Stats503HW4

David Li

March 18, 2018

Data Report for Stats503 Homework 4

– Introduction: Classification w/ Machine Learning –

A short introduction on classification was heeded in the last Data Report (3). This data report is still focused upon classification techniques, but this time with more machine-learning involved techniques such as Support Vector Machines, Decision Trees, and Neural Networks.

– Data & Procedure –

The dataset contains 4601 email messages, with 57 attributes based on the messages themselves (such as particular word frequencies, length of character chains, etc). These 57 attibutes are then used to tune the machine learning algorithm and predict whether the message was likely a spam (i.e. unsolicited intent) message. A final 58th column indicates the true spam / not spam status of the message. The dataset can be found at the UCI Machine Learning Repository website: https://archive.ics.uci.edu/ml/datasets/spambase.

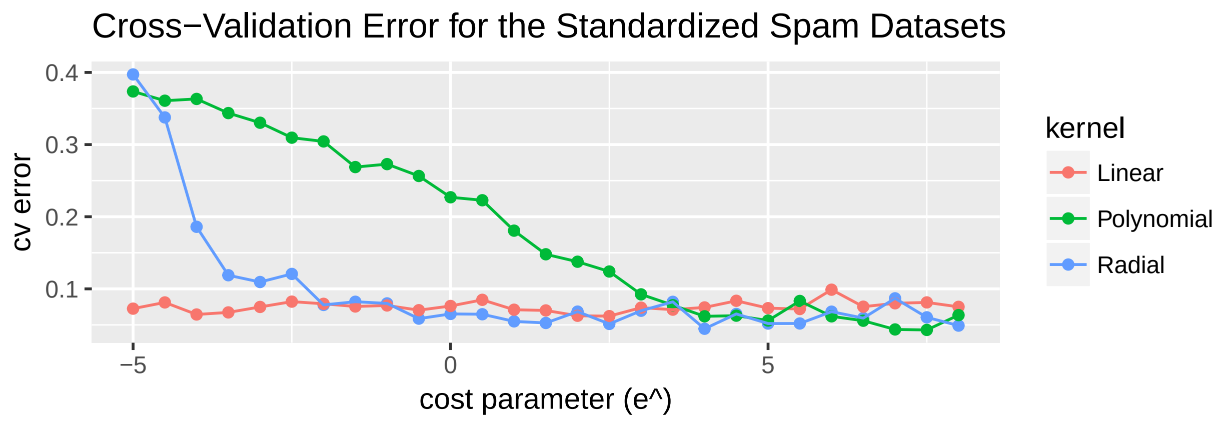
Our analysis also will be performed on a "balanced" version of training and test data as well as an "imbalanced" version of training and test data, split into two parts in this report. Aside from performing the analyses on the data of interest, we may be interested in the performance of the algorithms in the case that the data is constructed differently hence the motivation to compare "balanced" and "imbalanced" versions of the dataset. In general for comparing performance, we will be commenting and comparing classification error rates for the test dataset as well as cross-validation error.

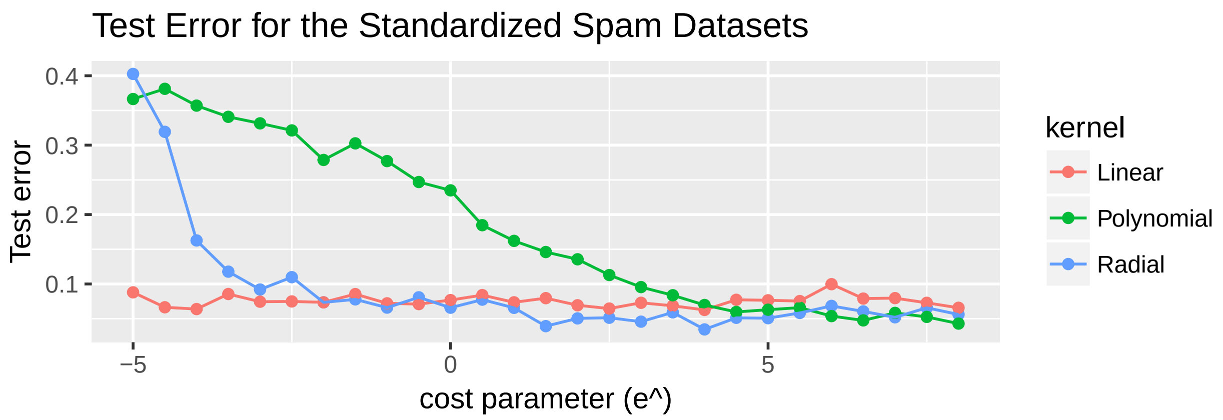
Part 1: Balanced Training / Test Data

– Support Vector Machines (known as SVM) –

We are interested in comparing the performance of cross-validation error and testing error when applying SVM on multiple different varying parameters. For example, the kernel type can describe the desired similarly function to run the SVM on, providing a basis of model for the algorithm to run. In another case, the cost parameter (which allocates how much allowed slack there is in controlling misclassification) can dictate the choice of the hyperplane in classifying the training points correctly. In large values, it chooses a smaller margin for the hyperplane. In small values, it looks for a larger-margin hyperplane which may cause misclassification.

