Department of Electronic & Telecommunication Engineering
University of Moratuwa

EN3150
Pattern Recognition

Assignment 02

27/09/2024

210642G - Thennakoon T.M.K.R.

1. Logistic regression

Question 1

2.
The purpose of y_encoded = le.fit_transform(df_filtered['species']) is to encode the categorical variable species into numerical values. This step converts the species labels into numerical form. In this case, the LabelEncoder assigns 0 to ‘Adelie’ and 1 to ‘Chinstrap’.

3.
The purpose of X = df.drop([’species’, ’island’, ’sex’, ’class_encoded’], axis=1) is to remove the columns named ’species’, ’island’, ’sex’, ’class_encoded’ which are not relevant for model training. Specially species is the dependent variable and it is not needed in independent set.

4.
Since ‘island’ and ‘sex’ features are not numerical, we can not use them directly, since machine learning models like logistic regression are work with numerical data.

6.
I the train_test_split function, the dataset is split into training and testing sets randomly. By setting random_state=42, we are fixing the seed for the random number generator so that every time the code is run, the same random split is produced.

7.
The saga solver is sensitive to the scaling of features. The features in the dataset are not properly scaled, it could affect the convergence of the solver, resulting in poor performance. And the saga solver is an optimization algorithm designed for large-scale problems and is particularly useful when the dataset is large.

8.
Accuracy : 1.0

9.
liblinear is better at handling unscaled dataset and smaller, well-behaved datasets with a few features and provides stable convergence, which can result in higher accuracy for dataset like ‘penguin’ than saga which is more suitable for large datasets.

10.

	Without feature scaling	**With feature scaling**
saga	0.5813953488372093	0.9767441860465116
liblinear	1.0	0.9767441860465116

Accuracy with the saga solver improved significantly with feature scaling, while the liblinear solver's accuracy changed more slightly.
This is because, Solvers like saga and algorithms using gradient-based optimization are sensitive to the magnitude of features. Scaling brings all features to the same range, allowing the solver to converge faster and more accurately.

11.
The issue with the code is that some features in the dataset such as island, sex contain categorical values, and logistic regression expects numerical values. Therefore, attempting to use categorical features without converting them to numeric form will result in an error.

We can encode the categorical columns using one-hot encoding or label encoding to convert them into numeric values before training the model.

12.
Label Encoding converts categorical values into integers (e.g., red -> 0, blue -> 1, green -> 2). Then after scaling them using Standard or Min-Max Scaler will introduce an ordinal relationship between the categories, implying a ranking or order between them, which is not meaningful for nominal categories. For example, the model may infer that green (encoded as 2) is "greater" or "more important" than blue (encoded as 1), which is incorrect for categories like colors, where no inherent order exists. The recommended approach is to use One-Hot Encoding instead of label encoding for categorical features that have no intrinsic order.

Question 2

1.

Assume that receiving A+ class is yi = 1.

P(yi=1 | w,x) = sigm(w0 + w1*x1 + w2*x2)
= sigm(-5.9 + 0.06*x1 + 1.5*x2)
= sigm(-5.9 + 0.06*50 + 1.5*3.6)
= sigm(-2.5)
= 0.924

Probability that a student who has studied for 50 hours and has an undergraduate GPA of 3.6 will receive an A + in the class is 0.924

2.

P(yi=1 | w,x) = sigm(w0 + w1*x1 + w2*x2)
0.6 = sigm(-5.9 + 0.06*x1 + 1.5*3.6)
0.6 = sigm(0.06*x1 - 0.5)

$$0.6 = \frac{1}{1 + e^{-(0.06*x1 - 0.5)}}$$

x1 = 15.09

To achieve a 60% chance of receiving an A + in the class, a student like the one in part 1 need to study 15.09 hours.

2. Logistic Regression on Real World Data

1.

