Logistic Regression Optimization Algorithms Implementation and Performance Analysis

Project Final Report

Authors: Bartosz Grabek, Izabela Telejko, Grzegorz Zbrzeżny

28 Mar 2024

Abstract

This project[1] aims to investigate, implement, and compare the performance of three optimization algorithms—Iterative Weighted Least Squares (IWLS), Stochastic Gradient Descent (SGD), and Adaptive Moment Estimation (ADAM) for logistic regression. In the study the effectiveness of each of these methods is assessed and evaluated across a range of datasets with various number of features, with and without interactions. The comparative analysis includes metrics such as convergence speed and classification balanced accuracy. Furthermore, implemented classifiers are compared with Scikit-learn models commonly used for binary classification task such as Random Forest or LDA. The results of the project help to gain insight into the popular optimisation methods and shows how they perform on various datasets.

Contents

[1]Source code: `https://github.com/Bartosz7/AML-2024L-Optimization-Algorithms`

1 Methodology

Datasets

In order to check the performance between the optimization methods we selected 3 small datasets with less than 10 features and 6 large datasets with more than 10 features. Additional requirements for the data included that no more than 10% of the data was missing. Some of the datasets have higher class imbalance than other (see Table 1).

Data Preprocessing

As part of data preprocessing before the train-test-split we performed:

- data loading (from multiple formats `.csv`, `.arff`, `.txt`)
- one-hot-encoding the categorical ordinal features and label-encoding the categorical nominal features,
- identifying the target columns and encoding them to integers 0, 1 for binary classification (for multi-class datasets classes were combined to form 2 classes in the end, or only the 2 most numerous classes were preserved),
- optional addition of the interactions terms between the features of the input dataset (see Section 4)

After the train-test split, we performed:

- removal of multicollinear columns on the whole dataset using Variance Inflation Factor (`vif`) computed based on the training data,
- standard scaling fitted on the training data and applied to both training and test sets,
- for *Water Quality* dataset imputation of the missing data based on dominant columns in the training set,
- addition of a column of ones to the data for the bias term.

Stopping Rule

We propose the following stopping rule for the optimisation algorithms (see pseudocode of Algorithm 1):

Stop the algorithm if the difference between the current minus log-likelihood is higher than the best (smallest) recorded minus log-likelihood reduced by a fixed $\epsilon_2 = 0.1$ *for* $t = 5$ *(tolerance) consecutive iterations.*

This stopping rule aims to prevent the optimization process from continuing when it appears to be diverging or oscillating around a suboptimal solution. By comparing the minus log-likelihood at each iteration to the best minus log-likelihood observed up to that point, the algorithm can halt if it fails to make progress over several consecutive iterations. The parameters t and ϵ_2 were set based on empirical analysis of convergence trend for the algorithms.

Optimiser Evaluation with Balanced Accuracy

$$BalancedAccuracy = \frac{TPR}{TNR} \tag{1}$$

The custom implementations of Iterative Weighted Least Squares (IWLS), Stochastic Gradient Descent (SGD) and Adaptive Moment Estimation (ADAM) were evaluated for the logistic regression

Algorithm 1 Early stopping rule

```
if $CurrentLogLike < BestLogLike - \epsilon_2$ then
    $BestLogLike \leftarrow CurrentLogLike$
    $BestWeights \leftarrow CurrentWeights$
    $NoChangeCounter \leftarrow 0$
else
    $NoChangeCounter \leftarrow NoChangeCounter + 1$
end if
if $NoChangeCounter > t$ then
    early stopping
end if
```

classification on the above datasets using balanced accuracy metric (see Table 2), computed as a ratio of the True Positive Rate to the True Negative Rate. The differences in accuracies are subtle, yet the rates of convergence of different optimizers differ substantially (see Section 2). The observation of worse balanced accuracy results for some datasets can often be attributed to the prevailing class imbalance. This suggests, that class imbalance is one of the key factor affecting the training of classification models.

