Danmarks Tekniske Universitet

Report 2

Course:
02450

Dataset: SPAM E-mail Database (Spam)

Author(s):
Claus Michael Oest Lensbøl (s132308)
Stephan Thordal Larsen (s146907)

DTU Compute
Institut for Matematik og Computer Science

02450
Morten Mørup

3 November 2015

Contents

1 Introduction **1**

2 Regression **2**
2.1 Linear Regression 2
2.2 Artificial Neural Network 4
2.3 Comparison 5

3 Classification **7**
3.1 Analysis on the data 7
3.2 Analysis of the classification methods 9
3.3 Comparison of methods 9

4 Conclusion **10**

1 Introduction

As a continuation of the first report for the course 02450, *Introduction to Machine Learning and Data Mining* at DTU, this document describes the process of creating a regression and classification analysis of the SPAM dataset from `http://statweb.stanford.edu/~tibs/ElemStatLearn/`. The report will describe, compare and discuss the results of several different methods used both for regression and classification. The problem to be solved by regression, is the prediction of the variable `capital_run_length_total`, which is the number of total capital letters in the email. The classification will identify spam from non-spam. K-fold cross validation will be used in this report to measure the quality of the results generated.

2 Regression

Since frequent use of capital letters is a key trait of spam mails, the chosen regression problem to be solved, is the prediction of the 57th variable, `capital_run_length_total`, which is the total number of capital letters in each email.

2.1 Linear Regression

Before running linear regression and feature selection on the dataset, all parameters have been normalized. This is to ensure that the magnitude of the coefficients can be analyzed. By running linear regression on the data, with 10-fold cross validation, and 10-fold internal cross validation for sequential feature selection. We get the feature selection result seen in *figure 1*, The selected features indicate that `$`-signs and text strings such as `re`, `000`, `money` are good predictors of the amount of capital letters in an email.

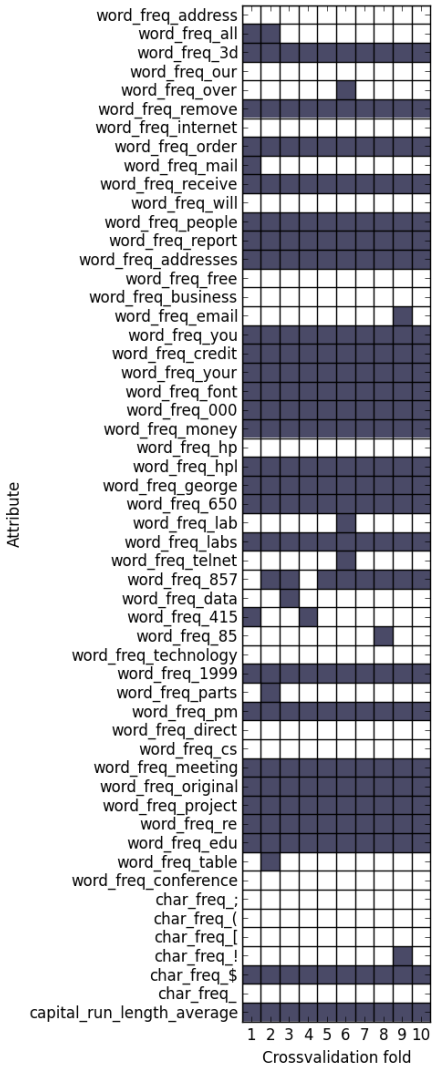

Figure 1: Selected features, from a linear regression run with 10-fold crosse validation.

By looking at the residual error, one finds that the residual error is highest on low values of

all the attributes, and the larger the values of the parameters get, the lower the error becomes. See *figure 2*.

