Department of Electronics and Telecommunication Engineering

EN3150 Assignment 02

Learning from data and related challenges and classification

Name: S.M.S.M.B.Abeyrathna
Index No: 210005H
Date: 2024/10/02

1. <u>Logistic regression</u>

2)

This line applies label encoding to the species feature in the dataset. Since the species names are strings, they cannot be used directly in logistic regression. To address this, label encoding converts the species names into numerical values, assigning "Adelie" the label 0 and "Chinstrap" the label 1.

3)

It is used to remove specific columns from the dataset, leaving only the relevant features for the machine learning model. The ‘species’ column is dropped because it is the target variable that the model is trying to predict, and it has already been encoded separately in the ‘y’ variable. The ‘island’ column, which indicates the island where the penguins were observed and the ‘sex’ column, which specifies the penguins' gender, are also removed.

4)

Both the island and sex features are categorical variables. Since logistic regression requires numerical input, these variables can't be used directly in the model. Therefore, they can either be excluded or transformed into numerical values using methods like one-hot encoding or label encoding.

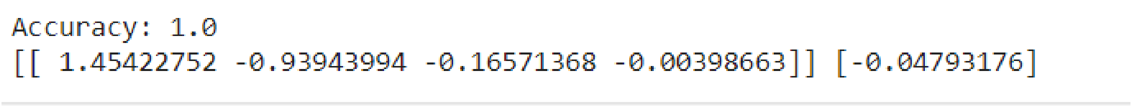
6)

The use of ‘random_state=42’ ensures code reproducibility. It guarantees that the random splitting of the training and testing data remains consistent every time the code is run. This is useful for debugging and for comparing results across different runs and models.

7)

The accuracy achieved was only 0.5814. This poor performance is likely due to the features not being scaled. Applying techniques like standard scaling could improve the results. Another possible reason is class imbalance, as there are 146 Adelie penguins compared to just 68 Chinstrap penguins.

8)



```
Accuracy: 1.0
[[ 1.45422752 -0.93943994 -0.16571368 -0.00398663]] [-0.04793176]
```

9)

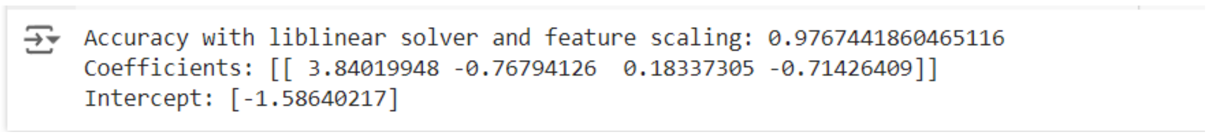
The reason liblinear performs better than saga is likely because the data is not scaled. Liblinear uses a coordinate descent algorithm, which tends to be more robust with unscaled data. It optimises one parameter at a time, allowing it to handle varying feature scales more effectively. On the other hand, SAGA relies on a stochastic gradient descent approach, which is sensitive to feature scaling. Without scaling, the gradients can differ greatly in magnitude, leading to poorer performance and slower convergence.

10)

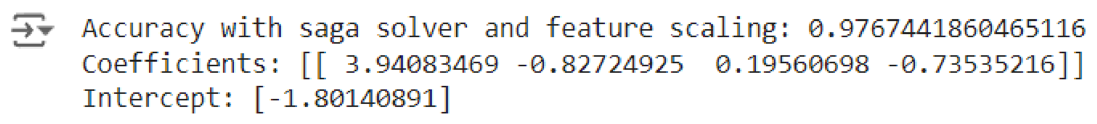
The performance of the liblinear and saga solvers is quite similar when using feature scaling, as both achieved an accuracy of about 0.9767. This means that both solvers are doing a good job in this case, and there isn't a big difference in their performance.

When we use feature scaling, it helps the model learn better because it makes sure all features are on the same scale. This is important for algorithms like logistic regression because it treats all features equally. Without feature scaling, features with larger values could dominate the learning process, making it hard for the model to find the right patterns.

