Table of Contents

List of Figures

Table of Abbreviations

HS: Happiness_Score

DL: Decibel_Level

CLI: Cost_of_Living_Index

AQI: Air_Quality_Index

GSA: Green_Space_Area

HI: Healthcare_Index

TD: Traffic_Density

1 Problem Analysis

1.1 Rationale/Motivation

For this project, students were required to choose a dataset category. As a previous economics student, I chose to base my project off National Production Index. The National Production Index (NPI) is a measurement of the economic output produced by a country over a period. An increase in NPI suggests that the country has experienced economic growth, whereas a decrease in NPI suggests economic decline. I chose this subject because I am interested in how numerous factors can affect a country's NPI as well as the differences between countries with different NPI scores (how a nation with a higher NPI and one with a lower NPI can differ). I am also interested in how a country's NPI can affect the decisions made by governments and international businesses.

1.2 Problem Background

The dataset given, entitled "City Happiness Index" explores factors affecting the satisfaction of people in cities. This dataset was prepared by Emirhan BULUT. It compares different factors (directly related to NPI) which affect the Happiness Score of different cities over a series of present and future years (2024-2029). All columns are as follows:

- City (the name of the cities in which the row data is collected/predicted)
- Month (the month in which the row data is collected/predicted)
- Year (the year in which the row data is collected/predicted)
- Decibel_Level (the average noise level of the city in decibels)
- Traffic_Density (the level of traffic density ranging from low to very high of the city)
- Green_Space_Area (the percentage of green spaces in the city)
- Air_Quality_Index (a standardized measure from 0-500 to represent the city's air quality)
- Happiness_Score (the average happiness score of the city's inhabitants ranging from 1-10)
- Cost_Of_Living_Index (a standardized measure of the cost of living of members of the city)
- Healthcare_Index (a standardized measure of the quality of healthcare accessible to the city's inhabitants)

In my project, I analyzed how the different factors directly impacted by the NPI of the country through urbanization (decibel level, traffic density, green space area, air quality, cost of living and healthcare) affect the happiness of a city's inhabitants. In this analysis, the following columns were utilized:

- Decibel_Level (numerical data, type int)
- Green_Space_Area (numerical data, type int)
- Air_Quality_Index (numerical data, type int)
- Happiness_Score (numerical data, type float/double)
- Cost_of_Living_Index (numerical data, type int)
- Healthcare_Index (numerical data, type int)

Traffic_Density was excluded from the analysis since it utilizes categorical data of type string which would be less straightforward to analyse.

1.3 Existing Solutions

Happiness Score, or Happiness Index, is an indexation of happiness based off calculating the average of survey results. It ranges from 0-10. It can be affected by factors including urbanization. The advantages and disadvantages of urbanization are represented in this dataset. According to the World Happiness Report, disadvantages of urbanization include a lack of affordable housing which correlates to an increase in the Cost-of-Living Index, a lack of public transportation infrastructure which results in congestion (traffic density) and hence hazardous air pollution levels in cities (lower Air Quality Index). It also causes inefficient land use which causes a decrease in Green Space Area.

Urban development causes an increase in noise pollution (decibel levels) in a city. This can be due to an increase in human population, traffic, industrial activities, etc. It can also be exaggerated due recreational entertainment such as bars and other outdoor events. According to Stansfeld and Matheson (2003), noise pollution has significant non-auditory effects on health, including impacts on mental health, stress levels, sleep quality, and overall well-being.

Green space areas can be lost due to rapid urbanization. This can negatively affect human happiness. Living close to gardens, parks and other natural environments can possibly reduce stress, improve mental health, and improve overall life satisfaction (Mackay & Neill, 2010).

Air quality can be negatively impacted by urbanization in multiple ways. These include the loss of green spaces and the increase of traffic congestion in cities. This can negatively affect human happiness since it can cause numerous adverse health issues. It is also related to increased stress, anxiety, and depression (Zhang et al., 2019).

Urbanization can also cause an increase in the cost of living. However, according to Oishi & Schimmack (2010), it is subjective as to whether this affects human happiness, since it is more the subjective perception of whether one can provide for their needs that could affect happiness.

Healthcare is increased with urbanization. Benefits of this include better access to healthcare and more timely medical treatment. This can lead to increased human happiness since it would improve human satisfaction (Blanchflower & Oswald, 2008).

1.4 Solution Objectives

This project seeks to analyse how the Happiness Score of a city is affected by:

- Decibel Level (it is expected that a higher decibel level would result in a lower happiness score due to increased noise pollution)
- Green Space Area (it is expected that a higher green space area would result in a higher happiness score since increased exposure to nature can have positive mental and physical impacts on people)
- Air Quality Index (it is expected that a higher air quality index would result in a higher happiness score due to the health benefits of a lack of air pollution)
- Cost of Living Index (it is expected that a higher cost of living index would result in a lower happiness score due to the negative impacts of financial stress)
- Healthcare Index (it is expected that a higher healthcare index would result in a higher happiness score due to the positive impacts of accessible and quality healthcare)

These trends will be analysed by:

- Successfully opening the file and processing the data (scrubbing the dataset, separating into column vectors, and normalizing).
- Performing regression analysis on the data to analyse the relationships between the dependent variable (Happiness Score) and the independent variables (Decibel Level, Green Space Area, Air Quality Index, Cost of Living Index and Healthcare Index).
- Outputting a csv file with points on the regression line to visualize the trends observed on Microsoft Excel (the data will be normalized due to their differences in ranges).

2 Algorithm

2.1 Overview of Algorithm Description

This project will analyse the trends between the Happiness Score and other factors of a city (as seen in the objectives). This will be achieved in a series of steps.

1. The file is opened and set to read mode. If successfully opened, the csv file will be read by each row.
2. The data is scrubbed:

- If any of the data in the columns are out of range, the row of the erroneous reading will be removed. A new matrix is then created which excludes any rows with errors.

3. The columns of this matrix is then turned into separate vectors which are named in relation to the column data it represents. The necessary typecasting is done to these column vectors (since they were originally from a string vector) and the data is normalized.

4. The regression coefficient and respective regression line equation is found for each objective. Using this and the x-variable values, the corresponding y-values (Happiness Score) of the regression line are found. This data is made into a csv file whereby the data can now be visualized using Microsoft Excel.

2.2 Function Description

Several functions were used to achieve the algorithm. This algorithm is split into 4 parts: data scrubbing and processing, data normalization, finding the regression line and outputting the values on a csv file. This is then implemented on all objectives.

Data Scrubbing and Processing

1. **vector<vector<string>> scrubCSV(string filename)**

i. The file is opened in read mode.
ii. If the file is successfully opened, the data is named the filename (from main), scrubbed, and prepared to be analysed.

- The header row is skipped so that only the numerical data can be analysed.
- Each line of the csv is read.

- For each line, the data is split into fields using a comma (.) as the delimiter.
- If the value of the 8th column (Happiness_Score) is less than 0 or the value of the 6th column (Green_Space_Area) is more than 100, the row is marked for skipping. This is because Happiness_Score ranges from 0-10 and because Green Space Area is a percentage which cannot exceed 100.
- If the row is not marked for skipping, it adds the row to a vector called raw_data.
- After all the lines of the file is processed, the file is closed.

iii. If the file is not successfully opened, the function prints an error message.

iv. The function returns the vector 'raw_data'.

2. vector<vector<string>> separateColumns(const vector<vector<string>>& matrix)

i. The number of columns is determined by finding the size of the first row (matrix[0]) and saved as an integer (int cols).

ii. A vector of vectors called 'separatedColumns' is initialized with a size equal to the number of columns in the matrix.

iii. Each row of the input matrix is iterated using a loop.

iv. Inside the loop, it iterates every column index ('i') up to the number of columns. It pushes the value of that element (row[i]) into the corresponding column vector in 'separatedColumns'.

v. The value of i increases by 1 until it exceeds the loop conditions.

vi. After iterating over the rows of the input matrix, it returns the vector 'separatedColumns' where each position of the matrix holds a single column vector. The column vectors are named in main based on the data it holds.

3. **vector<int>stringToInt(vector<string> x**)

The column vectors from the dataset had to be read as string variables. As a result, typecasting must take place to convert the type int data (Decibel_Level, Green_Space_Area, Air_Quality_Index, Cost_of_Living_Index and Healhcare_Index).

i. The function accepts the separate column vectors (string vectors).

ii. It creates a vector to store the converted values (convertValue) and an integer variable (convertValue).

iii. The string vector is iterated using an int iterator ('i') and a stringstream element ('iss'), initializing the stringstream element with the contents of the string x[i] where x is the input vector. This attempts to extract an integer value from the stringstream and stores it in convertValue.

iv. If this is successful, it pushes the converted integer value into the converted.

v. After iterating over all elements in x, it returns convertValue.

vi. The converted column vectors are named in main.

