Assignment 2: Pulsar Classification
Bayesian Learning

Antonio Gañán Mora
100438082@alumnos.uc3m.es

From gravitational waves measurements to exoplanet detection the applications of advanced statistical methods are more than ever required in Astrophysics. The large number of observatories and telescopes provides huge amount of data, but that data require corrections, either due to the noise (interstellar, atmospheric, devices...) or due to transformations that are required to obtain relevant astrophysical variables. Rigorous and computationally efficient statistical methods are mandatory for astrophysical data analysis nowadays.

Classification of astronomical objects is a classical example of usage of machine learning. Pulsars are rotating starts emitting radio waves, which are captured on Earth via radio telescopes. Pulsars eject a strong magnetic field through their poles. The axis of radiation emission and the axis of rotation need not to coincide, thus their emission beam sweeps across the sky, and when this crosses the line of sight of the telescope we detect a high pulse of radiation. Collecting the radiation as a function of time, we observe a periodic pattern due to the rotation. Individual pulses differ one from another, but considering a large set of pulses clear statistical characteristics are observed. If there is a potential detection for a pulsar, it is categorized as *candidate*. Almost all detections are caused by radio frequency interference (RFI) and noise, making legitimate signals hard to find. The objective of this assignment is to construct logistic regressors, both frequentists and Bayesians, that are able to automatically classify pulsar candidates.

1 `HTRU2` dataset

The `HTRU2` dataset contains samples of pulsar candidates collected during the Universe Resolution survey [1]. The dataset consists on 17898 instances, of which 1639 are positive examples, that is, pulsars, and 16259 are negative examples, that is, RFI/noise. There are 8 continuous variables in the dataset and a single class variable: mean of the integrated profile; standard deviation of the integrated profile; excess kurtosis of the integrated profile; skewness of the integrated profile; mean of the DM-SNR curve; standard deviation of the DM-SNR curve; excess kurtosis of the DM-SNR curve; skewness of the DM-SNR curve; and the class. The names used in R and in plots are `mean_ip`, `sd_ip`, `kurt_ip`, `skew_ip`, `mean_dmsnr`, `sd_dmsnr`, `kurt_dmsnr`, `skew_dmsnr`, and `is_pulsar`, respectively. Let us introduce the concept of integrated profile and DM-SNR curve.

As stated before, from Earth we observe periodic pulses of radiation. From the superposition of such periodic pulses we can infer statistical properties. Such a statistical superposition is called the integrated profile, and it is a fingerprint for each pulsar. The

[1] R. J. Lyon, B. W. Stappers, S. Cooper, J. M. Brooke, J. D. Knowles, *Fifty Years of Pulsar Candidate Selection: From simple filters to a new principled real-time classification approach MNRAS*, 2016.

DM-SNR is a more technical concept. When radiation travels through space, there are scattering processes until it reaches the Earth. This scattering is dominated by electrons in the interstellar medium and produces dispersion in the frequencies, which can be measured as a delay in the reception of the signals depending on the frequency. The dispersion measure (DM) is an indicator of such delay. The signal to noise ratio (SNR) is the quotient of the power of the signal and the power of noise. From the plot of the DM-SNR curves we are able to detect whether a signal is noise or a pulsar: a legitimate pulsar signal will be dispersed by electrons in the interstellar medium, therefore its SNR should peak at a DM greater than zero [2].

2 Logistic Regression

Linear regression is no longer valid when the response is binary. For these cases, logistic regression is a natural extension of linear regression. Let Y be the binary response variable, and let $X_1, \cdots, X_p$ be the predictors. The binary variable is a Bernoulli variable,

$$Y \sim \text{Ber}(p), 0 \leq p \leq 1, \tag{1}$$

which satisfies $E[Y] = P[Y = 1|p] = p$. Logistic regression assumes that the logit of the mean of the response given the predictors is a linear function of the predictors,

$$\text{logit}(E[Y|X_1 = x_1, \cdots, X_p = x_p]) = \nu, \quad \nu = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p, \tag{2}$$

where $\text{logit}(p)$ is the inverse function of the logistic function,

$$\text{logit}(p) = \text{logistic}^{-1}(p) = \log\left(\frac{p}{1-p}\right). \tag{3}$$

We want to estimate the optimum β_i's for our dataset.

From a frequentist perspective, the estimation is done by means of the maximum likelihood estimation. This procedure is implemented by the `glm` R function. The `stepAIC` function starts from a given model and a given scope of predictors, and sequentially includes or removes predictors maximizing some information measure. We have considered the Bayesian information criterion (BIC), which intuitively is a trade-off between how well the model fits the data and the complexity, the number of parameters, of the model. The lower the BIC, the better. BICs for different datasets cannot be compared, but BICs from different models trained with the same data can be compared.

