A2. Regression

## Logistic Regression for the Diabetes Dataset

When selecting a machine learning algorithm for a particular dataset, it's crucial to consider the nature of the data, the problem at hand, and the strengths of various algorithms.

1. Nature of the Problem

The primary task with the diabetes dataset is binary classification: determining whether an individual has diabetes (1) or does not have diabetes (0). Logistic regression is specifically designed for binary classification tasks. It directly estimates the probability of the default class, making it inherently suitable for this kind of problem.

2. Interpretability

Logistic regression is one of the most interpretable machine learning models. The model provides coefficients for each feature, which can be directly interpreted in terms of odds ratios. For instance, you can understand how a unit change in glucose level affects the odds of having diabetes. This interpretability is crucial in medical applications where understanding the influence of different factors is important for both practitioners and patients.

3. Performance

Logistic regression performs well when the relationship between the features and the outcome is approximately linear. Given the nature of the diabetes dataset, which includes features such as glucose level, BMI, and age, there is reason to believe that the relationship between these features and the likelihood of diabetes could be approximated linearly.

In the given results:

The training accuracy was 77.65%, and the test accuracy was 78.79%.

These are respectable performance metrics, indicating that the logistic regression model has learned the underlying patterns in the training data and generalizes reasonably well to the test data.

## Training the Logistic Regression Model

Logistic Regression is a statistical model used for binary classification tasks. It estimates the probability that a given input belongs to a certain class. It’s commonly used in scenarios where the outcome is binary, such as predicting whether a person has a disease (yes/no).

First, I split the features (X) and targets (Y) of the dataset into training and testing data, choosing a ratio of 70% for training and 30% for testing.

I trained the Logistic Regression Model by importing the model from scikit-learn and used the training sets, both for the features (X) and target (Y) to fit the model.

### Regression Coefficients

* Intercept: The intercept is a constant term in the regression equation. It represents the log-odds of the outcome when all predictors are zero.
* Coefficients: These are weights assigned to each feature in the model, indicating how much each feature contributes to the outcome.

For the training data upon fitting the model, I received the following intercept and regression coefficients:

* Intercept: -5.59784584
* Coefficients: [[ 9.80245029e-02, 2.57171472e-02, -1.64398748e-02, 5.56812776e-03, -1.16629340e-04, 4.97494922e-02, 8.27767282e-01, 1.35718227e-02]]

### Model Evaluation

Using another scikit-learn module, I found out the accuracy score on the training data.

* Accuracy score of training data: 0.776536312849162 (77%)

I also used a confusion matrix algorithm for evaluating the model’s performance. Confusion matrix, an essential tool in evaluating the performance of a classification model, encompasses elements representing True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) values. Furthermore, validation techniques ensure the model’s robustness, guard against overfitting, and enhance its generalization capabilities.

#### Training Confusion Matrix Results:

* TP: 112, FN: 78, FP: 42, TN: 305

I used the predict method on the LogisticRegression model from scikit-learn to predict the target values for the test set. Based on that, I got this accuracy score for the test data:

* Accuracy score of test data: 0.7532467532467533 (75%)

Once again, I used the confusion matrix algorithm for the model’s performance on the test data.

#### Testing Confusion Matrix Results:

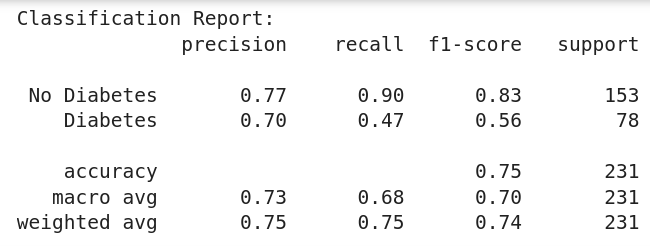
* TP: 37, FN: 41, FP: 16, TN: 137

### Precision and Recall

* Precision: The ratio of true positives to the sum of true positives and false positives.
  + Formula: TP / ( TP + FP )
  + For my trained model: 0.70
* Recall: The ratio of true positives to the sum of true positives and false negatives.
  + Formula: TP / ( TP + FN )
  + For my trained model: 0.47

### Classification Report

The classification report provides a comprehensive view of the precision, recall, and F1-score for each class, as well as the overall accuracy, macro average, and weighted average metrics.



### Model Performance Analysis

#### 1. Model Accuracy:

The accuracy of around 75-77% on both the training and test sets indicates that the model is reasonably good at predicting whether an individual has diabetes. However, accuracy alone is not always the best metric, especially in imbalanced datasets.

#### 2. Confusion Matrix:

The confusion matrix provides a more detailed insight into the performance:

* Training Data: There are 112 true positives and 78 false negatives, indicating some cases of diabetes are missed. The number of false positives is 42, which is relatively lower.
* Testing Data: There are 37 true positives and 41 false negatives, indicating the model misses more cases of diabetes in the test set. The number of false positives is 16, which is relatively lower.

#### 3. Precision and Recall:

* Precision: 70% suggests that when the model predicts diabetes, it is correct 70% of the time.
* Recall: 47% indicates that the model is able to identify 47% of the actual diabetes cases, which is somewhat lower and suggests that the model might be improved to better capture more of the actual positive cases.

#### 4. Classification Report:

* No Diabetes: The model performs better for the "No Diabetes" class with higher precision (0.77) and recall (0.90).
* Diabetes: For the "Diabetes" class, the model has a precision of 0.70 and recall of 0.47, indicating that it struggles more to correctly identify diabetic cases.

### Feature Importance:

The largest positive coefficient, 0.827767282 for the Diabetes Pedigree Function, suggests that this feature has a strong positive impact on the likelihood of diabetes. In contrast, the negative coefficients suggest a negative relationship with diabetes.

### Integration of Regularization Parameters:

Using regularization parameters and grid search cross-validation improved the model’s performance. The best model was found with C=1, penalty='l2', and solver='liblinear'. This approach helped in finding the optimal balance between bias and variance, leading to a slightly better cross-validation score and consistent accuracy on both training and test datasets.

### Conclusion:

The inclusion of regularization and hyperparameter tuning improved the model’s generalization ability. The accuracy remained stable around 75-77%, but the detailed evaluation metrics revealed the model’s strengths and weaknesses, particularly in identifying positive cases of diabetes. Further improvements could be made by exploring different feature engineering techniques, other algorithms, or additional data.

## Jupyter Notebook source code:

<https://github.com/alexban14/DataMining_Diabetes_DS>

## Resources utilized:

https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix-machine-learning/

https://www.datacamp.com/tutorial/understanding-logistic-regression-python

https://books.google.ro/books/about/Data\_Mining\_with\_Python.html?id=LHCd0AEACAAJ&redir\_esc=y