Report 2

Machine Learning and Data Mining [02450]

Mia Due Paarup - s213304

Joaquim Siqueira Lucena - s232535

Gokul Desu - s242580

Section	Mia	Joaquim	Gokul
Regression	25%	50%	25%
Classification	25%	25%	50%
Discussion	50%	25%	25%
Exam problems	Equally	Equally	Equally

Table 1: Contribution (%) to each section of the report.

DTU Technical University of Denmark

1 Regression

In our original assignment, we did not clearly define how our dataset would be used for regression analysis. Since our primary target variable, which represents the death event of a patient during the follow-up period, is a binary (yes/no) value, we cannot use linear regression techniques to model it.

Instead, we have chosen to select another feature from our dataset to conduct regression studies and demonstrate our proficiency in creating and using regression models. This will allow us to showcase our ability to apply appropriate regression techniques when the target variable is not binary.

However, our main focus remains on classification tasks. Our primary goal is to understand the correlations between the various features in our dataset and whether a patient dies from cardiac diseases or not. The regression analysis is a supplementary component to exhibit our broader analytical capabilities.

By clarifying our approach to both classification and regression, we can provide a more comprehensive overview of our data analysis strategies and techniques in the report.

1.1 Regression target

For our regression analysis, we have chosen to use the level of serum sodium in the blood (measured in mEq/L) as the target variable. We believe this is an interesting choice, as serum sodium levels play a crucial role in maintaining proper fluid balance and nerve/muscle function, and may be significantly affected by various cardiovascular conditions.

Some potential reasons why the serum sodium variable could generate interesting regression results include:

1. Relationship with age: As patients age, we may observe changes in sodium regulation that could be captured by the regression model. The age feature may be a significant predictor of serum sodium levels, particularly as kidney function often declines with age.

2. Influence of medical conditions: Features like diabetes, ejection fraction, and high blood pressure could all potentially impact an individual's serum sodium levels. For instance, heart failure patients often show abnormal sodium levels due to fluid retention and neurohormonal adaptations. Regression analysis could help uncover these relationships.

3. Gender differences: The sex feature may reveal interesting differences in sodium regulation between males and females, which could be worth investigating further, especially given known gender differences in cardiovascular disease manifestation.

4. Interaction with other blood markers: Variables like creatinine phosphokinase and serum creatinine may show meaningful associations with sodium levels, potentially revealing patterns in kidney function and overall blood chemistry. Regression can help identify these interdependencies.

By focusing on serum sodium as the regression target, we can gain valuable insights into the factors that influence this crucial electrolyte. This knowledge could ultimately contribute to a better understanding of fluid balance regulation in cardiovascular patients and potentially identify early warning signs of heart failure decompensation.

Additionally, the regression analysis will complement our primary focus on classification tasks, where we aim to determine the correlations between various features and the patient's risk of dying from cardiac diseases. The combination of classification and regression modeling will provide a more comprehensive view of the dataset and the underlying relationships within it, particularly as abnormal sodium levels are often associated with poor cardiovascular outcomes.

1.2 Implementation

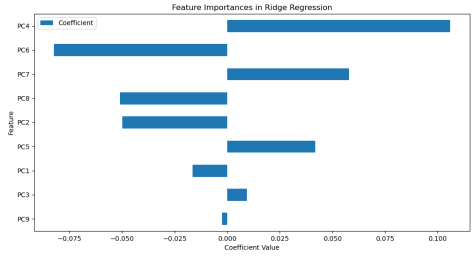
To prepare our data for analysis, we first standardize all features using scikit-learn's `StandardScaler`. This preprocessing step centers the data by subtracting the mean and scales it by dividing by the standard deviation, ensuring all features have zero mean and unit variance. This standardization is crucial as it prevents features with larger numerical ranges from dominating the analysis.

To reduce the dimensionality of our feature space, we apply Principal Component Analysis (PCA). Our objective is to retain enough principal components to explain at least 85% of the total variance in our dataset. However, as previously observed in Report 1, our features exhibit relatively low correlation, resulting in a distributed variance structure. Consequently, we could only achieve a modest dimensionality reduction from 11 to 9 components while maintaining our target explained variance threshold.

