


DANMARKS TEKNISKE UNIVERSITET



## 02450 - PROJECT 2

INTRODUCTION TO MACHINE LEARNING AND DATA MINING

Martin Mikšík s212075



Lukáš Málek s212074



1

---

<sup>1</sup> Art work by Allison Horst [6]

## Table of Contents

1	Introduction .....	1
2	Regression part I .....	1
2.1	Applying only non-categorical attributes .....	1
2.2	Applying all of the available attributes .....	2
2.3	Regularization and K-Fold Cross-Validation.....	2
2.4	Tuning.....	3
2.5	Testing on new data .....	3
3	Regression part II .....	4
3.1	The Artifitial Neural Network.....	4
3.2	The comparison .....	4
3.3	Statistical testing .....	5
4	Classification .....	5
4.1	Data Preparation .....	5
4.2	Regularization for Logistic Regression .....	6
4.3	Classification Models Comparison .....	7
4.3.1	KNN .....	8
4.3.2	Decision Tree.....	8
4.3.3	SVM (Support Vector Machine) .....	8
4.3.4	Logistic Regression .....	8
4.3.5	Baseline: .....	9
4.4	Statistical Testing .....	9
5	Discussion .....	10
6	Exam Questions .....	10
7	References.....	11
8	Appendix 1.....	12

# 1 Introduction

Project2 is a continuation of previous work on the Data set Visualisation Project [1], further referred to as the "Project1", for Denmark Technical University course 02450 Machine Learning & Data Mining [2] which we will further refer to as "the Course".

This document will further visualize, apply, and deepen our understanding of various Machine Learning and Artificial Intelligence algorithms like Regularized Regression, Neural Networks, and Classification.

Each of the following headings has a footnote with each team member's percentage contribution.

## 2 Regression part I <sup>2</sup>

In this section, we solve a relevant regression problem and statistically evaluate the result

### 2.1 Applying only non-categorical attributes

First, we will try Linear Regression on all non-categorical data, as discussed in Project1, as it would be a relatively straightforward and easily implemented approach. We are working with a data set with 344 instances and nine attributes:

Data columns (total 9 columns):				
#	Column	Non-Null Count		Dtype
---	-----	-----		-----
0	rowid	333	non-null	int64
1	species	333	non-null	object
2	island	333	non-null	object
3	bill_length_mm	333	non-null	float64
4	bill_depth_mm	333	non-null	float64
5	flipper_length_mm	333	non-null	int64
6	body_mass_g	333	non-null	int64
7	sex	333	non-null	object
8	year	333	non-null	int64

Figure 1: List of all attributes

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 features. Then, we have standardized the data, as there were excessive value differences in attributes:

$$\hat{X}_{ij} = \frac{X_{ij} - \mu_j}{\hat{s}_j}, \quad \mu_j = \frac{1}{N} \sum_{i=1}^N X_{kj}, \quad \hat{s}_j = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (X_{ij} - \mu_j)^2}$$

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

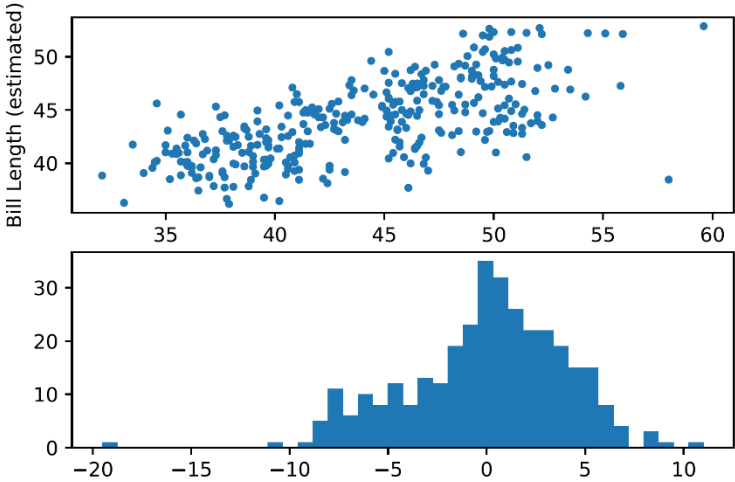


Figure 2: Histogram showing the variance of residuals

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

<sup>2</sup> Martin 100%, Lukáš 0%

## 2.2 Applying all of the available attributes

We need to improve the strategy by applying more attributes, as concluded above. As the other features are categorical, we need to be careful and pre-process 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 a computer understandable, integer value. However, using integer values for categorical data can cause a problem in several Machine Learning algorithms that could mistakenly see one Species as more important than the other.

Therefore, we will apply the Leave-One-Out Encoding [3] (also known as One-Hot) to all categorical data. We will also remove attributes that don't carry any meaningful value, like a 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 features we want to predict and store them in a vector  $y$ , which we will later use for training purposes. We have now created an input matrix  $X$ . ( $N = 333$  and  $M = 11$ , where  $y = \text{Bill Length}$ .)

