DEPARTMENT OF APPLIED MATHEMATICS
AND COMPUTER SCIENCE

02450 - Introduction to Machine Learning

Project 2

Authors:
Georgios Tsimplis (s200166)
Jorge Sintes Fernandez (s202581)
Klara Wesselkamp (s202382)

Course responsible:
Jes Frellsen
Morten Mørup

November 17, 2020

Exercise	Contribution		
	Jorge Sintes	Georgios Tsimplis	Klara Wesselkamp
1.1			100%
1.2			100%
1.3			100%
2.1		100%	
2.2		67%	33%
2.3		100%	
3.1	100%		
3.2	100%		
3.3	100%		
3.4	67%	33%	
3.5	33%	33%	33%
4.1	50%		50%
4.2	50 %		50 %

Contents

1 Introduction

We remind the reader of the dataset, which is called ”spambase”. It consists of the analyses of word frequencies in e-mails which were either classified as Spam or No-Spam. A central characteristic, as remarked in the last report, is, that we have a very low matrix, meaning a lot of the words don't appear in every e-mail. This is to be expected, however will pose some problems in the calculation.

2 Regression, Part A

2.1 Chose and explain variable

The variable we chose is the frequency of the character ”!”. The idea here is that this character holds some higher significance in the forming of sentences and thus the choice of words to do the linear regression on is interesting and significant. We do the regression here based on the first 10 most correlated parameters. In the following graphic, it can be seen how a different number of variables affect the accuracy of the model. One important finding is, that the testing error (using a 10-fold cross-validation and taking the mean) is not going down significantly after more then 10 parameters. The training error keeps going down, explainable so, but this could here be considered overfitting.

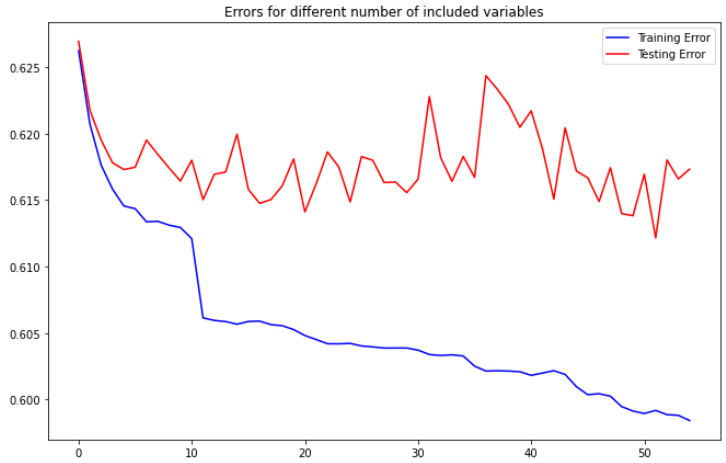
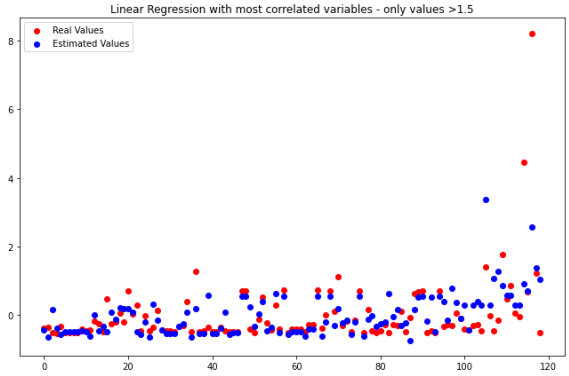
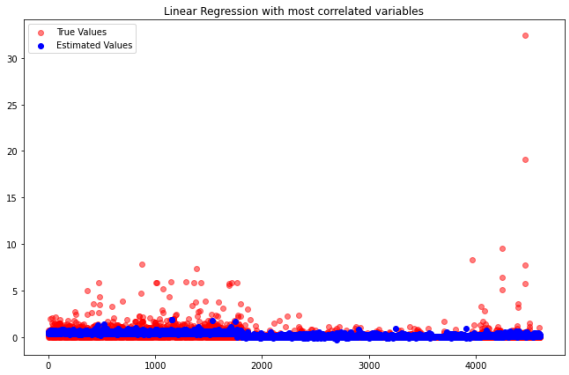


Figure 1: Training/Testing error for the different number of variables

Commentary: While doing the regression it became clear that the abundancy of 0 in the matrix is a big problem, as the relevant values estimated will be much lower than they should. We therefore, chose to clean the matrix of minor values and compare the two in order to examine how the method could work on a less sparse dataset. We are conscious that this is not applicable to new data. A suggestion would be to enchain a classification ”Minor/Major-Value” with a regression only on the major values, setting all other values to 0.