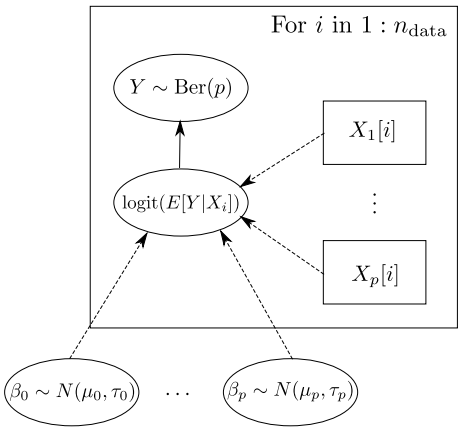

Figure 1: Dependencies for the Bayesian logistic regression model.

[2] R. J. Lyon, *Why Are Pulsars Hard To Find?*, PhD Thesis, University of Manchester, 2015.

From a Bayesian perspective, the β_i's are not fixed parameters to be found, rather they are considered as random variables known as priors. These priors reflect known information about the β_i's. By the Bayes Theorem, when the priors are shown the data, the distribution is updated and includes that new knowledge. The conditioned distribution of β_i's on the data is known as posterior. Point estimations are based on the posterior distribution. Unfortunately, the posterior distribution is rarely analytical and generally quite complex, thus we cannot easily sample from it. Note that the logit$(E[Y])$ (and also p) depends on the distributions for β_i and the precise value for the logit$(E[Y])$ depends as well on the data, the X_i's. All the conditional dependencies of our Bayesian logistic model can be seen in Figure 1. Generally, though not always, we assume normal priors with some mean and some tolerance ($\tau = \frac{1}{\sigma^2}$).

The Gibbs sampling or Markov chain Monte Carlo allows to more easily compute the posterior distribution. This sampling can be done automatically by means of the program `OpenBUGS` and its R package, `R2OpenBUGS`. For `R2OpenBUGS` there is no `stepAIC` function, so there is no automatic selection of predictors. Nevertheless, we can manually select some models and choose the model with the lowest deviance information criterion (DIC), which plays a similar role than BIC for Bayesian models.

