ADVANCED MACHINE LEARNING - PROJECT 1

Klaudia Gruszkowska, Zofia Łągiewka, Jacek Zalewski
Faculty of Mathematics and Information Science
Warsaw University of Technology

April 3, 2024

Contents

1 Introduction

The report documents the first project from the Advanced Machine Learning course. The main goal of this project was to implement optimization algorithms for parameter estimation in Logistic Regression: *IWLS*, *SGD*, *Adam*. Then, compare the results achieved by these algorithms on 9 selected datasets (including 3 small and 6 large), make a convergence analysis, and check the impact of the stopping rule. Lastly, to compare implemented methods with other popular classifiers and check the impact of adding interactions to datasets.

In the first part of the report, we will describe the methodology used to implement the algorithms and then move on to a detailed description of the experimental results.

2 Implementation

The main part of the project concerns the implementation of 3 algorithms for parameter estimation in Logistic Regression.
The aforementioned optimizers are implemented in *Python*. We were using publicly available libraries such as *numpy*, *pandas*, *matplotlib*.

Implementation was combined into class `LogisticRegression` with a parameter that indicates the chosen optimizer. The class contains several methods, among which the ones used for training the model and getting the predicted values/probabilities are:

- `fit(X, y)`,
- `predict(X)`
- `predict_proba(X)`

Creating one class with the choice of the optimizer as a parameter allows for the unification of the entire implementation and ensures that the same methods, conditions, or parameters are used to compare methods.

2.1 IWLS

Iterative Re(weighted) Least Squares - `IWLS` - is an algorithm for estimation coefficients in logistic regression. In this method, weights are updated by computing weighted least squares estimates at each iteration.

The algorithm starts with the initialization step. Weights are initialized randomly from a normal distribution.
Next, coefficients are iteratively updated using a Newton-Raphson update procedure.

$$\beta^{new} = \beta^{old} H^1(\beta^{old}) \frac{\partial l}{\partial \beta}_{\beta=\beta^{old}}$$

where H(β) - matrix of second-order derivatives (Hessian).

2.2 SGD

Stochastic Gradient Descent - `SGD` - is our next algorithm for estimation coefficients in logistic regression.
Unlike traditional gradient descent, which calculates the gradient of the loss function with respect to all training examples, *SGD* computes the gradient and updates the model parameters using only one or a subset (mini-batch) of training examples at each iteration. In our implementation, we choose to add batch size as a parameter of the function, but the default value is set to 1. The default value was used in all experiments.

Similarly to the previous method, the algorithm starts with an initial guess for the model parameters (weights and bias). What is important is that the training dataset is shuffled to ensure that the examples are presented in a random order.

Next, in each iteration, the gradient of the loss function is calculated and used to update the weights and bias.

$$\beta^{t+1} = \beta^t - \eta \nabla L(x_i, y_i; \beta^t)$$

where η is the learning rate, a hyperparameter that controls the size of the step taken in the direction opposite to the gradient, and $\nabla L(x_i, y_i; \beta^t)$ represents the gradient of the loss function.

2.3 Adam

Adaptive Moment Estimation - `Adam` - is an extension to stochastic gradient descent. This method considers the moving average of the first and second-order moments of the gradient, which allows the adaptation of the learning rates for each parameter in a more robust way. Adam combines the ideas applied in *RMSProp* and *SGD* with *Momentum*.

The initialization step is identical to *SGD* and *IWLS*. Next, the updates of weights consist of two parts: update moving averages

$$m^{t+1} = \beta_1 \cdot m^t + (1 - \beta_1) \cdot \nabla L(x_i, y_i; \beta^t)$$
$$v^{t+1} = \beta_2 \cdot v^t + (1 - \beta_2) \cdot \nabla L(x_i, y_i; \beta^t)^2$$

bias correction

$$\hat{m}^t = \frac{m^t}{1 - \beta_1^t}$$
$$\hat{v}^t = \frac{v^t}{1 - \beta_2^t}$$

updates of weights

$$\beta^{t+1} = \beta^t - \frac{\eta}{\sqrt{\hat{v}^t} + \epsilon} \cdot \hat{m}^t$$

where η is the learning rate, a hyperparameter that controls the size of the step taken in the direction opposite to the gradient, and $\nabla L(x_i, y_i; \beta^t)$ represents the gradient of the loss function.