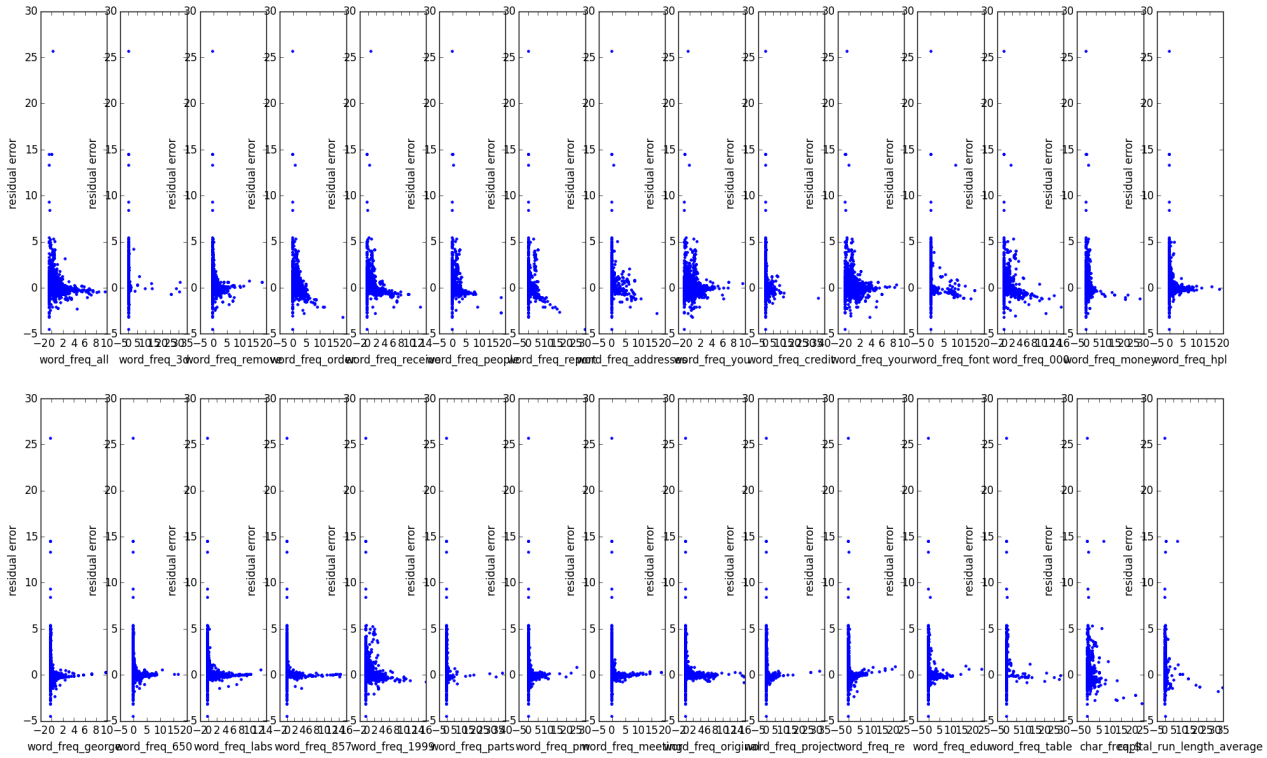

Figure 2: Residual error on selected features.

But comparing the resulting errors from running with feature selection versus running without, the difference seems negligible, see *table 1* .

Measure	Without Feature Selection	With Feature Selection
Training Error Mean Squared	0.80217	0.82299
Test Error Mean Squared	0.84734	0.84176

Table 1: Feature selection errors on training and test data.

What we do get with feature selection, is a less complex model, which is also less prone to overfitting. With about half of the attributes, the complexity is drastically reduced. By inspecting the linear model, *table 2*, we see what attributes affect the predicted value of total capital letters (`capital_run_length_total`). This indicates that frequent occurrences of `order`, `report` and `$` results in many capital letters, whereas frequent occurrences of `remove`, `edu` and `re` results in less capital letters.