(a) Regression using highly correlated variables greater than 1.5



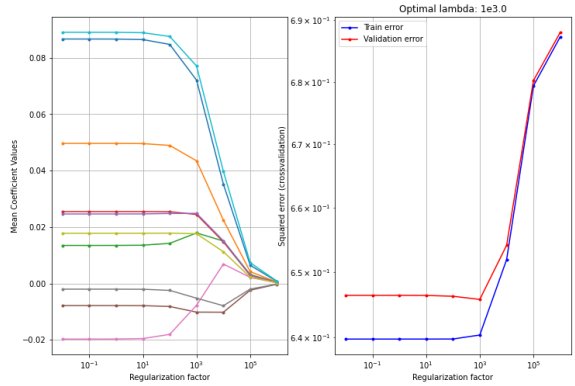
(b) Regression using highly correlated variables

Figure 2

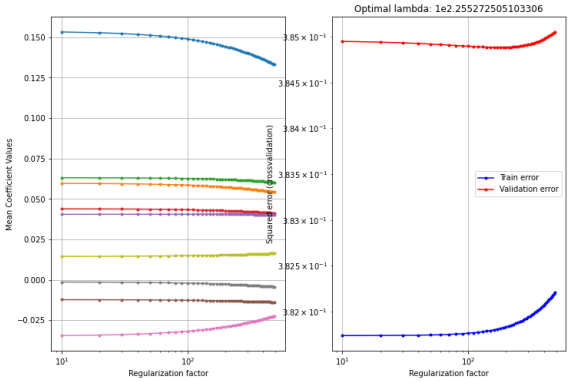
2.2 Effect of Regularization Parameter

A 10-fold cross-validation was used.

For the first image, we see that the regularization parameter did not do a good job, the minimum is barely visible.



(a) Coefficient Weights & Errors across λ



(b) Closer range

A zoom of optimal values in the ranges from 100-4000 gives a closer hint as to where the minimum might lie, as well as a more precise value, that is around 252.

If we were to look at the reduced vector we can see an actual minimum and optimal value of λ, showing that this method has some merit.

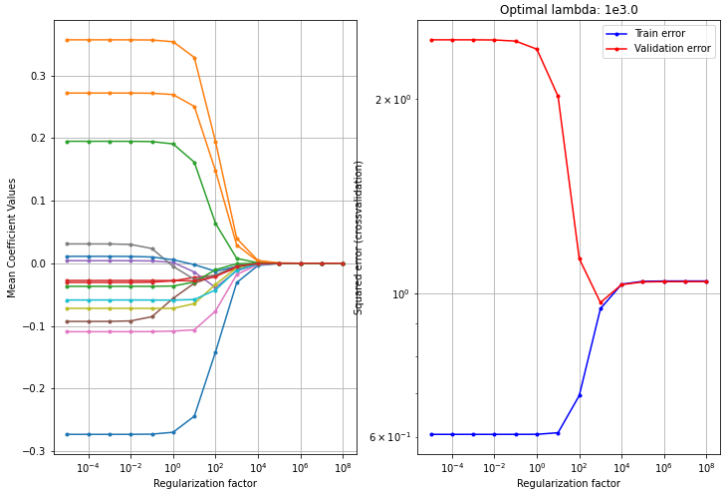


Figure 4: Coefficient Weights & Errors across λ

If we zoom in more towards the optimal λ - around 256, and we chose values ranging from 10 to 500 in steps of ten, we get the following generalization errors for the 10 different folds.