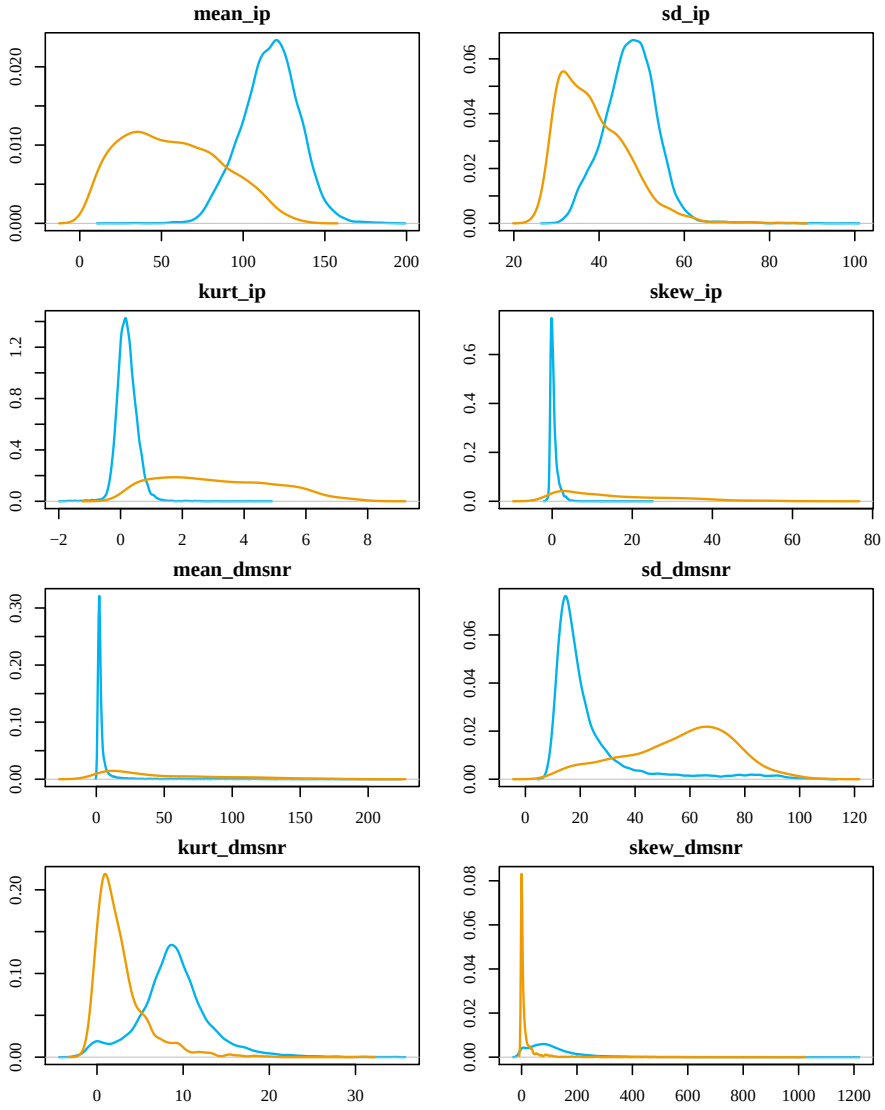

Figure 2: Gaussian kernel densities for each predictor conditioned on `is_pulsar`. Blue lines correspond to noise signals and gold lines correspond to pulsar signals.

3 Visual Analysis

Before computing any model, let us visualize the data. The gaussian kernel distributions for each predictor conditioned on `is_pulsar` can be seen in Figure 2. Observe that all predictors show different values depending whether they are noise or pulsar signals, thus in principle all variables can be relevant. Now consider the covariance matrix of Figure 3. We observe that there are two clear blocks in the matrix: variables related to the integrated profile and variables related to the DM-SNR curve. For each block most of the variables are heavily correlated, while the correlations for variables of different blocks are fairly small in general. Therefore, we might face problems of collinearity. A set of low-correlated variables is: `sd_ip`, `kurt_ip` (or `mean_ip`, or `skew_ip`), `skew_dmsnr`, `sd_dmsnr`. Notice as well that the integrated profile variables are more highly correlated to the `is_pulsar` variable.

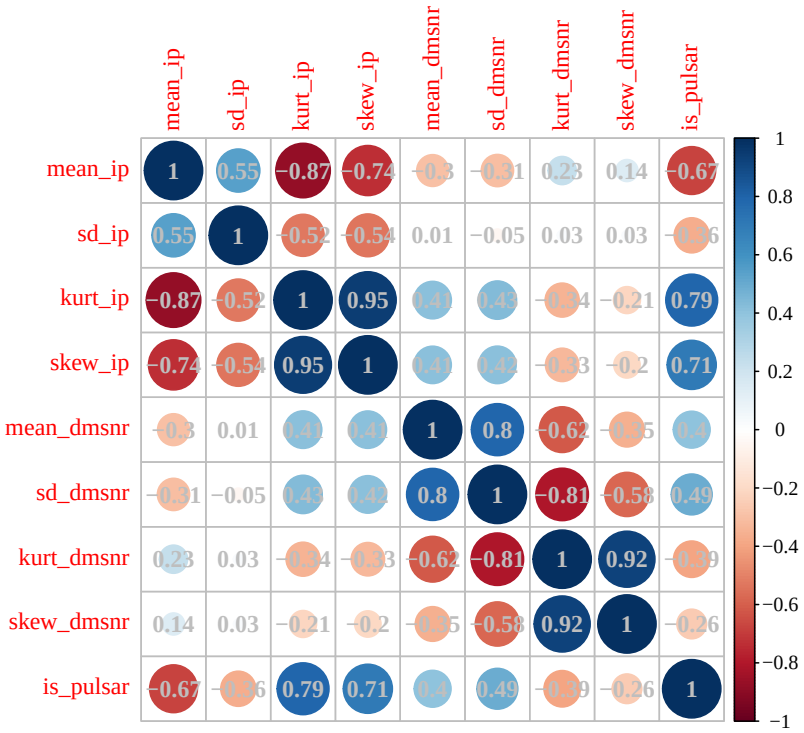

Figure 3: Covariance matrix for all variables.

4 Model Training and Testing

The `HTRU2` dataset has been splitted into a train and test dataset. The train dataset contains 75% of the total number of entries. Notice that the fraction of pulsars to noise signals is 10%, thus the dataset is imbalanced, though not very imbalanced. We have prepared two train datasets: one where we preserved this proportion and other where we set equal number of noise and pulsar instances (that is, we remove noisy instances). The removal of noisy instances causes information loss at the end, but the benefits of a balanced training dataset might be greater. The test dataset contains the other 25% data and the imbalanced problem is not corrected, as real data —like our dataset— contains much more noisy instances that actual pulsars, so it is a more realistic training scenario. The prediction is done by rounding the returned values for p: if $p \leq 0.5$, then the instance is classified as noise, and if $p > 0.5$, as a pulsar. From the results for p for our models we have seen that

most of the values are in $[0, 0.1] \bigcup [0.9, 1]$, thus we do not expect much improvement by setting another threshold for classification.

