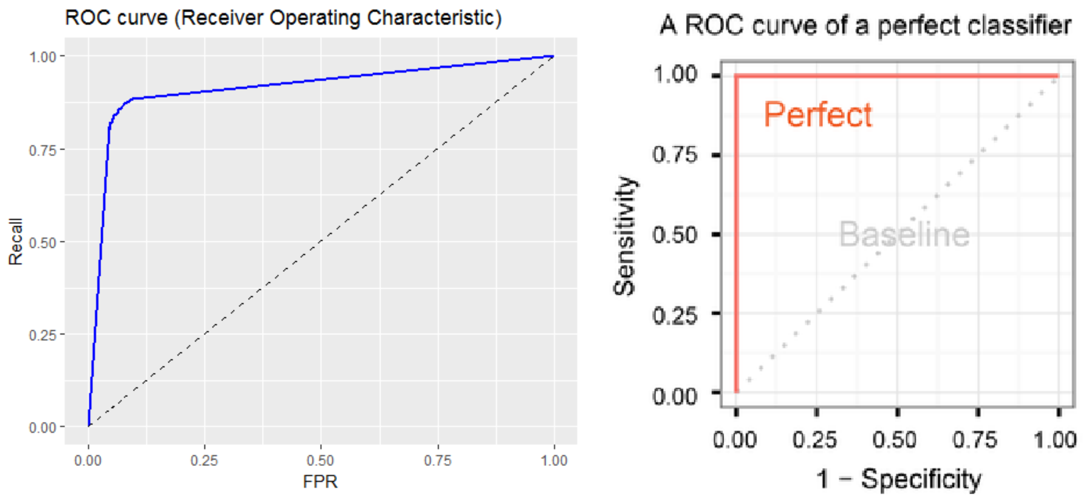


Fig.3 ROC Curve for Adaline

The Precision-recall curve (*Fig. 4*) plots the one-to-one relationship between ROC and Precision-recall curve. Baseline is defined by the ratio of positive and negatives: baseline = P/(P+N). Apparently, our curve is above baseline and roughly close to perfect line which indicates this model may be a good classifier.

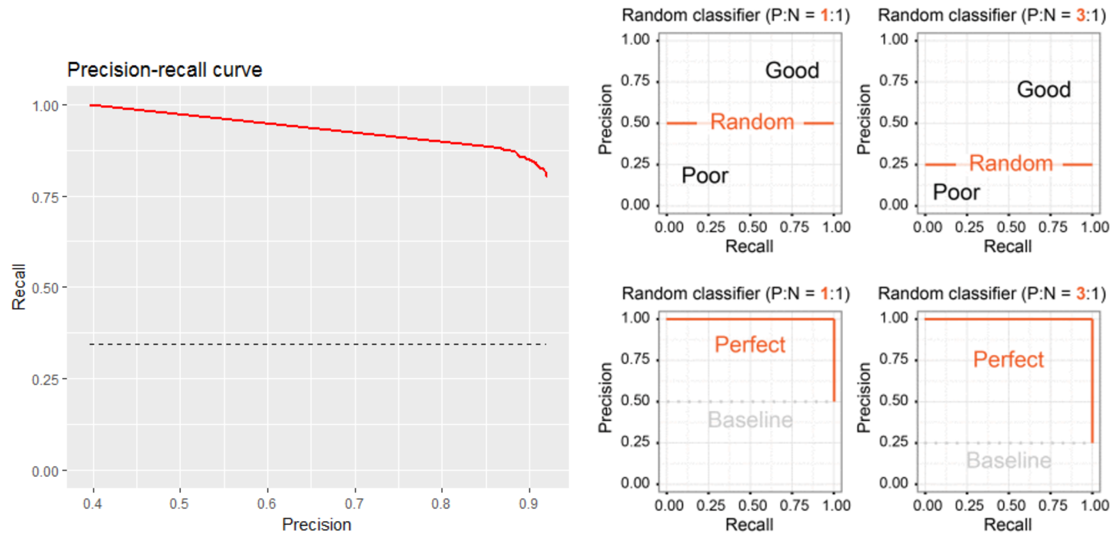
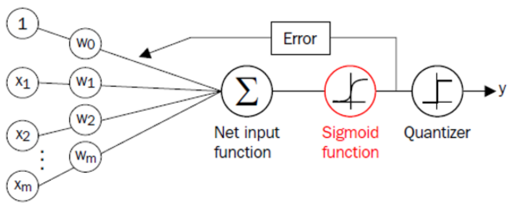


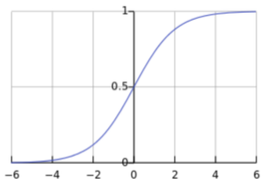
Fig.4 Precision-recall curve

Pros & Cons
Typically, we notice a set of strengths as well as weaknesses of utilizing the Adaline algorithm. For instance, it runs very fast in R. It is a straightforward approach but could create great potential value since it can adopt a number of different activation functions. However, it could be disturbed with bias if there exist strange correlation among predictors.

Logistic Regression

Logistic Regression, as what we have learned, can be regarded as Adaline using a new activation function and new objective function, so they have similar algorithm. The activation function becomes a sigmoid function: $g(z) = 1/(1+e^{-x})$ which generates a value between 0 to 1, indicating how likely that a given sample of Xs belongs to a certain class y (spam or not spam). (Xiong class note 7, 2017)





The quantizer is defined as O(g(z)) = 1 if g(z) > 0.5; otherwise equals to 0. Instead of using SSE, we define the objective function as maximization of the likelihood function for all training samples since it could be a better substitute of SSE (Xiong class note 7, 2017).

$$\max_{w}: \; L(w) = \prod_{i=1}^{N} \left(g\left(w^T x^{(i)}\right)\right)^{y^{(i)}} \left(1 - g\left(w^T x^{(i)}\right)\right)^{1-y^{(i)}}$$

Converting it into a minimization problem, the cost function become (Xiong class note 7, 2017):

$$\min_{w}: \; f(w) = -l(w) = \sum_{i=1}^{N} \left[-y^{(i)} log\left(g\left(w^T x^{(i)}\right)\right) - \left(1 - y^{(i)}\right) log\left(1 - g\left(w^T x^{(i)}\right)\right) \right]$$

We follow by doing global optimization to solve the minimization problem via Gradient Descent. The searching direction is similar to direction in Adaline algorithm (Xiong class note 7, 2017):

$$p(k)_j \;\; = \sum_{i=1}^{N} \left(y^{(i)} - g\left(w^T x^{(i)}\right)\right) x_j^{(i)}$$

One possible risk of implementing logistic regression is that it attempts to predict outcomes based on a set of independent variables, but logit models are vulnerable overconfidence. That is, the model may appear to have more predictive power than it actually does as a result of sampling bias. But this problem does not happen in this case.

