DANMARKS TEKNISKE UNIVERSITET



02450 – Introduction to Machine Learning and Data Mining

PROJECT 2

Martin Mikšík s212075





Lukáš Málek s212074

A picture containing icon

Description automatically generated[[1]](#footnote-2)

Table of Contents

[1 Introduction 1](#_Toc101527733)

[2 Regression part I 1](#_Toc101527734)

[2.1 Applying only non-cathegorical attributes 1](#_Toc101527735)

[2.2 Applying all of the available attributes 2](#_Toc101527736)

[2.3 Regularization and K-Fold Cross-Validation 2](#_Toc101527737)

[2.4 Tuning 3](#_Toc101527738)

[2.5 Testing on new data 3](#_Toc101527739)

[3 Regression part II 4](#_Toc101527740)

[3.1 The Artifitial Neural Network 4](#_Toc101527741)

[3.2 The comparison 5](#_Toc101527742)

[3.3 Statistical testing 5](#_Toc101527743)

[4 Classification 5](#_Toc101527744)

[4.1 Data Preparation 5](#_Toc101527745)

[4.2 Regularization for Logistic Regression 6](#_Toc101527746)

[4.3 Classification Models Comparison 8](#_Toc101527747)

[4.3.1 KNN 8](#_Toc101527748)

[4.3.2 Decision Tree 8](#_Toc101527749)

[4.3.3 SVM (Support Vector Machine) 9](#_Toc101527750)

[4.3.4 Logistic Regression 9](#_Toc101527751)

[4.3.5 Baseline: 9](#_Toc101527752)

[4.4 Statistical Testing 9](#_Toc101527753)

[5 Discussion 10](#_Toc101527754)

[6 References 10](#_Toc101527755)

[7 Appendix 1 11](#_Toc101527756)

# Introduction

This is a continuation of previous work on Data set Visualisation Project [1], further referred as the „Project1“, for Denmark Technical University course 02450 Machine Learning & Data Mining [2] which we will further refer to as „the Course“.

In this document, we will further visualize, apply and deepen the understanding of various Machine Learning and Artifitial Intelligence algorithms like Regularized Regression, Neural Networks and Classification.

Each of the following headings have a footnote with a percentage contribution of each team member.

# Regression part I [[2]](#footnote-3)

## Applying only non-cathegorical attributes

First, we will try Linear Regression on all of the non-cathegorical data, as discussed in Project1, as it would be fairly straight forward and easily implemented approach.

We are working with a data set with 344 instances and 9 attributes:

A screen shot of a computer

Description automatically generated with low confidence

First, we compose a data structure of the dataset as in Project1. Our goal is to accurately predict the Bill Length attribute by Bill Depth, Flipper Length and Body Mass. We have therefore removed the other unnecessary attributes.

Then, we have standardized the data, as there was excessive value differences in attributes:

We have then applied a linear regression function to the data set with poor results. The visualization shows the difference between estimated and true values, as well as a histogram, showing the variance of residuals:

Chart, scatter chart

Description automatically generated

At this point, a visual analysis is enough to conlude that the model would not perform well due to a high variance of residuals. We will therefore need to apply a new strategy.

## Applying all of the available attributes

As concluded above, we need to improve the strategy by applying more attributes. As the other attributes are cathegorical, we need to be more careful and preproccess the data first. For example, one of the attributes is Species. A standard approach would be creating a dictionary to perceive the attribute’s name and give it an, computer understandable, integer value. This might, however, cause a problem in several Machine Learning algorithms that could mistakenly see one Species as more important than the other, which is not the case. Therefore, we will apply the Leave-One-Out Encoding [3] (also known as One-Hot) to all of the categorical data. We will also remove attributes that don’t carry any meaningful value, like Year, which only states when an element was added.

We can therefore maximize the use of most attributes while minimizing future algorithmic problems. The last change to our raw dataset is to remove the attribute we want to predict and store it in a vector , which we will later use for training purposes. We have now created input matrix .

The transformation of raw data into input matrix X:

Text

Description automatically generatedA screenshot of a computer

Description automatically generated with medium confidence

N = 333 and M = 11, where y = Bill Length.

