Instituto Superior Técnico

Computational Statistics 2023

# Generalized Linear Models and Markov Chain Monte Carlo Method

**Report**


| **Name** | **IST Number** |
|---|---|
| Davi Giordano Valério | 108497 |
| Lenka Vincenova | 108911 |


Lisbon
2023

INSTITUTO SUPERIOR TÉCNICO

# Generalized Linear Models and Markov Chain Monte Carlo Method

**Report**

Report of the project to prof. Giovani Loiola da Silva during the Computational Statistics Course, at Instituto Superior Técnico, in Lisbon, Portugal.

Lisbon, Portugal
2023

# Contents

# List of Figures

# 1. Introduction

## 1.1 Context

Infections acquired in healthcare settings, known as healthcare-associated infections (HAI) or nosocomial, are those contracted during medical treatment that were not present when the patient was admitted. [1]

This project aims to analyze data from a study in this domain. In 1975, 338 hospitals in the United States went through infection surveillance and control programs to assess the rates of nosocomial infection. During the study, several metrics were recorded, and their infection risk was estimated, as seen in table 1.1.

| Variable name | Description |
|---|---|
| Length of stay | Average length of stay of all patients in hospital (days) |
| Age | Average age of patients (years) |
| Routine culturing ratio | Ratio of number of cultures performed to number of patients without signs or symptoms of hospital-acquired infection |
| Routine chest X-ray ratio | Ratio of number of X-rays performed to number of patients without signs or symptoms of pneumonia |
| Number of beds | Average number of beds in hospital during study period |
| Medical school affiliation | If the hospital was affiliated to a medical school or not |
| Region | Geographic region (NE, NC, S, W) |
| Average daily census | Average number of patients in hospital per day during study period |
| Number of nurses | Average number of full-time equivalent registered and licensed practical nurses during study period (number full-time plus one half the number part time) |
| Available facilities and services | Percent of 35 potential facilities and services that are provided by the hospital |
| Infection risk | Average estimated probability of acquiring infection in hospital (%) |

Table 1.1: Table of recorded metrics during the study

From this study, a random sample of 113 hospitals was taken and statistically ana-

lyzed. With the data, the goal of this project is to construct a Generalized Linear Model by means of Markov Chain Monte Carlo methods with the goal of predicting the infection risk of hospitals. Inn this project, the data are considered independent and so are the variables of the study.

The software, package versions, and the code used in this project can be visualized in the Appendix.

## 1.2 Generalized Linear Models

Generalized Linear Models generalize the ordinary linear regression by allowing the covariates to be related to the predicted variable through a "link function". This enables its use when the response variable $Y$ does not follow a Normal distribution. Instead, $Y$ could have been generated from any distribution of the exponential family.

In order to fit a generalized linear model to a sample with a bayesian approach, one must choose the prior distribution for the response variable $Y$, the prior distributions of the parameters $\beta$ that multiply the covariates $\mathbf{X}$ and the link function $g$. With this setup, it is then possible to estimate the posterior distributions of the parameters with methods such as the Markov Chain Monte Carlo. Finally, the posterior predictive distribution of $Y$ can be calculated, and estimation statistics can be performed.

## 1.3 Markov Chain Monte Carlo Methods

Markov Chain Monte Carlo methods provide algorithms for sampling from a distribution. This can be useful when calculating the posterior distribution of the parameters $\boldsymbol{\beta}$ is not viable. The method starts with a random guess for the parameters, based on the priors, and it iteratively samples new parameters from a proposal distribution. In the Metropolis-Hastings algorithm, the new sample for the betas can be accepted or not, depending on the likelihood of the data under the new sample and on its probability, given the priors. If the iterations converge, the distribution of the sampled parameters approximates the true posterior distributions.

So as to assess the non convergence of the chains, several statistics can be used. One example is the Gelman-Rubin metric, which analyzes the chain's variance and provides a

ratio that can be used to indicate convergence. In addition, visualizing the trace plot of the chain can also be helpful in this analysis.

Finally, it is common to have multiple models learned, and it becomes necessary to select the best one. For this, the Widely Applicable Information Criterion (WAIC) can be used. This metric allows for model comparison by rewarding goodness of fit and penalizing the number of parameters in the model, which aims to prevent overfitting.

# 2. Methods

## 2.1 Exploratory Analysis

First, the data was analyzed through a classical approach. To visualize the distribution of the variables, boxplots and histograms were plotted for each variable. These visualizations informed the definition of prior distributions for both the covariates and the response variable.

Further, scatterplots were created with covariates on the x-axis and infection risk on the y-axis, revealing the nature of their relationships. Variables that showed a logarithmic relationship with the infection risk were transformed with the application of a natural logarithm function. Subsequently, a correlation matrix was calculated, and the variables that were correlated to the infection risk were identified.

Lastly, a Linear Regression model using the Least Squares Method was developed for a comparative analysis with the Bayesian approach. The model was first fitted using all the data. Then, the examples that had outlier values in *infection risk* were removed and the model was re-fitted. It was then possible to plot the distribution of the observed values of the target variable against the predicted ones according to the model. Finally, these charts were used to evaluate the results of the Linear Regression.

## 2.2 Generalized Linear Regression

For the Generalized Linear Regression (GLM), three distinct prior distributions (Normal, Student's t, and Cauchy) were evaluated for the target variable $Y$. In each case, the priors for the parameters were defined. Normal distributions were assigned to parameters that could take on negative values. Conversely, Half-Normal distributions were used for parameters limited to positive values.

For all the distributions, the data was normalized and no entry was removed. Then, for each one, six distinct models with a varying number of variables were tested. The first model included all ten variables from the dataset. Subsequent models were more selective, using only the top eight, five, four, three, and two variables, respectively, based on their correlation with infection risk. The variables used in each model can be seen in

Table 2.1, where "1" indicates that the variable was used in the model and "0" means that it was not.

| Model | Complete | Top 8 | Top 5 | Top 4 | Top 3 | Top 2 |
|---|---|---|---|---|---|---|
| routine_culturing_ratio | 1 | 1 | 1 | 1 | 1 | 1 |
| ln_num_nurses | 1 | 1 | 1 | 1 | 1 | 1 |
| lenght_of_stay | 1 | 1 | 1 | 1 | 1 | 0 |
| ln_avg_census | 1 | 1 | 1 | 1 | 0 | 0 |
| routine_xray_ratio | 1 | 1 | 1 | 0 | 0 | 0 |
| ln_num_beds | 1 | 1 | 0 | 0 | 0 | 0 |
| avelbl_services | 1 | 1 | 0 | 0 | 0 | 0 |
| med_school_affil | 1 | 1 | 0 | 0 | 0 | 0 |
| region | 1 | 0 | 0 | 0 | 0 | 0 |
| age | 1 | 0 | 0 | 0 | 0 | 0 |

Table 2.1: Variables used in each model

In each case, the six models were compared with the WAIC metric, and the best models were selected based on this metric and their complexity. Then, the posterior distribution of each parameter was plotted, and their high density credible intervals were calculated. In addition, convergence analysis was performed. Lastly, the posterior predictive was sampled and compared to the observed data. With this result, it was possible to select which prior distribution for Y led to the best result and to select the final model.

# 3. Results

## 3.1 Exploratory Data Analysis

### 3.1.1 Variable study

The exploratory data analysis (EDA) was conducted to obtain an overview of the dataset and to achieve a deeper understanding of the features and their relationships. The dataset has 10 features, 1 target variable (*infection risk*), and it comprises of 113 observations. It is complete, so data imputation was unnecessary. The feature variables contain 2 categorical variables: *region* and *medical school affiliation*, which were one-hot encoded. The remaining 8 features are continuous variables. The descriptive statistics of every continuous variable can be seen in Table 3.1. This table also includes the statistics for the 4 logarithmically transformed variables, that are labelled with a prefix `ln_`.

| | count | mean | std | min | Q1 | median | Q3 | max |
|---|---|---|---|---|---|---|---|---|
| **length_of_stay** | 113 | 9.6 | 1.9 | 6.7 | 8.3 | 9.4 | 10.5 | 19.6 |
| **age** | 113 | 53.2 | 4.5 | 38.8 | 50.9 | 53.2 | 56.2 | 65.9 |
| **infection_risk** | 113 | 4.4 | 1.3 | 1.3 | 3.7 | 4.4 | 5.2 | 7.8 |
| **routine_culturing_ratio** | 113 | 15.8 | 10.2 | 1.6 | 8.4 | 14.1 | 20.3 | 60.5 |
| **routine_xray_ratio** | 113 | 81.6 | 19.4 | 39.6 | 69.5 | 82.3 | 94.1 | 133.5 |
| **num_beds** | 113 | 252.2 | 192.8 | 29.0 | 106.0 | 186.0 | 312.0 | 835.0 |
| **med_school_affil** | 113 | 1.8 | 0.4 | 1.0 | 2.0 | 2.0 | 2.0 | 2.0 |
| **region** | 113 | 2.4 | 1.0 | 1.0 | 2.0 | 2.0 | 3.0 | 4.0 |
| **avg_census** | 113 | 191.4 | 153.8 | 20.0 | 68.0 | 143.0 | 252.0 | 791.0 |
| **num_nurses** | 113 | 173.2 | 139.3 | 14.0 | 66.0 | 132.0 | 218.0 | 656.0 |
| **avelbl_services** | 113 | 43.2 | 15.2 | 5.7 | 31.4 | 42.9 | 54.3 | 80.0 |
| **ln_num_beds** | 113 | 5.3 | 0.7 | 3.4 | 4.7 | 5.2 | 5.7 | 6.7 |
| **ln_avg_census** | 113 | 4.9 | 0.8 | 3.0 | 4.2 | 5.0 | 5.5 | 6.7 |
| **ln_num_nurses** | 113 | 4.8 | 0.8 | 2.6 | 4.2 | 4.9 | 5.4 | 6.5 |

Table 3.1: Descriptive Statistics

The Figure 3.1 shows the boxplot and the histogram of the target variable. The histogram resembles a gaussian distribution. Thus, the Gaussian, Cauchy and Student's t were used to model it in the Generalized Linear Regression, since the three have similar shapes. It can be also observed from the boxplot that there are 6 outliers in the response variable. They can be identified by the hospital identification number.

