Course: DVA262 — Machine Learning Concepts

Objective: The purpose of the Lab is to apply your understanding of Logistics Regression algorithm by implementing the Logistics Regression algorithm from scratch in the C++ programming using the Lab skeleton.

Lab 3: Logistics Regression Implementation

Execution and demonstration

This lab is executed a group of two students or individually. The completed lab should be demonstrated to a lab assistant.

Useful aids

- The course literature
- Lecture slides and notes
- Google & Internet

Logistics Regression for Classification

You will implement the Logistic Regression algorithm for the classification problem.

Dataset

The classification problem will be based on the **Iris dataset,** which is already available in the project folder.

The Visual Interface: The main form (i.e., MainForm.h) calls all the class objects and functions that are already implemented in the skeleton. However, you may need to edit some parts of the code to make it consistent with your code to display the results correctly.

- *Loading and splitting the dataset*

The code for loading and preparing the dataset is already implemented, so no additional code is required for this part. Use the following function in your code:

- You can use the function loadAndPreprocessDataset() to prepare the dataset, which is located in the class name DataLoader.cpp under DataUtils folder.
- Use the splitDataset() function in the class name DataPreprocessor.cpp under the DataUtils folder to split the dataset into training and test sets.

Your Task: You will mainly work with the LogisticsRegression.cpp C++ Class located under the Classification folder.

- *Creating class member functions*

The LogisticsRegression.cpp class may contain some or all of the following member functions:

- Constructor: initialises the parameters for Logistics Regression
- fit() function: This function train a Logistics Regression model according to the given training data.
- predict() function: This function is used to predict values for a given input vector.

Task 1: In this Lab work you must complete the Logistics Regression implementation by completing the fit() and predict() functions. To complete the fit() function you need to implement the Gradient descent algorithm.

- *Evaluating*

Evaluating the model's performance (in training and test sets) using accuracy, precision, recall and F1 score. The code for these metrics is already done, so no additional code is required.

- You can use accuracy(), precision(), recall() and f1Score() functions, which are located in the class name Metrics.cpp under the Evaluation folder for this evaluation.

Task 2: What value of *learning_rate* and *num_epochs* have you used? Try different values for these parameters. What are optimal values that give you best results? Discuss your findings during the demonstration.

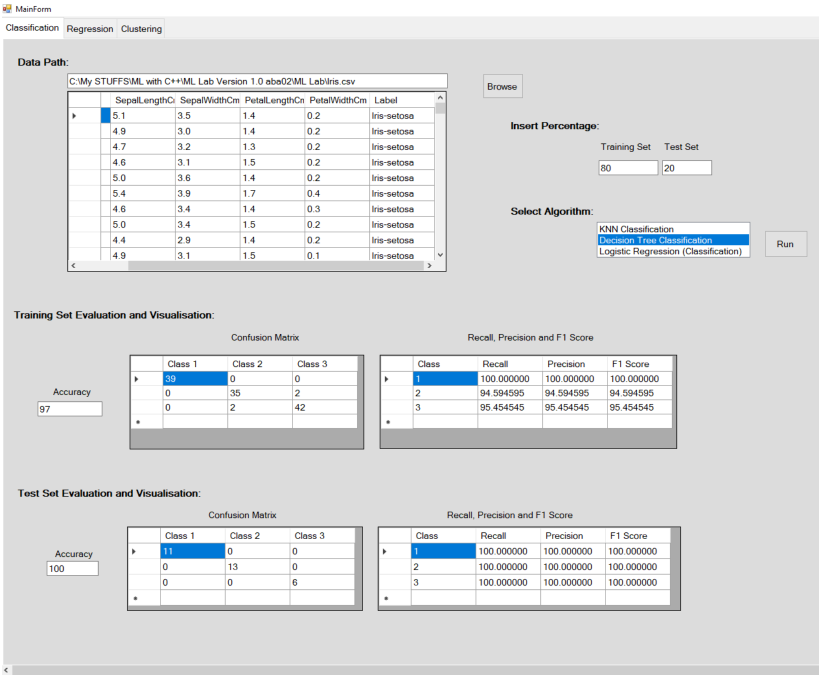
Note: You can change the values that are passed as parameters in the LogisticsRegression() function definition at the LogisticsRegression.h file.

- *Visualisation:*

A visual representation (for training and test sets results) is needed. However, the code for this task is already done, so use the following functions:

- Use the confusionMatrix function located in the class name Metrics.cpp under the Evaluation folder, to calculate the confusion matrix for the specified number of class labels.
- Use the DisplayConfusionMatrixInGrid function located in the MainForm.h, to Display the confusion matrix in the grid.

An example of the result from the Logistics Regression classification is shown in the Figure 1. The visual interface is same for all classifiers.



Data Path:

C:\My STUFFS\ML with C++\ML Lab Version 1.0 aba02\ML Lab\Iris.csv

SepalLengthCı	SepalWidthCm	PetalLengthCn	PetalWidthCm	Label
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3.0	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5.0	3.6	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa
4.6	3.4	1.4	0.3	Iris-setosa
5.0	3.4	1.5	0.2	Iris-setosa
4.4	2.9	1.4	0.2	Iris-setosa
4.9	3.1	1.5	0.1	Iris-setosa

Insert Percentage:

Training Set: 80
Test Set: 20

Select Algorithm:

KNN Classification
Decision Tree Classification
Logistic Regression (Classification)

Training Set Evaluation and Visualisation:

Accuracy: 97

Confusion Matrix

Class 1	Class 2	Class 3
39	0	0
0	35	2
0	2	42

Recall, Precision and F1 Score

Class	Recall	Precision	F1 Score
1	100.000000	100.000000	100.000000
2	94.594595	94.594595	94.594595
3	95.454545	95.454545	95.454545

Test Set Evaluation and Visualisation:

Accuracy: 100

Confusion Matrix

Class 1	Class 2	Class 3
11	0	0
0	13	0
0	0	6

Recall, Precision and F1 Score

Class	Recall	Precision	F1 Score
1	100.000000	100.000000	100.000000
2	100.000000	100.000000	100.000000
3	100.000000	100.000000	100.000000

Figure 1: Example result of Decision Tree classification shown in the visual interface. Logistics Regression uses the same visual interface.

Some Hints:

- Include the necessary header files at the beginning of your code.
- See lecture slides for Gradient descent algorithm.
- Random initialization of weights is better than initializing with zeros.