As a measure for more sensible data to perform on, we standardized all columns of the features with mean 0 and variance 1 except for the final class column. The following below figures are mean't to show the comparison of the Cross-Validation Error in combination with Test Error for different Kernels at different cost parameter levels.





We can immediately see that the polynomial kernel usage tends to have very high test classification error and high cross-validation error. Radial begins at approximately the same level of error (with starting small cost parameter) for both CV and Test as Polynomial, but quickly drops to the levels of error of a linear kernel as we increase the cost parameter. By eˆ4, all the CV and Test levels between the different kernels level out to around the same error rate of < 0.1. Linear kernel performance is fairly consistent at a low CV and test error rate. Therefore we can conclude that choice of kernel is a huge impact on the pattern of error rates, the cost parameter generally causes high error rates at our smaller cost parameter values but gradually lowers these error rates as we increase the cost parameter value.

Cross-validation was able to assist us, as we can see the lowest error rates for Cross-Validation as well as Testing error rates. We can deduce that the cost parameters that yield in correspondence to these lowest error rates would be the ideal choice of tuning parameters for the SVM procedure.

– Neural Networks –

Neural networks are another form of classification that can be used for the spam dataset. The foundation of neural networks is built upon the idea of connected nodes that “talk” to each other by passing information. For our analysis, we will retrieve the overall cross-validation error and test error by varying the parameters of the number of hidden layers and the hidden nodes per layer. More specifically, the 4 examples we will compare involve 1 layer with either 3 versus 4 hidden nodes as well as 2 layers with either 3, 4 hidden nodes per layer or 3, 5 hidden nodes per layer. We observe the retrieved Cross-Validation errors and test dataset errors below as follows:

Table 1: Neural Net: CV and Test Error Rates for the Standardized Spam Datasets

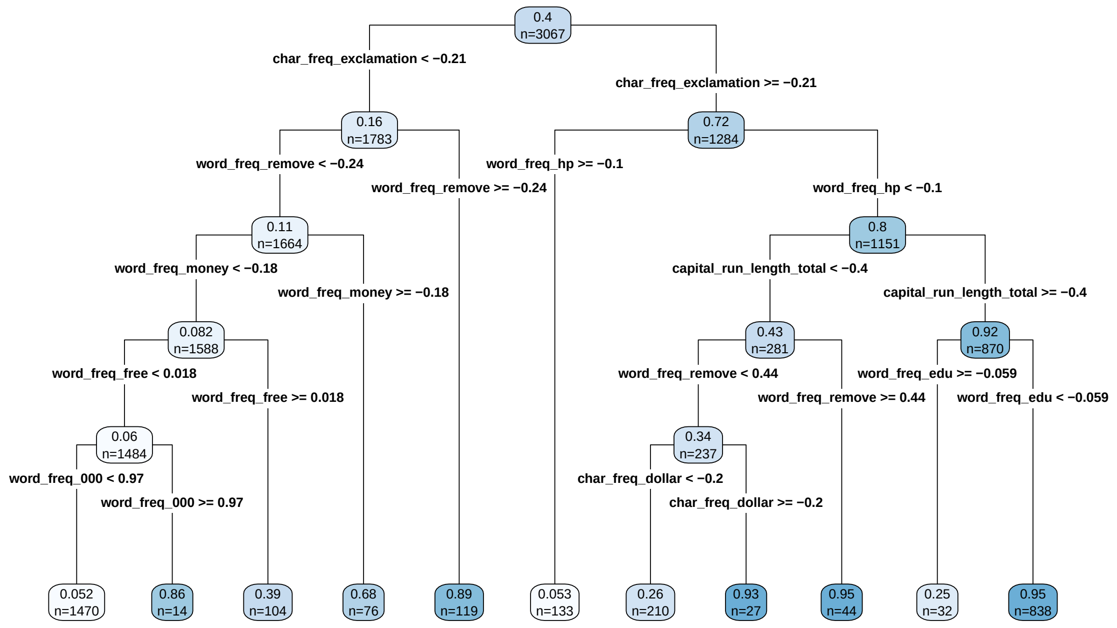
	Cross-Validation Error	Test Error
1 Hidden Layer, 2 Nodes	0.0270650562	0.0573663625
1 Hidden Layer, 3 Nodes	0.0185827165	0.0599739244
2 Hidden Layers, [2,2] Nodes	0.0397760786	0.0645371578
2 Hidden Layers, [2,3] Nodes	0.0401087193	0.0743155150

