Faculty of Mathematics and Information Sciences

WARSAW UNIVERSITY OF TECHNOLOGY

course

Advanced Machine Learning

Project 1 – Optimization algorithms for logistic regression

Weronika Plichta, Maja Wasielewska, Jakub Kubacki

335725 | 335210 | 313494

supervisor

Anna Kozak

WARSAW April 3, 2024

Contents

1. Methodology . . . 3

2. Convergence Analysis . . . 3

2.1. Datasets . . . 3

2.1.1. Diabetes detection dataset . . . 4

2.1.2. Credit fraud detection dataset . . . 5

2.1.3. Banknote authentication dataset . . . 6

2.1.4. Connectionist Bench (Sonar, Mines vs. Rocks) . . . 7

2.1.5. UCI ML Breast Cancer Wisconsin (Diagnostic) dataset . . . 9

2.1.6. Ionosphere . . . 9

2.1.7. Heart Disease . . . 10

2.1.8. Parkinsons . . . 10

2.1.9. HCV . . . 10

2.2. Optimization algorithms . . . 12

2.2.1. Iterative Reweighted Least Squares (IWLS) . . . 12

2.2.1.1. Overview . . . 12

2.2.1.2. Implementation . . . 12

2.2.2. Stochastic Gradient Descent (SGD) . . . 13

2.2.2.1. Overview . . . 13

2.2.2.2. Implementation . . . 13

2.2.3. Adaptive Moment estimation (ADAM) . . . 14

2.2.3.1. Overview . . . 14

2.2.3.2. Implementation . . . 14

2.3. Stopping rule . . . 14

2.4. Balanced Accuracy as performance measure . . . 15

2.4.1. Overview . . . 15

2.4.2. Results . . . 15

2.4.2.1. IWLS . . . 15

2.4.2.2. SGD . . . 16

2.4.2.3. ADAM . . . 16

2.5. Convergence analysis . . . 16

3. Comparison of classification performance . . . 17

4. Comparison of classification performance of models with and without interactions . . . 19

4.1. IWLS . . . 19

4.2. SGD . . . 19

4.3. ADAM . . . 19

5. Development environment . . . 20

1. Methodology

The purpose of this project is to compare the performance of various optimization algorithms used in logistic regression, focusing on three key tasks. The initial task is to collect and prepare nine datasets suitable for binary classification problems, divided into three smaller and six larger datasets. The second task focuses on the practical implementation of three optimization techniques: Iterative Reweighted Least Squares (IWLS), Stochastic Gradient Descent (SGD) and Adaptive Moment estimation (ADAM), with an emphasis on coding these algorithms from scratch and integrating variable interactions that increase model complexity. The final task is an experimental evaluation using balanced accuracy as a performance metric and comparing the optimization algorithms with four popular classification methods. This structured study aims to elucidate the relative effectiveness of each optimization strategy in the context of logistic regression, contributing valuable insights to the field.

2. Convergence Analysis

2.1. Datasets

The basis of every successful classification project is the meticulous preparation of datasets, which includes both the selection of appropriate data and their subsequent preprocessing. The purpose of this study is to cover the initial but crucial steps in analyzing binary classification problems by focusing on collecting and preparing datasets to perform experiments with logistic regression algorithms.

Considering the key role of data in machine learning and the challenges of finding suitable and properly formatted datasets, we took on the task of collecting 9 different datasets corresponding to binary classification problems. These datasets were selected based on specific criteria to ensure variety and relevance to the task at hand. Specifically, we searched 3 small datasets with at most 10 variables and 6 larger datasets with more than 10 variables, ensuring that all datasets contain more observations than variables to facilitate effective model training and evaluation.

In our dataset search, we targeted well-known repositories such as the UCI Machine Learning Repository, among others.

Preparing these datasets for logistic regression analysis involved two key steps: checking for empty cells and collinear variables in the dataset. Additionally, recognizing the detrimental impact of collinearity on the performance of logistic regression models, we have carefully analyzed the datasets to identify and remove collinear variables, thus ensuring that our models are based on datasets that are not only clean, but also optimally structured for analysis.

2.1.1. Diabetes detection dataset

The Diabetes detection dataset contains the following eight independent variables: number of pregnancies the patient has had (Pregnancies), glucose level (Glucose), blood pressure (BloodPressure), skin thickness (SkinThickness), insulin level (Insulin), BMI level (BMI), diabetes pedigree function (DiabetesPedigreeFunction), age (Age) and class (Outcome), which is the dependent variable indicating the presence (marked as value 1) or absence (marked as value 0) of diabetes.

First, a verification was carried out for the presence of NaN values in the data set, but their presence was not found. A correlation matrix was then used to see how the variables were interdependent, which would help in assessing multicollinearity.