#	Column	Non-Null Count	Dtype
0	bill_depth_mm	333 non-null	float64
1	flipper_length_mm	333 non-null	int64
2	body_mass_g	333 non-null	int64
3	Adelie	333 non-null	uint8
4	Chinstrap	333 non-null	uint8
5	Gentoo	333 non-null	uint8
6	Biscoe	333 non-null	uint8
7	Dream	333 non-null	uint8
8	Torgersen	333 non-null	uint8
9	female	333 non-null	uint8
10	male	333 non-null	uint8

#	Column	Non-Null Count	Dtype
0	rowid	333 non-null	int64
1	species	333 non-null	object
2	island	333 non-null	object
3	bill_length_mm	333 non-null	float64
4	bill_depth_mm	333 non-null	float64
5	flipper_length_mm	333 non-null	int64
6	body_mass_g	333 non-null	int64
7	sex	333 non-null	object
8	year	333 non-null	int64

Figure3: The transformation of raw data into input matrix  $X$

We will standardize the input data, as discussed in the previous attempt. However, we will standardize only the non-categorical data, as it would not help standardize One-Hot encoded attributes. Now, our first row of matrix  $X$  looks like this:

	0	1	2	3	4	5	6	7	8	9	10
0	0.780732	-1.42675	-0.568475	1	0	0	0	0	1	0	1

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

To quickly confirm the improvements made in this step, we will again plot the residuals. We can see a revision, and we can finally continue regularization and cross-validation.

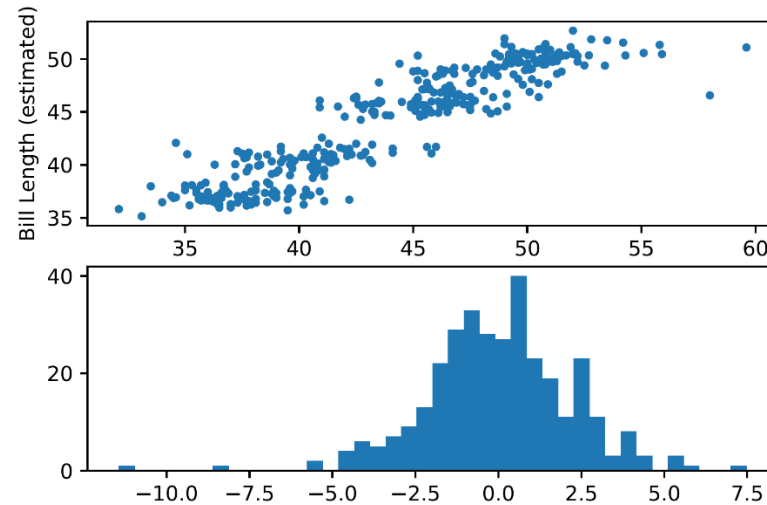


Figure 4: Plot of the residuals

## 2.3 Regularization and K-Fold Cross-Validation

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

Let's introduce the regularization term  $\lambda \|\omega\|^2$  that will penalize large weights, where  $\omega$  is the Model weights,  $y$  is the actual value and  $\hat{y} = y - \mathbb{E}$  is the prediction.  $\hat{X}$ ,  $\hat{x}$  are standardized  $X$  and  $x$ , respectively.

$$E_{\lambda}(\omega, \omega_0) = \sum_{i=1}^N (y_i - \omega_0 - \hat{x}^T \omega)^2 + \lambda \|\omega\|^2$$

By re-writing this into algebraic form, we can see that the model with  $\lambda = 0$  would be the same as the Linear Regression LSQ problem as in the first chapter.

$$E_\lambda = \|\hat{y} - \hat{X}\omega\|^2 + \lambda\|\omega\|^2$$

By solving the optimization problem, we can find the optimal coefficients (weights), Where  $\omega^*$  is the optimal weight, and  $\mathbb{I}$  is an identity matrix.

$$\omega^* = \frac{\hat{X}^T \hat{X} + \lambda \mathbb{I}}{\hat{X}^T \hat{y}}$$

We will conduct a 10-Fold Cross-Validation that will split the  $X$  data into training and testing sets in a 9:1 ratio and compute the error by comparing the predicted values with true values of  $y$  and repeat for different permutations of a split. Each split will give us the best  $\lambda$  that maximizes the performance of the fold

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.

## 2.4 Tuning

We have used the range of  $10^{-2}$  up to  $10^{2.5}$  linearly distributed by tens of  $\lambda$  values and tried different degrees of estimation polynomial to be 1 to 5. The second degree is the best for regression fitting with a Generalization error of  $5.557 \pm 1.904$  and regularization parameter  $\lambda$  of  $99.39 \pm 10.47$ , where the Generalization error is estimated by

$$E_{\mathcal{M}}^{\text{gen}} \approx \sum_{k=1}^K \frac{N_k^{\text{test}}}{N} E_{\mathcal{M},k}^{\text{test}}$$

$$E_i^{\text{Test}} = \frac{1}{N^{\text{Test}}} \sum_{i=1}^{N^{\text{Test}}} (y_i - \hat{y}_i)^2$$

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  $\lambda$  (right)

