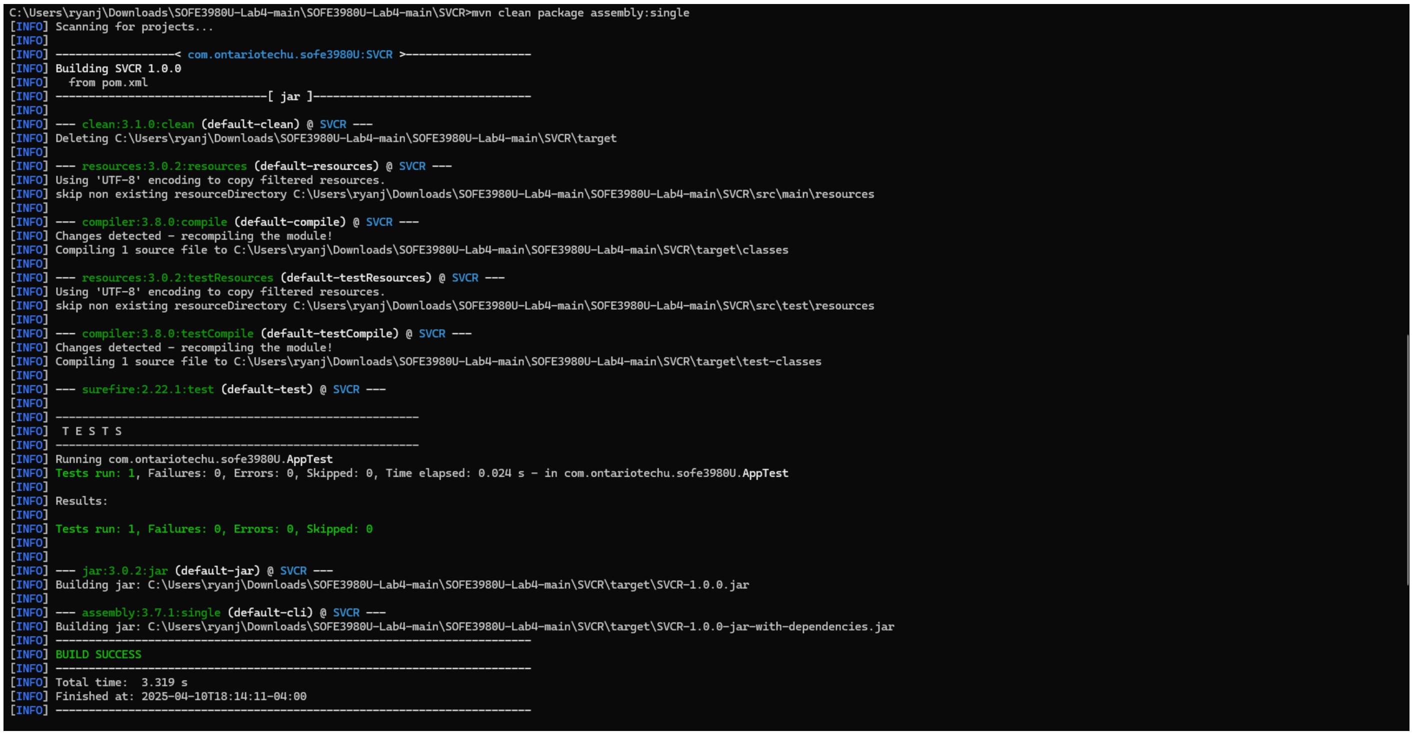
Lab 4: ML Model Testing and Evaluation

Date: Apr 10, 2025

Ryan Varghese, 100870665

1. Single-variable Continuous Regression Problem



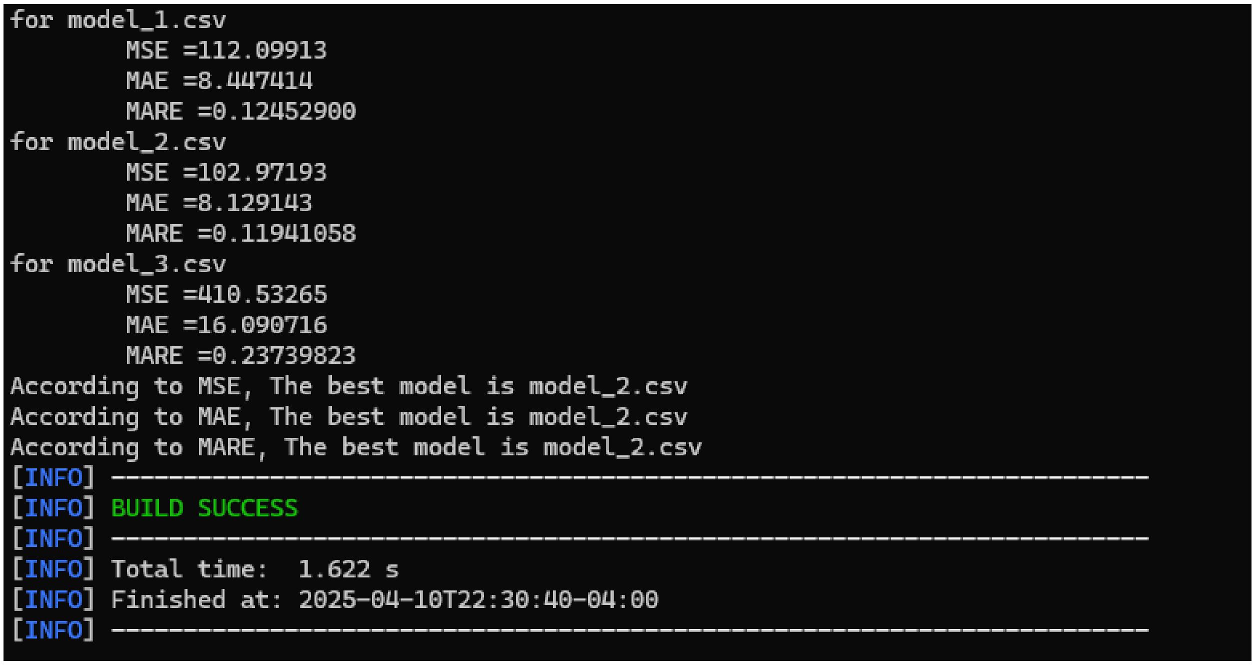
```
C:\Users\ryanj\Downloads\SOFE3980U-Lab4-main\SOFE3980U-Lab4-main\SVCR>mvn clean package assembly:single
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< com.ontariotechu.sofe3980U:SVCR >-------------------
[INFO] Building SVCR 1.0.0
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- clean:3.1.0:clean (default-clean) @ SVCR ---
[INFO] Deleting C:\Users\ryanj\Downloads\SOFE3980U-Lab4-main\SOFE3980U-Lab4-main\SVCR\target
[INFO]
[INFO] --- resources:3.0.2:resources (default-resources) @ SVCR ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory C:\Users\ryanj\Downloads\SOFE3980U-Lab4-main\SOFE3980U-Lab4-main\SVCR\src\main\resources
[INFO]
[INFO] --- compiler:3.8.0:compile (default-compile) @ SVCR ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to C:\Users\ryanj\Downloads\SOFE3980U-Lab4-main\SOFE3980U-Lab4-main\SVCR\target\classes
[INFO]
[INFO] --- resources:3.0.2:testResources (default-testResources) @ SVCR ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory C:\Users\ryanj\Downloads\SOFE3980U-Lab4-main\SOFE3980U-Lab4-main\SVCR\src\test\resources
[INFO]
[INFO] --- compiler:3.8.0:testCompile (default-testCompile) @ SVCR ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to C:\Users\ryanj\Downloads\SOFE3980U-Lab4-main\SOFE3980U-Lab4-main\SVCR\target\test-classes
[INFO]
[INFO] --- surefire:2.22.1:test (default-test) @ SVCR ---
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running com.ontariotechu.sofe3980U.AppTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.024 s - in com.ontariotechu.sofe3980U.AppTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- jar:3.0.2:jar (default-jar) @ SVCR ---
[INFO] Building jar: C:\Users\ryanj\Downloads\SOFE3980U-Lab4-main\SOFE3980U-Lab4-main\SVCR\target\SVCR-1.0.0.jar
[INFO]
[INFO] --- assembly:3.7.1:single (default-cli) @ SVCR ---
[INFO] Building jar: C:\Users\ryanj\Downloads\SOFE3980U-Lab4-main\SOFE3980U-Lab4-main\SVCR\target\SVCR-1.0.0-jar-with-dependencies.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  3.319 s
[INFO] Finished at: 2025-04-10T18:14:11-04:00
[INFO] ------------------------------------------------------------------------
```

```
package com.ontariotechu.sofe3980U;

import java.io.FileReader;
import java.io.Reader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import org.apache.commons.csv.CSVFormat;
