02450
Machine Learning
Project 2

Written by:

Ditte Gilsfeldt *(s210666)*

Martin Surlykke *(s224793)*

	Regression	Classification	Discussion	Exam Question
Ditte	60%	40%	50%	50%
Martin	40%	60%	50%	50%

Dato: March 13, 2024

Contents

Introduction

Through the previous project, it wasn't possible to find a pattern of heart disease, even after the principal component analysis.

Based on the *cosine similarity martix*, from the previous project, there was a higher correlation between *Adiposity (adiposity)* and *Systolic Blood Pressure (sbp)*, where the correlation was 0.36. There was also a higher correlation between *Adiposity (adiposity)* and *Density Lipoprotein Cholesterol (ldl)* on 0.44 and *Adiposity (adiposity)* and *Obesity (obesity)* on 0.72 and *Adiposity (adiposity)* and *Age At Onset (age)* on 0.63, which indicate, there can be an association and a similarity pattern. The scatterplot below shows the correlation between the different predictor variables, which leads to the choice of the dependent and independent variables.

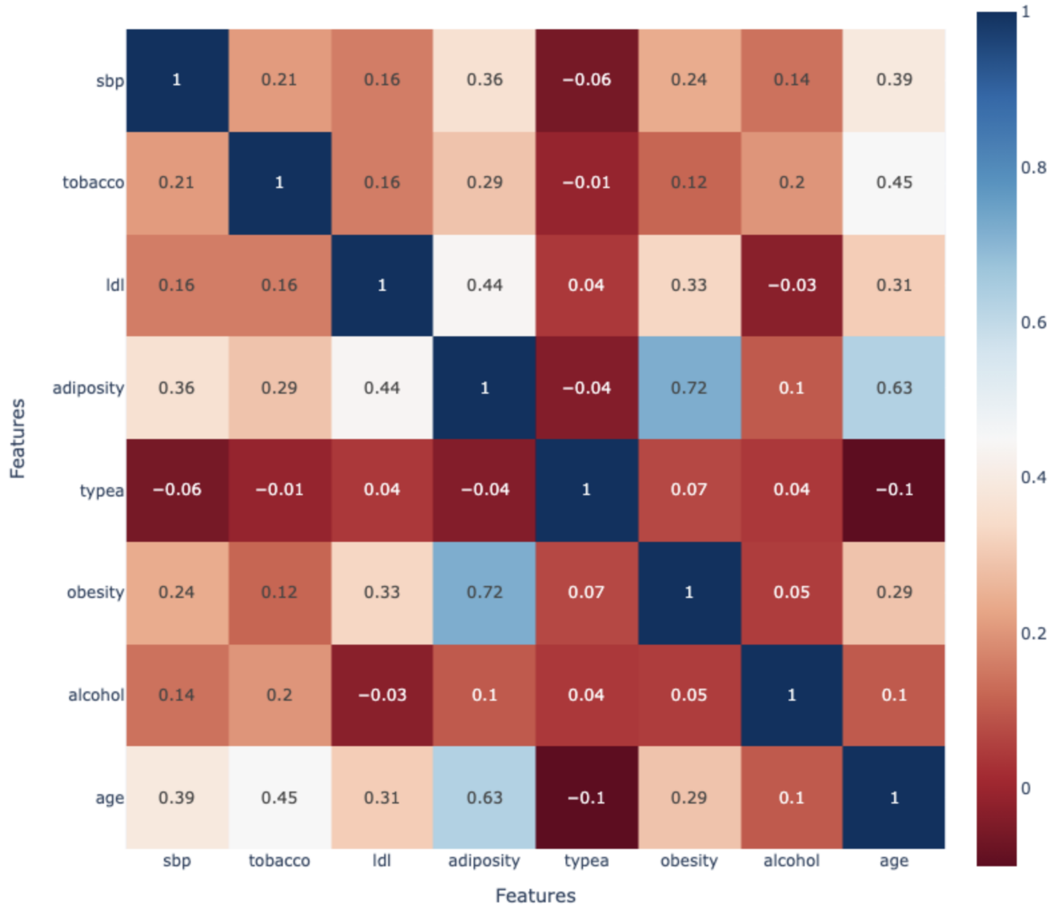


Figure 1: Cosine Similarity Matrix

In this project, the focus will be on trying to predict the value of adiposity in a given individual based on the aforementioned factors. This will be performed using *supervised learning*, where *classification* and *regression*, will be a part of the *supervised learning.*

Regression

Regression, part a:

A regression model is a model based on supervised learning with a given training dataset, where there is N observation $(x_1, ..., x_N)$ and N targets $(y_1, ..., y_N)$. The objective is to predict y from x, where in this case, the prediction is to estimate the value of the dependent variable(*adiposity*) based on the value of the independent variables (*sbp, ldl, obesity, age*). It is in interest to find out if there is a correlation between the dependent variable and the independent variable.