Since both solvers have similar accuracy with feature scaling, it shows that they can learn effectively from the scaled data. This is why feature scaling is often an important step in preparing data for machine learning models.



```
Accuracy with liblinear solver and feature scaling: 0.9767441860465116
Coefficients: [[ 3.84019948 -0.76794126  0.18337305 -0.71426409]]
Intercept: [-1.58640217]
```



```
Accuracy with saga solver and feature scaling: 0.9767441860465116
Coefficients: [[ 3.94083469 -0.82724925  0.19560698 -0.73535216]]
Intercept: [-1.80140891]
```

11)

The issue with the provided code is that categorical data, such as the island and sex features, are being used without first converting them to numerical values. This can be resolved by either dropping these columns, as done previously or by applying techniques like label encoding or one hot encoding to convert them into numerical form.

12)

- Using label encoding for categorical features like 'red', 'blue' and 'green' can lead to issues when applying feature scaling techniques such as Standard Scaling or Min-Max Scaling. Label encoding assigns numeric values to categories (ex: 'red' = 0, 'blue' = 1, 'green' = 2), and scaling these values may cause the model to interpret the numeric differences as meaningful. For example, the model might incorrectly assume that 'green' (2) is "greater" than 'blue' (1) or 'red' (0), which introduces a false ordinal relationship among categories.

- A better approach for categorical data is one hot encoding. One-hot encoding creates separate binary columns for each category, avoiding any implied ranking. For instance, 'red' would be represented as [1, 0, 0], 'blue' as [0, 1, 0] and 'green' as [0, 0, 1]. This ensures that the model treats each category equally. After applying one hot encoding, scaling is generally not required for the one hot encoded features because they already fall within the [0, 1] range. However, numerical features in the dataset can still benefit from scaling.

Question 2:

1)

The logistic regression formula can be used:

$$P(y = 1) = \frac{1}{1 + e^{-(\omega 0 + \omega 1.x1 + \omega 2.x2)}}$$

Substitute the values:

$$P(y = 1) = \frac{1}{1 + e^{-(-5.9 + 0.06\times 50 + 1.5\times 3.6)}} = 0.92414$$

Therefore the estimated probability is 0.92414 (92.414%)

2)

By using logistic regression equation and set the probability *P(y=1)* to 0.60. x_1 (hours studied) can be determined:

$$0.60 = \frac{1}{1 + e^{-(\omega 0 + \omega 1.x1 + \omega 2.x2)}}$$

$$0.60 = \frac{1}{1 + e^{-(-5.9 + 0.06\times x1 + 1.5\times 3.6)}}$$

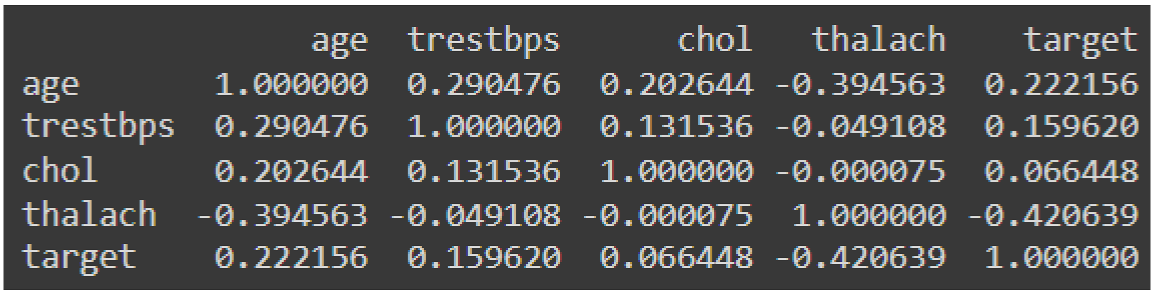
$$x_1 \approx 15.08 \text{ hours}$$

2) Logistic regression on real world data

1)