Choosed heart diseas as the dataset

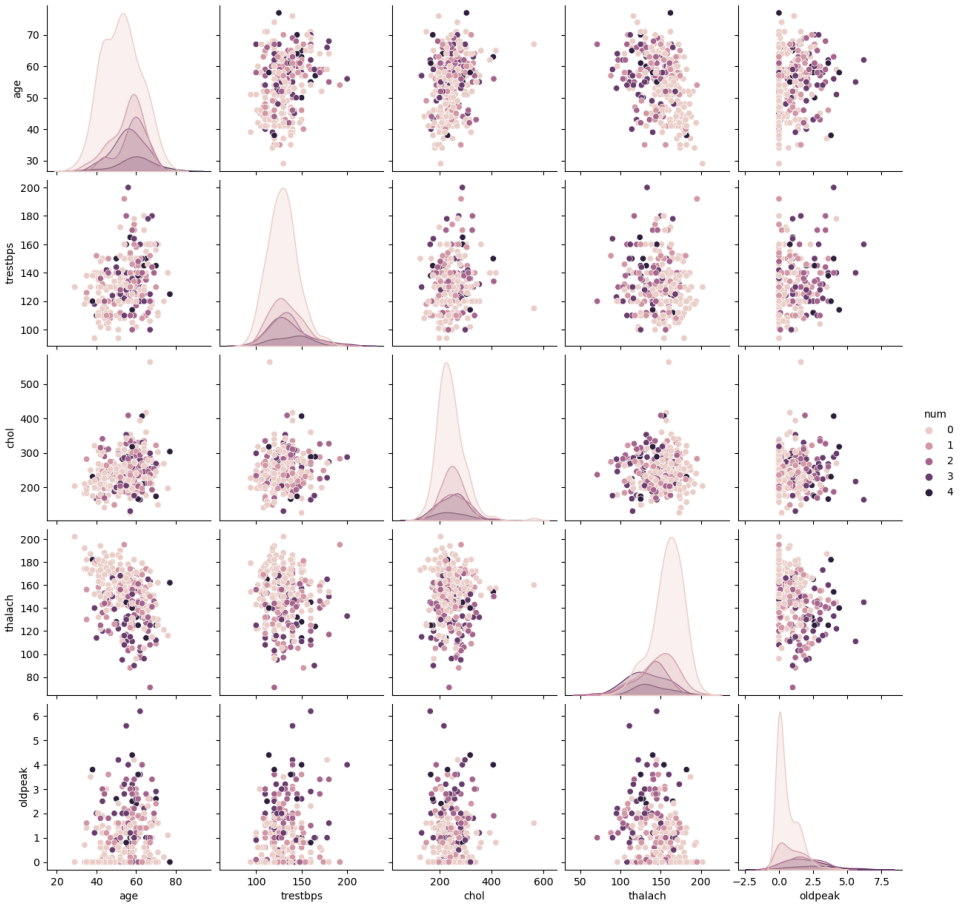
2.

selected_features = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']

Correlation Matrix

	age	trestbps	chol	thalach	oldpeak	num
age	1.000000	0.284946	0.208950	-0.393806	0.203805	0.222853
trestbps	0.284946	1.000000	0.130120	-0.045351	0.189171	0.157754
chol	0.208950	0.130120	1.000000	-0.003432	0.046564	0.070909
thalach	-0.393806	-0.045351	-0.003432	1.000000	-0.343085	-0.415040
oldpeak	0.203805	0.189171	0.046564	-0.343085	1.000000	0.504092
num	0.222853	0.157754	0.070909	-0.415040	0.504092	1.000000

Pair Plots



In the correlation matrix, almost all the correlation values are below 0.5 meaning there is very small correlation between features.

3.

Accuracy: 0.492 - This indicates that the model correctly predicted 49.2% of the total instances. An accuracy of less than 50% suggests that the model is performing poorly overall.

Precision: 0.322 - This means that when the model predicts a positive class, it is correct only 32.2% of the time. This low precision indicates a high number of false positives.

Recall: 0.492 - The model identifies 49.2% of the actual positive cases. A recall of less than 50% means the model is missing many true positives.

F1-Score: 0.385 - The F1-score is the harmonic mean of precision and recall, and it reflects a balance between the two. A score of 0.385 suggests that the model is not effectively balancing precision and recall.

Confusion Matrix:

```
[[28  0  1  0  0]
 [ 9  0  1  2  0]
 [ 4  0  1  4  0]
 [ 4  0  2  1  0]
 [ 2  0  1  1  0]]
```

The model performs best for Class 0 with a relatively high accuracy, while the accuracy for other classes is quite low. The model struggles particularly with Class 1, Class 2, Class 3, and Class 4, indicating a significant issue with either the training data or model architecture. Given the low precision and recall, the model is not reliably distinguishing between classes, leading to many false predictions.

4.

Since this is a multiclass classification, sm.MNLogit() model was used.

I. In the first model, oldpeak is the only predictor with a significant p-value (< 0.05), and thalach is marginally significant. Other predictors (age, trestbps, chol) could be considered for removal.