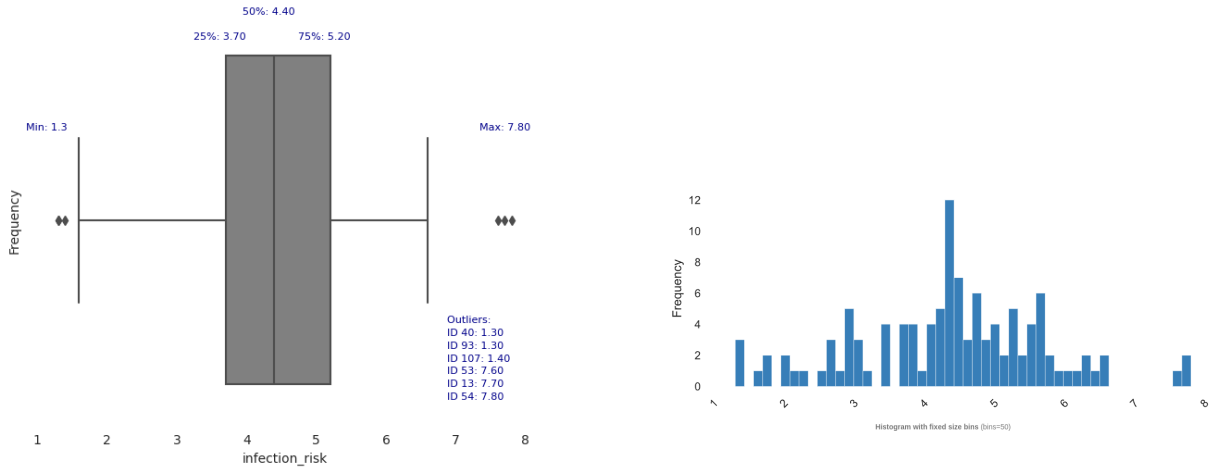
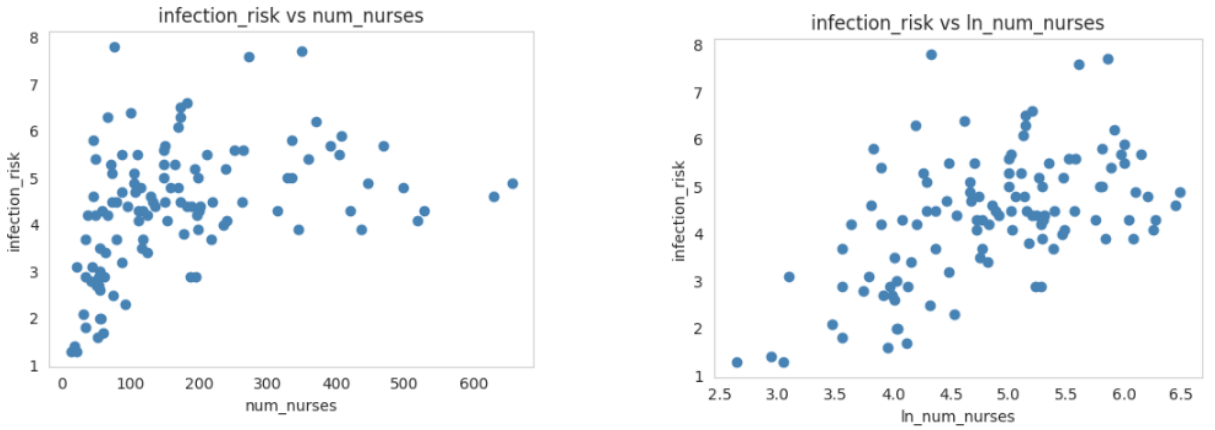



Figura 3.1: The boxplot and histogram graphs for the target variable *infection risk*

Next, scatterplots were used to inspect the shape of each feature data against the target variable. The features *lenght of stay* and *culturing ration* show a tendency to a linear relationship to the *infection risk*. The features *number of beds*, *number of nurses*, and *average census* resembled a logarithmic shape when plotted against the target variable. Thus, they were transformed with the application of the natural logarithm function, and their initial versions were dropped. As an illustration, the scatter plot of *number of nurses* and of it's transformed version can be seen in the figures 3.2a and 3.2b,




(a) Original variable

(b) Transformed variable

Figura 3.2: Scatterplot of infection risk and number of nurses

Finally, a correlation matrix was generated to visually represent the correlations between the variables. The matrix shows Pearson correlation values ordered by *infection risk* in both axis and is presented in Figure 3.3. From this matrix, it can be observed that *routine culturing ratio* is the variable that has the highest linear correlation with *infection risk*, followed by the transformed *ln number of nurses* and *length of stay*.

It is also worth noting that the variables transformed through the application of the natural logarithm showed a higher correlation to *infection risk* than their initial versions. Specifically, the correlation of *number of nurses* is 0.39 but the transformed variable has a value of 0.55. The *average census* improved from 0.38 to 0.46, and *number of beds* improved from 0.36 to 0.45 after the log transformation.

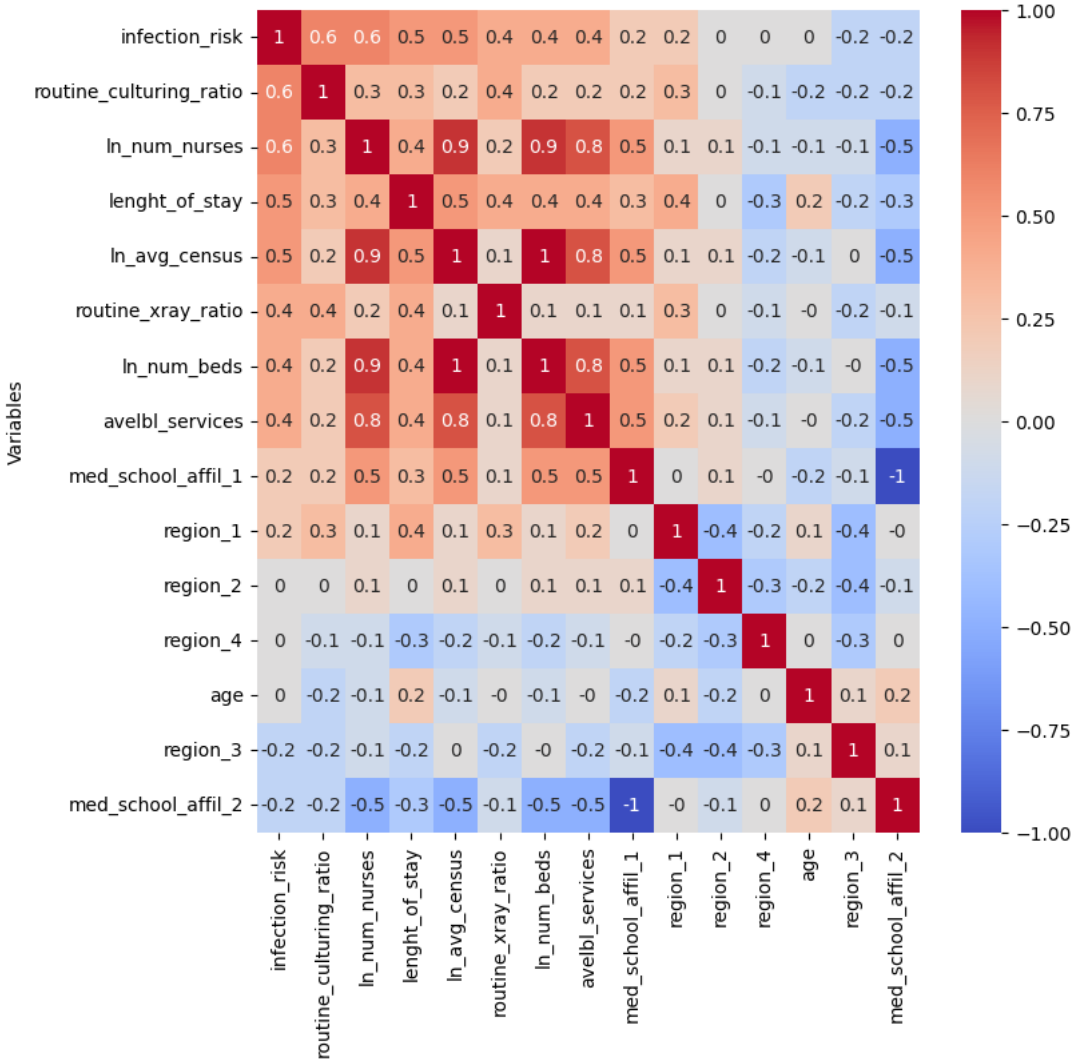
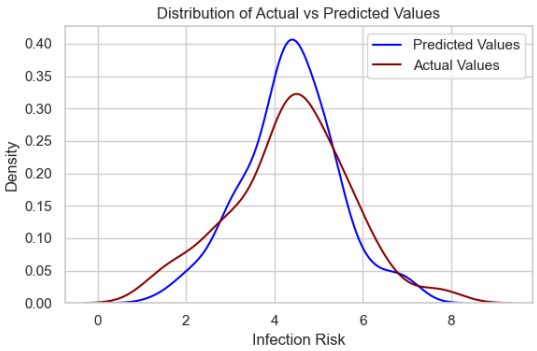


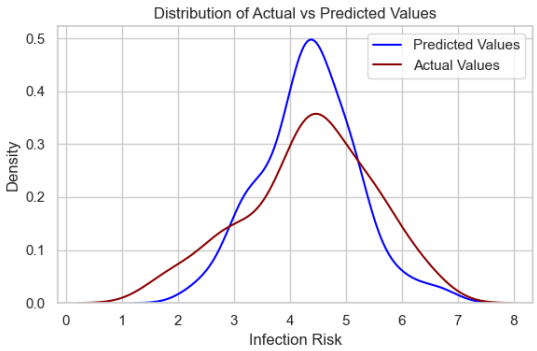

Figura 3.3: Correlation matrix depicting the linear correlation between individual features

### 3.1.2 Linear Regression Model

To identify a base model for comparison with our GLM models, two versions of a linear model were fitted using the Least Squares Method (LSM). Initially, all available data was used. Next, the rows with outliers in the target variable were removed and the model was refitted. The true and predicted densities are depicted in Figures 3.4a and 3.4b and their coefficients are shown in Table 3.2.