Dataset	**% of Class 0**	**% of Class 1**	**Imbalance**
Tour & Travels Customer Churn	76.50%	23.50%	High
Seeds	66.66%	33.33%	Medium
Pima Indians Diabetes	65.10%	34.90%	Medium
Hotel Booking Cancellation	67.20%	32.80%	Medium
Water Quality	88.56%	11.44%	High
Ionosphere	64.10%	35.90%	Medium
Jungle Chess	55.34%	44.66%	Low
League of Legends	50.01%	49.99%	None
Sonar (Rock vs. Mine)	53.37%	46.63%	Low

Table 1: Class Imbalance in Binary Classification Datasets

Dataset	**Size**	**obs.**	**feat.**	**Code Name**	**IWLS**	**SGD**	**ADAM**
Tour & Travels Customer Churn	S	954	7	`churn`	0.652	0.655	0.659
Seeds	S	210	8	`seeds`	0.970	0.974	0.977
Pima Indians Diabetes	S	768	9	`diabetes`	0.753	0.755	0.756
Hotel Booking Cancellation	L	36285	17	`booking`	0.750	0.752	0.750
Water Quality	L	7999	21	`water`	0.664	0.665	0.663
Ionosphere	L	351	35	`ionosphere`	0.830	0.835	0.985
Jungle Chess	L	4559	47	`jungle`	0.986	0.985	0.985
League of Legends	L	26834	58	`challenger`	1.000	1.000	1.000
Sonar (Rock vs. Mine)	L	208	61	`sonar`	0.773	0.780	0.780

Table 2: Dataset Information and Balanced Accuracy Results for three custom implementations of IWLS, SGD and ADAM (averaged over 5 train-test splits)

2 Convergence Analysis

In the previous section, we established that the optimizers show similar performance in terms of final recorded balanced accuracy. However, if we consider the rate of convergence, we can clearly observe that IWLS, SGD and ADAM differ significantly.

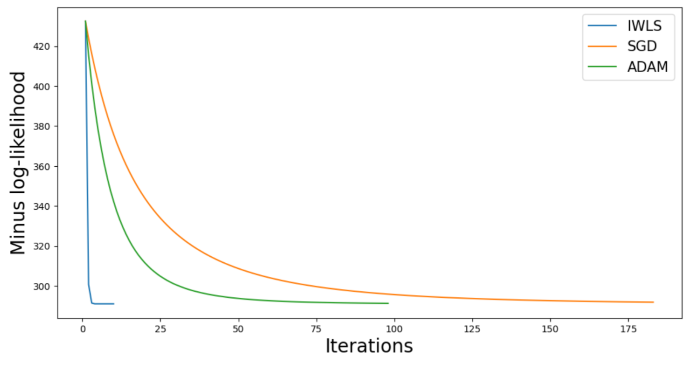
In Table 3, we compared the average number of iterations (epochs) required until the stopping rule was triggered across five train-test splits. IWLS exhibited the fastest convergence across most datasets. Additionally, between the two gradient descent algorithms considered, ADAM converged faster than SGD for every dataset.

Furthermore, we conducted an analysis of the changes in the minus log-likelihood throughout the iterations (Figure 1), which validated the previous results. We selected plots for one split for Diabetes, Churn, and Ionosphere datasets to be included in the report as they were the most interesting. From these plots we can observe that standardization of the data had a significant impact on the learning process, particularly for the SGD method, where the minus log-likelihood change curve became much smoother post-standardization. Moreover, learning with standardized data converged faster than without it. These are the reasons why we decided to run experiments on the standarized data. Analysis of the convergence plots further confirmed that our stopping rule makes sense, as the algorithms halted after the minus log-likelihood curve flattened.

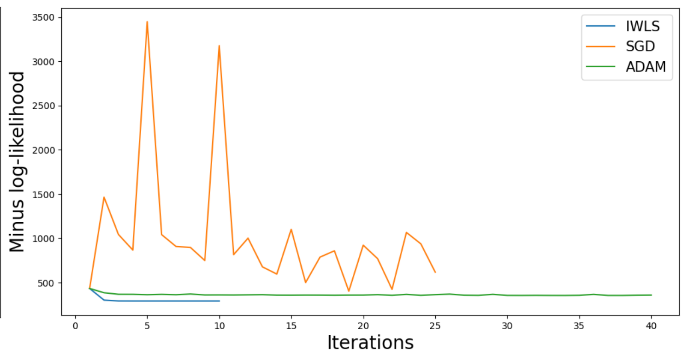
As we can see from the Table 3 for SGD for some datasets the early termination mechanism was not initiated and the algorithm stopped after reaching the maximum number of epochs. This suggests that the impact of early stopping may be highly influenced by the specific characteristics of the dataset and that SGD in particular is the method most prone to slow convergence, possibly due to the noise in gradient estimates based on single observations.