import org.apache.commons.csv.CSVParser;
import org.apache.commons.csv.CSVRecord;

public class App {

    public static void main(String[] args) {
        evaluateModel("model_1.csv");
        evaluateModel("model_2.csv");
        evaluateModel("model_3.csv");
        recommendBestModel();
    }

    static class Metrics {
        String modelName;
        double mse;
        double mae;
        double mare;
    Metrics(String modelName, double mse, double mae, double mare) {
        this.modelName = modelName;
        this.mse = mse;
        this.mae = mae;
        this.mare = mare;
    }
}

static List<Metrics> allMetrics = new ArrayList<>();

public static void evaluateModel(String fileName) {
    double mse = 0, mae = 0, mare = 0;
    int count = 0;

    try {
        Reader in = new FileReader("src/main/resources/" + fileName);
        CSVParser parser = CSVFormat.DEFAULT
                .withFirstRecordAsHeader()
                .parse(in);

        for (CSVRecord record : parser) {
            double actual = Double.parseDouble(record.get("actual"));
            double predicted = Double.parseDouble(record.get("predicted"));
            double error = predicted - actual;

            mse += error * error;
```

```
                mae += Math.abs(error);
                mare += Math.abs(error / actual);
                count++;
            }

            mse /= count;
            mae /= count;
            mare /= count;

            allMetrics.add(new Metrics(fileName, mse, mae, mare));

            System.out.println("For " + fileName);
            System.out.println("    MSE = " + mse);
            System.out.println("    MAE = " + mae);
            System.out.println("    MARE = " + mare);
            System.out.println();

        } catch (IOException e) {
            System.err.println("Error reading file: " + fileName);
            e.printStackTrace();
        }
    }

    public static void recommendBestModel() {
        Metrics bestMSE = allMetrics.get(0);
        Metrics bestMAE = allMetrics.get(0);
        Metrics bestMARE = allMetrics.get(0);
        for (Metrics m : allMetrics) {
            if (m.mse < bestMSE.mse) bestMSE = m;
            if (m.mae < bestMAE.mae) bestMAE = m;
            if (m.mare < bestMARE.mare) bestMARE = m;
        }

        System.out.println("According to MSE, the best model is " + bestMSE.modelName);
        System.out.println("According to MAE, the best model is " + bestMAE.modelName);
        System.out.println("According to MARE, the best model is " + bestMARE.modelName);
    }
}
```