(a) Predicted vs true KDE plot of the linear model fitted with all data




(b) Predicted vs true KDE plot of the linear model fitted without outliers

## 3.2 Generalized Linear Regression

In the Generalized Linear Regression, the target variable was modelled using Cauchy, Normal and Student's t distributions. For each case, six different models were fitted, according to Table 2.1. The Tables 3.3, 3.4, and 3.5 show the WAIC results of the variations of each model, respectively. The rank column represents which model performed the best according to expected log pointwise predictive density (elpd), while the $p$ column represents complexity of the model. With these results, it was possible to choose one model from each distribution for further analysis.

According to the WAIC results from the Table 3.3, the selected model for further analysis for the Cauchy distribution is the one with 3 variables.

| **Coefficient** | **LSM** | **LSM without outliers** |
|---|---|---|
| $\alpha$ | 4.35 | 4.33 |
| $\beta_{\text{lenght\_of\_stay}}$ | 0.39 | 0.33 |
| $\beta_{\text{age}}$ | 0.13 | 0.12 |
| $\beta_{\text{routine\_culturing\_ratio}}$ | 0.52 | 0.42 |
| $\beta_{\text{routine\_xray\_ratio}}$ | 0.20 | 0.18 |
| $\beta_{\text{avelbl\_services}}$ | -0.09 | -0.09 |
| $\beta_{\text{ln\_num\_nurses}}$ | 0.68 | 0.56 |
| $\beta_{\text{ln\_num\_beds}}$ | -0.54 | -0.80 |
| $\beta_{\text{ln\_avg\_census}}$ | 0.49 | 0.82 |
| $\beta_{\text{med\_school\_affil\_1}}$ | -0.10 | -0.07 |
| $\beta_{\text{med\_school\_affil\_2}}$ | 0.10 | 0.07 |
| $\beta_{\text{region\_1}}$ | -0.15 | -0.11 |
| $\beta_{\text{region\_2}}$ | 0.01 | 0.01 |
| $\beta_{\text{region\_3}}$ | -0.05 | -0.09 |
| $\beta_{\text{region\_4}}$ | 0.24 | 0.24 |

Table 3.2: Regression Coefficients

| | rank | elpd | p |
|---|---|---|---|
| df_top3 | 0 | -166.2 | 6 |
| df_top5 | 1 | -166.8 | 10.2 |
| df_top4 | 2 | -167.8 | 9.4 |
| df_complete | 3 | -168.1 | 25.3 |
| df_top8 | 4 | -171.2 | 18.1 |
| df_top2 | 5 | -174.4 | 4.4 |

Table 3.3: The WAIC results for *infection risk* represented by the Cauchy distribution

According to the WAIC results from the Table 3.4, the selected model for further analysis for the Normal distribution is the one with 5 variables. This is because it has a similar performance (*elpd*) to the model with all the features, but with half the complexity.

| | rank | elpd | p |
|---|---|---|---|
| df_complete | 0 | -151.5 | 13.4 |
| df_top5 | 1 | -151.9 | 6.9 |
| df_top4 | 2 | -153 | 6.2 |
| df_top8 | 3 | -153.2 | 9.5 |
| df_top3 | 4 | -154.2 | 5.6 |
| df_top2 | 5 | -160 | 4.8 |

Table 3.4: The WAIC results for *infection risk* represented by Normal distribution

Finally, for the Student's t, the selected model is the one with 5 features, based on the WAIC scores from the Table 3.5.

| | rank | elpd | p |
|---|---|---|---|
| df_top5 | 0 | -156.9 | 9.2 |
| df_complete | 1 | -157.6 | 18.9 |
| df_top4 | 2 | -157.6 | 8.3 |
| df_top3 | 3 | -158.1 | 6.5 |
| df_top8 | 4 | -159.1 | 13.4 |
| df_top2 | 5 | -164.1 | 5.1 |

Table 3.5: The WAIC results for *infection risk* represented by Student's t distribution

### 3.2.1 Selected Model for GLM with Cauchy Distribution

The Figure 3.5 shows the posterior distribution of the parameters for the Cauchy distribution modelled with 3 variables. The distributions have been truncated on their respective 95% Highest Posterior Density Credible Intervals (HPDI) and a vertical line was drawn, to indicate their values.

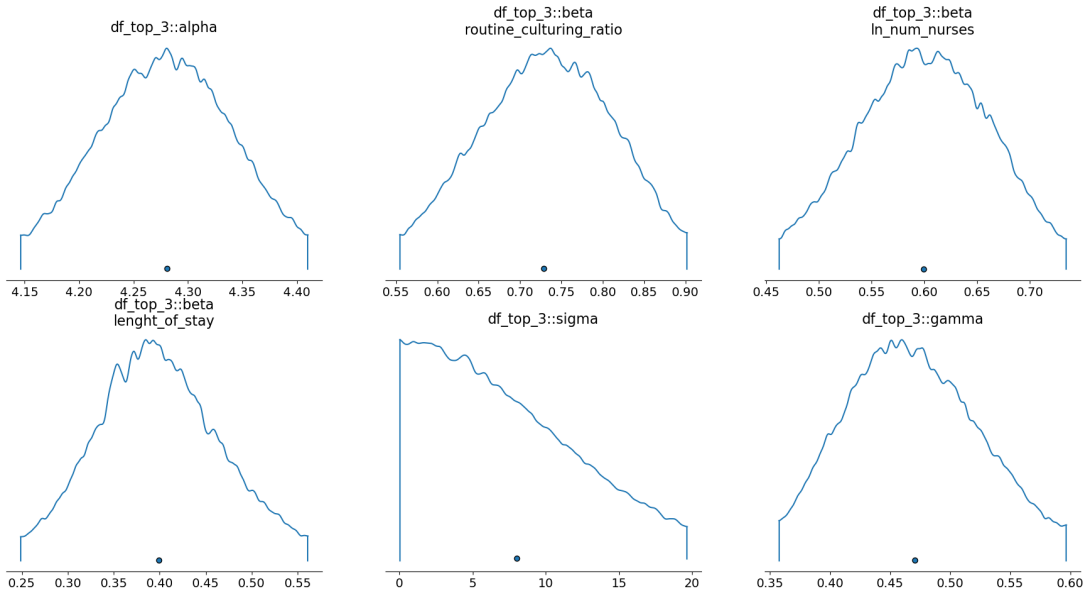



Figura 3.5: The parameters' distributions for Cauchy model, the dot represents the distribution mean.

The Figure 3.6 shows a forest plot of the HPDIs for the coefficients of the covariates, which enables a direct comparison. Furthermore, Table 3.6 shows the values of the mean,

the standard deviation, the approximate 95% HPDI and of the Durbin-Watson statistic ($\hat{r}$) for all the parameters of the model.

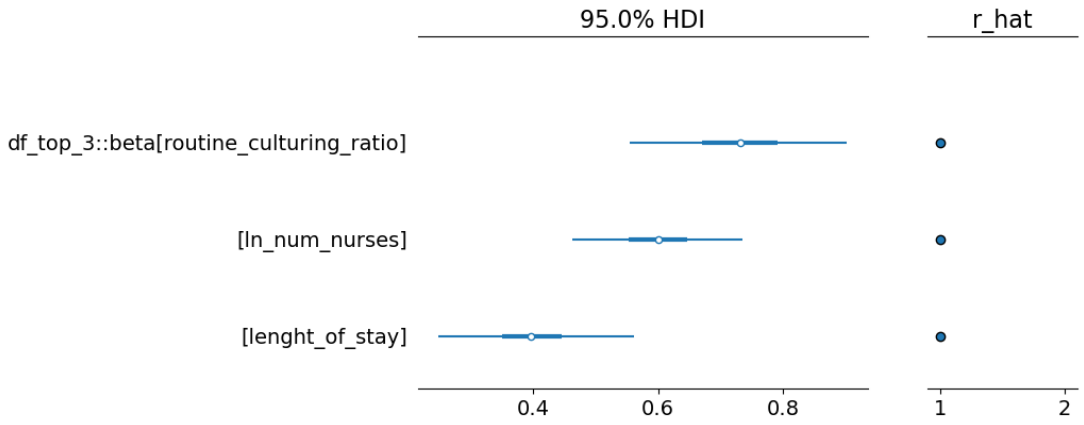
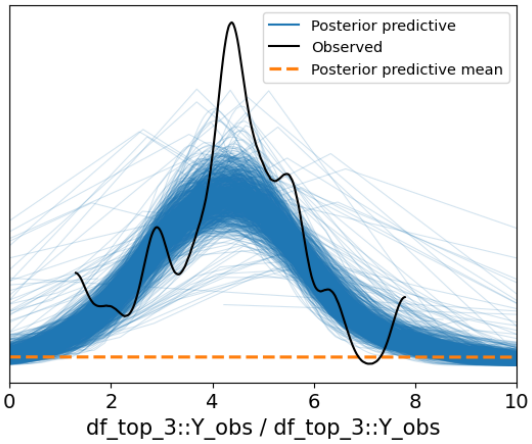



Figura 3.6: Forest plot for Cauchy distribution.

