

EN3150 -Pattern Recognition

Assignment 02

Learning from data and related challenges and classification

W.C.A Wickramaarachchi

210699K

01/10/2024

**1) Logistic Regression**

1) Load the data

2) The line in the code both fits the Label Encoder and transforms the “species” column into numerical labels. This process assigns unique integers to each species name 'Adelie' and 'Chinstrap'. This process of converting String label to numeric label is essential for making logistic regression models. Other thing is because this “species” feature is the target variable of our model we must have numerical inputs.

3) This given line is the one create the feature matrix X by excluding the columns “species, island and sex” from the data frame df. Because the species is the target variable for modeling, we must exclude it from X matrix. “Island” and “sex“ may not be necessary for predictions. “axis=1” in the code specifies the things should be dropped are columns, not rows.

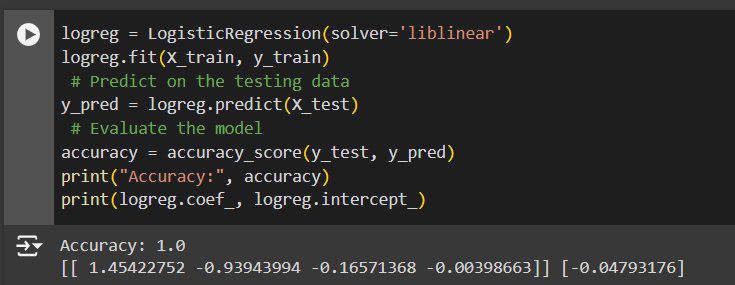
4) First thing is “island and “sex” features are categorical variables. Logistic regression requires numerical inputs, and before feeding categorical variables to the model wehave to encode them. Additionally these features might be considered less directly relevant because this model is train by morphological measurements. Removing less important features can help in simplifying the model, reducing overfitting effects.

5) training logistic regression model.

6) The random state parameter in the train\_test\_split function ensures the consistent data splits, Reproducibility and No Data leakages happen. While maintaining the integrity of testing data by ensuring it is randomly selected but consistent across different runs, it prevent any accidental data leakage that might be happen during training phases.

7) Logistic regression model “saga” is sensitive to feature scaling. If the numerical features have very different ranges, the model will not perform well. It is better to use “standardScaller” or similar method before training. The other thing which affect to the “saga” regression model is, it is designed for large datasets and handles L1 or elastic-net regulations. If the regularization is not needed or the data set is small, it might not perform well. Theres a convergence warning saying that the solver struggle with the current feature scales, coefficients didn’t converge.

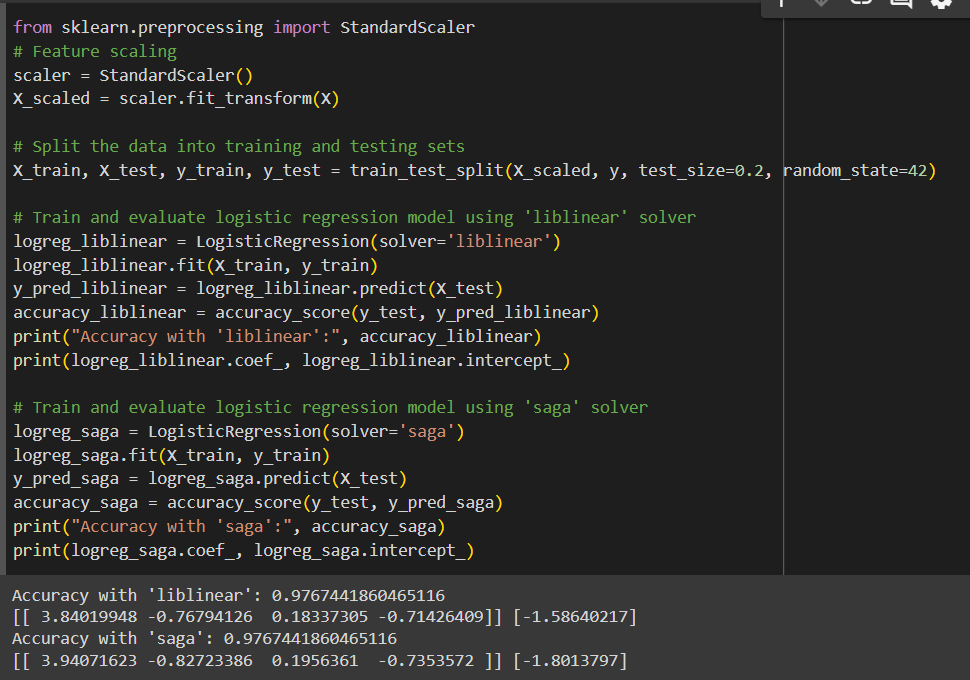
8)