II. In the second model, both thalach and oldpeak have significant p-values and should be retained, while age, trestbps, and chol have high p-values and could be discarded

III. In the third model also, both thalach and oldpeak have significant p-values and should be retained, while age, trestbps, and chol have high p-values and could be discarded

IV. In the fourth model, oldpeak is the only predictor with a significant p-value (< 0.05). Other predictors (age, trestbps, chol, thalac) could be considered for removal.

num=1	coef	std err	z	P>|z|	[0.025	0.975]
const	-1.9045	2.383	-0.799	0.424	-6.575	2.766
age	0.0166	0.024	0.707	0.480	-0.029	0.063
trestbps	0.0072	0.011	0.663	0.507	-0.014	0.028
chol	0.0059	0.004	1.489	0.137	-0.002	0.014
thalach	-0.0188	0.010	-1.945	0.052	-0.038	0.000
oldpeak	0.3951	0.196	2.014	0.044	0.011	0.780

num=2	coef	std err	z	P>|z|	[0.025	0.975]
const	-2.5510	2.991	-0.853	0.394	-8.413	3.311
age	0.0378	0.031	1.233	0.218	-0.022	0.098
trestbps	0.0018	0.014	0.126	0.900	-0.026	0.029
chol	0.0089	0.005	1.883	0.060	-0.000	0.018
thalach	-0.0319	0.012	-2.688	0.007	-0.055	-0.009
oldpeak	0.9498	0.217	4.371	0.000	0.524	1.376

num=3	coef	std err	z	P>|z|	[0.025	0.975]
const	-0.4959	2.884	-0.172	0.864	-6.149	5.158
age	-0.0155	0.031	-0.507	0.612	-0.075	0.044
trestbps	0.0227	0.014	1.605	0.109	-0.005	0.050
chol	0.0063	0.005	1.272	0.203	-0.003	0.016
thalach	-0.0416	0.012	-3.518	0.000	-0.065	-0.018
oldpeak	1.0498	0.214	4.906	0.000	0.630	1.469

num=4	coef	std err	z	P>|z|	[0.025	0.975]
const	-8.9717	5.181	-1.732	0.083	-19.127	1.183
age	0.0616	0.053	1.158	0.247	-0.043	0.166
trestbps	0.0217	0.020	1.060	0.289	-0.018	0.062
chol	0.0058	0.007	0.821	0.411	-0.008	0.020
thalach	-0.0208	0.020	-1.060	0.289	-0.059	0.018
oldpeak	1.1453	0.295	3.878	0.000	0.566	1.724

3. Logistic regression First/Second-Order Methods

2.

We will initialize the weights randomly using a small random value since small values prevent exploding gradients. Used weights = np.random.randn(X.shape[1]) as the code to generate random values for weights.

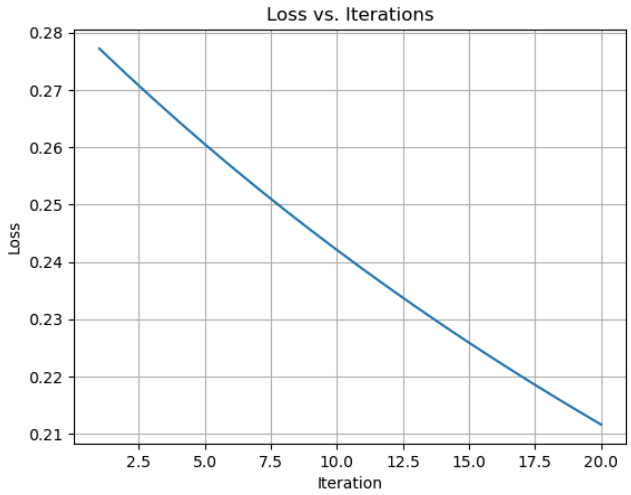
3.

Since this is a binary classification problem, I used **cross-entropy error function** as the loss function.

-np.mean(y_true * np.log(y_pred) + (1 - y_true) * np.log(1 - y_pred))

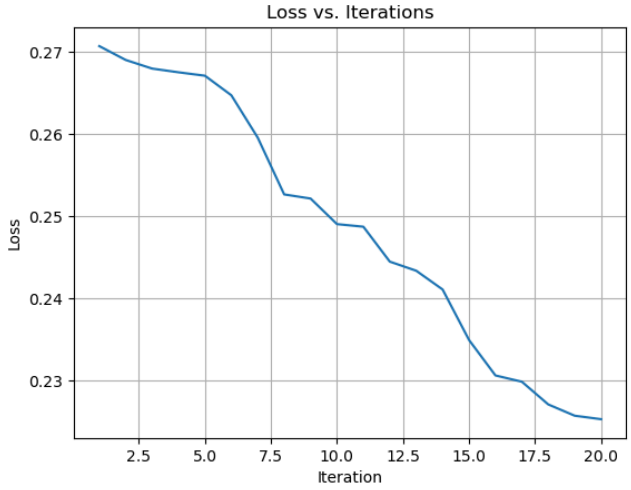
4.