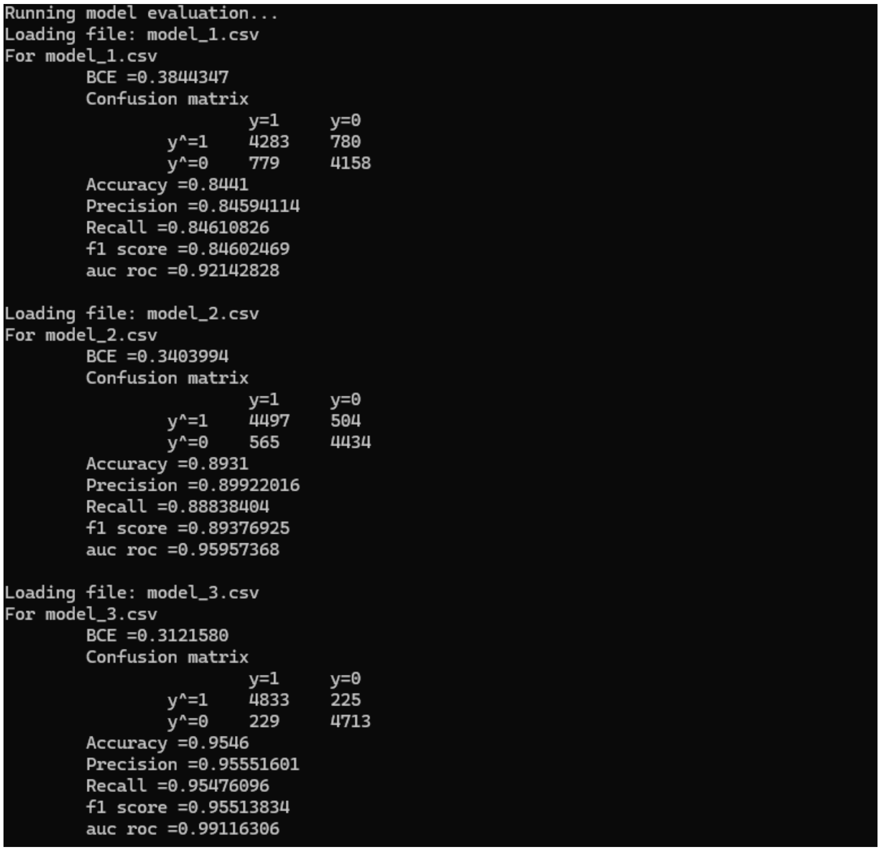
Task 1:

```
package com.ontariotechu.sofe3980U;

import org.apache.commons.csv.CSVFormat;
import org.apache.commons.csv.CSVRecord;

import java.io.FileReader;
import java.io.Reader;
import java.util.*;

public class App {

    public static void main(String[] args) throws Exception {
        Map<String, double[]> modelMetrics = new LinkedHashMap<>();
    String[] modelFiles = {"model_1.csv", "model_2.csv", "model_3.csv"};

    for (String file : modelFiles) {
        double[] metrics = evaluateModel("src/main/resources/" + file);
        modelMetrics.put(file, metrics);
        System.out.printf("for %s%n", file);
        System.out.printf("\tMSE =%.5f%n", metrics[0]);
        System.out.printf("\tMAE =%.6f%n", metrics[1]);
        System.out.printf("\tMARE =%.8f%n", metrics[2]);
    }

    // Find best model for each metric
    String bestMSE = getBestModel(modelMetrics, 0);
    String bestMAE = getBestModel(modelMetrics, 1);
    String bestMARE = getBestModel(modelMetrics, 2);

    System.out.println("According to MSE, The best model is " + bestMSE);
    System.out.println("According to MAE, The best model is " + bestMAE);
    System.out.println("According to MARE, The best model is " + bestMARE);
}

public static double[] evaluateModel(String filepath) throws Exception {
    Reader in = new FileReader(filepath);
    Iterable<CSVRecord> records = CSVFormat.DEFAULT.withFirstRecordAsHeader().parse(in);

    double sumSquared = 0.0;
    double sumAbsolute = 0.0;

    double sumRelative = 0.0;

    int count = 0;


    for (CSVRecord record : records) {

        double actual = Double.parseDouble(record.get("true"));

        double predicted = Double.parseDouble(record.get("predicted"));


        double error = predicted - actual;

        sumSquared += error * error;

        sumAbsolute += Math.abs(error);

        if (actual != 0) {

            sumRelative += Math.abs(error / actual);

        }

        count++;

    }


    double mse = sumSquared / count;

    double mae = sumAbsolute / count;

    double mare = sumRelative / count;


    return new double[]{mse, mae, mare};

}


public static String getBestModel(Map<String, double[]> metrics, int index) {

    return metrics.entrySet().stream()

            .min(Comparator.comparingDouble(e -> e.getValue()[index]))
            .map(Map.Entry::getKey)

            .orElse("No model");

}

}
```