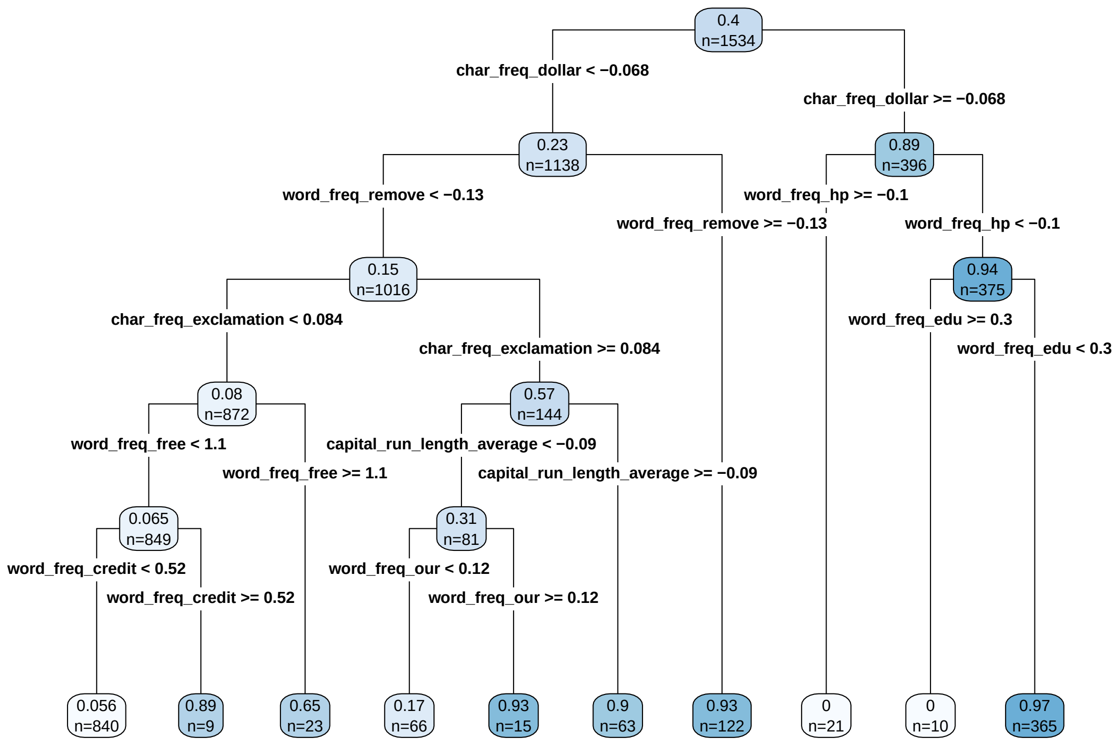
We can observe that in either 1 or 2 layers, the number of nodes does impact the CV and test error rates. More specifically, it looks to generally reduce the CV and test error rates. Though it looks like 2 layers has more error rate convincingly than 1 layer, the threshold was changed to 0.05 for computational time constraints so more different combinations of layers and nodes should be done to arrive at a conclusive answer about varying the number of layers. For now, it is evident that there is always a relative decrease in both kinds of error rates from increasing the number of hidden nodes. Intuitively one may think that with more layers then the classification may be better with more components being provided to the neural net algorithm. The interesting comparison will be to a lopsided spam to not spam ratio aka an imbalanced dataset done later in this report.