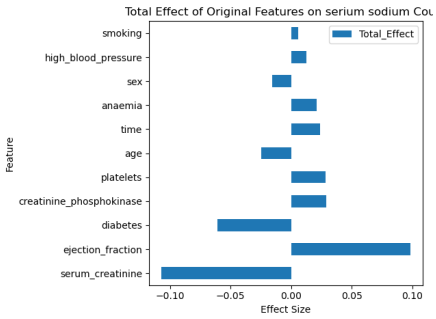
For the regression analysis, we implement the `Ridge` regression model from scikit-learn, employing a 10-fold cross-validation strategy to ensure robust evaluation. The regularization parameter λ is optimized by testing 40 different values, logarithmically spaced between 10^{-4} and 10^{4}. This logarithmic spacing allows us to explore a wide range of regularization strengths efficiently, from very weak (10^{-4}) to very strong (10^{4}) regularization.

1.3 Results

The feature importance from PCA and the effects of the original features on the output are shown in Figures 1a and 1b. These results confirm the correlation matrix observed in Report 1, where serum creatinine and ejection fraction were identified as the most strongly correlated features with serum sodium.



(a) PCA importance on the regression of serum sodium



(b) Importance of original features on the regression of serum sodium

Figure 1: Importance plot

Our results for the λ experiment are displayed on Fig. 2. It shows a typical curve for regularization, in which the generalization error initially decreases as λ increases, until the regularization ends up being too strong and causing the model to underfit. In our case the best λ is around 230.

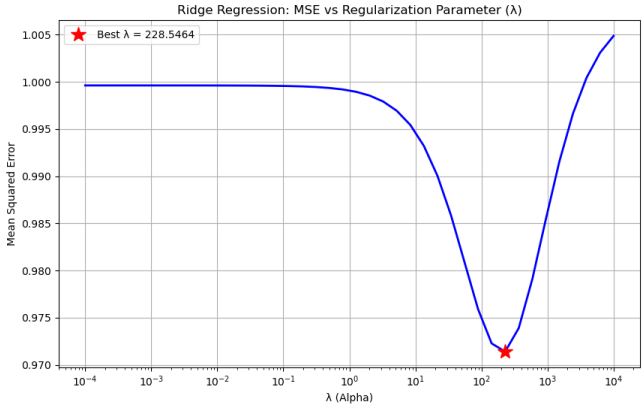


Figure 2: Plot of mean squared error per lambda value

The output of the linear model is calculated by $\sum w_i x_i$ + b, where w_i are the coefficients of the model, x_i are the features on the input vector, and b is the intercept term. Each feature input gets multiplied by its weight on the model and we get the final results. Referring back to Figure 1, our findings align well with the previous data from Report 1, reinforcing the validity of our model.

2 Regression: Model Comparison

For comparison, we evaluate three models: the Ridge model from Section 1, an Artificial Neural Network (ANN), and a baseline model that computes only the average.

The ANN model uses the `MLPRegressor` from `scikit-learn` with the following parameters:

- `hidden_layer_sizes=(param,)`
- `activation='relu'`
- `solver='adam'`
- `learning_rate='adaptive'`
- `learning_rate_init=0.01`
- `max_iter=2000`
- `early_stopping=True`
- `validation_fraction=0.1`
- `n_iter_no_change=10`
- `random_state=42`
- `tol=1e-4`

We selected these parameters for the neural network to ensure stable and efficient learning. The ReLU activation function is widely used for its efficiency in deep networks, helping the model learn complex patterns by adding non-linearity. Adam was chosen as the solver because it combines the benefits of adaptive learning rates and momentum, making it well-suited for noisy gradients and sparse data. The adaptive learning rate allows for flexibility, slowing down as training progresses, which can help in fine-tuning the model's performance. We set learning_rate_init=0.01 to start with a relatively high learning rate, then

adjusted it based on model feedback. Early stopping and a validation fraction of 0.1 prevent overfitting by monitoring the model's performance on the test set, and n_iter_no_change=10 ensures training stops if no improvement is seen. The maximum iterations of 2000 should provide enough epochs for convergence, while random_state=42 ensures reproducibility, and tolerance (tol) of 1e-4 sets a threshold for stopping once the model reaches a satisfactory level of accuracy. We experiment with various numbers of hidden units, using powers of 2 from 1 to 256. Training is performed using a nested cross-validation scheme, with 10 outer folds and 10 inner folds.

Table 2 contains the results of all models throuought the ten folds. On a first glance we can see that the ridge model consistently outperforms the baseline, 7 out of the 10 folds. While the ANN seems to be worse than the baseline. Additionally, we can see that the λ we got on the section 1 returns here on folds 3,5,7,8.

