Assignment 3: Probabilistic ML

Marcus Hansen

November 2024

Exercise 1 (20%)

Suppose we collect data from a group of students in a Machine learning class with variables $x_1 =$ hours studied, $x_2 =$ grade point average, and $y =$ a binary output if that student received grade 5 ($y = 1$) or not ($y = 0$). We learn a logistic regression model

$$p(y = 1 \mid x) = \frac{e^{\beta_0+\beta_1 x_1+\beta_2 x_2}}{1 + e^{\beta_0+\beta_1 x_1+\beta_2 x_2}} \tag{1}$$

with parameters $\hat{\beta}_0 = -6$, $\hat{\beta}_1 = 0.05$, $\hat{\beta}_2 = 1$.

Q1: Estimate the probability according to the logistic regression model that a student who studies for 40 h and has the grade point average of 3.5 gets a 5 in the Machine learning class.

To estimate the probability that a student who studies 40 hours per week and has a grade point average of 3.5 will get a 5 in the machine learning class, we need to find $p(y = 1 \mid x, \hat{\beta})$ using our model parameters: $\hat{\beta}_0 = -6$, $\hat{\beta}_1 = 0.05$, $\hat{\beta}_2 = 1$, with $x_1 = 40$ and $x_2 = 3.5$.

$$p(y = 1|x, \hat{\beta}) = \frac{e^{\beta_0+\beta_1 x_1+\beta_2 x_2}}{1 + e^{\beta_0+\beta_1 x_1+\beta_2 x_2}} = \frac{e^{-6+0.05*40+1*3.5}}{1 + e^{-6+0.05*40+1*3.5}} \approx 0.38$$

The estimated probability that a student who studies 40 hours per week and has a grade point average of 3.5 will receive a 5 in the machine learning class, according to the logistic regression model, is approximately 38%.

Exercise 2 (20%)

Q1: Let $\sigma(a) = \frac{1}{1+e^{-a}}$ be the sigmoid function. Show that

$$\frac{d\sigma(a)}{da} = \sigma(a)(1 - \sigma(a)). \tag{2}$$

Using the definition of $\sigma(a)$ into the right side of equation 2 we get:

$$\begin{aligned}
\sigma(a)(1-\sigma(a)) &= \frac{1}{1+e^{-a}}\left(1-\frac{1}{1+e^{-a}}\right) \\
&= \frac{1}{1+e^{-a}}\left(\frac{\frac{1}{1+e^{-a}}}{\frac{1}{1+e^{-a}}}-\frac{1}{1+e^{-a}}\right) \\
&= \frac{1}{1+e^{-a}}\left(\frac{1+e^{-a}-1}{1+e^{-a}}\right) \\
&= \frac{e^{-a}}{(1+e^{-a})^2}
\end{aligned} \tag{3}$$

Now taking the derivative of $\sigma(a)$ we have:

$$\begin{aligned}
\frac{d\sigma(a)}{da} &= \frac{d}{da}\left[\frac{1}{1+e^{-a}}\right] \\
&= \frac{d}{da}\left[(1+e^{-a})^{-1}\right] \\
&= (-1)(1+e^{-a})^{-2}(-e^{-a}) \\
&= \frac{e^{-a}}{(1+e^{-a})^2}
\end{aligned} \tag{4}$$

Since equation 3 equals equation 4 we have shown 2.

Q2: Using the previous result and the chain rule of calculus, derive an expression for the gradient of the log likelihood for logistic regression.

For a binary classification problem solved by logistic regression we can use a similar expression to the one from Exercise 1, equation 1. We define the data as in the lectures $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$. For one sample in $\mathcal{D}$ we have the likelihood of observing that datapoint as:

$$p(\boldsymbol{y}_i \mid \boldsymbol{x}_i, \beta) = \sigma(a(\boldsymbol{x}_i, \beta)) = \frac{1}{1+e^{-(\beta^T \boldsymbol{x}_i)}} \tag{5}$$

With $\beta = [\beta_0, \beta_1, ..., \beta_M]^T$ and $\boldsymbol{x}_i = [1, x_{i1}, ..., x_{iM}]^T$ we have $a(\boldsymbol{x}_i, \beta) = \beta^T \boldsymbol{x}_i$ (with $M = 2$ we get the model from exercise 1). We then make the assumption that the samples are i.i.d. and can write the likelihood function over $\mathcal{D}$ as:

$$p(\boldsymbol{y}_\mathcal{D} \mid \boldsymbol{x}_\mathcal{D}, \beta) = \prod_{i=1}^{N} p(\boldsymbol{y}_i \mid \boldsymbol{x}_i, \beta) \tag{6}$$

Setting $\boldsymbol{y}_i = 1$ if $\boldsymbol{x}_i \in C_1$ and $\boldsymbol{y}_i = 0$ if $\boldsymbol{x}_i \in C_2$ we note that:

$$
\begin{aligned}
&p(\boldsymbol{y}_i = 1 \mid \boldsymbol{x}_i, \beta) = p(\boldsymbol{y}_i \mid \boldsymbol{x}_i, \beta)^{y_i} \\
&p(\boldsymbol{y}_i = 0 \mid \boldsymbol{x}_i, \beta) = p(\boldsymbol{y}_i \mid \boldsymbol{x}_i, \beta)^{(1-y_i)} \\
&\qquad \Longrightarrow \qquad (7) \\
&p(\boldsymbol{y}_i \mid \boldsymbol{x}_i, \beta) = p(\boldsymbol{y}_i \mid \boldsymbol{x}_i, \beta)^{y_i} p(\boldsymbol{y}_i \mid \boldsymbol{x}_i, \beta)^{(1-y_i)} \\
&= \sigma(a(\boldsymbol{x}_i, \beta))^{y_i} \sigma(a(\boldsymbol{x}_i, \beta))^{(1-y_i)}
\end{aligned}
$$

which we can use to rewrite we can rewrite the likelihood, expression 6 as:

$$
p(\boldsymbol{y}_{\mathcal{D}} \mid \boldsymbol{x}_{\mathcal{D}}, \beta) = \prod_{i=1}^{N} \sigma(a(\boldsymbol{x}_i, \beta))^{y_i} \sigma(a(\boldsymbol{x}_i, \beta))^{(1-y_i)}
$$

