Data Science

Faculty of Mathematics and Information Science

Warsaw University of Technology

Advanced ML Project 1

Władysław Olejnik

Łukasz Zalewski

Warsaw, August 27, 2023

1 IRLS implementation

The Iteratively Reweighted Least Squares (IRLS) method is a widely used optimization algorithm in statistical inference and machine learning. It is particularly useful for solving problems with sparse and noisy data.

The IRLS method is an iterative algorithm that updates the solution at each iteration by solving a weighted least squares problem. According to [1] we can use the Newton-Raphson update, to estimate new, slightly better weights, in each iteration.

The implementation of this method works as follows:

1. Reshape the target variable y into a column vector using the *reshape* method.
2. If the *interaction_pairs* attribute is not empty, generate interaction features by computing the product of the columns of X corresponding to each pair of indices in *interaction_pairs*.
3. Append the intercept term to the input data X by adding a column of ones.
4. Initialize the coefficients to zero.
5. For a maximum of *max_iter* iterations:
   (a) compute the predicted probabilities using the sigmoid function,
   (b) compute the diagonal weight matrix,
   (c) compute the Hessian,
   (d) update the coefficients using Newton's method,
   (e) check for convergence.
6. Set the *coef_* attribute of the logistic regression model to the final values of the coefficients.
7. Return the coefficients.

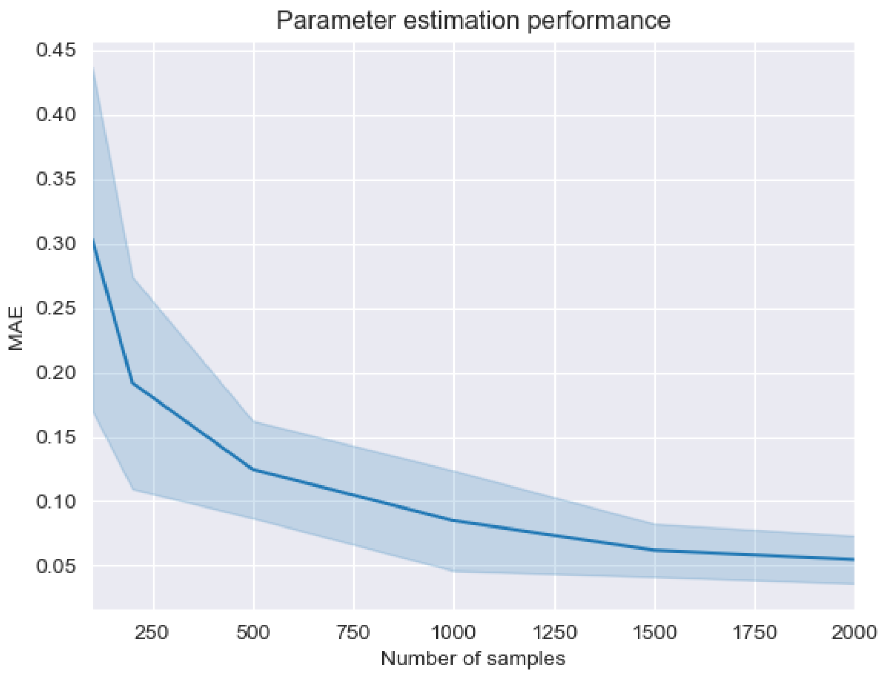
2 Experiments

2.1 Verifying correctness

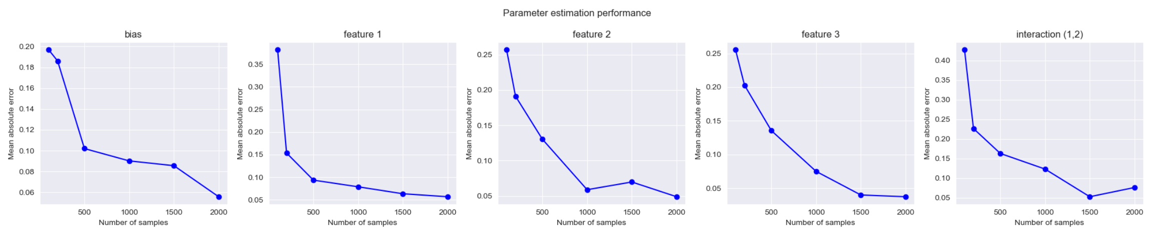
To test the implemented algorithm, we created an artificial dataset according to the following steps:

