Regularized Logistic Regression using Cyclic Coordinate Descent

Binda Michal Kutak Wojciech Legczylin Michail

March 31, 2025

Abstract

This report presents the implementation of logistic regression with L1 penalty using the Cyclic Coordinate Descent (CCD) algorithm. The performance is compared with standard logistic regression on synthetic and real-world datasets using various evaluation metrics.

Contents

1 Introduction

Logistic regression is a statistical model that is used in classification tasks. Since it is part of supervised learning, based on training data (both features and target), the model learns to predict a certain outcome of limited variability. We will concentrate only on the binary case, as multiclass case can be iteratively achieved with binary model in the sense of predicting certain class vs. all the other classes.

Suppose we have $(x_i, y_i)_{i=1}^n, x_i \in \mathbb{R}^p, y_i \in \{0; 1\}$. x_i is p-dimensional vector of features and y_i is class that this observation belongs to. Values 1 and 0 correspond to positive and negative classes respectively.

Logistic regression assumes that the conditional probabilities of the target vector being a positive class depend on the affine transformation of features:

$$p(x) := \mathbb{P}(Y = 1|X = x) = \frac{1}{1 + e^{-(\beta_0 + x^T\beta)}},$$

where β_0, β are parameters of the model to be estimated.

We classify an observation as a positive class if $p(x) > \theta$, where θ is a hyperparameter, which means a certain threshold. Usually (and in our project) $\theta = 0.5$.

In order to estimate parameters of the model, we need to assess how well the model performs, and so we use a certain loss function. In our case, we use cross-entropy (or log-likelihood) loss:

$$l_i(\beta_0, \beta) = y_i \cdot \ln p(x_i) + (1 - y_i) \cdot \ln(1 - p(x_i)),$$

where l_i is cross-entropy loss value of i-th observation, y_i is true class value of i-th observation and $p(x_i)$ is estimated probability of i-th observation to be of positive class. We explicitly write, that β_0, β are arguments of loss function as these are what we want to change. Note, that if $y_i = 1$, than we want $p(x_i)$ to be as close to 1 as possible and when $y_i = 0$, we want $p(x_i)$ be as close to 0 as possible, and so no matter what the value of y_i is, we want to maximize l_i.

Average value of cross-entropy loss we denote as:

$$R(\beta_0, \beta) = \frac{1}{n} \sum_{i=1}^{n} l_i(\beta_0, \beta)$$

and name risk. Based on the same logic, in order to find the best model parameters, we want to maximize R. Thus:

$$(\beta_0^*, \beta^*) = \underset{(\beta_0,\beta)\in\mathbb{R}^{p+1}}{\operatorname{argmax}} R(\beta_0, \beta).$$

Regularization In order to control values of model parameters we use regularization. Some of them:

- L1 (lasso) $P(\beta) = ||\beta||_1 = \sum_{i=1}^n |\beta_i|$
- L2 (ridge) $P(\beta) = ||\beta||_2^2 = \sum_{i=1}^n \beta_i^2$
- Elastic-Net $P_\alpha(\beta) = (1 - \alpha) \cdot ||\beta||_2^2 + \alpha \cdot ||\beta||_1$, $\alpha \in [0; 1]$

And then what we want to optimize is

$$(\beta_0^*, \beta^*) = \underset{(\beta_0,\beta)\in\mathbb{R}^{p+1}}{\operatorname{argmax}} \left[R(\beta_0, \beta) + \lambda P(\beta)\right],$$

where λ and α (if Elastic-Net was used) are hyperparameters. In our project we implemented Elastic-Net, so all three mentioned regularization technics are possible.

2 Methodology

2.1 Criteria for real datasets

The selection of real datasets was based on the following criteria:

- The task must be binary classification.
- The dataset should contain only numerical features.
- The number of features should be at least 50% of the number of observations.

2.2 Sources used (UCI, OpenML)

We used four real-world datasets from OpenML, focusing on high-dimensional binary classification tasks:

- **AP_Colon_Kidney** (ID: 1137) – gene expression data for colon and kidney tumors.
- **AP_Breast_Kidney** (ID: 1158) – gene expression data for breast and kidney tumors.
- **Toxicity** (ID: 46611) – molecular descriptors for predicting toxicity towards CRY1 protein.
- **DLBCL** (ID: 45088) – gene-expression data for predicting outcomes in Diffuse Large B-cell Lymphoma patients.

All datasets are high-dimensional and suitable for benchmarking regularized logistic regression.

2.3 Preprocessing: Missing Values and Collinearity

Before conducting experiments, all datasets underwent a uniform preprocessing procedure to handle missing values and reduce multicollinearity. Numerical features with missing values were imputed using column-wise means. To address collinearity, we calculated the absolute Pearson correlation matrix for all features and removed variables that had a correlation higher than 0.9 with any other feature.

Table 1 presents a summary of the datasets before and after preprocessing. All datasets met the required condition that the number of features is at least 50% of the number of observations. After removing collinear features, the dimensionality of the datasets was significantly reduced, while retaining sufficient variability for modeling.

Table 1: Summary of datasets before and after preprocessing

Dataset	Obs.	Orig. Feat.	After Prep.	Removed	Binary
Toxicity (46611)	171	1203	477	726	Yes
DLBCL (45088)	77	5469	5197	272	Yes
AP Colon-Kidney (1137)	546	10935	9548	1387	Yes
AP Breast-Kidney (1158)	604	10935	9501	1434	Yes

2.4 Synthetic Data Generation

The synthetic dataset is generated using the following procedure:

1. A binary class variable $Y \in \{0, 1\}$ is sampled from a Bernoulli distribution with class prior probability p.

2. For each class:

   - If $Y = 0$, the feature vector $\mathbf{X} \mid Y = 0$ follows a d-dimensional multivariate normal distribution with:
     - Mean vector: $\boldsymbol{\mu}_0 = (0, 0, \ldots, 0)$
     - Covariance matrix: $S[i, j] = g^{|i-j|}$
   - If $Y = 1$, the feature vector $\mathbf{X} \mid Y = 1$ follows a d-dimensional multivariate normal distribution with:
     - Mean vector: $\boldsymbol{\mu}_1 = \left(1, \frac{1}{2}, \frac{1}{3}, \ldots, \frac{1}{d}\right)$
     - Covariance matrix: $S[i, j] = g^{|i-j|}$

3. A total of n observations are generated by repeating the above steps.

The resulting dataset consists of a feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ and a binary target vector $\mathbf{y} \in \{0, 1\}^n$.

2.5 Algorithm Implementation

The main concern lies in implementation of seeking for optimal parameters when it comes to estimating them, because in contrast to linear regression case, we do not have explicit formula here. Thus we need to use some methods of iterative search for optimum. In this project we used Cyclic Coordinate Descent algorithm as defined in *Journal of Statistical Software*.