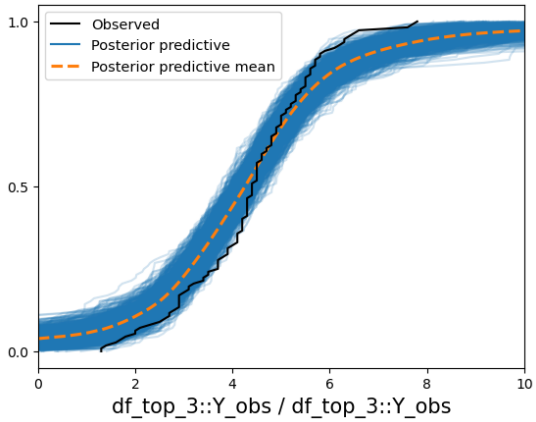
| | **Mean** | **SD** | **HPDI 2.5%** | **HPDI 97.5%** | $\hat{r}$ |
|---|---|---|---|---|---|
| **alpha** | 4.28 | 0.07 | 4.15 | 4.41 | 1.0 |
| **beta[routine_culturing_ratio]** | 0.73 | 0.09 | 0.56 | 0.90 | 1.0 |
| **beta[ln_num_nurses]** | 0.60 | 0.07 | 0.47 | 0.73 | 1.0 |
| **beta[lenght_of_stay]** | 0.40 | 0.08 | 0.25 | 0.55 | 1.0 |
| **sigma** | 7.99 | 6.05 | 0.00 | 18.87 | 1.0 |
| **gamma** | 0.47 | 0.06 | 0.36 | 0.59 | 1.0 |

Table 3.6: The values for parameters' HPDIs in Cauchy distribution

Finally, Figure 3.7a shows the posterior predictive checks (PPC) and the Figure 3.7b shows the cumulative PPC for Cauchy distribution with 3 variables, where the simulated data are plotted in blue against the observed data are plotted in black. The dashed orange line shows the mean of the predictive samples. It is interesting to note that the library responsible for calculating the mean did not succeed in doing so for the PPC of the Cauchy distribution.




(a) PPC for Cauchy distribution




(b) Cumulative PPC for Cauchy distribution

### 3.2.2 Selected Model for GLM with Normal Distribution

The Figure 3.8 shows the posterior distribution of the parameters for the Normal distribution modelled with 5 variables. As before, the distributions have been truncated on their respective 95% Highest Posterior Density Credible Intervals (HPDI).

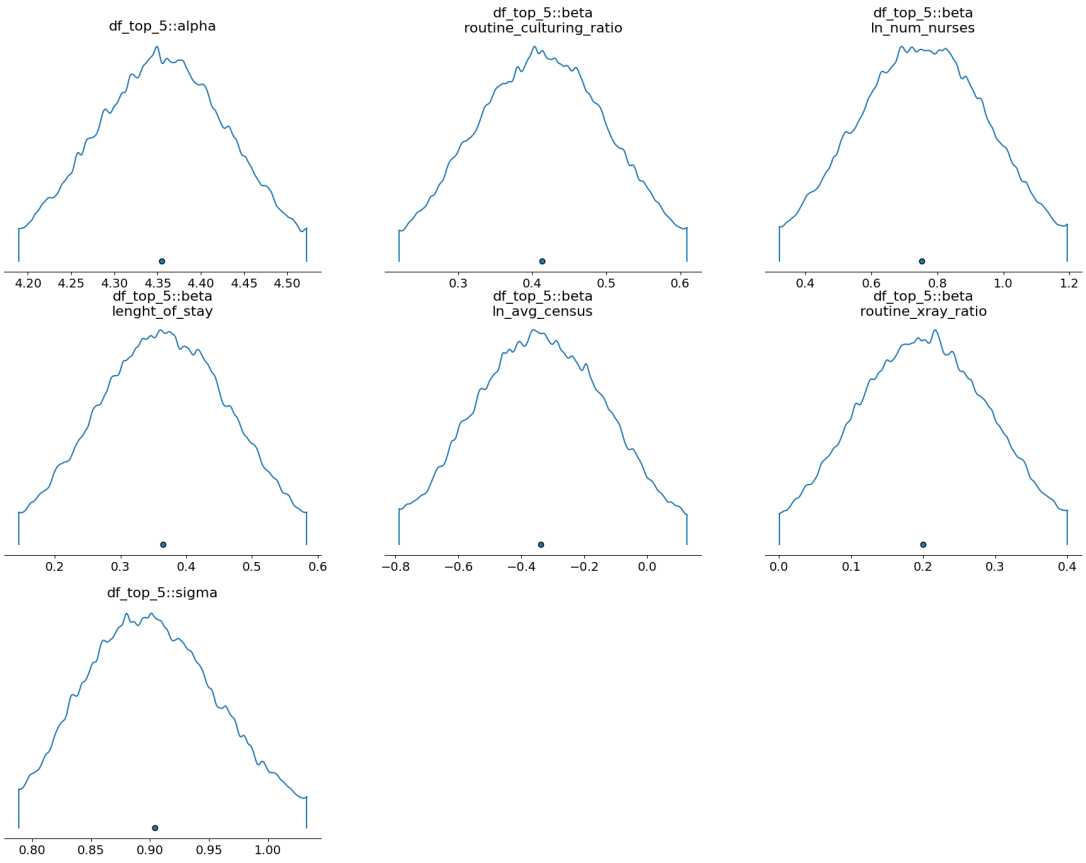



Figura 3.8: The parameters' distributions for Normal model, the dot represents the distribution mean.

The Figure 3.9 shows a forest plot of the HPDIs for the coefficients of the covariates. On the other hand, Table 3.7 shows the values of the mean, the standard deviation, the approximate 95% HPDI and of the Durbin-Watson statistic ($\hat{r}$) for all the parameters of the model.

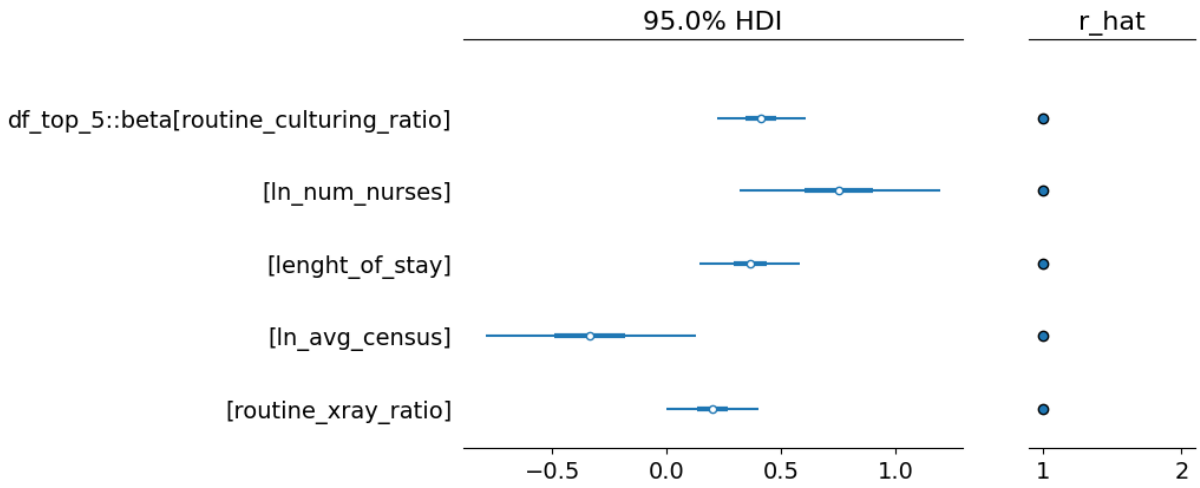
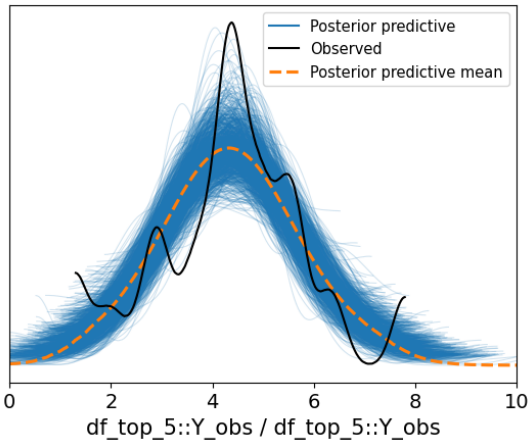



Figura 3.9: Forest plot for Normal

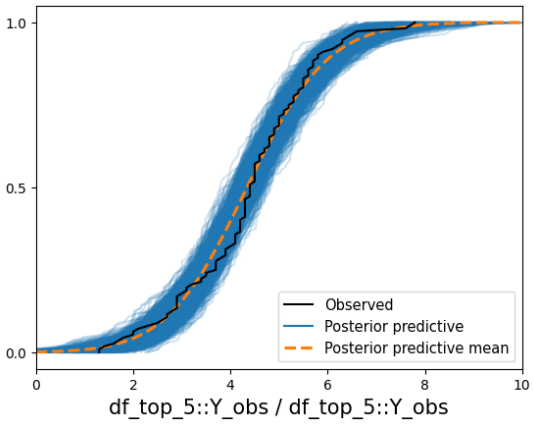
| | **Mean** | **SD** | **HPDI 2.5%** | **HPDI 97.5%** | $\hat{r}$ |
|---|---|---|---|---|---|
| **alpha** | 4.35 | 0.09 | 4.19 | 4.51 | 1.0 |
| **beta[routine_culturing_ratio]** | 0.41 | 0.10 | 0.23 | 0.60 | 1.0 |
| **beta[ln_num_nurses]** | 0.75 | 0.22 | 0.34 | 1.18 | 1.0 |
| **beta[lenght_of_stay]** | 0.37 | 0.11 | 0.15 | 0.57 | 1.0 |
| **beta[ln_avg_census]** | -0.34 | 0.23 | -0.78 | 0.10 | 1.0 |
| **beta[routine_xray_ratio]** | 0.20 | 0.10 | 0.01 | 0.40 | 1.0 |
| **sigma** | 0.90 | 0.06 | 0.79 | 1.02 | 1.0 |

Table 3.7: The values for parameters' HPDIs in Normal Distribution

The Figure 3.10a shows the posterior predictive checks (PPC) and the Figure 3.10b shows the cumulative PPC for the normal distribution with 5 variables.