Fold Number.	1	2	3	4	5	6	7	8	9	10
λ^*	420	270	310	310	290	240	400	200	380	180
E^{gen}	0.67	0.65	0.67	0.63	0.65	0.65	0.63	0.64	0.57	0.38

2.3 Predictions with the model

The model was trained with a λ-value of 252. The weights of the different attributes are:

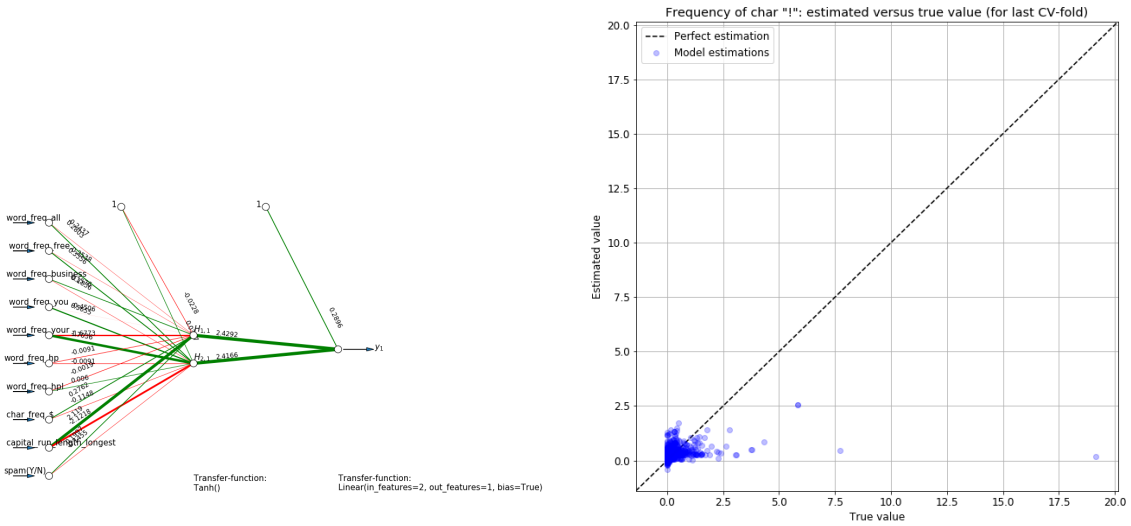
Offset	0.27
spam(Y/N)	0.13
word_freq_you	0.08
char_freq_$	0.05
word_freq_all	0.04
word_freq_free	0.04
word_freq_hp	-0.01
word_freq_your	-0.03
word_freq_hpl	-0.01
capital_run_length_longest	0.02

It is not easy to say if this makes sense in order to make predictions. The choice of an exclamation mark vis-a-vis other punctuation is a personal one. However, when looking at the most weighted words you can see a certain sensationalist attitude their, indicating an (over-)emphasized sentense. We'll now proceed to different analyses to see if our results hold up.

3 Regression, Part B:

In this section, we will implement two additional models -Artificial Neural Network and the so called ”baseline” (the mean of the values) - and finally we will compare the three models applying 2-layer cross-validation. The aforementioned process is done in order to compare the performance of the different models.

Before applying 2-layer cross-validation, we built a simple ANN for the regression problem using 5-fold split in order to have an overview on how the model operates on our data-set. Our input layer consists of 10 neurons -which represent the same attributes that we used in Part A- and our output layer consists of 1 neuron which represents the attribute that we attempt to predict. In the hidden layer we chose 2 neurons equipped with the activation function of *hyperbolic tangent*. Below is presented the architecture of the model along with a plot which illustrates the differences between the predicted and the true value.



(a) ANN (b) True value vs Predicted

Figure 5

In Figure 1.b we can see that the most of the observations are located at the on either side of $y = x$. However, there are some totally wrong predictions. This fact indicates that we should increase the number of the hidden layers or the number of nodes in the hidden layer to achieve a better performance. By this simple implementation as well as from the previous part we realized that the fact that the mode of each attribute is 0 results in a situation that every non-zero value - which represents the frequency of a word/character- is considered an outlier but should not be eliminated from the data-set because it contains valuable information. However, at this point we understand that working with a data-set like this we have to implement complex models in order to have more accurate predictions.

3.1 Cross-Validation

In this section we will implement a 2-level cross-validation in order to measure the performance of each model. We will use $K_1 = K_2 = 10$ outer and inner folds respectively. The algorithm that we used to achieve the cross-validation is the following [1].