Which we take the log of yielding:

$$
\log\left(p(\boldsymbol{y}_{\mathcal{D}} \mid \boldsymbol{x}_{\mathcal{D}}, \beta)\right) = \sum_{i=1}^{N} y_i \log\left(\sigma(a(\boldsymbol{x}_i, \beta))\right) + (1 - y_i) \log\left(1 - \sigma(a(\boldsymbol{x}_i, \beta))\right) \tag{8}
$$

and then the gradient of, yielding:

$$
\begin{aligned}
\nabla_\beta \log\left(p(\boldsymbol{y}_{\mathcal{D}} \mid \boldsymbol{x}_{\mathcal{D}}, \beta)\right) &= \sum_{i=1}^{N} y_i \nabla_\beta \log\left(\sigma(a(\boldsymbol{x}_i, \beta))\right) + (1 - y_i) \nabla_\beta \log\left(1 - \sigma(a(\boldsymbol{x}_i, \beta))\right) \\
&= \sum_{i=1}^{N} y_i \frac{1}{\sigma(a(\boldsymbol{x}_i, \beta))} \frac{\partial}{\partial \boldsymbol{\beta}} \left(\sigma(a(\boldsymbol{x}_i, \beta))\right) + (1 - y_i) \frac{1}{1 - \sigma(a(\boldsymbol{x}_i, \beta))} \frac{\partial}{\partial \boldsymbol{\beta}} \left(1 - \sigma(a(\boldsymbol{x}_i, \beta))\right) \\
&= \sum_{i=1}^{N} y_i \frac{1}{\sigma_i} \frac{d\sigma}{da} \frac{\partial a}{\partial \boldsymbol{\beta}} + (1 - y_i) \frac{-1}{(1 - \sigma_i)} \frac{d\sigma}{da} \frac{\partial a}{\partial \boldsymbol{\beta}}, \quad \sigma_i \equiv \sigma(a(\boldsymbol{x}_i, \beta))
\end{aligned}
$$

where we have $\dfrac{d\sigma}{da}$ from equation 2 in exercise 2, yielding the gradient of the logarithm of the likelihood for the general logistic regression case (with σ having properties as in equation 2) as:

$$
\begin{aligned}
\nabla_\beta \log\left(p(\boldsymbol{y}_{\mathcal{D}} \mid \boldsymbol{x}_{\mathcal{D}}, \beta)\right) &= \sum_{i=1}^{N} y_i \frac{1}{\sigma_i} \sigma_i \left(1 - \sigma_i\right) \frac{\partial a}{\partial \boldsymbol{\beta}} - \left(1 - y_i\right) \frac{1}{(1 - \sigma_i)} \sigma_i \left(1 - \sigma_i\right) \frac{\partial a}{\partial \boldsymbol{\beta}} \\
&= \sum_{i=1}^{N} y_i \left(1 - \sigma_i\right) \frac{\partial a}{\partial \boldsymbol{\beta}} - \left(1 - y_i\right) \sigma_i \frac{\partial a}{\partial \boldsymbol{\beta}} \\
&= \sum_{i=1}^{N} \left(y_i - \sigma_i\right) \begin{bmatrix} 1 \\ x_{i1} \\ \vdots \\ x_{iM} \end{bmatrix} \qquad (9)
\end{aligned}
$$

since

$$\frac{\partial a}{\partial \boldsymbol{\beta}} = \frac{\partial}{\partial \boldsymbol{\beta}} \left[\boldsymbol{\beta}^T \boldsymbol{x}_i \right] = \begin{bmatrix} 1 \\ x_{i1} \\ \vdots \\ x_{iM} \end{bmatrix}$$

Exercise 3 (20%)

In this assignment, you will explore generative classification using Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA). Additionally, you will investigate the connection between LDA and dimensionality reduction. You will derive theoretical results, implement these models in Python, and analyze their performance using the MNIST dataset.

Q1 Maximum Likelihood for Class Probabilities

Show that the maximum likelihood estimate for the class probabilities π is given by:

$$\pi_c = \frac{N_c}{N},$$

where N_c is the number of data points assigned to class c, and N is the total number of data points.

For the binary case (to make math less tedious) we denote our data, $D = \{(x_i, y_i)\}$, where x_i is the input and $y_i = 1$ for class 1, C_1 and $y_i = 0$ for C_2. We also denote $p(C_1) = \pi$ and $p(C_2) = 1 - \pi$, $\theta = \{\pi, \mu_1, \mu_2, \Sigma\}$ and assume $x \mid t = C_k \sim \text{pdf}(\mu_k, \Sigma)$. We have:

$$p(x_i \mid C_1)p(C_1) = \pi \text{pdf}\left(x_i \mid \mu_1, \Sigma\right)$$
$$p(x_i \mid C_2)p(C_2) = (1 - \pi)\text{pdf}\left(x_i \mid \mu_2, \Sigma\right)$$

and using previous results (expression in equation 7) we get the likelihood of observing the data as:

$$p(y_D, \mid x_D, \pi, \mu_1, \mu_2, \Sigma) = \prod_{i=1}^{N} \left(\pi \text{pdf}\left(x_i \mid \mu_1, \Sigma\right)\right)^{y_i} \left((1 - \pi)\text{pdf}\left(x_i \mid \mu_2, \Sigma\right)\right)^{1 - y_i}$$

The maximum likelihood criterion is:

$$\hat{\pi}_{\text{ML}} = \arg\max_{\pi} p(y_D, \mid x_D, \theta) \iff$$
$$\hat{\pi}_{\text{ML}} = \arg\min_{\pi} -\log\left(p\left(y_D, \mid x_D, \theta\right)\right)$$

which we find by setting $\frac{\partial}{\partial \pi}\left(-\log\left(p(y_D, \mid x_D, \theta)\right)\right) = 0$ and solving for π

$$
\begin{aligned}
0 &= \frac{\partial}{\partial \pi}\left(\log\left(p(y_D, \mid x_D, \theta)\right)\right) \iff \\
0 &= \frac{\partial}{\partial \pi}\left(\log\left(\prod_{i=1}^{N}\left(\pi \text{pdf}\left(x_i \mid \mu_1, \Sigma\right)\right)^{y_i}\left((1-\pi)\text{pdf}\left(x_i \mid \mu_2, \Sigma\right)\right)^{1-y_i}\right)\right) \iff \\
0 &= \frac{\partial}{\partial \pi}\left(\sum_{i=1}^{N}\left(y_i \log(\pi) + y_i \log(\text{pdf}\left(x_i \mid \mu_1, \Sigma\right))\right) + (1-y_i)\log(1-\pi) + (1-y_i)\log(\text{pdf}\left(x_i \mid \mu_2, \Sigma\right))\right) \iff \\
0 &= \sum_{i=1}^{N} y_i \frac{\partial}{\partial \pi}(\log(\pi)) + (1-y_i)\frac{\partial}{\partial \pi}(\log(1-\pi)) \iff \\
0 &= \sum_{i=1}^{N} y_i \frac{1}{\pi} + (1-y_i)\frac{1}{1-\pi} \iff \\
\hat{\pi}_{\text{ML}} &= \frac{1}{N}\sum_{i=1}^{N} y_i
\end{aligned}
$$