Accuracy:- 1.0

9) The “liblinear” solver is better for small data sets because it is optimized for binary classification . Saga is not good in smaller or less complex datasets. Other thing is liblinear solver don’t need feature scaling methods . But because “saga” solver use stochastic gradient descent it need feature scaling. Other thing is the way ‘liblinear” and “saga:” handle regularization can affect their performances. The default regularization in logistic regression is L2 and “liblinea” might converge to a more suitable solution with L2 regularization.

10)



|  |  |  |
| --- | --- | --- |
| Solver | Unscaled Accuracy | Scaled Accuracy |
| Saga | 0.5813953488372093 | 0.9767441860465116 |
| Liblinear | 1.0 | 0.9767441860465116 |

It is clear that because of standard scaling the accuracy of the model increase to 97% using saga solver. But the liblinear solver model not benefits due to scaling and the accuracy get reduced to 97%. Anyway after feature scaling both solvers recorded a similar accuracy. This highlights , if we use gradient descent-based optimization method for modelling, feature scaling is very much important for the accuracy of the model.

11) In the code they again separate the Y value (target variable) and X values (X matrix) from the data set and in the code they didn’t remove the “island” and “sex” features which are categorical variables which are not transform to numeric values. Because they are not in numeric format we can’t applied them directly with logistic regression. If we have to use those features we have to encode them using a method like one-hot encoding. Otherwise we should drop the “island and “sex” columns from X matrix.

12) This approach is not correct.

Label Encoding convert categorical features in to arbitrary numerical value. As an example, “red”, “blue”, “green” might to encoded as 0,1,2 respectively. This encoding implies an ordinal relationship that may not exist. (as ‘green’>’blue’>’red’).  
When we scale these numbers, it essentially normalizing these arbitrary distinctions, not actual meaningful distances or relationships. This can mislead the machine learning model.   
Instead of using label encoding we can use one hot encoding to handling categorical variables. This technique transforms each category into a separate binary column, ensuring the model treats each category as an independent feature without implying any ordinality. By doing so, it preserves the unique characteristics of categorical data and prevents the model from mistakenly assigning an unintended order to the categories. This is essential for accurate model training. One-hot encoding also avoids distorting relationships between categories that might arise from scaling, ultimately improving model performance by capturing the distinct influence of each category on the target variable.

**Question 2**

1) In logistic regression, the probability p of receiving an A+ is given by the following formula.