Dataset	**IWLS**	**ADAM**	**SGD**
`churn`	10	108	211
`Seeds`	12	268	500
`diabetes`	10	95	185
`booking`	13	38	44
`water`	12	45	130
`ionosphere`	14	479	500
`jungle`	16	384	500
`challenger`	18	14	500
`sonar`	13	424	475

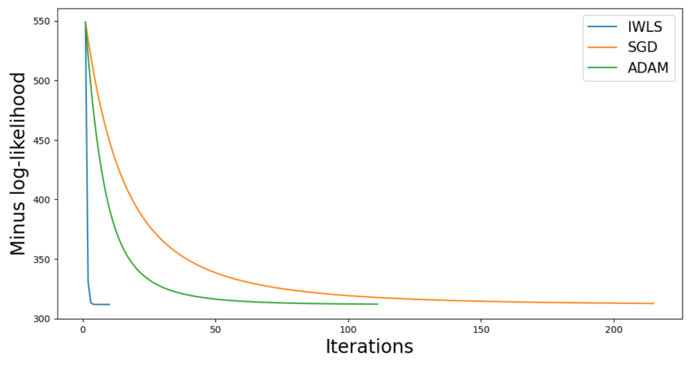
Table 3: Average number of epochs for 5 train test splits until convergence (stopping rule) or reaching the maximum number of epochs (500)



