Advanced Techniques in Regression and Unsupervised Learning for Hepatitis Mortality Analysis

Iñigo Exposito & Oriol Gelabert

December 2024

Contents

1 Introduction

Hepatitis is a viral liver infection that can lead to severe complications such as cirrhosis, liver cancer and even death. Understanding the factors influencing its progression is essential for improving patient outcomes, as early diagnosis and appropriate treatment are crucial to preventing patient mortality. For example, El-Serag (2012) highlighted that chronic hepatitis C can lead to liver cancer (a condition associated with a very high mortality rate), with prognosis depending on disease stage and liver condition [6]. Lok and McMahon (2009) emphasized that age, gender and cirrhosis presence significantly affect hepatitis B progression [7]. Early diagnosis and treatment, as noted by Choi et al. (2018), are essential in preventing complications such as cirrhosis and liver cancer, particularly in patients with coexisting conditions like HIV [8].

The goal of this study is to prevent death from hepatitis, as many patients remain undiagnosed or receive insufficient care. It also aims to assist healthcare professionals in making more informed decisions regarding treatment and surgical options. By analyzing detailed patient data, patterns can be identified that enable more accurate disease predictions, enhancing diagnostics and providing better guidance for medical decision-making, as discussed by Borgia et al. (2017) [9].

The outcome variable in this study is the mortality status of a patient. The input variables consist of various medical parameters and physical symptoms related to liver health. As a result, its primary objective remains the prediction of patient mortality due to liver complications within a high-dimensional feature space, where the interactions between all variables are considered. In order to achieve this task, we will perform different logistic regression techniques and Bayesian model averaging approach to try to both predict and infer a model. To better understand the most relevant factors affecting hepatitis, we performed principal component analysis to capture and summarize the most significant features. Additionally, cluster analysis was conducted to group individuals with similar clinical characteristics and symptoms based on previous obtained results.

2 Related works

The prediction of hepatitis progression has been studied using traditional statistical methods and machine learning techniques. Early studies, like McHutchison et al. (1998), used logistic regression [1], while later research, such as Nguyen (2020) and Majzoobi (2022), applied Support Vector Machines and Random Forests, respectively, to improve prediction accuracy [14], [15]. These hybrid approaches enhanced prediction accuracy but require large datasets and significant computational resources.

In addition, recent advances in deep learning, as seen in the work of Pubmed (2021), offer promising potential for automating the analysis of medical data [**?**]. However, despite their advantages, these methods have not yet been fully integrated into clinical practice, and manual intervention remains common. Combining traditional methods with machine learning techniques presents a balanced approach, offering both interpretability and improved accuracy.

3 Dataset

The database used for this project, named *hepatitis*, was sourced from *Kaggle*. It includes a sample of 150 individuals, with 20 distinct variables, all related to patients who are currently suffering from or have previously suffered from hepatitis. The recorded data encompasses various details such as the patient's age, gender, symptoms, treatment status and other relevant medical information.The collected variables can be categorized into different groups:

- **Patient Characteristics:** age, sex
- **Symptoms:** fatigue, malaise, anorexia
- **Liver Complications:** liver_big, liver_firm
- **Treatment:** steroid, antivirals
- **Clinical Indicators:**
  - *Indicators of Portal Hypertension:* spleen, spiders, varices, ascites
  - *Severity Indicators:* bilirubin, alkphosphate, sgot, albumin, protime, histology
- **Response Variable:** class

3.1 Features

This data will be analyzed to assess its impact on individuals with hepatitis. The collected information is summarized in the following table:

Variable Name	**Description**	**Type**	**Categories**
class	Whether the patient is deceased or alive	Binary	Deceased, Survived
age	Age of the patient	Discrete	-
sex	Gender of the patient	Binary	Male, Female
steroid	Whether the patient is taking corticosteroids	Binary	Yes, No
antivirals	Whether the patient is taking antivirals	Binary	Yes, No
fatigue	Whether the patient has generalized fatigue	Binary	Yes, No
malaise	Whether the patient has general discomfort	Binary	Yes, No
anorexia	Whether the patient has loss of appetite	Binary	Yes, No
liver_big	Whether the patient has liver enlargement (hepatomegaly)	Binary	Yes, No
liver_firm	Whether the patient has liver hardness (cirrhosis)	Binary	Yes, No
spleen	Whether the patient has spleen enlargement (splenomegaly)	Binary	Yes, No
spiders	Whether the patient has capillary dilation (telangiectasia)	Binary	Yes, No
ascites	Whether the patient has free fluid in the abdomen	Binary	Yes, No
varices	Whether the patient has varices in the abdomen	Binary	Yes, No
bilirubin	Bilirubin level	Continuous	-
alkphosphate	Alkaline phosphatase level	Continuous	-
sgot	Hepatic enzyme GOT level	Continuous	-
albumin	Albumin level	Continuous	-
protime	Blood coagulation time	Continuous	-
histology	Whether liver damage is observed at the cellular level	Binary	Yes, No

Table 1: Description of variables in the hepatitis dataset

3.2 Handling Missing Values

Once we have understood our database, we perform a Descriptive Statistics analysis. Based on the results, we have observed a significantly high number of missing values (NA). Specifically, the missing values represent 5.33% of the entire dataset. The variables obtained through blood analyses, in particular, show a notably high percentage of missing values. The case of the variables *protime*, *alkphosphate* and *albumin* are especially significant, with 42%, 18.6% and 10.6% of its data missing respectively. Given these observations, it is crucial to proceed with caution when applying various statistical techniques, as missing values could introduce complications. To address this, we opted to impute them utilizing the `mice()` function in R. Specifically, for binary variables, we employed logistic regression (`logreg`), while for continuous variables, we used predictive mean matching (`pmm`). On the other hand, no outliers were observed in the collected data, and thus, we assume that the data is accurate. For a more detailed understanding of the clinical variables, additional information about hepatitis can be found in Appendix A.

4 Regression

We aim to explore various logistic regression techniques to evaluate both predictive performance and model selection. In his paper, Dr. John McHutchison [1] performed logistic regression analysis to assess the probability of treatment response based on multiple variables, as well as Kaplan-Meier, survival analysis to evaluate the time to events such as cure or complications. Similarly, in our case, we will initially apply different regression techniques to identify the best predictive models. Following this, for a more in-depth analysis, we will employ additional criteria, including Bayesian Model Averaging, to infer the most significant covariates that influence our binary outcome.

4.1 LASSO logistic regression

We begin by considering our 19 covariates, including all possible two-way interactions. This results in a total of 190 parameters, while the dataset contains only 155 patients. Since the number of parameters exceeds the number of observations, the maximum likelihood estimators for the model parameters are likely to exhibit high variance. To address this issue, we apply LASSO regression, which helps reduce the number of covariates by imposing a penalty that encourages sparsity in the model. Let Y be the response vector, X the design matrix and β the vector of coefficients. Then its mathematical formulation is as follows:

$$\hat{\beta} = \arg\min_{\beta} \left(\frac{1}{2n} \|Y - X\beta\|^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right).$$

After implementing LASSO, the number of non-zero parameters was substantially reduced. The optimal regularization parameter($\lambda = 0.02$) was determined through cross-validation. Furthermore, the model achieved an AIC value of 32.62.

Taking into account these estimates, we proceed to perform Adaptive LASSO. Adaptive LASSO is a regularization technique that enhances variable selection by assigning weights (w_j) to each coefficient. Unlike the standard LASSO, Adaptive LASSO allows different penalties for each coefficient. Its mathematical formulation is as follows:

$$\hat{\beta}^{\text{adaptive}} = \arg\min_{\beta} \left(\frac{1}{2n} \|Y - X\beta\|^2 + \lambda \sum_{j=1}^{p} w_j |\beta_j| \right).$$

The regularization parameter, $\lambda = 53.062.490$, was selected through cross-validation. For the weights (w_j), the initial LASSO estimates were used, assigning a minimal value to coefficients that were originally null to impose an infinite penalty. This approach ensures that the number of non-zero parameters is, at most, the same as in the LASSO model. The resulting AIC value is 85.5.

After fitting our models, we evaluate their performance by analyzing the ROC curves, which compare the accuracy of the LASSO and Adaptive LASSO regression models. These curves illustrate the trade-off between sensitivity, the proportion of actual positives correctly identified, and specificity, the proportion of actual negatives accurately classified. Together, sensitivity and specificity provide a balanced assessment of a model's ability to detect true positives while minimizing false positives. The area under the curve (AUC) serves as a key measure of discriminative ability. A higher AUC indicates better overall performance, with an AUC of 1 representing a perfect model and an AUC of 0.5 indicating no discriminative power.

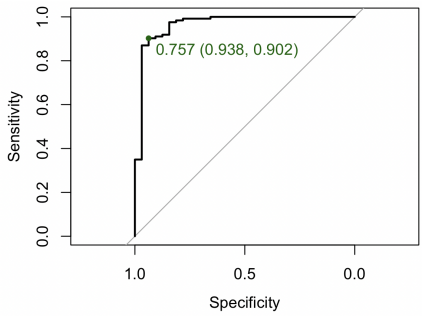

Figure 1: ROC curve for LASSO

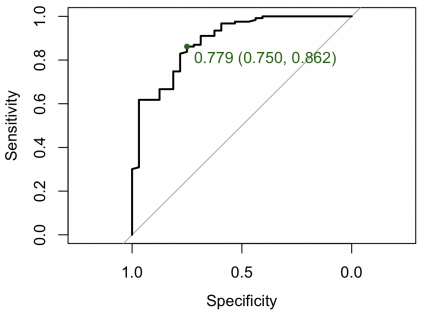

Figure 2: ROC curve for Adaptive LASSO

The LASSO model demonstrates high predictive accuracy, as evidenced by its sensitivity and specificity values. The AUC value of approximately 0.965 indicates near-perfect performance. This is further supported by the model's classification results, with 111 true positives and only 12 false negatives, alongside 30 true negatives and just 2 false positives. These metrics reflect strong performance in correctly predicting both positive and negative instances.