Algorithm 1 Two-level cross-validation for model selection

Require: K_1, K_2, folds in outer and inner cross-validation
Require: $\mathcal{M}_1, ...\mathcal{M}_\mathcal{S}$, The S different models to cross-validate
Ensure: $\hat{E}^{gen}$ is the estimate of generalization error
for $i = 1, ..., K_1$ **do**
  Outer cross-validation loop. First make the outer split into K1 folds
  Let $\mathcal{D}_i^{par}$, $\mathcal{D}_i^{test}$ be the i^{th} split of $\mathcal{D}$
  for $j = 1, ..., , K_2$ **do**
    Inner cross-validation loop. Use cross-validation to select optimal model
    Let $\mathcal{D}_j^{train}$, $\mathcal{D}_j^{val}$ be the j^{th} split of $\mathcal{D}_i^{par}$
    for $s = 1, ...S$ **do**
      Train $\mathcal{M}_S$ on $\mathcal{D}_j^{train}$
      Let $E_{\mathcal{M}s,j}^{val}$ be the validation error of the model $\mathcal{M}_s$ when it is tested on $\mathcal{D}_j^{val}$
    end for
  end for
  For each s compute: $\hat{E}_S^{gen} = \sum_{j=1}^{K_2} \frac{|\mathcal{D}_j^{val}|}{|\mathcal{D}_i^{par}|} E_{\mathcal{M}s,j}^{val}$
  Select the optimal model $\mathcal{M}^* = \mathcal{M}_{f^*}$ where $s^* = arg\min_s \hat{E}_S^{gen}$
  Train $\mathcal{M}^*$ on $\mathcal{D}_i^{par}$
  Let E_i^{test} be the test error of the model $\mathcal{M}^*$ when it is tested on $\mathcal{D}_i^{test}$
end for

As different models for the ANN case we considered the different values of h nodes in the hidden layer. The ideal number of hidden nodes and hidden layers is a subject of study in many universities around the world. We used the strategy which is described by the following steps.

1. We started to run the algorithm using as numbers of the hidden nodes from the range [1-5].

2. While the optimal model that we had as result were the maximum of the used range we were increasing the range.

3. We stopped the above process when we obtained a range that the optimal values of the hidden nodes varied more in respect to outer fold.

After the above process we realized that a good range for the number of the hidden nodes is: [10-19]. We implemented the same process to find a reasonable range for the regularization parameter λ. The formula that we used to compute the test error was:

$$E^{test} = \frac{1}{N^{test}} \sum_{i=1}^{N^{test}} (y_i - \hat{y}_i)^2$$

3.2 Comparison Table

After implementing the 2-level cross-validation for all the models following the process which was described in the previous section, we created the following table which includes for each model, the best performed among the sub-models in order to compare them.

Outer Fold	ANN		Linear Regression		Baseline
i	h_i^*	E_i^{test}	λ_i^*	E_i^{test}	E_i^{test}
1	14	0.230971	390	0.26187	0.298395
2	18	0.357946	290	0.369143	0.432962
3	19	0.238038	380	0.320162	0.374838
4	14	0.465597	330	0.572299	0.638142
5	11	0.979307	260	0.966766	0.999129
6	14	0.1921	360	0.153335	0.194833
7	16	0.240335	490	0.258734	0.298691
8	18	0.291925	310	0.293547	0.346115
9	19	0.26627	400	0.27422	0.28958
10	17	0.252194	140	0.224569	0.275823

We can observe that at 7 out of 10 folds the ANN has a better performance from the Generalized Linear Regression. The Baseline has the worst performance across all over the different outer folds. The values of λ_i^* fluctuates in the same range as in the 1st Part and the values of the errors vary in the interval [0.19-0.99]. This fact depends is due to the different folds (part of the data-set).

3.3 Evaluation of performance differences

In this section we will compare all the different pairs of models using the (t-test) setup-I[1]. Our null hypothesis is the following:

H_0 : The compared models have the same performance.

Model A / Model B	Confidence Interval	p-value
ANN / Reg. L.R	[-0.0895 , 0.0520]	0.301
ANN / Baseline	[-0.1584 , -0.0416]	1.24e-09
Reg. L.R / Baseline	[-0.0724 , -0.0367]	1.92e-07

Interpreation From the above table we can see that in the first case the confidence interval contains 0, so we cannot say whether there's an improvement or a degradation from ANN to the linear regression. Additionally, the p-Value is significantly higher than 0.05%. The interpretation is, that we have a high probability of the performance difference being around 0, and a considerable probability that we would get the differences in performance if the Null-Hypothesis was true. So we would not say that ANN and Linear Regression yield significantly different results, although it seems to be the case at first.