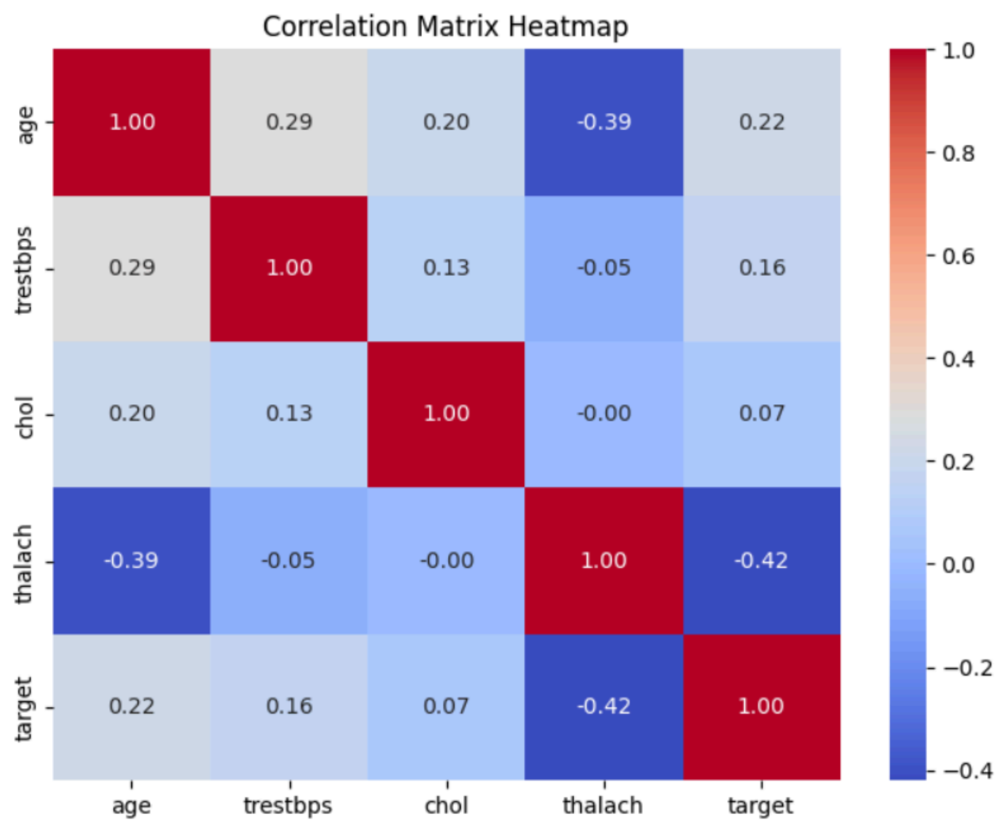
The Cleveland heart disease dataset was chosen. This dataset contains 303 records of patients, with each record describing various factors related to heart health. The dataset includes 14 features, with attributes such as age, sex, chest pain type, resting blood pressure, serum cholesterol, fasting blood sugar, resting electrocardiographic results, maximum heart rate and more. The target feature is a categorical variable indicating the presence or absence of heart disease.

2)



```
               age  trestbps      chol   thalach    target
age       1.000000  0.290476  0.202644 -0.394563  0.222156
trestbps  0.290476  1.000000  0.131536 -0.049108  0.159620
chol      0.202644  0.131536  1.000000 -0.000075  0.066448
thalach  -0.394563 -0.049108 -0.000075  1.000000 -0.420639
target    0.222156  0.159620  0.066448 -0.420639  1.000000
```