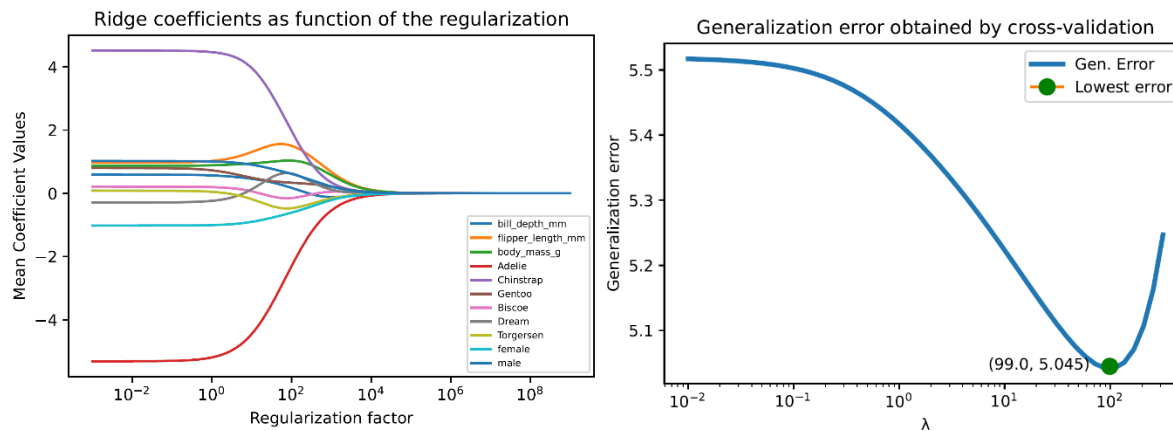


Figure 5: Effect of regularization

Even though we have paid the price for more complicated pre-processing and tuning methods, we have done everything we could to minimize the Generalization error in the Machine Regression problem with nested cross-validation.

## 2.5 Testing on new data

Now we know our model should be good at predicting 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 a few instances did not significantly impact 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 write a function<sup>3</sup> to mirror precisely this with our new data.

<sup>3</sup> Found in Appendix 1

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

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

Table 1: Results on seven unseen penguins

### 3 Regression part II <sup>4</sup>

This chapter will compare the Regression model from the previous section, Baseline and ANN<sup>5</sup>.

In the table below, we will show the best regularization parameter  $\lambda_i^*$  and the best number of hidden layers of the artificial neural network,  $h_i^*$  for each outer fold, together with the error measurement  $E_i^{Test}$  (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 in the previous section shows the importance of regularization in reducing the error. (we followed 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),

#### 3.1 The Artificial Neural Network

We create an Artificial Neural Network with a variable number of hidden units (from 1 to 10). We use nested 10-Fold cross-validation and validate the best number of hidden units. 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  $h^* = 1$  and  $h^* = 10$ :

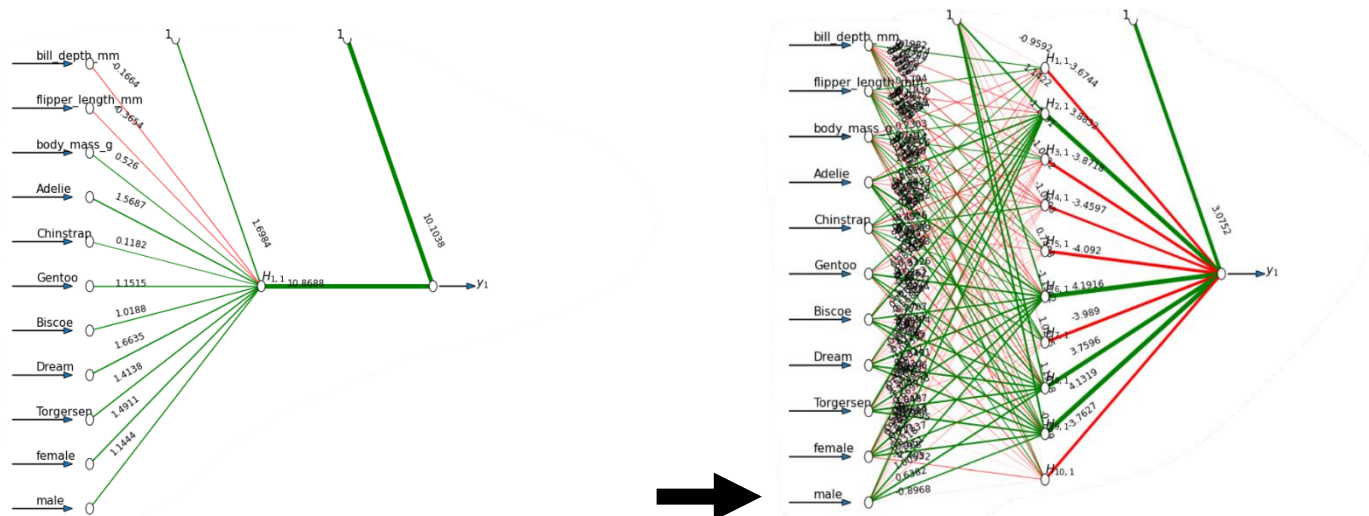


Figure 6: ANN visualisation of the inner fold

#### 3.2 The comparison

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 Artificial Neural Network model, as it is computationally heavy and needs hours to complete without CUDA<sup>6</sup> enabled. Furthermore, the ANN needs far more tuning parameters to be established. Although the Nested Cross-Validation for hidden units selection

<sup>4</sup> Martin 100%, Lukáš 0%

<sup>5</sup> Artificial Neural Network

<sup>6</sup> Nvidia Compute Unified Device Architecture for GPU acceleration

slows down the process tremendously, it is only one of many parameters to be optimized. Therefore, further analysis for stable results would be needed.