Part 1. Linear Regression Suppose we have $(x_i, y_i)_{i=1}^n, x_i \in \mathbb{R}^p, y_i \in \mathbb{R}$. We assume that $\mathbb{E}[Y|X = x] = \beta_0 + x^T\beta$. We want:

$$(\beta_0^*, \beta^*) = \underset{(\beta_0,\beta)\in\mathbb{R}^{p+1}}{\operatorname{argmin}} \left[\frac{1}{2n} \sum_{i=1}^{n} (y_i - \beta_0 - x_i^T\beta)^2 + \lambda P_\alpha(\beta) \right].$$

Assuming that $\tilde{\beta_0}^{(k)}, \tilde{\beta}^{(k)}$ are estimates of β_0, β in k-th step, we update $\tilde{\beta}_j$ based on coordinate descent method, that is:

$$\tilde{\beta}_j^{(k+1)} = \frac{S(\frac{1}{n}\sum_{i=1}^n x_{ij}(y_i - \tilde{y}_i^{(j)}), \lambda\alpha)}{1 + \lambda(1-\alpha)},$$

where

- $\tilde{y}_i^{(j)} = \tilde{\beta}_0^{(k)} + \sum_{l \neq j} x_{il}\tilde{\beta}_l^{(k)}$
- $S(z, \gamma) = \text{sign}(z)(|z| - \gamma)_+$

Part 2. Logistic Regression Suppose we have $(x_i, y_i)_{i=1}^n, x_i \in \mathbb{R}^p, y_i \in \{0; 1\}$. We assume that $\mathbb{P}(Y = 1|X = x) = \frac{1}{1+e^{-(\beta_0 + x^T\beta)}}$. We want:

$$(\beta_0^*, \beta^*) = \underset{(\beta_0,\beta)\in\mathbb{R}^{p+1}}{\operatorname{argmax}} \left[\frac{1}{n} \sum_{i=1}^{n} (y_i \cdot \ln p(x_i) + (1 - y_i) \cdot \ln(1 - p(x_i))) - \lambda P_\alpha(\beta) \right].$$

Assuming that $\tilde{\beta_0}^{(k)}, \tilde{\beta}^{(k)}$ are estimates of β_0, β in k-th step, after approximating log-likelihood part with its quadratic approximation we get:

$$(\beta_0^*, \beta^*) = \underset{(\beta_0,\beta)\in\mathbb{R}^{p+1}}{\operatorname{argmax}} \left[-\frac{1}{2n} \sum_{i=1}^{n} w_i(z_i - \beta_0 - x_i^T\beta)^2 + C(\tilde{\beta}_0^{(k)}, \tilde{\beta}^{(k)}) - \lambda P_\alpha(\beta) \right],$$

where

- $\tilde{p}(x_i) = \frac{1}{1+e^{-\left(\tilde{\beta}_0^{(k)} + x_i^T \tilde{\beta}^{(k)}\right)}}$
- $w_i = \tilde{p}(x_i)(1 - \tilde{p}(x_i))$
- $z_i = \tilde{\beta}_0^{(k)} + x_i^T \tilde{\beta}^{(k)} + \frac{y_i - \tilde{p}(x_i)}{w_i}$

And thus updates for parameters are as follows:

$$\tilde{\beta}_j^{(k+1)} = \frac{S(\frac{1}{n}\sum_{i=1}^n w_i x_{ij}(z_i - \tilde{z}_i^{(j)}), \lambda\alpha)}{1 + \lambda(1-\alpha)},$$

where $\tilde{z}_i^{(j)} = \tilde{\beta}_0^{(k)} + \sum_{l \neq j} x_{il}\tilde{\beta}_l^{(k)}$.

Optimizations Several computational optimizations were implemented:

- We do not actually need to calculate z_i or $\tilde{z}_i^{(j)}$, as after expansion expression $w_i x_{ij}(z_i - \tilde{z}_i^{(j)})$ equals $x_{ij}(y_i - \tilde{p}(x_i)) + w_i x_{ij}^2 \tilde{\beta}_j^{(k)}$. And so whole first argument passed into $S()$ function computes as $< \frac{1}{n}(y - \tilde{p}(x) + \tilde{\beta}_j^{(k)} w \cdot x_{\cdot j}); x_{\cdot j} >$, where $<;>$ denotes scalar product, $a \cdot b$ denotes element-wise product of vectors, $x_{\cdot j}$ denotes j-th column of x.
- Instead of $\frac{1}{n}$ we can associate different, often observation-dependent, weights. In that case the update formula changes to:

  $$\tilde{\beta}_j^{(k+1)} = \frac{S(\sum_{i=1}^n v_i w_i x_{ij}(z_i - \tilde{z}_i^{(j)}), \lambda\alpha)}{\sum_{i=1}^n v_i x_{ij}^2 + \lambda(1-\alpha)},$$

  where w_i are weights of logistic regression model being appropriately changed into linear regression and v_i are weights of coordinate descent optimization.
- Rather than computing solutions for constant value of penalty parameter λ, we generate decreasing sequence of λ in such a way, that $\lambda_{\max}$ is the smallest value of penalty parameter that causes all $\tilde{\beta} = 0$. It can be shown that for logistic regression case, $\alpha\lambda_{\max} = \max_j |\sum_{i=1}^n v_i x_{ij}(y_i - \tilde{p}(x_i))|$. Strategy proposed in the article is to select a minimum value $\lambda_{\min} = \varepsilon\lambda_{\max}$ and construct sequence of K values of λ decreasing from $\lambda_{\max}$ to $\lambda_{\min}$ on the logarithmic scale. Common values are $\varepsilon = 0.001$ and $K = 100$.

Implementation We implemented class `LogRegCCD` that with how it is utilized resembles `LogisticRegression` class in `scikit-learn` package.

Class possesses the following attributes:

- `beta0_` — the value corresponding to intercept or, as mentioned in theoretical part of this report, $\tilde{\beta}_0$,
- `beta_` — vector of values corresponding to coefficients of features, $\tilde{\beta}$,
- `C_` — inverse of penalty parameter,
- `alpha_` — value of L1-ratio in Elastic-Net, α.

Class possesses the following methods:

- `fit()` — takes training data as well as parameters `max_iter` (corresponds to K in generating sequence of λ), `fit_intercept` (specifies if model should use free expression or set it to zero by default), `use_weights` (allows to use weights optimization). Method works in the following flow:

  - standarize features,
  - generate sequence of λ and start outer loop iterating over this sequence,
  - update log-likelihood approximation, i.e. calculate $\tilde{p}(x_i), w_i$,
  - start inner loop iterating over $\tilde{\beta}$ values,
  - based on formula update $\tilde{\beta}_j$,
  - conclude inner loop,
  - conclude outer loop.

- `validate()` — takes validating data and classification measure. Predicts class labels or probabilities of belonging to class 1 (depends on selected measure) and supports them to measure. Returns value of measure,
- `predict_proba()` — takes features data and applies sigmoid function on them, using current parameters estimates,
- `predict()` — takes features data and returns predicted class lables,
- `soft_threshold()` — takes two parameters and calculates soft-threshold for them as described in *Part 1. Linear Regression*,
- `sigmoid()` — applies sigmoid function on given numeric argument,
- `gaussian()` — applies cumulative distribution function of Normal (Gaussian) distribution on given numeric argument,
- `plot()` — for given training and validating data, as well as specified measures, visualize how these measures change based on value of penalty parameter. Penalty parameter sequence is set automatically based on training data,
- `plot_coefficient()` — takes training data and visualizes values of coefficients of model based on value of penalty parameter. Penalty parameter sequence is set automatically based on training data.