In contrast, the Adaptive LASSO model exhibits lower sensitivity and specificity, resulting in a weaker overall performance. The AUC value of 0.88 underscores this decline compared to the LASSO model. The adaptive model records more misclassifications, with 8 false positives and 17 false negatives, which contribute to its diminished predictive accuracy. Despite this, the Adaptive LASSO still achieves relatively strong performance, albeit not on par with the LASSO model.

	Predicted 0	Predicted 1
Fitted 0	30	2
Fitted 1	12	111

Figure 3: Confusion matrix for LASSO

	Predicted 0	Predicted 1
Fitted 0	24	8
Fitted 1	17	106

Figure 4: Confusion matrix for Adaptive LASSO

4.2 Bayesian Model Averaging

In previous approach we performed the selection of our model through cross-validation as the aim was to obtain good predictions of the binary output of the database. Now, we will try to perform model inference to understand which covariates have more effect on the output using both BIC/EBIC criterion and Bayessian Model Averaging (BMA). The latter method will be used to obtain marginal posterior inclusion probabilities and understand which variables are indispensable for the model.

We begin by introducing the BIC and EBIC criteria for model selection. Both of these criteria are consistent for model selection and, unlike the cross-validation approach used previously, are particularly useful when the goal is to understand or infer a model. The BIC and EBIC aim to minimize the following expressions:

$$BIC : -2\log p(y \mid \theta) + \log(n)\|\beta\|_0$$

$$EBIC : -2\log p(y \mid \theta) + \log(n)\|\beta\|_0 + 2\log\left(\frac{d}{\|\beta\|_0}\right)$$

Both criterion have a penalty through the L_0 norm but take into consideration the dimensionality in the penalty. Initially we consider our 19 covariates and fit the model both via *BIC* criterion and *EBIC* criterion using the *mombf* package, *BestBIC* and *BestEBIC* functions. The selected significant individual covariates are the following for each criterion:

BIC: protime, albumin,spiders, bilirubin
EBIC: protime, ascites, bilirubin

After this first approach, BMA is considered. Bayesian model averaging generates posterior inclusion probabilities for each covariate (and interaction if its the case) and also can generate point estimates. First, to consider a brief interpretation of the model, the individual 19 covariates are selected and using *mombf* package *model_selection* function posterior inclusion probabilities for each variable are computed and coefficient values for the highest probability model are analyzed. This results are shown in Table 4 at Apendix B.

In a second approach contemplating more complex models, we recover first order interactions approach used for predictive analysis, adding up to 19 principal covariates and 171 pairwise interactions. An important thing to consider is hierarchy between interactions: if we include an interaction in the model, both of its principal covariates should be included too. To perform this method we use *BAS* package for BMA models which allows forcing hierarchy constraints when computing probabilities for all possible models. In order to perform Bayesian model averaging and selection, first we set prior probabilities. For the model space, we define a Beta-Binomial prior with parameters $a = 1$ and $b = 1$, that is a uniform prior over model space. This prior assigns equal prior probabilities to all subsets of predictors. For the coefficients a CCH Mixture prior has been used, a mixture of Zellner's g-priors that introduces flexibility in specifying the prior beliefs about the regression coefficients.

The inclusion probabilities obtained via BMA with hierarchical interactions can be found in Table 5 at Appendix B.

4.2.1 Interpretation

We can interpret the significant variables obtained by each method. While information criterion only selected few variables, BMA provided us inclusion probabilities for every variable and in a more complex model even each interaction. Blood coagulation time named *protime* seems one of the most important variables as it has been selected in all 4 methods proposed. *Bilirubin* and *Albumin* levels also have been selected as an important variable in 4 and 3 methods respectively. An interesting result arouses from the *alk_phosphate* covariate that despite being neglected in 3 previous methods, has high inclusion posterior probabilities due to its interaction with other covariates, the most important being with binnary covariate *spiders*.

Now, we will interpret the following model obtained using BMA approach of the original 19 covariates in terms of odds ratios. The logistic regression model estimates the probability of survival, and the odds ratios (OR) represent the effect of each predictor variable on the odds of survival. The variables with highest marginal probabilities present the following coefficients:

Variable	**Coefficient**	**Odds Ratio (OR)**
Intercept	-4.614	$e^{-4.614} \approx 0.009$
protime	0.036	$e^{0.036} \approx 1.036$
albumin	1.240	$e^{1.240} \approx 3.45$
bilirubin	-0.564	$e^{-0.564} \approx 0.569$
sex=male	-0.0012	$e^{-0.0012} \approx 0.988$
spiders=True	-0.654	$e^{-0.654} \approx 0.5199$
anorexia=True	0.699	$e^{0.699} \approx 2.01$
steroid=True	0.454	$e^{0.454} \approx 1.57$

Table 2: Coefficient and Odds Ratio for each variable

An odds ratio greater than 1 indicates a positive relationship, while an odds ratio less than 1 indicates a negative relationship. Odds ratios for each predictor variable are summarized in Table 2.

After analyzing the previous table, the intercept has an odds ratio of $e^{-4.614} \approx 0.009$, meaning that when all predictor variables are set to their reference category, the odds of survival are very low, around 1%. Regarding the sex of the patient, being male slightly reduces the odds of survival by about 2% compared to females. Additionally, using steroids increases the odds of survival by a factor of 1.57 compared to not using steroids. This is very logical since steroids are useful medical treatment for liver diseases.

The presence of anorexia significantly increases the odds of survival, with an odds ratio of $e^{0.699} \approx 2.01$, meaning that having anorexia increases the odds of survival by a factor of 3.08 compared to not having anorexia. This result may seem counterintuitive, as loss of appetite is generally associated with a higher mortality rate. However, anorexia might indicate underlying health issues, which could prompt a patient to seek medical attention sooner. Early diagnosis and intervention could therefore improve the chances of survival. In this context, anorexia is more of an indicator suggesting liver problems, rather than a direct factor causing death. It may not directly determine survival, but it could be a useful signal for reducing the probability of mortality when addressed promptly.

On the other hand, spider-like lesions decrease the odds of survival. The odds ratio for spiders is $e^{-0.654} \approx 0.5199$, meaning that having spider-like lesions reduces the odds of survival to about 49% of the odds for those without spider-like lesions.

Biochemical indicators of severity can assist medical professionals in interpreting the results. The coefficient for bilirubin has an odds ratio of $e^{-0.564} \approx 0.569$, which means that for each unit increase in bilirubin, the odds of survival decrease to approximately 53.1% of the previous odds. In contrast, albumin has a positive effect on survival, with an odds ratio of $e^{1.240} \approx 3.45$. This indicates that for each unit increase in albumin, the odds of survival increase by a factor of 3.45. Lastly, prothrombin time also has a small positive effect on survival, with an odds ratio of $e^{0.036} \approx 1.036$, suggesting that for each unit increase in prothrombin time, the odds of survival increase by about 3.6%.

5 Unsupervised learning

Our objective in this section is to cluster individuals based on similar clinical indicators to assess the relevance of these parameters in determining the mortality status of liver diseases. Given the frequent interconnections among medical variables, we will begin by reducing the number of parameters to pinpoint the most significant ones within each component. Previous studies have applied similar clustering techniques to identify key clinical parameters related to liver disease outcomes. For instance, [10] utilized clustering methods to identify prognostic factors for liver cirrhosis, while [11] explored the clustering of clinical features for predicting liver cancer survival. By reducing dimensionality and focusing on the most significant clinical variables, we aim to improve the interpretability and predictive accuracy of our model.

5.1 Principal Component Analysis

The continuous variables in the dataset reflect various indicators obtained from blood tests. Additionally, some binary variables describe physical symptoms related to liver diseases and hypertension. Our objective is to group the most highly correlated variables under the same principal component, such that each component will correspond to multiple clinical parameters that are interrelated. After calculating the covariance matrix, we find that the most correlated variables are `spleen_palpable`, `spiders`, `ascites`, `varices`, `bilirubin`, `alk_phosphate`, `sgot`, `albumin` and `protime`.

Dimensionality reduction methods include PCA and Probabilistic PCA. A Shapiro-Wilk test revealed that the variables do not follow a normal distribution with a significance level of $\alpha = 0.05$, making Probabilistic PCA unsuitable due to its Gaussian assumption. In our case, traditional PCA is more adequate for analyzing the data. Mathematically, PCA relies on the eigenvalue decomposition of the covariance matrix to identify the directions (eigenvectors) that capture the most variability in the data. The data is then projected onto a lower-dimensional subspace defined by the principal components with the largest eigenvalues, preserving as much information as possible.

Before performing Principal Component Analysis, we examined the eigenvalues of the obtained correlation matrix. In order to determine the number of eigenvalues to be selected for our analysis, we use the elbow method. If we choose the number of eigenvalues based on the eigenvalue ratio, only one eigenvalue is selected, which will definitely lead to a difficult interpretation. Taking into account that one of the main objectives of PCA is to reduce dimensionality while keeping the maximum possible variance of data, we select the number of components based on the amount of variance explained by each component.