(a) PPC for Normal distribution




(b) Cumulative PPC for Normal distribution

### 3.2.3 Selected Model for GLM with Student's t Distribution

The Figure 3.11 shows the posterior distribution of the parameters for the Normal distribution modelled with 5 variables. As before, the distributions have been truncated on their respective 95% Highest Posterior Density Credible Intervals (HPDI).

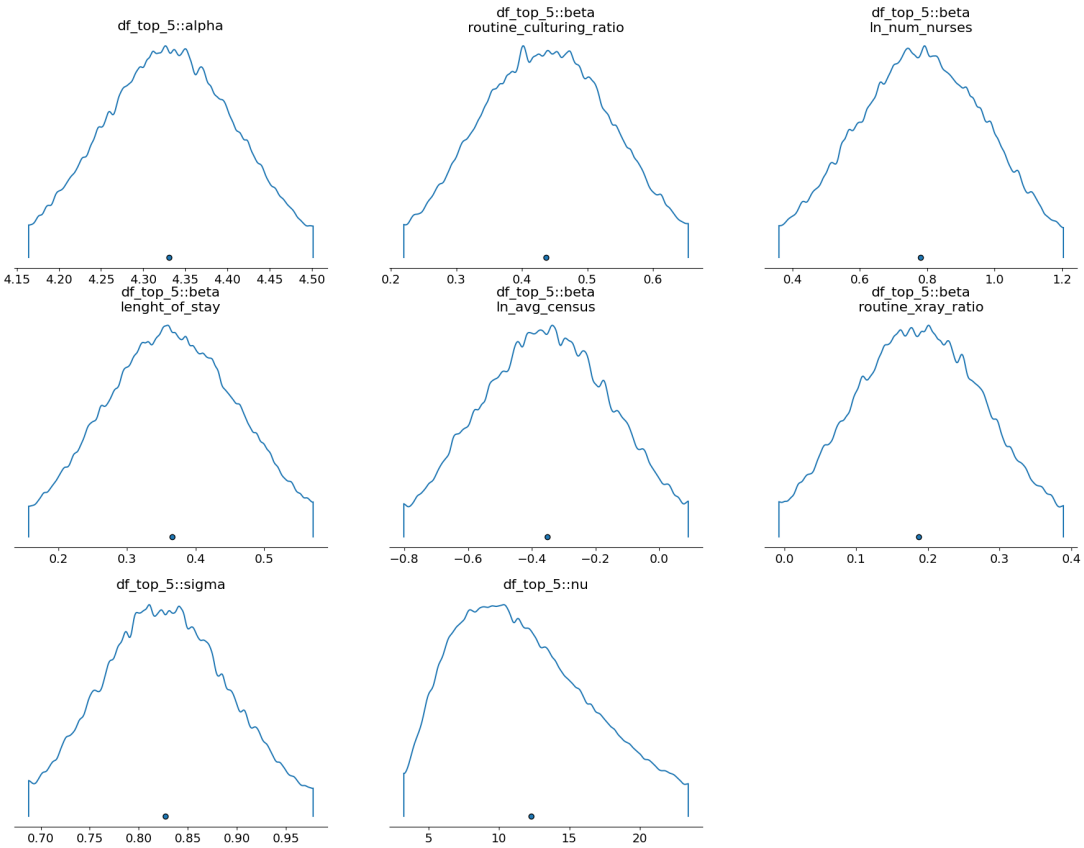



Figura 3.11: The parameters' distributions for Student's t model, the dot represents the distribution mean.

The Figure 3.12 shows a forest plot of the HPDIs for the coefficients of the covariates. Moreover, Table 3.8 shows the values of the mean, the standard deviation, the approximate 95% HPDI and of the Durbin-Watson statistic ($\hat{r}$) for all the parameters of the model.

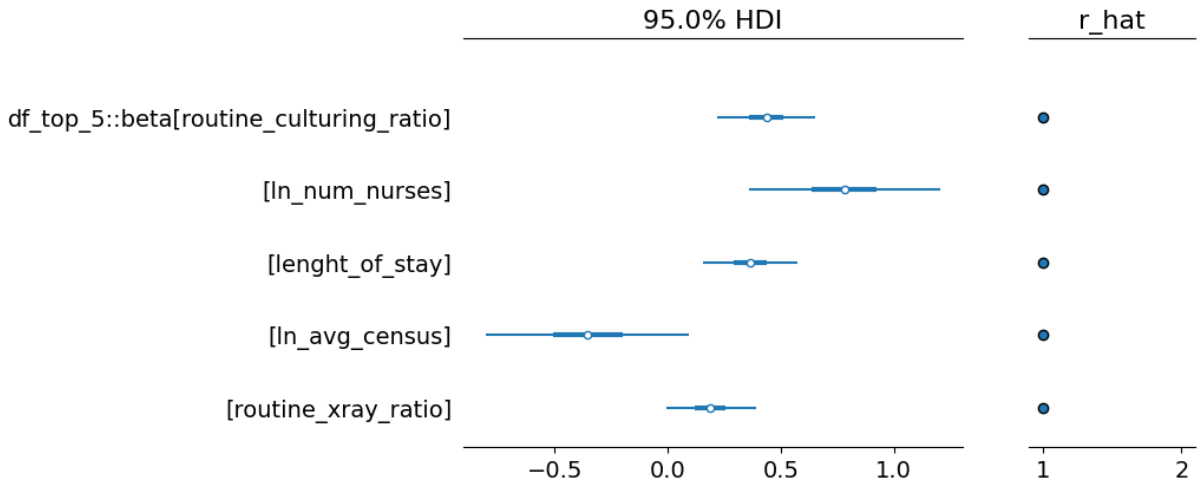
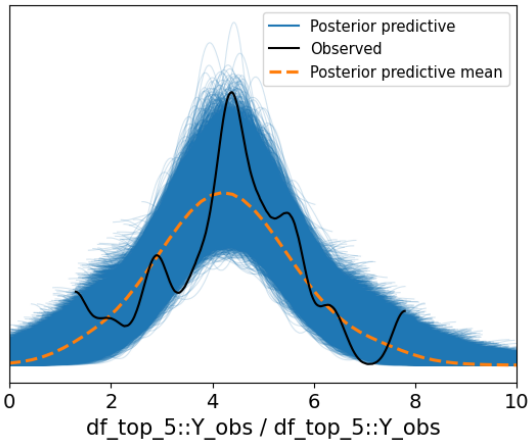



Figura 3.12: Forest plot for Student's t

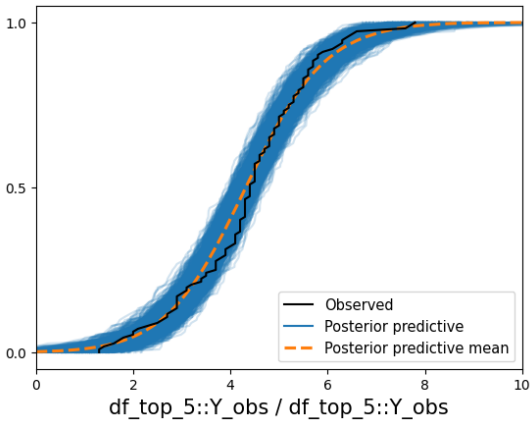
| | **Mean** | **SD** | **HPDI 2.5%** | **HPDI 97.5%** | $\hat{r}$ |
|---|---|---|---|---|---|
| **alpha** | 4.33 | 0.09 | 4.17 | 4.49 | 1.0 |
| **beta[routine_culturing_ratio]** | 0.44 | 0.11 | 0.23 | 0.65 | 1.0 |
| **beta[ln_num_nurses]** | 0.78 | 0.21 | 0.38 | 1.19 | 1.0 |
| **beta[lenght_of_stay]** | 0.37 | 0.11 | 0.17 | 0.57 | 1.0 |
| **beta[ln_avg_census]** | -0.35 | 0.23 | -0.77 | 0.09 | 1.0 |
| **beta[routine_xray_ratio]** | 0.19 | 0.10 | -0.00 | 0.38 | 1.0 |
| **sigma** | 0.83 | 0.07 | 0.69 | 0.97 | 1.0 |
| **nu** | 12.28 | 5.60 | 3.22 | 22.62 | 1.0 |

Table 3.8: The values for parameters' HPDIs in Student's t distribution

The Figure 3.13a shows the posterior predictive checks (PPC) and the Figure 3.13b shows the cumulative PPC for the normal distribution with 5 variables.




(a) PPC for Student's t distribution




(b) Cumulative PPC for Student's distribution

# 4. Discussion

## 4.1 Linear Regression Model

In the first version of the LM, it can be observed from Figure 3.4a that the distribution of the predicted values is similar to the observed values. This is an indication of a good fit with the data.

The Linear Model indicates that the average $\alpha$ is 4.35. And, when further analyzing the coefficients, it can be observed from the Table 3.2 that the most positively impactful variables are *ln num nurses*, *routine culturing ratio*, *ln avg census* and *lenght of stay*, with coefficient values 0.68, 0.52, 0.49 and 0.39, respectively. On the contrary, *ln num beds* is the most negatively impactful variable, with a value of $-0.54$. This means that the higher the number of nurses, of the ratio of routine culturing, of the average census of the hospital and of the average lenght of stay, the highest the risk of nosocomial infection. And, it appears that an increase in the number of hospital beds leads to a decrease in infection risk, which can be expected.