4. vector<double>stringToDouble(vector<string> x)

This function does the same action as function 3 (vector<int>stringToInt(vector<string> x)), except the vector is converted to one of type double since Happiness_Score is of type double.

Data Normalization

Data normalization is useful in comparing results within datasets since it scales all datasets to the same scale (between 0 and 1). The formula used is:

$$x_{normalized} = \frac{(x - x_{minimum})}{range\ of\ x}$$

5. vector<double> normalize_int_vector(vector<int> x)

This vector accepts the converted int vector from function 3 (vector<int>stringToInt(vector<string> x)).

The maximum value of the vector is first found:

i. The maximum value (int max) is initialized as 0.

ii. The vector is iterated. If the value of x[i] is greater than max, the value of max is x[i].

iii. At the end of the iteration, max should be the maximum value of the vector.

The minimum value is then found.

i. The minimum value (int min) is initialized to the maximum value of the vector (max).

ii. The vector is iterated. If the value of x[i] is less than min, the value of min is x[i].

iii. At the end of the iteration, min should be the minimum value of the vector.

The range (int range) is found as max - min. Each value of the input vector is normalized and is stored into another vector:

i. A double vector (norm_vector) and a double variable (norm_value) are initialized.
ii. The input vector is iterated. The normalization calculation was done for each value x[i] and the result is stored as norm_value. This is then pushed back into norm_vector.

The vector norm_vector is returned. This function is called in main and the resulting vectors are named in main.

6. vector<double> normalize_HS_vector(vector<double> x)

This vector does the same as function 5 (vector<double> normalize_int_vector(vector<int> x)) but the input vector has elements of type double. This is for the Happiness_Score values. This function is called in main, and the resulting vector is named in main.

Finding the Equation of the Regression Line

A regression line best represents a mathematical relationship between two variables in a dataset. It can be described as the line of best fit. The equation of the regression line is:

$$y = c + bx$$

where 'b' is the gradient and 'c' is the y-intercept.

Formulae used include:

To find mean of x:

$$\bar{x} = \frac{\sum x}{n}$$

To find the mean of y:

$$\bar{y} = \frac{\sum y}{n}$$

Variance of x values:

$$S_{xx} = \frac{\sum(x - \bar{x})}{n}$$

Covariance of x and y values:

$$S_{xy} = \frac{\sum(x - \bar{x})(y - \bar{y})}{n}$$

Gradient of regression line b:

$$b = \frac{S_{xy}}{S_{xx}}$$

To find 'c', substitute $\bar{x}$, $\bar{y}$ and b into the equation $y = c + bx$.

The following functions are used to get the regression line of y (Happiness_Score) on x.

7. double average(vector<double> x)

This function finds the average of the input vector x.

i. The integer sum is initialized as 0.
ii. The vector is iterated, and the sum of all values is found as sum.
iii. The average is sum/x.size(). This value is returned.

8. double Sxx(vector<double> x)

This function finds the variance of the input vector x.

i. The average of the values (xavg)is found using function 7 (double average(vector<double> x))
ii. The variable sumOfSquares is initialized to 0.
iii. The vector is iterated. For every iteration, the value of sumOfSquares is updated to $sumOfSquares + (x[i] - xavg)^2$.
iv. S_{xx} is found by dividing the sumOfSquares by x.size(). S_{xx} is then returned.

9. double Sxy(vector<double> y, vector<double> x)

This function finds the covariance of input vector x and input vector y.

i. The average of vector x (xavg) and of vector y (yavg) are found using function 7 7 (double average(vector<double> x)).
ii. The variable sumOfProducts is initialized to 0.
iii. The vectors are iterated. For every iteration, the value sumOfSquares is updated to $sumOfSquares = sumOfSquares + (y[i] - yavg) * (x[i] - xavg)$.
iv. S_{xy} is found by dividing the sumOfSquares by y.size(). S_{xy} is then returned.

10. double regressionCoeff(double Sxx, double Sxy)

This function finds the gradient of the regression line. From this, the general trend of the relationship of the variables can be found.

i. Double b = S_{xy}/S_{xx}
ii. b is returned.

11. double yintercept(double b, double xavg, double yavg)

This function finds the y-intercept of the regression line by rearranging the formula y=bx + c to find c.

i. double c= y-bx
ii. The complete equation of the regression line is printed.
iii. c is returned

12. vector<double> regressionPoints(vector<double> x, double b, double c)

This function finds the corresponding y values of the x values using the equation of the regression line.

i. A vector y (double) is initialized as well as a variable yval for type double.
ii. The input vector is iterated. For every iteration, yval= bx + c.
iii. The resulting value of yval is pushed back into the vector y.
iv. Vector y is returned.

Outputting resulting values onto a csv file.

This is done so that data visualization could take place using Microsoft Excel.

13. void resulltCSV(string filename, vector<double> x, vector<double>y, string xcolumn)

i. A file (filename) is opened in output mode using an fstream object (resultFile).
ii. If the file is opened successfully, it writes a header line containing the labels xcolumn and "HS" (constant y variable name).
iii. It then iterates over vectors x and y, writing each pair of corresponding values from 'x' and 'y' vectors to the file, separated by a comma and followed by an ending the line ('endl').
iv. When all the data is output to the csv, it outputs a message saying that the file is successfully made.
v. The file is closed.

14. void yVSx(vector<double>y, vector<double>x, string xname, string filename)

i. S_{xx} and S_{xy} are calculated using functions 8 and 9 respectively. The results are stored in variables called S_{xx}Val and S_{xy}Val.
ii. The regression coefficient b is calculated using function 10.
iii. If b is positive, it prints that Happiness Score increases as x increases. If b is negative, it prints that Happiness Score decreases as x increases.
iv. The averages of the y vector and x vector are calculated using function 7.
v. The y-intercept c of the regression line is calculated using function 11.
vi. The corresponding y points to the x points using the equation of the regression line is computed using function 12.
vii. A csv file is created to output the regression points using function 13.

2.3 Algorithm

1. Data Scrubbing and Processing:

- Open the CSV file and skip the header.
- Iterate through each line, checking for validity.
- Save valid rows in the raw_data vector.
- Split the raw_data rows into columns.
- Store each column in a separate vector.

- Convert string columns to integers or doubles where necessary.

2. **Normalization**

- Find values such as max, min and range of the vectors.
- Use these to normalize the vectors.

3. **Regression Analysis:**

- Calculate statistical metrics for the data (variance of x vector, covariance of x and y vectors, average of both vectors).
- Perform regression analysis for each objective.
- Determine regression line equations, coefficients, and y-intercepts.
- Generate regression points for plotting.

4. **Output to CSV Files:**

- Create CSV files for each regression analysis.
- Populate the files with x-values and their corresponding y-values.