Evaluation of Logistic Regression Classifier

Fig 5. plots cost function along learning process for two different learning rates. As we can see, they both converge. In this case we decide to use 0.00015 learning rate since it leads to a lower minimized cost and its curve seems to have better convergence result.

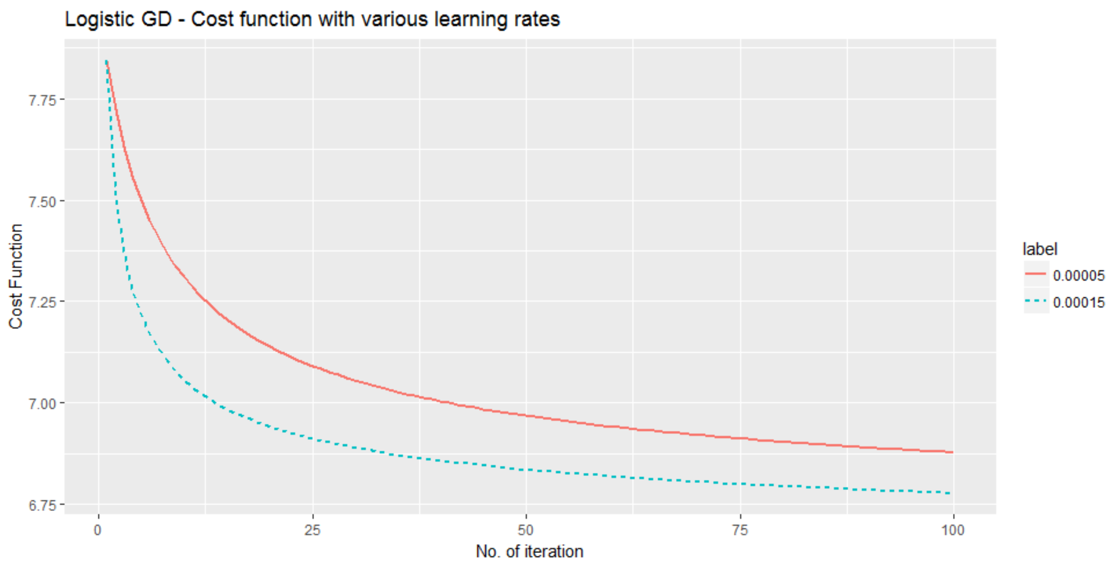


Fig. 5 Cost function for Logistic Regression

Like the model evaluation which we performed in Adaline model, we plot the ROC curve and precision-recall curve (Fig. 6) to evaluate performance of logistic classifier. Both plots imply that it is a good classification model as its ROC curve is above diagonal and precision-recall curve is above baseline.

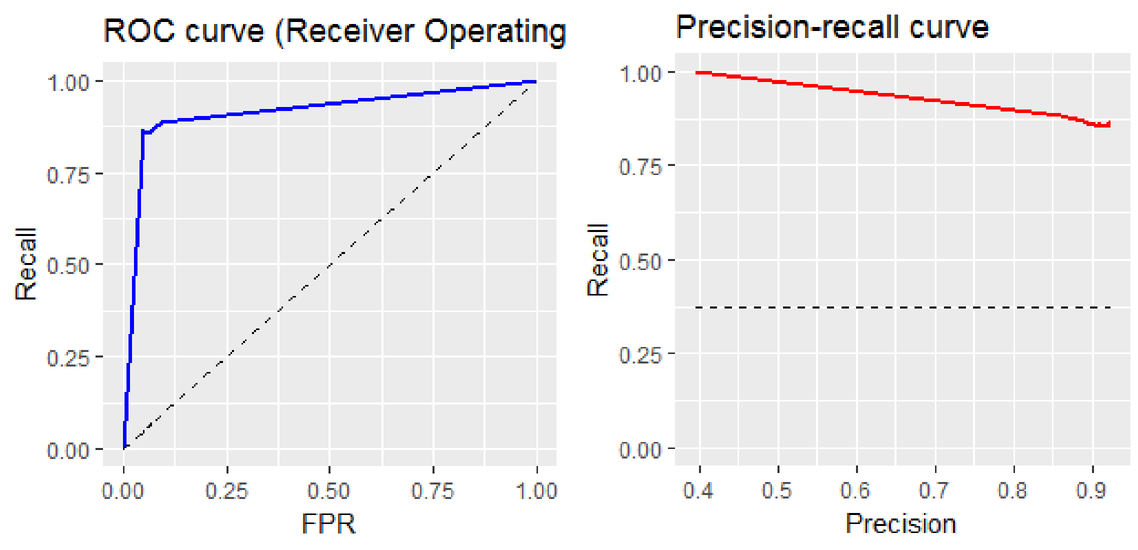


Fig.6 ROC Curve and precision-recall curve for Logistic Regression

Pros & Cons
Typically, we are attracted by the noticeable advantages of the logistic regression technique such as (Rae part 1):

1. It generates in a faster speed than other machine learning models (e.g. SVM);
2. It works more effectively if there is a single decision boundary which is not necessarily parallel to the axis;
3. It is a fairly simple approach with low variance and less prone to overfitting.
4. The logistic regression model also delivers better performance in higher dimensions than some normal techniques (e.g. decision trees);
5. It is also very effective in predicting categorical outcomes.

Unfortunately, there still exist some disadvantages of this algorithm such that we need to identify the independency of both variables and observations before applying such model to the dataset. What is worse, the logistic regression cannot be used to predict continuous outcomes. It is also vulnerable to the problem of overfitting (Robinson paragraph 2-5). So, we should be more careful when choosing models.