The starting point will be the linear transformation of the input features x, because the output y is a linear combination of the input features, such as the linear function can be written as:

$$f(\mathbf{x},\mathbf{w}) = w_0 + w_1 x_1 + ... + w_M x_M$$

This function represents the prediction of the model. If the coefficient is positive, the attributes will increase the output y, and if the coefficient is negative the attributes will increase the size of the effect on the output [1].

The dataset chosen in this assignment will be standardized by performing a data transformation. This data transformation consists of calculating and subtracting the mean for all observations in each prediction variable, whereafter the standard deviation will be calculated, and the observations will be divided by this value. This leaves a dataset exclusively of observations with a mean of 0, and a standard deviation of 1.
When transforming the data, the regression models can have a tendency to overfit and the technique, *regularization*, can be used to control the overfitting of a complex model[1].

The effect of regularization would be to look at the expression for $E_\lambda(w)$ when $\lambda = 0$, for the reason that it turns into an ordinary linear regression. The models will have a high variance and low bias when the regularization factor (λ) is small. When the regularization factor (λ) is large the model will have low variance and a high bias.
To find a better model, it is necessary to find the optimal λ value through the generalization error.

The graph below shows the mean values of the coefficients of predictor variables in the model (sbp, ldl, obesity, and age), as a function of the regularization strength (λ).

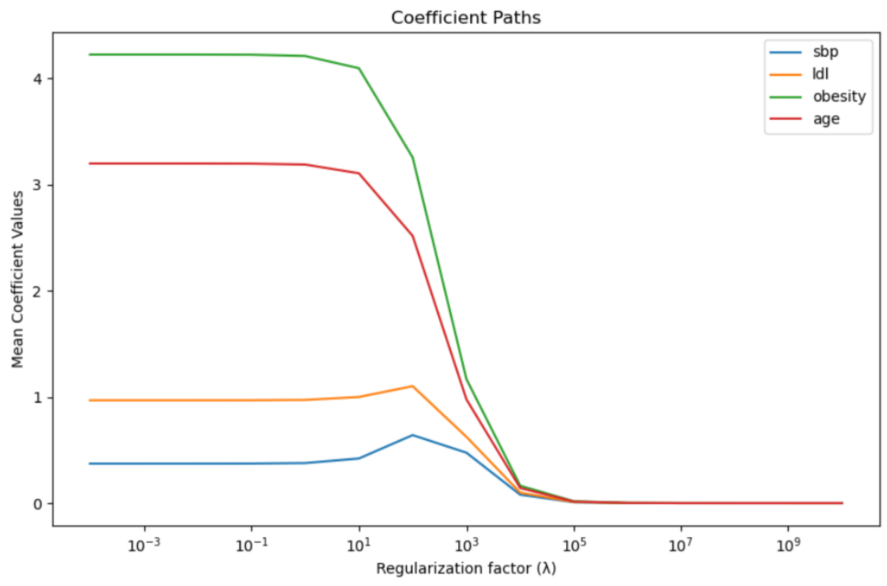


Figure 2: Coefficients for λ

The graph shows how regularization affects the coefficients. As λ increases, the coefficients are dragged closer toward 0. The values of λ tested are exponential, leading to a very drastic falloff in the range 10^{-4} to 10^{12}. This was done to show the full effect of regularization. When reaching the value $\lambda = 10^2$, a drastic decrease in the mean values is noticed, and after reaching $\lambda = 10^4$ the values of all coefficients are roughly the same, and very close to 0, effectively nullifying their effect on the regression model. With a regularization factor this large, the model will almost certainly be underfitted, leading to a model that can neither predict based on the training set or the validation set consistently. This is further shown in Figure 3 below.