Table 2: Model Comparison across 10 Folds

Fold	Ridge Lambda	ANN Hidden Units	Ridge Error	ANN Error	Baseline Error
1	142.5103	4	0.9044	1.1828	0.9520
2	142.5103	256	1.0260	1.2676	0.9538
3	228.5464	16	1.4132	1.8440	1.4762
4	142.5103	8	0.7596	0.7032	0.8479
5	228.5464	1	0.7427	0.8600	0.7949
6	142.5103	1	0.7442	0.6811	0.8836
7	228.5464	4	1.1918	1.2773	1.2867
8	228.5464	64	1.5407	1.6952	1.5826
9	88.8624	128	0.8617	1.1622	0.7542
10	88.8624	128	0.6186	0.7502	0.5487
Mean	166.1952	61.0000	0.9803	1.1424	1.0081
Std	57.4991	85.3385	0.3091	0.4046	0.3325

Comparing the baseline and the Ridge in 3 we can see that the ridge box has a lower mean and has a lower upper bound, consistent with our observations from table 2. Additionally, Fig. 4 highlights that the ANN is worse, as its errors are more distributed into the right, indicating higher error rates. This also matches our observations for table 2

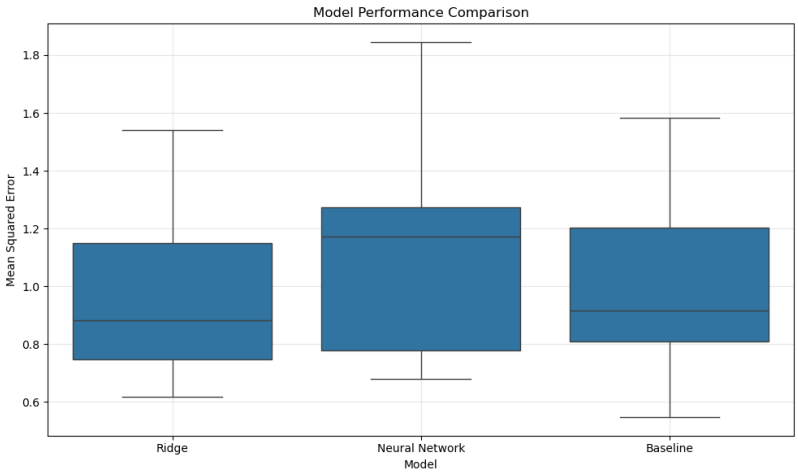


Figure 3: Boxplot of the results

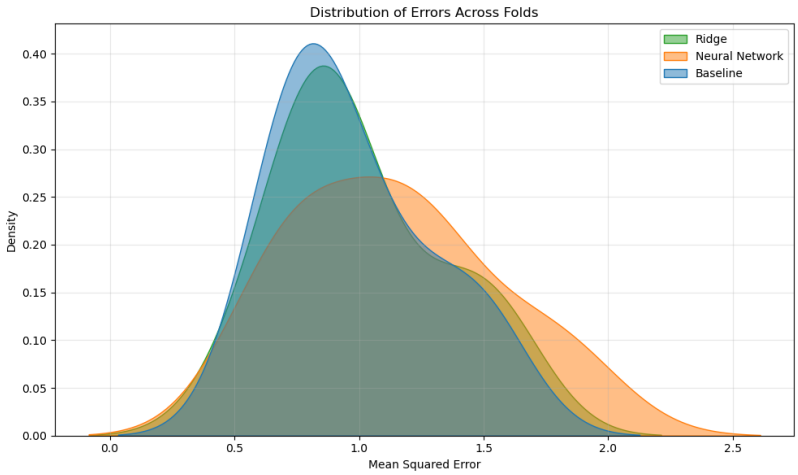


Figure 4: Distribution of errors for each model

Lastly, we chose the paired t-test to conduct a statistical analysis on the three models. Results are displayed on table 3. Confidence intervals were calculated using `t.ppf` function from scipy-stats to get the critical t for the mean difference between the two error sets.

The ridge vs ANN shows that the Ridge performs better(mean difference is negative), the very low p-value suggests that this difference is statistically significant and the performance is not due to chance, and the confidence interval does not include zero, further confirming that the ridge is better for our case.

For the ridge vs baseline we again have a negative mean difference, which could suggest that the ridge performs better. However, a high p-value potentially explains that this difference is not statistically significant, and the confidence intervals very close to zero further support the conclusion that there is no significant performance gains between the two.