SVM
Support Vector Machine (SVM) is a supervised machine learning model that generates a great convenience on the problem of both classification and regression (Ray paragraph 5).
The highlighted mechanism for this approach is by defining an optimal separating hyperplane that will categorize new inputs or examples.Unlike the “hard-margin” formulation and the “soft-margin” formulation of SVM, the method we take advantage of is the Kernel SVM, or more specifically, the RBF (Radial Basis Function) kernel (Xiong class note 8, 2017).

$$K\left(x^{(i)}, x^{(j)}\right) = \exp\left(-\frac{\|x^{(i)} - x^{(j)}\|^2}{2\sigma^2}\right), \sigma > 0$$

The reason we choose this approach is that we are holding a relatively large dataset which contains 58 variables. Besides, what requires much attention is that we need to find out the optimal value of tuning parameter, , before we fit the SVM model onto the training dataset. It is convenient for our group to generate these processes due to the built-in functions in R.

```
sigDist <- sigest(spam ~ ., data = training.f, frac = 1)
```

Turning into the concrete steps, the first action we conduct is to transform the splitted datasets, training and testing with binary labels as 0, 1, into their factor forms . Because the function we employ in R is more sensitive to factor values. Instead of explicitly mapping the original data to a higher dimension, we just calculate the inner product in that higher dimension (Xiong class note 8, 2017). Therefore, the next step is to define the dot product in the transformed space with the following mechanism:

$$\begin{tikzcd}
\ldots, x^{(i)}, \ldots, x^{(j)}, \ldots \arrow[r] \arrow[d, "\psi(\cdot)"] & \left(x^{(i)}\right)^T x^{(j)} \arrow[d, "K(\cdot,\cdot)"] \\
\ldots, \psi\left(x^{(i)}\right), \ldots, \psi\left(x^{(j)}\right), \ldots \arrow[r] & \psi\left(x^{(i)}\right)^T \psi\left(x^{(j)}\right)
\end{tikzcd}$$

$$K\left(x^{(i)}, x^{(j)}\right) = \psi\left(x^{(i)}\right)^T \psi\left(x^{(j)}\right)$$

(where is the implicit mapping embedded in the RBF kernel.)

After we transform the dataset and figure out the optimal range of values for the "sigma" inverse width parameter in the Radial Basis kernel for use with SVM, we apply an optimization technique to solve for the following problem (Xiong class note 8, 2017):

$$\begin{array}{l} max_{\lambda}: F(\lambda) = \lambda^T 1 - \frac{1}{2}\lambda^T D \lambda \\ \text{s.t.:} \quad \lambda^T y = 0, \lambda \geq 0, \\ \text{where } D_{i,j} = y_i y_j K\left(x^{(i)}, x^{(j)}\right) \end{array} \Rightarrow \begin{array}{c} \text{The support vectors } \left(x^{(s)}, y_s\right) \text{ correspond to those} \\ \lambda_s \neq 0 \end{array}$$

The cardinal principle behind such an optimization problem is to find out the hyperplane that generates the maximum margin of the training data.

Pros & Cons

Generally speaking, Support Vector Machine is quite useful when there is a large number of dimensions. Several outstanding advantages of this method are listed as below (Ray part 7):

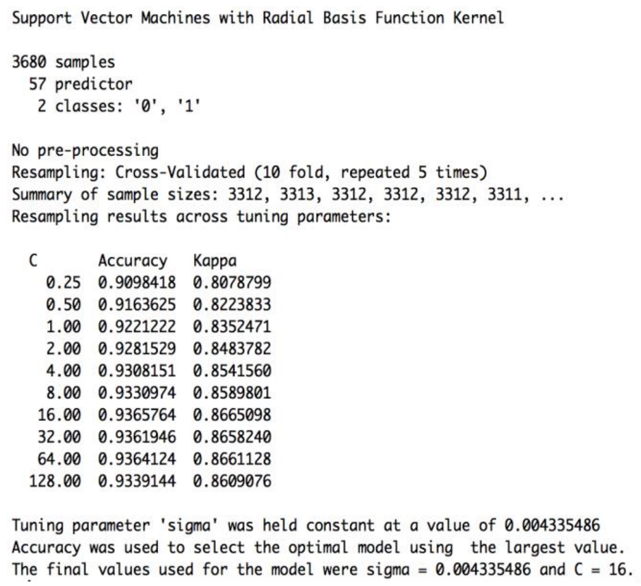
- It provides a relatively clear margin of separation by drawing a hyperplane;
- It performs effectively in high dimensional spaces;
- It is more effective when the number of dimensions is greater than the number of samples;
- It is also considered as memory-efficient because it uses a subset of training data in the decision function;
- Especially, the RBF kernel could effectively deal with nonlinear problems (Scikit-learn line 1).

As the saying goes, every coin has two sides. People may encounter a set of difficulties when they try to fit the SVM model onto their datasets. Potential disadvantages could be:

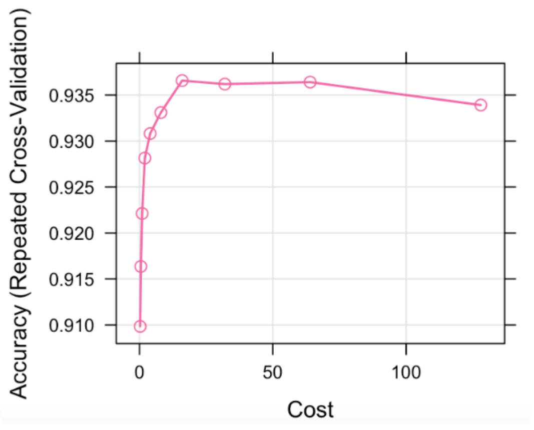
- It is such a high-cost method that will take more training time when people have a large dataset;
- It also will be disturbed when the data set has many noises or outliers;
- SVM does not directly estimate probabilities which are calculated using an expensive five-fold cross-validation (Ray part 7).

Result Analysis

In brief, a result table and a cost-accuracy plot are shown in the following context. Just as what we do in the previous methods, Adaline and logistic regression, the original dataset is splitted into training and testing sets by the ratio of 0.8. Basically, 57 predictors and a binary dependant variable with values of 0 and 1 are used to generate the cross-validation. In this case, 10 folds of cross validations are made to achieve the largest accuracy. As a result, holding the optimal tuning parameter Sigma= 0.004335486, the maximum accuracy we can get is about 93.67% with C = 16.