Attribute	Coefficient Value
word_freq_3d	0.00689388
word_freq_remove	-0.05189366
word_freq_order	0.16971404
word_freq_receive	0.06606249
word_freq_people	0.05359396
word_freq_report	0.12692896
word_freq_addresses	0.06479205
word_freq_you	-0.05170948
word_freq_credit	0.01743045
word_freq_your	-0.0463832
word_freq_font	0.10282471
word_freq_000	0.05129523
word_freq_money	0.03745141
word_freq_hpl	-0.01895157
word_freq_george	-0.06980303
word_freq_650	-0.02765476
word_freq_labs	-0.03253546
word_freq_857	0.02156261
word_freq_1999	0.03873031
word_freq_pm	-0.02447841
word_freq_meeting	-0.0371604
word_freq_original	-0.01822303
word_freq_project	-0.02589949
word_freq_re	-0.05371097
word_freq_edu	-0.02800611
char_freq_$	0.12146869
capital_run_length_average	0.12147069

Table 2: Linear regression model attributes and coefficients, from fold 10.

2.2 Artificial Neural Network

As with Linear Regression all values were normalized before training the Artificial Neural Network, ANN. The ANN was run inside the same 10 cross validation folds as the Linear Regression to ease comparability. The ANN was trained with two hidden units and 5 networks trained each fold.

The resulting squared errors and training errors can be found in *figure 3*, with an average mean squared error of 0.76498.

The resulting error is very high, and by looking at the difference between the test data and the predicted values, see *figure 4*, one sees that the ANN often misses the test values with it's estimations.

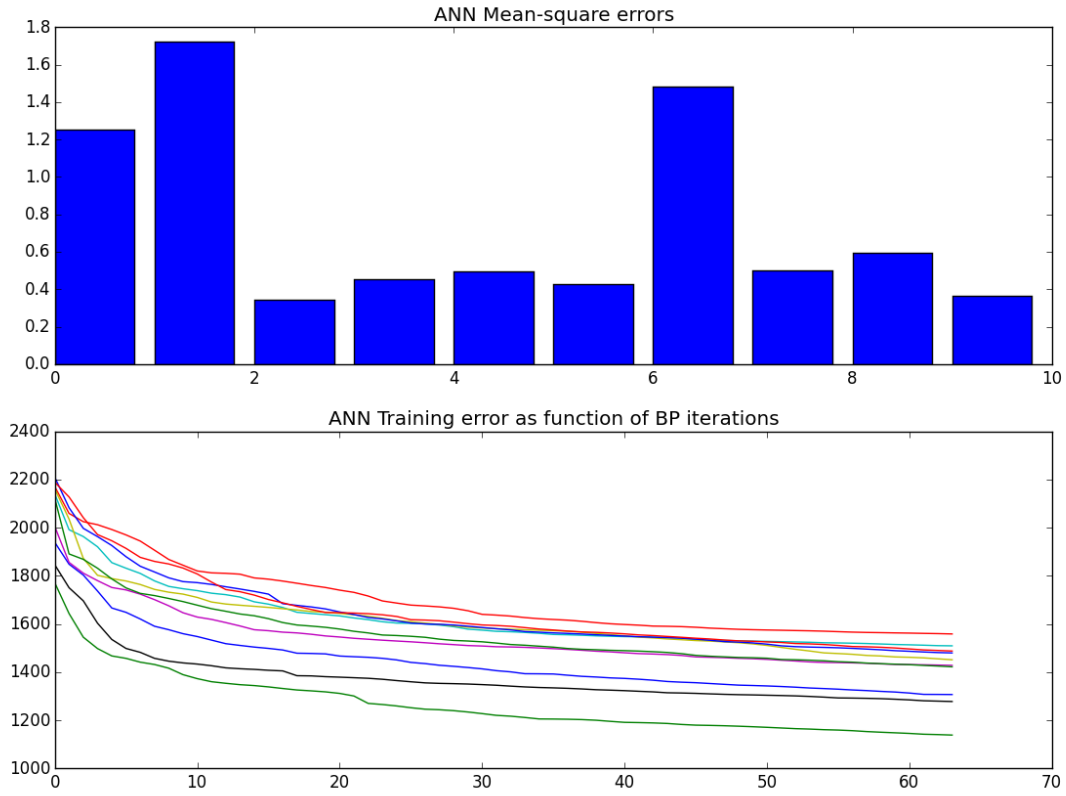

Figure 3: Errors from each fold of the Artificial Neural Network.