3 Methodology

3.1 Datasets

All chosen datasets are from UC Irvine Machine Learning Repository. Overall, 9 datasets were chosen, of which 3 contain at most 10 features, and 6 contain more than that. Details of the datasets are presented in Tables 1 and 2

Table 1: Details of small datasets

Name	Features	Rows	Description
banknote_authentication	4	1372	Data extracted from images taken for the evaluation of an authentication procedure for banknotes.
fertility	10	100	Semen samples collected from 100 volunteers and analyzed according to the WHO 2010 criteria.
magic_gamma_telescope	10	19020	Data are MC generated to simulate registration of high energy gamma particles in an atmospheric Cherenkov telescope.

Table 2: Details of large datasets

Name	Features	Rows	Description
spambase	57	4601	Classifying Email as Spam or Non-Spam.
taiwanese_bankruptcy_ prediction	95	6819	The data were collected from the Taiwan Economic Journal for the years 1999 to 2009.
connectionist_bench_ sonar_mines_vs_rocks	60	208	Contains 111 patterns obtained by bouncing sonar signals off a metal cylinder at various angles and under various conditions and 97 patterns obtained from rocks under similar conditions.
ionosphere	34	351	"Good" radar returns are those showing evidence of some type of structure in the ionosphere. "Bad" returns are those that do not.
algerian_forest_fires	14	244	The dataset includes 244 instances that regroup data from two regions of Algeria.
waveform_database_ generator_version_1	21	5000	CART book's waveform domains. Waveforms were divided into 3 classes.

All chosen datasets are quite clean. During the preprocessing step, datasets were checked for the presence of null values, and if such a value was found, a given row was deleted. We decided to delete the rows with missing values instead of imputing the values because of very few occurrences (in most datasets, there were no missing values at all).

Moreover, categorical features were mapped to numerical values, and collinear variables were removed. (Variable was removed when the Pearson correlation coefficient was greater than 0.9)
Additionally, algerian_forest_fires was cleaned because labels were at the beginning applied with or without spaces, and from waveform_database_generator_version_1, we chose to leave only two target classes from three available in the source data (in that case, we leave dataset well balanced).

3.2 Stopping rule

The training process is stopped when it reaches a given number of iterations. The default value for this parameter is set to 500, however, any number can be passed when initializing the model.

This method may not always be the best solution as, in many cases, the model can converge before the given iteration number is up, and if it happens, we could stop it earlier.

After each iteration, the *log-likelihood* loss was computed. Then, if for some number of iterations (by default set to 5), the log-likelihood is not increasing (with some tolerance, by default, set to 0.0001), then the training process is stopped.

3.3 Balanced Accuracy

Balanced Accuracy was used to measure the performance of the model. To compute it, we used a function from *sklearn*. However, the formula to compute this score is:

$$Balanced\_accuracy = \frac{(Sensitivity + Specificity)}{2}$$

where: Sensitivity - The "true positive rate" – is the percentage of positive cases the model can detect, and Specificity - The "true negative rate" – is the percentage of negative cases the model can detect.

Each dataset was divided into five different train and test splits using *kFold* method. On each of the train sets, models with different optimizers were trained, and their performance was evaluated on the corresponding test sets. From the obtained results, boxplots were created, and they are visible below in Figure 1.

We can observe that on most datasets, the models with *Adam* optimizer perform the best. In most cases, the *SGD* optimizer seems to perform quite well, while the models with *IWLS* usually cover the widest range, sometimes being the best and sometimes the worst.

