# Advanced Machine Learning — Project 1 Report

Maja Andrzejczuk, Jakub Kasprzak, Maciej Orsłowski

## Contents

# 1 Methodology

The methodology section presents the approach undertaken to implement and evaluate optimization algorithms for logistic regression. The primary objective of this project is to compare the performance of different optimization techniques in logistic regression modeling.

## 1.1 Data

In our solution, we used 9 different datasets for the classification task, all sourced from the open platform OpenML, designed for dataset sharing. The platform enables straightforward access to datasets through the built-in `fetch_openml` function in the `sklearn.datasets` package, facilitating easy replication of our experiments. Within our solution, we employed 6 large and 3 small datasets. We define large datasets as those containing more than 10 variables after preprocessing.

To ensure consistency across all datasets, we first removed categorical variables and transformed the target variable into binary values, 0 and 1. This standardized dataset undergoes preprocessing before training. During preprocessing, columns dominated by only one unique value are removed to avoid redundancy. A value is considered to be dominant if it occurs in more than 80% of the rows. Missing values in each column are replaced with the mean to maintain data integrity. Next, highly correlated columns, identified using Pearson correlation and surpassing a predetermined threshold of 0.9, are eliminated to reduce multicollinearity. Finally, min-max scaling is applied to normalize relevant columns, ensuring that all features are on a similar scale for effective modelling.

## 1.2 Experiments setup

The implementation encompasses optimization algorithms for parameter estimation in logistic regression, including Iterative Reweighted Least Squares (IRLS), Stochastic Gradient Descent (SGD), and Adaptive Moment Estimation (ADAM). The metric used to evaluate models performance is balanced accuracy, which can be calculated as follows:

$$\text{Balanced Accuracy} = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right)$$

Models are trained on a training set, and performance is evaluated on a separate set. To ensure robust evaluation, results are averaged over 5 train-test splits. In case an algorithm does not converge within 500 iterations, the training process is terminated, and the solutions from the last iteration are used. In our solution, we save the parameters and loss value for the validation set from each epoch. After the training is completed, the fit method sets the model parameters to those that achieved the best results, minimizing the validation loss value.

The algorithm in our solution implements a stopping rule based on tol and patience. Tol represents a tolerance threshold for the change in loss between successive epochs, determining whether results have improved since the last epoch. Patience specifies the number of epochs during which the algorithm is allowed to exhibit no improvement. If there is no improvement in validation loss over a certain number of epochs, early stopping is triggered. During the experiments, we used default settings for the stopping rule, with a tolerance of value $10^{-5}$ and patience of value 10.

During the training process in all experiments, we maintained default learning settings for each algorithm. For SGD and IRLS algorithms, we used the learning rate value of 0.01, while for the ADAM optimizer, we set it to 0.001 in accordance with recommendations from available implementations [1] [2].

# 2 Convergence analysis

Convergence analysis is performed to check how the value of the loss function depends on the number of iterations for the three algorithms under consideration. This analysis is conducted on the training data to assess the behavior and performance of the algorithms as the number of iterations increases. The loss function we use is the log-likelihood function, which we calculate as follows:

$$\text{loss} = -\frac{1}{N}\sum_{i=1}^{N}\Big[y_i \log(p_i + \epsilon) + (1 - y_i)\log(1 - p_i + \epsilon)\Big]$$

where $N$ is the number of observations, $y_i$ represents $i$-th observation's class (0 or 1), $p_i$ denotes the predicted probability of belonging to class 1 for $i$-th observation, and $\epsilon$ is a very small value.

## 2.1 Setup details

We conducted this experiment using both small and large datasets. In all cases, we select the option without variable interactions. All models are trained on the same train/test splits. During each iteration, the algorithm updates its parameters based on the training data, and calculates the value of the loss function. The process continues until a stopping criterion is met, such as reaching a maximum number of epochs or achieving a certain tolerance threshold for the change in validation loss.

To ensure smoother and clearer convergence plots, the following steps were taken in the data preparation process for analysis. We limited the length of each run to match the longest one, and filled missing values (NaN) in each run with the final loss value. This process resulted in more uniform data used for generating convergence plots of the algorithms.

## 2.2 Results Analysis

It can be observed that for both small and large datasets, the algorithm often triggers early stopping (fails to achieve significant improvement over many epochs) when using IRLS. Additionally, it is noticeable that in the case of the Arrhythmia and Philippines datasets, where training proceeds unstably for ADAM and SGD optimizers, the IRLS optimizer is the only one that maintains stability. What we can additionally observe with the IRLS optimizer is that in the majority of large datasets, as well as in the small dataset 'Blood Transfusion', the algorithm performed poorly with fewer epochs, learning slowly. However, with a larger number of epochs close to 500, the algorithm achieved comparable results, sometimes even better than other optimizers.