1. Generate an original feature matrix with random values using the normal distribution.
2. If specified, generate interaction features by multiplying columns of the original feature matrix together.
3. Concatenate the original and interaction features, adding a column of ones for the intercept term.
4. Generate a true weight vector with random values using the normal distribution.
5. Compute logits by multiplying the extended feature matrix by the true weight vector.
6. Apply the sigmoid function to the logits to generate probabilities.
7. Generate a target vector by sampling from a binomial distribution with the generated probabilities.

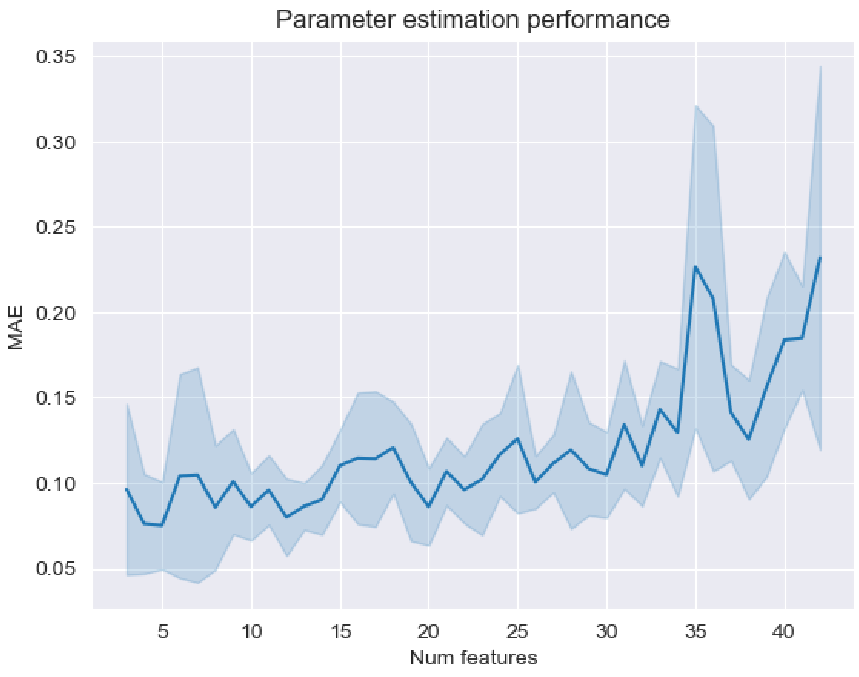
With the data ready, we moved on to testing the correctness of the implementation. The mean absolute error (MAE) was chosen as a metric to evaluate the performance of the algorithm. The MAE dropped from around 0.3 for 100 samples to 0.05 for 2000 samples. The more datapoints are used for training, the lower the error, indicating the correctness of implementation.



Next, we visualize how error decreases per each weight parameter individually during training. The MAE dropped similarly for every feature weight (including intercept and interactions).



Finally, we test how algorithm behaves with increasing numbers of features. We can vary number of features since we control how dataset is generated. The number of samples was fixed at 1000. The light area around the line on the chart denotes one standard deviation across 5 seeds. The results indicate that increasing number of features decreases the performance, which is expected since model is becoming more complex and starts having tr.



