Course: DVA262 — Machine Learning Concepts

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

Lab 4: Linear Regression Algorithm 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
- Eigen https://eigen.tuxfamily.org/dox/GettingStarted.html
- Google & Internet

Linear Regression for Regression

In this task, you will use both the matrix computation and the Gradient descent algorithms to implement the Linear Regression algorithm for the regression problem.

Dataset

The regression problem will be based on the **Boston Housing Price 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. However, you may need to edit some parts of the code to make it consistent with your code to display the results correctly.

Your Task: You will mainly work with the LinearRegression.cpp class located under the Regression folder.

- *Loading and splitting the dataset*

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

- You can use the function readDatasetFromFilePath() 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.

- *Creating class member functions*

The LinearRegression.cpp class will have the following member functions:

- Constructor: initialises the number of neighbours
- fit() function: this function trains the linear regression model by using the training data set and labels.
- predict() function: This function is used to predict values for a given input vector.

Task 1: In this Lab work you must complete the Linear regression implementation by completing the fit() and predict() functions. You need to implement the *Matrix Form* computation to complete the fit() function.

Task 2: Now apply C++ Function Overloading to implement new fit() and predict() functions using Grading descent algorithm.

- *Evaluating*

Evaluating the regression model's performance using mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and R Squared. The code for these metrics is already done, so no additional code is required.

- You can use meanAbsoluteError, rootMeanSquaredError, and rSquared functions, which are located in the class name Metrics under the Evaluation folder for this evaluation.

Task 3: What differences have you noticed between Matrix Form and Gradient descent methods? Which executes faster? Which approach gives better predictions?

Task 4: Try different values of *learning_rate* and *num_epochs* parameters for the Gradient descent method. What are optimal values that give you the best results? Discuss your findings during the demonstration.

- *Visualisation*

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

- Use the VisualizeParityPlot function, located in the MainForm.h, to display the parity plot for the actual and predicted test labels.

An example of the result from the Linear regression is shown in Figure 1. The visual interface is the same for all regression algorithms.

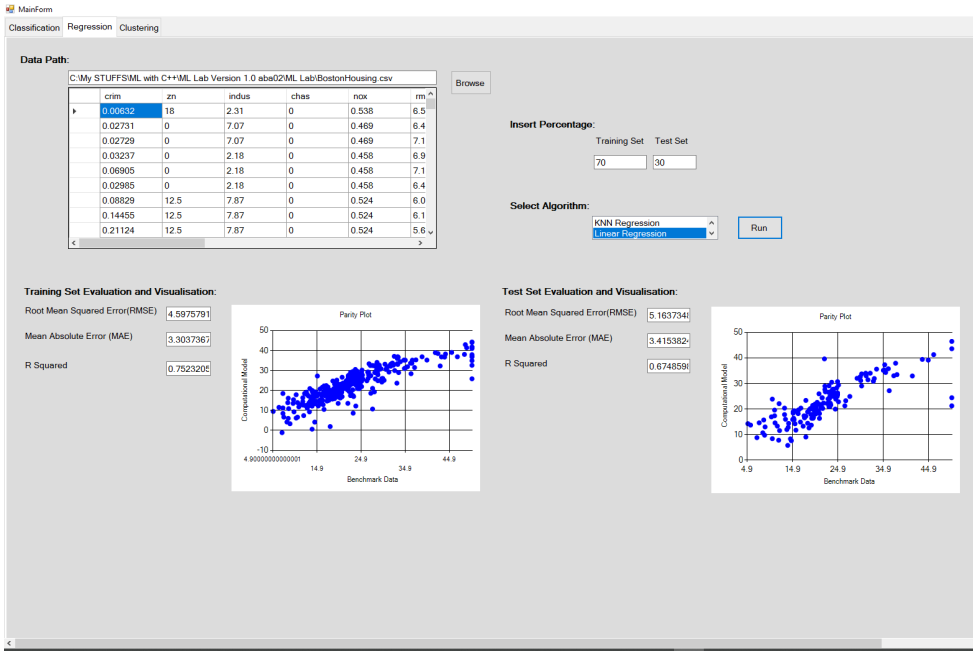

Figure 1: Example results of Linear Regression in the visual interface.

Some Hints:

- Include the necessary header files at the beginning of your code.
- For Gradient Descent, please refer to the instructions provided for the fit() and predict() functions to implement logistic regression.
- Matrix computation can be done using the Eigen library.