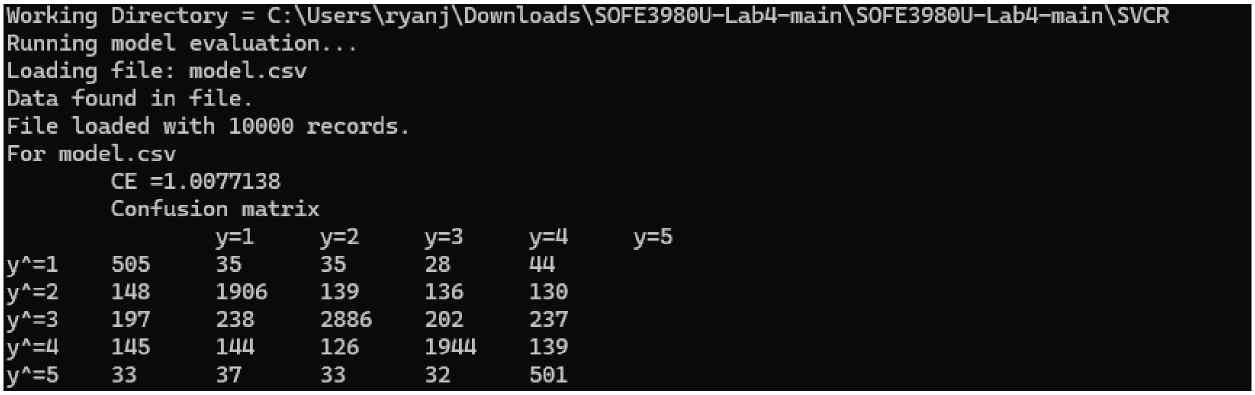
```
for model_1.csv
        MSE =112.09913
        MAE =8.447414
        MARE =0.12452900
for model_2.csv
        MSE =102.97193
        MAE =8.129143
        MARE =0.11941058
for model_3.csv
        MSE =410.53265
        MAE =16.090716
        MARE =0.23739823
According to MSE, The best model is model_2.csv
According to MAE, The best model is model_2.csv
According to MARE, The best model is model_2.csv
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.622 s
[INFO] Finished at: 2025-04-10T22:30:40-04:00
[INFO] ------------------------------------------------------------------------
```