Only now we will standardize the input data, as discussed in previous attempt. We will however standardize only the non-cathegorical data, as it would not help to standardize One-Hot encoded attributes.

Now, our first row of matrix looks like this:



*Bill depth, flipper len, mass, Adelie, Chinstrap, Gentoo, Biscoe, Dream, Torgersen, Female, Male*

To quickly confirm the improvements made in this step, we will again plot the residuals:

Chart, scatter chart

Description automatically generated

We can see an improvement and we can finally continue to regularization and cross-validation.

## Regularization and K-Fold Cross-Validation

We will introduce an L2 regularization parameter that will penalize squared magnitude of Model coeffitients , so they properly contribute to the predictions. We will use several of these parameters and try their performance on 10-Fold Cross-Validation.

Let´s therefore introdeuce the regularization term that will penalize large weights, where are the Model weights, is the true value and is the prediction, are standardized and respectively.

Re-writing this into algebraic form, we can clearly see that the model with would be the same as Linear Regression LSQ problem as in the first chapter.

By solving the optimization problem, we can find the optimal coeffitients (weights), Where is the optimal weight and is an identity matrix*.*

Now we will conduct a 10-Fold Cross-Validation that will split the data into training and testing sets in 9:1 ratio and compute the error by comparing the predicted values with true values of and repeat for different permutation of a split.

Each split will give us the best that maximizes the performance of the split.

As we can’t use only the model that works the best for one split only, we will average the best weight values of all of the validation splits and use this as the best regularization parameter that will improve the prediction model of Bill Length.

## Tuning

We have used the range of upto linearely distributed by tens of values.

We have also tried different degrees of estimation polynomial to be 1 to 5 and concluded, that the second degree is the best for regression fitting with Generalization error of and regularization parameter of

Where the Generalization error is estimated by

We can see the visualization for one of the Folds of the effect of regularization factor on the weights of the predict function (left) or the optimization problem when looking for the best (right)

Chart

Description automatically generatedChart

Description automatically generated

Concluded, even though we have paid the price for more complicated pre-processing and several brute-force tuning methods of testing the several degrees of the model, we have done everything we could to minimize the Generalization error in the Machine Regression problem with nested cross-validation.

## Testing on new data

Now we know our model should be good at predicting of new unseen data. To rigorously test and validate that our algorithm and standardization work in real-life scenarios, we will remove a few instances from the raw data set and re-train the model. Removing only few instances did not have a significant impact on the performance of the predictions. We will then use the removed exemplars as a never-seen before input to the predictor by the *predict()* function. As the predictions happen on standardized and transformed into one-hot data, we will, of course, write a function[[3]](#footnote-5) to mirror exactly this with our new data.

Testing on 7 new instances of penguins:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Penguin: | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Predictions | 40.07 | 38.01 | 48.33 | 49.41 | 47.81 | 47.61 | 37.83 |
| True y | 40.6 | 35.7 | 48.7 | 51.3 | 46.4 | 43.5 | 34.4 |
| ||True err|| | 0.53 | 2.31 | 0.37 | 1.89 | 1.41 | 4.11 | 3.43 |
| Error %: | 1.31 | 6.47 | 0.76 | 3.68 | 3.04 | 9.45 | 9.97 |

We can conclude that the model behaves effectively, even with the never-seen-before instances of penguins, with a mean error of 4.95%, which further confirms the previous assumptions.

# Regression part II [[4]](#footnote-6)

This chapter will compare the Regression model from the previous section, Baseline and ANN[[5]](#footnote-7).

In the table below, we will show the best regularization parameter and the best number of hidden layers of the artifitial neural network, for each outer fold, together with the error measurement , which is the mean squared loss per observation.

We first created and cross-validated the Baseline model, the simplest linear regression 1-degree fitting polynomial model without regularization. Comparing this to a two-layer cross-validated model we created from the fundations of the model from previous section, closely following Algorithm 6 in the Book [4], where we further optimized parameters and used an increased number of regularization parameters on a smaller interval (namely 150 value on interval 0-300), we can see the importance of regularization in the reduction of the error.