For the last two parts the p-Value is close to 0. The interpretation is, that under the null-Hypothesis, we would rarely see the performance differences that we have here. We can thus, with great certainty say, that the baseline-model has the worst performance.

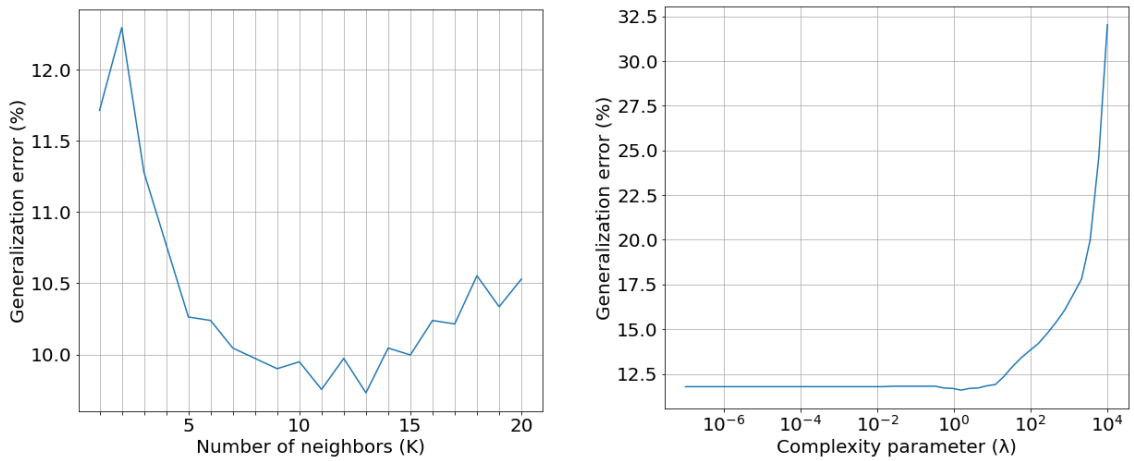
4 Classification

4.1 Explanation of Classification Problem

In this we're dealing with a binary classification problem, where the goal is to classify E-mail according to the Spam (1) and No-Spam (0) Attributes. Before we started we tried to figure out the optimal number of variables to use and which variables they are concretely. Analogous to exercise 1a) we tried with an increasing number of most correlated attributes, as well as a random selection of attributes. The most correlated attributes, unsurprisingly did remarkably better. The results indicate that the 10 most correlated attributes would again be a reasonable number to chose. The data was standardized before, in order to make the variables comparable.

4.2 Comparison of different methods

We chose to compare the K-nearest-neighbour model to logistic regression as well as the so-called baseline model, corresponding to the mode of the Spam-Attribute. The mode in this case is a 0, not spam. There's two parameter choices to take here: The Error obtained in the KNN model depends a lot on the chosen number of neighbours, so it's essential to find the optimal number here as well. The other parameter is the regularization term λ. Through numerous runs, we have observed that the number of λ has barely any effect on the generalization error. To illustrate the behaviours of both parameters we include two plots. For KNN, it can be observed that somewhere around 10 neighbours lays the optimal value of K. For logistic regression we can observe the little incidence λ has over the error, given λ low enough.



(a) Test Error vs. No. of neighbours

(b) Test Error vs. λ

Figure 6

4.3 Cross-Validation Table

Outer Fold	Logistic Regression		K-Nearest Neighbours		Baseline
i	λ_i^*	E_i^{test}	K_i^*	E_i^{test}	E_i^{test}
1	1.52642	0.125813	8	0.117137	0.438178
2	1.52642	0.121739	8	0.104348	0.38913
3	0.19307	0.132609	8	0.1178261	0.38913
4	4.29193	0.121739	9	0.102174	0.373913
5	1.52642	0.123913	11	0.891304	0.415217
6	1.52642	0.108696	9	0.978261	0.352174
7	4.29193	0.956522	8	0.891304	0.380435
8	0.910298	0.102174	8	0.869565	0.426087
9	1e-07	0.123913	8	0.117391	0.38913
10	0.19307	0.130435	8	0.956522	0.386957

Comment on the findings: The baseline model does a bad job, as was expected. The error moves in the 40% range, corresponding to the percentage of Spam in the dataset.
The logistic regression yielded a considerably better results, with its error being around 10-12%. The optimal λ has some outliers where it's significantly higher or almost 0, but on the whole, the value 1.5 appears most often so we should probably chose that value as the optimal logistic regression.