Task 2:



```
Running model evaluation...
Loading file: model_1.csv
For model_1.csv
        BCE =0.3844347
        Confusion matrix
                        y=1     y=0
                y^=1    4283    780
                y^=0    779     4158
        Accuracy =0.8441
        Precision =0.84594114
        Recall =0.84610826
        f1 score =0.84602469
        auc roc =0.92142828

Loading file: model_2.csv
For model_2.csv
        BCE =0.3403994
        Confusion matrix
                        y=1     y=0
                y^=1    4497    504
                y^=0    565     4434
        Accuracy =0.8931
        Precision =0.89922016
        Recall =0.88838404
        f1 score =0.89376925
        auc roc =0.95957368

Loading file: model_3.csv
For model_3.csv
        BCE =0.3121580
        Confusion matrix
                        y=1     y=0
                y^=1    4833    225
                y^=0    229     4713
        Accuracy =0.9546
        Precision =0.95551601
        Recall =0.95476096
        f1 score =0.95513834
        auc roc =0.99116306
```

Task 3:



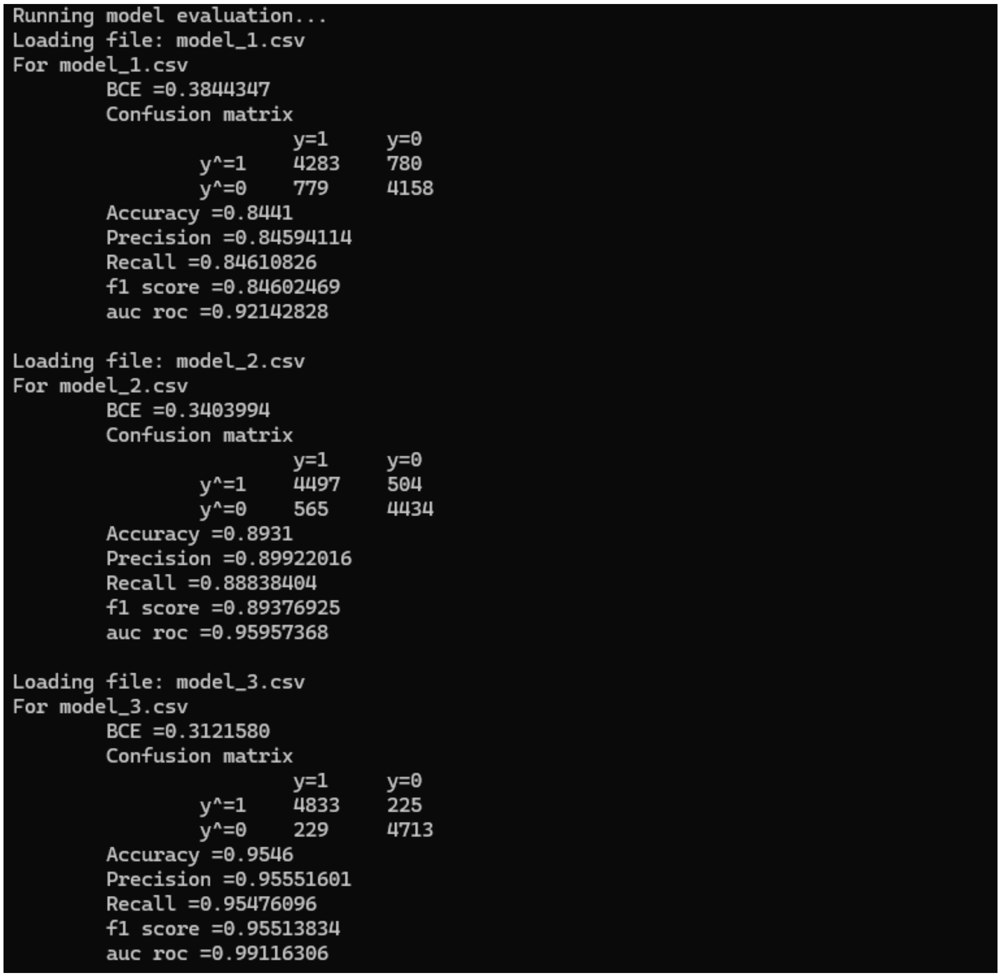
```
Working Directory = C:\Users\ryanj\Downloads\SOFE3980U-Lab4-main\SOFE3980U-Lab4-main\SVCR
Running model evaluation...
Loading file: model.csv
Data found in file.
File loaded with 10000 records.
For model.csv
        CE =1.0077138
        Confusion matrix
                y=1     y=2     y=3     y=4     y=5
y^=1    505     35      35      28      44
y^=2    148     1906    139     136     130
y^=3    197     238     2886    202     237
y^=4    145     144     126     1944    139
y^=5    33      37      33      32      501
```

2. Single-variable Binary Regression Problem

```
package com.ontariotechu.sofe3980U;


import org.apache.commons.csv.CSVFormat;

import org.apache.commons.csv.CSVRecord;


import java.io.FileReader;

import java.io.Reader;

import java.util.ArrayList;

import java.util.List;


public class App {


    public static void main(String[] args) throws Exception {

        evaluateModel("model_1.csv");

        evaluateModel("model_2.csv");

        evaluateModel("model_3.csv");

    }
```

```
public static void evaluateModel(String fileName) throws Exception {
    Reader in = new FileReader("src/main/resources/" + fileName);
    Iterable<CSVRecord> records = CSVFormat.DEFAULT.withFirstRecordAsHeader().parse(in);