3 Implementation

```
#include <fstream>
#include <sstream>
#include <string>
#include <vector>
#include <iostream>
using std::cin;
using std::cout;
using std::endl;
using std::vector;
using std::string;
using std::fstream;
using std::ios;
using std::stringstream;


// Function which reads the csv file
vector<vector<string>> scrubCSV(string filename) {
    fstream dataFile;
    dataFile.open(filename, ios::in); //read mode
    //make sure is open
    vector<vector<string>>raw_data;
    if (dataFile.is_open()) {

        //scrub data
        //take out rows with happiness_index < 0 and green_space_data > 100

        //skip header
        string header;
```

```
        getline(dataFile, header);

        string line;
        while (getline(dataFile, line)) {
            stringstream ss(line);
            vector<string> row;
            string field;
            int columnCount = 0;
            bool skipRow = false; //marker

            while (getline(ss, field, ',')) {
                if ((columnCount == 7 && stoi(field) < 0) || (columnCount
== 5 && stoi(field) > 100)) {
                    skipRow = true; //if this occurs, this row has an error
                    break;
                }
                row.push_back(field);
                columnCount++;
            }

            // Only adds the row if it should not be skipped
            if (!skipRow) {
                raw_data.push_back(row);
            }


    }
    dataFile.close();
}
    else {
        cout << "Error opening file." << endl; //error message if file
cannot open
    }
    return raw_data;
    }

// Separate columns into different vectors
vector<vector<string>> separateColumns(const vector<vector<string>>&
matrix) {
    int cols = matrix[0].size(); // Number of columns
    vector<vector<string>> separatedColumns(cols); // Vector to store
separated columns

    for (const auto& row : matrix) {
        for (int i = 0; i < cols; ++i) {
            separatedColumns[i].push_back(row[i]); // Push each element
into the corresponding column vector
        }
    }

    return separatedColumns;
}

//convert column vector value from string to int
vector<int>stringToInt(vector<string> x) {
    vector<int> converted;
    int convertValue;
    for (int i = 0; i < x.size(); i++) {
        stringstream iss(x[i]);
        if (iss >> convertValue) {
            converted.push_back(convertValue);
```

```
        }
    }
    return converted;
}

//convert column vector value from string to double (Happiness Score)
vector<double>stringToDouble(vector<string> x) {
    vector<double> converted;
    double convertValue;
    for (int i = 0; i < x.size(); i++) {
        stringstream iss(x[i]);
        if (iss >> convertValue) {
            converted.push_back(convertValue);
        }
    }
    return converted;
}
//normalize data set
vector<double> normalize_int_vector(vector<int> x) {
    //find maximum
    double max = 0;
    for (int i = 0; i < x.size(); i++) {
        if (x[i] > max) {
            max = x[i];
        }
    }
    //find minumum
    double min = max;
    for (int i = 0; i < x.size(); i++) {
        if (x[i] < min) {
            min = x[i];
        }
    }
    //find range
    double range = max - min;
    //xnormalized = (x - xminimum) / range of x
    vector<double> norm_vector;
    double norm_value;
    for (int i = 0; i < x.size(); i++) {
        norm_value = (x[i] - min) / range;
        norm_vector.push_back(norm_value);
    }

    return norm_vector;
}

//do the same for happiness score (diff type of vector)

vector<double> normalize_HS_vector(vector<double> x) {
    //find maximum
    double max = 0;
    for (int i = 0; i < x.size(); i++) {
        if (x[i] > max) {
            max = x[i];
        }
    }
    //find minumum
    double min = max;
    for (int i = 0; i < x.size(); i++) {
        if (x[i] < min) {
            min = x[i];
```

```
        }
    }
    //find range
    double range = max - min;
    //xnormalized = (x - xminimum) / range of x
    vector<double> norm_vector;
    double norm_value;
    for (int i = 0; i < x.size(); i++) {
        norm_value = (x[i] - min) / range;
        norm_vector.push_back(norm_value);
    }

    return norm_vector;
}

double average(vector<double> x) {
    double sum = 0;
    for (int i = 0; i < x.size(); i++) {
        sum = sum + x[i];
    }
    double avg = sum / x.size();
    return avg;
}

//find Sxx
double Sxx(vector<double> x) {
    //find xavg
    double xavg = average(x);

    //Find Sxx
    double sumOfSquares = 0;
    for (int i = 0; i < x.size(); i++) {
        sumOfSquares = sumOfSquares + pow((x[i] - xavg), 2);
    }
    double Sxx = sumOfSquares / x.size();
    return Sxx;
}

// Calculate Sxy
double Sxy(vector<double> y, vector<double> x) {

    // Calculate the means of x and y
    double yavg = average(y);
    double xavg = average(x);

    // Calculate the sum of the products of differences
    double sumOfProducts = 0;
    for (int i = 0; i < y.size(); i++) {
        sumOfProducts += (y[i] - yavg) * (x[i] - xavg);
    }

    // Calculate Sxy
    double Sxy = sumOfProducts / y.size();

    return Sxy;
}
//find regression coefficient
double regressionCoeff(double Sxx, double Sxy) {
    double b = Sxy / Sxx;
    return b;
}
```

```
//get equation of line given b, xavg, yavg (find c)
//y=bx+c

double yintercept(double b, double xavg, double yavg) {
    double c = yavg - (b * xavg);
    cout << "eqn of regression line: y=" << b << "x" << "+ (" << c << ")"
<< endl;
    return c;
}

//use equation to make y points for x vector and plug into a csv file
vector<double> regressionPoints(vector<double> x, double b, double c) {
    //y= mx + c
    vector<double> y;
    double yval;
    for (int i = 0; i < x.size(); i++) {
        yval = (b * x[i]) + c;
        y.push_back(yval);
    }
    return y;
}

//write x value and corresponding y value into csv file
void resulltCSV(string filename, vector<double> x, vector<double>y, string
xcolumn){
        fstream resultFile;
        resultFile.open(filename, ios::out);
        if (resultFile.is_open()) {
            resultFile << xcolumn << "," << "HS" << endl;
            for (int i = 0; i < x.size(); i++) {
                resultFile << x[i] << " , " << y[i] << endl;
             }
            cout << filename << " result csv made successfully." << endl;
            resultFile.close();
        }

        else {
            cout << "ERROR opening resultfile" << filename<< endl;
        }
    }

//get regression equations and result csv files for objectives
void yVSx(vector<double>y, vector<double>x, string xname, string filename)
{

    double SxxVal = Sxx(x);
    double SxyVal = Sxy(y, x);
    double b = regressionCoeff(SxxVal, SxyVal);

    if (b > 0) {
        cout << "Regression coefficient is positive (" << b << ") thus
Happiness Score increases as " <<  xname  << "increases" << endl;
    }
    else {
        cout << "Regression coefficient is negative (" << b << ") thus
Happiness Score decreases as "<< xname<< " increases" << endl;
    }
    double xavg = average(x);
    double yavg = average(y);
    double c= yintercept(b, xavg, yavg);
```

```
    vector<double> yreg= regressionPoints(x, b, c); //get regression y
points for x points
    resulltCSV(filename, x, yreg, xname);   //make csv to graph regression
line

}


int main() {
    //open file with train or test
    vector<vector<string>> scrubdata = scrubCSV("train.csv"); //change to
suit
    // Separate columns into different vectors
    vector<vector<string>> separatedColumns = separateColumns(scrubdata);

    //names of used columns
    vector<string> D_L, T_D, G_S_A, A_Q_I, H_S, C_L_I, H_I;
    D_L = separatedColumns[3];
    T_D = separatedColumns[4];
    G_S_A = separatedColumns[5];
    A_Q_I = separatedColumns[6];
    H_S = separatedColumns[7];
    C_L_I = separatedColumns[8];
    H_I = separatedColumns[9];
    //typecast
    //function to convert string vector to int vector for D_L, G_S_A,
A_Q_I, C_L_I, H_I.
    vector<int> convD_L = stringToInt(D_L);
    vector<int> convG_S_A = stringToInt(G_S_A);
    vector<int> convA_Q_I = stringToInt(A_Q_I);
    vector<int> convC_L_I = stringToInt(C_L_I);
    vector<int> convH_I = stringToInt(H_I);

 //normalize int vectors (will turn into double)
    vector<double> normD_L = normalize_int_vector(convD_L);
    vector<double> normG_S_A = normalize_int_vector(convG_S_A);
    vector<double> normA_Q_I = normalize_int_vector(convA_Q_I);
    vector<double> normC_L_I = normalize_int_vector(convC_L_I);
    vector<double> normH_I = normalize_int_vector(convH_I);


    //function to convert string vector to double vector for H_S, and then
normalize
    vector<double>convH_S = stringToDouble(H_S);
    vector<double> normH_S = normalize_HS_vector(convH_S);


    //regression result for H_S vs D_L
    yVSx(normH_S, normD_L,"DL", "HSvsDL.csv");

    //regression result for H_S vs G_S_A
    yVSx(normH_S, normG_S_A, "GSA", "HSvsGSA.csv");

    //regression result for H_S vs A_Q_I
    yVSx(normH_S, normA_Q_I, "AQI", "HSvsAQI.csv");

    //regression result for H_S vs C_L_I
    yVSx(normH_S, normC_L_I, "CLI", "HSvsCLI.csv");

    //regression result for H_S vs H_I
```

```
    yVSx(normH_S, normH_I, "HI", "HSvsHI.csv");

      return 0;
}
```

4 Test Cases

4.1 Achievement

In the given dataset, the csv file “test” was given to test the program. The following results display the outcome from using this csv instead of the “train” csv.

```
Regression coefficient is negative (-1.00233) thus Happiness Score decreases as DL increases
eqn of regression line: y=-1.00233x+ (1.01092)
HSvsDL.csv result csv made successfully.
Regression coefficient is positive (0.802804) thus Happiness Score increases as GSAincreases
eqn of regression line: y=0.802804x+ (0.208569)
HSvsGSA.csv result csv made successfully.
Regression coefficient is negative (-1.09979) thus Happiness Score decreases as AQI increases
eqn of regression line: y=-1.09979x+ (0.864084)
HSvsAQI.csv result csv made successfully.
Regression coefficient is positive (0.760251) thus Happiness Score increases as CLIincreases
eqn of regression line: y=0.760251x+ (0.238368)
HSvsCLI.csv result csv made successfully.
Regression coefficient is positive (1.13351) thus Happiness Score increases as HIincreases
eqn of regression line: y=1.13351x+ (-0.127319)
HSvsHI.csv result csv made successfully.
```

Figure 1: Console Output from Test csv.