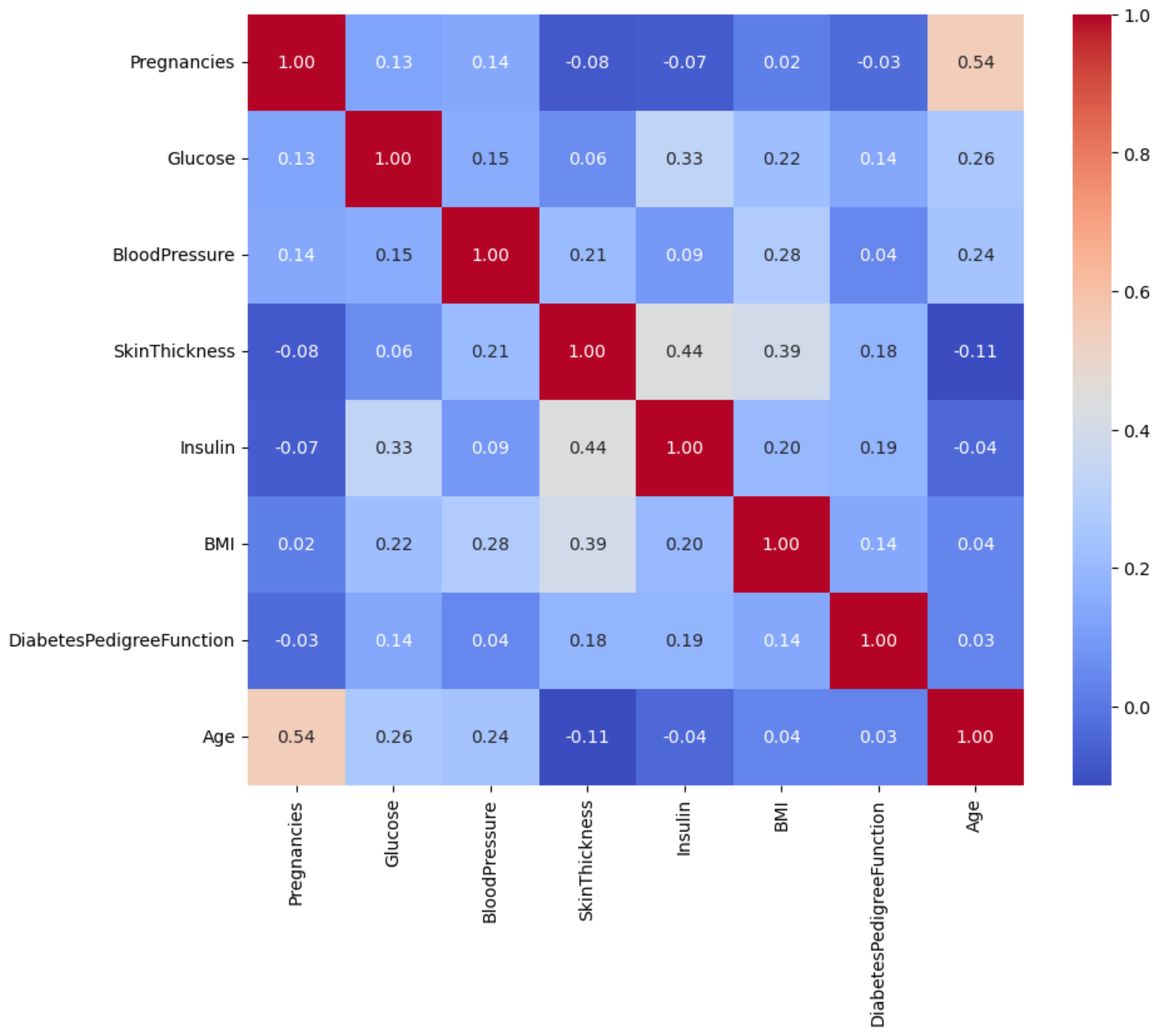

Figure 2.1. Correlation matrix of Diabetes dataset variables.

The correlation matrix is plotted (Figure 2.1). Correlation values range from -1 to 1, where 1 means perfect positive correlation, -1 means perfect negative correlation, and 0 means no correlation. Significant correlation coefficients (close to 1 or -1) may indicate multicollinearity between variables.

If we look at the matrix, the highest positive level of correlation coefficient of 0.54 is between the Age and Pregnancy variables, while the highest negative level between the Age and Skin Thickness variables is -0.11. This means that most variables do not show strong multicollinearity (values above 0.7 and below -0.7 [chcę tu podać książkę, ale muszę sprawdzić jak się nazywała]), which means that they may be relatively independent. However, the correlation values are not very high for any pair of variables as proven, suggesting that the multicollinearity problem may not be relevant for this dataset.

2.1.2. Credit fraud detection dataset

The credit fraud detection dataset contains 5 independent quantitative variables and a categorical variable that is a binary indicator indicating whether a transaction is fraudulent (1) or legitimate (0). The explanatory variables in the dataset include: Amount (the transaction amount in currency), Age (the user's age), Income (the user's income), Debt (the amount of debt associated with the user), and CreditScore (the user's credit score).

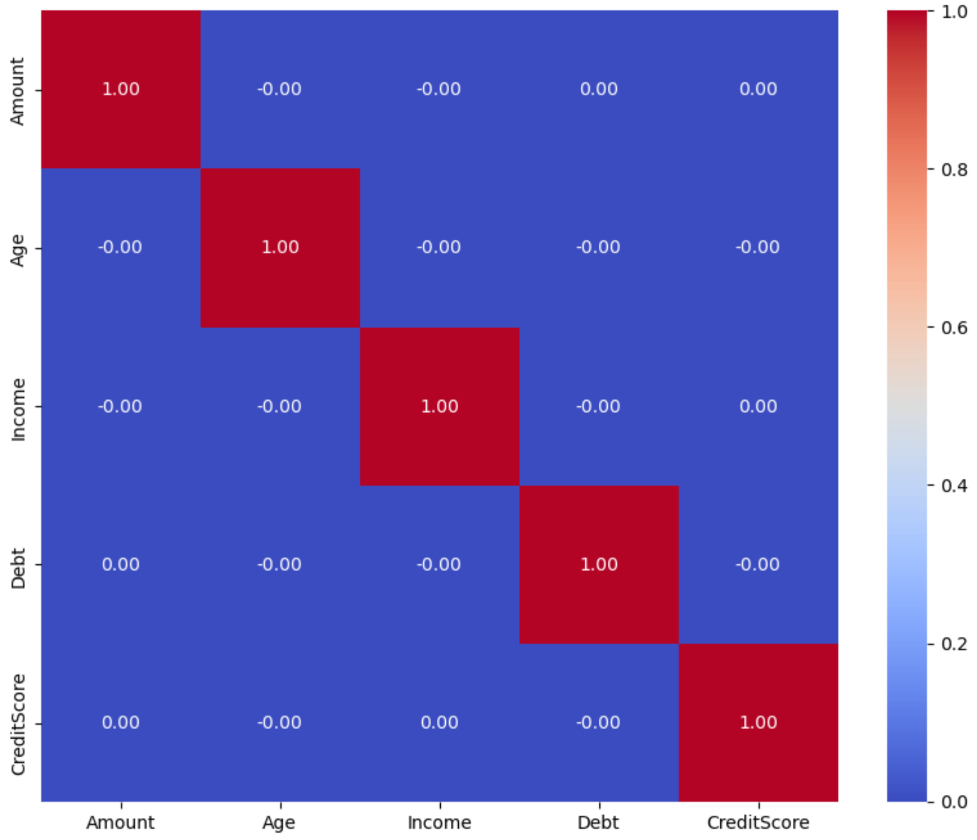

Figure 2.2. Correlation matrix of Credit fraud detection variables.