Outer Fold	ANN		Linear Regression		Baseline
i	$h_i^*$	$E_i^{Test}$	$\lambda_i^*$	$E_i^{Test}$	$E_i^{Test}$
1	8	4.7821	96.956	5.041	7.420
2	6	4.7852	119.448	5.045	3.251
3	5	4.7812	90.443	5.042	4.928
4	6	4.7836	111.424	5.043	4.568
5	7	4.7852	111.424	5.043	2.834
6	5	4.7802	96.956	5.041	6.480
7	6	4.7853	90.443	5.042	4.281
8	6	4.7875	96.956	5.041	5.198
9	8	4.7812	103.938	5.042	7.815
10	6	4.7803	84.367	5.043	4.260

Table 2: Two-level cross-validation table used to compare the three models

### 3.3 Statistical testing

We have conducted statistical tests between the models on unseen data<sup>7</sup> with Student's T-test [5]:

$$p_T(x | \nu, \mu, \sigma) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right) \sqrt{\pi\nu\sigma^2}} \left(1 + \frac{1}{\nu} \left[\frac{x-\mu}{\sigma}\right]^2\right)^{-\frac{\nu+1}{2}}$$

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 statistical testing for higher accuracy, the trade-off is a less confident test. We are using only a small set of new testing values, and we cannot fully satisfy the Central Limit Theorem and assumption of the Normal distribution 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 is the same, with a strong statistical significance.

## 4 Classification<sup>8</sup>

In classification, we are trying to predict the sex of the penguin. To make the classification as accurate as possible, we used five models: KKN, Decision Tree, SVM, Logistic Regression and Baseline. The classification problem of sex prediction is binary, as only female and male sex were observed for this dataset.

### 4.1 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%. After applying a one-hot encoder for species and islands, 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 it should not play on the penguin's size. The Pearson matrix is included in the first part of the report. Still, from the Correlation matrix, we can see that some attributes are highly correlated, especially the flipper length vs species and the flipper\_length vs body mass.