3 Correctness of LogRegCCD

3.1 Convergence analysis

Figures (1) and (2) show that implemented algorithm converges to certain optimum.

3.2 Comparison of not regularized models

Figures (3), (4), (5), (6) show performence comparison of LogRegCCD and LogisticRegression models without regularization.

4 Experimental Results

The performance of the custom implementation of the regularized logistic regression using cyclic coordinate descent LogRegCCD was compared with the scikit-learn implementation of LogisticRegression without regularization. The experiments were conducted on both synthetic and real ones.

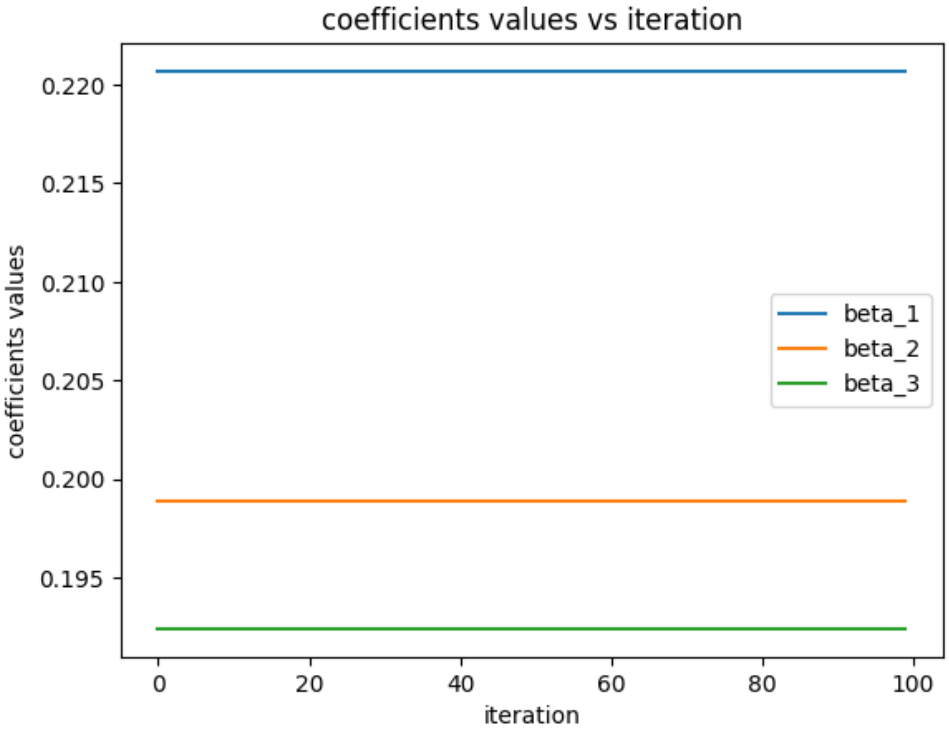

Figure 1: Coefficient values of LogRegCCD model plotted against iteration.

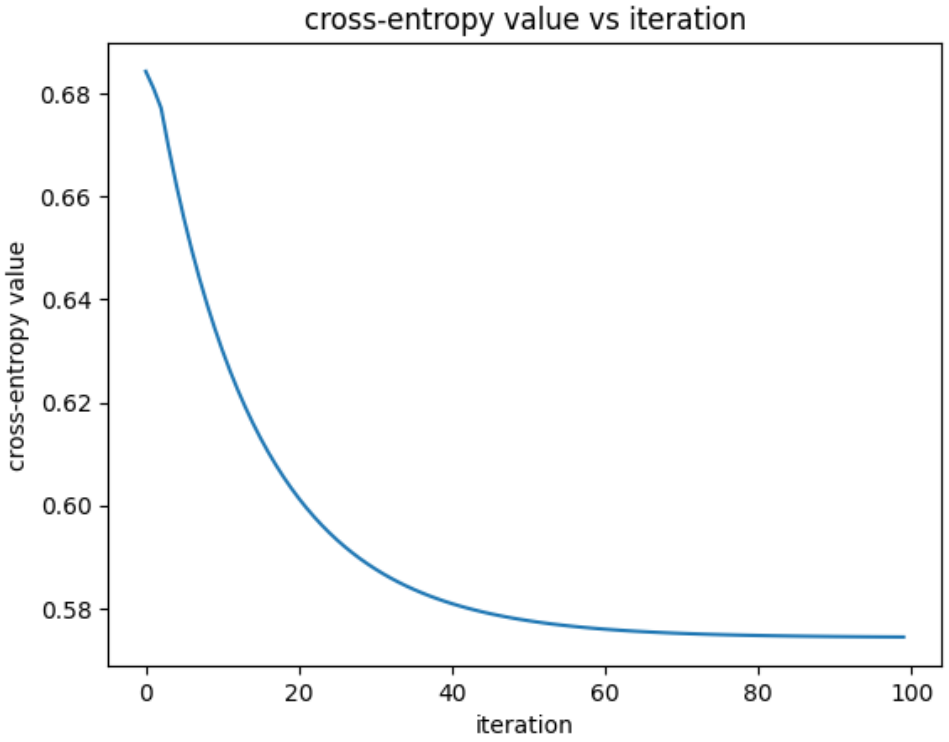

Figure 2: Cross-entropy values of LogRegCCD model plotted against iteration.

4.1 Synthetic Datasets

In order to test performance and robustness of custom implementation, synthetic datasets were generated with the procedure described in (2.4). The impact of different parameters of the

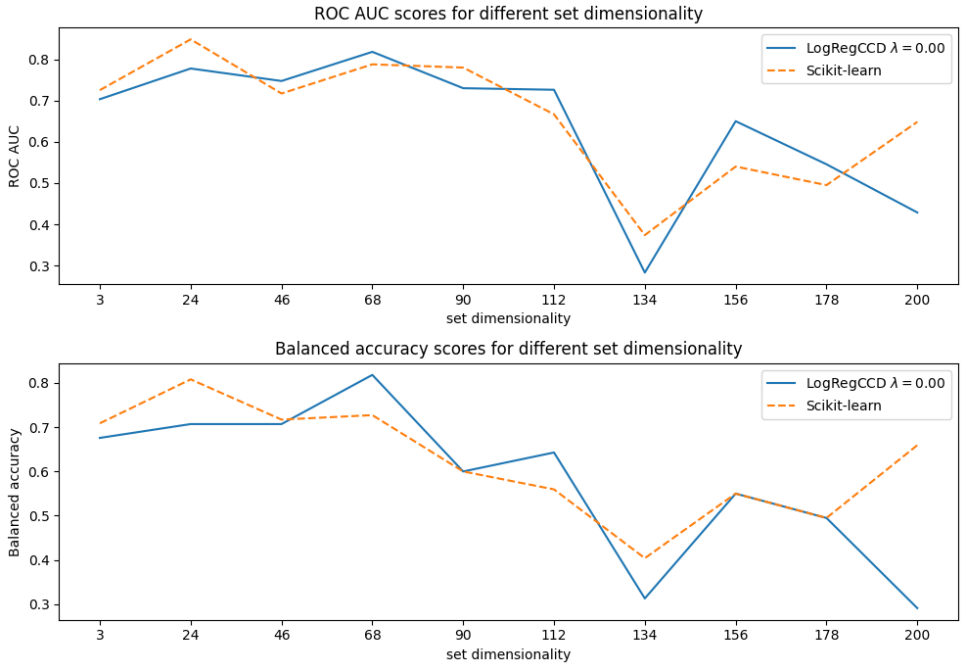

Figure 3: Performance of LogRegCCD with $\lambda = 0$ against logistic regression w.r.t. set dimensionality