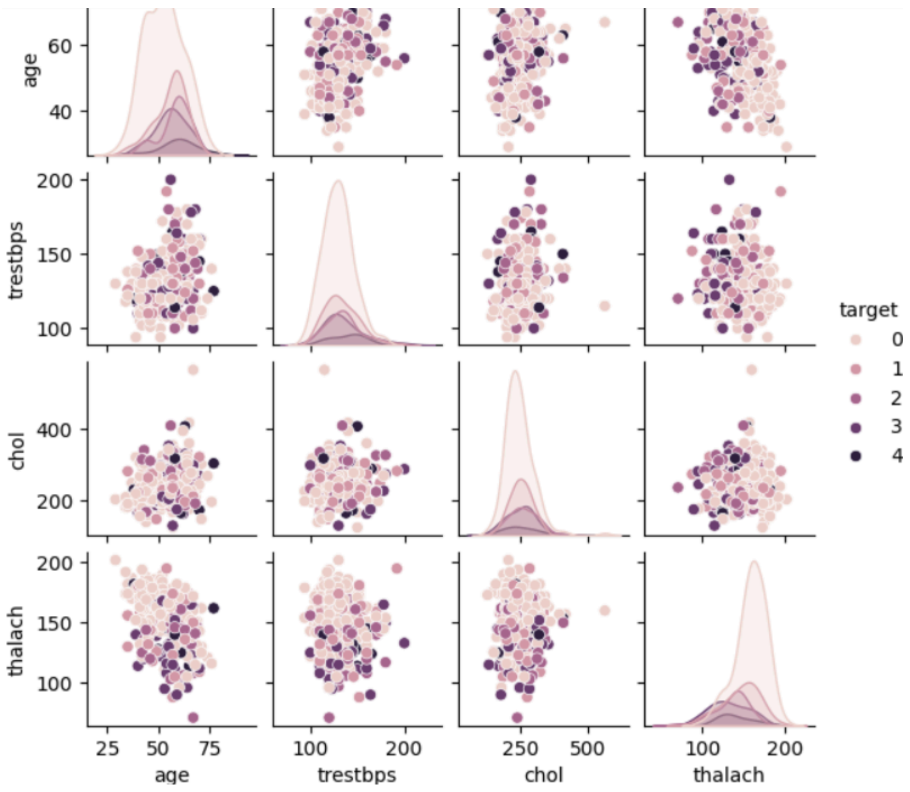
Figure: Correlation matrix



- The correlation matrix reveals several insights into the relationships between the selected features:

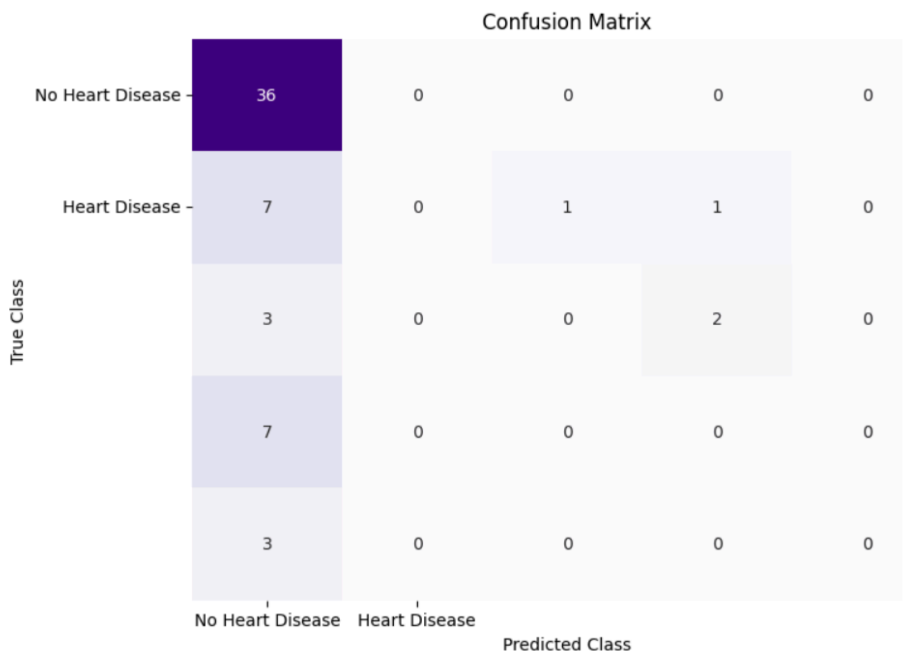
1. Age has a weak positive correlation with both ‘trestbps’ (0.29) and ‘target’ (0.22), indicating that as age increases, there may be a slight rise in blood pressure and heart disease occurrence.
2. ‘Thalach’ (maximum heart rate achieved) shows a moderate negative correlation with target (-0.42), suggesting that higher heart rates are associated with a lower likelihood of heart disease.
3. Other correlations between features, such as ‘chol’ and ‘target’ (0.07), show weak or negligible relationships.