    List<Double> trueValues = new ArrayList<>();
    List<Double> predictedValues = new ArrayList<>();

    for (CSVRecord record : records) {
        double actual = Double.parseDouble(record.get("true"));
        double predicted = Double.parseDouble(record.get("predicted"));
        trueValues.add(actual);
        predictedValues.add(predicted);
    }

    int n = trueValues.size();
    double bce = 0;
    int TP = 0, TN = 0, FP = 0, FN = 0;

    for (int i = 0; i < n; i++) {
        double y = trueValues.get(i);
        double y_hat = predictedValues.get(i);

        // BCE calculation
        if (y == 1) {
            bce += Math.log(y_hat);
        } else {
            bce += Math.log(1 - y_hat);
    }

    // Confusion matrix (threshold = 0.5)
    int y_bin = (y_hat >= 0.5) ? 1 : 0;
    if (y == 1 && y_bin == 1) TP++;
    if (y == 1 && y_bin == 0) FN++;
    if (y == 0 && y_bin == 0) TN++;
    if (y == 0 && y_bin == 1) FP++;
}

bce = -bce / n;
double accuracy = (TP + TN) / (double) n;
double precision = TP / (double) (TP + FP);
double recall = TP / (double) (TP + FN);
double f1 = 2 * precision * recall / (precision + recall);

// AUC ROC
List<Double> tprList = new ArrayList<>();
List<Double> fprList = new ArrayList<>();
int positives = 0, negatives = 0;
for (double y : trueValues) {
    if (y == 1) positives++;
    else negatives++;
}

for (int t = 0; t <= 100; t++) {
    double threshold = t / 100.0;
    int tp = 0, fp = 0;

    for (int i = 0; i < n; i++) {
        double y = trueValues.get(i);
        double y_hat = predictedValues.get(i);
        if (y == 1 && y_hat >= threshold) tp++;
        if (y == 0 && y_hat >= threshold) fp++;
    }

    double tpr = tp / (double) positives;
    double fpr = fp / (double) negatives;
    tprList.add(tpr);
    fprList.add(fpr);
}

double auc = 0;
for (int i = 1; i < tprList.size(); i++) {
    auc += (tprList.get(i - 1) + tprList.get(i)) * Math.abs(fprList.get(i - 1) - fprList.get(i)) / 2.0;
}

// Print results
System.out.println("For " + fileName);
System.out.printf("\tBCE =%.7f\n", bce);
System.out.println("\tConfusion matrix\n\t\t\ty=1\ty=0");
System.out.printf("\t\ty^=1\t%d\t%d\n", TP, FP);
System.out.printf("\t\ty^=0\t%d\t%d\n", FN, TN);
System.out.printf("\tAccuracy =%.4f\n", accuracy);
```