2.3 Comparison

To compare the Artificial Neural Network and Linear Regression a paired t-test was run on the mean squared errors of the ANN and the mean squared test errors from the Linear Regression. With a t-value of 2.89173 and corresponding p-value of 0.01783, there is a strong indication that the Artificial Neural Network is a better fit for regression on the dataset.

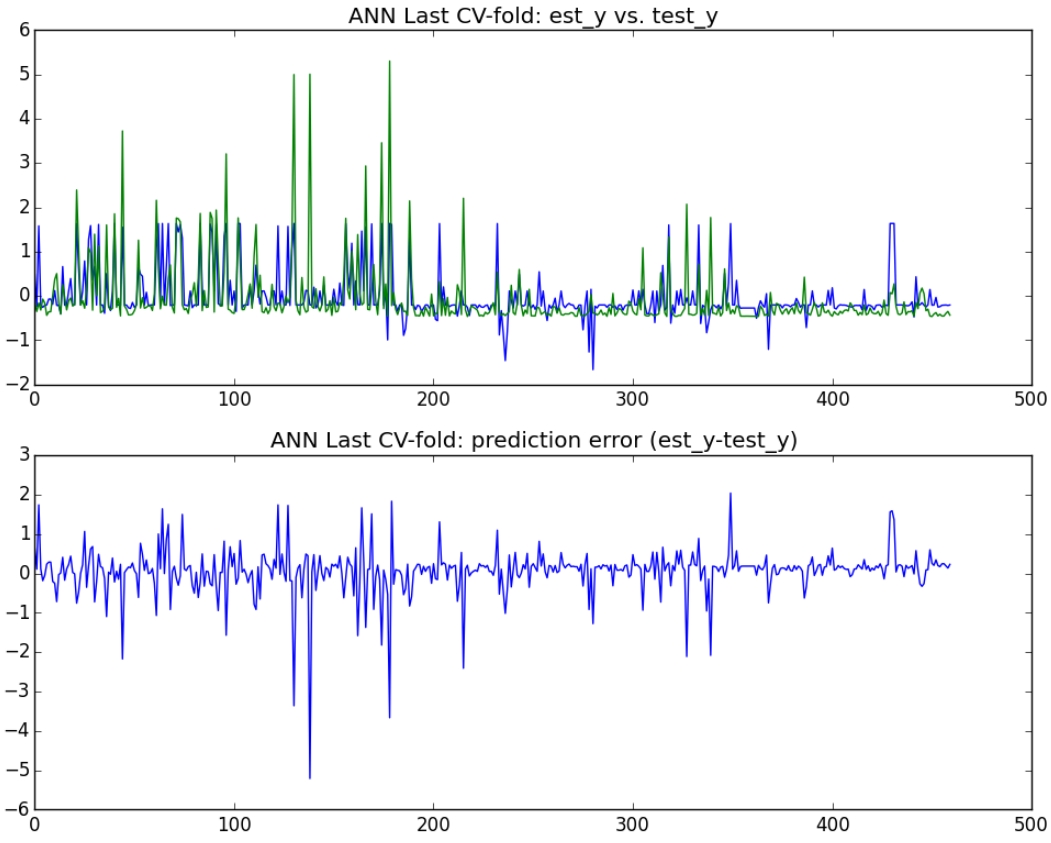

Figure 4: Comparison between estimated values and test values, from the Artificial Neural Network.

3 Classification

The problem chosen to solve is trying to predict the 58th attribute, the binary attribute indicating *Not-Spam*[0] and *Spam*[1].

The earlier test on the data described in the previous report, gives a classification rate of 7%. The method used to get to these 7% is unfortunately unknown.

3.1 Analysis on the data

When analyzing the data the three methods chosen were a *Decision Tree*, *K Nearest Neighbor*, and an *Artificial Neural Network*.