We observe that in the case of large datasets, algorithms using ADAM and SGD optimizers learned quite rapidly, especially within the first 100-200 epochs. Subsequently, they continued to learn at a slower pace but typically did not trigger early stopping, suggesting that with a larger number of epochs, they would potentially achieve even better results.

## Small Datasets

The blue color represents IRLS, orange - SGD, and green - ADAM.

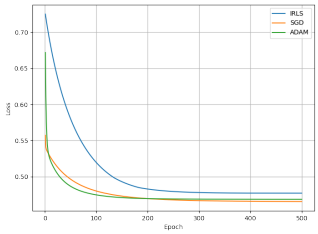


Figure 1: Convergence - Blood Dataset

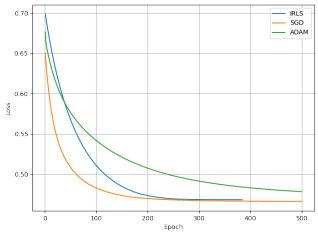


Figure 2: Convergence - Diabetes Dataset

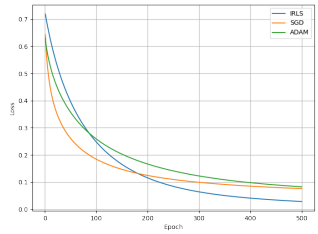


Figure 3: Convergence - Banknotes Dataset

## Large Datasets

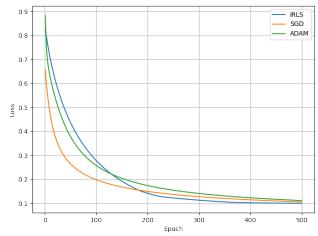


Figure 4: Convergence - Auto Dataset

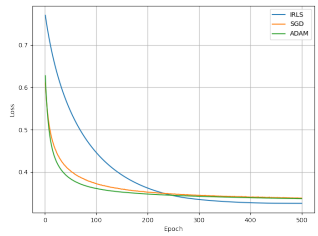


Figure 5: Convergence - Biodeg Dataset

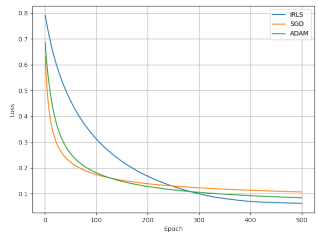


Figure 6: Convergence - Breast Dataset

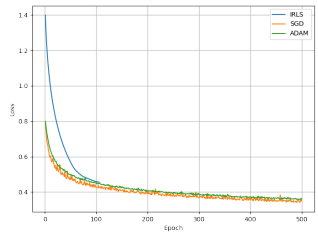


Figure 7: Convergence - Arrhythmia Dataset

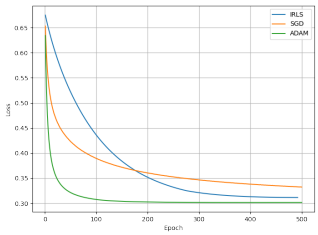


Figure 8: Convergence - Spambase Dataset

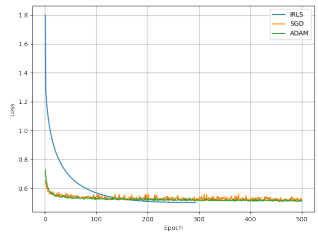


Figure 9: Convergence - Large Philippine Dataset

# 3 Comparison of classification performance

In our experiments we compared the performance of four well-known classification algorithms: LDQ, QDA, decision tree and random forest using their implementations from scikit-learn[3] to our implementations of SGD, IRLS and ADAM optimisers.

## 3.1 Setup details

The setup details are similar to the previous experiment – all datasets are used, all models are trained on the same train/test splits, and interactions are not used. Balanced accuracy is calculated after the training is finished. For each model, balanced accuracy values from every train/test split are averaged to obtain the final score.

## 3.2 Results Analysis

The optimizers we have implemented achieve results comparable to the other algorithms. The specific results and the leaderboard of algorithms varies amongst different datasets. However, in most cases, scores achieved by our algorithms were not inherently lower than those of scikit-learn implementations.