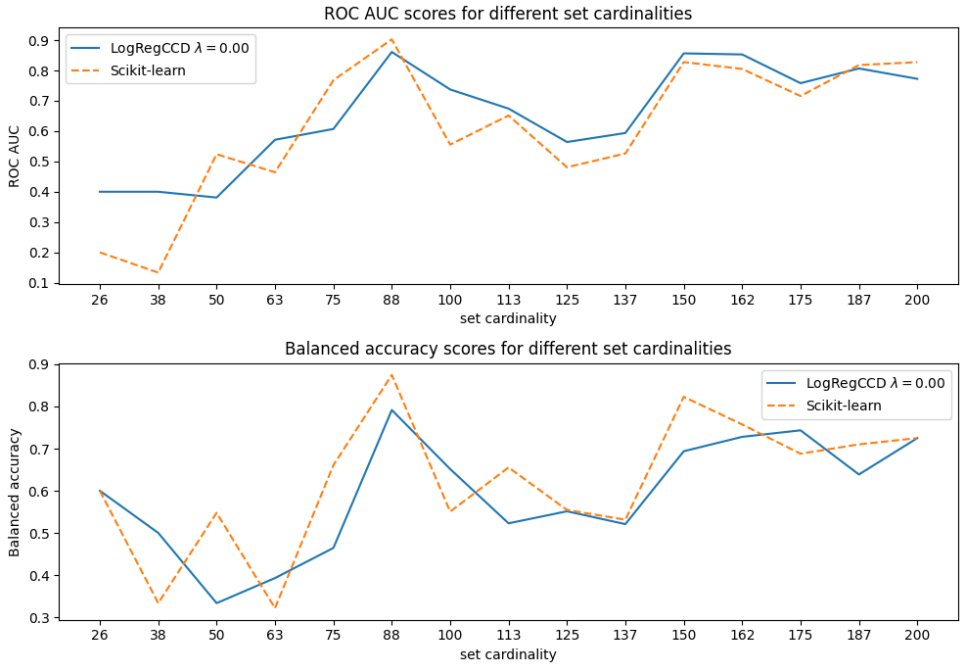

Figure 4: Performance of LogRegCCD with $\lambda = 0$ against logistic regression w.r.t. set cardinality

distributions was explored. The balanced average and ROC AUC metrics were used to measure the performance of the model in a given data set against the regularization parameter

$$\lambda = \{\lambda_1 = 0.1, \lambda_2 = 1.55, \lambda_3 = 3\}.$$

For each data set, 4 models were trained: 3 models based on custom implementation with different regularization parameters (where i-th model is regularized by λ_i), and one model based on the Scikit-Learn implementation of logistic regression.

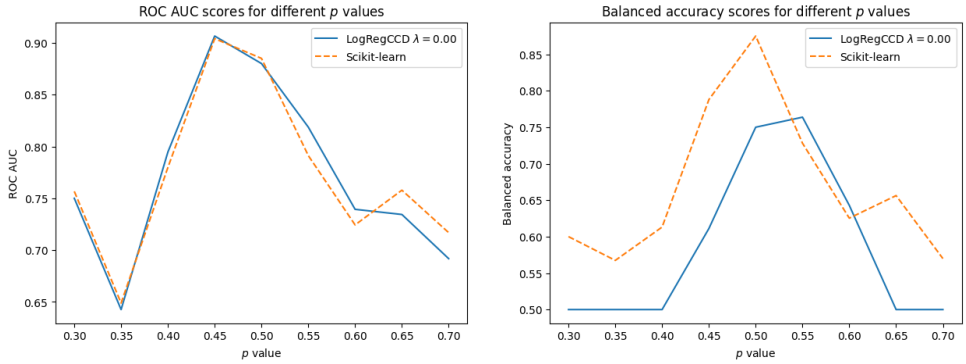

Figure 5: Performance of LogRegCCD with $\lambda = 0$ against logistic regression w.r.t. class balance parameter.

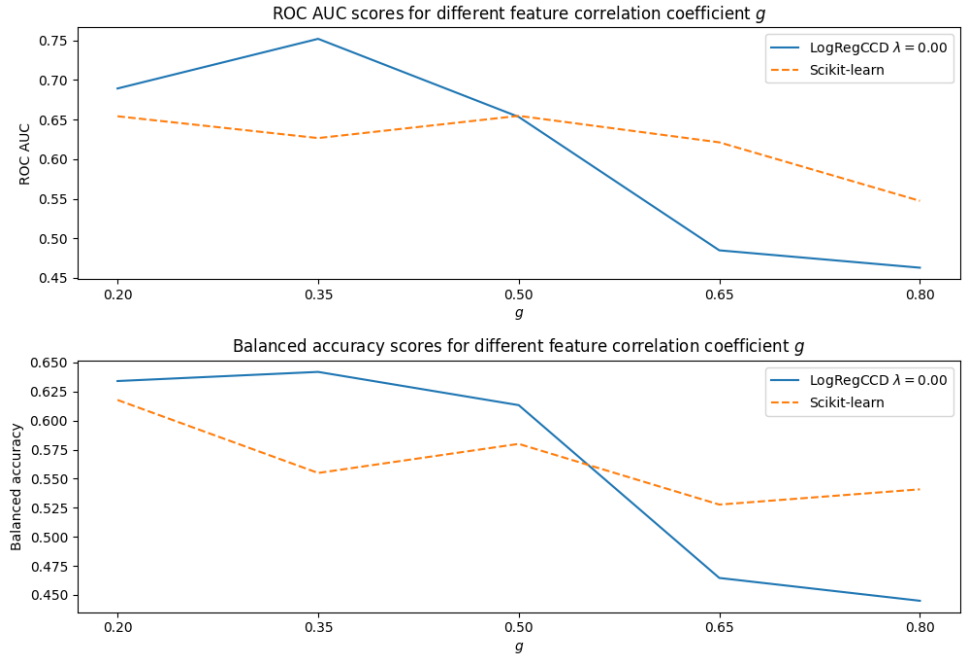

Figure 6: Performance of LogRegCCD with $\lambda = 0$ against logistic regression w.r.t. feature correlation.

4.1.1 Cardinality

To test the performance of the model w.r.t. to the cardinality parameter n of the dataset, 15 synthetic datasets $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_{15}\}$ were drawn from the distribution described in (2.4). The distribution was parametrized by: $p = 0.5$, $g = 0.2$, $d = 25$. Cardinality of the i-th set is equal to

$$|\mathcal{D}_i| = N_i = 26 + (200 - 26)(i - 1).$$

Performance results are shown in figure (7).

Takeouts Models with stronger regularization have substantially lower metrics values when set cardinality is sufficiently close to set dimensionality.