## The Artifitial Neural Network

To finish the table, we will need to create a third model, the Artifitial Neural Network, with a variable number of hidden units, ranging from 1 to 10, with nested 10-Fold cross-validation, validating the best-chosen number of hidden units, which, itself, is chosen by cross-validation. For the training, we have a maximum iteration limit selected to 10 000 and without replicates. As for the Network Activation function, we chose *tanh().*

We can visualize the last inner fold for and :

Chart, radar chart

Description automatically generatedChart, diagram, radar chart

Description automatically generated

## The comparison

Comparing the created table below and the ease of use, the Baseline model is undoubtedly the quickest method to implement, establish and compute. It could be used for „Initial engineering guess“ when starting to work on a data-mining project, but the results are not very accurate. The regularized regression model gave excellent & stable results with reasonable computation time. If this method is still not accurate enough for the nature of the task, only then we would recommend building an Artifitial Neural Network model, as it is computationally heavy and needs hours to complete without CUDA[[6]](#footnote-8) enabled. Furthermore, the ANN needs far more tuning parameters to be established and, although the Nested Cross-Validation for hidden units selection slows down the proccess tremendously, it is only one of many parameters to be optimized. Therefore, further analysis for stable results would be needed.



*Table 1: Two-level cross-validation table used to compare the three models*

## Statistical testing

We have conducted statistical tests between the models on unseen data[[7]](#footnote-9) with Student’s T test [5]:

|  |  |  |
| --- | --- | --- |
| Model comparison | Confidence Interval | P-value |
| Baseline – Linear Regression | (-5.6, 2.43) | 0.372 |
| Linear Regression – ANN | (-3.38, 8.88) | 0.314 |
| ANN – Baseline | (-4.84, 13.51) | 0.291 |

As we are using unseen data for the statistical testing for higher accuracy, the trade-off is less confident test, as we are using only small set of new testing values and we cannot fully satisfy the Central Limit Theorem and assumption of the Normal distirbution of the error. The results, however, can tell us that, even though the confidence intervals tend to overlap, most notably for Baseline and Linear Regression, the probability, p-value, reflects the evidence against the null hypothesis of the models being same, with a strong statistical significance.

# Classification [[8]](#footnote-10)

In classification we are trying to predict the sex of penguin. To make the classification as accurate as possible, we used 5 models: KKN, Decision Tree, SVM, Logistic Regression and Baseline. F

The classification problem of sex prediction is binary, as only female and male sex was observed for this dataset.

## Data Preparation

In the first step we again standardized the data as in the regression part. Then the data were tested using two layer cross validation without encoding the data to see the power of the one-hot encoder. Without the encoding the Total Generalization Error Estimate was 12%. As expected, after applying one-hot encoder for species and island, the total error rate was improved to 11.1%.

Although the feature selection was not required for the second part of the project, all these tests were executed without using the year attribute, as by applying common sense it should not play a big role in machine learning models. The Pearson matrix is included in previous part of the report, but from the Correlation matrix below we can see that some of the attributes are highly correlated, especially the flipper length vs. species and the flipper\_length vs. body mass.

Chart

Description automatically generated

*Figure: Correlation between the attributes*

However, this was a mistake for classification, as the final result improved to total Total Generalization Error Estimate of 9.61% after considering the year as well. This might be caused by the fact that each year the penguins well-being might be different due to amount of food, temperature, or animal migration. Also the regularization will be performed and the risks of overfitting thus are mitigated, but on the otherhand, the regularization is implemented only for logistic regression, but the improvement was seeable in other models as well.

For example. lack of the food in one of the year can result in more significant weight difference between males and females, especially for males, when they are taking care of the childs. Also, the global warming might play a big role near the poles where the observations for this dataset were collected.

To sum up. The non-categorical data were standardized, categorical data were encoded using one-hot encoder and and all attributes were considered for the classification.

## Regularization for Logistic Regression

Regularization improves the performance for new unseen data, but instead of squared loss we use log loss, where *(x,y)* is the dataset D containing many labeled examples, *y* is the labeled example which is 0 or 1 since it is logistic regression and *y’* is predicted value between 0 and 1.