Figure 2.2 shows the correlation matrix of fraud detection data. There is no significant multicollinearity between the variables, in the expansion no correlation between the variables is greater than between Age and Debt (-0.001130), which indicates that this dataset does not require the reduction of multicollinearity of variables.

2.1.3. Banknote authentication dataset

The banknote authentication dataset consists of 5 columns representing various attributes associated with banknote images. These attributes are used to classify banknotes as genuine or counterfeit (binary class, dependent variable, values marked as 0 and 1, respectively). The explanatory variables in this dataset are: Variance (this variable contains the variance values of the transformed banknote images), Skewness (skewness is a measure of the asymmetry of the distribution of pixel values in the transformed banknote image), Kurtosis (kurtosis refers to a measure of the "peak" distribution of pixel values), Entropy (in the context of banknote authentication, entropy may provide information about the degree of order or chaos in the image structure, which may differ between a real and a counterfeit banknote).

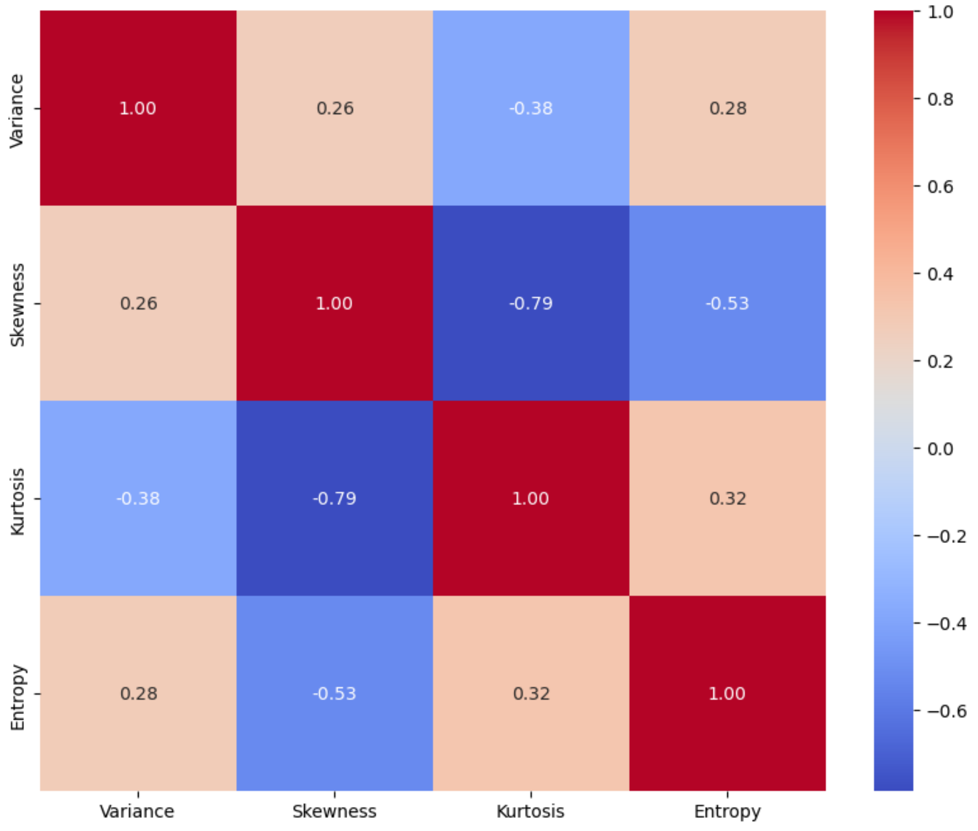

Figure 2.3. Correlation matrix of Banknote authentication dataset variables.

The dataset is fully populated with values. Analyzing the correlation plot in Figure 2.3, the highest negative correlation is between Skewness and Kurtosis, with a correlation coefficient of -0.79. This indicates a strong inverse relationship between these two characteristics. As Skewness increases (indicating asymmetry in the distribution of values), Kurtosis tends to decrease (indicating a flatter distribution). Other notable correlations include the negative correlation between Skewness and Entropy (-0.53) and the negative correlation between Variance and Kurtosis (-0.38), although these are less pronounced than the correlation between Skewness and Kurtosis. The correlations between Variance and Skewness, Variance and Entropy, and Kurtosis and Entropy are positive, but relatively weak.

Given the strong negative correlation between Skewness and Kurtosis, we paid special attention to how these features were used in modeling. While it may be tempting to remove one of them to reduce multicollinearity, we took into account the potential loss of information and the impact on model performance. Techniques such as Principal Component Analysis (PCA) for dimensionality reduction can be a good approach to managing multicollinearity without directly removing variables, so we used this method.

After applying PCA to reduce the dimensionality of the banknote authentication dataset to two principal components, we observe that the first principal component explains approximately 54.5% of the variance, while the second principal component accounts for approximately 32.3% of the variance. Together, these two components capture approximately 86.8% of the total variance in the dataset, indicating a significant reduction in dimensions while retaining a significant amount of information. This transformation results in new features (principal components) that are linear combinations of the original variables, with these new features being orthogonal to each other (uncorrelated).