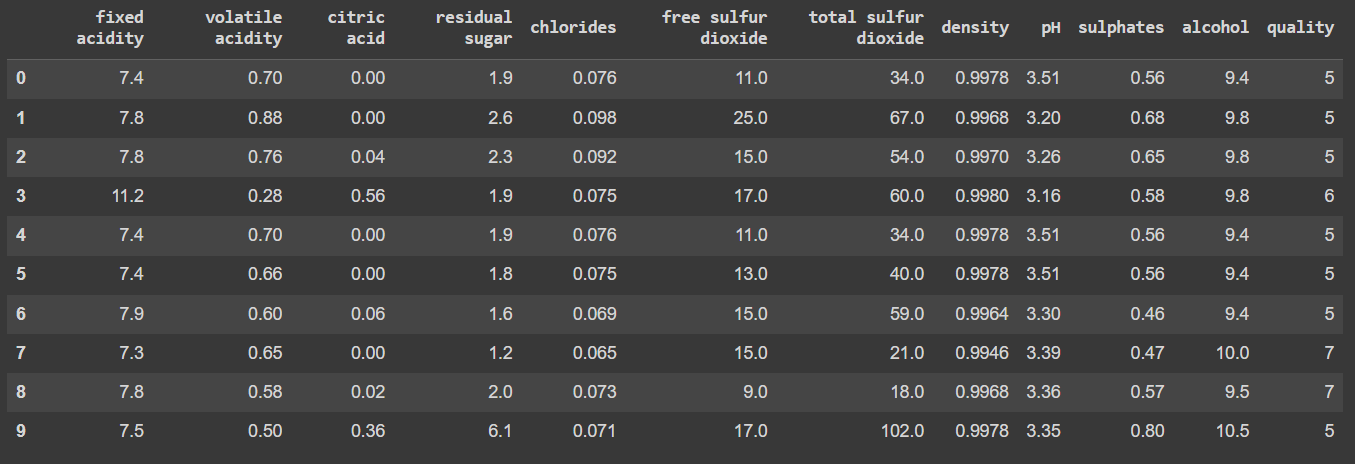
The estimated probability that the student will receive an A+ is 92.4%.

2)

Student should study 15.091 hours to achieve a 60% chance of receiving an A+

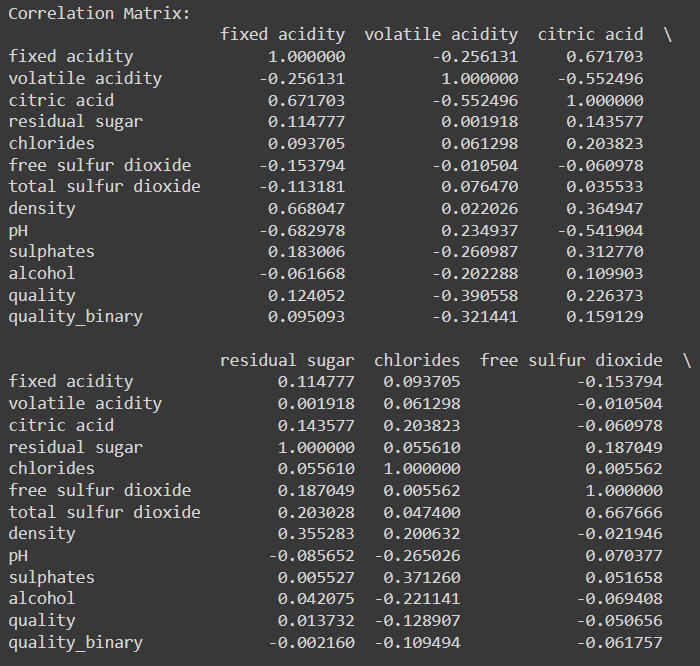
**2) Logistic regression on real world data**