Regularization could be understood as penalization for complexity. It adds bias if our model suffers from high variance, which simply means that we overfit our data. Using L2 regression, we can lower the importance for the parameters that worsen our results. Our goal is to minimalize the Log Loss using aa new hyperparameter

But as we can see, the regularization parameters are very often zero, which means that the regularization is not considered for the fold. For the graph with L2 norm (right) our goal is to have as low number as possible, since with higher number the algorithm is more affected by the regularization.

To illustrate how the values are calculated the graph for fold 6 is shown below, where we take the lambda with lowest error. The results could be seen

Chart

Description automatically generated A picture containing shape

Description automatically generated

Figure 2: Choosing the correct regularization parameter

The values do fluctuate quite a lot which is more interesting and different than the low values themselves, but the folds can be quite different in nature (i.e. varying difficulty) and the lambdas can vary a lot. Having a lambda of 0 means that it probably doesn't overfit on this specific fold and no regularisation is needed to avoid the overfitting.

Chart, line chart

Description automatically generated

Figure 3: Regularization parameter

For the graphs that resulted in we can see that the error rate is the same until around . Therefore it does not matter which on interval (0,100) we decide to choose. The error will be the same. To experiment with it, we tried to run the algorithm for K1=K2=50. In the end, the parameters were very close to zero in most of the folds.

## Classification Models Comparison

In this section we compare models from the inner loop of the two layer cross validation.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| INNER LOOP | KNN | DECISION TREE | SVM | LOGISTIC REGRESSION | BASELINE |
| i | Ems,jval | Ems,jval | Ems,jval | Ems,jval | Ems,jval |
| 1 | 10.37% | 16.72% | **10.03%** | 10.03% | 54.63% |
| 2 | 14.38% | 18.95% | **12.64%** | 13.38% | 55.37% |
| 3 | 16.72% | 18.90% | **11.52%** | 12.26% | 55.00% |
| 4 | 14.67% | 17.78% | **11.48%** | 11.85% | 54.81% |
| 5 | 15.00% | 14.81% | **10.37%** | 10.74% | 55.19% |
| 6 | 12.33% | 16.30% | 12.96% | **11.85%** | 55.19% |
| 7 | **10.67%** | 18.89% | 11.11% | 11.48% | 55.19% |
| 8 | 12.00% | 14.44% | 11.48% | **9.63%** | 55.19% |
| 9 | 16.33% | 14.81% | 11.85% | **11.48%** | 55.19% |
| 10 | 14.00% | 19.63% | 11.48% | **11.11%** | 54.81% |
| avg | 13.65% | 17.12% | 11.49% | 11.38% | 55.06% |

*Table 2: Comparison of different Classification models in Inner Loop*

### KNN

Icon

Description automatically generatedFor KNN we choose the best k neighbors in interval 1-10 for each iteration, resulting in very computational expensive model, which does not deliver very great results, as the SVM and logistic regression algorithms outperform this model. Could be fasten up by choosing a fixed value for k, but then it would deliver even worst results

Chart, line chart

Description automatically generated

Figure 3: Finding the best K for KNN model (graph for one of the many folds tested)

### Decision Tree

Not great, not terrible, from the 10 folds in the inner loop of the cross validation the error is stable around 17%, which is about 5% more than for KNN, SVM or Logistic Regression. The decision tree algorithms are otherwise highly efficient and very often one of the best fore wide variety of datasets. Nevertheless, this dataset is not the case. It can be caused by the fact that the decision tree was not controlled by any changing parameter, which could improve the results. The default setting for DecisionTreeClassifier from Python library were used in this case. Nevertheles,, we could have controled the complexity of a classification tree by prescribing for example the maximum depth of the tree, the minimum number of samples required to split an internal node, the maximum number of leaf nodes, the minimum value of the decrease of the impurity to split a node and others.

### SVM (Support Vector Machine)

Icon

Description automatically generatedThe best performing model for our dataset, chosen five times as the best one from total 10 folds. Generally speaking, SVM are effective for small training datasets, which this one definitely is, with 333 instances in total (which are than separated to even smaller testing dataset in two layer cross validation. Also, in our case the points can be separated by a hyperplane, which is not true for every dataset.