Lastly for ANN vs baseline we get a positive mean difference, which could mean that the ANN performs worse than the baseline. Our p-value is above the 5% threshold for this comparison, meaning that the difference may not be meaningful. Finally, the confidence interval is very in favor of the baseline with a large positive number, but as the interval includes zero the performance difference between both models is not conclusive.

Table 3: Paired T-Test Results (Comparing Model Performance)

Model Comparison	Mean Difference	p-value	95% Confidence Interval
Ridge vs ANN	-0.1621	0.0094	(-0.2735, -0.0506)
Ridge vs Baseline	-0.0278	0.3135	(-0.0866, 0.0311)
ANN vs Baseline	0.1343	0.0726	(-0.0151, 0.2837)

3 Classification

For classification, our target is the death_event attribute. It is a binary attribute that represents if a patient dies of heart failure during the follow-up period. So in this case our classification problem is a binary classification problem.

3.1 Implementation

Standardization and principal component selection follow the same patterns as in the regression. Additionally, the three models for comparison also follow similarly to the regression experiment. We are using a logistic regression, an ANN and a baseline model.

The baseline model is implemented using sklearn's DummyClassifier, with selection criteria being the most frequent class on our observations. For the logistic regression we used the sklearn's LogisticRegressor model. Finally, the ANN we implemented using sklearn's MLPClassifier with the same parameters as the MLPRegressor for the regression section.

For parameters we also used exactly the same combination as in the regression experiment. λ are sampled from 40 logarithmically distributed values from 10^{-4} to 10^4, and the hidden parameter count goes from 1 to 256.

3.2 Results

Results are displayed on table 4. By analyzing it superficially we could say that the logistic regression performed the best, as it has the lowest mean. Fig. 5 also supports this claim. It shows our logistic regressor potentially performs better than the competition.

fold	logistic_C	ann_hidden_units	logistic_error	ann_error	baseline_error
1	0.615848	4.000000	0.233333	0.433333	0.400000
2	0.615848	8.000000	0.200000	0.333333	0.433333
3	4.281332	2.000000	0.200000	0.400000	0.400000
4	0.615848	1.000000	0.166667	0.566667	0.433333
5	0.615848	2.000000	0.200000	0.166667	0.166667
6	0.233572	4.000000	0.066667	0.166667	0.300000
7	1.623777	4.000000	0.200000	0.233333	0.266667
8	0.233572	4.000000	0.266667	0.200000	0.233333
9	0.615848	4.000000	0.333333	0.366667	0.400000
10	0.615848	8.000000	0.172414	0.241379	0.172414
Mean	1.006734	4.100000	0.203908	0.310805	0.320575
Std	1.211796	2.330951	0.068985	0.131957	0.105900

Table 4: Two-Level Cross-Validation Results (Classification)

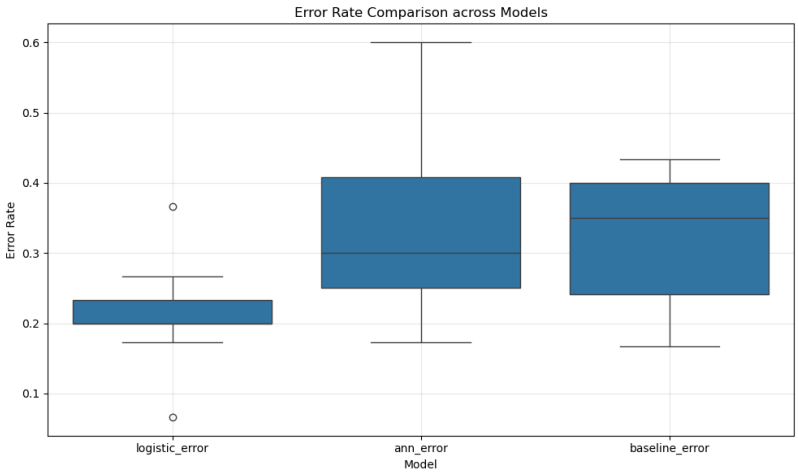


Figure 5: Boxplot for classification models

The Logistic Regression model makes binary predictions by computing a weighted sum of the input features, and passing this result through the logistic sigmoid function to obtain a probability between 0 and 1. If the probability exceeds 0.5, the model predicts the positive class (patient died); otherwise, it predicts the negative class (patient survived).

As for comparison of the features between the regression and classification, we cannot make any conclusions from it, as we had to use different features from our dataset.