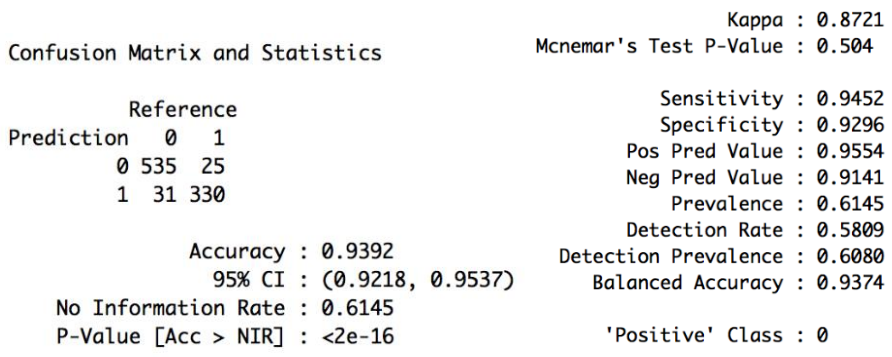


```
Support Vector Machines with Radial Basis Function Kernel

3680 samples
  57 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 3312, 3313, 3312, 3312, 3312, 3311, ...
Resampling results across tuning parameters:

  C       Accuracy   Kappa
    0.25  0.9098418  0.8078799
    0.50  0.9163625  0.8223833
    1.00  0.9221222  0.8352471
    2.00  0.9281529  0.8483782
    4.00  0.9308151  0.8541560
    8.00  0.9330974  0.8589801
   16.00  0.9365764  0.8665098
   32.00  0.9361946  0.8658240
   64.00  0.9364124  0.8661128
  128.00  0.9339144  0.8609076

Tuning parameter 'sigma' was held constant at a value of 0.004335486
Accuracy was used to select the optimal model using  the largest value.
The final values used for the model were sigma = 0.004335486 and C = 16.
```



Result Table & Cost-Accuracy Plot

After training our SVM model using the training dataset, we apply such a model to the testing test and receive the following result. As we can see from the table below, the total number of error (false negative + false positive) is 54, which is considerably small compared to the total number of 921 observations in the testing set. What is more, the 93.92% accuracy is relatively higher than other method of machine learning, which also indicates the feasibility of our SVM model.



```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 535  25
         1  31 330

               Accuracy : 0.9392
                 95% CI : (0.9218, 0.9537)
    No Information Rate : 0.6145
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8721
 Mcnemar's Test P-Value : 0.504

            Sensitivity : 0.9452
            Specificity : 0.9296
         Pos Pred Value : 0.9554
         Neg Pred Value : 0.9141
             Prevalence : 0.6145
         Detection Rate : 0.5809
   Detection Prevalence : 0.6080
      Balanced Accuracy : 0.9374

       'Positive' Class : 0
```

Confusion Matrix & Various Metrics

Table below (*Tab. 1*) shows the confusion matrices and relevant metrics for both training and testing datasets using three classifiers. From the table, we can conclude following attentional findings:

- Accuracies are higher in testing set compared to training set for all classifiers;
- Errors are lower in testing set than in training set for all classifiers;
- Precision rates improve when using testing data for Adaline and Logistic classifiers;
- Recall rate is lower using testing data when apply Adaline model but higher using testing data when apply Logistic model;
- FPRs change differently in different models as well.

Adaline	
Training Dataset	**Testing Dataset**

`         Negative Positive`
`Negative     2119      103`
`Positive      288     1170`
	`         Negative Positive`
	`Negative      544       22`
	`Positive       75      280`

Accuracy = 0.8937500
Error = 0.1062500
Precision = 0.9190888
Recall = 0.8024691
FPR = 0.04635464
	Accuracy = 0.8946797
	Error = 0.1053203
	Precision = 0.9271523
	Recall = 0.7887324
	FPR = 0.0388693

Logistic Regression	
Training Dataset	**Testing Dataset**

`         Negative Positive`
`Negative     2114      108`
`Positive      189     1269`
	`         Negative Positive`
	`Negative      538       28`
	`Positive       36      319`

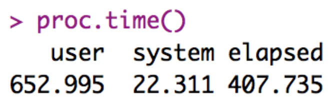
Accuracy = 0.9192935
Error = 0.08070652
Precision = 0.9215686
Recall = 0.8703704
FPR = 0.04860486
	Accuracy = 0.930510315
	Error = 0.069489685
	Precision = 0.919308357
	Recall = 0.89859155
	FPR = 0.049469965

SVM	
Training Dataset	**Testing Dataset**

N/A	`          Reference`
	`Prediction   0   1`
	`         0 540  26`
	`         1  26 329`

Accuracy = 0.9339179
Error = 0.0660821
Precision = NA
Recall = NA
FPR = NA
	Accuracy = 0.9435
	Error = 0.0565
	Precision = 0.9541
	Recall = 0.9541
	FPR = 1-0.9268

In addition, running time for SVM is much longer than other two models although it differs a little in each experiment and using different machines. Syntax below shows the running time of one typical SVM model running experiment. We run 100 iterations in every experiment. It took around 10 minutes (553.41 seconds) in this try. In contract, other two models only take less than 30 seconds. However, accuracy rate is improved a lot applying SVM algorithm.



```
> proc.time()
   user  system elapsed 
652.995  22.311 407.735 
```

In consideration of the question that which metric is more relevant for our task, we need to firstly identify what the task is. Briefly speaking, in this project, we are expected to build up an effective spam detection algorithm by exploring different machine learning techniques. In this regard, the accuracy of prediction should be the primary metric that we need to consider. The method to calculate accuracy is listed as below (Xiong class note 3, 2017):

$$Accuracy = 1 - Error = \frac{TP + TN}{FP + FN + TP + TN}$$

Where,

$$Error = \frac{FN + FP}{FP + FN + TP + TN}$$

Model with higher accuracy rate indicates a better prediction of the original dataset. In other words, small errors will be retrieved from the "better model". Therefore, the total number of true negative (TN) and true positive (TP) cases could be fairly large in these models. If speaking more practically, models generating higher rate of accuracy will make more effective detections on emails by correctly recognizing emails as their actual categories.