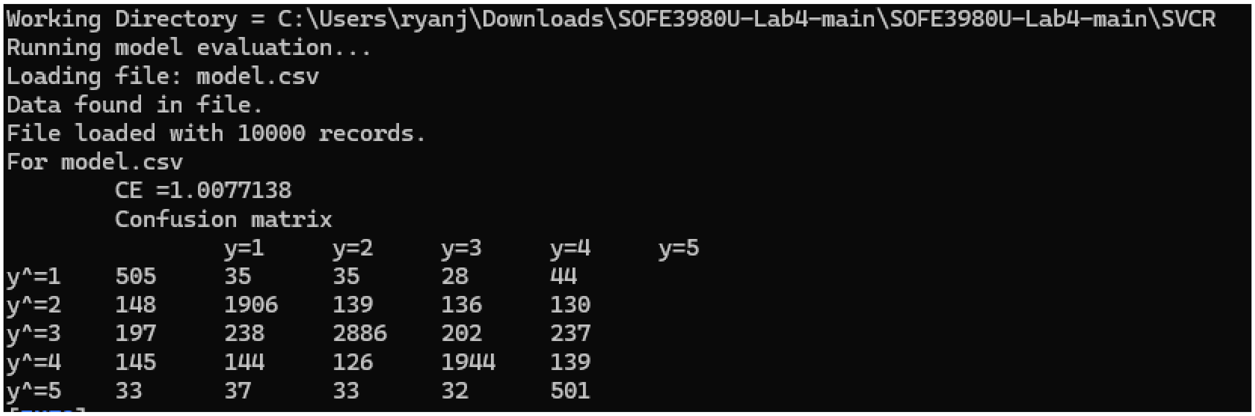
```
        System.out.printf("\tPrecision =%.8f\n", precision);

        System.out.printf("\tRecall =%.8f\n", recall);

        System.out.printf("\tf1 score =%.8f\n", f1);

        System.out.printf("\tauc roc =%.8f\n\n", auc);

    }

}
```



```
Running model evaluation...
Loading file: model_1.csv
For model_1.csv
        BCE =0.3844347
        Confusion matrix
                        y=1     y=0
                y^=1    4283    780
                y^=0    779     4158
        Accuracy =0.8441
        Precision =0.84594114
        Recall =0.84610826
        f1 score =0.84602469
        auc roc =0.92142828

Loading file: model_2.csv
For model_2.csv
        BCE =0.3403994
        Confusion matrix
                        y=1     y=0
                y^=1    4497    504
                y^=0    565     4434
        Accuracy =0.8931
        Precision =0.89922016
        Recall =0.88838404
        f1 score =0.89376925
        auc roc =0.95957368

Loading file: model_3.csv
For model_3.csv
        BCE =0.3121580
        Confusion matrix
                        y=1     y=0
                y^=1    4833    225
                y^=0    229     4713
        Accuracy =0.9546
        Precision =0.95551601
        Recall =0.95476096
        f1 score =0.95513834
        auc roc =0.99116306
```