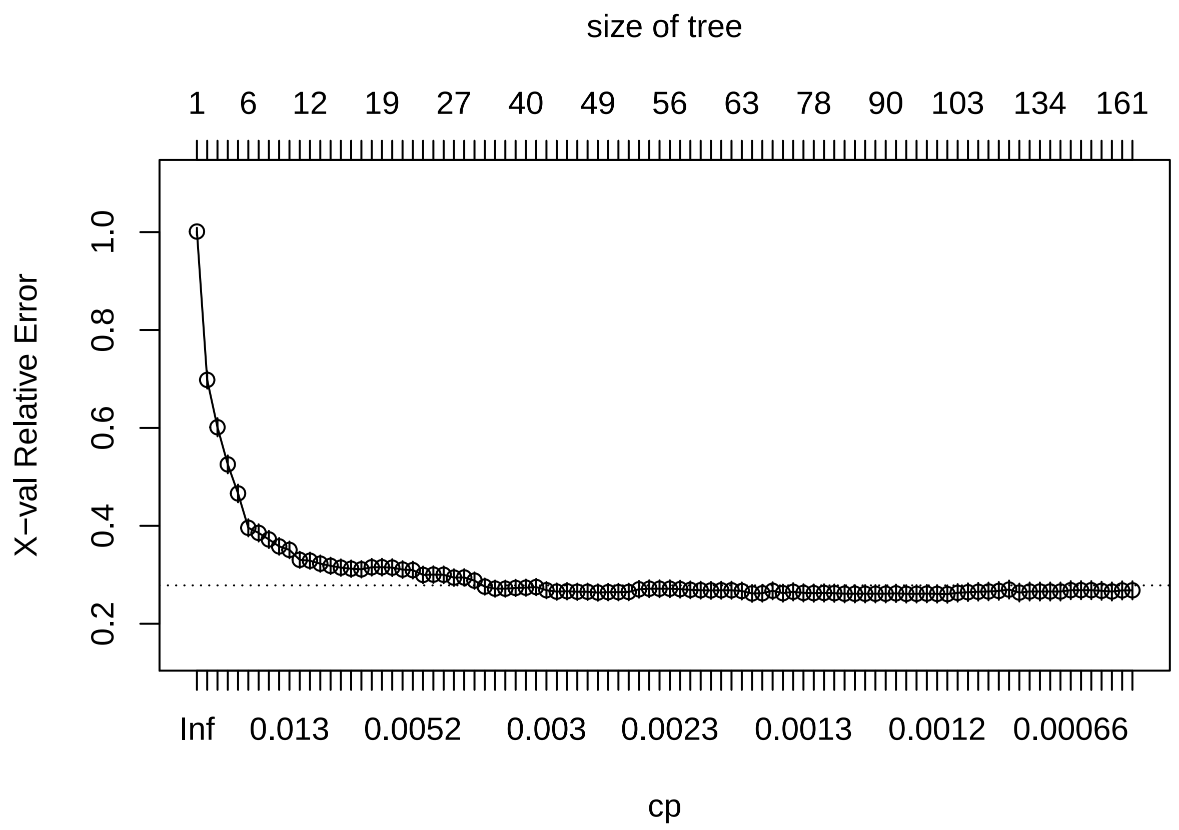
– Decision Trees –

Third and finally, we have Decision Trees which are the most interpretable and simple model for classification. Based on the distribution of the classifications in respect to all of the features, splits in the tree are determined by cutoffs that dictate a strong difference in features. On a more theoretical level, we can consider the complexity of the model as well as the error rates in regard to the tree size (sometimes alternatively thought of as the number of splits).

The complexity parameter (cp) is used to control the size of the decision tree and to select the optimal number of splits in the tree. We want to choose an optimal number of splits by balancing the amount of error along with the model complexity and cp value. After careful observation of each tree size and the associated cp and error values, it is reasonable to keep as many tree splits as possible whilst we are getting more useful information about the divisions in the model. Having too many tree splits creates a giant tree that is difficult to interpret. Some results of analysis are presented below:


0.4
n=3067
char_freq_exclamation < −0.21
char_freq_exclamation >= −0.21
0.16
n=1783
word_freq_remove < −0.24
word_freq_remove >= −0.24
0.72
n=1284
word_freq_hp >= −0.1
word_freq_hp < −0.1
0.11
n=1664
word_freq_money < −0.18
word_freq_money >= −0.18
0.8
n=1151
capital_run_length_total < −0.4
capital_run_length_total >= −0.4
0.082
n=1588
word_freq_free < 0.018
word_freq_free >= 0.018
0.43
n=281
word_freq_remove < 0.44
word_freq_remove >= 0.44
0.92
n=870
word_freq_edu >= −0.059
word_freq_edu < −0.059
0.06
n=1484
word_freq_000 < 0.97
word_freq_000 >= 0.97
0.34
n=237
char_freq_dollar < −0.2
char_freq_dollar >= −0.2
0.052
n=1470
0.86
n=14
0.39
n=104
0.68
n=76
0.89
n=119
0.053
n=133
0.26
n=210
0.93
n=27
0.95
n=44
0.25
n=32
0.95
n=838


0.4
n=1534
char_freq_dollar < −0.068
char_freq_dollar >= −0.068
0.23
n=1138
0.89
n=396
word_freq_remove < −0.13
word_freq_remove >= −0.13
word_freq_hp >= −0.1
word_freq_hp < −0.1
0.15
n=1016
0.94
n=375
char_freq_exclamation < 0.084
char_freq_exclamation >= 0.084
word_freq_edu >= 0.3
word_freq_edu < 0.3
0.08
n=872
0.57
n=144
word_freq_free < 1.1
word_freq_free >= 1.1
capital_run_length_average < −0.09
capital_run_length_average >= −0.09
0.065
n=849
0.31
n=81
word_freq_credit < 0.52
word_freq_credit >= 0.52
word_freq_our < 0.12
word_freq_our >= 0.12
0.056
n=840
0.89
n=9
0.65
n=23
0.17
n=66
0.93
n=15
0.9
n=63
0.93
n=122
0
n=21
0
n=10
0.97
n=365