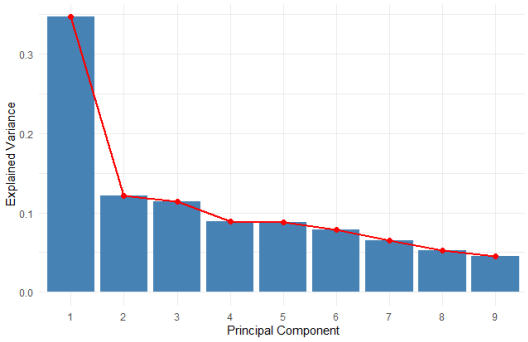

Figure 5: Explained variance by each principal component

	Eigenvalue	Variance proportion
PC1	3.13	0.347
PC2	1.1	0.121
PC3	1.026	0.113
PC4	0.79	0.088
PC5	0.78	0.087
PC6	0.71	0.078
PC7	0.58	0.065
PC8	0.47	0.052
PC9	0.4	0.044

Figure 6: Eigenvalues and Explained Variance for each Principal Component

After analyzing both previous figure and table, we reach the conclusion that the choice of 3 eigenvalues is the perfect decision leading to three different principal components. With three principal components, we manage to explain more than 57% of the total variance. If we add more eigenvalues the increase in the amount of variance explained does not increase considerably, leading to less than 9% per each principal component.

Once the number of components has been determined, we applied the Principal Component Analysis method to the initial data matrix. The following figure summarizes the results and allows the interpretation of each principal component based on the loadings:

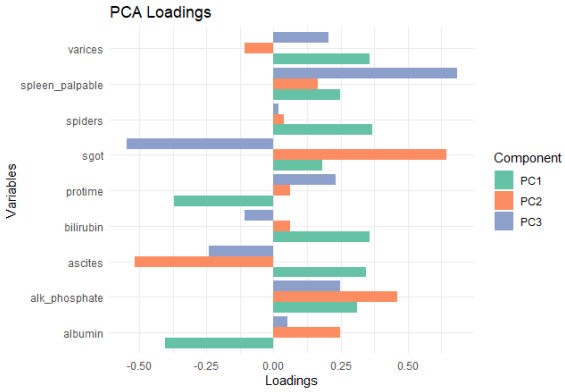

Figure 7: PCA loadings

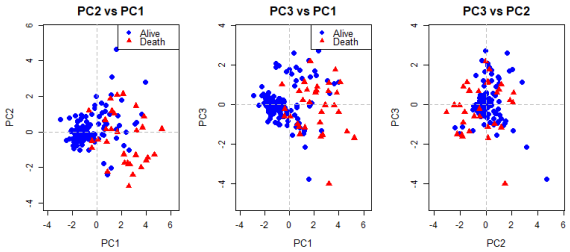

Figure 8: Mortality across different PCs

Interpretation

- **PC1: General Liver Dysfunction.** PC1 reflects a general pattern of liver dysfunction, with strong correlations to symptoms like *spiders*, *ascites*, *varices*, *bilirubin*, and *alkaline phosphatase*, which are clinical indicators of severe liver disease. Negative correlations with *albumin* and *protime* suggest that worsening liver function leads to more severe symptoms. Higher PC1 values indicate more advanced liver dysfunction.

- **PC2: Liver Enzyme Activity.** PC2 is primarily associated with liver enzyme activity, showing strong correlations with *sgot* and *alkaline phosphatase*, which are markers of liver injury. Negative correlations with *ascites* suggest that PC2 is more focused on biochemical markers than physical symptoms. Weak correlations with *albumin* and *protime* further support this idea.

- **PC3: Physical Symptoms of Liver Disease.** PC3 is influenced mainly by *spleen palpable*, indicating it represents physical symptoms like splenomegaly, common in advanced liver disease. Negative correlations with *sgot* suggest lower enzyme activity in later stages of liver damage. *Ascites* and *varices* also contribute to PC3, but less strongly than *spleen palpable*. This component captures physical symptoms related to severe liver dysfunction.

Finally, we want to study the relationship of principal components with the mortality status of each patient, as it was our target variable in logistic regression. As mentioned before, higher values in PC1 reflects more severe liver disease. In the first plot of figure 4, deceased patients (red triangles) tend to cluster on the right side of the PC1 axis, where values are positive. This suggests a strong association between advanced liver dysfunction and mortality. Along the PC2 axis, which represents liver enzyme activity, there is no clear separation between alive and deceased patients. This indicates that PC2 contributes less to differentiating mortality compared to PC1.

Similar to the first plot, the second plot shows that deceased patients cluster on the right side of the PC1 axis, reinforcing the idea that advanced liver dysfunction is strongly associated with mortality. On the PC3 axis, positive values reflect more severe physical symptoms, such as splenomegaly (spleen palpable). While deceased patients show slightly higher PC3 values, the trend is not as strong as with PC1. Therefore, PC1 remains the most significant differentiator of mortality in this comparison.

The last plot examines the relationship between *liver enzyme activity* (PC2) and *physical symptoms* (PC3). In this case, there is no clear separation between alive and deceased patients, as deceased patients are scattered throughout the plot. Although PC3 contributes slightly to mortality patterns, PC2 and PC3 alone are less effective in distinguishing mortality compared to PC1.

5.2 Cluster Analysis

Based on previous results, we found that PCA was a promising method for dimensionality reduction and proceed with cluster analysis, through implementation of both K-means and agglomerative hierarchical methods. As previously mentioned, for the clustering, the values that each individual takes on those three principal components will be considered in order to group individuals that share similar symptoms and clinical indicators.

5.2.1 Hierarchical agglomerative clustering

In this case, the Ward method will be applied to the data to form clusters. The Ward method takes all individuals and calculates the distances between them using squared Euclidean distance. Once this is done, the two individuals with the smallest distance are grouped together. Then, the distances between the remaining $n-1$ elements (where $n-2$ of them are individual data points and the other element is a cluster of 2 individuals). This process continues until all individuals are grouped into a single cluster.

Since the applied method is hierarchical, a dendrogram can be constructed. The dendrogram obtained in this case is as follows in Figure 9. Once this is done, it is necessary to determine the number of clusters to work with. Via inspection, 4 clusters are considered, marked in blue in Figure 9. Then the selected clusters are analyzed in Figure 10 and are represented in 3D plot of Figure 15 found at Apendix C.

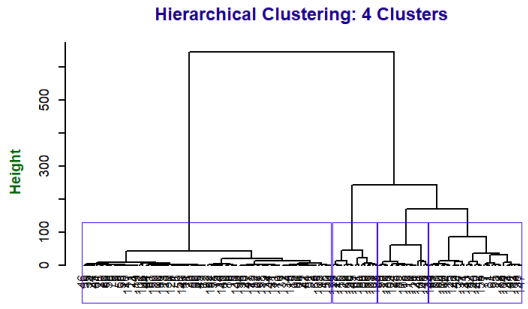

Figure 9: Dendrogram of the hierarchical clustering.

Cluster	PC1	PC2	PC3	Size
C1	-1.25	-0.13	-0.11	88
C2	1.49	0.51	1.34	33
C3	0.93	1.23	-1.2	18
C4	2.74	-1.75	-0.82	16

Figure 10: Ward's Method and PCA

In this table, the average values of each factor for the individuals in each cluster are shown, that is, the centroids, as well as the number of individuals in each cluster. Looking at the results, we can say that the value of PC1 in cluster 4 is significant. The centroids of each cluster were analyzed based on the principal components as follows:

- **Cluster 1:** This group shows low liver dysfunction (PC1 = -1.25), low liver enzyme activity (PC2 = -0.13), and few physical symptoms (PC3 = -0.11), indicating individuals with less severe liver disease.
- **Cluster 2:** This group reflects advanced liver dysfunction (PC1 = 1.49), higher liver enzyme activity (PC2 = 0.51), and pronounced physical symptoms (PC3 = 1.34), suggesting individuals with more severe liver disease.
- **Cluster 3:** This group displays moderate liver dysfunction (PC1 = 0.93), higher liver enzyme activity (PC2 = 1.23), and fewer physical symptoms (PC3 = -1.20), indicating individuals with moderate liver disease severity.
- **Cluster 4:** This group shows severe liver dysfunction (PC1 = 2.74), low liver enzyme activity (PC2 = -1.75), and few physical symptoms (PC3 = -0.82), suggesting individuals with advanced liver dysfunction but fewer physical symptoms.

5.2.2 K-means clustering

In this case, the non-hierarchical K-means method was applied to the data to form clusters. This algorithm iteratively assigns data points to the nearest cluster centroid, then updates the centroids based on the mean of the points in each cluster. This process continues until the centroids no longer change with the aims to minimize the within-cluster variance, resulting in compact and well-separated clusters. In order to decide the number of clusters, we used elbow method and the gap statistic. Both visual inspection and gap statistic gave rise to similar conclusion: the best number of clusters is four. In fact, the maximum gap is observed at $k = 4$ with a value of 0.727.

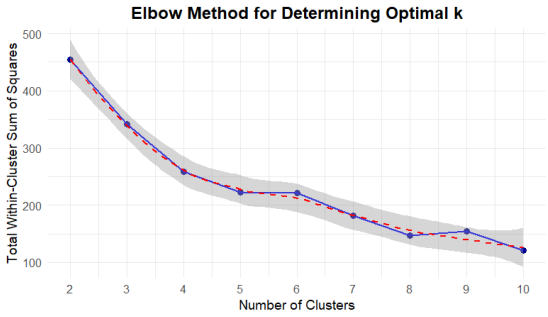

Figure 11: Elbow method

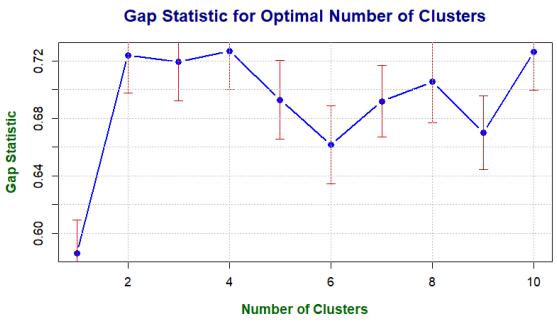

Figure 12: Gap Statistic for different K.