From figure 1, it is shown that Happiness Score:

- Decreases as Decibel Level increases (expected outcome)
- Increases as Green Space Area increases (expected outcome)
- Decreases as Air Quality Index increases (not expected)
- Increases as Cost-of-Living Index increases (not expected)
- Increases as Healthcare Index increases (expected outcome)

Although all expected outcomes did not occur, it should be noted that the same outcomes occur when using “train.csv”.

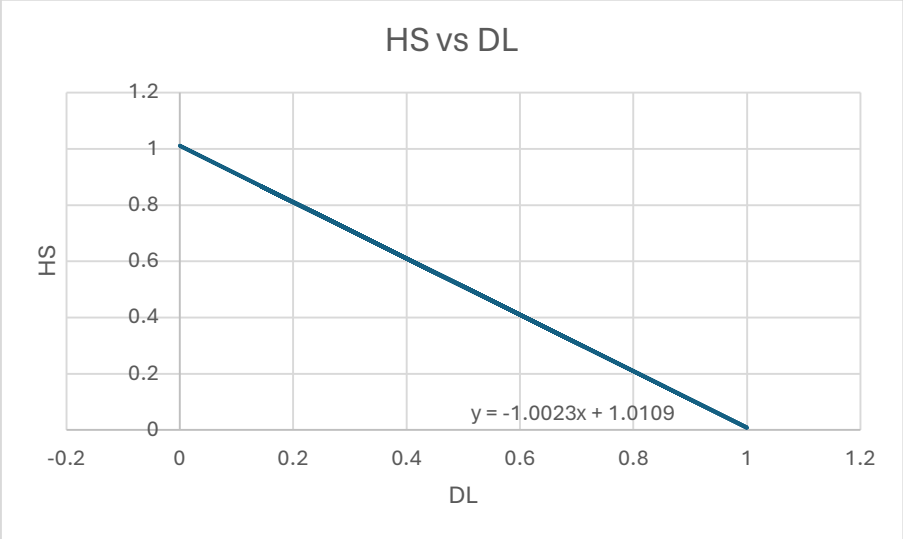

Figure 2: HS vs DL using test.csv.

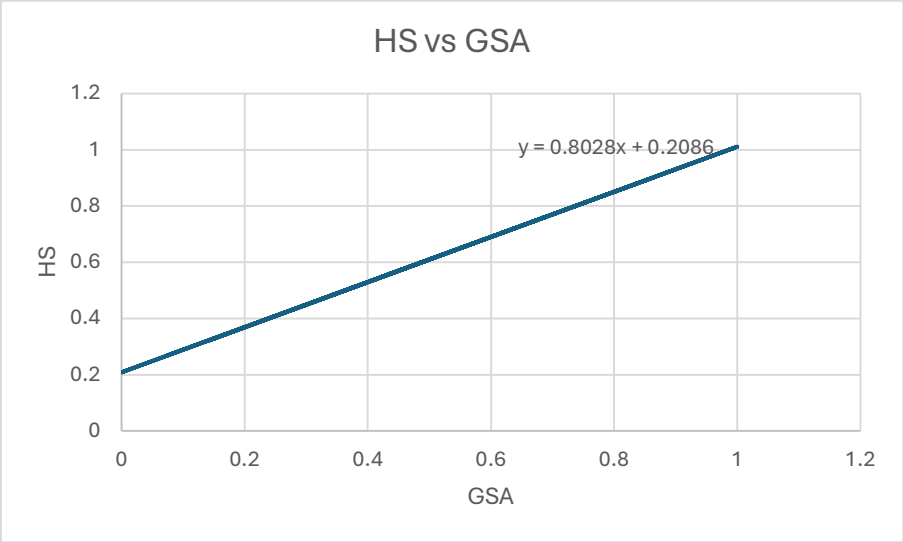

Figure 3: HS vs GSA for test.csv.

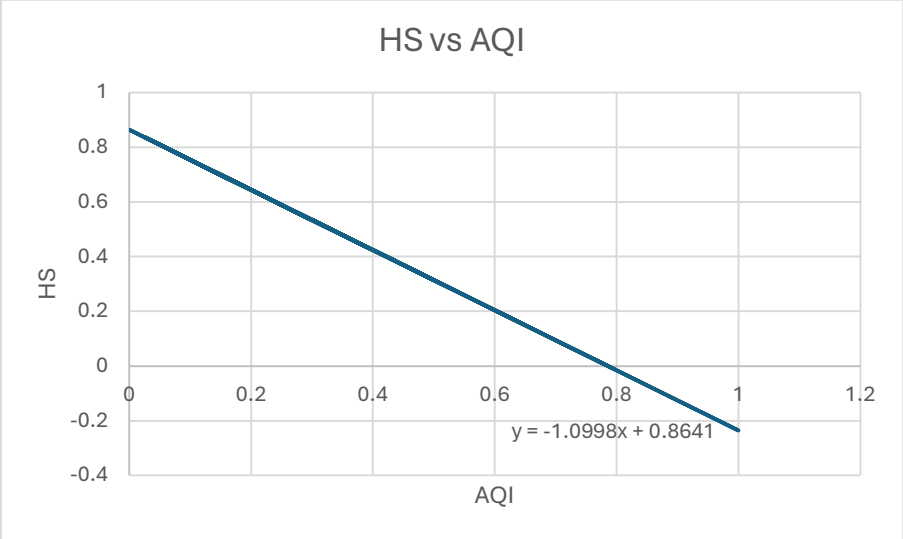

Figure 4: HS vs AQI for test.csv.

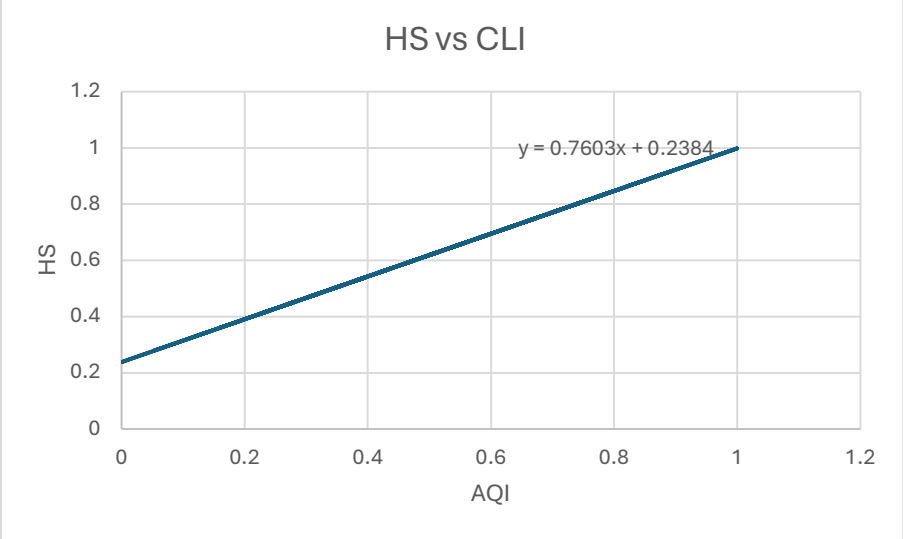

Figure 5: HS vs CLI for test.csv.

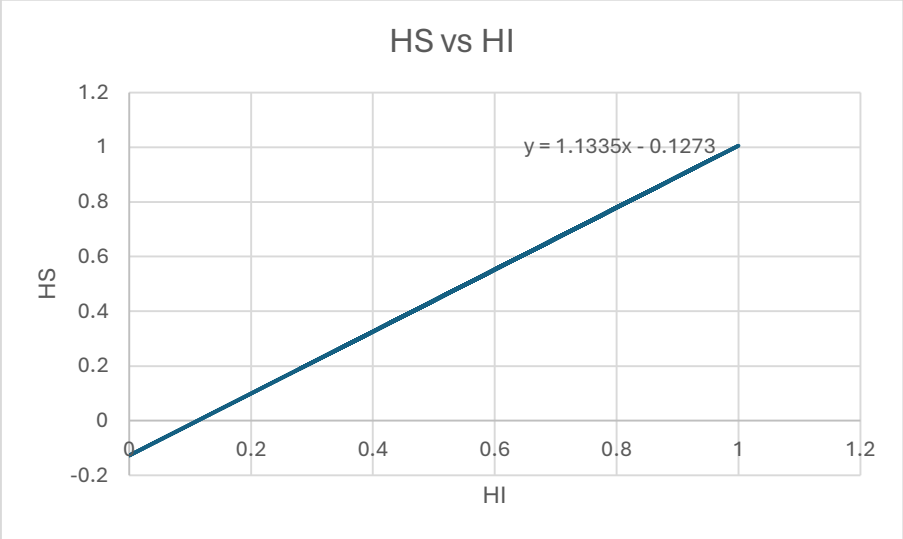

Figure 6: HS vs HI for test.csv.