The result of the LM without the outliers is not as well fitted as the complete dataset, as it can be seen in Figure 3.4b. However, its parameters leads to similar conclusions. It's mean value $\alpha$ is equal to 4.33, which is close to the previously obtained value. In addition, the variables with the highest impacts are the same. However, it gives a greater importance to *ln avg census*, with a value of 0.82, while the values of the other parameters decreased. Similarly, it gives a greater importance to the negative predictor *ln num beds*, with a value of $-0.80$

## 4.2 GLM With Cauchy distribution

The WAIC is a measure that accounts for both model fit and complexity, providing a balance between accuracy and overfitting. According to the results from Table 3.3, the best model for the Cauchy distribution is the one with 3 variables and WAIC score of -166.2.

Figure 3.5 shows that the parameters $\alpha$, $\beta_{routine\_culturing\_ratio}$, $\beta_{ln\_num\_nurses}$, $\beta_{lenght\_of\_stay}$ and $\gamma$ converged to symmetrical distributions. The $\sigma$ parameter, on the other hand, is

heavily skewed.

The Forest plot, in Figure 3.6, and Table 3.6 allow to interpreted the fitted model. First of all, it sets the average *infection risk* between 4.15 and 4.41, with a 95% degree of credibility. In a similar fashion, the coefficients of the variables *routine culturing ratio*, *ln num nurses* and *lenght of stay*, are within the intervals $[0.56, 0.9]$, $[0.47, 0.73]$ and $[0.25, 0.55]$, respectively. With means equal to 0.73, 0.6 and 0.4. This shows that the model with the Cauchy distribution gives the greatest importance to the *routineculturingratio* as a predictor of nosocomial infection. However, none of the HPDI contains 0, which means that all the three variables are significant to the prediction of the target variable. In addition, all the variables converged, which can be said from the plot of the posterior distributions and from the result of the Durbin-Watson statistic.

Despite this results, one can conclude that the model with the Cauchy distribution does not fit the data well. This can be seen by the large $\sigma$ value of 7.99 and through Figures 3.7a and 3.7b depicting the PPC and cumulative PPC. In the cumulative PPC plot, it can be seen that the only interval where the predicted values align with the observed ones is around an infection risk of 5%. Outside this region, sampled posterior predictive mean is far from the observed values.

## 4.3 GLM With Normal distribution

According to the results from Table 3.4, the selected model for the Normal distribution includes only 5 features and has an WAIC score of -151.9. With Figure 3.8, it can be seen that all parameters, except for $\sigma$ converged to symmetrical distributions, whereas $\sigma$ is lightly skewed. This may be due to the asymmetrical nature of its prior (Half-Normal).

First of all, the model sets the mean of *infection risk* between 4.19 and 4.51, with a 95% degree of credibility. Further, the Forest plot (Figure 3.9) and Table 3.7, show that *ln num nurses*, *routine culturing ratio* and *lenght of stay* are the greatest contributors to infection risk, with HPIDs ranging from $[0.34, 1.18]$, $[0.23, 0.60]$ and $[0.15, 0.57]$ and mean 0.75, 0.41 and 0.37, respectively. It is interesting to note that these variables are the same as in the Cauchy's model. In addition, it can be observed that the uncertainty in the value of *ln num nurses* is higher than the other two. Furthermore, the model includes the features *ln avg census* and *routine xray ratio*, with HPDIs $[-0.78, 0.10]$ and $[0.01, 0.40]$

and means $-0.34$ and 0.2, respectively. It is possible to say that these variables have low significance in the model, since the first HPDI includes 0 and the second one has a low average and its HPDI almost includes 0. As in the previous case, the Durbin-Watson statistic indicates conversion, since $\hat{R} = 1$ for all parameters.

Finally, from Figures 3.10a and 3.10b depicting the PPC and cumulative PPC, it can be seen that the GLM with Normal distribution for the target variable performs better than the Cauchy model. The cumulative PPC shows that the mean posterior predictive distribution is close to the observed one for most of the interval. However, the model is not well fitted around the infection risk of 4%, since in both plots the prediction is detached from the observed values.

## 4.4 GLM With Student's t distribution

The model with the Student's t distribution is very similar to the Normal one. The best set of variables, according to the WAIC test, is the one with 5 features and WAIC score of $-156.9$, as seen in Table 3.5. The posterior distributions follow the same pattern as in the Normal model, with symmetry occurring in all but the $\sigma$ paramater, which is more heavily skewed than before.

First of all, it sets the mean of *infection risk* between 4.17% and 4.49%$f$, with a 95% degree of credibility. Further, the Forest plot (Figure 3.12) and Table 3.8, shows that *ln num nurses*, *routine culturing gratio* and *lenght of stay* are the greatest contributors to infection risk, with HPIDs ranging from $[0.38, 1.19]$, $[0.23, 0.65]$ and $[0.17, 0.57]$ and means 0.78, 0.44 and 0.37, respectively. These intervals and means are very close to those obtained with the Normal distribution model.

In addition, the features *ln avg census* and *routine xray ratio*, with HPDIs $[-0.77, 0.09]$ and $[-0.00, 0.38]$ and means $-0.35$ and 0.19 have low significance in the model, since their HPDIs includes 0. The Durbin-Watson statistic also indicates convergence, since $\hat{R} = 1$ for all parameters.

However, it can also be observed that the $\nu$ parameter has a high standard deviation, with a value of 5.60, and that its HPDI is large, ranging from 3.22 to 22.62.

Finally, from Figures 3.13a and 3.13b depicting the PPC and cumulative PPC, it can be seen that the GLM with Student's t is very similar to the Normal model. It also

shows a detachment around the infection risk of 4%, while following the data in the other intervals. Thus, one can conclude that fitting the Student's t model did not add much value, since its results do not differ from the Gaussian model, despite the added parameter $\nu$

## 4.5 The selected model

With the analysis of the selected models for each distribution, the one with the best performance is the General Linear Regression with the Normal distribution fitted with the top 5 features. It showed the best fit with the data, and was able to inform which were the most relevant variables in predicting the infection risk.

# 5. Conclusion

In this project, the task of predicting the risk of nosocomial infection was tackled with the fit of two Linear Regression models with the Least Squares Method (LSM) and 18 different Generalized Linear Regression Models (GLM), with three different distributions. The model that performed the best in predicting the observed data was the GLM with the Normal distribution, since it showed the best fitted predictive posterior distribution. One could argue that the model with the Least Squares Method had a better fit with the data. However, this frequentist approach can only provide point estimates for the parameters, whereas the GLM outputs their distributions. This makes the second model more informative, since the uncertainty can be better measured. Thus, the LSM is only capable of providing a baseline of fit, so as to clarify what could be expected in the GLMs.

The Normal model shows that the average infection risk is between 4.19% and 4.15%. In addition, it shows that the best variables to predict an increase in nosocomial infection are the *number of nurses* in the hospital, the *average lenght of stay* and the *ratio of routine culturing*. This is aligned to what was expected, since these are the features that have the highest correlation with the target variable.

Finally, in next stages of this project, it would be necessary to re-fit the Normal model with only those variables that were indicated as relevant and compare their WAIC measures. Thus, it would be possible to further simplify the model or to discover new unseen patterns in the data.

# 6. Appendix

## 6.1 Jupyter notebook used for the MCMC method

Starting in the next page, the notebook used for fitting the Generalized Linear Regression model can be seen. The notebook uses several functions that were defined in auxiliary python files that can be seen in the following appendix sections.

# mcmc-notebook

December 19, 2023

## 1 Notebook for fitting Generalized Linear Model with Markov Chain Monte Carlo method

### 1.1 Import libraries

[ ]:
```
# Importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import arviz as az
import seaborn as sns
import pymc as pm

from aux_functions.mcmc_transformations import (import_dataset,␣
 ↪transform_variables, pop_variable, convert_to_float, normalize_dataset)
from aux_functions.mcmc_corr_matrix import (get_ordered_correlation_matrix,␣
 ↪plot_correlation_matrix)
from aux_functions.mcmc_modelling import (get_specified_dfs, get_models,␣
 ↪sample_models, get_identified_sampling_results, get_waic_measures,␣
 ↪get_model_from_list)
```

### 1.2 Define parameters for data transformation

[ ]:
```
# Specify path of the dataset
path = "data/dataset.txt"

# Specify column names of the dataset
column_names = ['lenght_of_stay', 'age', 'infection_risk',␣
 ↪'routine_culturing_ratio', 'routine_xray_ratio', 'num_beds',␣
 ↪'med_school_affil', 'region', 'avg_census', 'num_nurses', 'avelbl_services']

# Specify the columns to one-hot encode
columns_to_dummify = ['med_school_affil', 'region']

# Specify the columns to apply the ln() function
columns_to_apply_ln = ['num_nurses', 'num_beds', 'avg_census']
```

```
# Specify the target variable
target_variable = 'infection_risk'

# Specify the distribution of the target variable ("cauchy", "student_t" or␣
 ↪"normal")
y_proposal_distribution = "normal"

# Specify which models you wish to fit
# "model_name": [list of variables to include]
variables_of_dfs_to_build = {
    'df_complete':['routine_culturing_ratio', 'ln_num_nurses',␣
 ↪'lenght_of_stay', 'ln_avg_census', 'routine_xray_ratio', 'ln_num_beds',␣
 ↪'avelbl_services', 'med_school_affil_1', 'region_1', 'region_2', 'region_4',␣
 ↪'age', 'region_3', 'med_school_affil_2'],
    'df_top_8': ['routine_culturing_ratio', 'ln_num_nurses', 'lenght_of_stay',␣
 ↪'ln_avg_census', 'routine_xray_ratio', 'ln_num_beds', 'avelbl_services',␣
 ↪'med_school_affil_1', 'med_school_affil_2'],
    'df_top_5': ['routine_culturing_ratio', 'ln_num_nurses', 'lenght_of_stay',␣
 ↪'ln_avg_census', 'routine_xray_ratio'],
    'df_top_4': ['routine_culturing_ratio', 'ln_num_nurses', 'lenght_of_stay',␣
 ↪'ln_avg_census'],
    'df_top_3': ['routine_culturing_ratio', 'ln_num_nurses', 'lenght_of_stay'],
    'df_top_2': ['routine_culturing_ratio', 'ln_num_nurses',]
}
```