The data was to be analyzed using two level cross validation, selecting parameters in the inner fold. Here 10-fold was used in both the outer and inner fold.

Selecting parameters for the decision tree was based off of the tree depth ranging from $1-20$. Several static tests with depths up to 200 levels showed the optimal depth to be in the range of $1-5$ nodes, and thus the range was chosen to cover only up to 20 levels due to computational limitations. In this case every inner fold had the range of tree depth tested, and optimal levels found, resulting in $10 \cdot 10 \cdot (20+1) = 2100$ trees fitted. Figure 5 shows a boxplot for the optimal levels for each fold.

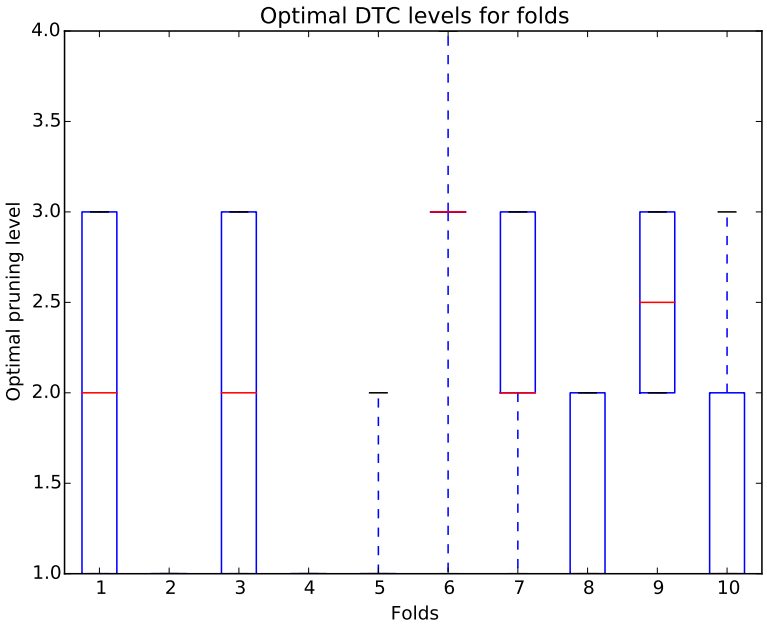

Figure 5: Optimal pruning levels of the decision trees in each of the ten out folds.

The error rate based on the depth for each fold can be seen in figure 6. The figure shows a high level of classification errors, with the mean error rate of the tests at 45.5%.

The data for the K-Nearest Neighbor was collected in the same way as the decision tree. The data can be seen in figure 7, showing significantly better results than the decision tree with a mean of the minimum test error at only 8.54%.