Figures 2-6 are graphs extracted from the csv files made using the program (see Appendix).

With respect to the stated objectives, it is evident that:

- The file was successfully opened, scrubbed, and processed (separated into workable column vectors and normalized).
- Regression analysis was performed for all comparisons with Happiness Score, and this was used to identify trends in the data.
- The points on the regression line were successfully calculated and output into csv files where they could be visualized and further analysed.

4.2 Invalid Input

1. Giving the wrong name to the file e.g. “train” instead of “train.csv”.

```
int main() {
    //open file with train or test
    vector<vector<string>> scrubdata = scrubCSV("train"); //change to suit
    // Separate columns into different vectors
    vector<vector<string>> separatedColumns = separateColumns(scrubdata);
```

Figure 7: Wrong Filename Input.

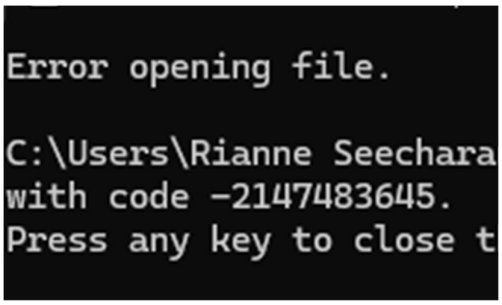

Figure 8: Result of Wrong Filename

2. Giving an invalid name to an output file

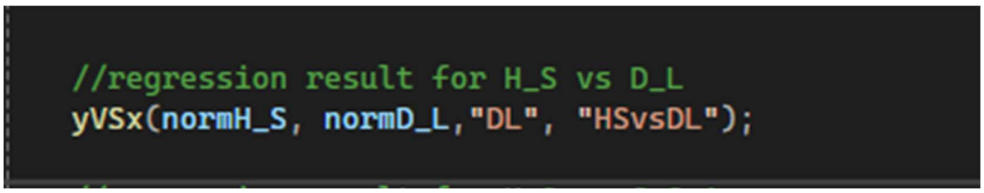

Figure 9: Invalid Name to Output File

When this occurs, the csv file is made.

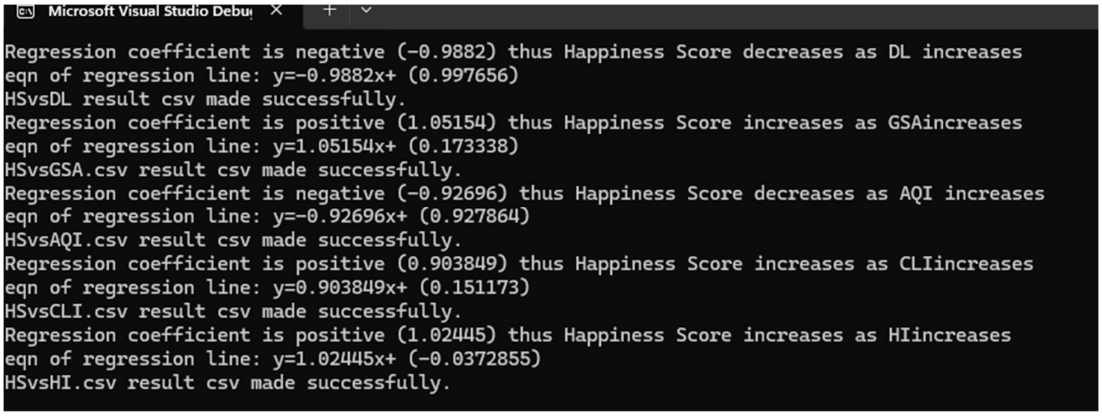

Figure 10: Invalid CSV format result from Console.

However, the csv is not in a spreadsheet format when opened in file explorer.

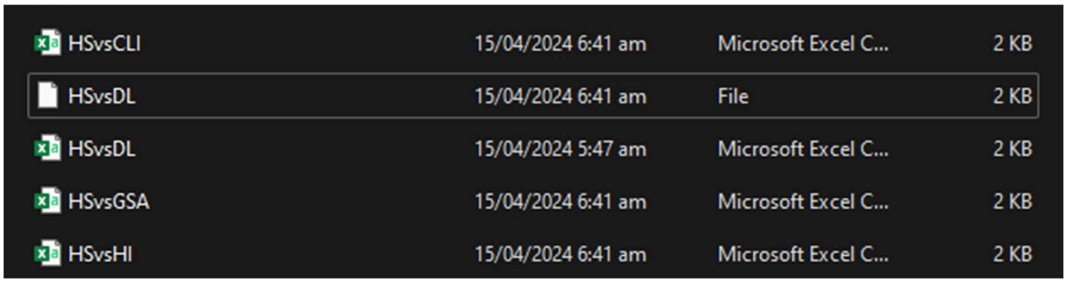

Figure 11: Result of Invalid csv Format in File Explorer.

A summary of the test results is shown in table 1.

Table 1: Summary of Test Cases

Test Case	**Outcome**
Use ‘test.csv’	• File is successfully opened. • Data is scrubbed to skip rows with invalid inputs (out of range inputs). • Data is successfully split into column vectors and processed. • Data undergoes normalization and regression calculations. • Points of the regression line for each case are successfully made into separate csv files and can be used to visualize data and conduct further investigation
Invalid file name was used	• Error message was shown on console.
Invalid CSV format was used	• Program did not successfully warn the user about the error in making the file. • File was made, but it was not made for Microsoft Excel to be used for further investigation

5 Analysis

The first test was to use “test.csv” to ensure that the program can work with data it has never used before. The result was successful since the program was able to accomplish all its objectives for every case as seen in table 1. The second test was also successful since, as the file was not found (it was given an invalid name), the console showed that there was an error as intended. However, the third test result did show an error in the program since the csv file was not successfully built (it was not built to be a spreadsheet). This was because it was named “HSvsDL” instead of “HSvsDL.csv”. Instead of delivering the intended error message, the csv file was made and was able to be opened on Notepad.

With respect to the general trends found in the program, it was noticed that some of the expected outcomes did not come to pass. Happiness Score decreases as Air Quality Index increases in both the train and test csv. A reason for this may be that the inhabitants of the country, although they may not like the decrease in air quality, appreciate the benefits of urbanization which outweighs the disadvantages.

It was also noticed that Happiness Score increases as Cost-of-Living Index increases for the test csv and the train csv. This may be because, as noted in section 1.3, that happiness being affected by cost of living can be perceptive based on the individual’s ability to meet their needs.

6 Results

6.1 “train.csv”