2.2 Testing on real datasets

We consider 3 binary classification datasets:

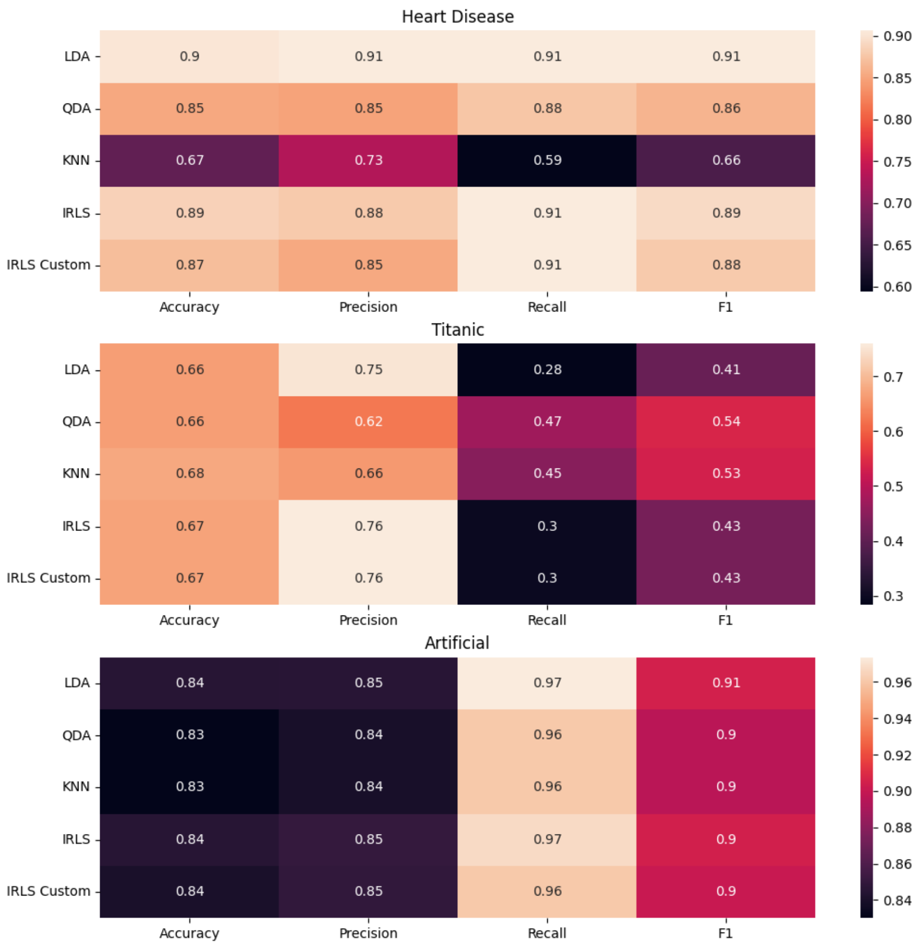
- Cleveland Heart Disease Dataset - a collection of medical records of 303 patients, which includes 16 attributes such as age, gender, cholesterol levels, and others. The dataset is commonly used in machine learning research for predicting the presence of heart disease.
- Titanic - a collection of passenger information from the ill-fated maiden voyage of the RMS Titanic in 1912. It includes data on 891 passengers, such as age, gender, ticket class, and survival status. This dataset is often used in machine learning research to predict the likelihood of survival based on passenger characteristics.
- Artificial dataset - the same default artificial dataset as in previous experiments.

We compare results across 5 methods:

- LDA (sklearn implementation)
- QDA (sklearn implementation)
- KNN (sklearn implementation)
- IRLS (sklearn implementation)
- IRLS (our custom implementation)