Since we have identified our task as obtaining higher accuracy, there are multiple actions we can undertake to realize such a goal. For instance, as shown in *Fig. 5* of logistic regression, the line with learning rate 0.00005 generates a larger error than the line with learning rate of 0.00015. Therefore, we choose the one with lower error as our learning rate. Other possible approaches to retrieve higher accuracy include

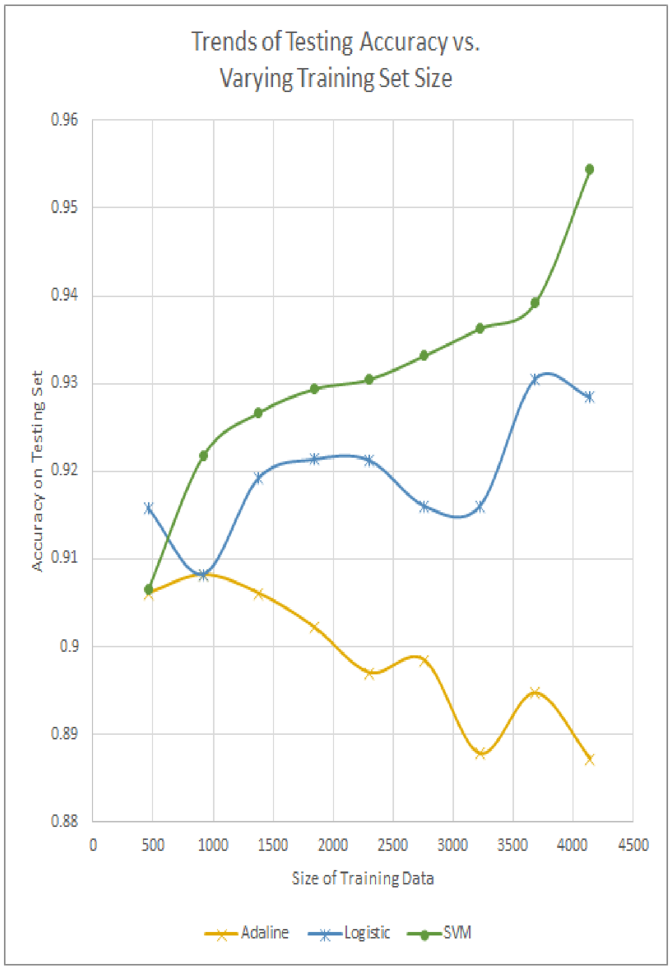
- changing the number of iterations;
- adding cross-validation into the model training process;
- using various activation functions (e.g. hyperbolic tangent sigmoid);
- taking predictors' independency into consideration, etc.

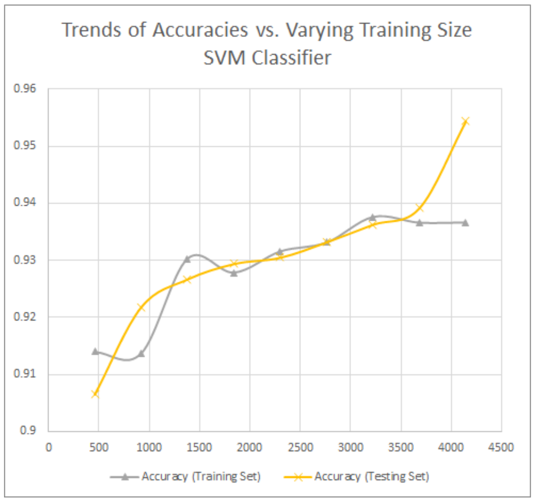
Comparison between various training & testing set sizes

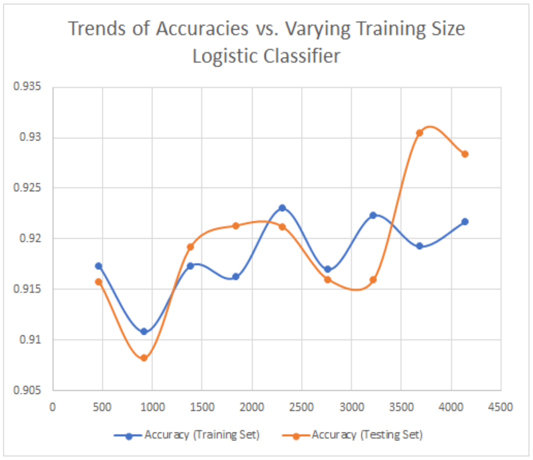
Our next step is randomly dividing original dataset into the training dataset and testing dataset with varying sizes, standardizing both sets based on training data mean and standard deviation, and re-perform previous steps. Tables in Appendix III compare the accuracies of both testing and training sets against setting various training data sizes, such as 90% of all data, 80%, 70% and etc..

Graph below is built to discover the influence of change in training size on test accuracy. It obviously represents the tendency of testing accuracy according to varying training size using different classifiers. One interesting finding from the graph is that when size of training dataset increases, accuracy of testing set generally grows if using Logistic model and SVM model. However, if we use Adaline model, testing accuracy gradually declines when we expand training data size. This may become an additional indication of which model fits best to data. In our case, SVM is highly possibly a better model.

In general, larger training size will lead to higher accuracy but it also depends on the model we apply on data. On the other side, training accuracies don't change a lot when using different training sizes.







Conclusion

In this machine learning project, we discovered mystery of classifications from various perspectives. We constructed our own Adaline and Logistic regression algorithms, and also performed the powerful SVM algorithm by using a R package. We built three different classifiers and tried different learning rates for the optimal convergence of each. In performance evaluation of three models, we compare multiple metrics and their relevance to our case, aiming to find feasibility of each classifier in detecting spam emails.

The experiment helped us to make a better understanding of classification techniques in machine learning. There are a number of taken-away that we learn from this research such as influence of training data size on testing accuracy.