3.3 Statistical analysis

The boxplot for classification models which is given above, tell us that Logistic Regression showcases the lowest error rate among all of them, from which we can infer it is the most accurate and consistent model of the three. ANN while it shows a wider spread in error rates is still better than Baseline model. ANN has variable performance across different folds and we can also see that it has some overlap with the Baseline as well. On the other hand, Baseline has a high error rate and narrow spread which shows its poor performance when compared to the other two.

We have also done McNemar's Test in addition to it which gives a statistical comparison of the predictions. Here, the analysis is done for each model pair amounting to three different results and focuses on whether they have different error rates on same data samples. The analysis done gave us the following data:

Comparison	**p-value**	**Confidence Interval**
Logistic Regression vs ANN	1.34×10^{-10}	[0.400, 0.760]
Logistic Regression vs Baseline	0.00013	[0.214, 0.650]
ANN vs Baseline	0.00187	[-0.482, -0.115]

Table 5: McNemar's Test Results with Confidence Intervals

The p-values in the above table indicate whether the error patterns are significantly different or not. We see that Logistic vs ANN has a very low p-value implying highly significant difference in error patterns. It also tell us that Logistic Regression outperforms the ANN. For the same comparison, the confidence interval indicates that Logistic Regression has has a large statistical advantage over ANN. Similarly, for Logistic Regression vs Baseline, the p-value while is higher the former is still less than <0.05 which indicates this model too has a statistically significant difference in model's error patterns. The confidence interval supports the same claim indicating the Logistic Regression's superior performance over the others. But when

comparing the ANN vs Baseline we see a positive p-value which implies a statistically significant difference with ANN performing better than Baseline and the confidence interval support the same.

To summarize, we can say that both the boxplot and McNemar's test results are consistent and they show that Logistic Regression outperforms ANN which in turn outperforms Baseline. McNemar's test adds the statistical significance to these results removing any claims of them being due to chance or bias. Confidence intervals in McNemar's test give an estimated performance difference between the models wherein Logistic Regression reigns supreme outperforming ANN by a margin.

4 Discussion

Our statistical analysis has revealed that relying solely on visualizations and error metrics can be misleading. For example, the ANN, though capable of capturing complex non-linear relationships, did not exhibit a statistically significant improvement over the baseline model in our regression task. In contrast, simpler models like linear and logistic regression, while less flexible, provided more accurate predictions for our dataset.

The previous study on this dataset [1] showed that the most important features for classification are ejection fraction and serum creatinine. Using all features in the dataset their best model(logistic regresson) got an accuracy of 0.833, and using only the two important variables plus Follow-up time they got an accuracy of 0.838 average throughout 100 executions.

We did not run our experiment 100 times, but our best classification model got close to 0.94 accuracy. The main differences from [1] are that we performed PCA to reduce the dataset's dimensionality, and that we also executed 2-level, 10-fold cross validation for the model.

5 Exam problems

Question 1 - Spring 2019 question 13

By sliding the classification threshold on the $\hat{y}$ line and looking for points that satisfy our ROC curve we get that option **C** is correct. The ROC starts at (0,0), meaning that all of the values are considered negative. If we change the classification threshold to before the first x, we get $TP = 1, FN = 3, TPR = 0.25$. Sliding through the two circles we get $TN = 2, FP = 2, FPR = 0.5$. Sliding through the next two x, we get $TP = 3, FN = 1, TPR = 0.75$. This lines perfectly with our ROC.

Question 2 - Spring 2019 question 15

Using the classError definition on 9.4, our problem has the following:

$$I_e = 1 - \max\left(\frac{n_{y=1}}{N}, \frac{n_{y=2}}{N}, \frac{n_{y=3}}{N}, \frac{n_{y=4}}{N}\right)$$

This gets us an impurity of $I_e \approx 0.72592$ before split and $I_e \approx 0.72388$ after split. Our delta then is option **D**.

Question 3 - Spring 2019 question 18

To determine the total number of parameters in the neural network, we calculate the parameters for both the input-to-hidden and hidden-to-output layers.

1. **Input-to-Hidden Layer**: We have 7 input features and 10 hidden units. Each input feature has a weight for each hidden unit, resulting in $7 \times 10 = 70$ weights. Additionally, each hidden unit has a bias, adding 10 more parameters:

$$70 + 10 = 80$$

2. **Hidden-to-Output Layer**: With 10 hidden units and 4 output units, we have $10 \times 4 = 40$ weights. Each output unit also has a bias, adding 4 more parameters:

$$40 + 4 = 44$$

Adding these together, the total number of parameters is:

$$80 + 44 = 124$$

Thus, the correct answer is **(A)** Network contains 124 parameters.