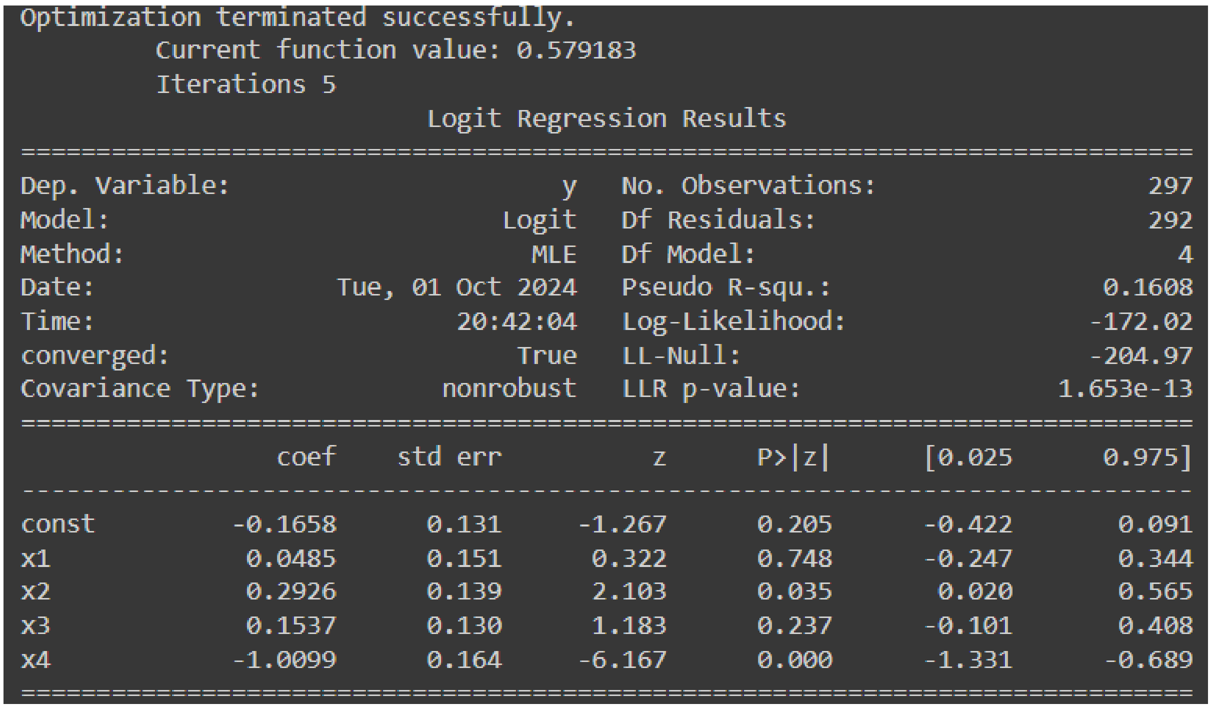
3)



- The achieved test accuracy was 0.6. And also without using a scaler, the accuracy dropped to 0.453 with the saga solver and 0.57 with liblinear. The confusion matrix is presented below.

4)





```
Optimization terminated successfully.
         Current function value: 0.579183
         Iterations 5
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  297
Model:                          Logit   Df Residuals:                      292
Method:                           MLE   Df Model:                            4
Date:                Tue, 01 Oct 2024   Pseudo R-squ.:                  0.1608
Time:                        20:42:04   Log-Likelihood:                -172.02
converged:                       True   LL-Null:                       -204.97
Covariance Type:            nonrobust   LLR p-value:                 1.653e-13
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.1658      0.131     -1.267      0.205      -0.422       0.091
x1             0.0485      0.151      0.322      0.748      -0.247       0.344
x2             0.2926      0.139      2.103      0.035       0.020       0.565
x3             0.1537      0.130      1.183      0.237      -0.101       0.408
x4            -1.0099      0.164     -6.167      0.000      -1.331      -0.689
==============================================================================
```

The model fits the data moderately well, explaining about 16% of the variation in the outcome, but it is statistically significant overall, meaning it provides valuable insights.

Looking at the individual variables, x2 and x4 are the most important. x2 has a positive effect, meaning as x2 increases, the odds of the outcome happening also increase and this effect is statistically significant. On the other hand, x4 has a strong negative effect, meaning it significantly reduces the chances of the outcome. x1 and x3 don't seem to have a significant impact, as their p values are too high to confidently say they affect the outcome. The constant (intercept) is also not statistically significant, meaning the model's baseline prediction doesn't differ much from zero when all variables are set to zero.