### 1.3 Define parameters for models to fit and compare

[ ]:
```
# Array with names of the models
models_names = variables_of_dfs_to_build.keys()

# Number of samples of the Markov Chain
number_of_samples = 100_000
```

### 1.4 Run transformation pipeline

[ ]:
```
# Imports the dataset
df = import_dataset(path, column_names)

# Applies one-hot encoding and ln() to the specified variables
df = transform_variables(df, columns_to_dummify, columns_to_apply_ln)

# Separates X and y
X, y = pop_variable(df, target_variable)
X = convert_to_float(X)
y = convert_to_float(y)
```

```
# Normalizes the features X
X = normalize_dataset(X)
```

### 1.5 Create dataframes and models according to specification

[ ]:
```
# Object that contains all the specified datasets
dfs = get_specified_dfs(X, variables_of_dfs_to_build)

# Example:
dfs[0].head(2)
```

[ ]:
```
# List in which each item is a pm.Model() object from the pymc library
# Each model will have a diferent dataset ex: models[1] has the dataset[1] as␣
 ↪covariate
models = get_models(dfs, models_names, y, y_proposal_distribution)
```

### 1.6 Sample from models

[ ]:
```
# List in which each item is the sample result of each model of the specified␣
 ↪models
# Ex: sampling_results[0] has the sampling result of model[0]
sampling_results = sample_models(number_of_samples, models)
```

[ ]:
```
# Dictionary with {model_name: sampling_result} for easier identification
# Example: {"df_top_8": sampling_result_of_df_top_8}
identified_sampling_results = get_identified_sampling_results(sampling_results,␣
 ↪models_names)
```

### 1.7 Run WAIC tests

[ ]:
```
# If there is more than one model being fitted, calculate the WAIC measure␣
 ↪between them
if len(variables_of_dfs_to_build) > 1:
    dfwaic = get_waic_measures(identified_sampling_results)
    dfwaic[['rank','elpd_waic', 'p_waic']]
```

### 1.8 Choose model for the next analysis

[ ]:
```
# Choose model for the next analysis, between the specified ones
selected_model = 'df_top_3'
```

### 1.9 View summary of selected model for the current distribution

[ ]:
```
az.summary(identified_sampling_results[selected_model], round_to=2)
```

### 1.10 Trace plots

[ ]:
```
az.plot_trace(identified_sampling_results[selected_model])
```

### 1.11 Forest plot

[ ]:
```
# Plot parameters for covariates
az.plot_forest(identified_sampling_results[selected_model],␣
↪var_names=[f"{selected_model}::beta" ], combined=True, hdi_prob=0.95,␣
↪r_hat=True, );
```

### 1.12 Density of posterior distributions

[ ]:
```
# Plot posterior densities of parameters
az.plot_density(identified_sampling_results[selected_model], group='posterior',␣
↪hdi_prob=0.95);
```

### 1.13 Sample posterior predictive distribution

[ ]:
```
Y_pred = pm.
↪sample_posterior_predictive(identified_sampling_results[selected_model],␣
↪model=get_model_from_list(models, selected_model)) # for each sample it␣
↪draws a beta from each found beta distribution -> finds Y_pred for all X
```

### 1.14 Plot cumulative posterior predictive distribution

[ ]:
```
num_pp_samples = 1000

fig, ax = plt.subplots()
plt.xlim(0,10)
az.plot_ppc(Y_pred, num_pp_samples=num_pp_samples, kind='cumulative', ax=ax)
```

### 1.15 Plot posterior predictive distribution

[ ]:
```
num_pp_samples = 1000

fig, ax = plt.subplots()
plt.xlim(0,10)
az.plot_ppc(Y_pred, num_pp_samples=num_pp_samples, ax=ax)
```

## 6.2 Auxiliary functions for variable transformation

In this section, the auxiliary functions used in the variable transformation pipeline can be seen.

### 6.2.1 import_dataset

```
def import_dataset(path, column_names):
    """
    Import a dataset from a specified file path.

    Parameters:
    path (str): The file path of the dataset to import.
    column_names (list of str): The list of column names to assign to the dataset.

    Returns:
    DataFrame: A pandas DataFrame containing the imported dataset with the specified
    column names.
    """
    return pd.read_csv(path, sep=" ", header=None, names=column_names)
```

### 6.2.2 transform_variables

```
def transform_variables(df_in, columns_to_dummify, columns_to_apply_ln):
    """
    Transform variables of a DataFrame by applying natural logarithm and
    dummification.

    Parameters:
    df_in (DataFrame): The input DataFrame to transform.
    columns_to_dummify (list of str): The list of column names to dummify.
    columns_to_apply_ln (list of str): The list of column names to apply the natural
    logarithm to.

    Returns:
    DataFrame: A transformed DataFrame with logarithmically scaled and dummified
    variables.
    """
```

```
    df = _apply_ln_and_drop_original(df_in, columns_to_apply_ln)
    df_out = _dummify_columns(df, columns_to_dummify)
    return df_out
```

### 6.2.3 _apply_ln_and_drop_original

```
def _apply_ln_and_drop_original(df, columns):
    """
    Apply natural logarithm to specified columns and drop the original columns.

    Parameters:
    df (DataFrame): The DataFrame to modify.
    columns (list of str): The list of column names to apply the natural logarithm to.

    Returns:
    DataFrame: The modified DataFrame with logarithmically scaled columns.
    """
    for column in columns:
        df[f'ln_{column}'] = np.log(df[column])
        df.drop(columns=column, inplace=True)
    return df
```

### 6.2.4 _dummify_columns

```
def _dummify_columns(df, columns):
    """
    Convert categorical variable(s) into dummy/indicator variables.

    Parameters:
    df (DataFrame): The DataFrame to modify.
    columns (list of str): The list of column names to convert into dummy variables.

    Returns:
    DataFrame: The modified DataFrame with additional dummy variables.
    """
    return pd.get_dummies(df, columns=columns)
```

### 6.2.5 pop_variable

```
def pop_variable(df, column_to_pop):
    """
    Pop a column from a DataFrame and return the remaining DataFrame and the popped
    column.

    Parameters:
    df (DataFrame): The DataFrame to modify.
    column_to_pop (str): The name of the column to pop.

    Returns:
    tuple: A tuple containing the modified DataFrame (X) and the popped column as a
    Series (y).
    """
    if column_to_pop in df.columns:
        y = df.pop(column_to_pop)
    return df, y
```

### 6.2.6 convert_to_float

```
def convert_to_float(df):
    """
    Convert the data type of DataFrame columns to float.

    Parameters:
    df (DataFrame): The DataFrame to modify.

    Returns:
    DataFrame: The modified DataFrame with columns converted to float data type.
    """
    return df.astype(float)
```

### 6.2.7 normalize_dataset

```
def normalize_dataset(df):
    """
    Normalize the dataset by subtracting the mean and dividing by the standard
    deviation.
```

```
4
5       Parameters:
6       df (DataFrame): The DataFrame to normalize.
7
8       Returns:
9       DataFrame: The normalized DataFrame.
10      """
11      df -= df.mean()
12      df /= df.std()
13      return df
```

## 6.3 Auxiliary functions for the MCMC method notebook

In this section, the auxiliary functions used in the fitting of the Generalized Linear Regression can be seen. It is important to note that the functions allow for the fit of multiple models of a chosen distribution. In addition, it allows for choosing the desired distribution between the Cauchy, the Student-t and the Normal.

### 6.3.1 get_specified_dfs

```
1  def get_specified_dfs(df_complete, variables_of_dfs_to_build):
2      """
3      Generate a list of DataFrames based on specified variables from a complete
       DataFrame.
4
5      Parameters:
6      df_complete (DataFrame): The complete DataFrame from which to extract variables.
7      variables_of_dfs_to_build (dict): A dictionary where each key-value pair consists
       of
8                                        a DataFrame name and a list of variables to
       include in that DataFrame.
9
10     Returns:
11     list of DataFrame: A list of DataFrames, each containing a specified subset of
       variables from the complete DataFrame.
12     """
```

```
    dfs = []
    for name, variables in variables_of_dfs_to_build.items():
        dfs.append(df_complete[variables].copy())
    return dfs
```

### 6.3.2 get_models

```
def get_models(dfs, models_names, y, y_proposal_distribution="normal"):
    """
    Create and return a list of probabilistic models for each DataFrame in a list.

    Parameters:
    dfs (list of DataFrame): A list of DataFrames to model.
    models_names (list of str): Names for the models corresponding to each DataFrame.
    y (array-like): The dependent variable for the model.
    y_proposal_distribution (str, optional): The type of distribution to use for the
    proposal distribution
                                            (default is "normal"). Options are
    "normal", "cauchy", and "student_t".

    Returns:
    list of pm.Model: A list of PyMC models, each corresponding to a DataFrame and
    model name in the input lists.
    """
    models = []
    for df, model_name in zip(dfs, models_names):
        with pm.Model(coords={'beta_names': df.columns.values}, name=model_name) as
    model:
            # Priors
            alpha = pm.Normal("alpha", mu=0, sigma=10)
            beta = pm.Normal("beta", mu=0, sigma=10, shape=df.shape[1],
    dims="beta_names")
            sigma = pm.HalfNormal("sigma", sigma=10)

            # Expected value of outcome
            mu = alpha + pm.math.dot(df, beta)

            # Different proposal distributions
            # [code continues as per the original content]
```

```
        models.append(model)
    return models
```