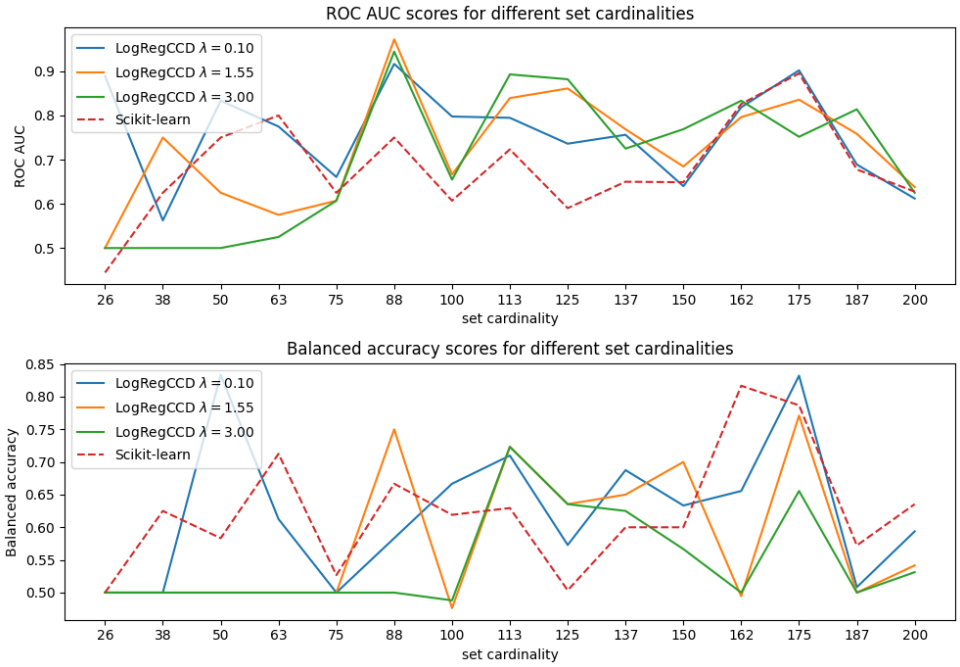

Figure 7: Performance of regularized LogRegCCD and not regularized LogisticRegression implementation on synthetic datasets w.r.t. set cardinality.

4.1.2 Feature correlation

To test the performance of the model w.r.t. to the feature correlation parameter g of the dataset, 5 synthetic datasets $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_5\}$ were drawn from the distribution described in (2.4). The distribution was parametrized by: $p = 0.5$, $d = 100$ and $n = 200$. Feature correlation of the i-th set is equal to

$$g_i = 0.2 + (0.8 - 0.2)(i - 1).$$

Performance results are shown in figure (8).

Takeouts Regularized models outperform not regularized model, especially when features are moderately correlated, $p \approx 0.5$.

4.1.3 Dimensionality

To test the performance of the model w.r.t. to the dimensionality parameter d of the dataset, 10 synthetic datasets $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_{10}\}$ were drawn from the distribution described in (2.4). The distribution was parametrized by: $p = 0.5$, $g = 0.0$ and $n = 200$. Dimensionality of the i-th set is equal to

$$d_i = 3 + (200 - 3)(i - 1).$$

Performance results are shown in figure (9).

4.1.4 Class balance

To test the performance of the model w.r.t. to the class balance parameter p of the dataset, 9 synthetic datasets $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_9\}$ were drawn from the distribution described in (2.4). The distribution was parametrized by:$g = 0.2$, $d = 5$ and $n = 200$. Parameter p of the i-th set is equal to

$$p_i = 0.3 + (0.7 - 0.3)(i - 1).$$

Performance results are shown in figure (10).

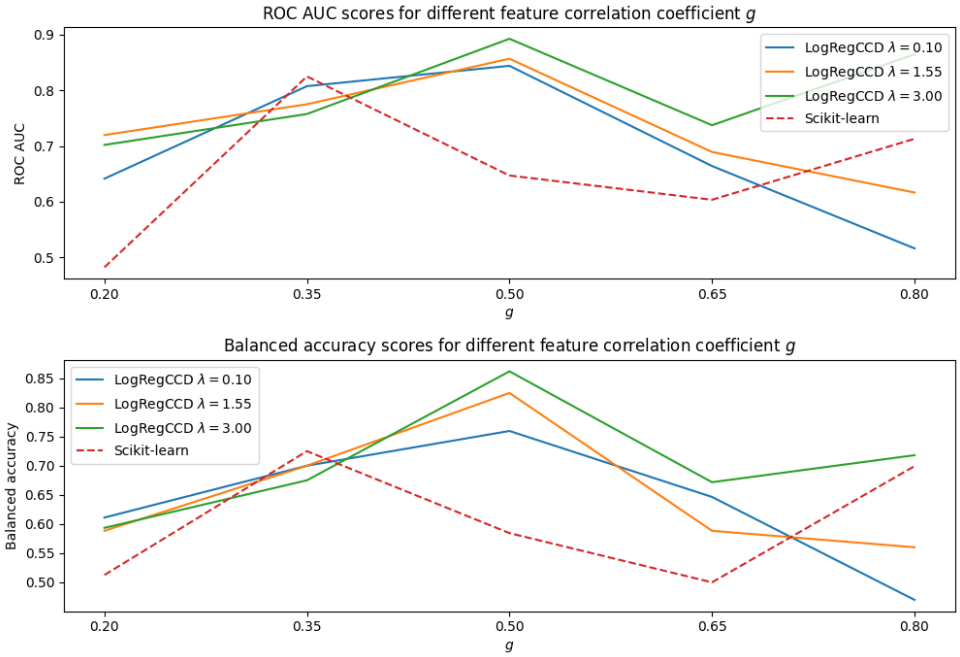

Figure 8: Performance of regularized LogRegCCD and not regularized LogisticRegression implementation on synthetic datasets w.r.t. feature correlation.

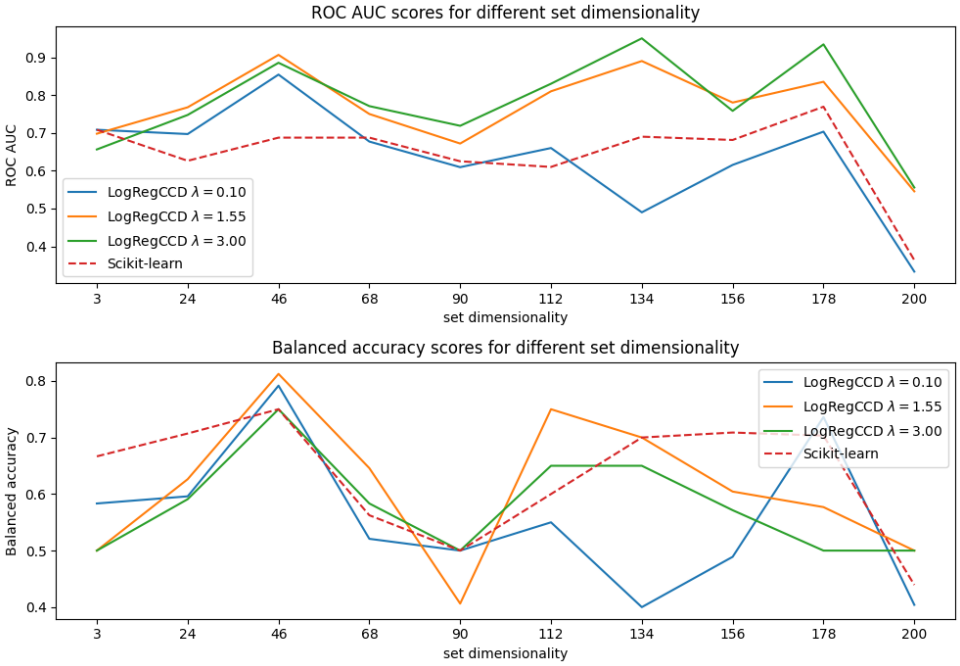

Figure 9: Performance of regularized LogRegCCD and not regularized LogisticRegression implementation on synthetic datasets w.r.t. set dimensionality.