<sup>7</sup> Found in Appendix 1

<sup>8</sup> Lukas 100% Martin 0%

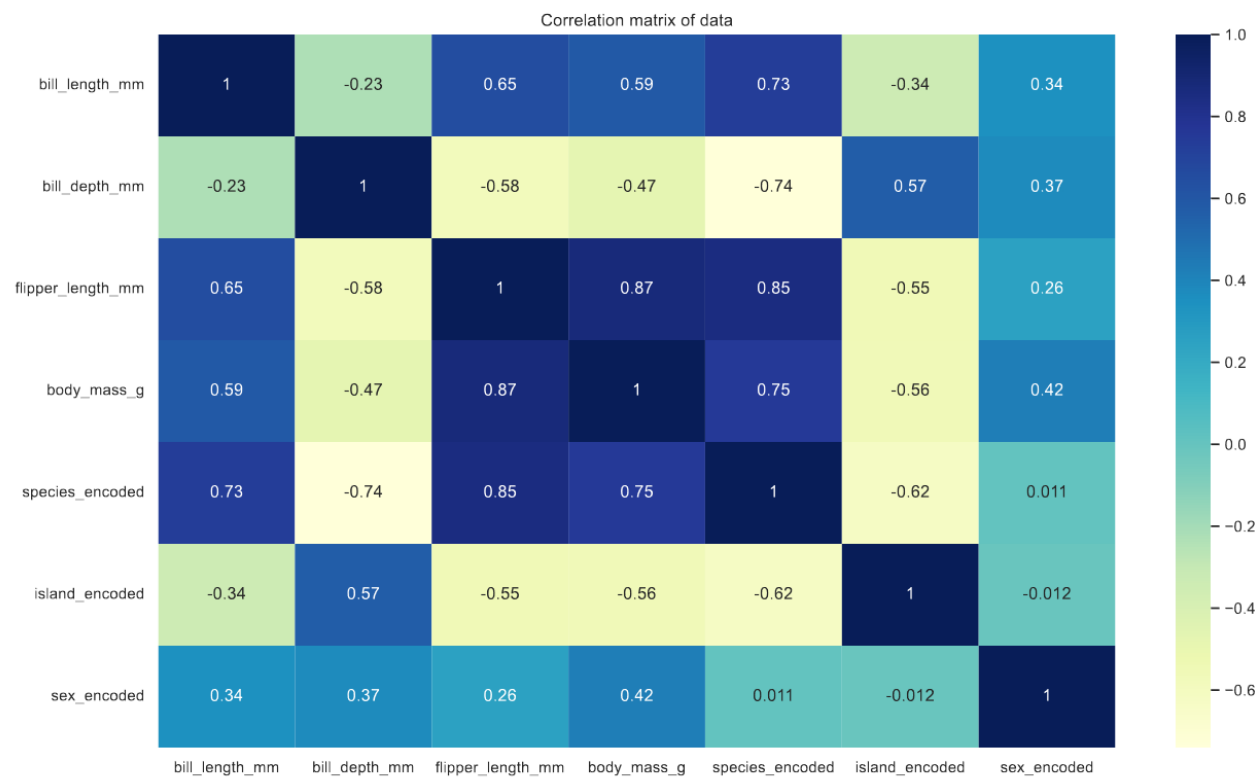


Figure7: Correlation between the attributes

However, not using the year attribute was a mistake, as the final result improved to a total Total Generalization Error Estimate of 9.61% after considering the year. It might be caused by different amounts of food in oceans or changing temperatures due to global warming. Also, removing the year feature was unnecessary since regularization will be performed, and the risks of overfitting thus are mitigated.

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

## 4.2 Regularization for Logistic Regression

Regularization improves the performance of new unseen data. Instead of squared loss, we use log loss, where  $(x,y)$  is the dataset  $D$  containing many labelled examples.  $Y$  is labelled as 0 or 1 since it is logistic regression, and  $Y'$  is the predicted value between 0 and 1.

$$\text{Log Loss} = \sum_{(x,y) \in D} -y \log(y') - (1 - y) \log(1 - y')$$

Regularization could be understood as a penalization for complexity. It adds bias if our model suffers from high variance, which means we overfit our data. Using L2 regression, we can lower the parameters' importance which worsens our results. Our goal is to minimize the Log Loss using a new hyperparameter  $\lambda$

$$L2: \quad \frac{\lambda}{2} \|w\|^2 = \frac{\lambda}{2} \sum_{j=1}^m w_j^2$$

$$J(w) = \sum_{i=1}^n [(-y^{(i)} \log y') - (1 - y^{(i)}) \log(1 - y')] + \frac{\lambda}{2} \|w\|^2$$

For the graph with the L2 norm (right), our goal is to have the lowest number possible since the algorithm is more affected by the regularization with a higher number. The graph below illustrates how the values were calculated. The lambda with the lowest error is used.