But since $y_i = 1$ for class 1, setting $\hat{\pi}_{\text{ML}} = \pi_1$ we have that

$$\pi_1 = \frac{1}{N}\sum_{i=1}^{N} y_i = N_1$$

which can be extended to the general case using $N_c =$ "number of samples in class c" and therefore:

$$\pi_c = \frac{N_c}{N}$$

Q2 Class-Conditional Densities with Shared Covariance Matrix (LDA)

Assuming the class-conditional densities follow a Gaussian distribution with a shared covariance matrix:

$$p(x \mid y = c, \theta) = \mathcal{N}(x \mid \mu_c, \Sigma),$$

derive the following maximum likelihood estimates:

(a) The mean for each class c:

$$\mu_c = \frac{1}{N_c}\sum_{n=1}^{N} x^{(n)} \quad \text{where } y^{(n)} = c.$$

With similar assumptions as in Q1 but with the Gaussian distribution instead of arbitrary distribution we get the likelihood of observing the data in class c as:

$$p(y_D = c \mid x_D, \pi, \mu_1, \mu_2, \Sigma) = \prod_{i:y_i=c} \pi_c \, \mathcal{N}\left(x_i \mid \mu_c, \Sigma\right) \tag{10}$$

Where the expression changes with a new product to explicitly show it for c classes, meaning π_c is the prior for class c and μ_c and N_c is defined as above. Again in similar fashion as in Q1 we have the maximum likelihood criterion as:

$$\hat{\mu}_{\mathrm{c}} = \arg\min_{\mu_c} -\log\left(p\left(y_D = c \mid x_D, \theta\right)\right)$$

which is again found by setting the derivative, in this case $\frac{\partial}{\partial \mu_c}\left(-\log\left(p(y_D = c \mid x_D, \theta)\right)\right) = 0$ and solving for μ_c:

$$\begin{aligned}
0 &= \frac{\partial}{\partial \mu_c}\left(-\log\left(p(y_D = c \mid x_D, \theta)\right)\right) \iff \\
0 &= \frac{\partial}{\partial \mu_c}\left(-\log \prod_{i:y_i=c} \pi_c \, \mathcal{N}\left(x_i \mid \mu_c, \Sigma\right)\right) \iff \\
0 &= \frac{\partial}{\partial \mu_c}\left(\sum_{i:y_i=c} \left(\log\left(\pi_c\right) + \log\left(\mathcal{N}\left(x_i \mid \mu_c, \Sigma\right)\right)\right)\right) \iff \\
0 &= \sum_{i:y_i=c} \frac{\partial}{\partial \mu_c}\left[\log(\mathcal{N}\left(x_i \mid \mu_c, \Sigma\right))\right] \iff \\
0 &= \sum_{i:y_i=c} \frac{\partial}{\partial \mu_c}\left[\frac{-1}{2}(x_i - \mu_c)^T \Sigma^{-1}(x_i - \mu_c)\right] \iff \\
0 &= -\Sigma^{-1} \sum_{i:y_i=c} (x_i - \mu_c), \quad \Sigma^{-1} \text{ invertible since covariance matrix} \implies \\
0 &= -\sum_{i:y_i=c} x_i + N_c \, \mu_c \implies \\
\hat{\mu}_c &= \frac{1}{N_c} \sum_{i:y_i=c} x_i
\end{aligned} \tag{11}$$

Which is what we wanted to show, with slightly different notation.

(b) The shared covariance matrix:

$$\Sigma = \sum_{c=1}^{C} \frac{N_c}{N} S_c, \quad \text{where } S_c = \frac{1}{N_c} \sum_{n=1}^{N} (x^{(n)} - \mu_c)(x^{(n)} - \mu_c)^T \quad \text{and } y^{(n)} = c.$$

In (a) we had the likelihood of observing the data of class c, equation 14, but since the covariance matrix is shared across all classes we need our likelihood to include all classes. Assuming that the samples are i.i.d we have the likelihood of observing the data from all classes as the product of the likelihoods of observing the data from each individual class:

$$\begin{aligned}
p(y_D, x_D \mid \pi, \mu_1, \mu_2, \Sigma) &= \prod_{c=1}^{K} p(y_D = c, \, x_D \mid \pi, \mu_1, \mu_2, \Sigma) = \\
&= \prod_{c=1}^{K} \prod_{i:y_i=c} \pi_c \, \mathcal{N}\left(x_i \mid \mu_c, \Sigma\right)
\end{aligned}$$

As previously we find $\hat{\Sigma}_{ML}$ by setting the derivative of the negative log likelihood, in this case $\frac{\partial}{\partial \Sigma}\left(\log\left(p(y_D, x_D \mid \theta)\right)\right) = 0$ and solving for Σ:

$$0 = \frac{\partial}{\partial \Sigma}\left(-\log\left(p(y_D, x_D \mid \theta)\right)\right) \iff$$

$$0 = \frac{\partial}{\partial \Sigma}\left(-\log \prod_{c=1}^{K} \prod_{i:y_i=c} \pi_c \, \mathcal{N}\left(x_i \mid \mu_c, \Sigma\right)\right) \iff$$

$$0 = -\frac{\partial}{\partial \Sigma}\left(\sum_{c=1}^{K} \sum_{i:y_i=c} \frac{-1}{2}\log\left(\det(\Sigma)\right) + \frac{-1}{2}(x_i - \mu_c)^T \Sigma^{-1}(x_i - \mu_c) + \text{const}\right) \iff$$

$$0 = N\frac{\partial}{\partial \Sigma}\left[\log\left(\det(\Sigma)\right)\right] + \sum_{c=1}^{K} \sum_{i:y_i=c} \frac{\partial}{\partial \Sigma}\left[(x_i - \mu_c)^T \Sigma^{-1}(x_i - \mu_c)\right] =$$