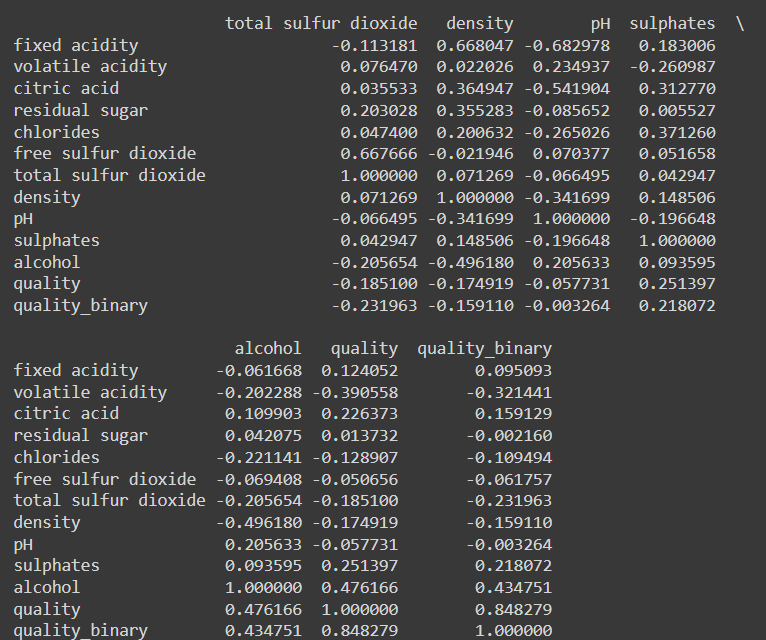
1) I chose Wine Quality data set

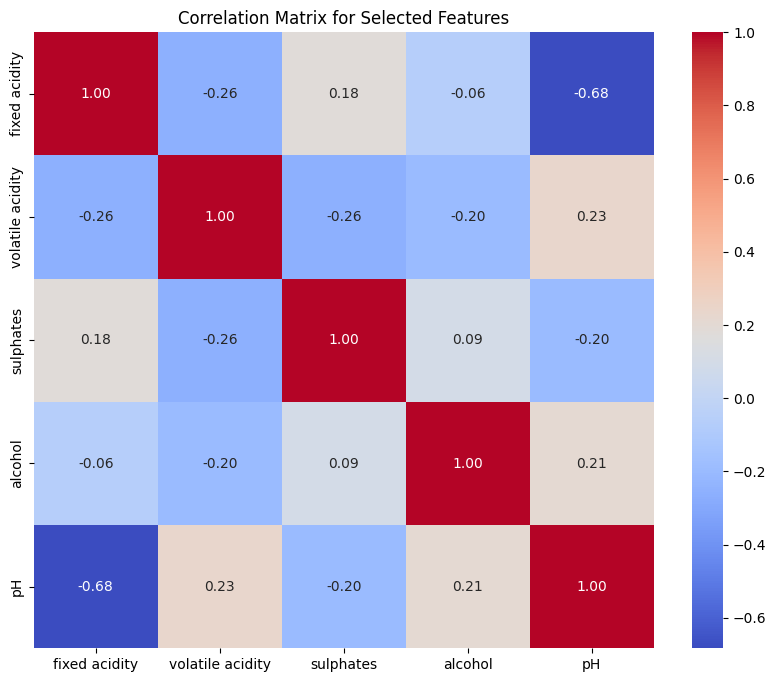


2) 

Correlation Matrix :-







I selected below 5 features to analyze the wine quality data set.