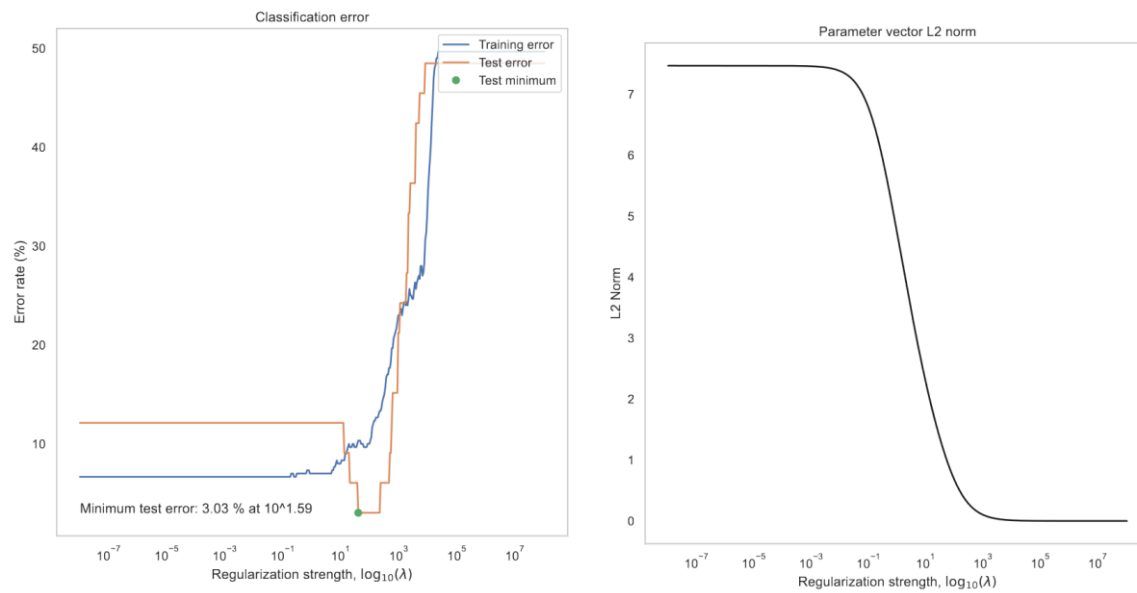


Figure 8: Choosing the correct regularization parameter  $\lambda$

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

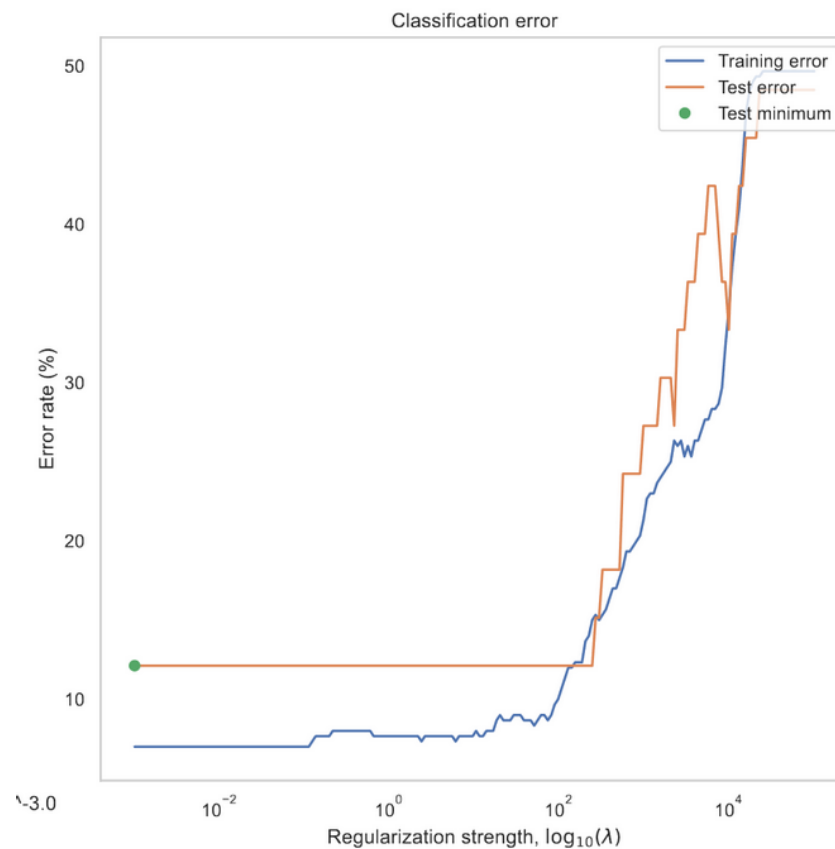


Figure 3: Regularization parameter

For the graphs that resulted in  $\lambda = 0$ , we can see that the error rate is the same until around  $\lambda = 100$ . Therefore it does not matter which  $\lambda$  on the 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.

### 4.3 Classification Models Comparison

This section compares models from the inner loop of the two-layer cross-validation. In the regression, we predict the bill depth. Still, in the classification problem, we solve the binary problem of sex. Therefore, we are not necessarily expecting the logistic and linear regression to be comparable in our case. Interestingly, the table shows an error rate higher than 50% for the baseline, although we have chosen the most frequent sex. The reason for it is the 2-layer cross-validation, as the test data in the outer layer contains primarily male instances.

INNER LOOP	KNN	DECISION TREE	SVM	LOGISTIC REGRESSION	BASELINE
i	$E_{ms,j}^{val}$	$E_{ms,j}^{val}$	$E_{ms,j}^{val}$	$E_{ms,j}^{val}$	$E_{ms,j}^{val}$
1	10.37%	16.72%	<b>10.03%</b>	10.03%	54.63%
2	14.38%	18.95%	<b>12.64%</b>	13.38%	55.37%
3	16.72%	18.90%	<b>11.52%</b>	12.26%	55.00%
4	14.67%	17.78%	<b>11.48%</b>	11.85%	54.81%
5	15.00%	14.81%	<b>10.37%</b>	10.74%	55.19%
6	12.33%	16.30%	12.96%	<b>11.85%</b>	55.19%
7	<b>10.67%</b>	18.89%	11.11%	11.48%	55.19%
8	12.00%	14.44%	11.48%	<b>9.63%</b>	55.19%
9	16.33%	14.81%	11.85%	<b>11.48%</b>	55.19%
10	14.00%	19.63%	11.48%	<b>11.11%</b>	54.81%
avg	13.65%	17.12%	11.49%	11.38%	55.06%

Table 2: Comparison of different Classification models in Inner Loop

#### 4.3.1 KNN

For KNN, we choose the best k neighbours in intervals 1-10 for each iteration, resulting in a costly computational model that does not deliver outstanding results. The SVM and logistic regression algorithms outperform this model. It could be fastened up by choosing a fixed value k, but then it would deliver even worst results

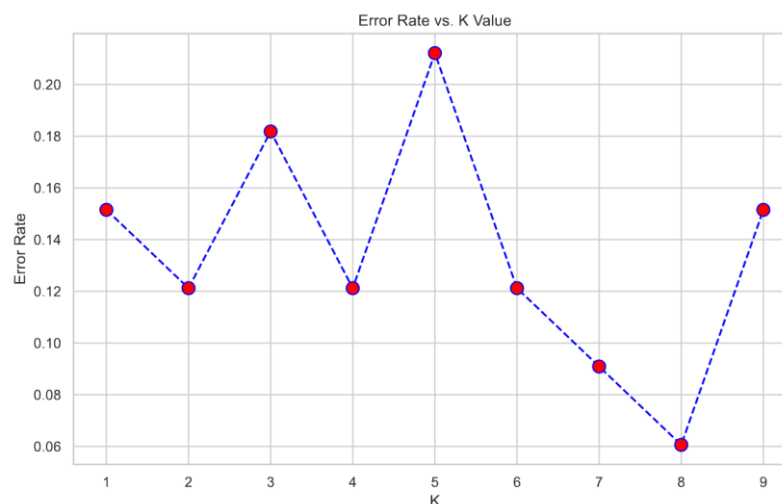


Figure 4: Finding the best K for the KNN model (graph for one of the many folds tested)

#### 4.3.2 Decision Tree

From the ten folds in the inner loop of the cross-validation, the error is stable at around 17%, which is about 5% more than for KNN, SVM or Logistic Regression. The decision tree algorithms often perform the best for various 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. We used the default parameter for DecisionTreeClassifier functions. Nevertheless, we could have managed 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.

