

Name: Ammu Ambika Ramachandran

Student ID: 23006390

Effect of Regularization on Logistic Regression Performance

1. Introduction to Logistic Regression

Often employed in machine learning, logistic regression is a somewhat simple and efficient classification method (Nick & Campbell, 2007). In high-dimensional datasets, it may, however, become prone to overfitting—where the model learns noise rather than significant patterns. Overfitting lowers generalisation, which results in worse performance on unprocessed data. Commonly used regularising methods are meant to help with this (Sperandei, 2014). By adding penalty terms to the loss function and hence preventing coefficients from becoming too big, regularisation helps regulate the complexity of the model. In logistic regression, two main forms of regularity used are L1 (Lasso) and L2 (Ridge). L1 regularisation is helpful for feature selection as it drives certain feature coefficients to become precisely zero, hence promoting sparsity. Conversely, L2 regularisation stabilises the model by lowering coefficient magnitudes without deleting features (Qin & Lou, 2019).

Choosing the right regularization technique depends on the dataset and problem at hand. Properly tuned regularization parameters help improve model performance by striking a balance between bias and variance. This report will demonstrate the impact of regularization on logistic regression, highlighting how it enhances generalization and prevents overfitting, ensuring better predictive accuracy on new data.

2. Understanding Regularization

In machine learning, regularisation is a fundamental method used to impose limits on model parameters hence preventing overfitting. Regularity in logistic regression guarantees that the model does not learn too complicated patterns from the training data, therefore preventing possible poor generalisation on unseen data. This is accomplished by including penalty terms into the loss function, hence regulating the model coefficient magnitude. Regularity therefore

helps to avoid the coefficients from being too big, hence strengthening the model and lowering variance. In logistic regression, two often used forms of regularity are L1 regularisation (Lasso) and L2 regularisation (Ridge) (Xu et al., 2012). Each one of these approaches has varied effects on the model and finds use in various situations.

2.1. L1 Regularization (Lasso)

L1 regularization—also called Lasso (Least Absolute Shrinkage and Selection Operator)—adds a penalty term to the loss function commensurate with the absolute values of the model parameters. Mathematically, this penalty is represented as $||w||_1$, the total of the absolute values of every coefficient in the model. L1 regularisation has one of the most important features: sparsity, which drives certain coefficients to become precisely zero. L1 regularisation is thus very helpful for feature selection as it automatically eliminates pointless or less significant elements from the model. Especially helpful when dealing with high-dimensional datasets, pushing certain coefficients to zero simplifies the model and improves interpretability (Xu et al., 2012).

L1 regularisation, for instance, may assist find the most essential predictors in a dataset containing hundreds of characteristics many of which could not substantially affect the classification job by removing the less relevant ones. This produces a more effective model with less calculations needed while still having great predictive capability. L1 regularisation has one drawback, however, in that it may cause instability in feature selection in cases with many connected features. Since it chooses just a fraction of the available characteristics, the choice of chosen variables may vary depending on little changes in the data (Mazilu & Iria, 2011).

2.2. L2 Regularization (Ridge)

Often called Ridge regression, L2 regularisation adds a penalty term commensurate with the square of the coefficients. This is stated mathematically as $||w||_2^2$, the total of the squared values of every model coefficient. L2 regularisation does not compel any coefficients to become precisely zero like L1 regularisation does. Rather, it keeps all characteristics in the model while shrinking all coefficients towards zero (Zou et al., 2015).

L2 regularisation mostly helps avoid multicollinearity—that is, the result of strongly connected independent variables. Reducing the size of coefficients helps Ridge regularisation to

distribute the significance of characteristics more fairly and stabilise the model, hence enhancing generalisation on fresh data. L2 regularisation is thus a useful option when working with datasets with numerous linked predictors as it prevents any one feature from controlling the model. Although L2 regularisation increases generalisation, it does not provide automated feature selection like L1 regularisation does. Though their impact is less in line with the penalty used, all traits remain in the model. In complicated datasets where every variable adds important information, this may help (Xu et al., 2012).

3. The Role of the Hyperparameter (C) in Regularization

In logistic regression, the hyperparameter C controls both L1 and L2 regularisation methods. With $C = 1/\lambda$, the parameter C is the inverse of the regularising strength. A greater value of C indicates less regularisation, which lets the model coefficients take more values, therefore maybe causing overfitting. On the other hand, a smaller value of C promotes regularity, hence lowering the coefficient values and producing a simpler model less prone to overfit (Zou et al., 2015).