4.2 Real Datasets

Performance metrics and coefficient comparison was performed on real datasets described in section (2.2). The results are shown in figures (11), (12), (13), (14), (16), (15)(17) and (18).

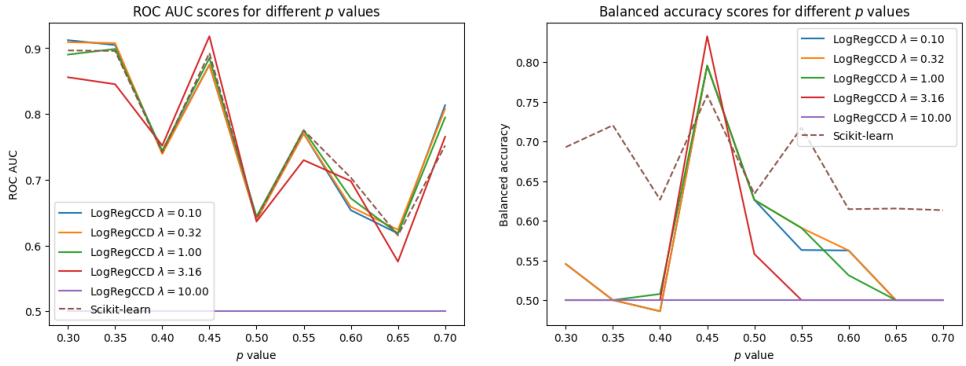

Figure 10: Performance of regularized LogRegCCD and not regularized LogisticRegression implementation on synthetic datasets w.r.t. set class balance.

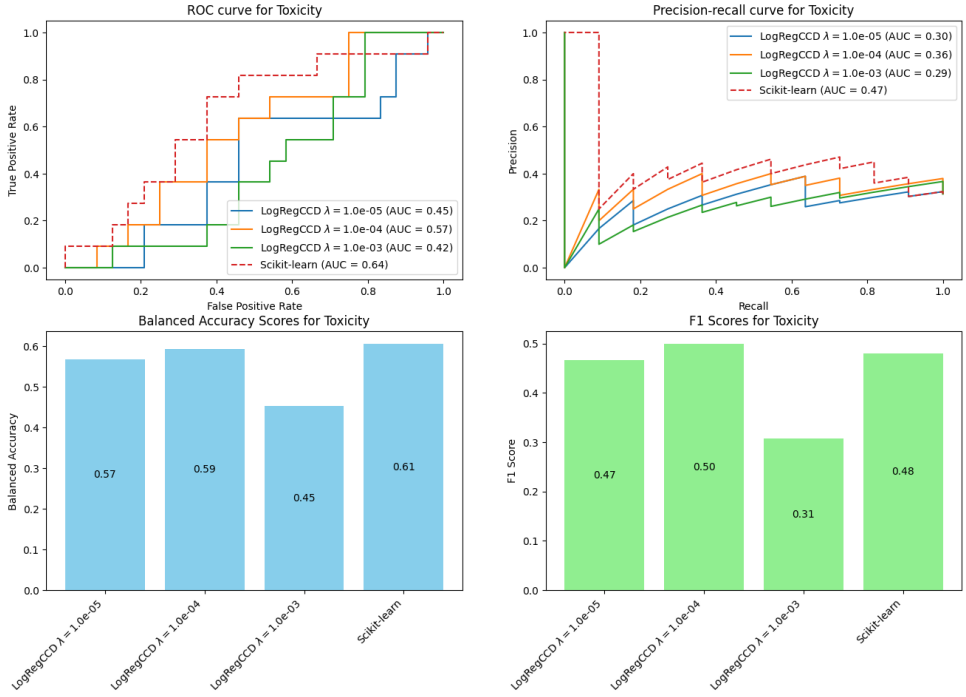

Figure 11: ROC AUC, Precision-Recall AUC, Balanced accuracy and F1 score of the models trained on Toxicity dataset.

5 Discussion and Conclusions

- Summary of key findings
- Strengths and limitations of LogRegCCD
- Potential improvements

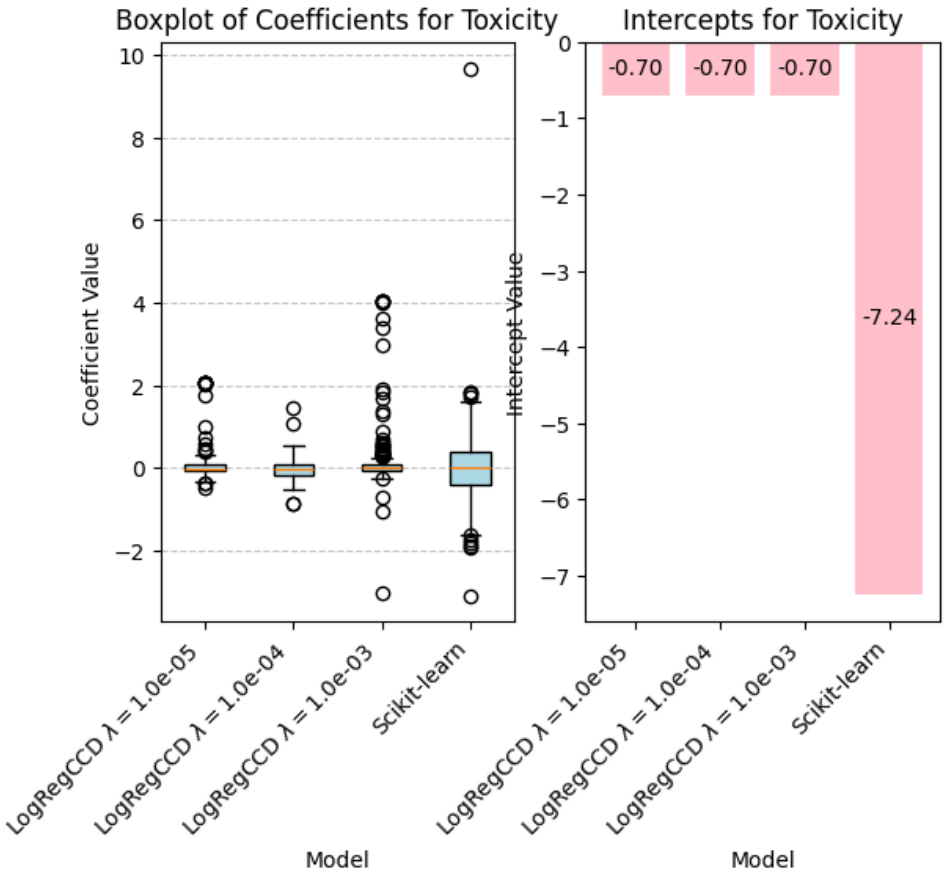

Figure 12: Coefficients and intercepts of models trained on Toxicity dataset.