### Logistic Regression

A picture containing text, metalware, gear

Description automatically generatedThe model was described in the previous section *Regularization for Logistic Regression*. We fit the “S” function giving us the probability of the binary output. This model performs very well for this dataset, sitting in the 2nd place, right behind the SVM, as it was chosen as the best model 4 out of 10 folds. In the regression we predict the bill depth, but in classification problem we solve the binary problem of sex. Therefore we are not nescesarily expecting them to be comparable.

### Baseline:

There are 168 males and 165 females in the dataset. Therefore for the baseline we are considering that every penguin is a male

The interesting fact for the table above is that we can see that baseline is returning error rate higher than 50%. This is caused because of the 2-layer cross validation, as the test data in the outer layer are containing mostly male instances.

The results could be seen in Table 3.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| OUTER FOLD | KNN | | DECISION TREE | SVM | LOGISTIC REGRESSION | | BASELINE |
| i | ki | Eitest | Eitest | Eitest | λ | Eitest | Eitest |
| 1 | 7 | 14.71% | 35.29% | **20.59%** | 1862 | 23.53% | 52.94% |
| 2 | 4 | 8.82% | 8.82% | **0.00%** | 0 | 0.00% | 47.06% |
| 3 | 4 | 2.94% | 23.53% | **5.88%** | 38 | 5.88% | 50.00% |
| 4 | 6 | 6.06% | 15.15% | **6.06%** | 3548 | 6.06% | 51.52% |
| 5 | 1 | 12.12% | 18.18% | **12.12%** | 0 | 12.12% | 48.48% |
| 6 | 5 | 6.06% | 9.09% | 12.12% | 38 | **12.12%** | 48.48% |
| 7 | 7 | **3.03%** | 6.06% | 6.06% | 31 | 6.06% | 48.48% |
| 8 | 7 | 3.03% | 12.12% | 12.12% | 0.93 | **12.12%** | 48.48% |
| 9 | 1 | 12.12% | 24.24% | 9.09% | 0.4 | **9.09%** | 48.48% |
| 10 | 8 | 6.06% | 36.36% | 6.06% | 3.47 | **9.09%** | 51.52% |
| avg | 5 | 7.50% | 18.88% | 9.01% | 552.2 | 9.61% | 49.54% |

Total Generalization Error: 9.09%

The bold numbers bold are the chosen models from the inner loop. As we can see, after using the model on unseen data, in most cases the correct model was chosen. The KNN delivered the best results, but since we are also tuning the parameter k, it is really computationally expensive. The same could be said about the logistic regression, where we tune the parameter lambda. Based on this result, personally I would choose the SVM, as it seems to deliver great results in both inner and outer loop, and also its computationally very efficient. On the other hand, SVM works well only for small datasets.

## Statistical Testing

For statistical testing we used McNemera’s test function from toolbox. (setup 1)

|  |  |  |
| --- | --- | --- |
|  | Confidence Interval | P-values |
| KNN-BASELINE | (0.28 ; 0.61) | 0.0001 |
| TREE - BASELINE | (0.096 ; 0.38) | 0.0078 |
| SVM - BASELINE | (0.258 ; 0.63) | 0.0003 |
| LOG. REG. - BASELINE | (0.211 ; 0.616) | 0.0013 |
| KNN - LOG.REG. | (-0.07 ; 0.13) | 1.0000 |
| TREE - LOG. REG. | (-0.353 ; -0.004) | 0.1090 |
| SVM - LOG.REG. | (-0.027 ; 0.087) | 1.0000 |
| KNN - SVM | (-0.0827 ; 0.0827) | 1.5000 |
| TREE - SVM | (-0.37 ; -0.05) | 0.0390 |
| KNN - TREE | (0.049 ; 0.369) | 0.0390 |

With higher p-values the confidence that the distributions of the two models are similar increases. Therefore we can see that from all the models that we tested the similarity between the KNN-SVM, SVM-LOG and KNN-LOG is the highest.