Question 4 - Spring 2019 question 20

For question 4, we can go by process of elimination. On our tree-graph we can see that the classification of congestion level 4 goes through node A and C. Additionally, on the classification boundary figure we can see that congestion level 4 is not dependent on b_2. So we can exclude any options that use b_2 or A or C. Leaving us with answer **D**.

Question 5 - Spring 2019 question 22

To determine the total time required to generate Table 5, we account for training and testing times over multiple cross-validation folds for both models (Neural Network and Logistic Regression). We use a two-level cross-validation approach with $K_1 = 5$ outer folds and $K_2 = 4$ inner folds.

- **Neural Network:**
  - Tested values for n_h: $\{1, 2, 3, 4, 5\}$.
  - Training time per model: 20 ms.
  - Testing time per model: 5 ms.
  - Inner fold time per n_h: $4 \times (20 + 5) = 100$ ms.
  - Total inner fold time for all n_h values in one outer fold: $5 \times 100 = 500$ ms.
  - Total inner fold time across all outer folds: $5 \times 500 = 2500$ ms.
  - Outer fold training and testing time: $5 \times (20 + 5) = 125$ ms.
  - **Total time for Neural Network:** $2500 + 125 = 2625$ ms.
- **Logistic Regression:**
  - Tested values for λ: $\{0.01, 0.06, 0.32, 1.78, 10\}$.
  - Training time per model: 8 ms.
  - Testing time per model: 1 ms.
  - Inner fold time per λ: $4 \times (8 + 1) = 36$ ms.
  - Total inner fold time for all λ values in one outer fold: $5 \times 36 = 180$ ms.
  - Total inner fold time across all outer folds: $5 \times 180 = 900$ ms.
  - Outer fold training and testing time: $5 \times (8 + 1) = 45$ ms.
  - **Total time for Logistic Regression:** $900 + 45 = 945$ ms.

Overall Total Time:

$$2625 + 945 = 3570 \text{ ms}$$

Thus, the approximate time taken to compose the table is **C) 3570.0 ms**.

Question 6 - Spring 2019 question 26

To determine which observation belongs to class $y = 4$, we need to analyze which point produces the highest probability $P(y = 4|\hat{y})$ under the given model. The probability for class 4 is given by:

$$P(y = 4|\hat{y}) = \frac{1}{1 + \sum_{k'=1}^{3} e^{\hat{y}_{k'}}}$$

This probability will be highest when all $\hat{y}_k$ values are small (ideally negative), as this minimizes the denominator.

For each observation $\mathbf{b} = [b_1, b_2]^T$, we calculate:

$$\hat{y}_k = [1 \quad b_1 \quad b_2]\mathbf{w}_k \quad \text{for } k = 1,2,3$$

Let's examine observation B: $\mathbf{b} = [-0.6, -1.6]^T$

$$\begin{aligned}
\hat{y}_1 &= 1(1.2) + (-0.6)(-2.1) + (-1.6)(3.2) \\
&= 1.2 + 1.26 - 5.12 = -2.66 \\
\hat{y}_2 &= 1(1.2) + (-0.6)(-1.7) + (-1.6)(2.9) \\
&= 1.2 + 1.02 - 4.64 = -2.42 \\
\hat{y}_3 &= 1(1.3) + (-0.6)(-1.1) + (-1.6)(2.2) \\
&= 1.3 + 0.66 - 3.52 = -1.56
\end{aligned}$$

Since all $\hat{y}_k$ values are negative:

$$e^{-2.66} + e^{-2.42} + e^{-1.56} \approx 0.07 + 0.09 + 0.21 = 0.37$$

Therefore:

$$P(y = 4|\hat{y}) = \frac{1}{1 + 0.37} \approx 0.73$$

For the other observations (A, C, D), their coordinates lead to large positive values of $\hat{y}_k$, resulting in very large values of $e^{\hat{y}_k}$ and consequently very small probabilities for class 4.

Therefore, observation B is assigned to class $y = 4$ as it yields the highest probability for this class.

References

[1] Davide Chicco and Giuseppe Jurman. Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone. *BMC medical informatics and decision making*, 20:1–16, 2020.