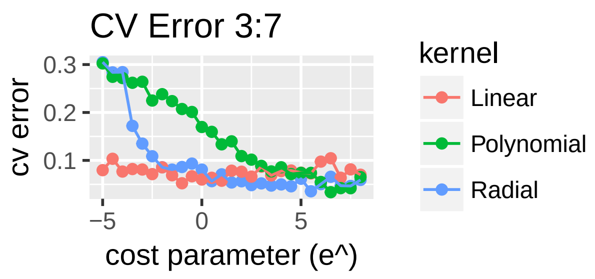


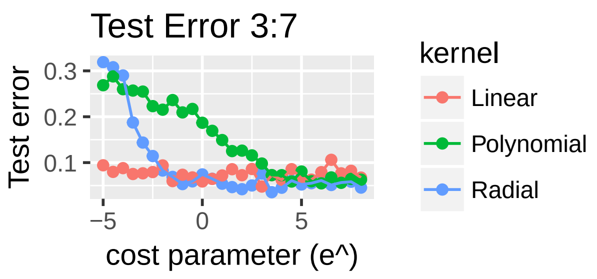
From an intuitive standpoint, a tree size of 6 (or 6 tree splits) looks to reduce the amount of error decently whilst still keeping the model simple and easy to interpret. Regardless, it looks that after about a tree size of 10 then the error is reduced at negligble rates. Before a tree size of 10, the amount of information gained from constructing the tree is vastly large and thus it is worth continuing to expand since also extremely small tree sizes will likely have moderate misclassification rates. As we look in increasing tree size, the error decreases exponentially; first in large quantities, then in small quantities.

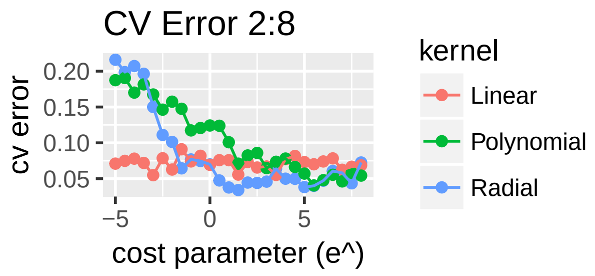
Part 2: Imbalanced Training / Test Data

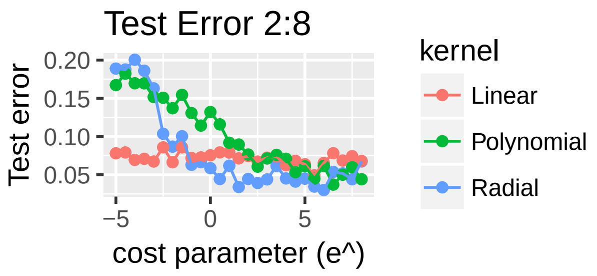
– Support Vector Machines (known as SVM) –

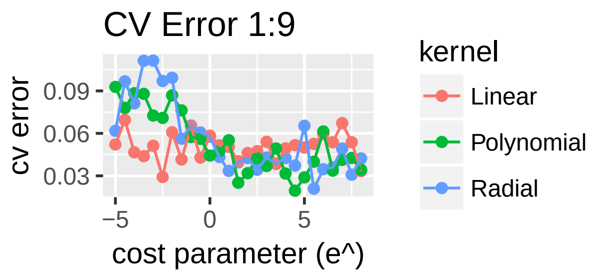
In the previous part, we had assumed that the datasets (both training and test) were balanced; that is, they had approximately an equal amount of spam to non-spam within the dataset. We may consider the impact on the patterns of test error and Cross-validation error from Part 1 of the balanced datasets when we choose to deliberately lopside the ratio of spam to non-spam; in other words, assessing the robustness of these techniques. For instance, what if we have ratios of spam to non-spam for 3:7? 2:8? 1:9? We investigate the changes in Cross-validation error and test error with the figures below. Note that the scales of the graphs are different, in order to promote visibility.