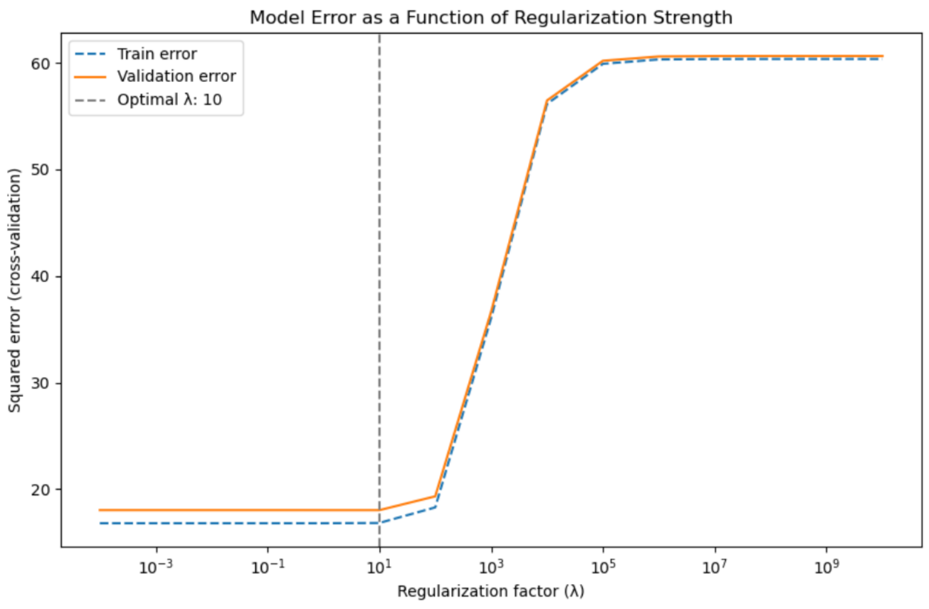


Figure 3: Training error

This graph shows the effect of the training error and the test error as a function of the regularization strength (λ) on the coefficients of predictor variables in the model (sbp, ldl, obesity, and age). As the regularization strength increases, it is seen that the delta-value between training error and validation error decreases. This shows how the regularization factor generalizes the regression model, making it more suitable to predict factors based on different data inputs. It is, however, also seen that after a certain size of regularization factor is reached (in this case $\lambda = 100$), the model becomes underfitted, leading to drastically increased training and validation errors.
At this point the regularization is impacting the coefficients to such an extent, that the model can no longer reliably predict the expected output on either the training set or the validation set. The optimal factor is chosen as the λ value where the validation error is lowest. In this case the dotted line points to this value: $\lambda = 10$

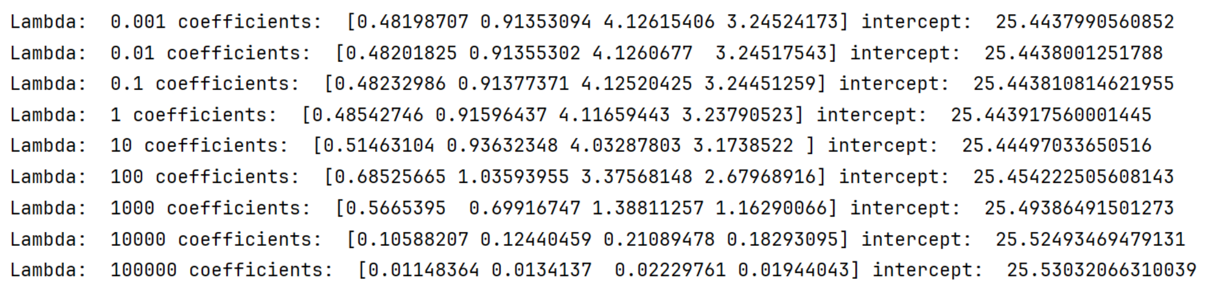
The output y of a linear model is calculated as a weighted sum of the input attributes x, with the addition of a bias term. For each attribute in x, the model assigns a coefficient that represents its importance or strength of association with the target variable y. Mathematically, this can be expressed as:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n$$

where:

- β_0 is the bias term.
- $\beta_1, \beta_2, ..., \beta_n$ are the coefficients for each attribute.
- $x_1, x_2, ..., x_n$ are the individual attributes of the input x.

To examine the computation of the value y, the $K = 10$ fold cross-validation is run, and the coefficients for all regularization parameters in a single loop of the cross-validation are recorded, as seen in Figure 4.



```
Lambda:  0.001 coefficients:  [0.48198707 0.91353094 4.12615406 3.24524173] intercept:  25.4437990560852
Lambda:  0.01 coefficients:  [0.48201825 0.91355302 4.1260677  3.24517543] intercept:  25.443801251788
Lambda:  0.1 coefficients:  [0.48232986 0.91377371 4.12520425 3.24451259] intercept:  25.443810814621955
Lambda:  1 coefficients:  [0.48542746 0.91596437 4.11659443 3.23790523] intercept:  25.443917560001445
Lambda:  10 coefficients:  [0.51463104 0.93632348 4.03287803 3.1738522 ] intercept:  25.44497033650516
Lambda:  100 coefficients:  [0.68525665 1.03593955 3.37568148 2.67968916] intercept:  25.454222505608143
Lambda:  1000 coefficients:  [0.5665395  0.69916747 1.38811257 1.16290066] intercept:  25.49386491501273
Lambda:  10000 coefficients:  [0.10588207 0.12440459 0.21089478 0.18293095] intercept:  25.52493469479131
Lambda:  100000 coefficients:  [0.01148364 0.0134137  0.02229761 0.01944043] intercept:  25.53032066310039
```