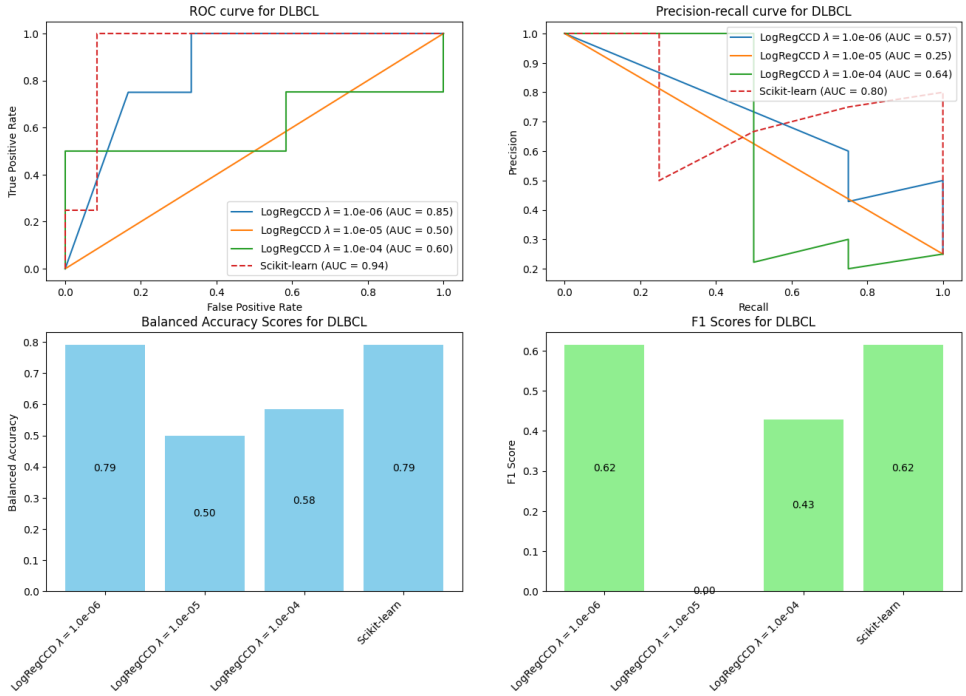

Figure 13: ROC AUC, Precision-Recall AUC, Balanced accuracy and F1 score of the models trained on DLBCL dataset.

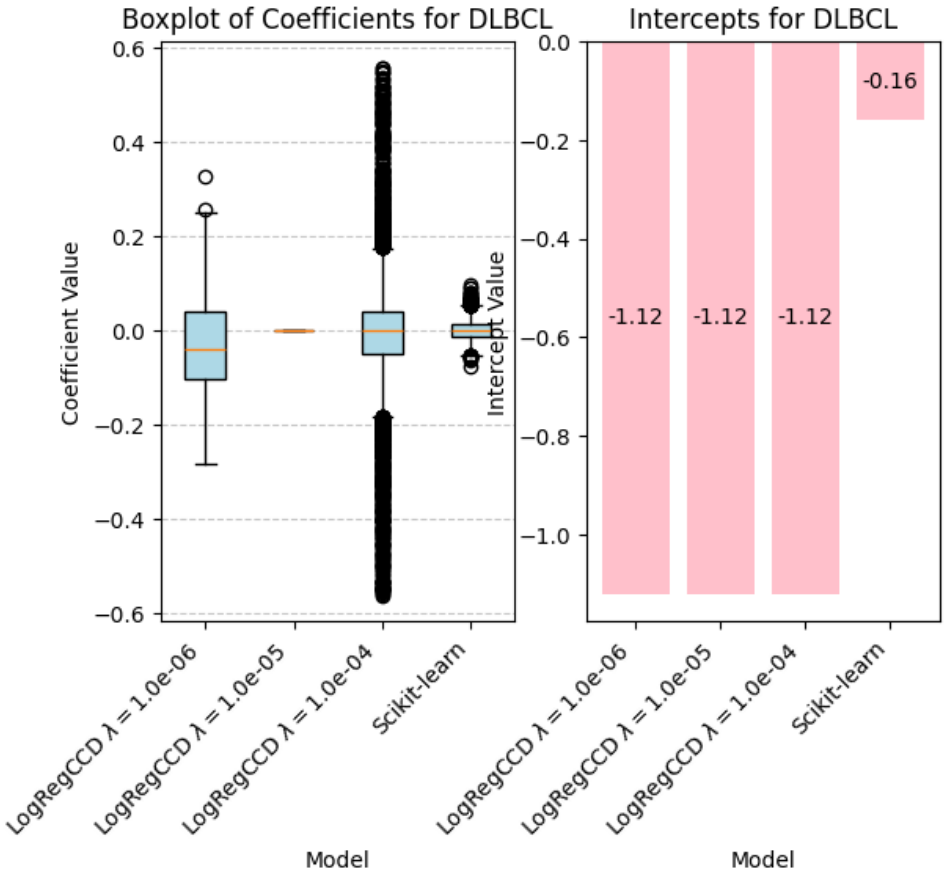

Figure 14: Coefficients and intercepts of models trained on DLBCL dataset.

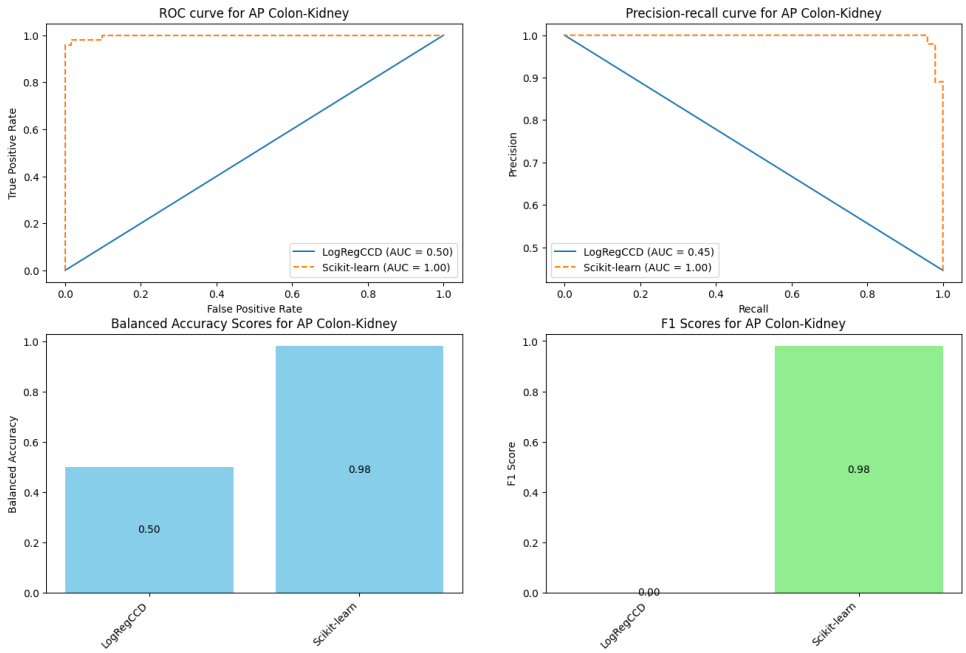

Figure 15: ROC AUC, Precision-Recall AUC, Balanced accuracy and F1 score of the models trained on AP Colon-Kidney dataset.