2.1.4. Connectionist Bench (Sonar, Mines vs. Rocks)

Each pattern is a set of 60 numbers in the range 0.0 to 1.0. Each number represents the energy within a particular frequency band, integrated over a certain period of time. The integration aperture for higher frequencies occur later in time, since these frequencies are transmitted later during the chirp. **sonar**. The label associated with each record contains the letter "R" if the object is a rock and "M" if it is a mine (metal cylinder). R is mapped to 0 and M is mapped to 1. Target classes are pretty balanced as per:

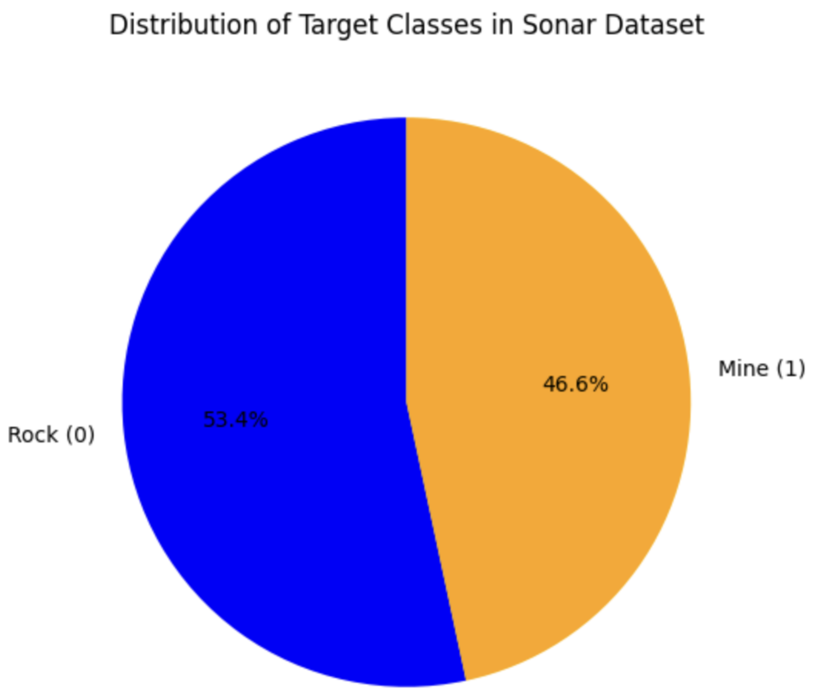

Figure 2.4. Distribution of target class in sonar dataset

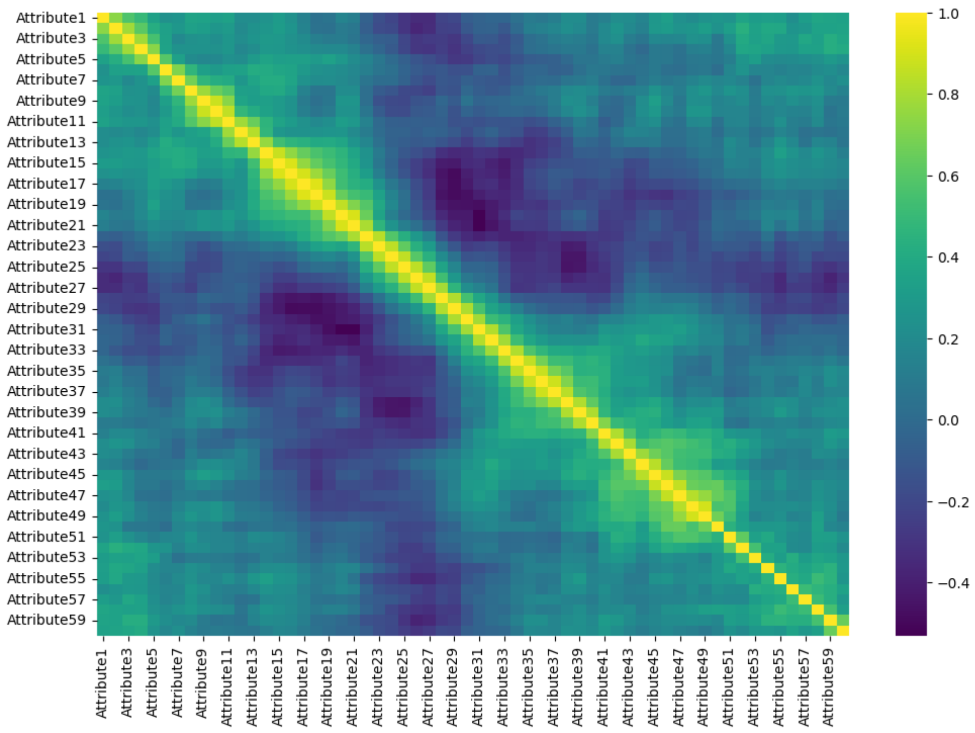

Figure 2.5. Correlation of Connectionist Bench (Sonar, Mines vs. Rocks) dataset

2.1.5. UCI ML Breast Cancer Wisconsin (Diagnostic) dataset

It contains features computed from digitized images of fine needle aspirate (FNA) biopsy samples of breast masses. The features quantify characteristics of the cell nuclei present in the images and include measurements of radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, and fractal dimension. The dataset has 569 instances with 32 attributes each, where each instance corresponds to a different image of a breast mass. The attributes include an ID number, diagnosis (benign or malignant), and the 30 real-valued input features Feauters that achieved correlation