$$= \left\{ \frac{\partial}{\partial \Sigma}\left[\log\left(\det(\Sigma)\right)\right] = \Sigma^{-1}, \quad a^T M a = tr(a^T M a) = tr(M a a^T) \text{ and } \frac{\partial}{\partial \Sigma} tr\left(a a^T \Sigma^{-1}\right) = -\Sigma^{-1} a a^T \Sigma^{-1} \right\} =$$

$$0 = N\Sigma^{-1} - \Sigma^{-1} \sum_{c=1}^{K} \sum_{i:y_i=c} \left[(x_i - \mu_c)(x_i - \mu_c)^T\right] \Sigma^{-1} \iff$$

$$\Sigma = \frac{1}{N} \sum_{c=1}^{K} \sum_{i:y_i=c} \left[(x_i - \mu_c)(x_i - \mu_c)^T\right]$$

and with

$$S_c = \frac{1}{N_c} \sum_{i:y_i=c} \left[(x_i - \mu_c)(x_i - \mu_c)^T\right]$$

We have:

$$\Sigma = \sum_{c=1}^{K} \frac{N_c}{N} S_c \tag{12}$$

and hence we have showed (b).

Q3 Class-Conditional Densities with Separate Covariance Matrices (QDA)

Assuming the class-conditional densities follow Gaussian distributions with separate covariance matrices:

$$p(x \mid y = c, \theta) = \mathcal{N}(x \mid \mu_c, \Sigma_c),$$

derive the maximum likelihood estimate for the class-specific covariance matrix:

$$\Sigma_c = \frac{1}{N_c} \sum_{n=1}^{N} (x^{(n)} - \mu_c)(x^{(n)} - \mu_c)^T \quad \text{where } y^{(n)} = c. \tag{13}$$

Similar to Q2 (a) we have the likelihood of observing the data in class c as:

$$p(y_D = c \mid x_D, \pi, \mu_1, \mu_2, \Sigma) = \prod_{i:y_i=c} \pi_c \, \mathcal{N}\left(x_i \mid \mu_c, \Sigma_c\right)$$

again we find $\hat{\Sigma}_c$ that satisfies that maximum likelihood criterion by setting the derivative of the negative log likelihood, $\frac{\partial}{\partial \Sigma_c}\left(-\log\left(p(y_D = c \mid x_D, \theta)\right)\right) = 0$ and solving for Σ_c:

$$
\begin{aligned}
0 &= \frac{\partial}{\partial \Sigma_c}\left(-\log\left(p(y_D = c \mid x_D, \theta)\right)\right) \iff \\
0 &= -\frac{\partial}{\partial \Sigma_c}\left(\log[\prod_{i:y_i=c} \pi_c \, \mathcal{N}\left(x_i \mid \mu_c, \Sigma_c\right)]\right) \iff \\
0 &= -\sum_{i:y_i=c} \frac{\partial}{\partial \Sigma_c}\left[\log(\mathcal{N}\left(x_i \mid \mu_c, \Sigma_c\right))\right] \iff \\
0 &= -\frac{1}{2}\sum_{i:y_i=c} \frac{\partial}{\partial \Sigma_c}\left[\log\left(\det(\Sigma_c)\right) + (x_i - \mu_c)^T \Sigma_c^{-1}(x_i - \mu_c)\right], \quad \text{as previously} \iff \\
0 &= \sum_{i:y_i=c} \Sigma_c^{-1} - \Sigma_c^{-1}(x_i - \mu_c)(x_i - \mu_c)^T \Sigma_c^{-1} \iff \\
0 &= N_c \Sigma_c^{-1} - \Sigma_c^{-1} \sum_{i:y_i=c} \left((x_i - \mu_c)(x_i - \mu_c)^T\right) \Sigma_c^{-1}, \quad \Sigma^{-1} \text{ invertible since covariance matrix} \implies \\
\hat{\Sigma}_c &= \frac{1}{N_c} \sum_{i:y_i=c} (x_i - \mu_c)(x_i - \mu_c)^T \qquad (14)
\end{aligned}
$$

which is equivalent to equation 13 and hence we have shown 13.

Q4 Model Implementation

Implement LDA and QDA fit function to learn

$$\mu_0, \mu_1, \Sigma_0, \Sigma_1, \pi, w_0, w_1$$

See notebook "Q3", maybe rename to "E3" for next year to clarify.

Q5 Performance Comparison

Evaluate the performance of LDA and QDA by calculating accuracy on the test set. Discuss the differences between the decision boundaries of LDA and QDA and their behavior under different class distributions.

Since there was no code for the test set we modified the `generate_data()` function to take the input arguments: *number of training samples, number of test samples* and also the input argument to `make_classification()`, `class_sep` that decides the separation of the clusters.

The different data configurations of small and big training sets and different class separations are presented in table 2 and their respective train and test accuracies for LDA and QDA are shown in table 1.

Looking at the train and test accuracy of configuration 1-3 (see table 2 for configurations) in table 1, where we have many training samples and we increase the separation of the clusters.

Config	LDA Train Acc.	QDA Train Acc.	LDA Test Acc.	QDA Test Acc.
Config 1	0.640	0.871	0.627	0.867
Config 2	0.874	0.923	0.840	0.913
Config 3	0.951	0.963	0.953	0.967
Config 4	0.800	1.000	0.612	0.702
Config 5	0.900	1.000	0.860	0.812
Config 6	1.000	1.000	0.939	0.871

Table 1: Train and test accuracies (acc.) for LDA and QDA across different dataset configurations.

Configuration	Train Samples	Test Samples	Class Separation (1 = default)
Config 1	350	150	0.25
Config 2	350	150	1
Config 3	350	150	1.5
Config 4	10	490	0.25
Config 5	10	490	1
Config 6	10	490	1.5

Table 2: Dataset Configurations: Number of train/test samples and class separation

We note that QDA outperforms LDA for all class separations but that the difference in accuracy becomes smaller as the separation between the clusters increases. This makes sense since the nonlinear decision boundaries makes the model more complex and hence the model performs better when the problem is more complex (smaller class separation). We also note that the variance (difference in performance for train and test dataset) is not that large, which makes sense since the training set is quite large and we have a given distribution, therefore the model to generalize quite well on the test set.