Looking at the following table, similar outcomes were obtained to those when 4 clusters were specified in the previous section. Therefore, we can conclude that the created clusters are stable. In this case, the second cluster (C2) is very significant to the first principal component. The interpretation is really similar to the previous part. In this case, Cluster 1 and Cluster 3 represent moderate liver dysfunction, while Cluster 2 reflects more severe liver dysfunction with biochemical markers. Cluster 4 represents the least severe cases, with a large group size. Visual representation of the clusters found via K-means can be found in Figure 16 at Apendix C.

Cluster	PC1	PC2	PC3	Size
1	0.602	0.469	1.619	21
2	2.942	-1.117	-0.12	24
3	1.16	1.411	-0.898	22
4	-1.236	-0.159	-0.129	88

Table 3: Principal Component Values, Clusters, and Group Sizes

Finally, we analyze the distribution of mortality status across the different clusters. A clear bijective correspondence can be established between the clusters in both the hierarchical and K-means models. Therefore, we will focus on interpreting the results from one of these methods, specifically the survival status in the Ward's model.

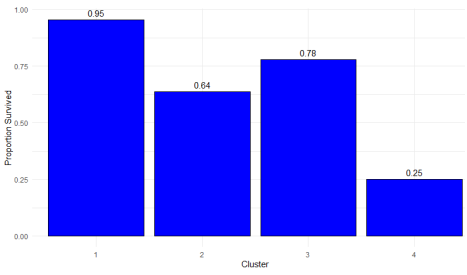

Figure 13: Survival status in Ward's method

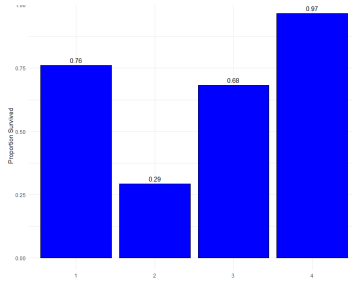

Figure 14: Survival status in K-Means

Cluster 1 is associated with less severe liver disease and, as a result, exhibits a lower mortality rate among its members. Clusters 2 and 3 are linked to moderate liver disease, with Cluster 2 patients experiencing more pronounced physical symptoms. Consequently, the survival rate is relatively high in these clusters, particularly in Cluster 3, where physical symptoms are less common. However, the survival rate in both clusters is still lower than that of Cluster 1. Cluster 4, on the other hand, shows the highest mortality rate, reflecting the advanced liver dysfunction and significantly reduced enzyme activity in these patients.

6 Conclusion

This project allowed us to explore various model selection techniques, including LASSO and Bayesian model averaging, in the context of a prediction problem. For regression tasks, we employed different approaches based on the specific objectives. LASSO and Adaptive LASSO were utilized for predictive analysis, with the optimal regularization parameter selected through cross-validation. These methods were particularly suited for improving predictive accuracy by shrinking coefficients and selecting relevant features. In contrast, Bayesian model averaging was employed to enhance interpretability by accounting for model uncertainty and combining insights from multiple models.

The results indicate that LASSO outperformed Adaptive LASSO in terms of predictive accuracy. This suggests that LASSO's regularization approach may better balance complexity and accuracy in this specific context, while Adaptive LASSO, despite its potential for refinement, showed relatively weaker performance in prediction tasks. Consistent model selection methods like BIC, EBIC and Bayessian Model Averaging let us identify the most significant variables for the survival output as well as consider some important interactions. Posterior inclusion probabilities and coefficient posterior mean estimates give us an idea of how changes on most important covariates impact the survival probability.

Given more time and better computational resources, additional methodologies could be explored to enhance the effectiveness of the models. In the context of logistic regression, incorporating higher-order interaction terms could be an avenue for improvement. Thus far, the analysis has been limited to second-order interactions, leaving the potential impact of more complex relationships between variables unexplored. Additionally, other regularization techniques, such as Elastic Net, could be explored to strike a balance between the L1 and L2 penalties, potentially improving performance when dealing with correlated features.

Regardind unsupervised learning, due to the non-normality of the variables, we applied Principal Component Analysis (PCA) for dimensionality reduction, which facilitated the interpretation of clinical patterns. We then used the reduced-dimensional data to identify clusters using both K-means and hierarchical clustering models. Both algorithms yielded similar results and interpretations. With more time, it would have been valuable to explore transforming the variables to achieve normality, which could have allowed us to apply Probabilistic PCA. Additionally, for a more probabilistic approach to clustering, we could have implemented Gaussian Mixture Models.

A Hepatitis overview

Hepatitis is an infectious, immunological or toxic disease that causes inflammation of the liver. The general symptoms of most types of hepatitis include weakness, generalized fatigue, loss of appetite, fever, chills, nausea, jaundice and darkened urine. Hepatitis caused by viruses is an endemic disease and can affect people of all ages. There are different types of viral hepatitis: A, B, C, D, E, F and G [12].

The diagnosis of this disease is performed through blood tests and/or liver biopsy. Blood tests measure the levels of various molecules produced in the liver, as well as other parameters related to liver function. These include bilirubin and albumin levels, hepatic enzyme counts such as GOT, alkaline phosphatase and blood coagulation time.

The bilirubin level is directly proportional to liver damage, while low albumin levels indicate liver failure. On the other hand, although alkaline phosphatase and GOT enzyme levels alone are not highly specific, deviations in both are indicative of liver damage. Finally, since most coagulation factors are synthesized in the liver, measuring blood coagulation time helps detect potential liver problems. All these parameters are also used to assess the severity of the disease. A liver biopsy is performed to detect liver damage at the cellular level.

In the initial phase of the disease, the liver enlarges (hepatomegaly). As the disease progresses, the liver begins to harden, leading to cirrhosis. Although these two conditions rarely occur simultaneously, they can appear in some patients. When cirrhosis develops in the liver, complications such as portal hypertension arise.

The portal vein is a blood vessel that carries blood from the intestines to the liver for filtration, after which the blood is transported to the heart. When the liver becomes fibrotic, there is increased pressure in the blood vessels of the liver, which impairs blood transport. Consequently, blood accumulates in blood vessels located upstream of the liver, such as the spleen, causing it to enlarge (splenomegaly). Thus, we can conclude that splenomegaly is a clinical sign of portal hypertension in the context of hepatitis. In addition to splenomegaly, other clinical manifestations of portal hypertension include capillary dilation (telangiectasia), the presence of free fluid in the abdominal cavity (ascites) and abdominal varices, among others.

Regarding treatment, corticosteroids, which are anti-inflammatory drugs, are often administered to counteract the liver's inflammatory response. Since the disease can sometimes be caused by a virus, antivirals may also be used. However, antivirals are generally ineffective for most types of viral hepatitis and are usually only administered in cases of hepatitis B and C. The best results are observed in hepatitis C, where many patients achieve recovery.

B BMA Inclusion Probabilities and Posterior Mean

Predictor	**Estimate**	**2.5% CI**	**97.5% CI**	**MargPP**
(Intercept)	-4.614	-8.897	0.938	0.993
protime	0.036	0.000	0.062	0.947
albumin	1.240	0.000	2.178	0.899
bilirubin	-0.564	-1.040	0.000	0.871
sexmale	-0.012	-1.523	1.531	0.861
spidersTrue	-0.654	-1.605	0.256	0.722
anorexiaTrue	0.699	-0.211	1.712	0.690
steroidTrue	0.454	-0.519	1.427	0.630
ascitesTrue	-0.385	-1.491	0.809	0.617
liver_firmTrue	0.450	0.000	1.468	0.551
malaiseTrue	-0.063	-1.032	0.911	0.360
varicesTrue	-0.040	-1.031	0.883	0.326
spleen_palpableTrue	-0.078	-1.096	0.779	0.299
fatigueTrue	0.127	-0.649	1.174	0.290
antiviralsTrue	0.055	-0.720	1.060	0.249
age	-0.007	-0.066	0.000	0.237
histologyTrue	0.013	-0.575	0.705	0.171
alk_phosphate	0.000	-0.001	0.006	0.111
liver_bigTrue	0.030	0.000	0.668	0.109
sgot	-0.000	0.000	0.000	0.042

Table 4: Coefficient Estimate and margin probabilities of each covariate

Predictor	Inclusion Probability
Intercept	1.000
protime	0.992
sexmale	0.978
spidersTrue	0.969
alk_phosphate	0.967
albumin	0.958
age	0.926
spidersTrue : alk_phosphate	0.925
liver_firmTrue	0.901
bilirubin	0.848
anorexiaTrue	0.759
liver_firmTrue : alk_phosphate	0.548
alk_phosphate : protime	0.347
fatigueTrue	0.302
malaiseTrue	0.219
ascitesTrue	0.146
varicesTrue	0.172
liver_bigTrue	0.114
spleen_palpableTrue	0.084
steroidTrue	0.084
histologyTrue	0.077
sgot	0.074
antiviralsTrue	0.080

Table 5: Inclusion Probabilities of each principal covariate and some significant interactions

C Clustering figures

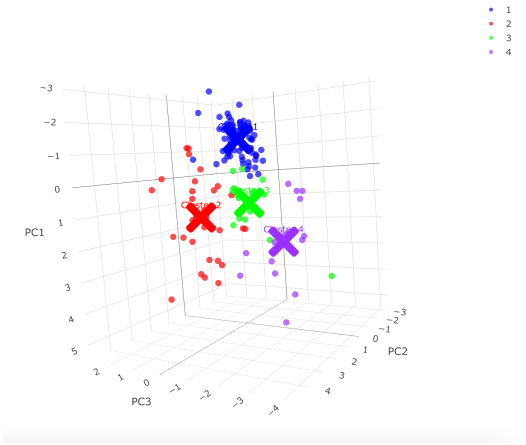

Figure 15: Representation through Ward's method of 4 clusters, with centroids marked with a cross.