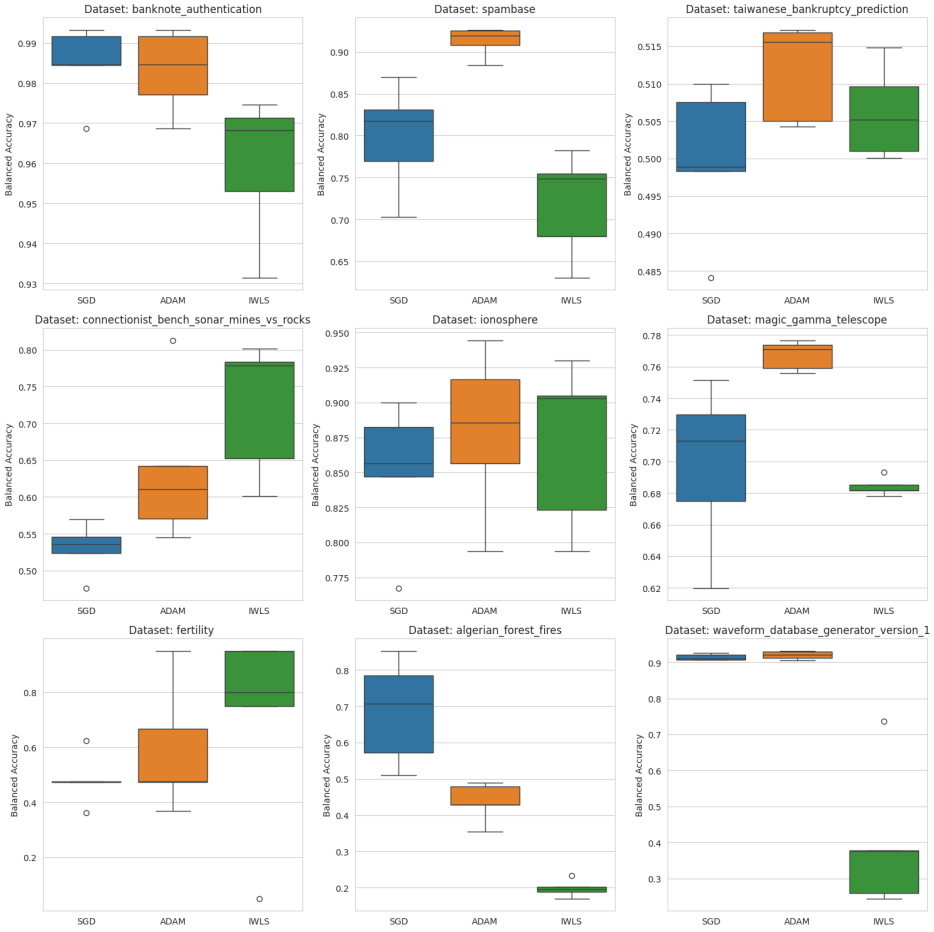

Figure 1: Balanced accuracy for implemented methods

4 Convergence analysis

For the purpose of convergence analysis, methods are run once, and results in the form of graphs were recorded and shown in Figure 2.

The graphs show the relationship between the log-likelihood function and the number of iterations. The plots on the left are from the models that were trained without any stopping condition other than the number of iterations, while the ones on the right contained an additional stopping condition described in Section 3.2.
If the line showing the results for a certain model breaks off, this indicates that a prior stop condition has been applied for this case.

SGD and *Adam* are similar to each other when we omit the fluctuations of *SGD*. *IWLS* is significantly different from the other methods. This is perfectly illustrated by the dataset fertility when *SGD* and *Adam* slowly improve the value of the log-likelihood function while *IWLS* rapidly rises after a few first iterations and stays at this level, which ends the run at only 7 iterations.

Also, for spambase dataset, we can see a strange behaviour of the *SGD* optimizer, which has a globally increasing character but with many fluctuations.

The example of banknote_authentication shows that the choice of stopping rule is important to achieve the best results and time performance. All 3 methods stop at the beginning of executions.