3) <u>Logistic regression First/Second-Order Methods</u>

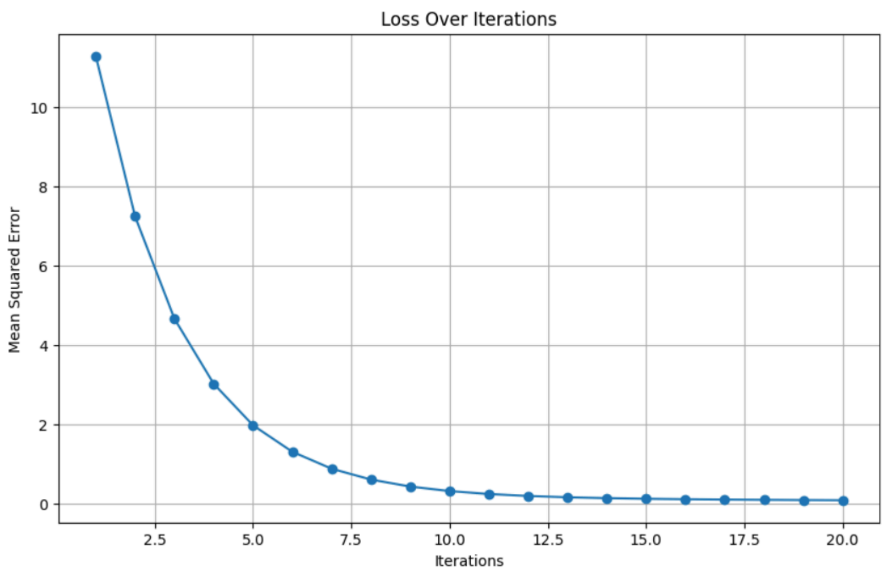
2)

If all weights are initialised to the same value (such as zero), each feature will initially contribute equally to the prediction. This symmetry can hinder effective learning, as the gradients for all weights will be identical, resulting in uniform updates. Randomly initialising weights ensures they start with different values, enabling the model to capture diverse features and break this symmetry. This approach allows for more effective exploration of the solution space and improves the model's ability to learn from the data.

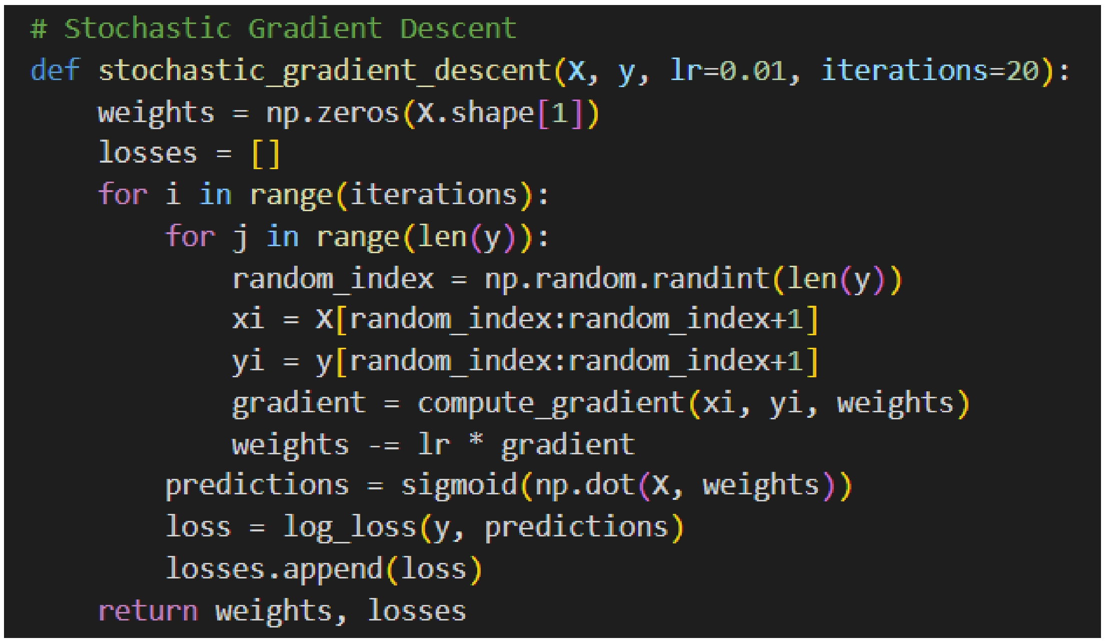
3)

The mean squared error loss function was employed due to its simplicity in computation, differentiability, and tendency to penalise larger residuals. These characteristics are desirable for an effective loss function, as they facilitate optimization and improve the model's ability to learn from data.