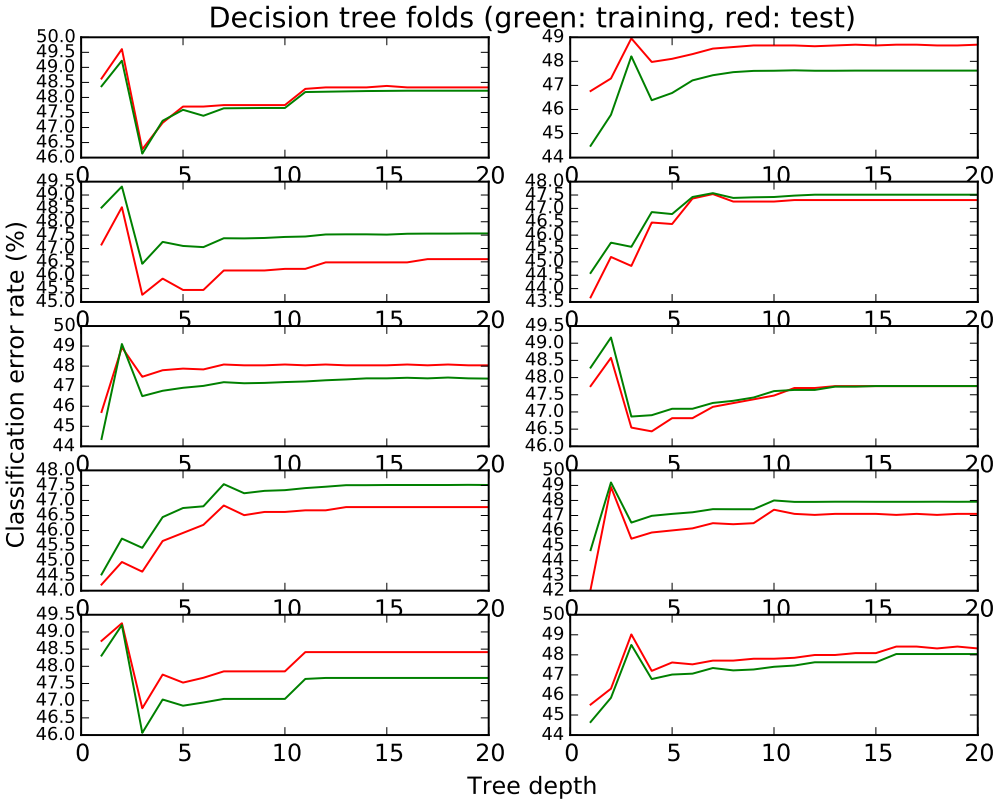

Figure 6: High training and test error rates in the 10-folds for the decision tree. Mean of tests is 45.5% making this method very unsuitable for classification of this dataset.

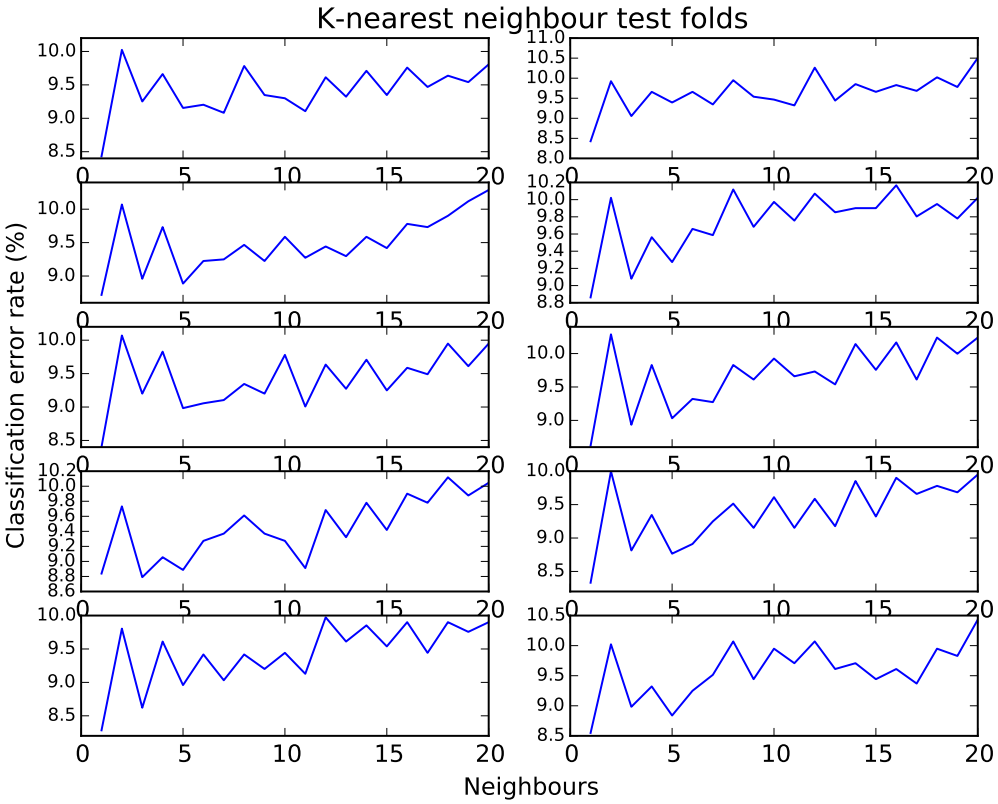

Figure 7: Test error rates for each of the folds. Mean of the minimums is 8.54%.