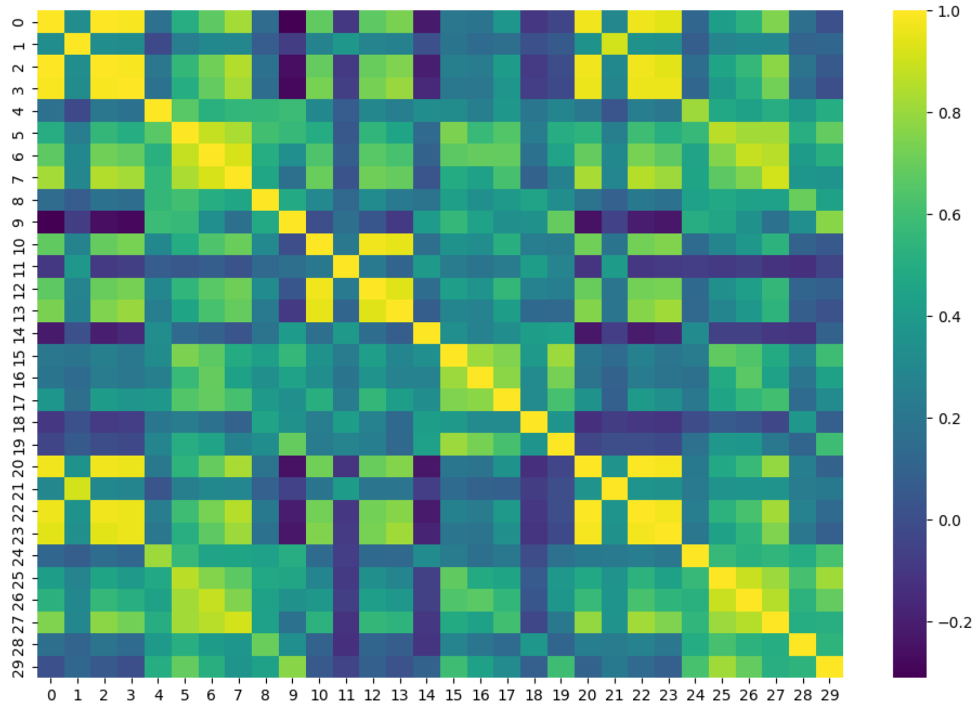

Figure 2.6. Correlation matrix for Breast Cancer Wisconsin (Diagnostic) dataset

greather than 0.9 were removed. Here is the list of them: mean perimeter, mean area, mean concave points, perimeter error, area error, worst radius, worst texture, worst perimeter, worst area, worst concave points. As imported from sklearn, dataset had already mapped diagnosic class into binary.

2.1.6. Ionosphere

This radar data was collected by a system in Goose Bay, Labrador. This system consists of a phased array of 16 high-frequency antennas with a total transmitted power on the order of 6.4 kilowatts. The targets were free electrons in the ionosphere. "Good" radar returns are those showing evidence of some type of structure in the ionosphere. "Bad" returns are those that do not their signals pass through the ionosphere. "Good" was mappeed to 1 and "bad" was mapped to 0. The majority of observations are marked as bad radar.

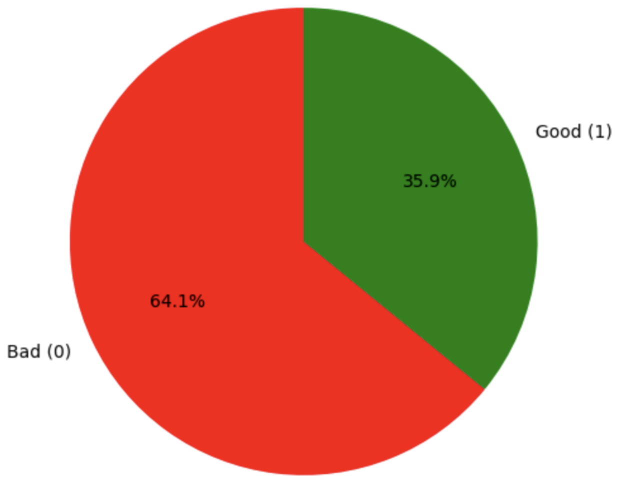

Figure 2.7. Distribution of target class in ionsphere dataset

2.1.7. Heart Disease

This dataset contains medical information about 303 patients with a target variable being their risk of developing heart problems. The dataset was preprocessed a bit before being uploaded to the https://archive.ics.uci.edu/, but it required a bit more work to make it suitable for a binary classification task. Namely we dropped the 6 rows containing missing values (as losing 2 percent of observations isn't problematic), we one-hot encoded the categorical columns and reduced the target variable to a binary 0 for no heart probles and 1 for some heart problems detected. No collinearity between columns was detected (obviously we one-hot encoded using the first-drop rule).

2.1.8. Parkinsons

This dataset contains preprocessed voice data from patients, some of which suffer from Parkinsons disease. The model intents to predict whether a patient is ill using the informations about a voice sample.

All the features in this data set are continuous and have no missing values, but in our preprocessing we had to remove several columns to achieve the linear independence of the experiment matrix.

2.1.9. HCV

This dataset contains medical information about 615 patients and categorizes them based on their ability to donate blood.

This dataset needed a bit of preprocessing, as five of the columns had missing values in them, two of which, a quite large number of them, so we dropped the three rows that had missing values in unusual places and then filled the missing values in the two critical columns using Linear Regression model built on remaining features. Furthermore there was a need to map some columns to a {0,1} binary values, including the target variable, which previously differentiated between different reasons for the individual to be banned from blood donations - which we turned into 0 if individual can donate blood and 1 if they cannot.

2.2. Optimization algorithms

2.2.1. Iterative Reweighted Least Squares (IWLS)

2.2.1.1. Overview

Iterative Reweighted Least Squares is a method one will get when attempting to create a maximum likelihood estimator for the parameters of the Logistic Regression using the Newton's approach. This approach uses the fact that the minus log-likelihood function in Logistic Regression is convex (it's even convex after introducting the L2 penalty for the norm of the parameter vector), so it's enough to find the point at which the gradient of said function takes the value of 0. That point is then found iteratively using the first-order Taylor Series approximation of the gradient. For more information about the method we recommend prof. Mielniczuk's lecture about the Logistic Regression, the slides for which were used as a reference.