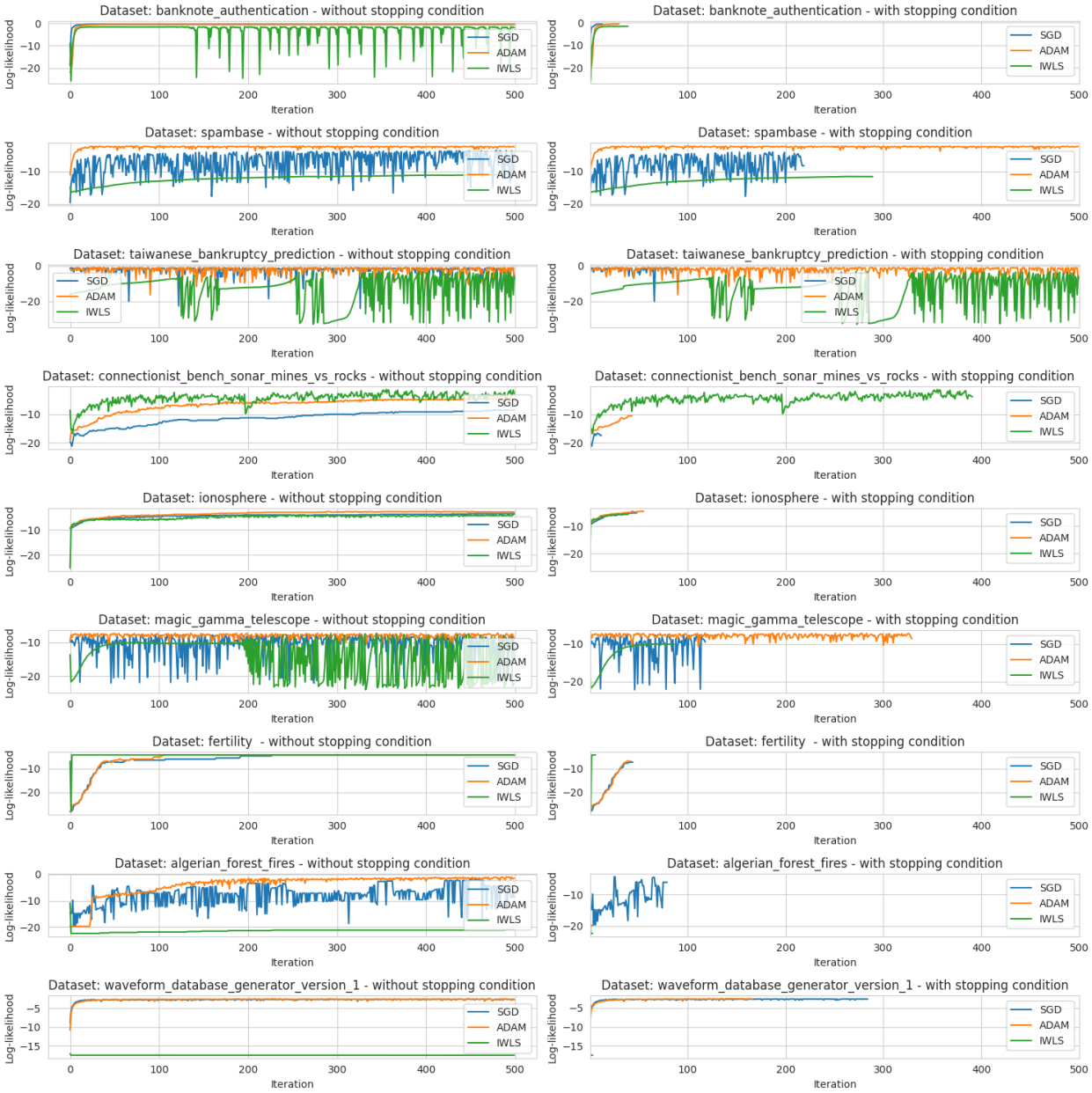

Figure 2: Convergence analysis

5 Comparison of classification performance

As an experiment, we compared the results of the implemented methods, with other popular classification methods from *sklearn*. We run all classifiers on the same datasets and the same divisions of training and test sets. The methods that we considered are:

1. LDA (Linear Discriminant Analysis),
2. QDA (Quadratic Discriminant Analysis),
3. Decision Tree,
4. Random Forest.

For each loaded dataset, 5 train_test splits were performed, and balanced accuracy was computed as a comparison metric.

As shown in Figure 3 results of methods vary depending on the dataset. However, we don't encounter a visible, consistent difference between our methods and those used for comparison. It can confirm that our implementation does not deviate significantly from the standard in quality and can be used interchangeably with popular classifiers.

Random forest classifier performs best in most cases. What is quite interesting for waveform_database_ generator_version_1 dataset *IWLS* is not stable, and its performance is significantly lower than the rest.

More even results can be seen for the dataset ionosphere. Here, only Random Forest is better, but all the rest of the methods have balanced accuracy between 0.85 - 0.90.