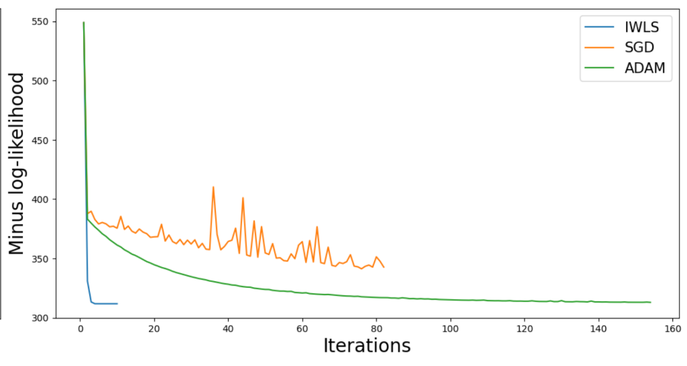
((a)) Diabetes with standard scaling



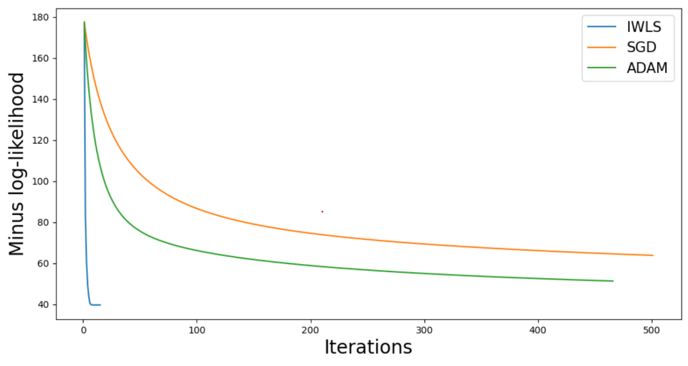
((b)) Diabetes without scaling



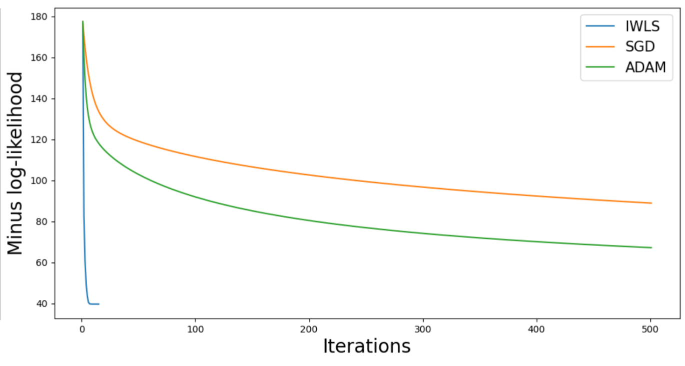
((c)) Churn with standard scaling



((d)) Churn without scaling



((e)) Ionosphere with standard scaling



((f)) Ionosphere without scaling

Figure 1: Convergence analysis for Ionosphere, Churn, and Diabetes dataset

3 Classification Performance Comparison

As part of our evaluation, we compared our custom logistic regression optimizers with other classification models made available throught the use of the `scikit-learn` library. We performed 5 train-test-splits and for every split we trained models on the training sample and evaluated their results on test sample. The Figure 2 shows the balanced accuracy throughout the splits for different models that are described in the Table 4. Most of the models use their default settings, however to prevent overfitting of tree classifiers, we introduced additional constraints such as maximum depth and minimum samples split.

Model	Model Name	Setting/Configuration
`iwls`	Iterative Weighted Least Squares (custom)	default
`sgd`	Stochastic Gradient Descent (custom)	`learning_rate=0.0002`
`adam`	Adaptive Moment Estimation (custom)	`learning_rate=0.0002`
`lr`	Logistic Regression	default
`qda`	Quadratic Discriminant Analysis	default
`lda`	Linear Discriminant Analysis	default
`dt`	Decision Tree Classifier	`max_depth=5`
`rf`	Random Forest Classifier	`max_depth=5`, `min_samples_split=3`

Table 4: Model configurations for comparison

Overall, our optimizers yielded comparable results to those of the Logistic Regression model across various datasets. In most cases the results obtained for each model were similar, except for Quadratic Discriminant Analysis (QDA) model, which was often predicting significantly worse (balanced accuracy $\approx$ 0.5). The only dataset where QDA performed slightly better than our optimizers was ionosphere dataset. However, when we calculated the accuracy score, QDA did not perform significantly different from the rest of models. For a few datasets the top-performing model was a single Decision Tree (booking, churn, water) or Random Forest (ionosphere, sonar). Remarkably, all models demonstrated exceptional learning capability on the challenger dataset (balanced accuracy $\approx$ 1). Moreover all models taken into account excluding the LDA and QDA, achieved near-perfect performance on the jungle dataset (balanced accuracy $\approx$ 0.99) and on the seeds dataset balanced accuracy was approximately 0.97 on every model, with the exception of QDA.

We do not see any relationship between the number of observations, nor the number of features and obtained results, nevertheless the conclusion drawn in Section 2 about the impact of unbalanced classes is visible also for the models from scikit-learn – churn and water are the most unbalanced datasets, while challenger's target is evenly distributed.