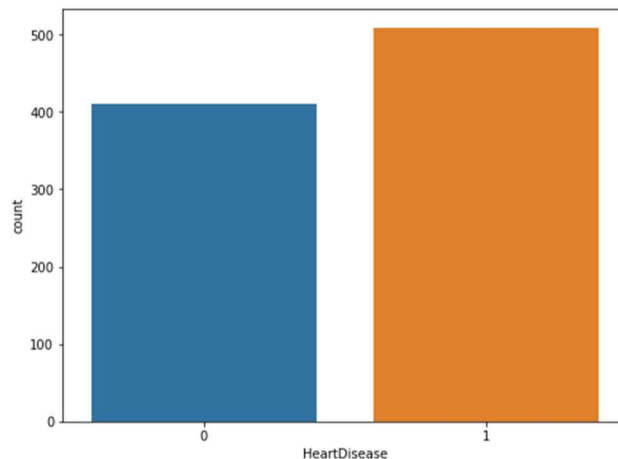
A correct balance between bias and variance depends on selecting a value for C that fits. Underfitting—where the model becomes too basic and misses significant data patterns—may result from a very low C value. Conversely, a very high C value could cause overfitting—that is, when the model memorises the training data and performs badly on fresh inputs. The best value of C for a particular dataset may be obtained by use of hyperparameter tuning methods like cross-valuation. Practitioners may choose the optimal regularising strength to improve model generalisation by assessing many C values and performance on validation data.

4. Implementation of Logistic Regression with Regularization

The effect of regularisation on logistic regression performance is analysed in this study using the Heart Failure Prediction Dataset. Two forms of regularization—L1 (Lasso) and L2 (Ridge)—are used to the logistic regression model; regularising aids in reducing overfitting by penalising huge coefficient values.

Data Pre-processing: First loaded using the pandas package, the dataset is then examined first using `df.head()`. This helps us to grasp the character of the characteristics and examine the dataset's structure. The `df.isnull()` helps one to guarantee a clean dataset.`total()`.Missing

values are search for using any() function. This phase guarantees that there are no null values influencing the study as missing data might cause disparities in model training. A count plot then created using seaborn to show the desired variable (HeartDisease) distribution The number of positive and negative instances in the dataset is therefore verified using the value counts() function of the df['HeartDisease']. This guides further preprocessing choices and clarifies the balance of the dataset.



For logistic regression, the dataset's categorical elements must be transformed into numerical form as they represent Categorical columns including Sex, Chest Pain Type, Resting ECG, Exercise Angina, and ST_Slope are converted from sklearn.preprocessing's LabelEncoder into numerical forms. This guarantees the model's successful processing of categorical data.

```
le = LabelEncoder()
df['Sex'] = le.fit_transform(df['Sex'])
df['ChestPainType'] = le.fit_transform(df['ChestPainType'])
df['RestingECG'] = le.fit_transform(df['RestingECG'])
df['ExerciseAngina'] = le.fit_transform(df['ExerciseAngina'])
df['ST_Slope'] = le.fit_transform(df['ST_Slope'])
```

After encoding is finished, all categorical characteristics are checked to have been transformed into numerical form using df.dtypes. The dataset is then separated into features (x) and target variable (y), wherein y stands for the dependent variable and x comprises all independent factors except HeartDisease.

Splitting the Data Set: Train and test sets created via train_test_split() from sklearn.model_selection help to train and assess the model. Twenty percent of the data is set aside for testing; eighty percent is utilised for training here. This split guarantees that, even leaving some unseen for assessment, the model gets trained on most of the data.

```
x = df.drop(columns = 'HeartDisease', axis = 1)
y = df['HeartDisease']
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
```

Features Scaling: StandardScaler's feature scaling is done prior to logistic regression application. Scaling is essential as it guarantees that no feature dominates the learning process by standardising the range of independent variables and therefore guaranteeing greater numerical values. Fit_transform() turns the testing and training sets (x_test) and x_train respectively.

```
scaler = StandardScaler(copy=True, with_mean=False, with_std=True)
x_trains = scaler.fit_transform(x_train)
x_tests = scaler.fit_transform(x_test)
```

4.1. Logistic Regression with L1 Regularization

Different logistic regression models are trained with varying C parameter values in order to assess the impact of L1 regularisation (Lasso). Since the C parameter is the inverse of the regularisation strength, lower values apply more robust regularisation. Trained with values of C ranging from 0.0001 to 100, the model is

```
c = [0.0001, 0.001, 0.01, 0.1, 1, 10, 100]

for i in c:
    lr = LogisticRegression(C=i, penalty='l1', solver='liblinear', n_jobs=-1).fit(x_trains, y_train)
    y_pred = lr.predict(x_trains)
    acc = accuracy_score(y_train, y_pred)
    print('L1 Reg - train accuracy score for C =', i, 'is:', (acc*100), '%')
```

Predictions on the training set follow training; accuracy is determined using accuracy_score() after training. The last model is trained using the found optimum C value and tested on the test set after it has been decided upon. To evaluate the prediction ability of the model, confusion matrix and classification report are produced.