### 6.3.3 _is_proposal_distribution_cauchy

```
def _is_proposal_distribution_cauchy(y_proposal_distribution):
    """
    Check if the proposal distribution type is 'cauchy'.

    Parameters:
    y_proposal_distribution (str): The type of proposal distribution to check.

    Returns:
    bool: True if 'cauchy', False otherwise.
    """
    return True if y_proposal_distribution == "cauchy" else False
```

### 6.3.4 _is_proposal_distribution_student

```
def _is_proposal_distribution_student(y_proposal_distribution):
    """
    Check if the proposal distribution type is 'student_t'.

    Parameters:
    y_proposal_distribution (str): The type of proposal distribution to check.

    Returns:
    bool: True if 'student_t', False otherwise.
    """
    return True if y_proposal_distribution == "student_t" else False
```

### 6.3.5 _is_proposal_distribution_normal

```
def _is_proposal_distribution_normal(y_proposal_distribution):
    """
    Check if the proposal distribution type is 'normal'.

    Parameters:
```

```
    y_proposal_distribution (str): The type of proposal distribution to check.

    Returns:
    bool: True if 'normal', False otherwise.
    """
    return True if y_proposal_distribution == "normal" else False
```

### 6.3.6 sample_models

```
def sample_models(num_samples, models):
    """
    Sample from a list of models using the Metropolis-Hastings algorithm.

    Parameters:
    num_samples (int): The number of samples to draw from each model.
    models (list of pm.Model): A list of PyMC models to sample from.

    Returns:
    list: A list of sampling results for each model.
    """
    sampling_results = []
    for i, model in enumerate(models):
        with model:
            step = pm.Metropolis()
            sample = _sample_model(num_samples, step)
            sampling_results.append(sample)
    return sampling_results
```

### 6.3.7 _sample_model

```
def _sample_model(num_samples, step):
    """
    Draw samples from a model using a given step method.

    Parameters:
    num_samples (int): The number of samples to draw.
    step (pm.step_methods.arraystep.ArrayStepShared): The step method to use for
    sampling.
```

```
    Returns:
    MultiTrace: A PyMC MultiTrace object representing the sampled values.
    """
    return pm.sample(num_samples, step=step, idata_kwargs = {'log_likelihood': True})
```

### 6.3.8 get_identified_sampling_results

```
def get_identified_sampling_results(sampling_results, models_names):
    """
    Associate model names with their corresponding sampling results.

    Parameters:
    sampling_results (list): A list of sampling results from models.
    models_names (list of str): The names of the models corresponding to each set of
    sampling results.

    Returns:
    dict: A dictionary with model names as keys and their sampling results as values.
    """
    model_trace_dict = dict()
    for i, num in enumerate(models_names):
        model_trace_dict.update({num: sampling_results[i]})
    return model_trace_dict
```

### 6.3.9 get_waic_measures

```
def get_waic_measures(identified_sampling_results):
    """
    Calculate the Widely Applicable Information Criterion (WAIC) for a set of models.

    Parameters:
    identified_sampling_results (dict): A dictionary with model names as keys and
    their sampling results as values.

    Returns:
    DataFrame: A DataFrame containing the WAIC measures for each model.
    """
```

```
    return az.compare(identified_sampling_results, ic='waic')
```

### 6.3.10 get_model_from_list

```
def get_model_from_list(models, model_name):
    """
    Retrieve a model from a list of models by its name.

    Parameters:
    models (list of pm.Model): A list of PyMC models.
    model_name (str): The name of the model to retrieve.

    Returns:
    pm.Model: The model with the specified name, if found in the list.
    """
    for model in models:
        if model.name == model_name:
            return model
```

## 6.4 Jupyter notebook used for the Linear regression with LSM

Starting in the next page, the notebook used for fitting the Linear Regression with LSM model can be seen. The notebook uses several functions that were defined in auxiliary python files that can be seen in the following appendix sections.

# linear-regression

December 19, 2023

## 1 Notebook for fitting Linear Regression Model with the Least Squares Method

### 1.1 Import libraries

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from aux_functions.aux_plots import (plot_distribution_actual_predicted)
from aux_functions.statistics import (apply_linear_regression_complete)
from aux_functions.mcmc_transformations import (import_dataset,␣
 ↪transform_variables, pop_variable, convert_to_float, normalize_dataset)
%matplotlib inline
```

### 1.2 Define parameters for data transformation

[ ]:
```
# Importing dataset
column_names = ['lenght_of_stay', 'age', 'infection_risk',␣
 ↪'routine_culturing_ratio', 'routine_xray_ratio', 'num_beds',␣
 ↪'med_school_affil', 'region', 'avg_census', 'num_nurses', 'avelbl_services']
columns_to_dummify = ['med_school_affil', 'region']
columns_to_apply_ln = ['num_nurses', 'num_beds', 'avg_census']
features = ['lenght_of_stay', 'age', 'routine_culturing_ratio',␣
 ↪'routine_xray_ratio', 'num_beds', 'med_school_affil', 'region',␣
 ↪'avg_census', 'num_nurses', 'avelbl_services']
target_variable = 'infection_risk'
```

### 1.3 Fitting linear model with all the data

#### 1.3.1 Transforming dataset

[ ]:
```
df = import_dataset("data/dataset.txt", column_names)
df = transform_variables(df, columns_to_dummify, columns_to_apply_ln)
X, y = pop_variable(df, target_variable)
X = convert_to_float(X)
```

```
y = convert_to_float(y)
X = normalize_dataset(X)
df_to_fit = pd.concat([X, y], axis=1)
```

#### 1.3.2 Apply linear regression

```
[ ]: y_test, y_prediction, LR = apply_linear_regression_complete(df_to_fit,␣
      ↪'infection_risk')
```

#### 1.3.3 Visualize results

```
[ ]: result_all_data = pd.DataFrame((list(zip(X.columns, np.round(LR.coef_, 2)))))
result_all_data.rename(columns={0: 'feature', 1: 'LSM'}, inplace=True)
print("Intercept:", np.round(LR.intercept_, 2))
result_all_data
```

#### 1.3.4 Visualize predicted and observed distributions

```
[ ]: plot_distribution_actual_predicted(y_test, y_prediction)
```

### 1.4 Linear model without outliers

#### 1.4.1 Transforming dataset

```
[ ]: df = import_dataset("data/dataset.txt", column_names)
df = transform_variables(df, columns_to_dummify, columns_to_apply_ln)
X, y = pop_variable(df, target_variable)
X = normalize_dataset(X)
df_to_fit = pd.concat([X, y], axis=1)
```

#### 1.4.2 Dropping outliers

```
[ ]: indexes_to_drop = [40, 93, 107, 53, 13, 54]
df_to_fit.drop(index=indexes_to_drop, inplace=True)
len(df_to_fit)
```

#### 1.4.3 Apply linear regression

```
[ ]: y_test, y_prediction, LR = apply_linear_regression_complete(df_to_fit,␣
      ↪'infection_risk')
```

### 1.4.4 Visualize results

[ ]:
```
result_without_outliers = pd.DataFrame((list(zip(X.columns, np.round(LR.coef_,␣
↪2)))))
result_without_outliers.rename(columns={0: 'feature', 1: 'LSM without␣
↪outliers'}, inplace=True)
print("Intercept:", np.round(LR.intercept_, 2))
result_without_outliers
```

### 1.4.5 Visualize predicted and observed distributions

[ ]:
```
plot_distribution_actual_predicted(y_test, y_prediction)
```

## 6.5 Auxiliary functions for the Linear Regression with LSM

In this section, the auxiliary functions used in the fitting and analysis of the Linear Regression with LSM can be seen.

### 6.5.1 apply_linear_regression_complete

```python
def apply_linear_regression_complete(df: pd.DataFrame, target: str) -> tuple:
    """
    Fits a Linear Regression model to the provided dataframe on the specified target
    variable.

    Parameters:
    df : pd.DataFrame
        The dataframe containing the dataset with features and target variable.
    target : str
        The name of the target variable in the dataframe.

    Returns:
    tuple
        A tuple containing the true values and the predicted values by the model.
    """
    if target not in df.columns:
        raise ValueError(f"Target column '{target}' not found in the dataframe.")

    y = df[target]
    x = df.drop(columns=[target])
    LR = LinearRegression()
    LR.fit(x, y)
    y_prediction = LR.predict(x)

    return y, y_prediction, LR
```

### 6.5.2 plot_distribution_actual_predicted

```python
def plot_distribution_actual_predicted(y_test: np.ndarray, y_prediction: np.ndarray)
    -> None:
    """
    Plots the distribution of actual vs. predicted values using Kernel Density
    Estimate (KDE) plots.

    Parameters:
    y_test : np.ndarray
        The true values of the target variable from the test dataset.
    y_prediction : np.ndarray
        The predicted values of the target variable from the model.

    Returns:
    None
    """
    sns.set_theme(style="whitegrid")  # Sets the theme for a cleaner look
    plt.figure(figsize=(6, 4))  # Sets a standard figure size

    # KDE plot for predicted values
    sns.kdeplot(y_prediction, color='blue', label='Predicted Values')

    # KDE plot for actual values
    sns.kdeplot(y_test, color='darkred', label='Actual Values')

    plt.title('Distribution of Actual vs Predicted Values')
    plt.xlabel('Infection Risk')
    plt.ylabel('Density')
    plt.legend()  # If you still want to show the legend
    plt.tight_layout()  # Adjusts plot to ensure everything fits without overlapping
    plt.show()  # Displays the plot
```

# 7. References

[1] A Sikora and F Zahra. *Nosocomial Infections*. In: StatPearls [Internet]. Updated 2023 Apr 27. Treasure Island (FL), Jan. 2023. URL: https://www.ncbi.nlm.nih.gov/books/NBK559312/.