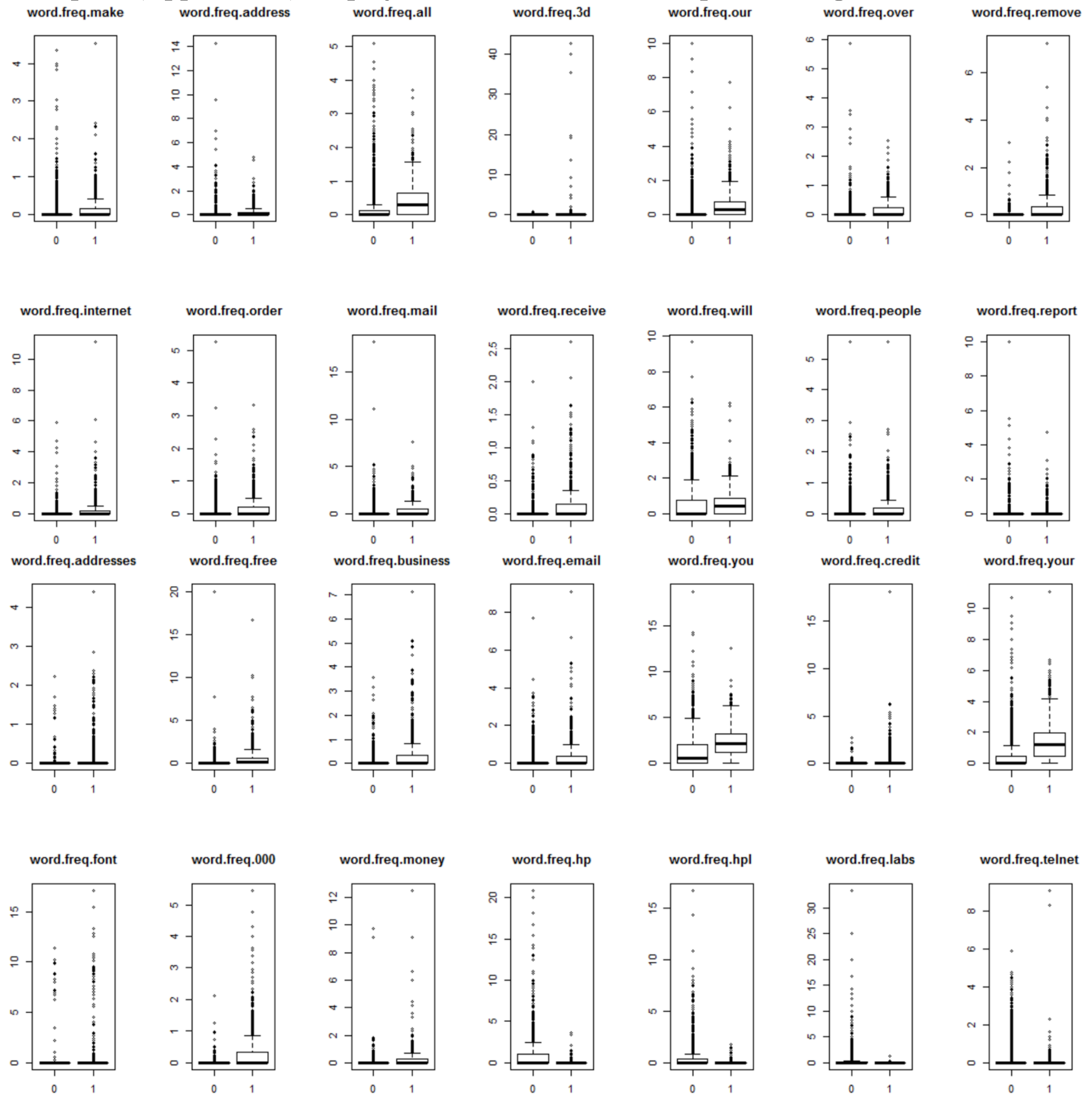
Appendix I
Summary Table For All Numeric Variables