2.2.1.2. Implementation

We have created a function which takes the training data and returns the parameter vector, a dictionary with labels for printing purposes, the history of the parameter vector across iterations and the history of the value of log-likelihood function across iterations.

Said function had the options of including the intercept (which is achieved by adding a column of ones to the experiment matrix), including the product interactions described in the section 2.2 of the project description, including a generator for the starting values of parameters (we didn't use it though, opting instead to go for initializing all parameters to 0 at the start), adjusting the L2 regularization strength (λ), setting the maximum number of iterations, the maximum time of running the algorhithm as well as the minimum change in the parameter vector (the threshold for the euclidian norm of the difference in parameter vectors between iterations, which was the basis for our main stopping rule).

In each iteration, the algorhithm used the linear equation to define the new value of the paramter vector, as follows:

$$H_{(\beta_{old})}\beta_{new} = H_{(\beta_{old})}\beta_{old} - D_{(\beta_{old})}$$

where H is the Hessian matrix and D is the value of the gradient. According to our calculations $H_{(\beta_{old})} = X^T W X + 2\lambda I$ and $D_{(\beta_{old})} = X^T(P - Y) + 2\lambda\beta_{old}$, where P is the vector of probability predictions for the β_{old} parameter vector and W is a diagonal matrix having $P \odot (1 - P)$ as its main diagonal (the $\odot$ signifies element-wise multiplication).

We chose to avoid the equation from the lecture slides, as inverting matrices is numerically dangerous and whenever possible, should be replaced with the *np.linalg.solve()* function.

2.2.2. Stochastic Gradient Descent (SGD)

2.2.2.1. Overview

The Stochastic Gradient Descent (SGD) algorithm is a popular optimization technique used in machine learning to train models efficiently, especially for large data sets. Its application to logistic regression, a widely used method for classification problems, allows rapid convergence toward an optimal set of parameters (or weights) that minimize the logistic loss function. Logistic regression models probabilities for classification problems with two possible outcomes, making it suitable for binary classification tasks.

2.2.2.2. Implementation

Our implementation of logistic regression with SGD in Python includes the process within the *LogisticRegressionSGD* class. This choice increases code reusability and readability. The class supports the inclusion of feature interactions - additional features derived from multiplying pairs of original features that can capture complex relationships in the data and potentially improve model performance.

Our implementation of logistic regression with SGD in Python includes the process within the LogisticRegressionSGD class. This choice increases code reusability and readability. The class supports the inclusion of feature interactions - additional features derived from multiplying pairs of original features that can capture complex relationships in the data and potentially improve model performance.

Key implementation elements:

Feature interactions: The *add_interactions* method dynamically generates interaction terms from a set of input features, increasing the expressivity of the model without manual feature engineering.

Gradient calculation: The core of SGD, the gradient of logistic losses with respect to model weights, is calculated in the *compute_gradient* method. This gradient tells you how to adjust the weights to reduce prediction error.

Model fitting: The *fit* method iterates through the dataset, using SGD to update the model weights based on a randomly selected instance at each step. This stochastic nature greatly speeds up convergence by using a subset of the data in each iteration, making it very efficient for large datasets.

Prediction: Once trained, the *predict_proba* and *predict* methods allow for positive class probability estimation and binary prediction, respectively, based on a specified threshold.

The implemented SGD optimizer uses default parameters recommended in the literature, such as a learning rate of 0.01 and a predefined number of iterations, to provide a balance between computation time and convergence quality. These parameters will be adjusted in laters for specific needs or datasets to achieve optimal performance.

2.2.3. Adaptive Moment estimation (ADAM)

2.2.3.1. Overview Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments. Its main advantage is that it requires minimal memory footprint and is invariant to diagonal rescale of the gradients. ADAM is computationally efficient, has little memory requirements, and is well suited for problems with large data or parameters.

2.2.3.2. Implementation

Given feature matrix $\mathbf{X}$, target vector $\mathbf{y}$, learning rate α, and exponential decay rates for moment estimates β_1 and β_2, the function optimizes the weights of the logistic regression model.

The algorithm initializes weights $\mathbf{w}$ and moments $\mathbf{m}$ and $\mathbf{v}$ as zero vectors. In each iteration $\mathbf{t}$, it computes the gradient $\mathbf{g}$ of the logistic loss function, updates the first moment:

$$\mathbf{m} = \beta_1 \mathbf{m} + (1-\beta_1)\mathbf{g} \tag{1}$$

and second moment:

$$\mathbf{v} = \beta_2 \mathbf{v} + (1-\beta_2)\mathbf{g}^2 \tag{2}$$

Then for bias correction for the moments:

$$\mathbf{m}_{\text{corrected}} = \frac{\mathbf{m}}{1-\beta_1^t} \tag{3}$$

$$\mathbf{v}_{\text{corrected}} = \frac{\mathbf{v}}{1-\beta_2^t} \tag{4}$$

and finally updates weights as following:

$$\mathbf{w} = \mathbf{w} - \alpha \frac{\mathbf{m}_{\text{corrected}}}{\sqrt{\mathbf{v}_{\text{corrected}}} + \epsilon} \tag{5}$$

2.3. Stopping rule

When developing and optimizing logistic regression models, we implemented a universal stopping rule in these three different optimization algorithms presented in Section 2.2. This principle aims to increase computational efficiency and ensure timely convergence of the optimization process, without compromising model performance.

The stopping rule is based on monitoring the change in the model's weights across iterations. Specifically, the rule triggers an early termination of the training process when the norm of the weight change falls below a predetermined *threshold*. This *threshold* is a critical hyperparameter that allows us to adjust the sensitivity of the stopping condition according to the specific requirements of each algorithm and the dataset at hand.