For configuration 4-6, see table 2, we instead have a very small training set, which we train for three different cluster separation. In the accuracies table, 1, we note that QDA gets a perfect accuracy on the training set for all separations, but an accuracy that is far from perfect on the test set. I.e QDA has overfitted the training data and therefore has a very high variance. LDA also overfits the data, but not to the same extent. We also see that the test set accuracy of LDA increases faster than that of QDA as the separation increases and that the LDA has a better test set accuracy for the class separation values 1 and 1.5 on this dataset with a small training set. This indicates that a less complex model is not a bad choice when there are limited training samples.

To make this investigation better we should probably have had a constant and very large test set size, instead of as we did now: fixed the the size of the whole dataset, specified the size of the training set and used remaining samples as test samples. The reason this method was chosen was that this enabled us to get the same clusters for each configuration as the cluster intrinsics changed with a new number of samples even though the same random seed was used.

Exercise 4 (20%)

This assignment focuses on implementing and analyzing logistic regression for binary classification. You will derive key mathematical components (negative log-likelihood, gradient, Hessian) and use the Newton-Raphson algorithm for optimization. The practical tasks involve preprocessing data, implementing logistic regression, and evaluating it on the Breast Cancer Survival dataset.

Q1 Negative Log-Likelihood Derivation

Derive the negative log-likelihood (NLL) for the logistic regression model.

We showed in Exercise 2, Question 2, that the log-likelihood for the logistic regression model can be written as:

$$\log\left(p(\boldsymbol{y}_{\mathcal{D}} \mid \boldsymbol{x}_{\mathcal{D}}, \beta)\right) = \sum_{i=1}^{N} y_i \log\left(\sigma(\beta^T \boldsymbol{x}_i)\right) + (1 - y_i) \log\left(1 - \sigma(\beta^T \boldsymbol{x}_i)\right)$$

To obtain the negative log-likelihood, we simply take the negative of the log-likelihood function:

$$NLL = -\sum_{i=1}^{N} y_i \log\left(\sigma(\beta^T \boldsymbol{x}_i)\right) + (1 - y_i) \log\left(1 - \sigma(\beta^T \boldsymbol{x}_i)\right)$$

Q2 Gradient and Hessian Analysis

a) Derive the gradient of the NLL with respect to the parameter vector β.

Here we will also use the result from Exercise 2, Question 2 where we derived the gradient for the log-likelihood, equation 9 and just add a minus sign in front and simplify notation a bit:

$$\nabla_\beta NLL = \frac{\partial NLL}{\partial \beta} = -\sum_{i=1}^{N} \left[y_i - \sigma(\beta^T \boldsymbol{x}_i)\right] x_i \tag{15}$$

b) Compute the Hessian matrix for the logistic regression model.

The Hessian matrix for the logistic regression model is defined as the second derivative of the negative log-likelihood (NLL) function with respect to the parameter vector β. To obtain the Hessian H_β, we take the derivative of the computed gradient in equation 15. This results in:

$$H_\beta = \frac{\partial \nabla_\beta NLL}{\partial \beta} = \sum_{i=1}^{N} \left[\sigma(\beta^T \boldsymbol{x}_i)(1 - \sigma(\beta^T \boldsymbol{x}_i)) x_i^T x_i\right] \tag{16}$$

c) How can we use the Hessian for optimization using Newton-Raphson algorithm.

The Newton-Raphson algorithm is used to find the roots of the derivative of a differentiable function f and performs the iterations as follows:

$$x_{k+1} = x_k + t = x_k - \frac{f'(x_k)}{f''(x_k)}.$$

In our case, we want to find the roots of the NLL function, which is concave and differentiable. Therefore, we can use the Newton-Raphson algorithm. The second derivative corresponds to the Hessian, equation 16, of the NLL function, and the first derivative corresponds to the gradient, equation 15. The algorithm iteratively updates the parameter vector β and we can write the Newton-Raphson algorithm as follows:

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \mathbf{H}^{-1}\nabla_{\beta}\text{NLL}$$

Q3 Data Exploration and Preprocessing

a) Load the Breast Cancer Survival dataset and explore its characteristics (e.g., summary statistics, missing values, distribution of classes)

We explored the dataset's characteristics by examining the summary, missing values, and class distribution. There were no missing values in the explanatory features we intended to use. A summary of the data is presented in table 3. The dataset is quite unbalanced, with more people

Table 3: Summary Statistics of the Dataset

Statistic	Age	Year	Nodes Detected	Survival Status
Count	306	306	306	306
Mean	52.46	62.85	4.03	1.26
Std	10.80	3.25	7.19	0.44
Min	30.00	58.00	0.00	1.00
25%	44.00	60.00	0.00	1.00
50%	52.00	63.00	1.00	1.00
75%	60.75	65.75	4.00	2.00
Max	83.00	69.00	52.00	2.00

surviving 5 years or longer compared to those who died within 5 years, as shown in figure 1. Survival status was also converted to binary, where 1 indicates that the patient survives 5 years or longer, and 0 indicates that the patient dies within 5 years.

b) Normalize or standardize the numerical features to prepare them for logistic regression.

We decided to standardize the numerical features x_i as:

$$x_{i,\text{standardized}} = \frac{x_i - \mu_i}{\sigma_i}$$

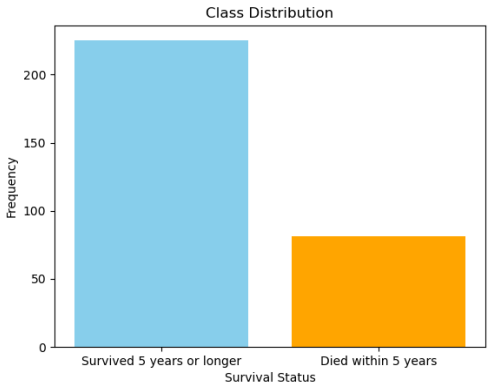


Figure 1: Class Distribution

Q4 Logistic Regression Implementation

a) Implement the logistic function $\sigma(x)$ in Python, b) Write a function to compute the NLL for a given dataset and parameter vector β.

See the "Q4&Q5" Jupyter Notebook!

c) Implement gradient computation and verify it numerically (e.g., using finite differences).

See the "Q4&Q5" Jupyter Notebook for the implementation.