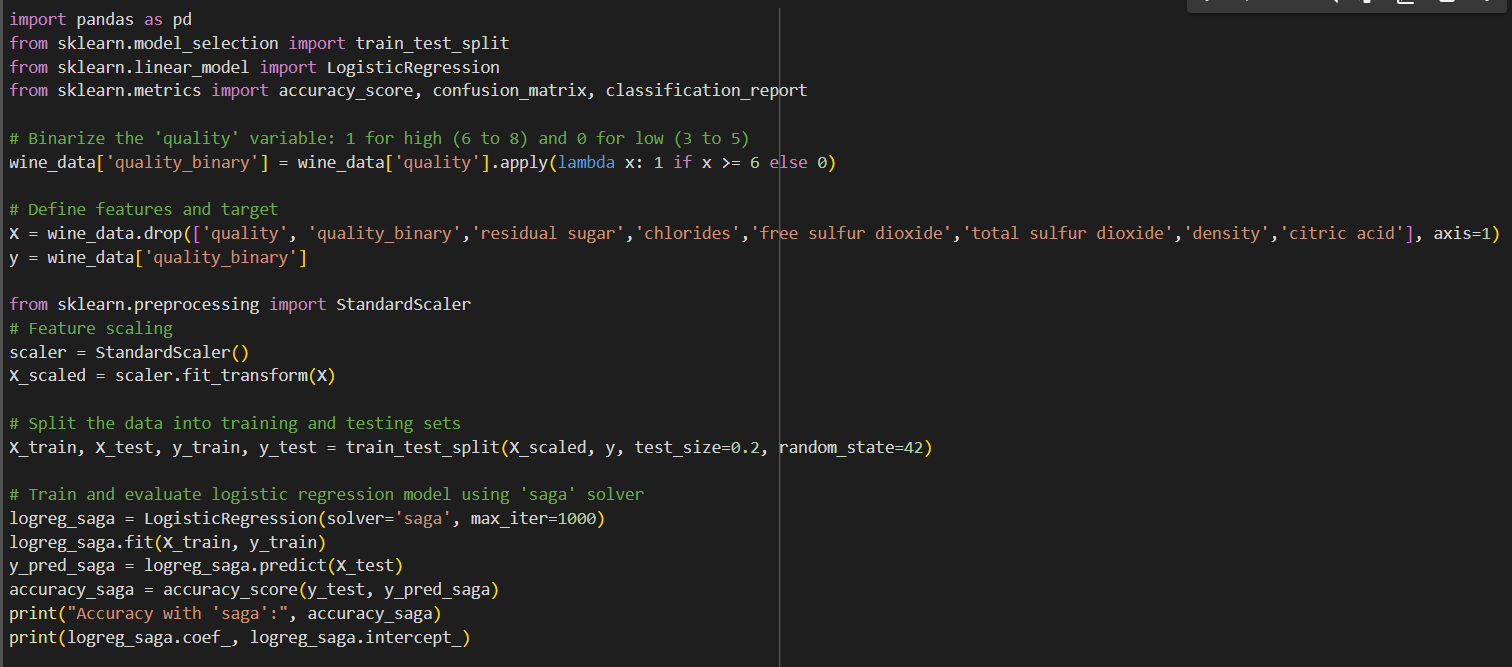
* Fixed acidity
* Volatile acidity
* Sulphates
* Alcohol
* quality

A collage of blue and white data

Description automatically generated

Pair plot of Selected feature

3)



In training my logistic regression model, I found no missing values in the dataset, eliminating the need for data cleaning to remove rows, which helped maintain model integrity. I selected five integer features for the model, and since there were no categorical features, no feature encoding was necessary. Given the large size and complexity of my dataset, I chose the 'saga' solver, which is well-suited for such conditions. To prevent any single feature from dominating due to its scale, I implemented feature scaling using the ‘StandardScaler’. Notably, this scaling improved the model's accuracy. Before scaling, the accuracy was 72.5%, and after applying standard scaling, it increased to 73.125%.

Another one special thing is in my target variable there are multiple categories regarding quality of wine. So I recategorize those values when the quality is greater then 6 it converts to 1 indicating high quality. Otherwise 0 indicating low quality.

4)

A computer screen shot of text

Description automatically generated

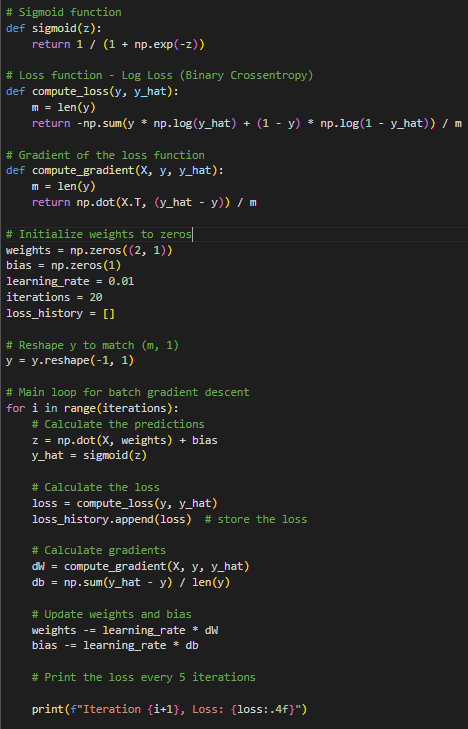
|  |  |
| --- | --- |
| Fixed Acidity | 0.052 |
| Volatile Acidity | 0.000 |
| PH | 0.255 |
| Sulphates | 0.000 |
| Alcohol | 0.000 |