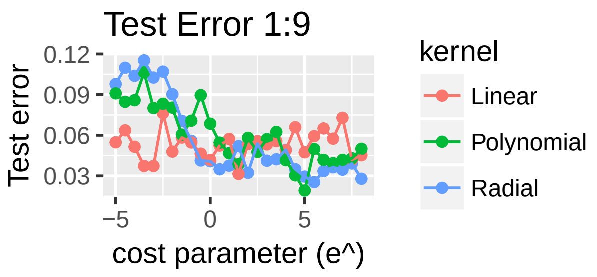






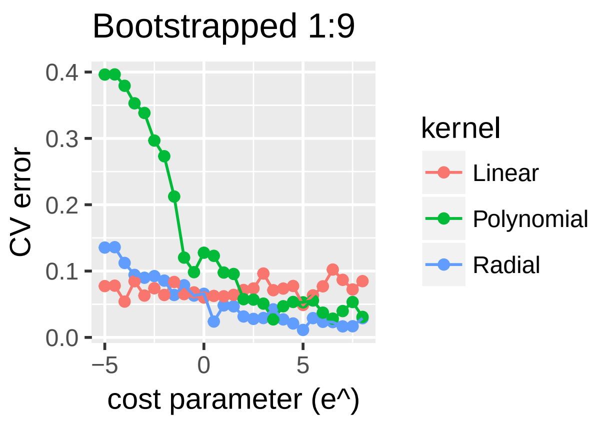


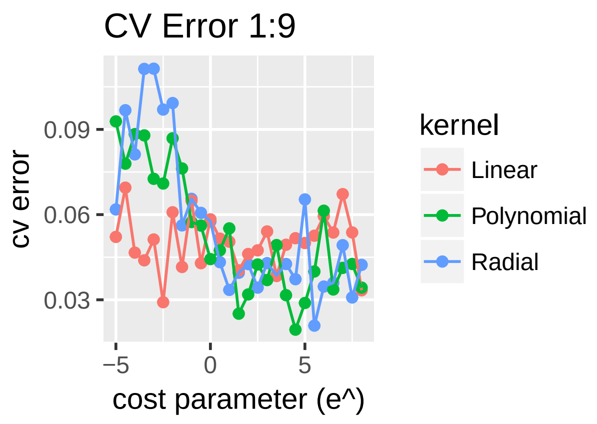


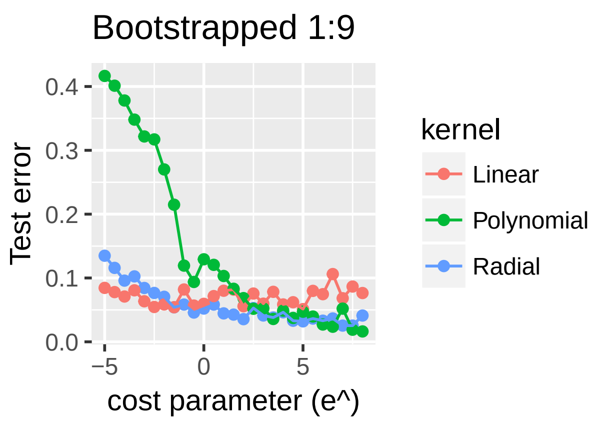


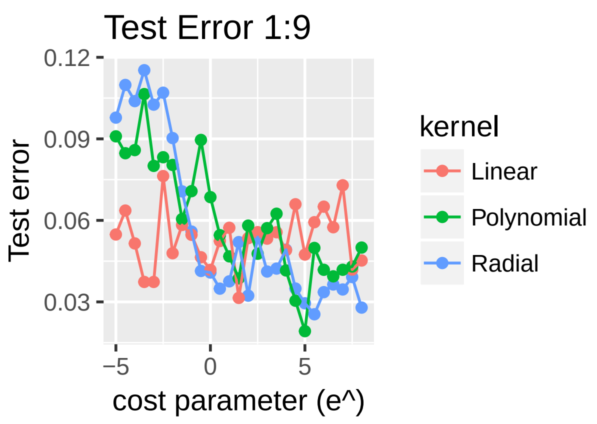
The most immediate and striking fact is that as we reduce the ratio of spam to non-spam, the overall CV error and test error decreases significantly regardless of kernel type and cost parameter delegation. For example, the range of errors are from 0 to 0.3 in the 3:7 ratio of spam, but the range of errors are from 0 to 0.1 in the 1:9. Both error type rates are fairly low and consistent with the linear kernel except at higher increasing cost parameters, but there is an interesting difference regarding the polynomial and radial kernel types. As the spam to non-spam ratio decreases, the radial kernel begins to perform more relatively poorly as compared to the polynomial kernel at low cost parameter values. For example, the graphs for spam ratio 1:9 show that the radial kernel performed significantly more poorly in both test error and CV error in the lower cost parameter values. Additionally, the linear kernel performance in the high cost parameters for the low spam ratios (such as 1:9) is significantly worse than the other two kernels and is higher for both CV and test error. As seen before, the same pattern of increasing cost parameter lowering CV and test error for all kernel types still holds in these figures.

Finally, we may consider the effect of bootstrapping on CV and test error rates, for the purposes of regenerating the underrepresented class for more instances of such. We present only the figures of CV error and test error for bootstrapping the imbalanced 1:9 spam to non-spam ratio side-by-side with the original imbalanced CV error and test error figures.









It may seem alarming that the Bootstrapped figures show extremely high CV and test error in the start at low cost parameters, but it quickly tapers off to low error rates as we increase the cost parameter. The catch seems to be that the CV and test error rates for the bootstrapped version converges more quickly to lower error levels than the non-bootstrapped error rates at the same comparable cost-parameter levels. There is also a large significant difference in the pattern of the error rates for the radial kernels, as the bootstrapped error rates are more controlled and have less frantic oscillations compared to the non-bootstrapped versions as we increase the cost parameter.