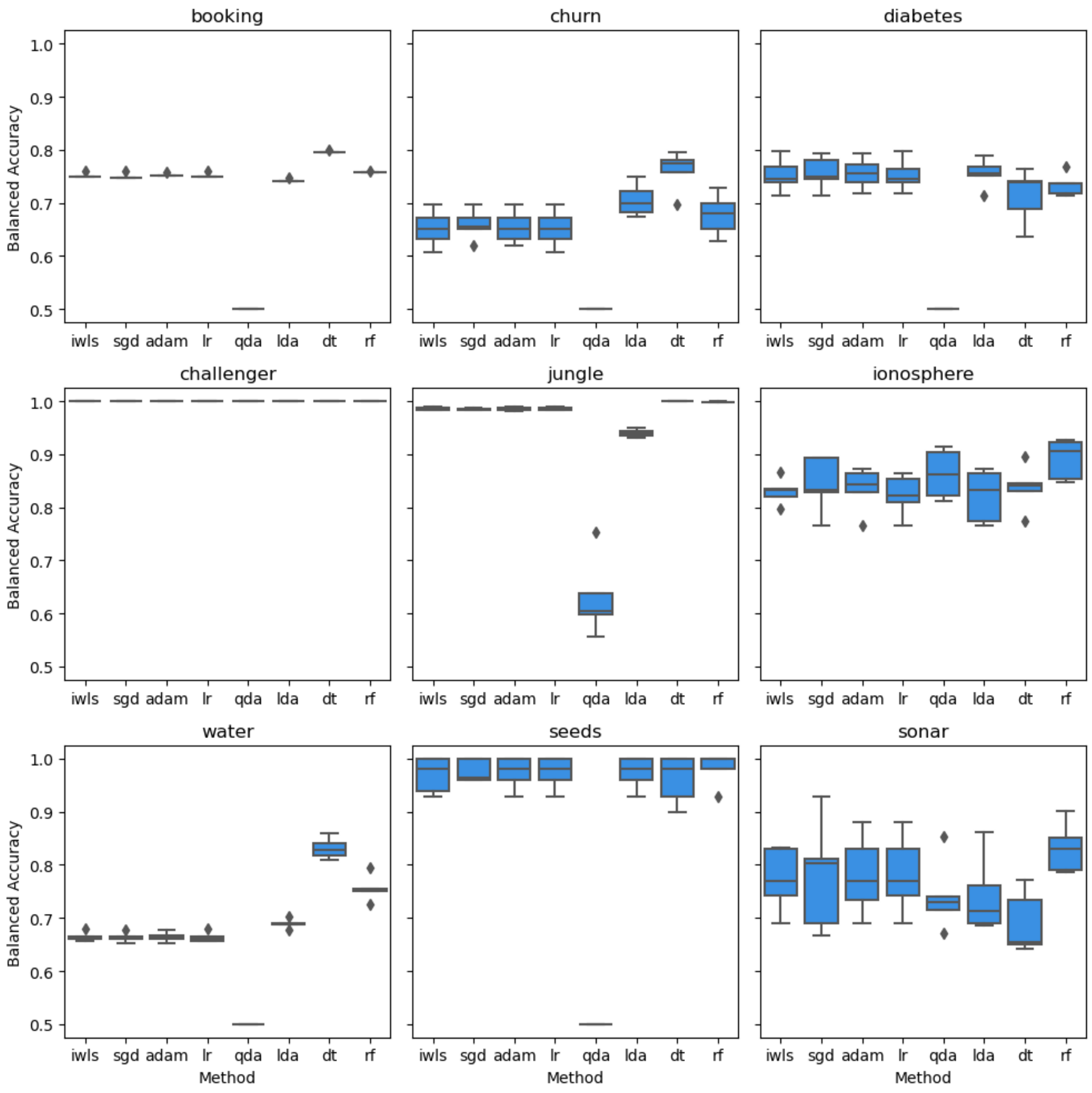


Figure 2: Averaged balanced accuracy for custom and scikit-learn models

4 Classification Performance on Data with Interactions

In another setting, we consider all the datasets but with pairwise feature interactions. The datasets extended with interactions are used for training and evaluation of the custom models IWLS, SGD and ADAM. The expectation of using such approach is that the combinations of the features of the input dataset may show some underlying synergy effects that are strong drivers for class predictions. The results presented in Table 5 and Figure 3 support this statement only to some extent. For booking, churn, and jungle datasets we can see a slight improvement of balanced accuracy (approx. +0.03 on average) compared to datasets without interactions. However, for diabetes and ionosphere datasets, the balanced accuracy metric is worse compared to datasets without interactions. Most notably the water dataset benefits from the inclusion of interactions (approx. +0.15 on average).

Overall, including the interaction terms may both increase and decrease the logistic regression classifier performance. The direction of change is mostly driven by the characteristics of the dataset in question.

Dataset	**IWLS**	**IWLS+INT**	**ADAM**	**ADAM+INT**	**SGD**	**SGD+INT**
`churn`	0.652	0.693	0.654	0.695	0.659	0.694
`seeds`	0.970	0.968	0.974	0.981	0.977	0.976
`diabetes`	0.753	0.711	0.756	0.721	0.755	0.724
`booking`	0.751	0.775	0.752	0.773	0.751	0.774
`water`	0.664	0.808	0.665	0.802	0.663	0.796
`ionosphere`	0.830	0.644	0.835	0.747	0.843	0.726
`jungle`	0.986	1.000	0.985	1.000	0.985	1.000
`challenger`	1.000	1.000	1.000	1.000	1.000	1.000
`sonar`	0.773	0.798	0.780	0.823	0.780	0.799

Table 5: Balanced Accuracy Results for implemented optimizers with and without interactions.