Fixed acidity has a p-value that is slightly above the accepted cutoff of 0.05, indicating that it is marginally significant. This implies that it is not statistically significant, even though it might have some effect on the model. The response variable is inversely correlated with volatile acidity, which is extremely significant. In the model, it is an important predictor. The pH is not a significant predictor of the result, as seen by its high p-value. This feature could be removed from the model without compromising its efficacy because it does not statistically add to the model. Additionally, alcohol has a strong beneficial impact on the response variable. It has a big impact on forecasting the result.

**3) Logistic regression First/Second-Order Methods**

1) Generated the data using code given in listing 4.

2)



Loss in each iteration:

**A screen shot of a computer

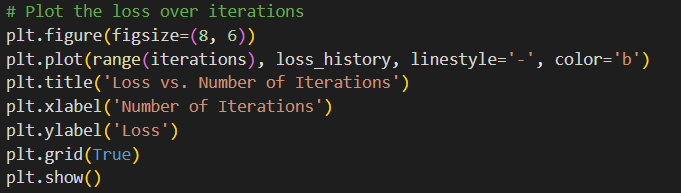
Description automatically generated**

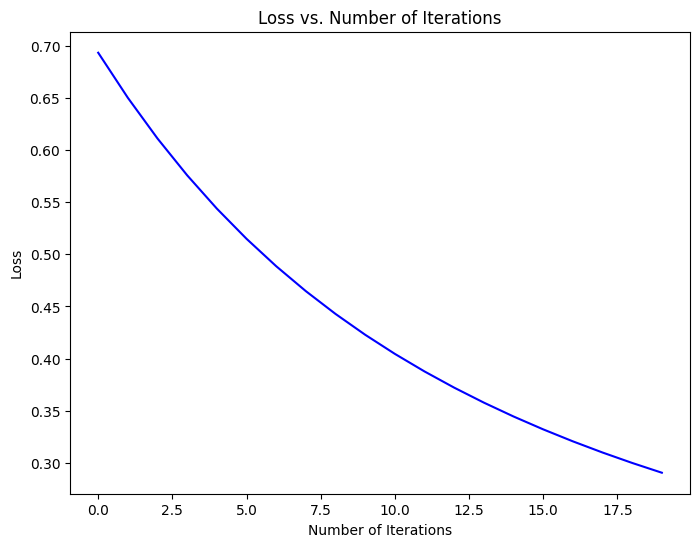
I initialized weights and bias to zeros.

Initializing weights and biases to zeros in logistic regression works well because it ensures symmetry . The cost function is convex, so gradient descent will still find the global minimum, and zero initialization doesn't prevent learning effectively. This simplicity makes zero initialization a good choice for logistic regression.

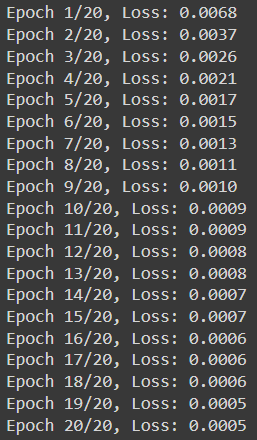
3) I used Logistic Loss function to reduce the error. Why I used Loss function in my binary classification problem is because it measures the difference between predicted probabilities and actual labels (good quality vs. bad quality). It complements the sigmoid function, penalizing incorrect predictions and ensuring smooth, differentiable gradients for batch gradient descent. This allows efficient updates to weights and biases, ensuring stable convergence to an optimal solution, as logistic loss provides a convex cost function with a guaranteed global minimum.