Figure 4: Coefficients and intercept

Figure 4 shows the calculated coefficients for all 4 prediction variables, based on the according regularization factor. The intercept corresponds to the bias term, a constant value added to the output y.
It is seen that the bias term is fairly constant across all values of λ, at roughly 25.4, only growing a little with very large regularization parameters. The coefficients, on the other hand, are shown to change based on the λ. As the lambda increases, the coefficients are dragged closer to 0, as shown in Figure 2.

y can hereby be calculated by simply inserting all coefficients for a specific value of λ. For example, when calculating the output for the value $\lambda = 0.001$ the function becomes:

$$y = 25.44 + 0.482 \cdot x_1 + 0.914 \cdot x_2 + 4.126 \cdot x_3 + 3.245 \cdot x_4$$

Where x_1, x_2, x_3, x_4 are the values of the prediction variables.

Regression, part b:

When working with regression, there are several different models to choose from, which work to different levels of effectiveness for different problems. In this project, 3 models, a *Linear Regression model, Artificial Neural Network (ANN) model* and a *baseline model* will be tested and compared, to explore which is the optimal model for the given objective.

To test and compare this, the method *two layer cross-validation* can be used. This requires obtaining the two-step procedure and making use of the *nested cross-validation procedures* method and both select the model and estimate its performance and this is the *two-layer cross-validation*[1].

To do this, the data set is split into K_1 parts, the K_1'th split is then further split into K_2 parts. In this loop, both the linear regression model and the ANN model are further tested with different hyperparameters. These are the regularization parameters for the linear regression and hidden layers for the ANN. Both models are tested, and the results for all used hyperparameters are saved.
After K_2 iterations of this loop, the test errors for all iterations are summed up for the hyperparameters respectively, and the best models are chosen. The best models for the linear regression and ANN are then trained on the training data in the K_1'th split, and used to predict the values in the validation data of the same split. The error in this prediction is then recorded and saved for further analysis.

The baseline model works by taking the mean of the training data in the K_1'th split, and then predicting all values in the validation data of the K_1'th split to be this value.

To create the models in Python, the libraries which are used are:

- *from sklearn.linear_model import Ridge*
- *from keras.models import Sequential*
- *from keras.layers import Dense*
- *from keras.optimizers import Adam*

The range of hyperparameters tested has been found using empirical testing. After the two-level cross-fold validation has been run, the results are gathered and entered into a table for comparison, which is shown in the table below:

Outer fold	ANN		Linear regression		baseline
i	h_i^*	E_i^{test}	λ_i^*	E_i^{test}	E_i^{test}
1	16	16.41	0.001	16.44	68.30
2	1024	31.45	0.001	29.96	35.87
3	16	14.99	0.001	14.90	62.74
4	256	13.75	0.001	14.05	67.47
5	1024	11.84	100	13.87	61.56
6	4096	14.97	100	17.43	62.22
7	1024	14.51	100	17.75	70.36
8	4096	30.24	0.001	28.31	73.28
9	4096	11.08	0.001	11.98	52.50
10	1024	17.82	0.001	17.91	53.24

Table 1: Generalization errors, and chosen regularizing factors/hidden units for the ANN, linear regression and baseline.

The table shows the ANN model's varying degrees of effectiveness across the folds, as indicated by the change in the number of hidden units (h_i^*) and the corresponding generalization errors (E_i^{test}). The best performance of the ANN model is observed in fold 9 with a generalization error of 11.08 and 4096 hidden units. The worst performance is seen in fold 2 with a generalization error of 31.45, and 1024 hidden units. This variability suggests that the number of hidden units alone does not determine performance and that other factors, possibly including the characteristics of the data in each fold, play a significant role.

The linear regression model's performance is also variable but shows less variability in error compared to the ANN. The best linear regression performance occurs in fold 9 with a very low regularization factor ($\lambda_i^* = 0.001$) and the lowest error of 11.98. The highest error is seen in fold 2, with a value of 29.96, also with a value of 0.001.