4.4 Statistical Evaluation of models

In this section we will compare all the different model pairs using (t-test) setup I [1]. Just as for the regression part, our null hypothesis is:

H_0 : The compared models have the same performance.

Model A / Model B	Confidence Interval	p-value
KNN / Log. Reg.	[-0.0273, -0.0105]	0.00033
KNN / Baseline	[-0.2751 , -0.3135]	3.37e-11
Log. Reg / Baseline	[-0.2561 , -0.2946]	6.24e-11

Interpretation: In all three cases we observe a minuscule p-Value, so we can discard the null hypothesis with certainty and say that all the models have a different performance.
The confidence intervals indicate the range in which the difference in performance will lie, which is a difference of around 25% difference to the baseline, and a slight improvement of 1-2% from Logistic Regression to K-nearest-neighbours.

4.5 Logistic regression with uncertainty parameter

The model was trained with the λ value of 1.52642, the most common value of the cross-validation results. The resulting coefficients are shown in the table below. As we expected, the coefficients are very different from the ones in the regression task, as we were not trying to predict the same attribute. However, the results we get are really enlightening, as we can observe that the character $ has the most positive weight (common character in spam e-mails). Also, the word hp has the most negative weight (the name of the company from which the data-set was extracted).

word_freq_your	0.37
word_freq_000	1.29
word_freq_remove	1.50
char_freq_$	1.70
word_freq_you	0.26
word_freq_free	0.68
word_freq_business	0.59
word_freq_hp	-3.39
capital_run_length_total	0.72
word_freq_our	0.32

5 Discussion:

5.1 Findings

The firt part of the report was the regression. The problem here, was a little artificial, as our problem is inherently a classification problem. However, it was interesting to see, that the regression worked to some extend. It didn't give the right values, by far, due to the sparse matrix, but it did give an inkling, a direction of where the data will lie. Additionally, it became clear, that the linear regression and ANN have a similar performance regarding the 1^st Part, underlined by the results of the statistical test. The baseline's performance was consistently worse across the different folds of the validation. The best generalization errors that we managed to achieve were in the 20-25%-range.

In the second part, the classification, we found the expected property, that the baseline-model was consistently the worst. In most instances, although not in all of them, the KNN-Method with 8 neighbours is better that the logistic regression with widely varying ranges of the regularization parameter λ. The best generalization error that we managed to achieved was in the 10-11%-range.

5.2 Compare to previous findings

The dataset, has been used as a playground for machine learning for quite some time, but, as mentioned has mostly been used in classification problem.
According to the maker's information, their best results was a 7% misclassification error. An insistence on zero false positives in the training/testing set (something that was not done in this reports, but could be an interesting starting point for future examination), 20-25% of the spam passed through the filter [2].
A documented classification with a decision-tree ('entropy'-criterion) yielded another 9% misclassificationerror [3]
A K-nearest-neighbour-method [4] done in R came to approximately the same results (8.8% accuracy for the optimal neighbour of 1).
On the whole the performance of our algorithms is close to what could be called an optimal performance. A 100% filter rate would obvioulsy be desirable, but is also utopian, since the definition of spam is not a clear one, and since the filtering out of non-spam messages should be avoided at all cost.

References

[1] Mikkel N. Schmidt Tue Herlau and Morten Møru. "Introduction to Machine Learning and Data Mining". In: Technical University of Denmark, August 2020.

[2] Mark Hopkins. *Descriptions of Dataset by it's makers*. 1 july 1999 (accessed November 17, 2020). URL: `https://archive.ics.uci.edu/ml/datasets/spambase`.

[3] Sam Pepose. *spam-classifyer*. 5 may 2016 (accessed November 17, 2020). URL: `https://github.com/sampepose/SpamClassifier`.

[4] Dhruvin Shah. *Email Spambase Data Set Analysis Using KNN*. 25 may 2020 (accessed November 17, 2020). URL: `https://rpubs.com/drshah96/625987`.