OOAD Java Assignment - MVC

April 1, 2022

Yashi Chawla	PES1UG19CS592

1 Problem Statement

The problem chosen for the implementation is **Linear Regression**. Linear Regression is a machine learning algorithm which is used to predict the a real-valued variable from a given set of parameters. The model learns to optimise and learn a solution to $Ax = b$, where A is a matrix consisting of training parameter values and b is a vector consisting of the actual values corresponding to the parameters.

The model implemented here uses the **Normal Equation** to solve for the matrix x, where x is given by $x = (A^T A)^{-1} A^T b$.

2 Model Component

```
import Jama.*;
import java.io.*;
import java.util.*;
import java.lang.Math;

public class LinearRegression
{
    String dataPath;
    Matrix X;
    Matrix Y;
    Matrix theta;

    public void loadDataFromCSV(String dataPath) throws Exception
    {
        File file = new File(dataPath);
        Scanner scanner = new Scanner(file);

        ArrayList<ArrayList<Double>> x = new ArrayList<ArrayList<
    Double>>();
        ArrayList<Double> y = new ArrayList<Double>();
```

```
        while (scanner.hasNextLine())
        {
            String line = scanner.nextLine();
            String[] lineSplit = line.split(",");
            ArrayList<Double> row = new ArrayList<Double>();
            for (int i = 0; i < lineSplit.length - 1; i++)
            {
                row.add(Double.parseDouble(lineSplit[i]));
            }
            x.add(row);
            y.add(Double.parseDouble(lineSplit[lineSplit.length - 1]));
        }

        System.out.println("Loaded " + x.size() + " rows of data.");
        System.out.println("Loaded " + x.get(0).size() + " columns of data.");
        System.out.println("Loaded " + y.size() + " true values.");
        getMatrices(x, y);
    }

    public void getMatrices(ArrayList x, ArrayList y)
    {
        int rows = x.size();
        int cols = ((ArrayList)x.get(0)).size();
        double[][] x_matrix = new double[rows][cols];
        double[][] y_matrix = new double[rows][1];
        for (int i = 0; i < rows; i++)
        {
            y_matrix[i][0] = (double)y.get(i);
            for (int j = 0; j < cols; j++)
                x_matrix[i][j] = (double)((ArrayList)x.get(i)).get(j);
        }
        this.X = new Matrix(x_matrix);
        this.Y = new Matrix(y_matrix);
    }

    public void train()
    {
        this.theta = this.X.solve(this.Y);
    }

    public double predict(ArrayList x)
    {
        double[][] x_matrix = new double[1][x.size()];
        for (int i = 0; i < x.size(); i++)
            x_matrix[0][i] = (double)x.get(i);
        Matrix X = new Matrix(x_matrix);
        Matrix Y = X.times(this.theta);
        double prediction = Y.get(0, 0);
        return prediction;
    }

    public void displayTheta()
    {
```

```
        System.out.println("Theta: ");
        for (int i = 0; i < this.theta.getRowDimension(); i++)
            System.out.print(this.theta.get(i, 0) + " ");
        System.out.println();
    }

    public double calculateLoss()
    {
        Matrix loss = this.Y.minus(this.X.times(this.theta));
        return loss.norm2();
    }
}
```

3 View Component

```
import javax.swing.*;
import java.awt.event.ActionListener;

public class ApplicationView extends JFrame
{
  private JLabel labelPath = new JLabel("Dataset Path",
    SwingConstants.CENTER);
  private JTextField dataPath = new JTextField(30);
  private JButton trainButton = new JButton("Train");
  private JLabel labelInput = new JLabel("Input Values");
  private JTextField inputField = new JTextField(35);
  private JButton predictButton = new JButton("Predict");

  ApplicationView()
  {
    JPanel panel = new JPanel();
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    this.setSize(500, 220);
      this.setLocationRelativeTo(null);

    panel.add(labelPath);
    panel.add(dataPath);
    panel.add(labelInput);
    panel.add(inputField);
    panel.add(trainButton);
    panel.add(predictButton);
    this.add(panel);
  }

  public String getDataPath(){
    String path = dataPath.getText();
    if (path.equals(""))
      JOptionPane.showMessageDialog(null, "Please enter path to
    dataset!");
    else
      return path;
    return null;
  }

  public String getInputValues(){
```

```
    String inputValues = inputField.getText();
    if (inputValues.equals(""))
      JOptionPane.showMessageDialog(null, "Please enter values to predict!");
    else
      return inputValues;
    return null;
  }

  public void addTrainListener(ActionListener listenerForTrainButton)
  {
    trainButton.addActionListener(listenerForTrainButton);
  }

  public void addPredictListener(ActionListener listenerForPredictButton){
    predictButton.addActionListener(listenerForPredictButton);
  }

  void displayErrorMessage(String errorMessage){
    JOptionPane.showMessageDialog(this, errorMessage);
  }
}
```