To verify it numerically we used the finite differences and implemented it as:

$$\frac{\partial NLL}{\partial \beta_i} \approx \frac{NLL(\beta_i + \epsilon, X, y) - NLL(\beta_i - \epsilon, X, y)}{2\epsilon}$$

From our implementation of the gradient, we obtained results very similar to those obtained using finite differences:

- Our Gradient Implementation: [9.17329379 -0.64370432 38.71362156]
- Finite Differences: [9.17329379 -0.64370433 38.71362158]

Q5 Logistic Regression Implementation

a) Write a Python function to optimize the NLL using the Newton-Raphson method.

See the Q4&5 Jupyter Notebook!

b) Apply this function to the Breast Cancer Survival dataset to find the MLE estimates for β.

The implemented function was applied to the Breast Cancer Survival dataset to find the MLE estimates for β. The coefficients we found were:

$$\hat{\beta}_{\text{ML}} = [\beta_0, \beta_1, \beta_2, \beta_3]^T = [1.1389, -0.3578, 0.1317, -0.7380]$$

Q6 Model Evaluation

a) Compute classification accuracy, precision, recall, and F1-score for the logistic regression model. Use a train-test split or cross-validation for evaluation.

To better understand the results, we will start by defining which labels are classified as positive and negative. **Positive** will indicate that the patient survives 5 years or longer, and **Negative** will indicate that the patient dies within 5 years.

We used a train-test split for the evaluation with a test size of 0.2, equaling 245 training samples and 61 test samples. The evaluation metrics on the test set are shown below:

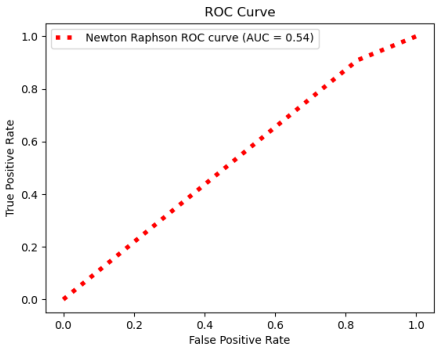
- Newton Raphson Accuracy: 0.69
- Newton Raphson Precision: 0.73
- Newton Raphson Recall: 0.91
- Newton Raphson F1 Score: 0.81

The model achieved an accuracy of 69%, meaning it correctly predicted outcomes 69% of the time. However, as shown in Figure 1, the dataset is unbalanced, and accuracy may not be a suitable metric. Therefore, we also consider precision, recall, and the F1-score.

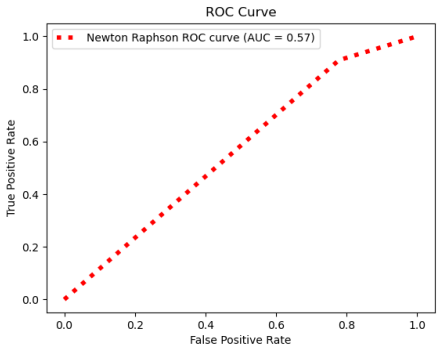
The precision on the test set was 73%, indicating that 73% of the patients was predicted to live 5 years or longer actually did. The remaining 27% represent false positives, indicating patients predicted to survive 5 years or longer who did not. A recall of 91% indicates that the model identified 91% of patients who lived 5 years or longer as positive cases. The remaining 9% represent false negatives, meaning patients who lived 5 years or longer but were predicted not to.

In our case, we prioritize having a higher precision because we want to avoid telling someone they will live 5 or more years, only for them to die sooner. It would be preferable to classify the cancer as being in a more severe stage and make further tests and investigations. The F1-score is the harmonic mean of precision and recall. A score of 81% indicates that both recall and precision are relatively high, suggesting that the overall performance of the model is quite good. However, as mentioned earlier, we would prefer achieving higher precision than recall.

To improve precision, we adjusted the threshold for logistic regression. By increasing the threshold from 0.5 to 0.55, we raised precision to 74% while maintaining recall at 91%. However, increasing the threshold further resulted in a precision of 75%, but recall decreased much more. We can also see that the AUC increased when changing the threshold to 0.57 from 0.55, see figure 2.



(a) ROC and AUC with a threshold for logistic regression at 0.5.



(b) ROC and AUC with a threshold for logistic regression at 0.55.

Figure 2: ROC and AUC

Q7 Feature Importance After fitting the model, analyze the learned coefficients β. Discuss which features contribute most to predicting survival status and why.

To analyze the learned coefficients and determine which features contribute the most to the predicted outcome, we can examine the coefficients.

$$\hat{\beta} = [\beta_0, \beta_1, \beta_2, \beta_3]^T = [1.1389, -0.3578, 0.1317, -0.7380]$$

Here, β_3 has the largest absolute value and is negative, indicating that it has the most significant impact on the prediction of survival status. Its negative sign suggests that higher values of this feature increase the likelihood of predicting the negative class, a higher risk of dying within 5 years. This coefficient corresponds to the third attribute, Positive Axillary Nodes, representing the number of lymph nodes affected by cancer. This is reasonable, as the spread of cancer to lymph nodes is a critical factor in prognosis.

Similarly, β_1 is also negative and relatively large in magnitude, implying that the first attribute, Age, negatively impacts survival. Older individuals are more likely to have worse outcomes. In contrast, β_2 has the smallest magnitude among the coefficients, suggesting that the year of surgery has a smaller impact on the survival prediction.

Q8 Comparison with Library Implementation

a) Use a standard library (e.g., scikit-learn) to fit a logistic regression model to the dataset.

See "Q4&5" Jupyter notebook!

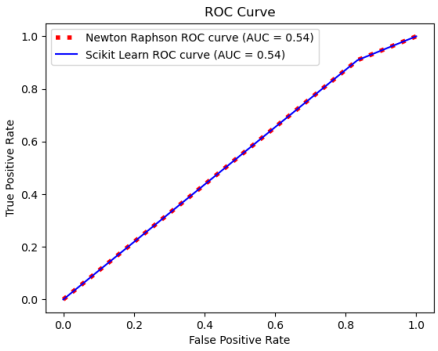
b) Compare the coefficients, accuracy, and other metrics with your implementation. All three models performed similarly, but using a threshold of 0.55 with the logistic regression

model yielded slightly better results across the metrics, see table 4. The coefficients of the scikit-learn model have a bit smaller magnitudes compared to the other models. The Newton-Raphson model with a threshold of 0.5 produced the exact same metrics as the scikit-learn model, resulting in an identical ROC curve, see figure 3. We can also observe that our model, using a threshold