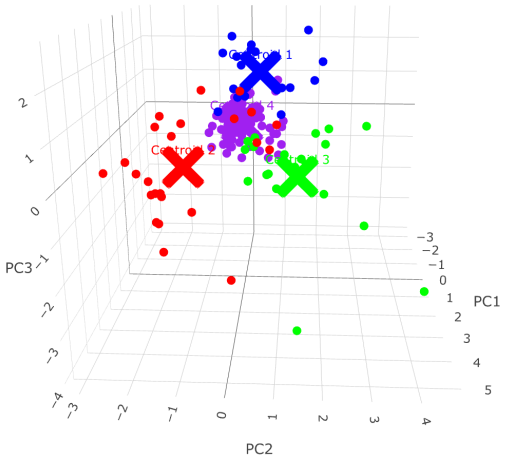

Figure 16: Representation through K-means algorithm of 4 clusters, with centroids marked with a cross.

D Bibliography

References

[1] McHutchison, J. G., Gordon, S. C., Schiff, E. R., et al. (1998). A comparison of treatment for chronic hepatitis C with interferon alfa-2b alone or in combination with ribavirin. *New England Journal of Medicine*, 339(21), 1485-1492.

[2] Tran, H. T., Nguyen, T. H., & Phan, T. L. (2016). Predicting progression from hepatitis B to cirrhosis: A statistical approach. *Journal of Hepatology*, 64(3), 623-629.

[3] Papatheodoridis, G. V., Manolakopoulos, S. S., & Hadziyannis, S. J. (2017). Efficacy of direct-acting antivirals in the treatment of hepatitis C: A meta-analysis. *World Journal of Gastroenterology*, 23(23), 4131-4140.

[4] El-Serag, H. B. (2012). Hepatitis C: Epidemiology and risk factors. *Nature Reviews Gastroenterology & Hepatology*, 9(2), 70-78.

[5] Zhang, Y., & Zhao, L. (2018). Predicting hepatitis C progression using deep learning models. *Computers in Biology and Medicine*, 102, 9-17.

[6] El-Serag, H. B. (2012). Hepatocellular carcinoma and hepatitis C in the United States. *American Journal of Gastroenterology*, 107(3), 411-417.

[7] Lok, A. S., & McMahon, B. J. (2009). Chronic hepatitis B: update 2009. *Hepatology*, 50(3), 661-662.

[8] Choi, H. Y., et al. (2018). Hepatitis B and C: Advances in the treatment and diagnosis of chronic liver diseases. *Clinical and Molecular Hepatology*, 24(4), 343-352.

[9] Borgia, S. M., et al. (2017). Personalized treatment regimens for hepatitis C virus infection: The next frontier in patient care. *Liver International*, 37(7), 1015-1023.

[10] Do, T., et al. (2017). Clustering methods for identifying prognostic factors of liver cirrhosis. *Journal of Hepatology*, 67(3), 487-495.

[11] Liu, J., et al. (2018). Application of clustering techniques for predicting survival in liver cancer patients. *Liver Cancer*, 7(4), 240-247.

[12] World Health Organization (WHO). (2020). *Global hepatitis report 2020*. Geneva: World Health Organization. Available at: `https://www.who.int/publications/i/item/global-hepatitis-report-2020`.

[13] *A deep learning model for predicting hepatitis C virus-related cirrhosis*. PubMed. Available at: `https://pubmed.ncbi.nlm.nih.gov/33222171/`.

[14] Majzoobi, M. M., Namdar, S., Najafi-Vosough, R., Hajilooi, A. A., & Mahjub, H. (2022). Prediction of hepatitis disease using ensemble learning methods. *Journal of Preventive Medicine and Hygiene*, 63(3), E424-E428. https://doi.org/10.15167/2421-4248/jpmh2022.63.3.2515

[15] Nguyen, T. P., Tran, T. H., & Le, H. T. (2020). *Predicting liver fibrosis in chronic hepatitis B using machine learning techniques. Journal of Hepatology*, 72(1), 123-130.