#### 4.3.3 SVM (Support Vector Machine)

The SVM was chosen five times in inner folds, performing the best of all the models for our dataset. Generally speaking, SVM is effective for small training datasets. Also, in our case, the points can be separated by a hyperplane, which is not valid for every dataset.



#### 4.3.4 Logistic Regression

The 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 2<sup>nd</sup> place, right behind the SVM, as it was chosen as the best model 4 out of 10 folds.



### 4.3.5 Baseline:

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

## 4.4 Conclusion

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

OUTER FOLD		KNN	DECISION TREE	SVM		LOGISTIC REGRESSION	BASELINE
i	$k_i$	$E_i^{\text{test}}$	$E_i^{\text{test}}$	$E_i^{\text{test}}$	$\lambda$	$E_i^{\text{test}}$	$E_i^{\text{test}}$
1	7	14.71%	35.29%	<b>20.59%</b>	1862	23.53%	52.94%
2	4	8.82%	8.82%	<b>0.00%</b>	0	0.00%	47.06%
3	4	2.94%	23.53%	<b>5.88%</b>	38	5.88%	50.00%
4	6	6.06%	15.15%	<b>6.06%</b>	3548	6.06%	51.52%
5	1	12.12%	18.18%	<b>12.12%</b>	0	12.12%	48.48%
6	5	6.06%	9.09%	12.12%	38	<b>12.12%</b>	48.48%
7	7	<b>3.03%</b>	6.06%	6.06%	31	6.06%	48.48%
8	7	3.03%	12.12%	12.12%	0.93	<b>12.12%</b>	48.48%
9	1	12.12%	24.24%	9.09%	0.4	<b>9.09%</b>	48.48%
10	8	6.06%	36.36%	6.06%	3.47	<b>9.09%</b>	51.52%
avg	5	7.50%	18.88%	9.01%	552.2	9.61%	49.54%

Table 3: Error rate for classification models after applying cross-validation

After using the best performing models for each fold, we received a total Generalization Error of 9.09% for unseen data.

## 4.5 Statistical Testing

For statistical testing, we used McNemera's test function from the 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

Table 4: Statistical testing

With higher p-values, the confidence of the models with similar distributions increases. The results are not surprising. Models that delivered the lowest error predicted the test splits nearly the same, as we can see very high p-values for KNN-LOG.REG., SVM-LOG.REG. and KNN-SVM. Moreover, the best performing models are making the mistakes for the same instances in our dataset, increasing the p-values, as they deliver nearly the exact predictions.

Confidence intervals are more useful for ratio values used in linear regression rather than the binary output of sex.

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 worse than all other models that we tested.

## 5 Discussion <sup>9</sup>

Discussion for part I and part II are at the end of each section. In classification, we used five models. The baseline proved not to be very efficient for our dataset. Still, it can work very well as a threshold we want to reach with other models. SVM and logistic regression gave us the best results with the lowest error rates. Although the logistic regression was slightly better overall, it was also more computationally demanding. On the other hand, the SVM was a great surprise. It's fast, easy to understand, and has a very low error rate.

We tried to review our results with reports available on the internet. Unfortunately, no one has previously made that detailed report about this dataset. Therefore we cannot compare our results with published studies.

## 6 Exam Questions<sup>10</sup>

### Question 1:

The ROC curve is inaccurate, as the angle of the curve is far from the top left corner. We look for a prediction when we can reach the 50% TPR and 50%FPR, which is only possible for prediction D.

### Question 2:

Looking at the slides from week 5, we can find the formula to calculate the impurity gain. In our example, we have only one observation with  $x_7=2$ . Therefore, the impurity gain for this node is  $1/134 = 0.0074$ , which is result C.

### Question 4:

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

### Question 5:

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

---

<sup>9</sup> Lukas 70%, Martin 30%

<sup>10</sup> Lukas 70%, Martin 30%

## 7 References

- [1] M. Mikšík a L. Málek, „Machine Learning & Data Mining Project1,“ 2022. [Online]. Available: [github.com/martimik10/Project1\\_ML](https://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/](https://machinelearningmastery.com/one-hot-encoding-for-categorical-data/).
- [4] T. Herlau, M. Schmidt a M. Mørup, "Introduction to Machine Learning and Data Mining, "Technical University of Denmark, 2021. [Online].
- [5] Student, "The probable error of a mean, "Biometrika, 1908. [Online].
- [6] A. Horst, "Artwork, "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](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RidgeCV.html).