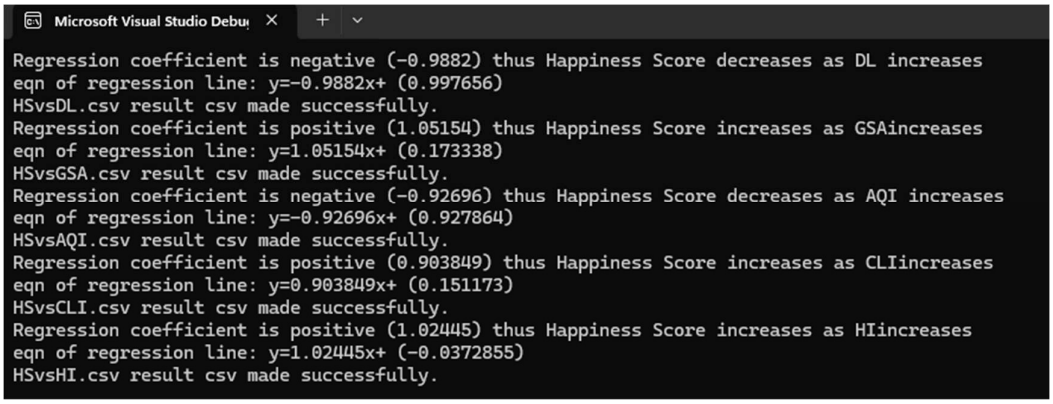

Figure 12: Results for train.csv

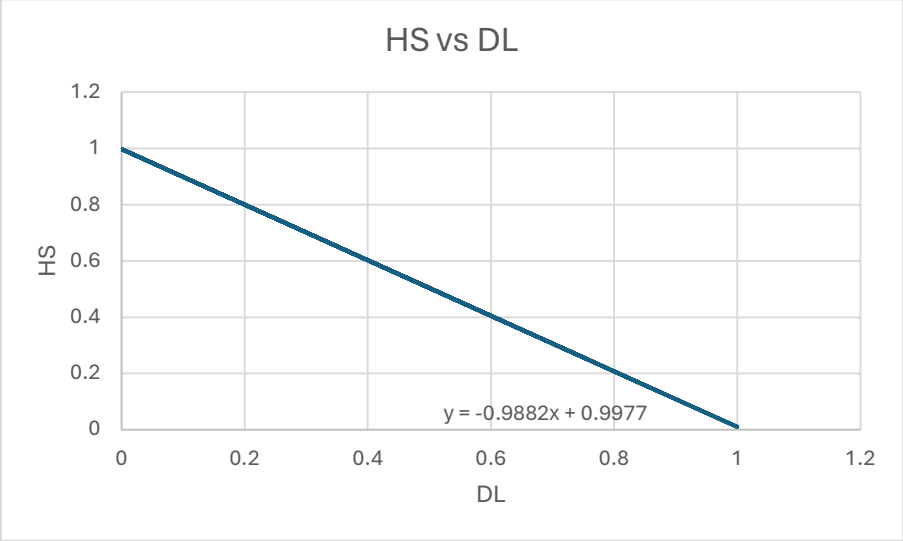

Figure 13: HS vs DL for train.csv.

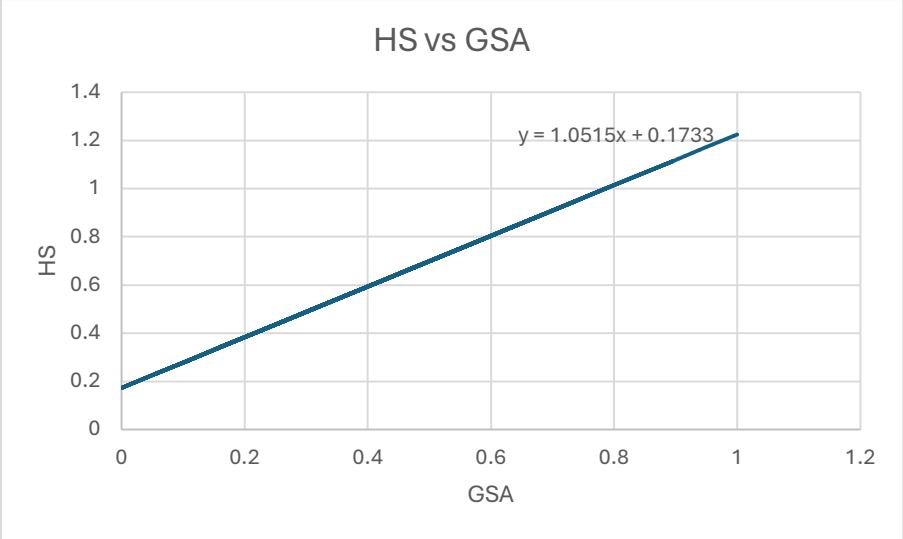

Figure 14: HS vs GSA for train.csv.

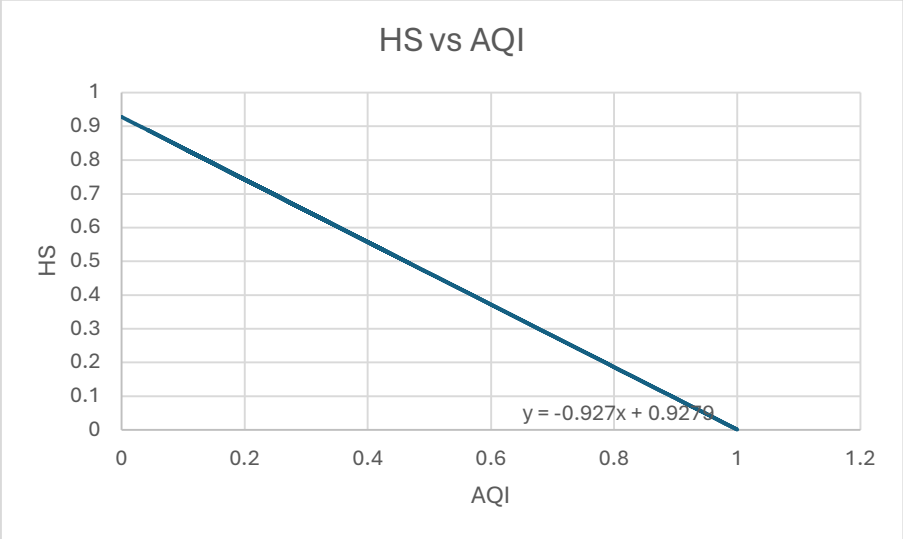

Figure 15: HS vs AQI for train.csv.

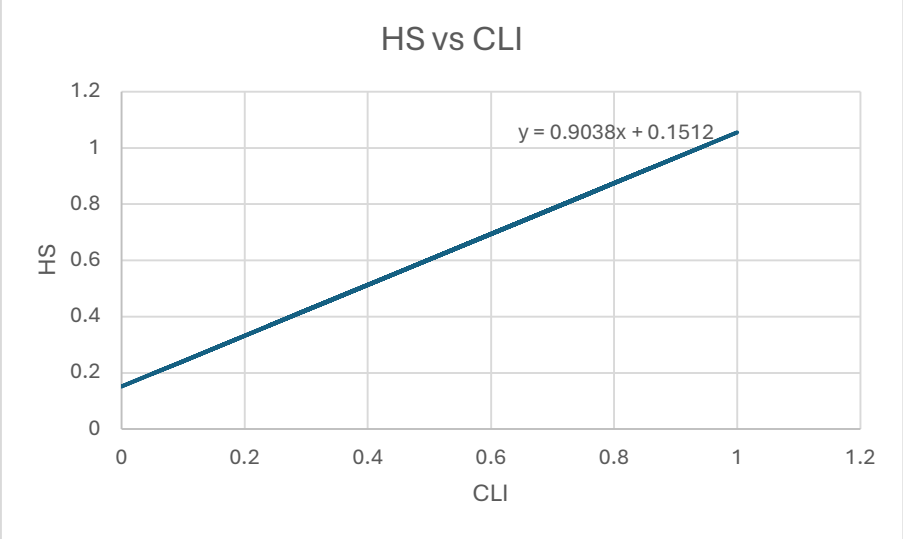

Figure 16: HS vs CLI for train.csv.

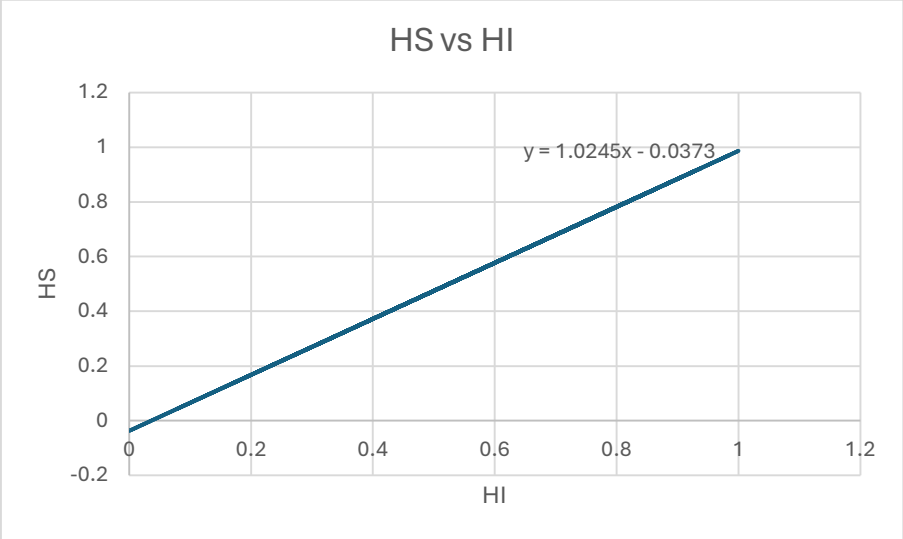

Figure 17: HS vs HI for train.csv.