The variability in the regularization factor suggests that different levels of penalty on the model's complexity are optimal for different subsets of the data.
The baseline model consistently underperforms in comparison to the ANN and linear regression models, with generalization errors ranging from 35.87 to 73.28. This high error rate across all folds indicates that the baseline model's simplicity is not sufficient to capture the underlying patterns in the data effectively.
When comparing the ANN and linear regression models, there is no clear, consistent winner. Each model outperforms the other in different folds, implying that the choice between using an ANN or linear regression could depend on the specific characteristics of the dataset being used in each fold.

To determine the most effective model for our dataset, statistical evaluation using a paired t-test is conducted based on the Mean Squared Error (MSE) derived from the 10 different K-folds in the cross-validation. This test involves a pairwise comparison of the p-value and the confidence interval for the three models: ANN, linear regression (referred to as 'ridge' in this context), and a baseline model. The significance level that has been used is $\alpha = 0.05$.

The paired t-test results are summarized below in a statistical analysis.
Ridge compared to ANN:

- Confidence interval: -0.627129205403734; 1.733072102338871
- P-value: 0.3167612203140713

Ridge compared to Baseline:

- Confidence interval: -52.526188734754676; -32.46240978794912
- P-value: 5.097962938908082e-06

ANN compared to Baseline

- Confidence interval: -53.69373740308674; -32.4008040165522
- P-value: 7.479021116988199e-06

Based on this analysis, we observe different comparative performances among the models:

- Ridge compared to the Baseline: The confidence interval and p-value indicate that the ridge regression model significantly outperforms the baseline. The large negative confidence interval range and a p-value of 5.098 demonstrate a much lower MSE for the ridge regression.
- ANN compared to the Baseline: Similar to the ridge model, the ANN also shows superior performance over the baseline, as indicated by the confidence interval and a p-value of 7.48. The ANN model achieves a considerably lower MSE.
- Ridge compared to the ANN: The comparison between ridge regression and ANN shows a more nuanced result. The confidence interval includes both negative and positive values, and the p-value is 0.317, suggesting no clear winner in terms of MSE between these two models.

Based on the data both the ANN- and linear regression models significantly outperform the baseline model. The Linear regression is more stable across folds than the ANN model, suggesting it is a preferable choice for consistency. There is no significant difference in performance between the ANN and linear regression, as indicated by the p-value and confidence intervals.
Consider the trade-off between stability (favoring linear regression) and the ability to model complex relationships (potentially favoring the ANN). Furthermore, the statistical analysis is recommended to make a definitive choice, along with considerations for model complexity and computational resources.

Classification

This part will explore binary and multi-class classification using logistic regression and ANN. Neural networks, typically used for multi-class cases, can be adapted for classification similarly to how linear regression transforms into logistic regression for binary classification. While logistic regression is originally for binary classification, it can also be extended to handle multi-class problems[1].
When deciding whether to do a multi-class or binary classification. Research was made into the effect of adiposity, to see if any exciting real-life cases could be predicted for the given dataset. No meaningful data about the effect of adiposity was found, and as such it was decided to use the data to make a binary classification.
The mean of the values in column 'adiposity' is calculated and every value under the mean is set to 0, and every value above is set to 1. From the 4 possible models, an ANN is again used, together with a logistic regression model.

The two-layer cross-validation approach is again used for training and evaluating the models. In the outer layer, by applying the 10-fold cross-validation to measure each model's overall performance. At the same time, the inner layer, also employing a 10-fold method, focuses on optimizing the hyperparameter. This includes adjusting the learning rate (λ) in the Logistic Regression model and tuning the number of hidden units in the ANN model. During each iteration of this inner process, the models are trained on a specific portion of the data and evaluated based on their accuracy.
The table below shows the accuracy of the generalization errors, of an ANN, logistic regression, and a baseline model across ten folds of cross-validation.

Outer fold	ANN		Logistic regression		baseline
i	h_i^*	E_i^{test}	λ_i^*	E_i^{test}	E_i^{test}
1	256	0.106	0.001	0.064	0.362
2	4	0.170	0.001	0.106	0.468
3	1	0.326	1	0.326	0.478
4	1	0.152	1	0.174	0.457
5	4	0.174	0.001	0.174	0.413
6	16	0.130	0.001	0.174	0.500
7	64	0.130	10	0.109	0.391
8	1	0.174	10	0.174	0.370
9	16	0.196	0.001	0.217	0.587
10	4	0.087	0.001	0.087	0.565

Table 2: Generalization errors, and chosen regularizing factors/hidden units for the ANN, linear regression and baseline.