4.1 Classical Logistic Regression

We have trained several models with `glm`. All the trained models and their BICs can be seen in Table 1. The first one considers all predictors. The second one considers the set with low correlation mentioned in the previous section. Notice that, comparing with Figure 3, we have chosen those uncorrelated predictors with the highest correlation with respect `is_pulsar`. The third model is the results of applying `stepAIC` to either the first or the second model, as both return the same model. These three first models were computed with the non-balanced dataset. The three last models were trained with the balanced dataset. Notice that the *step bal* model is quite different from the *step* model. From these results we observe that the best model for the non-balanced dataset is the one selected by `stepAIC`, as it has the lowest BIC, while the best model for the balanced dataset is *step bal*. Notice that a model with all variables (no variable selection) is quite good, despite the collinearity.

The summary for the best models can be seen in the code block below. Notice that in both cases all predictors are relevant on the t-test.

```
> summary(glm_BIC)

Call:
glm(formula = is_pulsar ~ mean_ip + sd_ip + kurt_ip + skew_ip +
    mean_dmsnr + sd_dmsnr, family = "binomial", data = pulsars)

Deviance Residuals:
    Min      1Q  Median      3Q     Max
-4.4009 -0.1664 -0.1040 -0.0606 3.6152

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -9.693222  0.858963 -11.285  < 2e-16 ***
mean_ip      0.035597  0.006905   5.155 2.53e-07 ***
sd_ip       -0.037255  0.011693  -3.186  0.00144 **
kurt_ip      6.824440  0.352685  19.350  < 2e-16 ***
skew_ip     -0.641934  0.045187 -14.206  < 2e-16 ***
mean_dmsnr  -0.030932  0.003391  -9.122  < 2e-16 ***
sd_dmsnr     0.056840  0.004597  12.365  < 2e-16 ***
---

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 8223.0 on 13422 degrees of freedom
Residual deviance: 2008.5 on 13416 degrees of freedom
AIC: 2022.5

Number of Fisher Scoring iterations: 8

> summary(glm_BIC_bal)

Call:
glm(formula = is_pulsar ~ kurt_ip + skew_ip + mean_dmsnr + sd_dmsnr,
    family = "binomial", data = pulsars_bal)
```

```
Deviance Residuals:
   Min     1Q  Median     3Q    Max
-3.7228 -0.3256 0.0014 0.0450 3.0099

Coefficients:
          Estimate Std. Error z value Pr(>|z|)
(Intercept) -4.988820 0.240214 -20.768 < 2e-16 ***
kurt_ip   5.357376 0.379743 14.108 < 2e-16 ***
skew_ip  -0.497004 0.082462 -6.027 1.67e-09 ***
mean_dmsnr -0.029141 0.005214 -5.589 2.28e-08 ***
sd_dmsnr  0.061456 0.007110 8.644 < 2e-16 ***
---

(Dispersion parameter for binomial family taken to be 1)

   Null deviance: 3410.28 on 2459 degrees of freedom
Residual deviance: 884.95 on 2455 degrees of freedom
AIC: 894.95

Number of Fisher Scoring iterations: 9
```

Table 1
Predictors used in each frequentist model. The first four correspond to integrated profile statistics and the last four to DM-SNR statistics. The first three models use the entire dataset; the three last, the balanced one.

	mean	sd	kurt	skew	mean	sd	kurt	skew	BIC
All	✓	✓	✓	✓	✓	✓	✓	✓	2083.2
Low corr	✗	✓	✓	✗	✗	✓	✗	✓	2216.2
Step	✓	✓	✓	✓	✓	✓	✗	✗	2075.1
All bal	✓	✓	✓	✓	✓	✓	✓	✓	936.36
L.C. bal	✗	✓	✓	✗	✗	✓	✗	✓	961.01
Step bal	✗	✗	✓	✓	✓	✓	✗	✗	923.99

Choosing the *step* and *step bal* models, we compute the confusion table and the balanced accuracy (BAC). The BAC is the mean of the diagonal element of each column divided by the sum of values for each column. Said in a different way, it is the quotient of correct classifications for a given class divided by the number of instances for that class. This metric is useful for imbalanced datasets, as it provides the same weight to either class. We use this metric for both models. The test dataset is the same, so that meaningful comparisons can be done. The confusion table can be seen in Table 2. From these results the BAC is $0.916 (= \frac{1}{2}[0.995 + 0.836])$ for *step* and 0.953 for *step bal*. The results are fairly good, specially for classifying noise, as we expected. Notice that balancing causes a notable improvement in classification for the minority class. Therefore, we prefer the *step bal* model.

Table 2
Confusion tables for *step* and *step bal* models.