4)



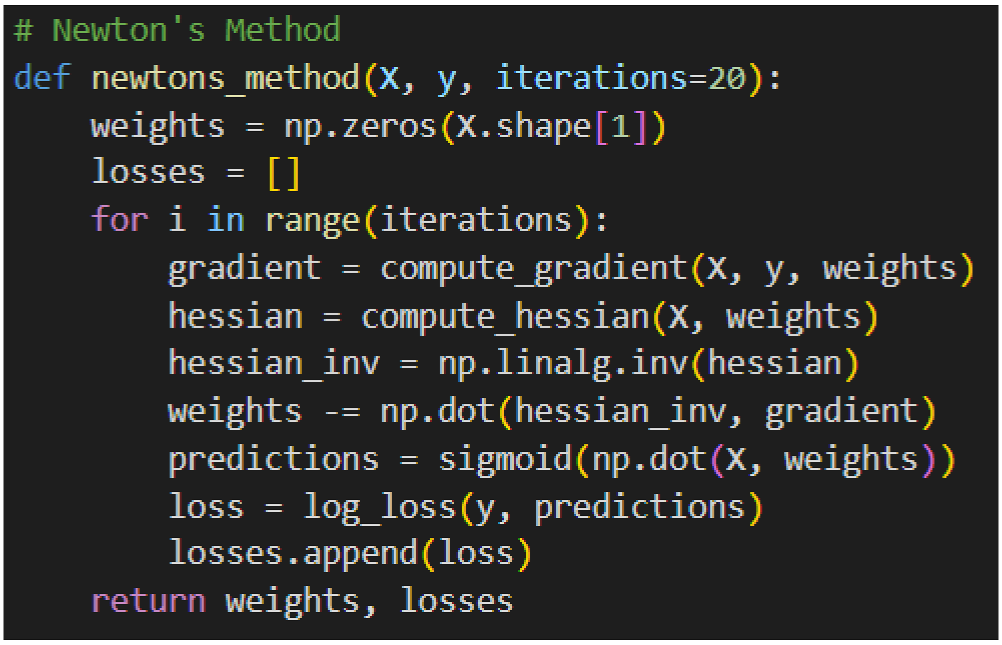
5)



```
# Stochastic Gradient Descent
def stochastic_gradient_descent(X, y, lr=0.01, iterations=20):
    weights = np.zeros(X.shape[1])
    losses = []
    for i in range(iterations):
        for j in range(len(y)):
            random_index = np.random.randint(len(y))
            xi = X[random_index:random_index+1]
            yi = y[random_index:random_index+1]
            gradient = compute_gradient(xi, yi, weights)
            weights -= lr * gradient
        predictions = sigmoid(np.dot(X, weights))
        loss = log_loss(y, predictions)
        losses.append(loss)
    return weights, losses
```

Figure: SGD implementation

6)

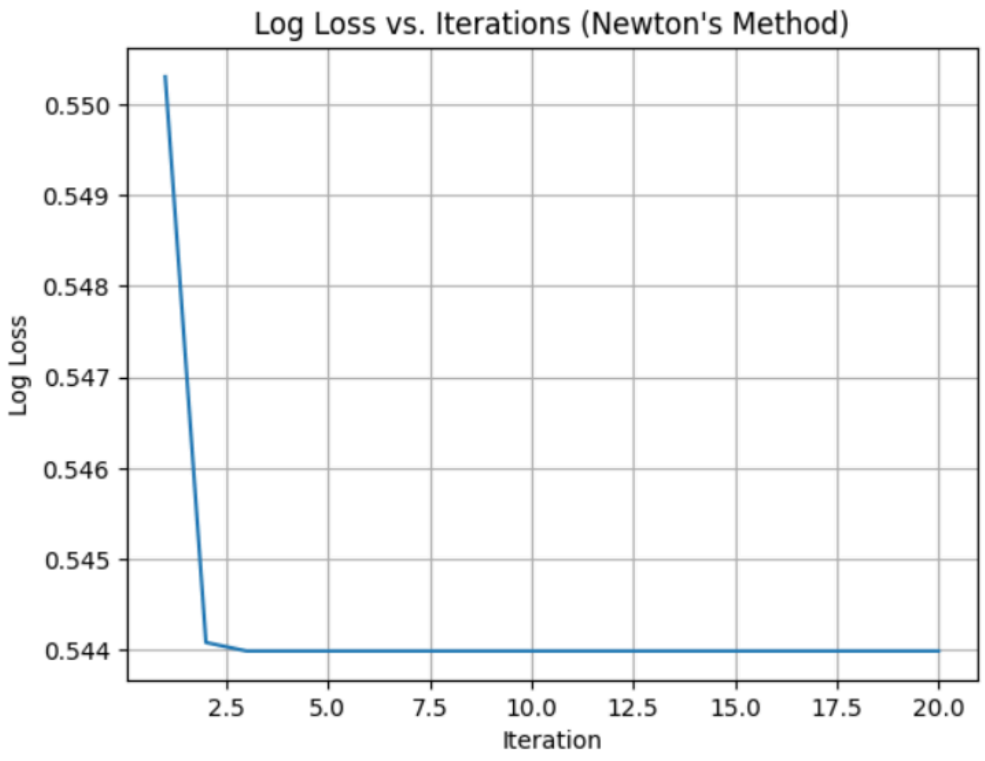


```python
# Newton's Method
def newtons_method(X, y, iterations=20):
    weights = np.zeros(X.shape[1])
    losses = []
    for i in range(iterations):
        gradient = compute_gradient(X, y, weights)
        hessian = compute_hessian(X, weights)
        hessian_inv = np.linalg.inv(hessian)
        weights -= np.dot(hessian_inv, gradient)
        predictions = sigmoid(np.dot(X, weights))
        loss = log_loss(y, predictions)
        losses.append(loss)
    return weights, losses
```