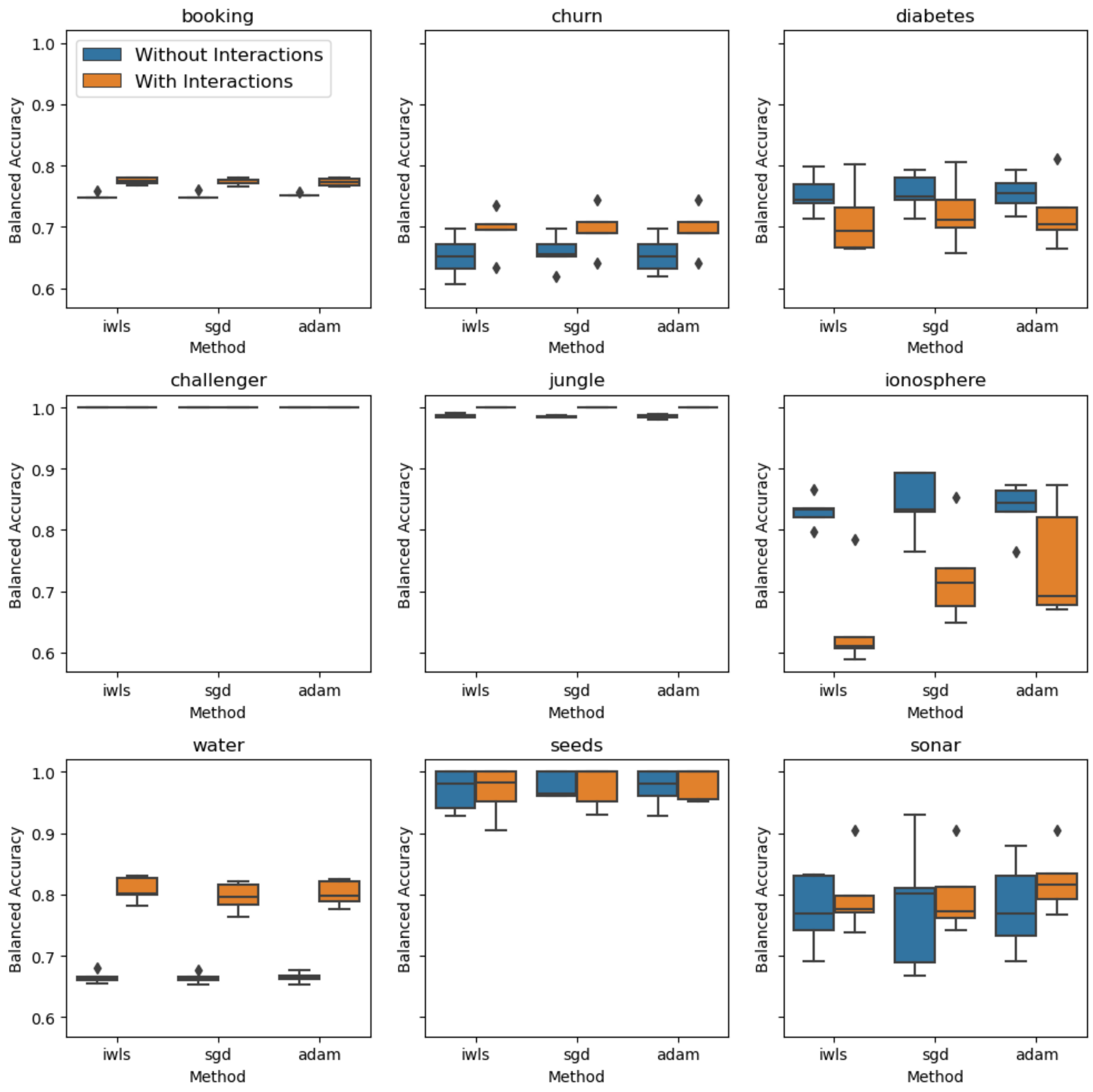


Figure 3: Averaged balanced accuracy for custom and scikit-learn models