	Noise	Pulsar
Noise	4047	67
Pulsar	19	342

(a) *step* model.

	Noise	Pulsar
Noise	3974	29
Pulsar	92	380

(b) *step bal* model.

4.2 Bayesian Logistic Regression

The usage of `glm` and `stepAIC` is straightforward and simple, but the usage of `R2OpenBUGS` is not. `glm` was able to manage collinearity without raising errors, but when we tried to train the *all* model (unbalanced dataset) with `R2OpenBUGS` an error appeared. The error message did not provide literally any information on the problem. On the removal of the last two predictors to get the *step* model and keeping the exact same code (with the obvious modifications for the removal of predictors) it finally worked. Furthermore, if we train the *step* model with two Markov chains and only 100 iterations, it takes more than half hour [3], therefore with several thousand iterations we expect an execution time of the order of one day. Also, none of these errors have been found using the balanced dataset. Based on these empirical observations of `R2OpenBUGS` we conclude that this packages suffers when there is imbalanced data and collinearity between predictors.

Table 3
Predictors used in each Bayesian model. The first four correspond to integrated profile statistics and the last four to DM-SNR statistics. The first model uses the entire dataset; the two last, the balanced one.

	mean	sd	kurt	skew	mean	sd	kurt	skew	DIC
Step	✓	✓	✓	✓	✓	✓	✗	✗	2023
All bal	✓	✓	✓	✓	✓	✓	✓	✓	1197.0
Step bal	✗	✗	✓	✓	✓	✓	✗	✗	894.6

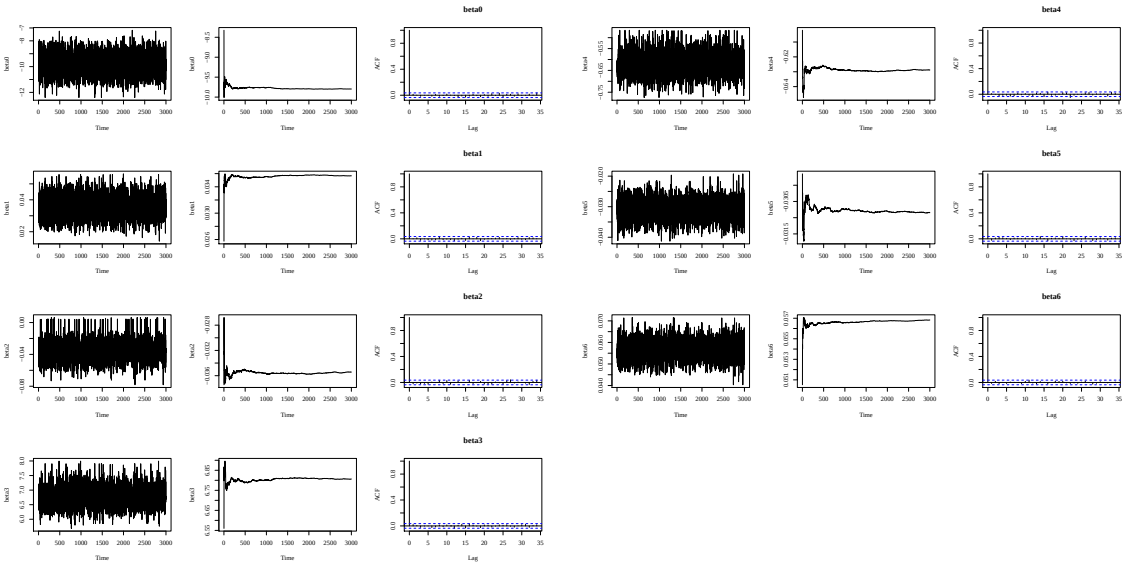

Figure 4: Bayesian *step* model convergence. For each coefficient β_i, there are three plots (left to right): series of values generated by MCMC, cumulative mean of the predictor, ACF of the predictor.

For the imbalanced dataset, due to the errors, we were not able to perform model selection using Ridge or Lasso regression. We have trained the *step* model with one Markov chain and 10000 iterations (7000 burn-in). We have also tested for models with all the combinations of two predictors, one for the integrated profile and one for the DM-SNR curve, so that there were surely no problems of collinearity. These two-predictors models

[3] The execution was halted after that time span.