In the case of smaller datasets, IRLS and SGD obtained better results than ADAM for most of them. As can be seen on graph 10, there has been an experiment, in which all of our implementations performed worse than the external implementations. However, this was the only one of nine datasets for which something like this happened. For the other small datasets, our implementations were much more competitive, as can be seen on graph 11 for dataset Banknotes.

In the case of large datasets, the results were even more diverse. For the Breast cancer dataset, as can be seen on plot 15, almost all of our implementations managed to outperform the external algorithms. For dataset Auto Price (graph 12) the results of all algorithms were highly similar, all of them achieving balanced accuracy close to 1. Results for datasets Spambase and Arrhythmia, shown on graphs 13 and 14 respectively, are more internally diverse in the sense of results obtained by different methods. However, in neither of these examples can one with certainty say whether our implementations or scikit-learn ones perform better. The only outlier among all the experiments seems to be the IRLS optimizer achieving visibly worse results on Arrhythmia dataset.

In conclusion, the results obtained by algorithms highly depend on the specific datasets. The experiments have not shown either our implementations or the external implementations to outperform the others vastly. A conclusion might be drawn, that the algorithms we have implemented are not worse than universally used implementations and can even perform better than them in some cases.

All remaining plots can be found in the attached Jupyter notebook.

## Small Datasets

The blue color represents our algorithms, while the orange one represents models from the scikit-learn library.