– Neural Networks –

We follow the same analysis as the balanced datasets, but will only explicitly present results from the 1:9 imbalanced dataset of spam:non-spam. The following results:

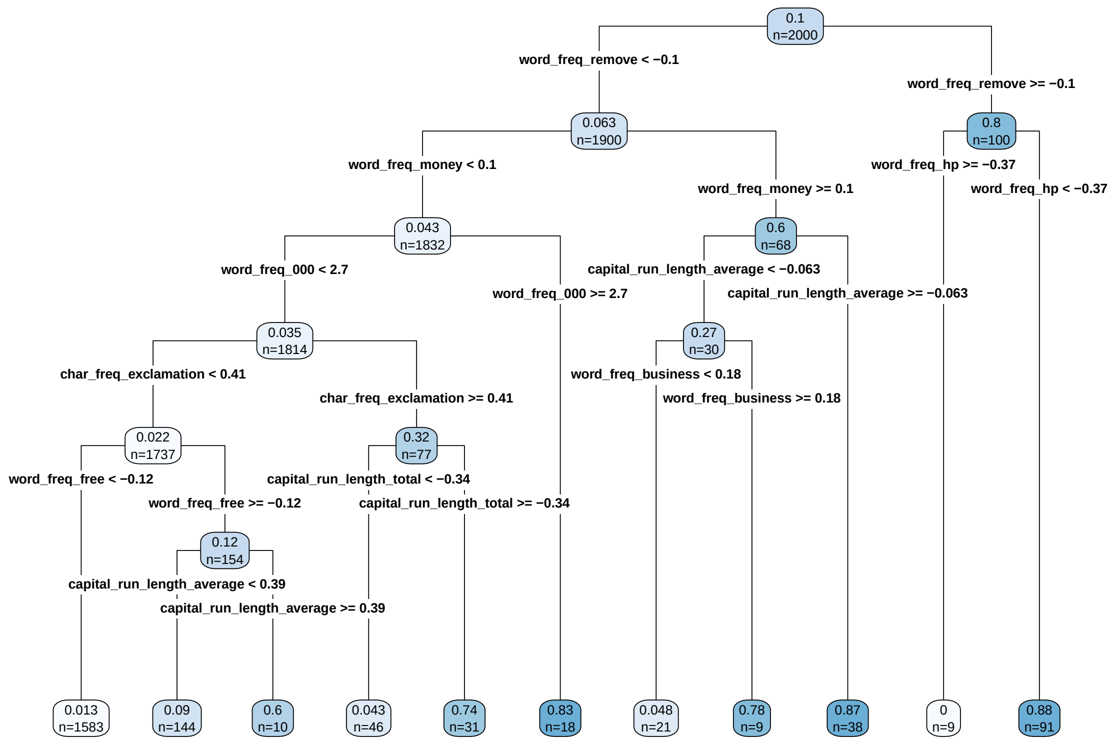
Table 2: Neural Net: CV and Test Error Rates for the Imbalanced 1:9 Spam Dataset

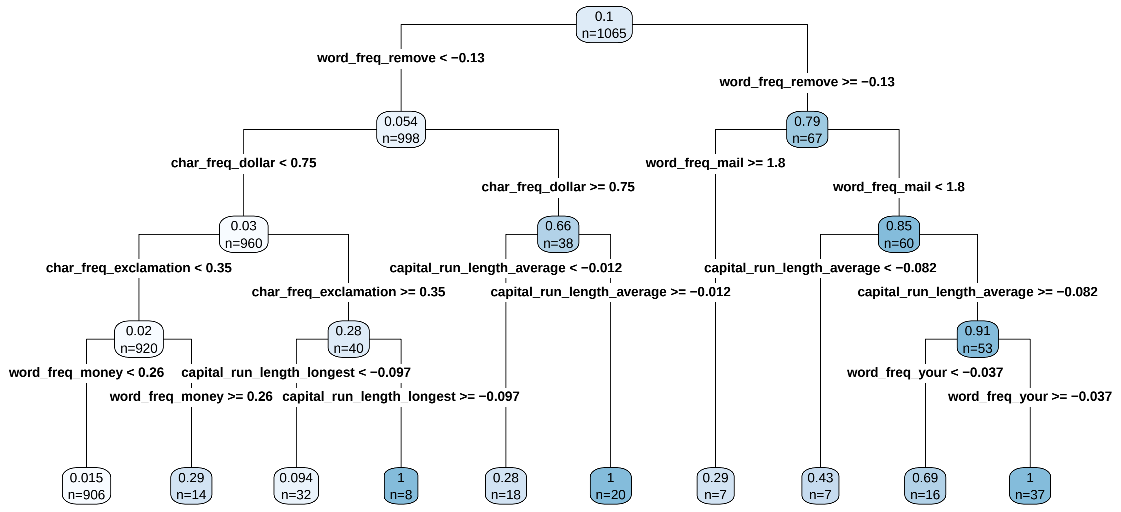
	Cross-Validation Error	Test Error
1 Hidden Layer, 2 Nodes	0.0120	0.0338028169
1 Hidden Layer, 3 Nodes	0.0075	0.0272300469
2 Hidden Layers, [2,2] Nodes	0.0205	0.0422535211
2 Hidden Layers, [2,3] Nodes	0.0235	0.0356807512