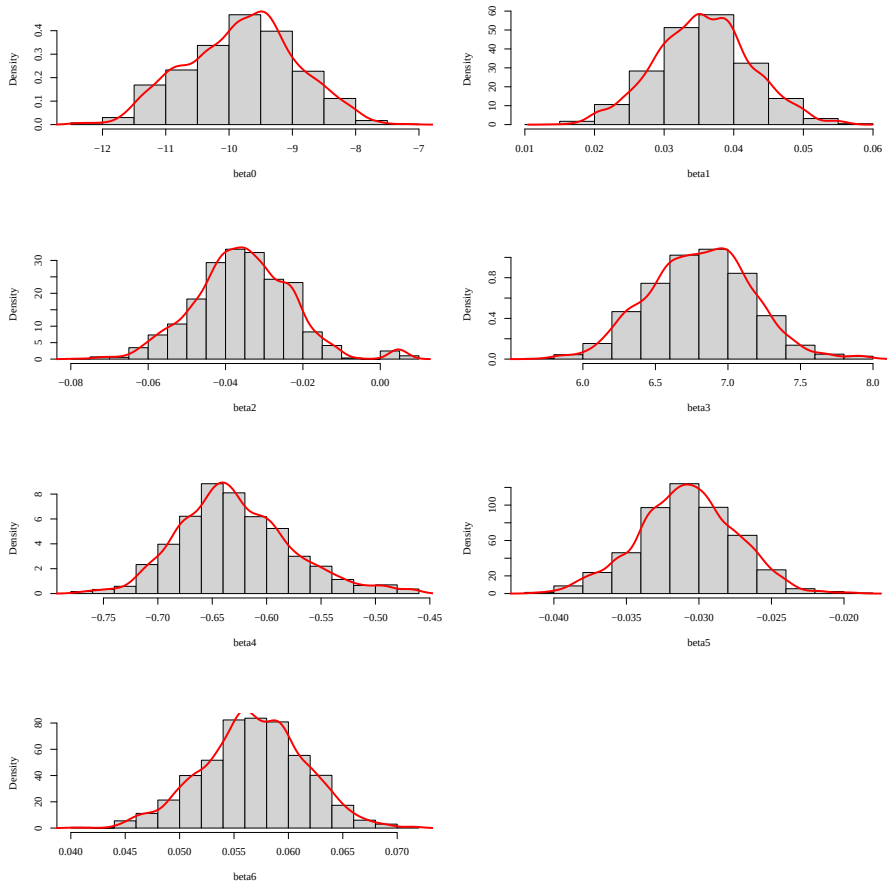

Figure 5: Posterior distributions for the coefficients of the Bayesian *step* model.

had a far greater DIC than the Bayesian *step* model. For this reason, we will focus only on the Bayesian *step* model.

For Bayesian *step* model we have chosen as priors $\beta_0 \sim N(-9, 0.1)$, $\beta_3 \sim N(7, 0.1)$, $\beta_4 \sim N(-1, 0.1)$, and $\beta_{1,2,5,6} \sim N(0, 0.1)$ (recall Figure 1). This selection of priors contains our previous knowledge on the coefficients: the means are the rounded values for the betas of the frequentist *step* model and the tolerances are small as we expect that the optimal values are around those means. From Figure 4 we observe that all coefficients have converged, as there are no high values for the autocorrelation function (ACF), the cumulative means do not show significant changes, and the series of coefficient values over the number of iterations show no pattern. From Figure 5 we observe the posterior distributions for the coefficients β. Notice that the distribution is almost zero at a $\sim 10\%$ distance from the mean. As none of the distributions are very skewed, the mean values of the distributions can be a good point estimator for the coefficients β,

$$\hat{\beta}_0 = -9.798$$
$$\hat{\beta}_1 = 0.03567, \hat{\beta}_2 = -0.03543, \hat{\beta}_3 = 6.806, \hat{\beta}_4 = -0.6288$$
$$\hat{\beta}_5 = -0.03085, \hat{\beta}_6 = 0.05683$$

Compare these estimations to the ones of the frequentist *step* on the code block: they are almost identical. Thus, we expect similar predictions, and, indeed, the confusion table is the same (see Table 2) and the same conclusions apply.

Considering now the balanced dataset, we have trained two models (see Table 3): one

with all predictors, and other with the predictors selected by the frequentist `stepAIC`. A model with all predictors can now be trained as it does not raise errors. For this model, double exponential (also known as Laplace) priors have been used, $\beta_i = \text{Laplace}(0, 2) \forall i$. The double exponential distribution is known to be equivalent to the frequentist Lasso regression. We have used Lasso and not Ridge in order to force non-relevant coefficients to be equal to zero (the first parameter is the mean and is set to zero). The second parameters measures the spread of the distribution. This parameter has been selected from the best value, 2, among several values between 1 and 4. We have done three Markov chains with 10000 iterations (7000 burn-in). From the results for this model (not shown here) we observe that no coefficient has been set exactly equal to zero and all coefficients, except for β_0 and β_6 contain the value zero in the 95% confidence intervals. Also, this model has a higher DIC compared to the Bayesian *step bal* model, so no further discussion is done.