However, the most even dataset is banknote_authenticate. All methods have scores between 1.00 - 0.95 balanced accuracy. These results are also the best for all presented datasets in this part of the experiments.

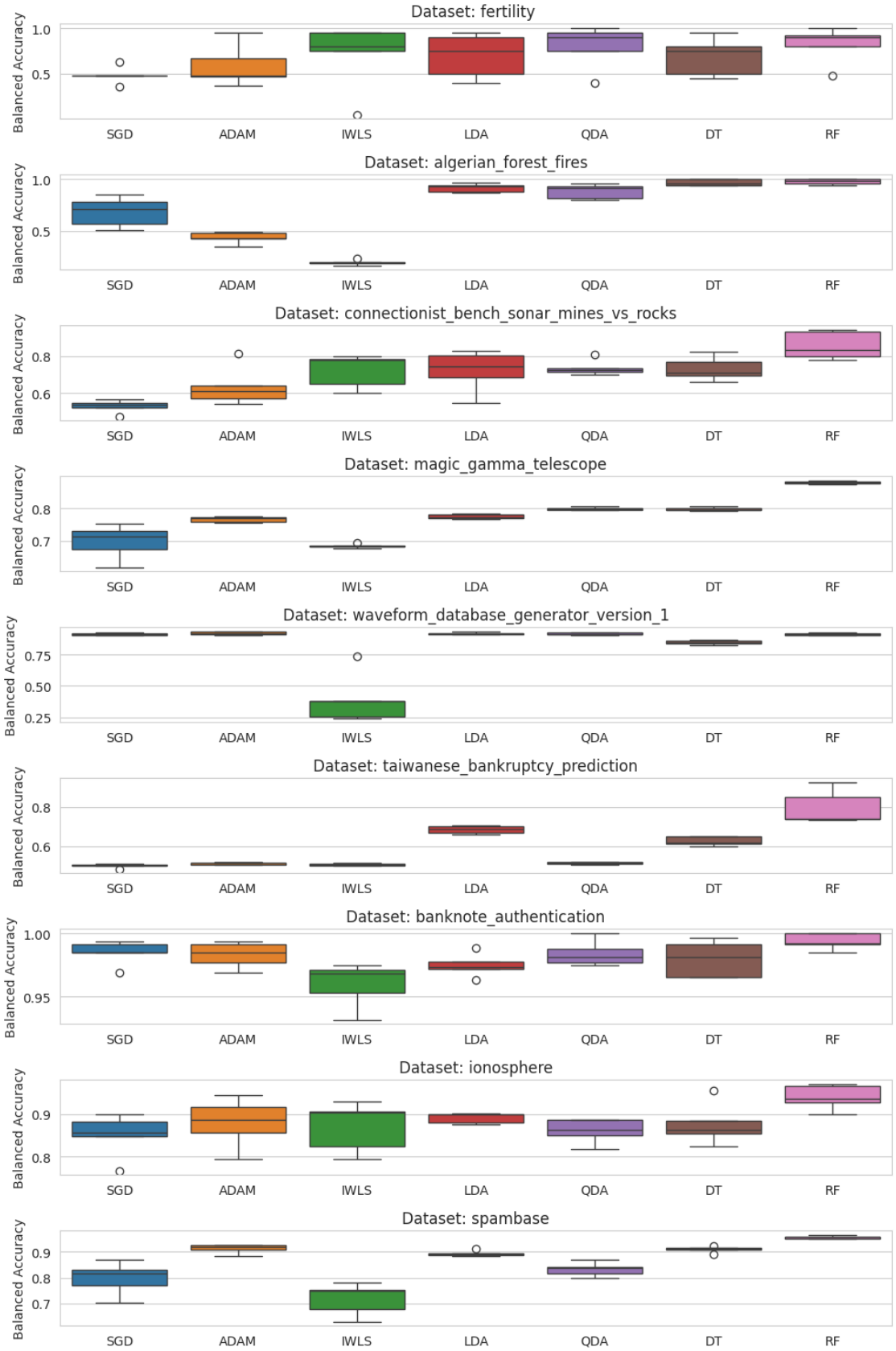

Figure 3: Comparison of implemented methods (IWLS, SGD, Adam) with 4 basic classifiers (LDA, QDA, Decision tree, Random Forest)

6 Comparison of classification performance of models with and without interactions

Interactions between variables are defined as the products of two variables. While using this functionality in our experiments, in addition to the original variables, new features were created and added to the model.