E Code

```r
#PART1: LOGISTIC REGRESSION, NON LINEAR MODELS AND LASSO
set.seed(142)
#Read database
db<-read.csv(file="hepatitis.csv",header = T, sep = ",", na.strings = "NA", dec = "." )
#Columns with no information will be given NA
db[db == ""] <- NA

#Edit class variable
db$class[db$class=="live"]<- "1"
db$class[db$class=="die"]<- "0"
db$class<- as.numeric(db$class)

#Identify columns of class character
char_cols <- sapply(db, is.character)

#Convert them to factor
db[, char_cols] <- lapply(db[, char_cols], as.factor)
summary(db)


#Handle NAN
#Install mice package
#install.packages("mice")
library(mice)
```

```r

methods <- make.method(db)
#Impute missing values with NAN choose m=3 since our database is not very big
imputed_data <- mice(db, m = 3, method = methods, maxit = 50, seed = 123)
summary(imputed_data)

#Know which methods have been used for each imputation
imputed_data$method
db <- complete(imputed_data)
summary(db)

db$class[db$class=="live"]<- "1"
db$class[db$class=="die"]<- "0"
db$class<- as.numeric(db$class)
plot(db$class,ylab = "")

#Check for outliers in continuous variables
boxplot(db$age)
boxplot(db$bilirubin)
boxplot(db$alk_phosphate)
boxplot(db$sgot)
boxplot(db$albumin)
boxplot(db$protime)


#LASSO REGRESSION

#install.packages("glmnet")
library(glmnet)

# Use colnames()
colnames(db)

#Load neccesary libraries
library(glmnet)

# Prepare data
db$class <- as.factor(db$class)


#Function to compute AIC, BIC and EBIC
calculate_criteria <- function(y, y_hat, coef_matrix, X, gamma = 1) {
  n <- length(y)
  p <- ncol(X)
  rss <- sum((y - y_hat)^2)
  df <- sum(coef_matrix != 0)
  log_lik <- -n/2 * (log(2 * pi) + log(rss/n) + 1)

  #information criterion
  aic <- -2 * log_lik + 2 * df
  bic <- -2 * log_lik + log(n) * df
  ebic <- bic + 2 * gamma * log(p) * df

  return(list(AIC = aic, BIC = bic, EBIC = ebic))
}


#ordinary logictic regression
modelo1 <- glm(class ~ age + sex + steroid + antivirals + fatigue + malaise + anorexia + liver_big +
                 liver_firm + spleen_palpable + spiders + ascites + varices + bilirubin + alk_phosphate +
                 sgot + albumin + protime + histology,
               family = binomial(link = "logit"),
               data = db)
summary(modelo1)

#take into account second order ineractions
model_formula <- as.formula(paste(
  "class ~ (",
  paste(names(db)[-length(names(db))], collapse = " + "),
  ")^2"
))
#ordinary logistic regression with interactions(more parameters than data)
modelo2 <- glm(model_formula, family = binomial(link = "logit"), data = db)
summary(modelo2)

#LASSO REGRESSION
set.seed(142)
X <- model.matrix(model_formula,data=db)[,-1]
#target variable
y <- db$class

#Fit Lasso ith cross validation
```

```r
cv_lasso <- cv.glmnet(X, y, family = "binomial", alpha = 1)
#Check best value for lamda
cv_lasso$lambda.min

#Get coefficients for best lambda value
coef_lasso<-coef(cv_lasso, s = "lambda.min")[,1]


#Create a data frame for coefficients
coef_df <- as.data.frame(as.matrix(coef_lasso))
coef_df$Variable <- rownames(coef_df)

#Filter non cero coeficients
coef_df_non_zero <- coef_df[coef_df$`V1` != 0, ]

#Compute AIC,BIC and EBIC for this model
y<-as.numeric(db$class)-1
fitted_probs <- predict(cv_lasso, newx = X, s = cv_lasso$lambda.min, type = "response")
fitted_probs <- as.numeric(as.character(fitted_probs))
coef_matrix <- as.matrix(coef(cv_lasso, s = cv_lasso$lambda.min))
matrix <- as.matrix(db[, -1])  # Asumiendo que la primera columna es la variable respuesta
criteria<-calculate_criteria(y,fitted_probs,coef_matrix,matrix)
print(criteria)

#ADAPTIVE LASSO
set.seed(142)
coef_initial <- as.vector(coef(cv_lasso, s = "lambda.min"))[-1]
weights <- 1 / abs(coef_initial)
weights[is.infinite(weights)] <- 1e10  #How to handle null coefficients

#Fix adaptive Lasso
adaptive_lasso <- glmnet(X, y, alpha = 1, family = "binomial", penalty.factor = weights)

cv_adaptive_lasso <- cv.glmnet(X, y, alpha = 1, family = "binomial", penalty.factor = weights)

#Final coefficients with best lambda in cross validation
best_lambda <- cv_adaptive_lasso$lambda.min
coef_adaptative<-coef(cv_adaptive_lasso, s = best_lambda)
coef_adaptative_df<- as.data.frame(as.matrix(coef_adaptative))
coef_adaptative_df$Variable <- rownames(coef_adaptative_df)
coef_adaptative_df_non_zero <- coef_adaptative_df[coef_adaptative_df$`s1` != 0, ]

#Compute AIC,BIC and EBIC for this model
y<-as.numeric(db$class)-1
fitted_probs_adapt <- predict(cv_adaptive_lasso, newx = X, s = cv_adaptive_lasso$lambda.min, type = "response")
fitted_probs_adapt <- as.numeric(as.character(fitted_probs_adapt))
coef_matrix_adapt <- as.matrix(coef(cv_adaptive_lasso, s = cv_adaptive_lasso$lambda.min))
matrix_adapt <- as.matrix(db[, -1])  # Asumiendo que la primera columna es la variable respuesta
criteria_adapt<-calculate_criteria(y,fitted_probs_adapt,coef_matrix_adapt,matrix_adapt)
print(criteria_adapt)


################################################################################
################          ACCURACY FOR EACH MODEL          ###################
################################################################################

#AUC and CLASSIFICATION TABLES
#install.packages("pROC")
library(pROC)
roc_el<-roc(db$class,fitted_probs, quiet=TRUE)
roc_el$auc
plot(roc_el,print.thres=TRUE,print.thres.col="dark green")

db$probs<-ifelse (fitted_probs < 0.757, 0, 1)
.Tablew1 <- xtabs(~db$class+db$probs)
.Tablew1
prop.table(.Tablew1,1)

roc_el2<-roc(db$class,fitted_probs_adapt, quiet=TRUE)
roc_el2$auc
plot(roc_el2,print.thres=TRUE,print.thres.col="dark green")
db$probs_adapt<-ifelse (fitted_probs_adapt < 0.665, 0, 1)
.Tablew1 <- xtabs(~db$class+db$probs_adapt)
.Tablew1
prop.table(.Tablew1,1)

#remove previous information about probabilities
colnames(db)
db<-db[,-c(21,22,23)]


#Model Inference
```

```r
#Install mombf packag for model selection and sparseMatrixStats as it is a necessary dependance
install.packages("sparseMatrixStats")
install.packages("mombf")

#load the packages into environment
library("sparseMatrixStats")
library("mombf")


#search the model with no interactions and best under BIC criterion:
fitbic = bestBIC((class ~ age + sex + steroid + antivirals + fatigue + malaise + anorexia + liver_big +
                    liver_firm + spleen_palpable + spiders + ascites + varices + bilirubin + alk_phosphate +
                    sgot + albumin + protime + histology),data=db, family='binomial')

fitbic

#display the best model obtained
summary(fitbic)

#search the model with no interactions and best under EBIC criterion:
fitebic = bestEBIC((class ~ age + sex + steroid + antivirals + fatigue + malaise + anorexia + liver_big +
                      liver_firm + spleen_palpable + spiders + ascites + varices + bilirubin + alk_phosphate +
                      sgot + albumin + protime + histology),data=db, family='binomial')

fitebic

#display the best model obtained
summary(fitebic)


# BMA on original covariates

fit1 <- modelSelection((class ~ age + sex + steroid + antivirals + fatigue + malaise + anorexia + liver_big +
                          liver_firm + spleen_palpable + spiders + ascites + varices + bilirubin + alk_phosphate
                              +
                          sgot + albumin + protime + histology),data=db, family='binomial')
coefbma <- coef(fit1)
# Convert matrix to a data frame for easier manipulation
coefbma_df <- as.data.frame(coefbma)

# Sort by the 'margpp' column (descending order)
coefbma_sorted <- coefbma_df[order(coefbma_df$margpp, decreasing = TRUE), ]

# View the sorted matrix
print(coefbma_sorted)


#BAS
library(BAS)
#BAS library is loaded to consider BMA models with hierarchical interactions
#This is that interactions are not included in the model if main variable is not


#First we normalize the data:
#Identify numerical and categorical columns
numerical_cols <- sapply(db, is.numeric)
categorical_cols <- !numerical_cols

#normalize numerical columns
preProc <- preProcess(db[, numerical_cols], method = c("center", "scale"))
db_normalized <- predict(preProc, db[, numerical_cols])

#combine back the db
db_combined <- bind_cols(db[, categorical_cols], as.data.frame(db_normalized))

#Run imposing force.heredity TRUE for hierarchical interactions
#The run took several hours
bas_model<-bas.glm(
  model_formula,
  family = binomial(link = "logit"),
  data=db_combined,
  method = "MCMC",
  force.heredity = TRUE)

# We compute inclusion probabilities
inclusion_probabilities <- bas_model$probne0.MCMC
names <- bas_model$namesx

#store it in a df for easier manipulation
results <- data.frame(
  Variable = names,
```

```
  Prob = as.numeric(inclusion_probabilities)
)
print(results)

#we can check now the highest posterior inclusion probability covariates/interactions
results <- results[order(-results$Prob), ]
print(results)

#we can also check de highest probability model HPM
HPM <- predict(bas_model, estimator = "HPM")
variable.names(HPM)

#Now we check the posterior means for each coefficient

#store coefficients information in a variable
coef_output <- coef(bas_model)

# Extract posterior means and inclusion probabilities
posterior_means <- as.numeric(coef_output$postmean)  # Ensure it's numeric
inclusion_probs <- as.numeric(coef_output$probne0)  # Ensure it's numeric

# Combine into a data frame
coef_table <- data.frame(
  Predictor = coef_output$namesx,  # Predictor names
  PosteriorMean = posterior_means,          # Numeric posterior means
  InclusionProb = inclusion_probs           # Numeric inclusion probabilities
)

# Round numeric columns to 3 decimal places
coef_table <- transform(coef_table,
                        PosteriorMean = round(PosteriorMean, 3),
                        InclusionProb = round(InclusionProb, 3))

# We reorder by inclusion probability
coef_table <- coef_table[order(-coef_table$InclusionProb), ]

# Print the results
print(coef_table)
```

Listing 1: Lasso and model inference Code

```
#PART2: PCA and Clustering

#Read database
db<-read.csv(file="hepatitis.csv",header = T, sep = ",", na.strings = "NA", dec = "." )
#Columns with no information will be given NA
db[db == ""] <- NA

#Edit class variable
db$class[db$class=="live"]<- "1"
db$class[db$class=="die"]<- "0"
db$class<- as.numeric(db$class)

#Identify columns of class character
char_cols <- sapply(db, is.character)

#Convert them to factor
db[, char_cols] <- lapply(db[, char_cols], as.factor)
summary(db)

#Handle NAN
#Install mice package
#install.packages("mice")
library(mice)

methods <- make.method(db)
#Impute missing values with NAN choose m=3 since our database is not very big
imputed_data <- mice(db, m = 3, method = methods, maxit = 50, seed = 123)
summary(imputed_data)

#Know which methods have been used for each imputation
imputed_data$method
db <- complete(imputed_data)
summary(db)

#load Package
#install.packages('dplyr')
library(dplyr)

#PCA
#columns to transform
```

```r
columns_to_transform <- c("steroid", "antivirals", "fatigue", "malaise",
                          "anorexia", "ascites", "spiders",
                          "liver_firm", "liver_big", "varices", "histology","spleen_palpable")

#Get numerical features in order to get dummies
db <- db %>%
  mutate(across(all_of(columns_to_transform),
                ~ as.factor(recode_factor(.x,"False" = "0", "True" = "1"))))
summary(db)
#Not take gender into account
db_numeric <- db[,c(1,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20)]
db_numeric[] <- lapply(db_numeric, function(col) {
  if (is.factor(col)) {
    return(as.numeric(as.character(col)))
  }
  return(col)
})
#Turn into numeric to compute covariance matrix
library(ggplot2)
library(reshape2)

# Calculate the correlation matrix
a <- cor(db_numeric)

# Convert the correlation matrix into long format
a_long <- melt(a)

# Create the heatmap
ggplot(data = a_long, aes(x = Var1, y = Var2, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white", midpoint = 0,
                       limit = c(-1, 1), space = "Lab") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  ggtitle("Covariance Matrix")


# Filter covariances
filtered_cov <- ifelse(abs(a) > 0.3, a, NA)
print(filtered_cov)

library(ggplot2)
library(reshape2)

filtered_cov_long <- melt(filtered_cov, na.rm = TRUE)

#Create heatmap
ggplot(data = filtered_cov_long, aes(x = Var1, y = Var2, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white", midpoint = 0,
                       limit = c(-1, 1), space = "Lab") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))


#Select most correlated variables(clinical variables)
mat<- db_numeric[,c(9,10,11,12,13,14,15,16,17,19)]
y<-mat[,c(10)]
mat1<-mat[,c(1,2,3,4,5,6,7,8,9)]
#Scale variables
mat1<-scale(mat1)
mat1 <- as.data.frame(mat1)
sapply(mat, class)

##Examine normality of numerical variables
library(nortest)

shapiro.test(mat1$bilirubin)
shapiro.test(mat1$alk_phosphate)
shapiro.test(mat1$sgot)
shapiro.test(mat1$albumin)
shapiro.test(mat1$protime)
shapiro.test(mat1$spleen_palpable)
shapiro.test(mat1$spiders)
shapiro.test(mat1$ascites)
shapiro.test(mat1$varices)
#none of them is normal. so we can not do probabilistic pca

#Compute correlation matrix
cor1<- cor(mat1)

#get eigenvalues of correlation matrix
```

```r
eigen(cor1)
eig <- eigen(cor1)
eigenvalues <- eig$values

#Plot to see the elbow of the curve
ggplot(data = data.frame(x = 1:length(eigenvalues), y = eigenvalues), aes(x = x, y = y)) +
  geom_point(color = "blue", size = 3) +
  geom_line(color = "blue", linewidth = 1) +
  labs(
    title = "Scree Plot",
    x = "Number of Components",
    y = "Eigenvalue"
  ) +
  theme_minimal()
#The elbow takes place when we have only one eigenvalue. As said before, it is senseless so we take three.
library(psych)
#PCA with the number of components specified to check the variance explained
pca_result <- prcomp(mat1, scale. = TRUE)
#Plot the variance explained by each component
explained_variance <- summary(pca_result)$importance[2, ]
explained_variance

# Create a data frame to explain it
variance_df <- data.frame(
  Component = 1:length(explained_variance),
  Variance_Explained = explained_variance
)
#Plot the results
ggplot(variance_df, aes(x = as.factor(Component), y = Variance_Explained)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  geom_line(aes(group = 1, y = Variance_Explained), color = "red", linewidth = 1) +
  geom_point(color = "red", size = 2) +
  theme_minimal() +
  labs(title = "Proportion of Explained Variance",
       x = "Principal Component",
       y = "Explained Variance") +
  scale_x_discrete(labels = as.character(variance_df$Component))


#loadings
loadings<-pca_result$rotation
loadings[,1:3]
#scores
pca_rotated_scores <- pca_result$x[, 1:3]

#Save loadings in a data frame
loadings_df <- as.data.frame(loadings[,1:3])
loadings_df$Variable <- rownames(loadings_df)

#plot the results
library(tidyr)
loadings_long <- gather(loadings_df, key = "Component", value = "Loading", -Variable)
#Barplot per each component
ggplot(loadings_long, aes(x = Variable, y = Loading, fill = Component)) +
  geom_bar(stat = "identity", position = "dodge") +
  coord_flip() +
  theme_minimal() +
  labs(title = "PCA Loadings",
       x = "Variables",
       y = "Loadings") +
  scale_fill_brewer(palette = "Set2")

#Save each component for cluster analysis
mat1$PC1 <- pca_rotated_scores[,1]
mat1$PC2 <- pca_rotated_scores[,2]
mat1$PC3 <- pca_rotated_scores[,3]
mat1$class<-y

#plot the results
library(car)
# Set up the plotting area for three plots in one row
par(mfrow = c(1, 3), mar = c(5, 5, 3, 1))

# Plot 1: PC2 vs PC1
plot(PC2~ PC1, data = subset(mat1, class == "1"),
     col = "blue", pch = 16, cex = 1.5,
     xlab = "PC1", ylab = "PC2",
     main = "PC2 vs PC1",
     xlim = c(min(mat1$PC1) - 1, max(mat1$PC1) + 1),
     ylim = c(min(mat1$PC2) - 1, max(mat1$PC2) + 1),
     cex.lab = 1.2, cex.main = 1.5, cex.axis = 1.1)