2.4. Balanced Accuracy as performance measure

2.4.1. Overview

In the context of this project, which aims to implement and compare different optimization algorithms for logistic regression, Balanced Accuracy was chosen as the primary performance measure, allowing for a fair and accurate assessment of the effectiveness of each algorithm. This metric is particularly important due to its ability to provide a more detailed picture of model performance in both classes in binary classification problems, which is a critical aspect for the datasets selected for experiments. By averaging the recall scores of each class, Balanced Accuracy ensures that the model's predictive capabilities are not biased towards the majority class, which is a common problem in unbalanced datasets. This choice of performance measure is consistent with the project's goal of accurately and fairly comparing different logistic regression optimization strategies, ensuring that the rating reflects true algorithmic performance in handling diverse and potentially unbalanced datasets.

2.4.2. Results

2.4.2.1. IWLS

The Fraud Detection dataset turned out to be too big to perform the algorhithm with the RAM resources of Google Colab, but here are the average balanced accuracies for the IWLS approach (all results have been achieved using 5-fold crossvalidation):

Diabetes Detection:

No interactions: 0.702

With interactions: 0.719

Banknote Authentications:

No interactions: 0.99

With interactions: 1.0 (and given the fact it's a crossvalidation estimate, the data is probably linearly separable using that kernel)

Sonar: 0.78

Breast Cancer: 0.937

Ionosphere: 0.81

Heart Disease: 0.84

Parkinsons: 0.739

HCV: 0.854

2.4.2.2. SGD

The performance of the Stochastic Gradient Descent algorithm varied across datasets, showing a range of results on both small and large datasets.

On small datasets, SGD achieved moderate Balanced Accuracy on the Diabetes detection dataset with a score of approximately 0.593. Its performance was lower on the credit Fraud detection dataset, with a Balanced Accuracy of 0.5. However, in the Banknote authentication dataset it performed much better, with a score of around 0.806.

Looking at larger datasets, SGD showed a Balanced Accuracy of around 0.631 in the Sonar dataset and slightly higher in the Breast dataset at around 0.659. The Ionsphere dataset improved further, with a sustainable accuracy of approximately 0.734.

For Heart disease, the Balanced Accuracy dropped slightly to around 0.606, while in the Parkinson's disease dataset it matched the lowest result observed in small datasets with a Balanced Accuracy of 0.5. Finally, SGD showed its best performance on the HCV dataset, achieving a sustainable accuracy of around 0.818.

In general, SGD performance was relatively more effective for datasets with very distinct features (banknote authentication, HCV) or less complex interactions between features (Breast, Ionsphere). In contrast, he struggled with datasets where perhaps more sophisticated models or feature engineering could improve performance (Heart disease, Parkinson's disease, Credit fraud detection).

2.4.2.3. ADAM Adam in comparision with other optimizers was slightly underperforming. The possible reason might be low complexity or small number of observations of the datasets. When we look at Credit fraud dataset with the biggest number of observations, we can observe the best performance. The biggest problem has with Parkinson's disease dataset which is often limited in size due to the difficulty of collecting medical data. ADAM, like other optimizers, may struggle to generalize from a small training set.

2.5. Convergence analysis

In this project, an element of our study is the analysis of the convergence of three optimization algorithms: IWLS, SGD and ADAM in the context of logistic regression. This analysis focuses on examining how the value of the log-likelihood function, a measure of how well the model fits the training data, evolves with each iteration of the algorithms. The essence of this exercise lies in its ability to illuminate the efficiency and effectiveness of each algorithm's approach to finding optimal parameter values that maximize the log-likelihood function. By tracking log-likelihood values across iterations, our goal is to gain insight into the convergence behavior of algorithms - that is, how quickly and reliably they approach a solution.

The convergence analysis for the SGD algorithm was performed using a synthetic dataset composed of 1000 samples and 3 informative features, designed to create a balanced and straightforward evaluation scenario. This dataset enabled a clear examination of SGD's iterative process in converging towards the optimal logistic regression model parameters.