8 Appendix 1

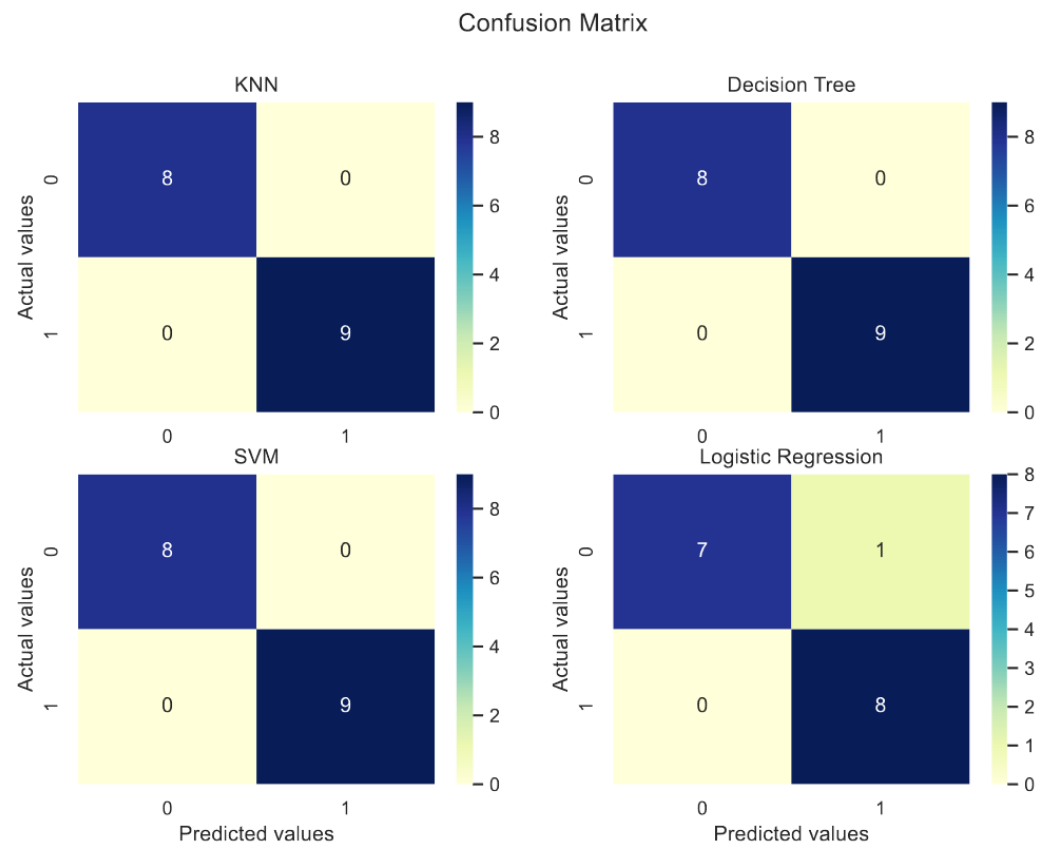


Figure: Confusion Matrix for Classification Models

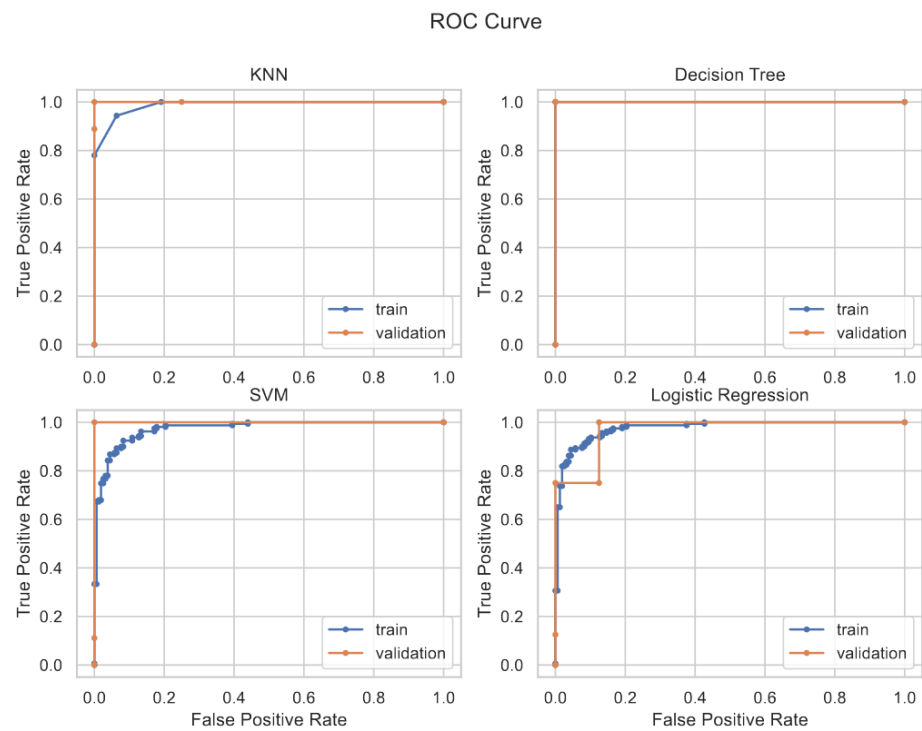


Figure: ROC Curve for Classification Models