6.2 "test.csv"

See Test Cases: Achievement (4.1).

References

Blanchflower, D. G., & Oswald, A. J. (2008). Is well-being U-shaped over the life cycle? Social Science & Medicine, 66(8), 1733–1749. https://doi.org/10.1016/j.socscimed.2008.01.030

Mackay, C., & Neill, C. (2010). The effect of "green exercise" on state anxiety and the role of exercise duration, intensity, and greenness: A quasi-experimental study. Psychology of Sport and Exercise, 11(3), 238–245. https://doi.org/10.1016/j.psychsport.2010.01.002

Oishi, S., & Schimmack, U. (2010). Culture and well-being: A new inquiry into the subjective well-being of immigrants. In E. Diener, J. F. Helliwell, & D. Kahneman (Eds.), International differences in well-being (pp. 319–331). Oxford University Press.

Stansfeld, S. A., & Matheson, M. P. (2003). Noise pollution: Non-auditory effects on health. British Medical Bulletin, 68(1), 243–257. https://doi.org/10.1093/bmb/ldg033

World Happiness Report. (2020). Cities and Happiness: A Global Ranking and Analysis. World Happiness Report. https://worldhappiness.report/ed/2020/cities-and-happiness-a-global-ranking-and-analysis/

Zhang, R., Wei, Y., & Li, F. (2019). Air pollution and happiness: Evidence from micro data in China. Journal of Cleaner Production, 235, 160-170. https://doi.org/10.1016/j.jclepro.2019.06.222

Appendix

Train:

DL	HS
0.5	0.503556
0.375	0.627081
0.25	0.750606
0.125	0.874131
0.25	0.750606
0	0.997656
0.375	0.627081
0.5	0.503556
0.75	0.256506
0.125	0.874131
0.625	0.380031
0.375	0.627081
0.25	0.750606
0.5	0.503556
0.125	0.874131
0.75	0.256506
0.375	0.627081
0	0.997656
0.625	0.380031
0.25	0.750606
0.375	0.627081
0.5	0.503556
0.625	0.380031
0.25	0.750606
0.125	0.874131
0.125	0.874131
0	0.997656
0.25	0.750606
0.375	0.627081
0.125	0.874131
0.25	0.750606
0	0.997656
0.125	0.874131
0	0.997656
0.125	0.874131
0.25	0.750606
0.375	0.627081
0.25	0.750606
0.5	0.503556
0.625	0.380031

0.5	0.503556
0.625	0.380031
0.375	0.627081
0.5	0.503556
0.625	0.380031
0.375	0.627081
0.5	0.503556
0.625	0.380031
0.375	0.627081
0.5	0.503556
0.625	0.380031
0.75	0.256506
0.75	0.256506
0.625	0.380031
0.5	0.503556
0.625	0.380031
0.75	0.256506
0.875	0.132981
0.75	0.256506
0.875	0.132981
1	0.009456
0.625	0.380031
0.75	0.256506
0.625	0.380031
0.75	0.256506
0.875	0.132981
0.75	0.256506
0.625	0.380031
0.875	0.132981
1	0.009456
0.875	0.132981
0.25	0.750606
0.125	0.874131
0.25	0.750606
0.125	0.874131
0	0.997656
0.125	0.874131
0	0.997656
0.125	0.874131
0	0.997656

GSA	HS
0.315789	0.505403
0.368421	0.560747

0.263158	0.450059
0.473684	0.671436
0.421053	0.616091
0.526316	0.72678
0.263158	0.450059
0.210526	0.394715
0.105263	0.284027
0.578947	0.782124
0.421053	0.616091
0.473684	0.671436
0.526316	0.72678
0.315789	0.505403
0.578947	0.782124
0.210526	0.394715
0.368421	0.560747
0.736842	0.948156
0.157895	0.339371
0.263158	0.450059
0.421053	0.616091
0.315789	0.505403
0.210526	0.394715
0.631579	0.837468
0.684211	0.892812
0.578947	0.782124
0.789474	1.0035
0.368421	0.560747
0.421053	0.616091
0.684211	0.892812
0.526316	0.72678
0.736842	0.948156
0.789474	1.0035
0.842105	1.05884
0.894737	1.11419
0.473684	0.671436
0.421053	0.616091
0.368421	0.560747
0.315789	0.505403
0.263158	0.450059
0.368421	0.560747
0.210526	0.394715
0.315789	0.505403
0.263158	0.450059
0.157895	0.339371
0.210526	0.394715
0.263158	0.450059

0.210526	0.394715
0.315789	0.505403
0.263158	0.450059
0.210526	0.394715
0.157895	0.339371
0.105263	0.284027
0.210526	0.394715
0.263158	0.450059
0.157895	0.339371
0.105263	0.284027
0.052632	0.228682
0.105263	0.284027
0.052632	0.228682
0	0.173338
0.210526	0.394715
0.157895	0.339371
0.210526	0.394715
0.157895	0.339371
0.105263	0.284027
0.052632	0.228682
0.105263	0.284027
0.052632	0.228682
0	0.173338
0.052632	0.228682
0.578947	0.782124
0.631579	0.837468
0.684211	0.892812
0.736842	0.948156
0.789474	1.0035
0.842105	1.05884
0.894737	1.11419
0.947368	1.16953
1	1.22488

AQI	HS
0.12766	0.809528
0.170213	0.770083
0.191489	0.750361
0.212766	0.730638
0.234043	0.710916
0.106383	0.829251
0.255319	0.691193
0.276596	0.67147
0.297872	0.651748

0.085106	0.848973
0.319149	0.632025
0.191489	0.750361
0.12766	0.809528
0.340426	0.612303
0.148936	0.789806
0.425532	0.533412
0.212766	0.730638
0.170213	0.770083
0.446809	0.51369
0.297872	0.651748
0.212766	0.730638
0.468085	0.493967
0.382979	0.572858
0.106383	0.829251
0.12766	0.809528
0.148936	0.789806
0.106383	0.829251
0.255319	0.691193
0.276596	0.67147
0.085106	0.848973
0.12766	0.809528
0.106383	0.829251
0.085106	0.848973
0.06383	0.868696
0.042553	0.888419
0.234043	0.710916
0.255319	0.691193
0.276596	0.67147
0.297872	0.651748
0.319149	0.632025
0.361702	0.59258
0.382979	0.572858
0.404255	0.553135
0.425532	0.533412
0.446809	0.51369
0.468085	0.493967
0.489362	0.474245
0.510638	0.454522
0.531915	0.4348
0.553191	0.415077
0.574468	0.395355
0.595745	0.375632
0.617021	0.355909
0.638298	0.336187

0.659574	0.316464
0.680851	0.296742
0.702128	0.277019
0.723404	0.257297
0.744681	0.237574
0.765957	0.217851
0.787234	0.198129
0.808511	0.178406
0.829787	0.158684
0.851064	0.138961
0.87234	0.119239
0.893617	0.099516
0.914894	0.079794
0.93617	0.060071
0.957447	0.040348
0.978723	0.020626
1	0.000903
0.170213	0.770083
0.148936	0.789806
0.12766	0.809528
0.106383	0.829251
0.085106	0.848973
0.06383	0.868696
0.042553	0.888419
0.021277	0.908141
0	0.927864

CLI	HS
0.714286	0.796779
0.619048	0.710699
0.571429	0.667658
0.809524	0.88286
0.666667	0.753739
0.52381	0.624618
0.904762	0.968941
0.761905	0.83982
0.428571	0.538537
0.857143	0.925901
0.47619	0.581577
0.52381	0.624618
0.666667	0.753739
0.380952	0.495497
0.428571	0.538537
0.238095	0.366375

0.333333	0.452456
0.761905	0.83982
0.285714	0.409416
0.428571	0.538537
0.952381	1.01198
0.809524	0.88286
1	1.05502
0.619048	0.710699
0.47619	0.581577
0.857143	0.925901
0.714286	0.796779
0.571429	0.667658
0.428571	0.538537
0.666667	0.753739
0.761905	0.83982
0.809524	0.88286
0.714286	0.796779
0.857143	0.925901
0.619048	0.710699
0.52381	0.624618
0.47619	0.581577
0.428571	0.538537
0.333333	0.452456
0.285714	0.409416
0.380952	0.495497
0.285714	0.409416
0.428571	0.538537
0.333333	0.452456
0.238095	0.366375
0.285714	0.409416
0.333333	0.452456
0.380952	0.495497
0.238095	0.366375
0.285714	0.409416
0.428571	0.538537
0.190476	0.323335
0.142857	0.280295
0.285714	0.409416
0.47619	0.581577
0.52381	0.624618
0.571429	0.667658
0.333333	0.452456
0.190476	0.323335
0.142857	0.280295
0.095238	0.237254