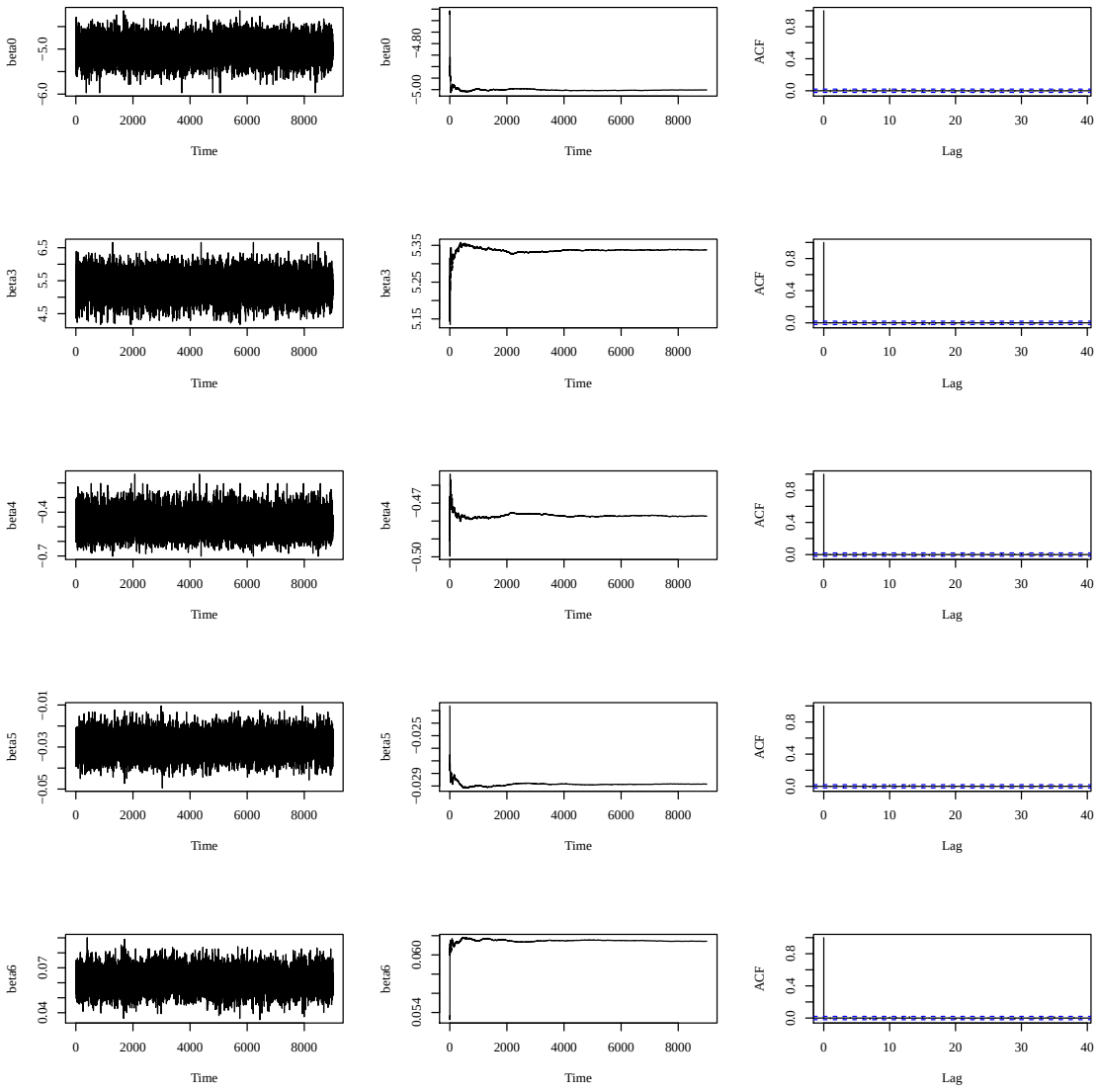

Figure 6: Bayesian *step bal* model convergence. For each coefficient β_i, there are three plots (left to right): series of values generated by MCMC, cumulative mean of the predictor, ACF of the predictor.