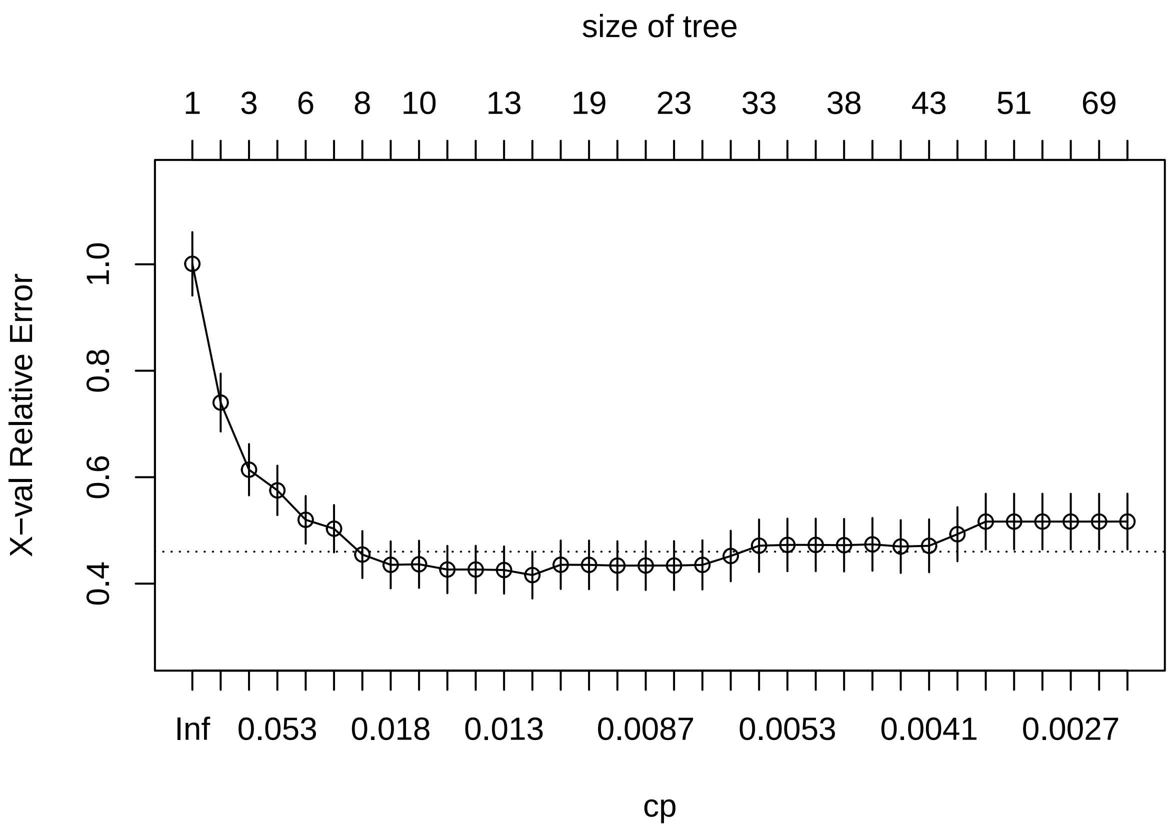
Interestingly, the Cross-Validation and Test errors (when varying the number of nodes) did not change much when performing the neural net analysis on the Imbalanced dataset. More simulations should be done if this is a coincidence, but there is a plausible connection and explanation to the very low occurence rate of spam within these imbalanced dataset.

Bootstrapping can identically be done for the neural network analysis, but we will infer based upon the pattern of results from the bootstrapping in the SVM that to add more instances of the underrepresented class will alleviate the consequences of an imbalanced dataset. Intuitively, bootstrapping can reduce variance and avoid overfitting and thus improve accuracy and stability of the machine learning procedure (and save computational stress). We would probably expect an increase in classification performance if we had applied bootstrapping, potentially lowering overall errors.

– Decision Trees –







This time, it took more tree splits before the error rates began to converge (about 12 before negligible error decreases). This intuitively would make sense since if there is much less spam data within the imbalanced set, the decision tree algorithm must process more data (with a lower ratio of spam!) and thus perform more tree splits before it is convinced that the error can hardly be reduced any further. Thus the tree had to be bigger to compensate and capture the classification info and divisions accurately.

Bootstrapping in our case would likely increase classification performance as consistent with the other classification techniques as reducing variance and improving accuracy would lead to stronger and more reliable tree models. Note that this does not necessarily mean that the tree size will get smaller, as more complicated models will understandably require more complicated trees so less model complexity is not always the prime goal.