Lastly an artificial neural network was trained with 10 networks per fold, giving a slightly lower test error rate at only 6.93%. Figure 8 shows the error rate for each hold, and figure 9 shows the errors for the last fold.

3.2 Analysis of the classification methods

Each of the models is based on vastly different methods of classifying the data.

For the decision tree, the two main or root factors were *char_freq_!* and *char_freq_$*. Figure 10 shows the 10 trees fitted from the model in each of the 10 outer folds. Another "noticeable" classifier was *word_freq_remove* that is presented in four of the trees.

For the K-Nearest Neighbor classification figure 7 indicates that only a small number of neighbors should be considered, in most cases only the single closest neighbor.

Lastly for the Artificial Neural Network it is very hard to interpret how the network classifies the data. For this reason the classification done with the network has not been examined.

3.3 Comparison of methods

When comparing the K-Nearest Neighbor and the Artificial Neural Network with a paired t-test the t-value becomes 5.5 and the corresponding p-value is 0.0004. This indicates an improvement using the Neural Network over the Nearest Neighbor method.

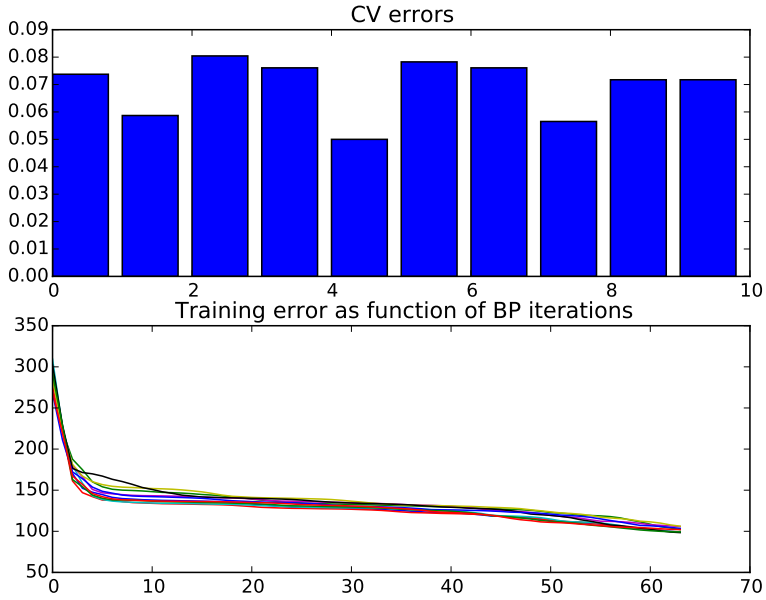

Figure 8: Training error for each of the 10 folds.

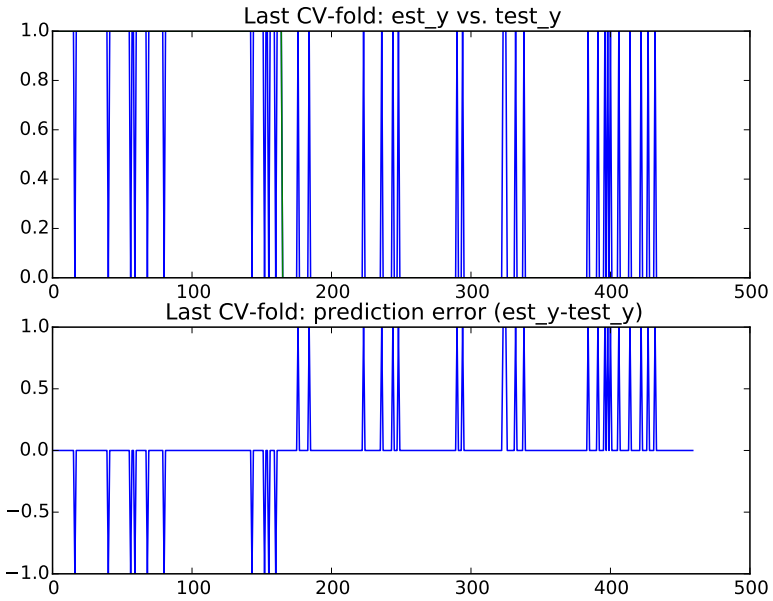

Figure 9: Comparison of the estimated values and the actual values for the last fold.