The ANN and logistic regression models show varying performance but consistently outperform the baseline, with the lowest errors observed in fold 10 for both models. This suggests that the complexity-controlling parameter chosen for fold 10 (256 hidden units for ANN and $\lambda = 0.001$ for logistic regression) was most effective for that data segment. The results in this table are shown as the amount of wrong predictions divided by the total amount of predictions. Thus an error score of 0.106 equates to an error rate of 10.6%.

The models have been statistically compared pairwise in a paired t-test. The paired t-test results are summarized below in a statistical analysis.
Ridge compared to ANN:

- Confidence interval: -0.02676787556080601; 0.01853475807699472
- P-value: 0.6906046748965202

Ridge compared to Baseline:

- Confidence interval: -0.36522153557491455; -0.2319107493464546
- P-value: 3.2058604485779045e-06

ANN compared to Baseline

- Confidence interval: -0.363823138770289; -0.22507602866726878
- P-value: 5.014291152224055e-06

The paired t-test results indicate that Ridge regression is not significantly different from the ANN model in performance, as suggested by a confidence interval of -0.027 to 0.019 and a high p-value of 0.69. However, both Ridge regression- and ANN models are significantly better than the baseline model, with confidence intervals of -0.365 to -0.232 for the Ridge model and -0.363 to -0.225 for the ANN model and very small p-values, approximately 3.21e-06 for the Ridge model and 5.01e-06 for the ANN model, which indicates a much lower error.

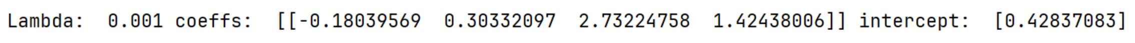
In logistic regression, again the coefficients for each of the prediction variables are calculated. However, in logistic regression, the output y is calculated using the Bernoulli distribution.
As in the regression part a, the coefficients are firstly used to calculate a value, in this case, z:

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$$

After this, the value z is used to calculate y with Bernoulli distribution:

$$y = \frac{1}{1 + e^{-z}}$$

With a chosen λ of 0.001, the logistic coefficients of the logistic regression are calculated to be:



```
Lambda:  0.001 coeffs:  [[-0.18039569  0.30332097  2.73224758  1.42438006]] intercept:  [0.42837083]
```

Figure 5: Coefficients for logistic regression

Thus the calculation of y becomes:

$$y = \frac{1}{1 + e^{-z}} = \frac{1}{1 + e^{-(0.428 - 0.180 \cdot x_1 + 0.303 \cdot x_2 + 2.732 \cdot x_3 + 1.424 \cdot x_4)}}$$

This equation will compute a value between 0 or 1 depending on the given inputs, denoting the probability of y being either 1 or 0.

Discussion

During this project, the learning process has been to explore and understand the method for regression and classification. From this, the knowledge about regression and classification is to find a balance between an over- or underfit model. Both a too high- or a too low value of λ will over- or underfit the model, so a right value for λ was necessary to find.

Through working with the regression part, the learning has been to find an appropriate regression model for this data such as a linear regression- and ANN model. By exploring the regularization method for the regression it helped in addressing overfitting and making the model generalize better to unseen data.

Through working with the classification part, the learning has been to determine the different values and how they affect the output and the model predictions. By using the hyperparameter, the learning was to find out that the model performance was affected significantly.

In general, the learning was to understand how regression and classification, could affect the dataset, and how to train a model, based on some different factors, which can lead to a "perfect" test of some unseen data in the dataset.