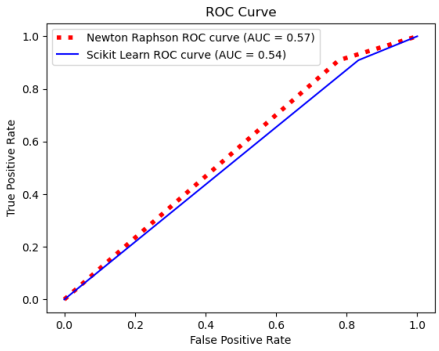
Table 4: Comparison of Logistic Regression Models

Metric	**Newton-Raphson (0.5)**	**Newton-Raphson (0.55)**	**Scikit-Learn**
Accuracy	0.69	0.71	0.69
Precision	0.73	0.74	0.73
Recall	0.91	0.91	0.91
F1 Score	0.81	0.82	0.81
Intercept	1.1389	1.1389	1.1354
Beta	$[-0.358, 0.132, -0.738]$	$[-0.358, 0.132, -0.738]$	$[-0.346, 0.126, -0.715]$

of 0.55, achieved a higher AUC compared to both the scikit-learn model and the model with a threshold of 0.5, see figure 3.



(a) ROC and AUC with our model (threshold at 0.50), and scikit-learn model.



(b) ROC and AUC with our model (threshold at 0.55), and scikit-learn model.

Figure 3: ROC and AUC

Extra Credit (Optional) Implement L2-regularized logistic regression using the Newton-Raphson method. Compare the results with the unregularized version.

The L2-regularized logistic regression was implemented as described in the lecture notes. In table 5, we can see that the metrics are the same, but the magnitudes of our weights are much smaller. This is reasonable because, with L2 regularization, we penalize large weights, as they tend to contribute to overfitting.

Table 5: Comparison of Logistic Regression Models with and without L2-Regularization

Metric	Newton-Raphson (0.55)	Newton-Raphson with L2-reg (0.55)
Accuracy	0.71	0.71
Precision	0.74	0.74
Recall	0.91	0.91
F1 Score	0.82	0.82
Intercept	1.1389	0.9090
Beta Coefficients	$[-0.358, 0.132, -0.738]$	$[-0.249, 0.088, -0.560]$

Exercise 5 (20%)

Q1

Given:

- **Likelihood:**

$$p(y_i \mid x_i, \beta) = \sigma(x_i^T \beta)^{y_i} [1 - \sigma(x_i^T \beta)]^{1-y_i},$$

where the logistic sigmoid function is defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

- **Prior:**

$$\begin{aligned} p(\beta) =& \mathcal{N}(\beta \mid 0, I), \\ =& \frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2}\beta^\top \beta\right). \end{aligned}$$

where $\mathcal{N}(\beta \mid \mu, \Sigma)$ denotes a Gaussian distribution with mean μ and covariance Σ, and I is the identity matrix.

The posterior distribution of β given the data y and X is proportional to the product of the likelihood and the prior:

$$p(\beta \mid y, X) \propto p(y \mid X, \beta)\, p(\beta).$$

We assume that the data are independent given β, the likelihood over all data points is therfore:

$$p(y \mid X, \beta) = \prod_{i=1}^{n} p(y_i \mid x_i, \beta).$$

Thus posterior is:

$$p(\beta \mid y, X) \propto \left[\prod_{i=1}^{n} \sigma(x_i^\top \beta)^{y_i}[1-\sigma(x_i^\top \beta)]^{1-y_i}\right] \times \exp\left(-\frac{1}{2}\beta^\top \beta\right).$$

Taking the logarithm of the posterior:

$$\ln p(\beta \mid y, X) = \sum_{i=1}^{n} \left[y_i \ln \sigma(x_i^\top \beta) + (1-y_i)\ln(1-\sigma(x_i^\top \beta))\right] - \frac{1}{2}\beta^\top \beta + C,$$

where C is a constant independent of β.

To find the MAP estimate β_{MAP}, we maximize the log-posterior:

$$\beta_{\text{MAP}} = \arg\max_{\beta} \ln p(\beta \mid y, X).$$

This is equivalent to minimizing the negative log-posterior:

$$\beta_{\text{MAP}} = \arg\min_{\beta} \left(-\ln p(\beta \mid y, X)\right).$$

Let $\ell(\beta) = -\ln p(\beta \mid y, X)$. Then:

$$\ell(\beta) = -\sum_{i=1}^{n} \left[y_i \ln \sigma(x_i^\top \beta) + (1-y_i)\ln(1-\sigma(x_i^\top \beta))\right] + \frac{1}{2}\beta^\top \beta + C'.$$

Laplace Approximation of posterior

The Laplace approximation is based on a second-order Taylor expansion of the log posterior around its mode, $\boldsymbol{\beta}_{\text{MAP}}$, found with Newton-Raphson as in exercise 4 question 5. Around $\boldsymbol{\beta}_{\text{MAP}}$, we expand $\ell(\boldsymbol{\beta})$ as:

$$\ell(\boldsymbol{\beta}) = \ell(\boldsymbol{\beta}_{\text{MAP}}) + (\boldsymbol{\beta} - \boldsymbol{\beta}_{\text{MAP}})^\top \nabla \ell(\boldsymbol{\beta}_{\text{MAP}}) + \frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{\beta}_{\text{MAP}})^\top \nabla\nabla \ell(\boldsymbol{\beta}_{\text{MAP}})(\boldsymbol{\beta} - \boldsymbol{\beta}_{\text{MAP}}) + Constant,$$

where:

- $\nabla \ell(\boldsymbol{\beta}_{\text{MAP}})$ is the gradient of the log posterior at $\boldsymbol{\beta}_{\text{MAP}}$,
- $\nabla\nabla \ell(\boldsymbol{\beta}_{\text{MAP}}) = \mathbf{H}$ is the **Hessian matrix**, the second derivative of $\ell(\boldsymbol{\beta})$, evaluated at $\boldsymbol{\beta}_{\text{MAP}}$.