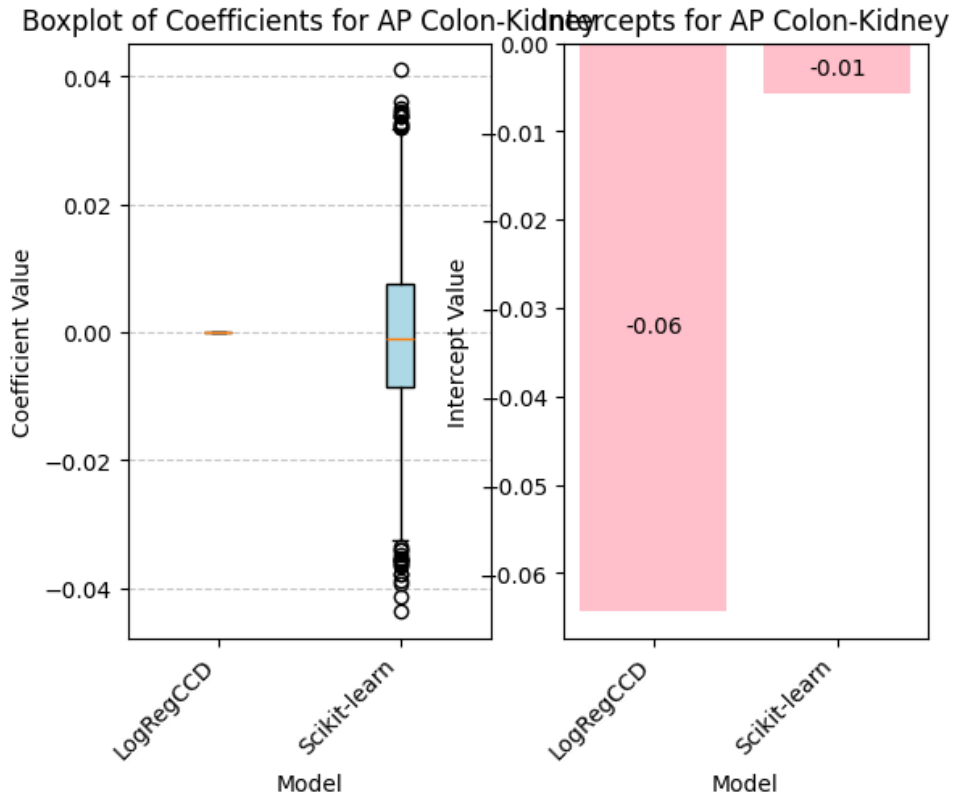

Figure 16: Coefficients and intercepts of models trained on AP Colon-Kidney dataset.

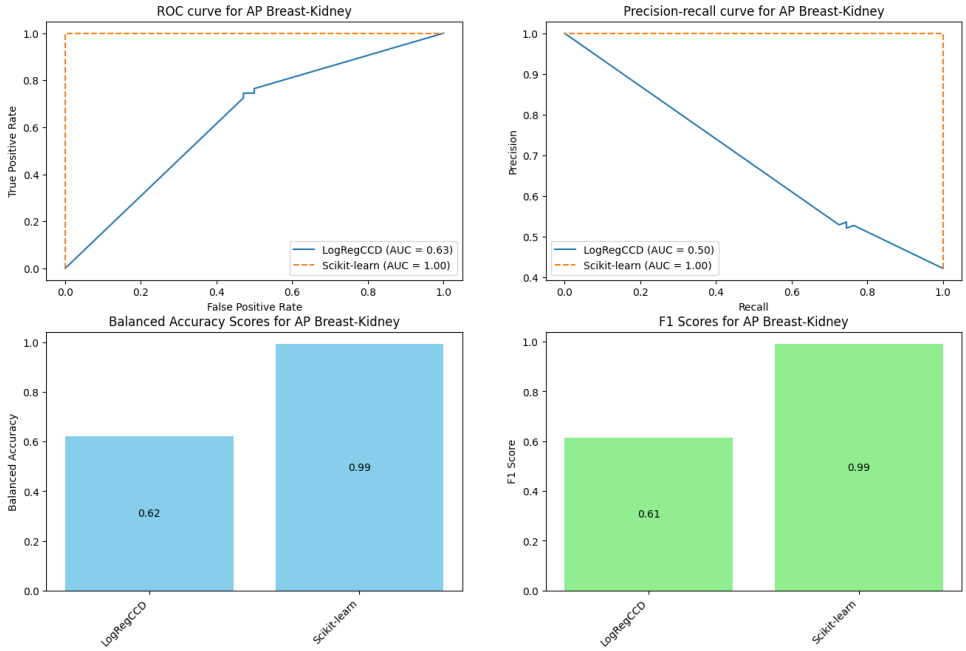

Figure 17: ROC AUC, Precision-Recall AUC, Balanced accuracy and F1 score of the models trained on AP Breast-Kidney dataset.

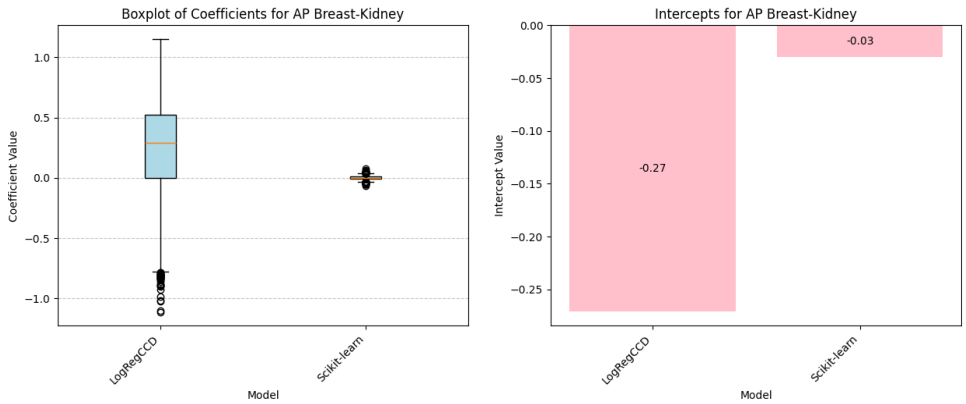

Figure 18: Coefficients and intercepts of models trained on AP Breast Kidney dataset.

References

- Genkin, A., Lewis, D. D., Madigan, D. (2007). Large-scale Bayesian logistic regression for text categorization. *Journal of Statistical Software*, `https://www.jstatsoft.org/article/view/v033i01`
- scikit-learn documentation: `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`
- Dataset sources: `https://archive.ics.uci.edu/`, `https://www.openml.org/`