After checking the previous results with the table, the results are not surprising. These two models delivered the best result. Moreover, in some cases the error rate was the same. After closely reviewing the results inside out script, we can see that these models were doing the wrong predictions for the same instances in most of the cases. Therefore it is more than clear why the p values are so high while comparing nearly the same models.

Confidence intervals are not considered in details for this case, since for ration values used in regression one it says on which interval we can be confident about our results. But since we are only looking for binary values in classification problem, it does say us very much of information. Also, the fold are containing not many values as the dataset is small, so the interval is dependant on the number of male and females for each fold, which does vary a lot.

If we look at the p-values of the baseline, we can say that the results are very different, since the p-values are lower than 0.05. The baseline model is much worst than all other models that we tested. As the final conlsuion we can therefore easily say that all of our models were much better than the baseline.

# Discussion [[9]](#footnote-11)

Discussion for part I and part II are in the end of each section. In classification, we used 5 models . The baseline proved not to be a very efficient for our dataset. Still, it can work very well as a threshold which we wanna reach with our other models. SVM and logistic regression gave us the best results with lowest error rates. Althoutgh the logistic regression was slightly better overall, it was also much more computationally demanding. Since we tuned the regression parameter for each iteration. On the other hand, the SVM was a great surprise. Its the fastest one from the 4 algorithms, still, it gave uss satisfying results.

We tried to review our results with reports avaible on the internet. Unfortuantelly, it seems that noonse has previously made that detailed report about this dataset. Therefore we cannot compare our results with published studies.

# Exam Questions[[10]](#footnote-12)

Question 4: D

In this question we chose the approach of finding the option which correctly show the congestion level 4 (white). From the tree we can see that A and C must havev b1>-0.16, which is correct only for option D.

Question 5: A

We have 5 outer folds, 4 inner folds, and for each fold we know that we train each ANN and logistic regression with 5 different parameters. So we have 5 outer folds \* 4 inner folds \* 5 parameters \* (time for ANN + time for logistic regression) 5\*5 \* 4 \*(25ms + 9ms)=3400. Since we have to test it two times for both outer and inner fold, we multiply it by 2 to receive the final result.

# References

|  |  |
| --- | --- |
| [1] | M. Mikšík a L. Málek, „Machine Learning & Data Mining Project1,“ 2022. [Online]. Available: github.com/martimik10/Project1\_ML. |
| [2] | DTU Compute, 2022. [Online]. Available: https://www.compute.dtu.dk/. |
| [3] | S. Stevens, „Ordinal and One-Hot Encodings for Categorical Data,“ 2020. [Online]. Available: machinelearningmastery.com/one-hot-encoding-for-categorical-data/. |
| [4] | T. Herlau, M. Schmidt a M. |
| [5] | Student, „The probable error of a mean,“ Biometrika, 1908. [Online]. |
| [6] | A. Horst, „Art work,“ 2021. [Online]. Available: https://github.com/allisonhorst/stats-illustrations. |
| [7] | S. Learn, „Linear Model Ridge Regression,“ 2022. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.RidgeCV.html. |

# Appendix 1

Chart, treemap chart

Description automatically generated

Figure: Confusion Matrix for Classification Models

Chart, scatter chart

Description automatically generated

*Figure: ROC Curve for Classification Models*

1. Art work by Allison Horst [6] [↑](#footnote-ref-2)
2. Martin 100%, Lukáš 0% [↑](#footnote-ref-3)
3. Found in Appendix 1 [↑](#footnote-ref-5)
4. Martin 100%, Lukáš 0% [↑](#footnote-ref-6)
5. Artifitial Neural Network [↑](#footnote-ref-7)
6. Nvidia Compute Unified Device Architecture for GPU acceleration [↑](#footnote-ref-8)
7. Found in Appendix 1 [↑](#footnote-ref-9)
8. Lukas 100% Martin 0% [↑](#footnote-ref-10)
9. Lukas 70%, Martin 30% [↑](#footnote-ref-11)
10. Lukas 70%, Martin 30% [↑](#footnote-ref-12)