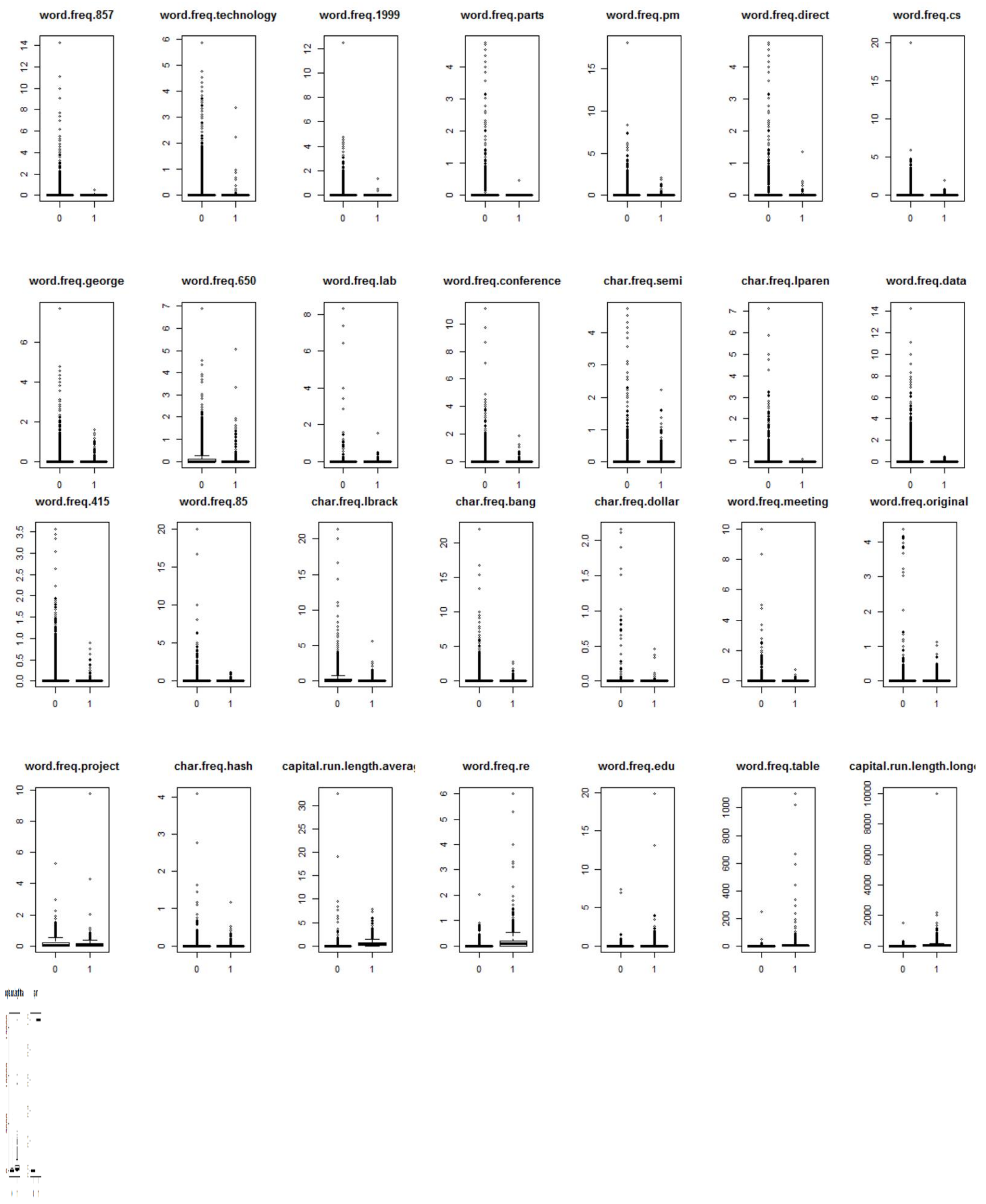
word.freq.technology.1%	0	0	0	0	0	0	0.65	1.72	0.097477	0.162105	0	7.69	0
word.freq.1999.1%	0	0	0	0	0	0	0.93	1.92	0.136953	0.179311	0	6.89	0
word.freq.parts.1%	0	0	0	0	0	0	0	0.19	0.013201	0.048687	0	8.33	0
word.freq.pm.1%	0	0	0	0	0	0	0.49	1.72	0.078629	0.18894	0	11.11	0
word.freq.direct.1%	0	0	0	0	0	0	0.36	1.58	0.064834	0.122441	0	4.76	0
word.freq.cs.1%	0	0	0	0	0	0	0	1.44	0.043667	0.130469	0	7.14	0
word.freq.meeting.1%	0	0	0	0	0	0	0.54	3.84	0.132339	0.588012	0	14.28	0
word.freq.original.1%	0	0	0	0	0	0	0.27	1.11	0.046099	0.050092	0	3.57	0
word.freq.project.1%	0	0	0	0	0	0	0.24	2.04	0.079196	0.386854	0	20	0
word.freq.re.1%	0	0	0	0	0	0.11	1.41	4.16	0.301224	1.023511	0	21.42	0
word.freq.edu.1%	0	0	0	0	0	0	1.02	4.16	0.179824	0.830138	0	22.05	0
word.freq.table.1%	0	0	0	0	0	0	0	0.04	0.005444	0.005818	0	2.17	0
word.freq.conference.1%	0	0	0	0	0	0	0	0.9	0.031869	0.081644	0	10	0
char.freq.semi.1%	0	0	0	0	0	0	0.172	0.645	0.038575	0.059278	0	4.385	0
char.freq.lparen.1%	0	0	0	0	0.065	0.188	0.53	1.03	0.13903	0.073092	0	9.752	0
char.freq.lbrack.1%	0	0	0	0	0	0	0.093	0.284	0.016976	0.011967	0	4.081	0
char.freq.bang.1%	0	0	0	0	0	0.315	1.116	2.631	0.269071	0.66532	0	32.478	0
char.freq.dollar.1%	0	0	0	0	0	0.052	0.377	0.894	0.075811	0.060458	0	6.003	0
char.freq.hash.1%	0	0	0	0	0	0	0.147	0.763	0.044238	0.184335	0	19.829	0
capital.run.length.average.1%	1	1	1.148	1.588	2.276	3.706	8.851	44.72	5.191515	1006.758	1	1102.5	0
capital.run.length.longest.1%	1	1	3	6	15	43	181	669	52.17279	37982.62	1	9989	0
capital.run.length.total.1%	3	6	10	35	95	266	1223	3027	283.2893	367657.7	1	15841	0
spam.1%	0	0	0	0	0	1	1	1	0.394045	0.238825	0	1	0

	quantile_.01	quantile_.05	quantile_.10	quantile_.25	quantile_.50	quantile_.75	quantile_.95	quantile_.99	mean	variance	min	max	percMissing
word.freq.make.1%	0	0	0	0	0	0	0.65	1.26	0.104553	0.093243	0	4.54	0
word.freq.address.1%	0	0	0	0	0	0	0.67	3.27	0.213015	1.665584	0	14.28	0
word.freq.all.1%	0	0	0	0	0	0.42	1.25	2.27	0.280656	0.25416	0	5.1	0
word.freq.3d.1%	0	0	0	0	0	0	0	0.04	0.065425	1.946447	0	42.81	0
word.freq.our.1%	0	0	0	0	0	0.38	1.49	2.94	0.312223	0.452273	0	10	0
word.freq.over.1%	0	0	0	0	0	0	0.63	1.21	0.095901	0.07498	0	5.88	0
word.freq.remove.1%	0	0	0	0	0	0	0.74	1.96	0.114208	0.153226	0	7.27	0
word.freq.internet.1%	0	0	0	0	0	0	0.6	1.62	0.105295	0.160858	0	11.11	0
word.freq.order.1%	0	0	0	0	0	0	0.65	1.33	0.090067	0.077627	0	5.26	0
word.freq.mail.1%	0	0	0	0	0	0.16	1.33	2.65	0.239413	0.41571	0	18.18	0
word.freq.receive.1%	0	0	0	0	0	0	0.38	1.02	0.059824	0.04062	0	2.61	0
word.freq.will.1%	0	0	0	0	0.1	0.8	2.18	4.1	0.541702	0.742524	0	9.67	0
word.freq.people.1%	0	0	0	0	0	0	0.61	1.44	0.09393	0.090623	0	5.55	0
word.freq.report.1%	0	0	0	0	0	0	0.2	1.49	0.058626	0.112348	0	10	0
word.freq.addresses.1%	0	0	0	0	0	0	0.17	1.61	0.049205	0.067	0	4.41	0
word.freq.free.1%	0	0	0	0	0	0.1	1.34	3.26	0.248848	0.681932	0	20	0
word.freq.business.1%	0	0	0	0	0	0	0.82	2.32	0.142586	0.197185	0	7.14	0
word.freq.email.1%	0	0	0	0	0	0	1.21	2.23	0.184745	0.282091	0	9.09	0
word.freq.you.1%	0	0	0	0	1.31	2.64	4.76	7.31	1.6621	3.152332	0	18.75	0
word.freq.credit.1%	0	0	0	0	0	0	0.32	2.54	0.085577	0.259862	0	18.18	0
word.freq.your.1%	0	0	0	0	0.22	1.27	3.17	5.26	0.809761	1.441944	0	11.11	0
word.freq.font.1%	0	0	0	0	0	0	0	6.75	0.121202	1.052175	0	17.1	0
word.freq.000.1%	0	0	0	0	0	0	0.73	1.62	0.101645	0.122701	0	5.45	0
word.freq.money.1%	0	0	0	0	0	0	0.57	1.31	0.094269	0.195926	0	12.5	0
word.freq.hp.1%	0	0	0	0	0	0	3.06	8.69	0.549504	2.793409	0	20.83	0
word.freq.hpl.1%	0	0	0	0	0	0	1.69	4.16	0.265384	0.78669	0	16.66	0
word.freq.george.1%	0	0	0	0	0	0	2.73	20	0.767305	11.33865	0	33.33	0
word.freq.650.1%	0	0	0	0	0	0	0.8	2.81	0.124845	0.290064	0	9.09	0
word.freq.lab.1%	0	0	0	0	0	0	0.5	2.32	0.098915	0.352036	0	14.28	0
word.freq.labs.1%	0	0	0	0	0	0	0.66	2.32	0.102852	0.208558	0	5.88	0
word.freq.telnet.1%	0	0	0	0	0	0	0.28	1.56	0.064753	0.162726	0	12.5	0
word.freq.857.1%	0	0	0	0	0	0	0	1.2	0.047048	0.107951	0	4.76	0
word.freq.data.1%	0	0	0	0	0	0	0.45	2.35	0.097229	0.309033	0	18.18	0
word.freq.415.1%	0	0	0	0	0	0	0	1.2	0.047835	0.108534	0	4.76	0
word.freq.85.1%	0	0	0	0	0	0	0.62	2.27	0.105412	0.283301	0	20	0