Multiclass Classification:

```
package com.ontariotechu.sofe3980U;

import org.apache.commons.csv.CSVFormat;
import org.apache.commons.csv.CSVRecord;

import java.io.FileReader;
import java.io.Reader;
import java.util.ArrayList;
import java.util.List;

public class App {

    public static void main(String[] args) {
        System.out.println("Working Directory = " + System.getProperty("user.dir"));
        System.out.println("Running model evaluation...");
        try {
            evaluateModel("model.csv");
        } catch (Exception e) {
            System.err.println("Error occurred: " + e.getMessage());
            e.printStackTrace();
        }
    }

    public static void evaluateModel(String fileName) throws Exception {
        Reader in = new FileReader(fileName);
        Iterable<CSVRecord> records = CSVFormat.DEFAULT.withFirstRecordAsHeader().parse(in);

        System.out.println("Loading file: " + fileName);
        List<Integer> trueValues = new ArrayList<>();
        List<List<Double>> predictedValues = new ArrayList<>();

        if (records.iterator().hasNext()) {
            System.out.println("Data found in file.");
        } else {
            System.out.println("No data found in the file.");
        }


        for (CSVRecord record : records) {
            int actual = Integer.parseInt(record.get("true")); // The actual class
            List<Double> predicted = new ArrayList<>();
            for (int i = 1; i <= 5; i++) { // Assuming 5 classes (columns 1 to 5 are predicted probabilities for each class)
                predicted.add(Double.parseDouble(record.get("predicted_" + i)));
            }
            trueValues.add(actual);
            predictedValues.add(predicted);
        }

        System.out.println("File loaded with " + trueValues.size() + " records.");
        int n = trueValues.size();
        double ce = 0;
        int[][] confusionMatrix = new int[5][5]; // 5 classes, hence a 5x5 confusion matrix

        // Cross-Entropy and Confusion Matrix Calculation
        for (int i = 0; i < n; i++) {
            int y = trueValues.get(i);
            List<Double> y_hat = predictedValues.get(i);

            // Cross-Entropy Calculation
            ce -= Math.log(y_hat.get(y - 1)); // y-1 since the classes are 1-indexed

            // Predicted class (choose the class with highest probability)
            int predictedClass = maxIndex(y_hat) + 1; // Add 1 for 1-indexed classes

            // Confusion Matrix
            confusionMatrix[y - 1][predictedClass - 1]++;
        }

        ce /= n;

        // Printing Results
        System.out.println("For " + fileName);
        System.out.printf("\tCE =%.7f\n", ce);
        System.out.println("\tConfusion matrix");
        System.out.print("\t\t");
        for (int i = 1; i <= 5; i++) {
            System.out.print("y=" + i + "\t");
        }
        System.out.println();
        for (int i = 0; i < 5; i++) {
            System.out.print("y^=" + (i + 1) + "\t");
            for (int j = 0; j < 5; j++) {
                System.out.print(confusionMatrix[i][j] + "\t");
            }
            System.out.println();
        }
    }

    // Helper function to get the index of the maximum element (for predicted class)
    private static int maxIndex(List<Double> values) {
        int maxIdx = 0;
        double maxVal = values.get(0);
        for (int i = 1; i < values.size(); i++) {
            if (values.get(i) > maxVal) {
                maxVal = values.get(i);
                maxIdx = i;
            }
        }
        return maxIdx;
    }
}
```



```
Working Directory = C:\Users\ryanj\Downloads\SOFE3980U-Lab4-main\SOFE3980U-Lab4-main\SVCR
Running model evaluation...
Loading file: model.csv
Data found in file.
File loaded with 10000 records.
For model.csv
        CE =1.0077138
        Confusion matrix
                y=1     y=2     y=3     y=4     y=5
y^=1    505     35      35      28      44
y^=2    148     1906    139     136     130
y^=3    197     238     2886    202     237
y^=4    145     144     126     1944    139
y^=5    33      37      33      32      501
```

Discussion:

- Accuracy: is how many instances were correctly predicted compared with the total number of instances
  - Example:Balanced image classification
- Precision: When a model has a positive prediction, how often is it correct?
  - Email spam filter
- Recall: how many of the actual positive cases the model catches
  - Cancer detection

https://github.com/RyanJohnV/RyanVarghese_100870665_SoftQuality_Lab4V2.git