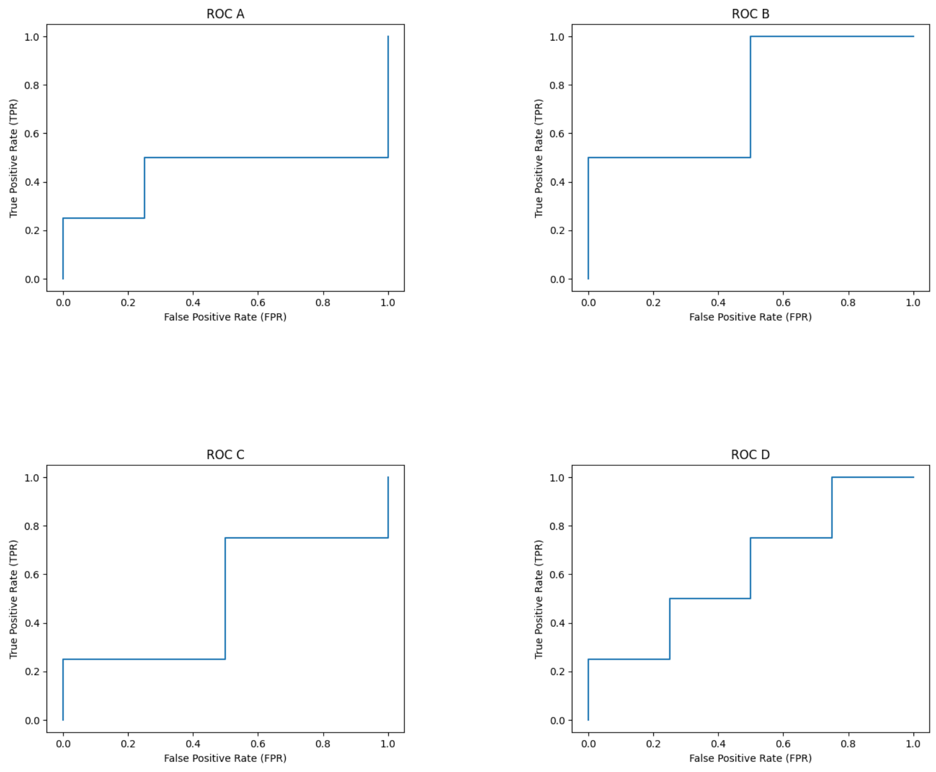
During this project, multiple methods of supervised learning have been performed, using different models, namely the linear regression, logistic regression and artificial neural network.
Through working with these models, it has been found that the linear regression and logistic regression are very effective at doing specific tasks, showing a performance equal or better than the artificial neural network, while the artificial neural network has been shown to be a very effective and versatile tool, being capable of solving multiple tasks.
The trade-off, however, is the performance of the models. Where the linear and logistic regressions could be trained in mere seconds training a neural network with an adequate number of hidden units would take close to an hour, showing the importance of choosing the correct model for the given task.

Comparing this project to another existing study has not been possible, for the reason that this study in this project, has not been done before.

Exam Question

Question 1. Spring 2019 question 13:

The answer is **C**.
The reason why the answer is **C** is that the curve of **ROC C** is the same as Figure 1 in the question. The figure below shows the result.



Question 3. Spring 2019 question 18:

The answer is **A**:
The reason why the answer is **A** is that is shown below:

1. Hidden Layer:
- Weight Parameters: $10\,\text{units} \cdot 7\,\text{input features} = 70\,\text{weight parameters}$.
- Bias Parameters: 10 units.

2. Output Layer:
- Weight Parameters: $4\,\text{units} \cdot 10\,\text{units in the hidden layer} = 40\,\text{weight parameters}$.
- Bias Parameters: 4 units.

Total Parameters = 70 (hidden layer) + 40 (output layer) + 10 (hidden layer bias) + 4 (output layer bias) = **124** parameters to be trained.

Question 4. Spring 2019 question 20:

The answer is **D**:
The reason why the answer is **D** node A in the decision tree is divided into two categories. The first category includes congestion points 1 and 2, which are associated with a false condition, and the second category includes congestion points 1, 3, and 4, associated with a true condition. Reviewing the separation line in Figure 4 leads to the conclusion for node A that $b_1 \geq -0.76$.

Further division of the decision tree shows congestion point 1 corresponding to a false condition and congestion point 2 corresponding to a true condition. This observation yields the statement for node B that $b_2 \geq 0.03$

Node C, following a similar pattern, splits into two. The first includes congestion points 3 and 1 with a false condition, and the second includes congestion point 4 with a true condition. This informs the statement for node C, with the limiting factor for congestion point 4 being b_1. This means that node C is $b_1 \geq -0.16$

Node D is also divided into two parts. The first part pertains to congestion point 3 with a false condition, and the second pertains to congestion point 1 with a true condition. The resulting statement for node D is $b_2 \geq 0.01$

Question 6. Spring 2019 question 26:

The answer is **B**:
The reason why the answer is **B** is that it has the highest probability in class 4, by creating a bias vector from the *observation* b vector, given from the answer choices, and calculating the probabilities for each observation.

References

[1] Tue Herlau, Mikkel N. Schmidt, and Morten Mørup. *Introduction to Machine Learning and Data Mining*, pages 25, 26, 139, 140, 174, 175, 177, 245, 247, 248, 265. Polyteknisk Kompendie, 2023.