4 Conclusion

Using regression, it is hard to predict the *capital_run_length_total* property from the other properties of the email. The ANN for regression gives a mean squared error of 0.76, and the linear regression model gives a mean squared error of around 0.84 with and without feature selection. Hereby the ANN is best for regression, but still not good enough for actual use.

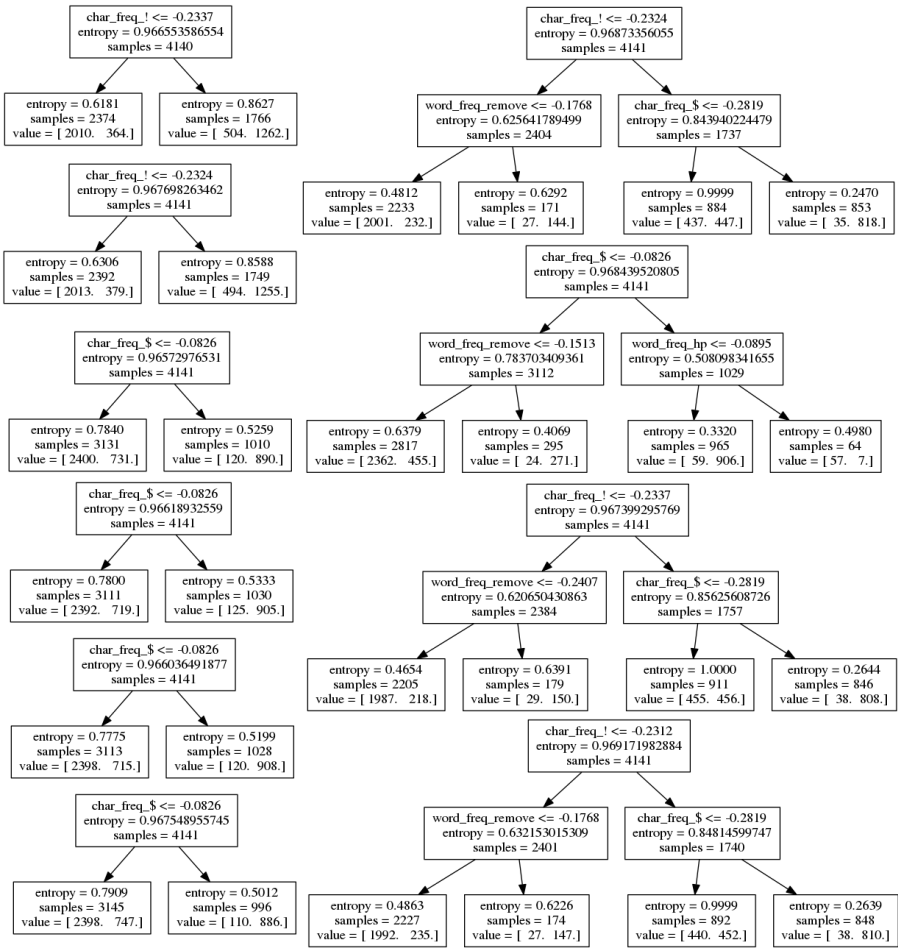

Figure 10: Comparison of the estimated values and the actual values for the last fold.

Doing classification using a decision tree is not usable either, but with the K-Nearest Neighbor and the Neural Network resulting in error rates as low as about 7%, these methods seem closer to earlier attempts of classification on the dataset.

If the unsupervised tools can give an equally low error rate, it might be possible to use this method for other "live" mailboxes as well.