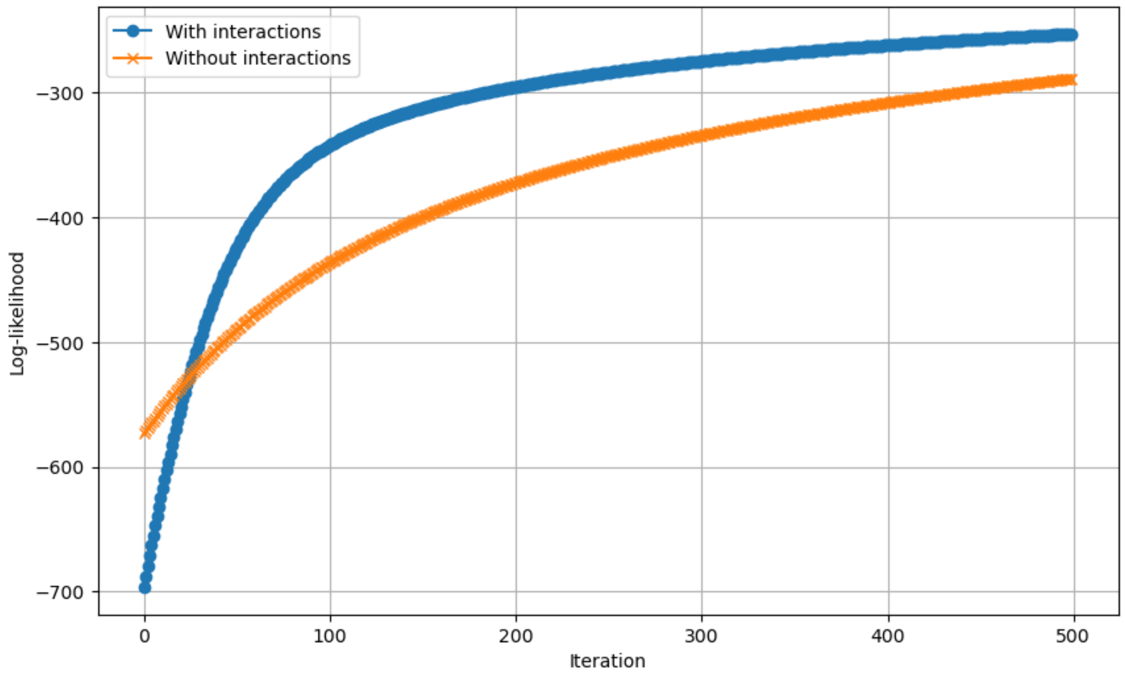

Figure 2.8. Convergence analysis (SGD) - log-likelihood over iterations.

The Figure 2.4 displays the convergence of SGD in logistic regression over 500 iterations, comparing models with and without variable interactions. The blue line represents the model with interactions, showing a lower initial log-likelihood that eventually exceeds the orange line, the model without interactions. Both models level off, indicating convergence, with the interactions model concluding with a higher log-likelihood, suggesting a potentially better fit.

3. Comparison of classification performance

In our comparative study, we evaluate the classification performance of three logistic regression optimization methods - IWLS, SGD and ADAM, and four established classification methods: Linear Discriminant Analysis (LDA), Quadratic Analysis Discriminant (QDA), Decision Tree and Random Forest. Across nine datasets, we use Balanced Accuracy as the sole performance metric to ensure a fair and consistent assessment of each method's ability to deal with class imbalance.

Small datasets:

Name	Heart Disease	Parkinsons	HCV	Sonar	Breast Cancer	Ionsphere
IWLS	0.84	0.74	0.85	0.78	**0.94**	0.81
SGD	0.80	0.60	0.77	0.71	0.69	0.68
ADAM	0.80	0.60	0.77	0.71	0.69	0.69
LDA	**0.87**	0.69	0.68	0.57	0.88	0.84
QDA	0.60	**0.80**	0.80	0.74	0.90	**0.93**
Decision Tree	0.85	0.78	0.86	0.62	0.89	0.88
Random Forest	0.82	**0.80**	**0.90**	**0.81**	0.91	**0.93**

Table 3.1. Comparision Table with Balanced Accuracy for Big datasets

Diabetes detection dataset: SGD shows lower balanced accuracy (0.593) compared to other methods, with LDA performing the best (0.7333).

Credit fraud detection dataset: SGD has a balanced accuracy of 0.5, significantly lower than other methods, with Decision Trees and Random Forest achieving scores (1.0000 - overfitting?).

Banknote authentication dataset: SGD offers reasonable performance (0.8063), but is outperformed by Decision Trees and Random Forest, with Random Forest having the highest balanced accuracy (0.8739).

Large datasets:

Connectionist bench dataset: SGD's performance (0.6312) is better than LDA but less than QDA, Decision Trees, and Random Forest, with Random Forest leading (0.8095).

Breast Cancer wisconsin (diagnostic) dataset: SGD (0.6591) is again outperformed by other methods, particularly QDA and Random Forest, which offer the highest accuracies.

Ionsphere dataset: While SGD achieves a balanced accuracy of 0.7343, it is significantly lower than that of QDA and Random Forest, with Random Forest topping the chart (0.9331).

Heart disease dataset: SGD shows a balanced accuracy of 0.6055, which is competitive with QDA (0.5957) but less than LDA, Decision Trees, and Random Forest.

Parkinsons dataset: SGD is at the baseline accuracy of 0.5, suggesting it may not be effectively capturing the patterns in the dataset, while all other methods perform markedly better.

HCV dataset: SGD (0.8182) demonstrates its best performance among the large datasets yet still falls short of Decision Trees and Random Forest, with Random Forest again achieving the highest balanced accuracy (0.9062).

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

Name	Banknote Authentication	Diabetes	Fraud
IWLS	0.99	0.99	-
SGD	0.81	0.59	0.5
ADAM	0.60	0.76	0.87
IWLS with interactions	1.0	0.72	-
SGD with interactions	0.816	0.526	0.5
ADAM with interactions	0.59	0.67	0.50

Table 4.1. Comparision Table with Balanced Accuracy for Big datasets

The models were compared on three small datasets (Diabetes detection dataset, Credit fraud detection dataset and Banknote authentication dataset) based on the Balanced Accuracy metric, both for the model without and with interactions.

4.1. IWLS

4.2. SGD

For the Diabetes detection dataset, the average Balanced Accuracy of the logistic regression model without interactions was approximately 0.593. When interactions between variables were included, the average Balanced Accuracy dropped to about 0.526.

In the Credit fraud detection dataset, the logistic regression models, both with and without variable interactions, achieved an average Balanced Accuracy of 0.5. This suggests that including interactions between variables did not enhance the model's performance for this particular dataset.

For the Banknote authentication dataset, the model's performance improved when interactions were considered. The average Balanced Accuracy without interactions stood at approximately 0.806, and with interactions, it slightly increased to about 0.816. This indicates that for this dataset, incorporating interactions between variables provided a better classification of authentic and inauthentic banknotes.

4.3. ADAM

Adding interactions to Adam optimizer resulted in worse results

5. Development environment

To implement the project, we used the Python programming language and the Project Jupyter programming environment. This project is an Open Source service that allows you to perform programming tasks in the field of data analysis and machine learning. Thanks to the Jupyter Notebook application provided by Project Jupyter, you can create the so-called notebooks that allow you to compile parts of the code at any time and make it easier for the user to focus on a specific part of the code. Another convenience for developers is the ability to connect notebooks to the GitHub online service, which enables code hosting, version control and remote user collaboration in IT projects.