4 Controller Component

```
import java.io.*;
import java.util.*;
import java.lang.Math;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

class Application
{
    ApplicationView view;
    LinearRegression model;
    Application()
    {
        this.view = new ApplicationView();
        this.view.addTrainListener(new TrainListener());
        this.view.addPredictListener(new PredictListener());
        this.view.setTitle("Linear Regression");
    }

    public LinearRegression createLinearRegressionModel(String dataPath)
    {
        LinearRegression model = new LinearRegression();
        try
        {
            model.loadDataFromCSV(dataPath);
            model.train();
            System.out.println("Training finished!");
        }
```

```
        catch (Exception e)
        {
            System.out.println("Error: " + e.getMessage());
        }
        return model;
    }

    public double predict(LinearRegression model, String values)
    {
        String[] valuesSplit = values.split(",");
        ArrayList<Double> x = new ArrayList<Double>();
        for (int i = 0; i < valuesSplit.length; i++)
            x.add(Double.parseDouble(valuesSplit[i]));
        double prediction = model.predict(x);
        return prediction;
    }

    class TrainListener implements ActionListener
    {
        public void actionPerformed(ActionEvent e)
        {
            String dataPath = view.getDataPath();
            if (dataPath != null){
                model = createLinearRegressionModel(dataPath);
                view.displayErrorMessage("Training finished!");
            }
        }
    }

    class PredictListener implements ActionListener
    {
        public void actionPerformed(ActionEvent e)
        {
            String values = view.getInputValues();
            if (values != null){
                double prediction = predict(model, values);
                prediction = Math.round(prediction * 1000.0) /
    1000.0;
                view.displayErrorMessage("Prediction: " +
    prediction);
            }
        }
    }
}
```

5 Output Screenshots

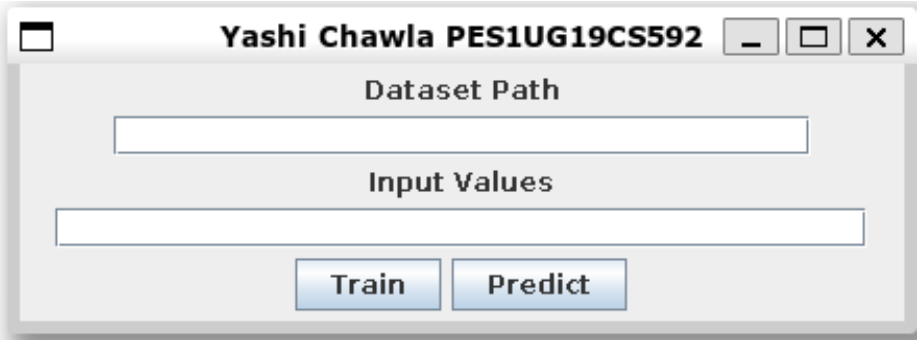

Figure 1: Startup window

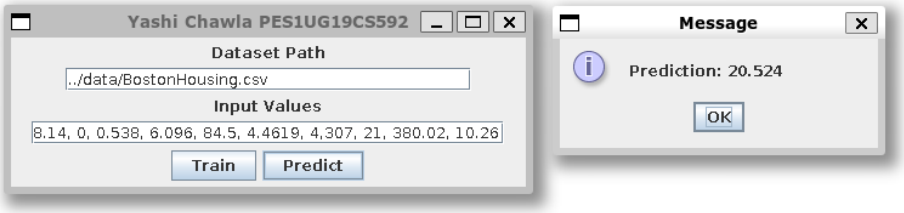

Figure 2: Obtaining Predictions

6 Database

Not applicable.

7 Tools Used

1. JAMA – JAva MAtrix library for Linear Algebra
2. Swing Framework for UI