Comparison of performance of models with and without interactions are shown for small datasets: fertility, magic_gamma_fertility, banknote_authentication. Results are illustrated in Figure 4. The interactions do not improve the results for all datasets. Interestingly, the results have improved significantly for *banknote_authentication* - the dataset with the lowest number of features.

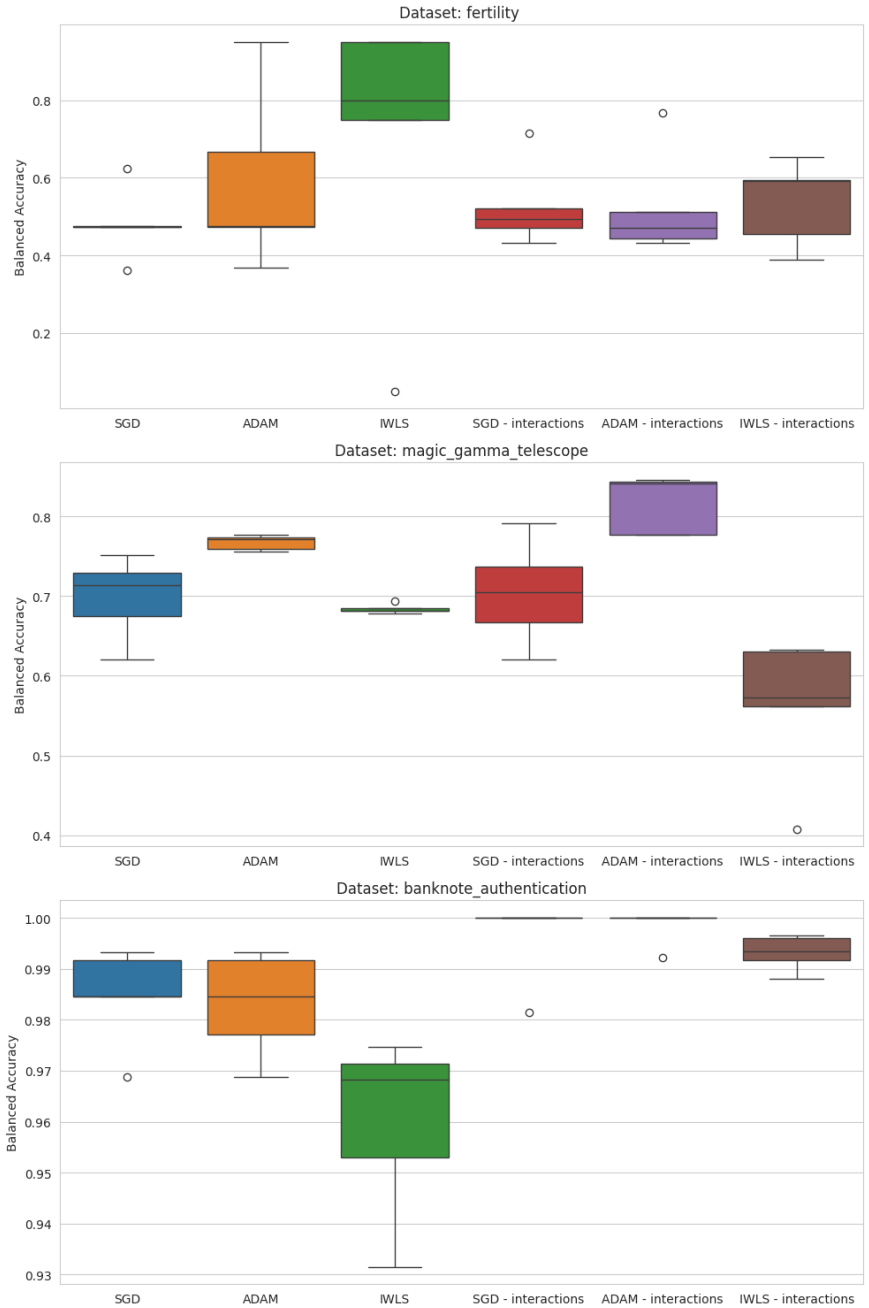

Figure 4: Comparison of classification performance of models with and without interactions

7 Additional experiment

In this section, we describe the effect of scaling the features of the dataset. We used the *StandardScaler* implementation from *scikit-learn* For every column, the following normalization is applied:

$$z = \frac{x - \mu}{\sigma}$$

Figure 5 presents the comparison of the accuracy of models trained with and without scaling. For *algerian_forest_fires, magic_gamma_telescope, waveform_generator_version_1, taiwanese_bankrupcy_prediction, spambase* the accuracy improved after scaling of the variables.