Appendix

Appendix 1 - Exam Question

Question 1. Spring 2019 question 13:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

def ROC_plot(x, y, label, threshold=[0, 1]):
    # Lists to store True Positive Rate (TPR) and False Positive Rate (FPR) values.
    tpr_values = []
    fpr_values = []

    for theta in [threshold[0]] + x + [threshold[1]]:
    # Convert probabilities to binary predictions based on the threshold.
        binary_predictions = (np.array(x) > theta).astype(int)

        # Calculate True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN).
        TP = np.sum((binary_predictions == y) & (binary_predictions == 1))
        FP = np.sum((binary_predictions != y) & (binary_predictions == 1))
        TN = np.sum((binary_predictions == y) & (binary_predictions == 0))
        FN = np.sum((binary_predictions != y) & (binary_predictions == 0))

        # Calculate the True Positive Rate (TPR) and the False Positive Rate (FPR).
        TPR = TP / (TP + FN)
        FPR = FP / (FP + TN)

        # Append the True Positive Rate (TPR) and the False Positive Rate (FPR) values to the lists.
        tpr_values.append(TPR)
        fpr_values.append(FPR)

    # Create a DataFrame to store the True Positive Rate (TPR) and the False Positive Rate (FPR) values.
    rates = pd.DataFrame({'FPR': fpr_values, 'TPR': tpr_values})

    # Plotting the ROC curve with the given label.
    plt.plot(rates['FPR'], rates['TPR'])
    plt.xlabel('False Positive Rate (FPR)')
    plt.ylabel('True Positive Rate (TPR)')
    plt.title(label)
    plt.show()

```

```

x = [0.4, 0.5, 0.6, 0.65, 0.7, 0.75, 0.9, 0.95]

Y = [
    [1, 1, 0, 0, 0, 1, 0, 1],
    [0, 0, 1, 1, 0, 0, 1, 1],
    [1, 0, 0, 1, 1, 0, 0, 1],
    [0, 1, 0, 1, 0, 1, 0, 1],
]

# Labels for ROC curves
roc_labels = ['ROC A', 'ROC B', 'ROC C', 'ROC D']

# Generate separate ROC plots for each set of data with labels.
for i, y in enumerate(Y):
    ROC_plot(x, y, label=roc_labels[i])
```

Question 6. Spring 2019 question 26:

```
import numpy as np

# Calculates y_k.
def y_k(b1, b2):
    return [1, b1, b2]

# Weight vectors for the assignment.
w1 = [1.2, −2.1, 3.2]
w2 = [1.2, −1.7, 2.9]
w3 = [1.3, −1.1, 2.2]
W = [w1, w2, w3]

# Bias vectors.
B = [
    [−1.4, 2.6],
    [−0.6, −1.6],
    [2.1, 5.0],
    [0.7, 3.8]
    ]

# Store the probability for class 4 for each observation.
class_4_probs = {}

# Calculate the probabilities for each observation.
for i, b in enumerate(B):
    PCD = np.array(y_k(*b))
    y_hat = lambda w: np.dot(PCD, np.array(w).T)
```

```python
    y_hats = [np.exp(y_hat(w)) for w in W]

    print(f"Observation {i+1}")

    # Calculate probabilities for classes 1 to 3.
    probabilities = [y_hat / (1 + sum(y_hats)) for y_hat in y_hats]
    for k, probability in enumerate(probabilities, start=1):
        print(f"P(y={k}|y_hat) = {probability:.2%}")

    # Calculate the probability for class 4.
    probability_4 = 1 / (1 + sum(y_hats))
    print(f"P(y=4|y_hat) = {probability_4:.2%}")

    # Store the probability for class 4.
    class_4_probs[i] = probability_4

    print('-' * 30)

# The highest probability for class 4.
max_class_4_observation = max(class_4_probs, key=class_4_probs.get)
max_probability = class_4_probs[max_class_4_observation]

# The observation with the highest probability for class 4.
print(f"Observation {max_class_4_observation + 1} has the highest probability for class 4: {max_probability:.2%}")
```

```
# Output

Observation 1
P(y=1|y_hat) = 78.00%
P(y=2|y_hat) = 20.42%
P(y=3|y_hat) = 1.58%
P(y=4|y_hat) = 0.00%
------------------------------
Observation 2
P(y=1|y_hat) = 5.11%
P(y=2|y_hat) = 6.50%
P(y=3|y_hat) = 15.35%
P(y=4|y_hat) = 73.05%
------------------------------
Observation 3
P(y=1|y_hat) = 63.38%
P(y=2|y_hat) = 32.76%
P(y=3|y_hat) = 3.85%
P(y=4|y_hat) = 0.00%
------------------------------
Observation 4
```

```
P(y=1|y_hat) = 67.89%
P(y=2|y_hat) = 28.73%
P(y=3|y_hat) = 3.38%
P(y=4|y_hat) = 0.00%
________________________________
Observation 2 has the highest probability for class 4: 73.05%
```