Loss vs iteration plot for batch gradient descent



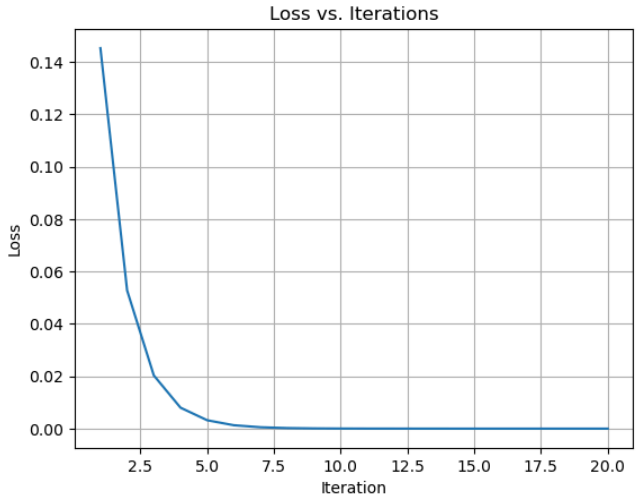
5.

Loss vs iteration plot for stochastic gradient descent



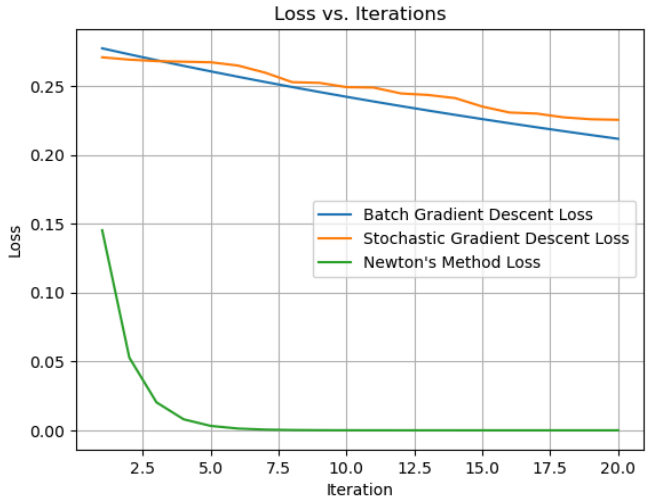
7.

Loss vs iteration plot for newton's method



8.

Comparing plot of newton's, batch and stochastic gradient descent



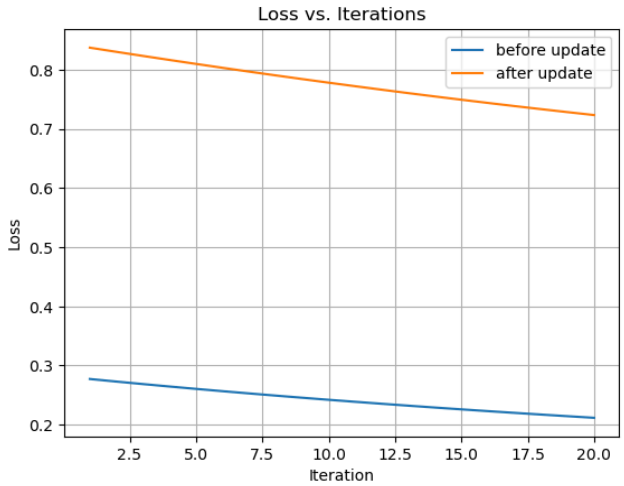
Newton's method is more accurate than batch gradient descent and stochastic gradient descent due to its loss values for 20 iterations. We can see that from the beginning, newton method gives a small loss value compared to others and converge to zero in low number of iterations compared to others.

9.

Gradient Descent - Set a convergence threshold based on the change in loss or weights. Monitor the loss value or the weights at each iteration, and stop the algorithm when the change between consecutive iterations is less than a specified threshold.

Newton's Method - Set a maximum number of iterations and a tolerance level. Terminate the process when either the maximum number of iterations is reached or when the change in the parameter estimates is less than a specified tolerance.

10.



The convergence of loss function to zero is faster before updating than after updating centers. Above plot shows that after updating has a higher loss value than before for corresponding iteration. This is because when we updated to the centers to [[3, 0], [5, 1.5]], datasets of two classes get closer and intersect with each other as shown in below images. Due to this, model become weaker in differentiating two classes.