For the Bayesian *step bal* model, we have considered as priors $\beta_0 \sim N(-5, 0.1)$, $\beta_3 \sim$

$N(5, 0.1)$, $\beta_3 \sim N(0, 0.1)$, $\beta_4 \sim N(0, 0.1)$, $\beta_5 \sim N(0, 0.1)$, using our knowledge from the frequentist regression. We have used 3 chains with 10000 iterations (7000 burn-in). From Figure 6 we observe that the chain has converged, as it does not show any trend in the series of values for the betas, the cumulative mean is stable, and the ACF shows no spikes. The posterior distributions can be seen in Figure 7. As before, the distributions are not very skewed and the mean can be a good point estimator for the betas. The computed means are

$$\hat{\beta}_0 = -5.0014,$$
$$\hat{\beta}_3 = 5.3378, \hat{\beta}_4 = -0.47726,$$
$$\hat{\beta}_5 = -0.028849, \hat{\beta}_6 = 0.061420.$$

Comparing to the estimations for the frequentist *step bal* model, we observe that all predictors are very similar. The confusion table is identical to Table 2, except that there is one more good classification for noise. Therefore, the same comments can be done as in the frequentist case.

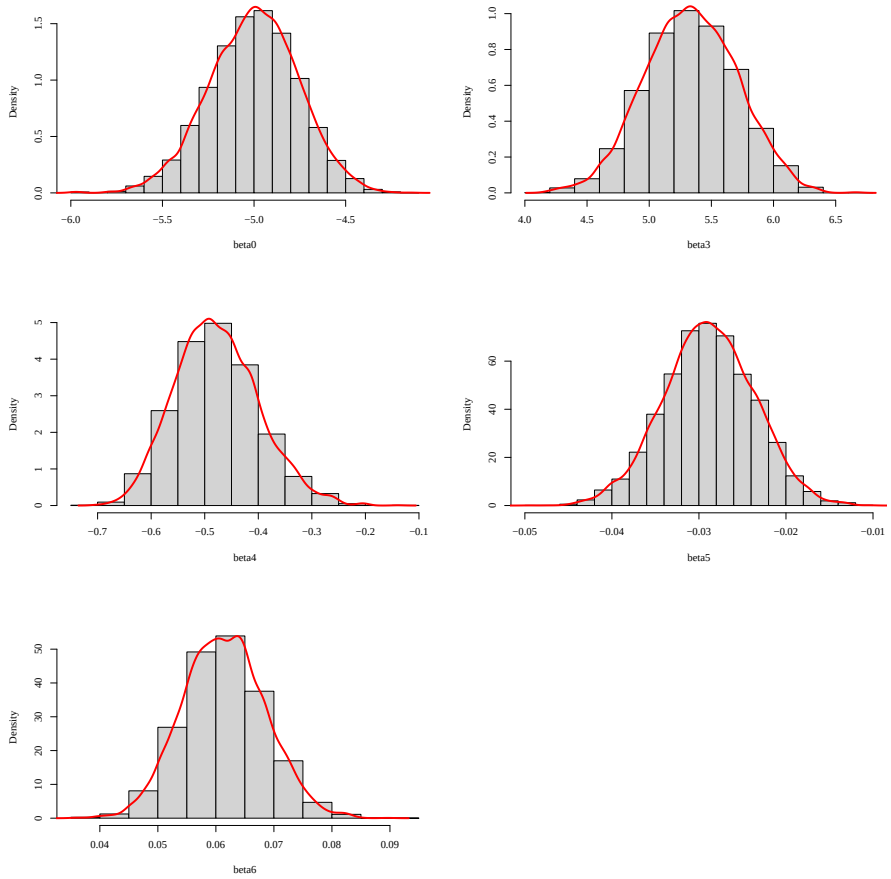

Figure 7: Posterior distributions for the coefficients of the Bayesian *step bal* model.

5 Conclusions

The dataset `HTRU2`, containing eight predictors with estatistics from the integrated profile and DM-SNR curve from possible pulsar candidates, has been used to produce logistic regressors to classify those candidates as noise or actual pulsars. The regressor with no

predictor selection, that is, with the eight predictors, does a good classification job because all variables have notable changes when conditioning on noise or pulsars. Nevertheless, there is large collinearity between variables. The `glm` function has selected a model with only six predictors, providing better a better BIC value. When using a balanced training dataset (in the original, 1:10 proportion), the `glm` selects a model with only four predictors. The balanced accuracy for the balanced dataset `glm` model is better than for the imbalanced dataset `glm` model, 0.953 vs 0.916, thus the correction for balancing the dataset provides good results. Considering Bayesian logistic regression from `R2OpenBUGS`, the two models with six and four predictors generate very similar point estimates and identical balanced accuracies. Other models, such as Bayesian Lasso regression with all predictors or two-model predictors, have also been explored, but return worse DICs. Therefore, both frequentist and Bayesian models provide nearly the same results. `glm` computes notably faster the results and handles model selection by BIC, while `R2OpenBUGS` is notably slower, more error-prone (with meaningless error messages), does not handle well collinearity and imbalanced data. Nevertheless, being a Bayesian approach, it provides a distribution for the fitting parameters, which can be useful for further analysis.