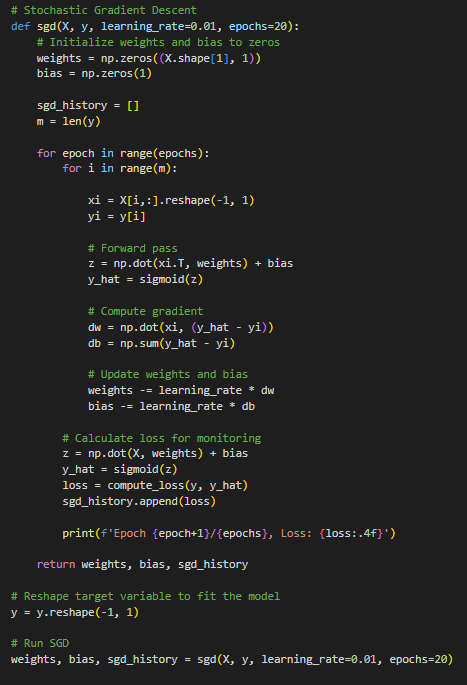
4)



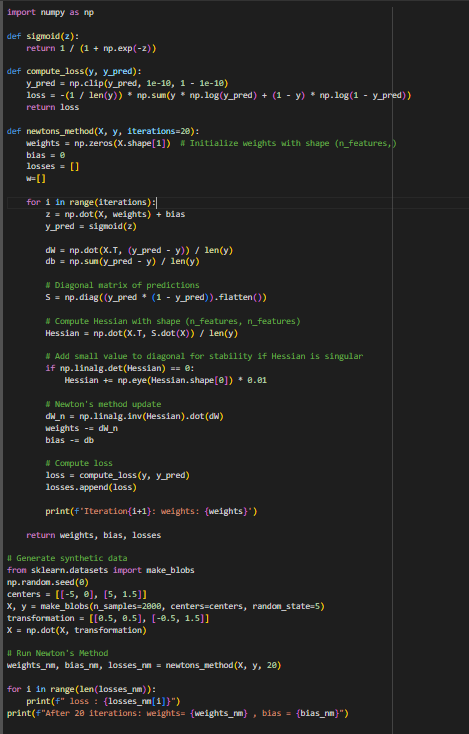
Graph of loses with respect to number of iterations 

5) Loss in each iteration:

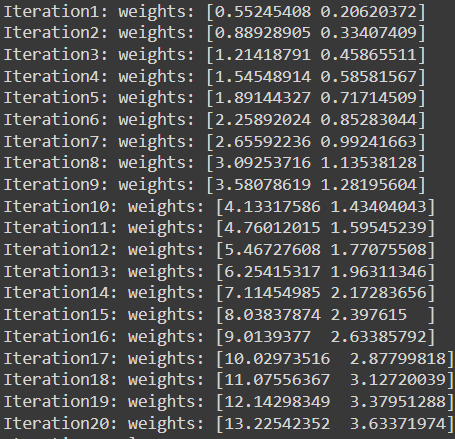
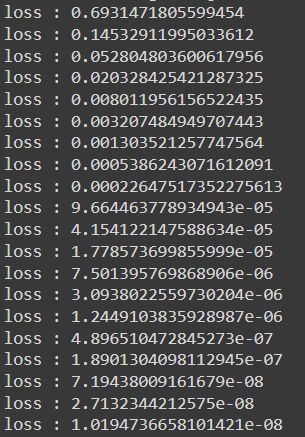




6)



Loss of Each iterations:  
 Weights in each iteration:

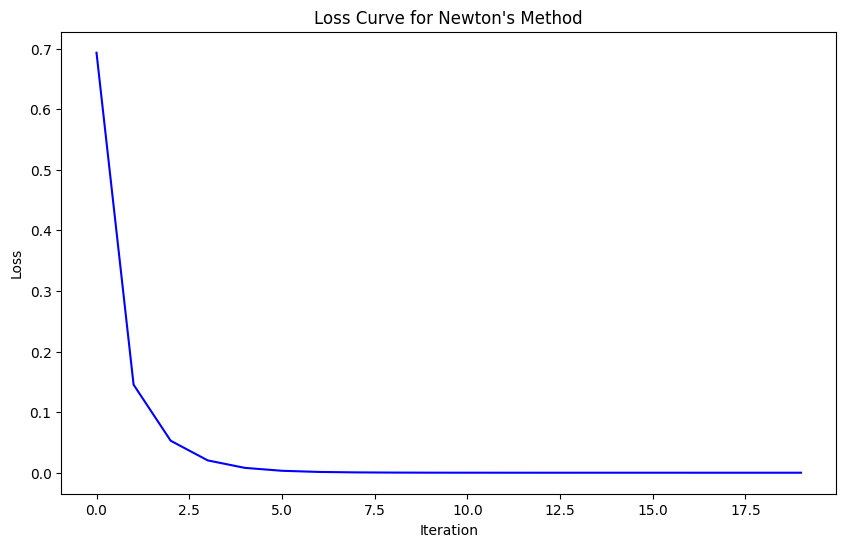
 

7)

A black screen with white text

Description automatically generated

Graph of losses with respect to number of iterations



8)

A computer screen with text on it

Description automatically generated

A graph of a comparison of optimization methods

Description automatically generated

Newton’s method exhibits the fastest convergence among the three, quickly reducing the loss and stabilizing near zero within just a few iterations. This demonstrates the method's efficiency, as it uses second-order derivatives to make more precise weight updates.

Stochastic Gradient Descent (SGD) converges more quickly than Batch Gradient Descent, with the loss dropping from 0.0068 in the first epoch to 0.0005 by the 20th. This rapid progress is typical of SGD, as it updates weights after processing each data point.

Batch Gradient Descent shows the slowest convergence, with the loss decreasing steadily but more gradually than the other methods. This is expected since BGD calculates gradients over the entire dataset in each iteration, leading to stable but slower movement towards the optimal solution.

9) **For Gradient Descent:** Two useful methods can be applied to identify the ideal number of iterations for Gradient Descent. In the first, a convergence threshold is determined by taking the loss tolerance into account. This technique maximizes computer resources by stopping the process when improvements become insignificant, when the difference in loss between the next iteration drops below a predefined modest number. In the second method, model performance is measured during training using a validation set. When validation performance declines or stops getting better, suggesting possible overfitting or convergence, the training is stopped. This technique guarantees that the model's generalization is maximized while also aiding in the prevention of overfitting.

**For Newton's Method:** For Newton’s Method, choosing the correct number of iterations can be based on the properties of the Hessian matrix. One method is to keep an eye on the size of the elements in the Hessian matrix and stop the iterations when the norm drops below a certain threshold. This usually signifies that the loss function's minimum is getting closer. Another technique involves observing the condition number of the Hessian matrix, a high condition number may signify numerical instability or poor convergence, requiring an adjustment or termination of iterations. These methods help to improve the stability and efficiency of the optimization process and are based on the theoretical foundations of Newton's Method.

10) The provided loss curve comparison for Batch Gradient Descent using updated centers shows that the new data, with centers closer together, converges slower and plateaus at a higher loss compared to the old data. This behavior likely results from the reduced separability in the new dataset configuration, where the proximity of the clusters makes it more challenging for the logistic regression model to effectively distinguish between classes. Consequently, the gradient updates do not decrease the loss as efficiently, resulting in a higher final loss and indicating the impact of data distribution on the convergence and effectiveness of the learning algorithm. This underscores the need for considering data characteristics in model training and possibly adopting more sophisticated models or preprocessing techniques to handle less separable data efficiently.