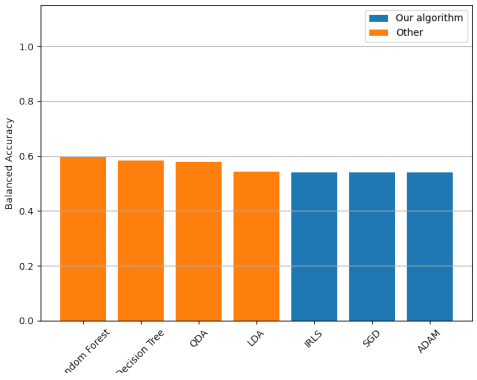


Figure 10: Comparison - Blood Dataset

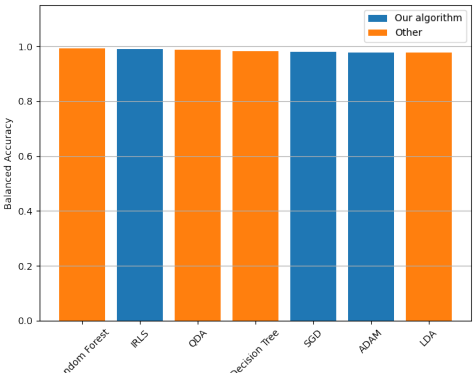


Figure 11: Comparison - Banknotes Dataset

## Large Datasets

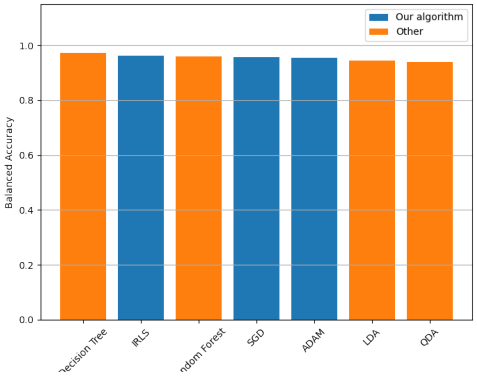


Figure 12: Comparison - Auto Dataset

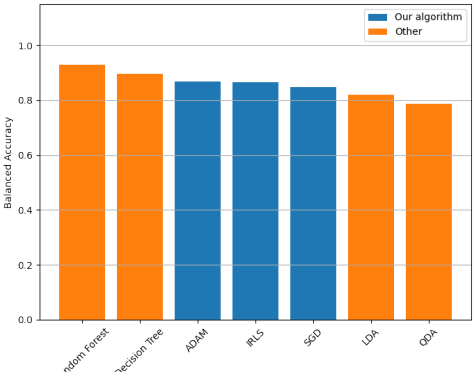


Figure 13: Comparison - Spambase Dataset

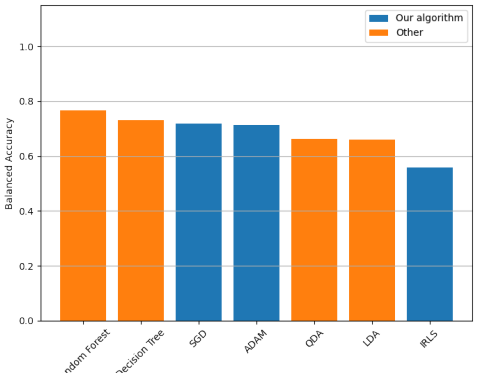


Figure 14: Comparison - Arrhythmia Dataset

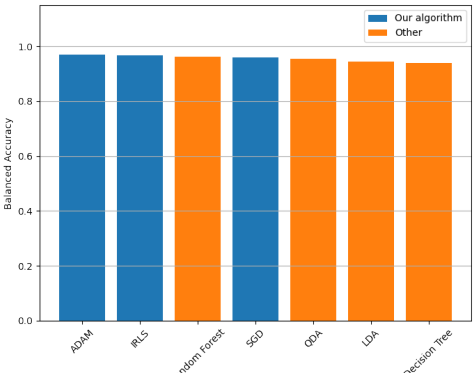


Figure 15: Comparison - Breast Dataset

# 4 Comparison of classification performance of models with and without interactions

In the following section, we delve into a comparative analysis of the classification performance of models with and without interactions. Interactions, in this context, refer to the consideration of the products of two variables in addition to the original input variables. By exploring this aspect, we aim to gain insights into how the inclusion of interactions impacts the predictive capabilities of various algorithms.

## 4.1 Setup details

In our solution, we have enabled the addition of interactions to each algorithm. Interactions involve considering the products of two variables in addition to the original input variables. A model without interactions only uses basic variables like X1, X2, X3, whereas a model with interactions adds new variables resulting from the multiplication of variable pairs, for example, X1 * X2, X1 * X3, X2 * X3. We conducted this experiment on three small datasets. For each dataset, we compared two versions of logistic regression: one without interactions and one with interactions. This resulted in a total of six variants of logistic regression (IRLS, SGD, ADAM, IRLS+INT, SGD+INT, ADAM+INT). Final models' performance is measured identically as in the previous experiment.

## 4.2 Results Analysis

We can observe that for each dataset, the use of interactions improves the performance of the algorithm for each optimizer, except in one case – the Diabetes dataset with the IRLS optimizer. This may indicate that employing interactions can aid in identifying relationships between variables that previous models failed to capture, thereby enhancing the overall performance of the model. However, one needs to keep in mind that this can be computationally expensive for large datasets, as the number of interactions grows rapidly as the number of features increases.

## Small Datasets

The orange color represents the model using interactions, while the blue one represents the model without using interactions between variables.

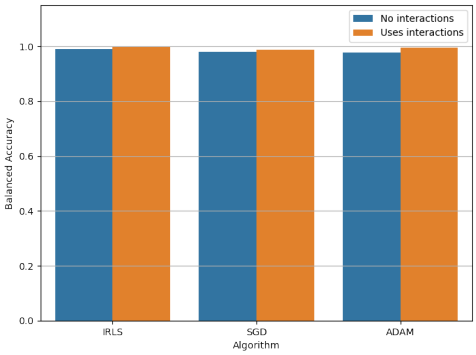


Figure 16: Interactions vs No interactions on Banknotes dataset

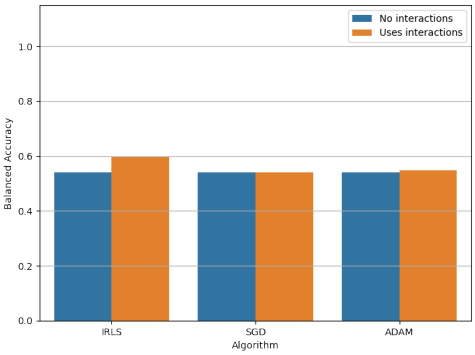


Figure 17: Interactions vs No interactions on Blood Transfusion dataset

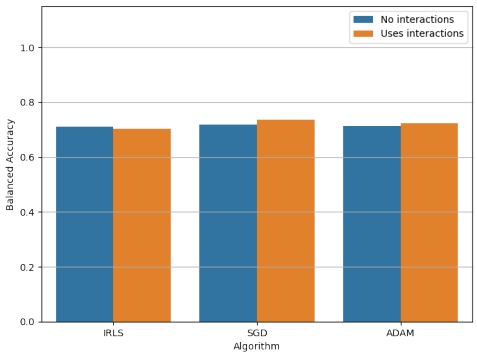


Figure 18: Interactions vs No interactions on Diabetes dataset

# References


[1] Christopher M. Bishop. *Pattern recognition and machine learning*. Springer Verlag, 8 2006.

[2] François Chollet et al. Keras. https://keras.io, 2015.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.