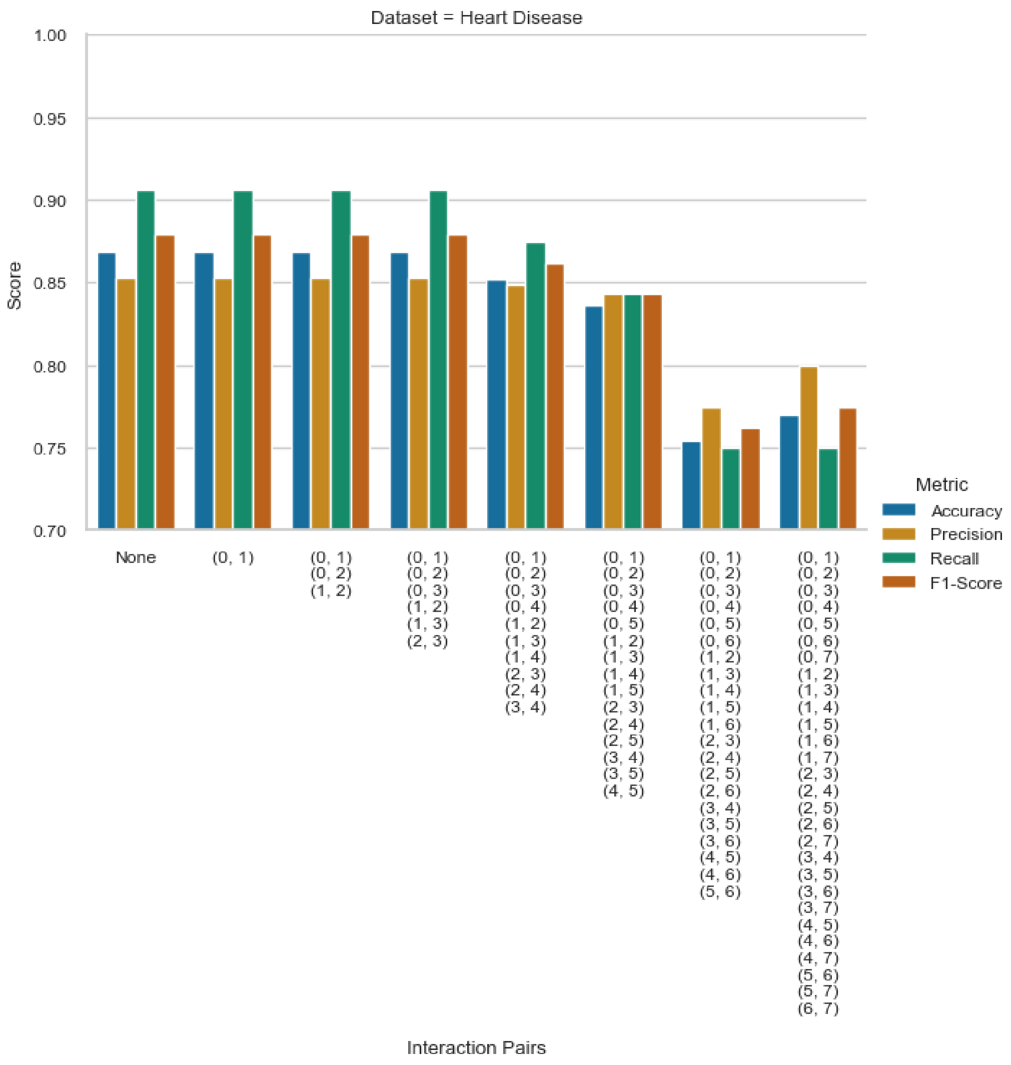
We don't introduce any feature interactions to reduce complexity of the experiment and because usually intreactions should be chosen specifically for each dataset.