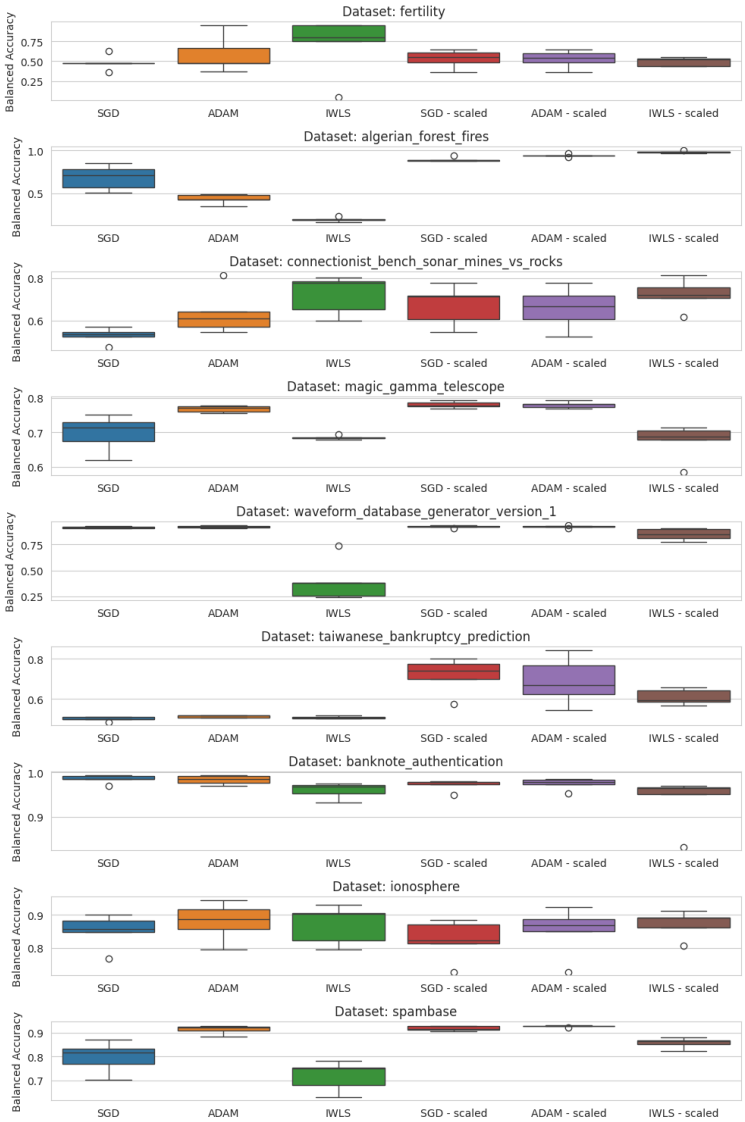

Figure 5: Accuracy of normalized and not normalized datasets

8 Conclusion

In conclusion, the Logistic Regression models with the implemented optimizers perform quite well in most cases. In comparison to different popular classification methods, they do not seem to stand out.

Moreover, the addition of interactions to the dataset proved to be a good choice, especially for datasets that contain very few features.

The performance of the models does not seem to have any clear pattern when comparing them on different datasets, however, in most cases, *SGD* and *Adam* optimizer yield better results than *IWLS*.

The chosen stopping condition proved to be a reasonable choice as the training process usually stops shortly after the function converges. It is not ideal, and possibly some other tools could be introduced in order to better detect the convergence when the *log-likelihood* fluctuates. Overall, no one optimizer or method seems to be an ideal choice for all datasets, and in order to obtain the best results, the decision on which method and convergence criteria to choose should be made separately for each case.