4.2. Logistic Regression with L2 Regularization

The same procedure is carried out for L2 regularisation (Ridge). The only variation is when training the model specifies penalty="l2". Once again, the C values vary to examine how regularising strength affects model performance.

```
c = [0.0001, 0.001, 0.01, 0.1, 1, 10, 100]

for i in c:
    lr = LogisticRegression(C=i, penalty='l2', solver='liblinear', n_jobs=-1).fit(x_trains, y_train)
    y_pred = lr.predict(x_trains)
    acc = accuracy_score(y_train, y_pred)
    print('L2 Reg - train accuracy score for C =', i, 'is:', (acc*100), '%')
```

Following different models training, test set predictions are produced and a confusion matrix and classification report is produced to assess the efficacy of the Ridge logistic regression model.

5. Critical Analysis of Results

Regularising methods' effects on model performance are shown using logistic regression with L1 and L2 regularisation. With an ideal test accuracy of 89.13%, both techniques were successful in avoiding overfit while preserving good classification performance. Still, there are minor variations in training accuracy and the general model qualities that call for further discussion.

5.1. Comparison of L1 and L2 Regularization

Training Accuracy

- L1 Regularization (Lasso): The model achieved a training accuracy of 85.28% for $C = 1$, meaning it effectively learned from the training data without excessive overfitting.
- L2 Regularization (Ridge): The model's training accuracy was slightly lower at 85.01% for $C = 1$, indicating that L2 regularization applies a stronger penalty to large coefficient values, thereby reducing model complexity slightly more than L1 regularization.

Despite these differences, both models generalize well to the test set, as reflected in their identical test accuracy of 89.13%.

Test Accuracy and Generalization

- The test accuracy being the same for both L1 and L2 regularization suggests that both models handle unseen data similarly well.
- This stability in test accuracy shows that neither model is overfitting significantly, as overfitting would have led to much higher training accuracy but lower test accuracy.
- It also indicates that logistic regression is performing well on this dataset, even without needing complex deep-learning models.

5.2. Precision, Recall, and F1-Score Analysis

Performance on Class 0 (No Heart Disease)

- Precision: The L1 and L2 models both have a precision of 0.86, meaning that 86% of the patients predicted as not having heart disease were correctly classified.
- Recall: The recall score of 0.88 means that 88% of actual non-heart disease patients were correctly identified.
- F1-score: At 0.87, this balanced metric confirms that the model performs well in distinguishing negative cases.

	precision	recall	f1-score	support
0	0.86	0.88	0.87	77
1	0.91	0.90	0.91	107
accuracy			0.89	184
macro avg	0.89	0.89	0.89	184
weighted avg	0.89	0.89	0.89	184

Performance on Class 1 (Heart Disease Present)

- Precision: Both models have a 0.91 precision score, meaning that 91% of patients predicted to have heart disease were correctly classified.
- Recall: With a 0.90 recall score, the models correctly identified 90% of patients who actually had heart disease.
- F1-score: The 0.91 F1-score suggests strong classification performance for heart disease cases.

By means of their prevention of overfitting and enhancement of generalisation, the findings show that both L1 and L2 regularisation improve logistic performance. L1 drives several coefficients to zero to typically provide a sparser model; L2 guarantees better weight distribution. The almost perfect result points to a well-structured dataset; regularisation helps maximise model resilience without appreciable variations in predicting ability.

References

- Mazilu, S. and Iria, J. (2011) 'L1 vs. L2 regularization in text classification when learning from labeled features', 2011 10th International Conference on Machine Learning and Applications and Workshops, pp. 166–171. doi:10.1109/icmla.2011.85.
- Nick, T.G. and Campbell, K.M. (2007) 'Logistic regression', *Methods in Molecular Biology*TM, pp. 273–301. doi:10.1007/978-1-59745-530-5_14.
- Qin, J. and Lou, Y. (2019) 'L1-2 regularized logistic regression', 2019 53rd Asilomar Conference on Signals, Systems, and Computers, pp. 779–783. doi:10.1109/IEEECONF44664.2019.9048830.
- Sperandei, S. (2014) 'Understanding logistic regression analysis', *Biochemia Medica*, pp. 12–18. doi:10.11613/bm.2014.003.
- Xu, C., Peng, Z. and Jing, W. (2012) 'Sparse kernel logistic regression based on L 1/2 regularization', *Science China Information Sciences*, 56(4), pp. 1–16. doi:10.1007/s11432-012-4679-3.
- Zou, F. et al. (2015) 'Supervised feature learning via L2-norm regularized logistic regression for 3D object recognition', *Neurocomputing*, 151, pp. 603–611. doi:10.1016/j.neucom.2014.06.089.