Figure: Newton's method implementation

7)

The loss function was plotted and the output as shown in below figure was obtained.



8)

- Gradient Descent: Loss decreases steadily over iterations. It usually converges more slowly than Newton's method but is more stable.

- Stochastic Gradient Descent: Loss tends to fluctuate more due to random sampling but can converge faster due to updates after each sample.

- Newton's Method: Tends to converge much faster, as it uses both the gradient and the Hessian matrix to make more accurate updates, especially near the minimum.

9)
Convergence-Based Approach (Loss Threshold):

- Gradient Descent: Stop the algorithm when the change in the loss function between iterations is below a certain threshold. This approach ensures that the algorithm stops when the model has converged close to the optimal solution.
- Newton's Method: Similar to gradient descent, you can stop the iterations when the change in loss or the norm of the gradient falls below a small threshold. Since Newton's Method typically converges faster, this threshold can be set a little higher compared to gradient descent.

Cross-Validation Approach:

- Gradient Descent: Use cross-validation to choose the number of iterations by evaluating the model's performance on a validation set after a fixed number of iterations. The number of iterations that gives the best performance on the validation set is chosen.
- Newton's Method: Similarly, evaluate the performance of the model using a validation set after each iteration and choose the number of iterations based on when the validation performance peaks.

10)

Convergence Behavior Analysis:

1. Closer Centers

- With centres [[3, 0], [5, 1.5]], the two classes are much closer to each other in the feature space compared to the previous configuration.
- **Impact on Convergence**: Since the classes are closer together, it may take more iterations for Gradient Descent to converge, as the decision boundary will be harder to distinguish. The gradient updates will be smaller, leading to slower progress towards the optimal weights.

2. Loss Function Behaviour:

- The loss function may decrease more gradually because the model will struggle to find a linear boundary that separates the classes well. The overlap between the classes introduces more uncertainty, so the model may plateau at a higher loss before convergence.
- **Expected Behaviour**: You might see a slower decline in the loss over iterations compared to when the centres were farther apart.

3. Gradient Magnitude:

- The gradient's magnitude will also be smaller due to the close proximity of the data points belonging to different classes. As a result, the weight updates will be smaller, leading to slower convergence.

4. Effect of Learning Rate:

- With the new centres, a lower learning rate might be required to avoid overshooting due to the subtle gradients. However, this would also slow down convergence further. A balance needs to be struck between learning rate and the number of iterations.

Explanation for Convergence Behavior:

- **Class Overlap**: Since the centres are close together, there will be more overlap between the classes, making the decision boundary harder to find. This results in slower convergence because Gradient Descent will need to perform finer adjustments to the weights to try and find a good boundary.
- **Slower Learning**: As the decision boundary becomes less clear, the gradient updates become smaller, making it harder for the algorithm to make large steps towards the optimal solution. This causes the loss to decrease at a slower rate, requiring more iterations for convergence.

With the updated centres, Batch Gradient Descent will converge more slowly because the decision boundary is harder to determine due to the proximity of the two class centres. The algorithm will still converge, but more iterations or a fine tuned learning rate might be needed for effective convergence.