Appendix II

Boxplots (Appendix II) display the distributions of each predictor against label feature




word.freq.857
word.freq.technology
word.freq.1999
word.freq.parts
word.freq.pm
word.freq.direct
word.freq.cs
word.freq.george
word.freq.650
word.freq.lab
word.freq.conference
char.freq.semi
char.freq.lparen
word.freq.data
word.freq.415
word.freq.85
char.freq.lbrack
char.freq.bang
char.freq.dollar
word.freq.meeting
word.freq.original
word.freq.project
char.freq.hash
capital.run.length.avera
word.freq.re
word.freq.edu
word.freq.table
capital.run.length.long
0
1

Appendix III
Change in Accuracies Against Setting Various Training Data Sizes

Adaline – learning rate = 0.00001, no. of iterations =100

Dataset Sizes (training: testing)	Accuracy (Training Set)	Accuracy (Testing Set)
90:10 / 4140:461	0.8915459	0.8872017
80:20 / 3680:921	0.8937500	0.8946797
70:30 / 3220:1381	0.8987578	0.8877625
60:40 / 2760:1841	0.8960145	0.8984248
50:50 / 2300:2301	0.8995652	0.8970013
40:60 / 1840:2761	0.8972826	0.9022093
30:70 / 1380:3221	0.9130435	0.9060613
20:80 / 920:3681	0.9010870	0.9081771
10:90 / 460:4141	0.9130435	0.9060613

Logistic Regression - learning rate = 0.00015, no. of iterations =100

Dataset Sizes (training: testing)	Accuracy (Training Set)	Accuracy (Testing Set)
90:10 / 4140:461	0.9217391	0.92841649
80:20 / 3680:921	0.9192935	0.93051032
70:30 / 3220:1381	0.9223602	0.91600290
60:40 / 2760:1841	0.9170290	0.92123846
50:50 / 2300:2301	0.9230435	0.92133855
40:60 / 1840:2761	0.9163043	0.91923216
30:70 / 1380:3221	0.9173913	0.90823473
20:80 / 920:3681	0.9108696	0.91578375
10:90 / 460:4141	0.9173913	0.90823473

SVM – no. of folds = 10

Dataset Sizes (training: testing)	Accuracy (Training Set)	Accuracy (Testing Set)
90:10 / 4140:461	0.9366208	0.9544
80:20 / 3680:921	0.9365764	0.9392
70:30 / 3220:1381	0.9375782	0.9363
60:40 / 2760:1841	0.9331108	0.9332
50:50 / 2300:2301	0.9315629	0.9305
40:60 / 1840:2761	0.9278449	0.9294
30:70 / 1380:3221	0.9302899	0.9267
20:80 / 920:3681	0.9137177	0.9218
10:90 / 460:4141	0.9139751	0.9065

Reference

Drucker, H., Wu, D., & Vapnik, V. (1999). Support vector machines for spam categorization. *IEEE Transactions on Neural Networks,10*(5), 1048-1054. doi:10.1109/72.788645

Ray, Sunil. "Understanding Support Vector Machine Algorithm from Examples (along with Code)." Analytics Vidhya. Analytics Vidhya, 13 Sept. 2016. Web. 17 Mar. 2017. <https://www.analyticsvidhya.com/blog/2015/10/understaing-support-vector-machine-example-code/>.

Scikit-learn. "Non-linear SVM." Scikit-learn. Scikit-learn Developers, n.d. Web. 17 Mar. 2017. <http://scikit-learn.org/stable/auto_examples/svm/plot_svm_nonlinear.html>.

Rae, Jack. "What Are the Advantages of Logistic Regression over Decision Trees?" What Are the Advantages of Logistic Regression over Decision Trees? - Quora. N.p., 20 May 2015. Web. 17 Mar. 2017. <https://www.quora.com/What-are-the-advantages-of-logistic-regression-over-decision-trees>.

Raschka, Sebastian. “Single-Layer Neural Networks and Gradient Descent” Web. 24 May. 2015 <http://sebastianraschka.com/Articles/2015_singlelayer_neurons.html>

Robinson, Nick. "The Disadvantages of Logistic Regression." Classroom. Synonym.com, n.d. Web. 17 Mar. 2017. <http://classroom.synonym.com/disadvantages-logistic-regression-8574447.html>.

Xiong, Jinjun. (2017) Machine Learning: Concepts & Application: Lecture 2 - Data Visualization and Exploration, Columbia University Course, NYC NY.

Xiong, Jinjun. (2017) Machine Learning: Concepts & Application: Lecture 3 - Data Preprocessing and Machine Learning, Columbia University Course, NYC NY.

Xiong, Jinjun. (2017) Machine Learning: Concepts & Application: Lecture 7 - Perceptron & Logistic Regression, Columbia University Course, NYC NY.

Xiong, Jinjun. (2017) Machine Learning: Concepts & Application: Lecture 8 - Support Vector Machines , Columbia University Course, NYC NY.