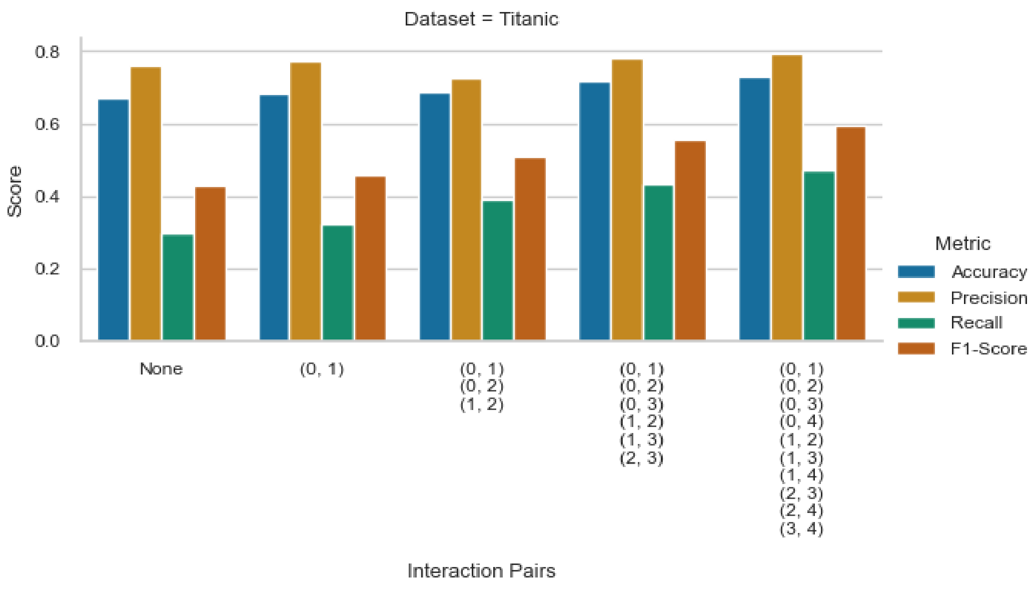
The results showed that our custom IRLS algorithm performed almost identically to the existing IRLS implementation.



2.3 Testing with different feature interactions

In this experiment, the logistic regression model was trained with feature interactions for the Heart Disease and Titanic datasets. The results showed that for the Titanic dataset, adding feature interactions slighly improved the performance of the logistic regression model. However, for the Heart Disease dataset, the performance of the model with feature interactions was worse than without feature interactions. The more feature interactions are included, the worse the performance, indicating that model is becoming too complex. The choice of whether to include feature interactions in the model depends on the specific dataset and should be determined through experimentation.


Dataset = Heart Disease
Score
1.00
0.95
0.90
0.85
0.80
0.75
0.70
None
(0, 1)
(0, 1) (0, 2) (1, 2)
(0, 1) (0, 2) (0, 3) (1, 2) (1, 3) (2, 3)
(0, 1) (0, 2) (0, 3) (0, 4) (1, 2) (1, 3) (1, 4) (2, 3) (2, 4) (3, 4)
(0, 1) (0, 2) (0, 3) (0, 4) (0, 5) (1, 2) (1, 3) (1, 4) (1, 5) (2, 3) (2, 4) (2, 5) (3, 4) (3, 5) (4, 5)
(0, 1) (0, 2) (0, 3) (0, 4) (0, 5) (0, 6) (1, 2) (1, 3) (1, 4) (1, 5) (1, 6) (2, 3) (2, 4) (2, 5) (2, 6) (3, 4) (3, 5) (3, 6) (4, 5) (4, 6) (5, 6)
(0, 1) (0, 2) (0, 3) (0, 4) (0, 5) (0, 6) (0, 7) (1, 2) (1, 3) (1, 4) (1, 5) (1, 6) (1, 7) (2, 3) (2, 4) (2, 5) (2, 6) (2, 7) (3, 4) (3, 5) (3, 6) (3, 7) (4, 5) (4, 6) (4, 7) (5, 6) (5, 7) (6, 7)
Interaction Pairs
Metric
Accuracy
Precision
Recall
F1-Score


Dataset = Titanic
Score
0.0
0.2
0.4
0.6
0.8
None
(0, 1)
(0, 1) (0, 2) (1, 2)
(0, 1) (0, 2) (0, 3) (1, 2) (1, 3) (2, 3)
(0, 1) (0, 2) (0, 3) (0, 4) (1, 2) (1, 3) (1, 4) (2, 3) (2, 4) (3, 4)
Interaction Pairs
Metric
Accuracy
Precision
Recall
F1-Score