At the mode $\boldsymbol{\beta}_{\text{MAP}}$, $\nabla \ell(\boldsymbol{\beta}_{\text{MAP}}) = 0$, so the expansion simplifies to:

$$\ell(\boldsymbol{\beta}) = \ell(\boldsymbol{\beta}_{\text{MAP}}) - \frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{\beta}_{\text{MAP}})^\top (\mathbf{H})(\boldsymbol{\beta} - \boldsymbol{\beta}_{\text{MAP}}) + Constant.$$

Get the posterior around β_{MAP} by taking the exponent becouse $\ell(\beta) = -\ln p(\beta \mid y, X) \Rightarrow p(\beta \mid y, X)|_{\beta=\beta_{MAP}} = -e^{\ell(\boldsymbol{\beta})} = f(z)$. Where $Z = e^{-Constant}$ is a normalization constant

$$p(\boldsymbol{\beta}|\mathbf{X}, \mathbf{y}) = \frac{1}{Z} * f(z) \exp\left(-\frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{\beta}_{\text{MAP}})^\top (\boldsymbol{H})(\boldsymbol{\beta} - \boldsymbol{\beta}_{\text{MAP}})\right).$$

To normalize, the integral of the Gaussian approximation is computed:

$$Z = \int f(z)\, d\boldsymbol{\beta} = f(z)\sqrt{\frac{(2\pi)^d}{|H|}},$$

and the normalized posterior around β_{MAP} becomes:

$$p(\boldsymbol{\beta}|\mathbf{X}, \mathbf{y})|_{\beta=\beta_{MAP}} = \sqrt{\frac{|H|}{(2\pi)^d}} e^{-\frac{1}{2}(\boldsymbol{\beta}-\boldsymbol{\beta}_{\text{MAP}})^\top H(\boldsymbol{\beta}-\boldsymbol{\beta}_{\text{MAP}})}.$$

Which by inspection results in the Laplace approximation approximates the posterior $p(\beta \mid y, X)$ with a Gaussian distribution centered at β_{MAP} with covariance H^{-1}:

$$p(\beta \mid y, X) \approx \mathcal{N}(\beta \mid \beta_{\text{MAP}}, H^{-1}).$$

Compute gradient

To find the mode of the posterior (the MAP estimate), we set the gradient of the negative log-posterior to zero:

$$\nabla \ell(\boldsymbol{\beta}) = \mathbf{0}.$$

Compute the gradient and using that $\sigma'(z) = \sigma(z)\,(1 - \sigma(z))$

$$\begin{aligned}
\nabla \ell(\boldsymbol{\beta}) &= -\sum_{i=1}^{N} \left[y_i \frac{\sigma'\left(\boldsymbol{x}_i^\top \boldsymbol{\beta}\right)}{\sigma\left(\boldsymbol{x}_i^\top \boldsymbol{\beta}\right)} \boldsymbol{x}_i + (1 - y_i) \frac{-\sigma'\left(\boldsymbol{x}_i^\top \boldsymbol{\beta}\right)}{1 - \sigma\left(\boldsymbol{x}_i^\top \boldsymbol{\beta}\right)} \boldsymbol{x}_i \right] + \boldsymbol{\beta} \\
&= -\sum_{i=1}^{N} \left[y_i \left(1 - \sigma\left(\boldsymbol{x}_i^\top \boldsymbol{\beta}\right)\right) - (1 - y_i)\sigma\left(\boldsymbol{x}_i^\top \boldsymbol{\beta}\right) \right] \boldsymbol{x}_i + \boldsymbol{\beta},
\end{aligned}$$

Simplifying the gradient:

$$\begin{aligned}\nabla \ell(\boldsymbol{\beta}) &= -\sum_{i=1}^{N}\left[y_i\left(1-\sigma\left(\boldsymbol{x}_i^\top \boldsymbol{\beta}\right)\right)-(1-y_i)\sigma\left(\boldsymbol{x}_i^\top \boldsymbol{\beta}\right)\right]\boldsymbol{x}_i+\boldsymbol{\beta} \\ &= -\sum_{i=1}^{N}\left[y_i-y_i\sigma\left(\boldsymbol{x}_i^\top \boldsymbol{\beta}\right)-\sigma\left(\boldsymbol{x}_i^\top \boldsymbol{\beta}\right)+y_i\sigma\left(\boldsymbol{x}_i^\top \boldsymbol{\beta}\right)\right]\boldsymbol{x}_i+\boldsymbol{\beta} \\ &= -\sum_{i=1}^{N}\left[y_i-\sigma\left(\boldsymbol{x}_i^\top \boldsymbol{\beta}\right)\right]\boldsymbol{x}_i+\boldsymbol{\beta}.\end{aligned}$$

Expressed in matrix notation which will be in use when solving the code assignment:

$$\nabla \ell(\beta) = -X^\top (y - \sigma(X\beta)) + \beta,$$

Compute the Hessian

Compute the Hessian matrix $H = \nabla^2 \ell(\beta)$:

The second derivative of $\ell(\beta)$ is:

$$\nabla^2 \ell(\beta) = -X^\top \left(\frac{\partial}{\partial \beta}(y - \sigma(X\beta)) \right) + I.$$

Compute $\frac{\partial}{\partial \beta}\sigma(X\beta)$:

$$\frac{\partial}{\partial \beta}\sigma(X\beta) = X^\top W,$$

so:

$$\nabla^2 \ell(\beta) = X^\top W X + I = H.$$

where W is an $n \times n$ diagonal matrix with entries $w_i = \sigma(x_i^\top \beta)(1 - \sigma(x_i^\top \beta))$,

Which would be in use in the code assignment

Q2

When implementing the Laplace approximation for Bayesian logistic regression on the Breast Cancer Survival dataset and overwriting the gradient and Hessian functions to compute classifi-

cation accuracy, precision, recall, and F1-score, the results remain consistent with those obtained earlier (see Table 6).

Metric	**Value**
Accuracy	0.71
Precision	0.74
Recall	0.91
F1 Score	0.82

Table 6: Performance metrics for Laplace approximation.

This consistency indicates that the method successfully finds an optimal solution for β using the Laplace approximation. Specifically, the Laplace method refines the posterior distribution of β as a Gaussian centered around the MAP estimate. This refinement does not alter the underlying optimal β, as expected, because the gradient and Hessian calculations in this approach closely resemble those derived from L_2-regularized logistic regression.

While the results align, further analysis could investigate whether the Laplace approximation improves uncertainty quantification compared to regularized logistic regression. Metrics such as confidence intervals for β or predictive distributions could provide deeper insights, as the Laplace approximation yields an approximate posterior distribution. One potential way to demonstrate this would be to present the results with confidence intervals derived from the posterior to be able to take more informed decision on a persons chance of survival