```

```r
# Add points for the second class (Death)
points(PC2 ~ PC1, data = subset(mat1, class == "0"),
       col = "red", pch = 17, cex = 1.5)  # Red color, triangle shape, and larger points

# Add reference lines at 0
abline(h = 0, v = 0, col = "gray", lty = 2, lwd = 1)

# Customize the legend
legend("topright", legend = c("Alive", "Death"),
       pch = c(16, 17), col = c("blue", "red"),
       cex = 1.2, bg = "white", box.lwd = 1.5)


# Plot 2: PC3 vs PC1
plot(PC3 ~ PC1, data = subset(mat1, class == "1"),
     col = "blue", pch = 16, cex = 1.5,
     xlab = "PC1", ylab = "PC3",
     main = "PC3 vs PC1",
     xlim = c(min(mat1$PC1) - 1, max(mat1$PC1) + 1),
     ylim = c(min(mat1$PC3) - 1, max(mat1$PC3) + 1),
     cex.lab = 1.2, cex.main = 1.5, cex.axis = 1.1)

# Add points for the second class (Death)
points(PC3 ~ PC1, data = subset(mat1, class == "0"),
       col = "red", pch = 17, cex = 1.5)

# Add reference lines at 0
abline(h = 0, v = 0, col = "gray", lty = 2, lwd = 1)

# Customize the legend
legend("topright", legend = c("Alive", "Death"),
       pch = c(16, 17), col = c("blue", "red"),
       cex = 1.2, bg = "white", box.lwd = 1.5)


# Plot 3: PC3 vs PC2
plot(PC3 ~ PC2, data = subset(mat1, class == "1"),
     col = "blue", pch = 16, cex = 1.5,  #
     xlab = "PC2", ylab = "PC3",
     main = "PC3 vs PC2",
     xlim = c(min(mat1$PC2) - 1, max(mat1$PC2) + 1),
     ylim = c(min(mat1$PC3) - 1, max(mat1$PC3) + 1),
     cex.lab = 1.2, cex.main = 1.5, cex.axis = 1.1)

# Add points for the second class (Death)
points(PC3 ~ PC2, data = subset(mat1, class == "0"),
       col = "red", pch = 17, cex = 1.5)

# Add reference lines at 0
abline(h = 0, v = 0, col = "gray", lty = 2, lwd = 1)

# Customize the legend
legend("topright", legend = c("Alive", "Death"),
       pch = c(16, 17), col = c("blue", "red"),
       cex = 1.2, bg = "white", box.lwd = 1.5)

# Reset the plot layout to default
par(mfrow = c(1, 1))


#CLUSTERING
mat2<-mat1
datuak <- mat2[,-c(1:9)]  # Remove unnecessary columns from mat2
datuak2 <- mat2[,-c(1:9)] #Create a copy for second analysis
# Get the distance matrix with Euclidean distance (squared)
distmatrix <- dist(datuak, method = "euclidean")
distmatrix2 <- distmatrix^2
distmatrix2  # Display the squared distance matrix
#set seed for reproducibility
set.seed(123)

# Use Ward method for hierarchical clustering based on the factors F1 and F2
HClust.1 <- hclust(dist(model.matrix(~-1 + PC1 + PC2+ PC3, datuak))^2, method = "ward.D")

# Plot dendrogram
plot(HClust.1, main = "Cluster Dendrogram", xlab = "Individual number in the dataset", ylab = "", sub = "")

# Set 4 clusters from the dendrogram
# Improved hierarchical clustering plot

# Set plot size and margins
par(mar = c(5, 4, 4, 2) + 0.1)
```

```r

# Create the basic dendrogram plot
plot(HClust.1,
     main = "Hierarchical Clustering: 4 Clusters",  # Title
     xlab ="",    # X-axis label
     ylab = "Height",                              # Y-axis label
     sub = "",                                     # Subtitle (leave empty)
     col.main = "darkblue",                        # Title color
     col.lab = "darkgreen",                        # Axis labels color
     cex.main = 1.5,                               # Title size
     cex.lab = 1.2,                                # Axis labels size
     cex.axis = 1.1,                               # Axis tick labels size
     font.lab = 2,                                 # Bold axis labels
     font.main = 2,                                # Bold title
     lwd = 2,                                      # Line width for dendrogram
     hang = -1,                                    # Make the labels aligned at the bottom
     col.axis = "black")                           # Color for axis tick labels

# Add colored rectangle around the 4 clusters
rect.hclust(HClust.1, k = 4, border = "blue")

# Reset plotting parameters (optional)
par(mfrow = c(1, 1))


# Print the number of individuals in each cluster
summary(as.factor(cutree(HClust.1, k =4 )))

# Calculate Cluster Centroids
by(model.matrix(~-1 + PC1 + PC2+ PC3, datuak), as.factor(cutree(HClust.1, k = 4)), colMeans)


# Graphical representation
# Calculate Cluster Centroids
centroids <- by(model.matrix(~-1 + PC1 + PC2 + PC3, datuak),
                as.factor(cutree(HClust.1, k = 4)),
                colMeans)

# Ensure centroids is a list of numeric vectors
centroids <- lapply(centroids, unlist)  # Convert each element to a vector if needed

# Combine them into a data frame
centroids_df <- do.call(rbind, centroids)

# Convert the result to a data frame if necessary
centroids_df <- as.data.frame(centroids_df)

# Assign column names
colnames(centroids_df) <- c("PC1", "PC2", "PC3")

# Add cluster identifiers (assuming there are 4 clusters)
centroids_df$cluster <- factor(1:4)  # Add cluster identifiers

# Plotly 3D scatter plot
library(plotly)

# Get the cluster assignments
clusters <- cutree(HClust.1, k = 4)

# Create the 3D scatter plot for the data points
fig <- plot_ly(
  x = mat1$PC1,  # First principal component (PC1)
  y = mat1$PC2,  # Second principal component (PC2)
  z = mat1$PC3,  # Third principal component (PC3)
  color = factor(clusters),  # Cluster colors
  colors = c("blue", "red1", "green1", "purple1"),  # Custom colors for clusters
  type = "scatter3d",  # 3D scatter plot
  mode = "markers",  # Display points as markers
  marker = list(size = 5, opacity = 0.7)  # Point size and opacity for data points
)

# Add centroids to the plot as a separate trace with larger and distinct markers
fig <- fig %>% add_trace(
  x = centroids_df$PC1,  # Centroid positions for PC1
  y = centroids_df$PC2,  # Centroid positions for PC2
  z = centroids_df$PC3,  # Centroid positions for PC3
  color = factor(centroids_df$cluster),  # Color the centroids by cluster
  colors = c("blue4", "red4", "green4", "purple4"),  # Matching centroid colors
  type = "scatter3d",
  mode = "markers+text",  # Display markers and text labels
  text = paste("Cluster", centroids_df$cluster),  # Cluster label
  marker = list(
```

```
    size = 10,  # Larger marker size for centroids
    symbol = "x",  # Use "x" symbol for centroids
    line = list(width = 3),  # Thicker border for centroid markers
    opacity = 2  # Full opacity for centroids
  ),
  showlegend = FALSE  # Hide the legend for centroids
)

# Customize layout
fig <- fig %>% layout(
  title = "3D Plot of Principal Components with Centroids",
  scene = list(
    xaxis = list(title = "PC1"),
    yaxis = list(title = "PC2"),
    zaxis = list(title = "PC3")
  )
)

# Show the plot
fig



# Add the cluster assignment to the 'datuak' dataframe
datuak$hclus.label <- cutree(HClust.1, k = 4)

# Check if the new cluster labels have been added
head(datuak)


###########
# K-means #
###########

#ELBOW METHOD

# Create the feature matrix
data_matrix <- model.matrix(~-1 + PC1 + PC2 +PC3, datuak2)

# Test k-means for different numbers of clusters
set.seed(123)  # Ensures reproducibility

total_withinss <- numeric()

# Iterate
for (k in 2:10) {
  kmeans_result <- kmeans(data_matrix, centers = k, iter.max = 10)  # Ejecuta k-means para cada k
  total_withinss[k] <- kmeans_result$tot.withinss  # Guarda el within-cluster sum of squares
}
# Create a data frame to store k values and the corresponding total within-cluster sum of squares
elbow_data <- data.frame(
  k = 2:10,
  total_withinss = total_withinss[2:10]
)

# Create the elbow plot with ggplot2
ggplot(elbow_data, aes(x = k, y = total_withinss)) +
  geom_point(color = "darkblue", size = 3) +  # Points for each k
  geom_line(color = "blue", linewidth = 1) +  # Line connecting the points
  geom_smooth(method = "loess", color = "red", linetype = "dashed", size = 1) +  # Smoothed line
  labs(
    title = "Elbow Method for Determining Optimal k",
    x = "Number of Clusters",
    y = "Total Within-Cluster Sum of Squares"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    plot.title = element_text(hjust = 0.5, size = 18, face = "bold"),  # Center title, larger font
    axis.title = element_text(size = 14),  # Axis title font size
    axis.text = element_text(size = 12),  # Axis labels font size
    plot.margin = margin(20, 20, 20, 20)  # Increase plot margins for readability
  ) +
  scale_x_continuous(breaks = 2:10)  # Ensure x-axis ticks correspond to the


#GAP STATISTIC
# Load necessary library
if (!require(cluster)) install.packages("cluster")
library(cluster)

data_matrix <- model.matrix(~-1 + PC1 + PC2+ PC3, datuak2)

```

```r
# Set parameters for the gap statistic
set.seed(123)  # For reproducibility
max_clusters <- 10  # Maximum number of clusters to test

# Compute the gap statistic
gap_stat <- clusGap(data_matrix,
                    FUN = kmeans,
                    K.max = max_clusters,
                    B = 50,  # Number of bootstrap samples for reference data
                    iter.max = 10)

# Display the gap statistic results
print(gap_stat)

# Improved gap statistic plot
plot(gap_stat,
     main = "Gap Statistic for Optimal Number of Clusters",   # Title with a larger font
     xlab = "Number of Clusters",                             # X-axis label
     ylab = "Gap Statistic",                                  # Y-axis label
     col = "blue",                                           # Line color
     lwd = 2,                                                # Line width for better visibility
     pch = 16,                                               # Point type (filled circles)
     cex = 1.2,                                              # Point size
     col.main = "darkblue",                                   # Title color
     col.lab = "darkgreen",                                   # Axis label color
     cex.main = 1.5,                                          # Title size
     cex.lab = 1.2,                                           # Axis labels size
     cex.axis = 1.1,                                          # Axis tick labels size
     font.lab = 2,                                            # Bold axis labels
     font.main = 2,                                           # Bold title
     xlim = c(1, length(gap_stat$Tab[,1]))                    # Ensure the x-axis includes all clusters
)

# Add gridlines for better readability
grid(col = "gray", lty = "dotted")

#Add horizontal line at y = 0 to better highlight the gaps
abline(h = 0, col = "red", lty = 2)


#The maximum gap is observed at k=4 with a value of 0.5739

#K-MEANS IMPLEMENTATION
#Do 4 clusters
set.seed(142)
kmeans_result <- kmeans(model.matrix(~-1 +PC1 + PC2+ PC3, datuak2), centers = 4, iter.max = 1000000)

#Cluster sizes
kmeans_result$size

# Calculate Cluster Centroids from kmeans_result$centers
centroids2_df <- as.data.frame(kmeans_result$centers)

# Assign column names based on your principal components
colnames(centroids2_df) <- c("PC1", "PC2", "PC3")

# Add cluster identifiers
centroids2_df$cluster <- factor(1:nrow(centroids2_df))

# View the centroids data frame
print(centroids2_df)


#inerties
kmeans_result$withinss # Within Cluster Sum of Squares
kmeans_result$tot.withinss # Total Within Sum of Squares
kmeans_result$betweenss # Between Cluster Sum of Squares

# Add cluster assignments to the original dataset
datuak2$cluster <- kmeans_result$cluster

assignCluster <- function(kmeans_result, data) {
  data$cluster <- kmeans_result$cluster
  return(data)
}
assignCluster(kmeans_result, datuak2)
# Use the function
datuak2 <- assignCluster(kmeans_result, datuak2)

# 3D Plot with Plotly
library(plotly)

```

```r
#Plot with colours each individual
fig <- plot_ly(
  data = datuak2,
  x = ~PC1,
  y = ~PC2,
  z = ~PC3,
  type = "scatter3d",
  mode = "markers",
  marker = list(size = 5),
  color = ~factor(cluster),  # Mapear clusters como factores
  colors = c("blue", "red", "green", "purple"),  # Definir colores para 4 clusters
  showlegend = TRUE  # Asegurar que la leyenda est  visible
)

#Add centroids
fig <- fig %>% add_trace(
  data = centroids2_df,
  x = ~PC1,
  y = ~PC2,
  z = ~PC3,
  type = "scatter3d",
  mode = "markers+text",
  text = paste("Centroid", centroids2_df$cluster),
  marker = list(
    size = 10,
    symbol = "x",
    line = list(width = 3),
    color = ~factor(cluster),
    colors = c("blue", "red", "green", "purple")
  ),
  showlegend = FALSE
)

#Legend
fig <- fig %>% layout(
  title = "3D K-Means Clustering with Centroids (4 Clusters)",
  scene = list(
    xaxis = list(title = "PC1"),
    yaxis = list(title = "PC2"),
    zaxis = list(title = "PC3")
  )
)

# Mostrar el gr fico
fig



#FINAL INTERPRETATION#
#Add the corresponding variable
datuak$class<-y
datuak2$class<-y
library(ggplot2)

#For K-MEANS
datuak2$survived=datuak2$class==1
# Calculate the proportion of survivors within each cluster
survival_proportion <- tapply(datuak2$survived, datuak2$cluster, function(x) mean(x, na.rm = TRUE))

# Plot the survival proportions using a bar plot
barplot(survival_proportion,
        names.arg = paste("Cluster", 1:length(survival_proportion)),
        col = "blue",
        main = "Proportion of Survivors in Each Cluster",
        xlab = "Cluster",
        ylab = "Proportion Survived",
        border = "black")

# Add text labels to the bars showing the proportion
text(x = seq_along(survival_proportion),
     y = survival_proportion + 0.05,
     labels = round(survival_proportion, 2),
     col = "black")

#Using ggplot2 for better visualization
survival_data <- data.frame(
  Cluster = factor(1:length(survival_proportion)),  # Cluster labels
  Proportion = survival_proportion  # Survival proportions
)

# Plot using ggplot2
ggplot(survival_data, aes(x = Cluster, y = Proportion)) +
```

```r
  geom_bar(stat = "identity", fill = "blue", color = "black") +
  geom_text(aes(label = round(Proportion, 2)), vjust = -0.5, color = "black") +
  labs(title = "Proportion of Survivors in Each Cluster",
       x = "Cluster",
       y = "Proportion Survived") +
  theme_minimal()


#For Ward's method
datuak$survived=datuak$class==1
# Calculate the proportion of survivors within each cluster
survival_proportion <- tapply(datuak$survived, datuak$hclus.label, function(x) mean(x, na.rm = TRUE))

# Plot the survival proportions using a bar plot
barplot(survival_proportion,
        names.arg = paste("Cluster", 1:length(survival_proportion)),
        col = "blue",
        main = "Proportion of Survivors in Each Cluster",
        xlab = "Cluster",
        ylab = "Proportion Survived",
        border = "black")

# Add text labels to the bars showing the proportion
text(x = seq_along(survival_proportion),
     y = survival_proportion + 0.05,  # Position the text slightly above the bars
     labels = round(survival_proportion, 2),
     col = "black")

# Using ggplot2 for better visualization
survival_data <- data.frame(
  Cluster = factor(1:length(survival_proportion)),  # Cluster labels
  Proportion = survival_proportion  # Survival proportions
)

# Plot using ggplot2
ggplot(survival_data, aes(x = Cluster, y = Proportion)) +
  geom_bar(stat = "identity", fill = "blue", color = "black") +
  geom_text(aes(label = round(Proportion, 2)), vjust = -0.5, color = "black") +
  labs(title = "Proportion of Survivors in Each Cluster",
       x = "Cluster",
       y = "Proportion Survived") +
  theme_minimal()
```

Listing 2: PCA and Clustering Code