0.190476	0.323335
0.142857	0.280295
0.095238	0.237254
0.047619	0.194214
0.095238	0.237254
0.047619	0.194214
0	0.151173
0.190476	0.323335
0.142857	0.280295
0.095238	0.237254
0.809524	0.88286
0.761905	0.83982
0.714286	0.796779
0.666667	0.753739
0.619048	0.710699
0.571429	0.667658
0.52381	0.624618
0.47619	0.581577
0.428571	0.538537

HI	HS
0.703125	0.683031
0.625	0.602996
0.546875	0.522961
0.78125	0.763066
0.703125	0.683031
0.78125	0.763066
0.859375	0.843101
0.625	0.602996
0.390625	0.36289
0.828125	0.811087
0.546875	0.522961
0.625	0.602996
0.78125	0.763066
0.390625	0.36289
0.703125	0.683031
0.3125	0.282855
0.625	0.602996
0.859375	0.843101
0.46875	0.442925
0.703125	0.683031
0.9375	0.923136
0.78125	0.763066
0.859375	0.843101

0.78125	0.763066
0.734375	0.715045
0.859375	0.843101
0.9375	0.923136
0.703125	0.683031
0.625	0.602996
0.9375	0.923136
0.78125	0.763066
0.984375	0.971158
0.96875	0.955151
1	0.987165
0.9375	0.923136
0.78125	0.763066
0.703125	0.683031
0.734375	0.715045
0.625	0.602996
0.546875	0.522961
0.625	0.602996
0.390625	0.36289
0.546875	0.522961
0.46875	0.442925
0.3125	0.282855
0.390625	0.36289
0.46875	0.442925
0.546875	0.522961
0.390625	0.36289
0.46875	0.442925
0.546875	0.522961
0.3125	0.282855
0.234375	0.20282
0.390625	0.36289
0.546875	0.522961
0.625	0.602996
0.546875	0.522961
0.3125	0.282855
0.234375	0.20282
0.15625	0.122785
0.078125	0.04275
0.390625	0.36289
0.3125	0.282855
0.234375	0.20282
0.15625	0.122785
0.078125	0.04275
0	-0.03729
0.078125	0.04275

0.3125	0.282855
0.234375	0.20282
0.15625	0.122785
0.84375	0.827094
0.859375	0.843101
0.875	0.859108
0.890625	0.875115
0.90625	0.891122
0.921875	0.907129
0.9375	0.923136
0.953125	0.939144
0.96875	0.955151

Test:

DL	HS
0.142857	0.867734
0	1.01092
0.714286	0.294976
0.285714	0.724545
0.142857	0.867734
0.285714	0.724545
0.142857	0.867734
0.285714	0.724545
0.571429	0.438166
1	0.008597
0.714286	0.294976
0.285714	0.724545
0.857143	0.151787
0.571429	0.438166
0.142857	0.867734
0.285714	0.724545
0.142857	0.867734
0.428571	0.581355
0.285714	0.724545
0.571429	0.438166
0.142857	0.867734
0.285714	0.724545
0.428571	0.581355
0.714286	0.294976
0.285714	0.724545
0.142857	0.867734
0.285714	0.724545
0.428571	0.581355
0.142857	0.867734

0.285714	0.724545
0.428571	0.581355
0.285714	0.724545
1	0.008597
0.142857	0.867734
0.571429	0.438166
0.714286	0.294976
0.285714	0.724545
0.428571	0.581355
0.571429	0.438166
0.285714	0.724545
0.142857	0.867734
0.571429	0.438166
0.428571	0.581355
0.285714	0.724545
0.142857	0.867734
0.571429	0.438166
0.285714	0.724545
0.142857	0.867734
0.428571	0.581355
0.285714	0.724545
0.428571	0.581355

GSA	HS
1	1.01137
0.714286	0.782
0.071429	0.265912
0.428571	0.552627
0.785714	0.839343
0.571429	0.667314
0.857143	0.896686
1	1.01137
0.357143	0.495284
0	0.208569
0.214286	0.380598
0.785714	0.839343
0	0.208569
0.285714	0.437941
0.928571	0.954029
0.428571	0.552627
0.571429	0.667314
0.357143	0.495284
0.5	0.609971
0.142857	0.323255

0.714286	0.782
0.857143	0.896686
0.571429	0.667314
0.285714	0.437941
0.428571	0.552627
1	1.01137
0.785714	0.839343
0.428571	0.552627
0.714286	0.782
0.642857	0.724657
0.357143	0.495284
0.857143	0.896686
0.071429	0.265912
0.928571	0.954029
0.142857	0.323255
0.214286	0.380598
0.5	0.609971
0.428571	0.552627
0.357143	0.495284
0.571429	0.667314
0.714286	0.782
0.214286	0.380598
0.571429	0.667314
0.357143	0.495284
0.857143	0.896686
0.142857	0.323255
0.642857	0.724657
1	1.01137
0.285714	0.437941
0.642857	0.724657
0.428571	0.552627

AQI	HS
0.076923	0.779484
0.102564	0.751284
0.435897	0.384686
0.128205	0.723084
0.153846	0.694885
0.179487	0.666685
0.051282	0.807684
0.025641	0.835884
0.282051	0.553885
0.641026	0.159088
0.692308	0.102688

0.076923	0.779484
0.794872	-0.01011
0.076923	0.779484
0	0.864084
0.205128	0.638485
0.128205	0.723084
0.230769	0.610285
0.025641	0.835884
0.230769	0.610285
0.051282	0.807684
0.025641	0.835884
0.25641	0.582085
0.282051	0.553885
0.307692	0.525686
0.076923	0.779484
0.102564	0.751284
0.128205	0.723084
0.153846	0.694885
0.076923	0.779484
0.205128	0.638485
0.025641	0.835884
1	-0.23571
0.076923	0.779484
0.179487	0.666685
0.538462	0.271887
0.205128	0.638485
0.230769	0.610285
0.358974	0.469286
0.076923	0.779484
0.128205	0.723084
0.487179	0.328287
0.153846	0.694885
0.076923	0.779484
0.025641	0.835884
0.384615	0.441086
0.051282	0.807684
0.025641	0.835884
0.333333	0.497486
0.076923	0.779484
0.25641	0.582085

CLI	HS
0.789474	0.838566
0.473684	0.598487

0.210526	0.398421
0.631579	0.718526
0.526316	0.6385
0.578947	0.678513
0.894737	0.918592
0.631579	0.718526
0.421053	0.558473
0.052632	0.278381
0.210526	0.398421
0.368421	0.51846
0.263158	0.438434
0.684211	0.758539
0.842105	0.878579
0.736842	0.798553
0.473684	0.598487
0.631579	0.718526
0.789474	0.838566
0.894737	0.918592
0.578947	0.678513
0.526316	0.6385
0.368421	0.51846
0.263158	0.438434
0.157895	0.358407
1	0.998619
0.736842	0.798553
0.789474	0.838566
0.368421	0.51846
0.736842	0.798553
0.684211	0.758539
0.526316	0.6385
0	0.238368
0.578947	0.678513
0.210526	0.398421
0.315789	0.478447
0.473684	0.598487
0.526316	0.6385
0.157895	0.358407
0.578947	0.678513
0.789474	0.838566
0.210526	0.398421
0.473684	0.598487
1	0.998619
0.368421	0.51846
0.105263	0.318394
0.789474	0.838566

0.578947	0.678513
0.210526	0.398421
0.684211	0.758539
0.315789	0.478447

HI	HS
0.885246	0.876119
0.819672	0.80179
0.42623	0.355818
0.754098	0.727461
0.803279	0.783208
0.852459	0.838954
0.934426	0.931865
0.918033	0.913283
0.606557	0.560222
0.278689	0.188578
0.442623	0.3744
0.721311	0.690297
0.327869	0.244324
0.688525	0.653132
0.967213	0.96903
0.622951	0.578804
0.770492	0.746043
0.557377	0.504475
0.836066	0.820372
0.786885	0.764626
0.852459	0.838954
0.868852	0.857537
0.737705	0.708879
0.52459	0.467311
0.409836	0.337235
1	1.00619
0.754098	0.727461
0.803279	0.783208
0.606557	0.560222
0.770492	0.746043
0.57377	0.523057
0.885246	0.876119
0	-0.12732
0.901639	0.894701
0.442623	0.3744
0.459016	0.392982
0.639344	0.597386
0.557377	0.504475

0.360656	0.281489
0.786885	0.764626
0.770492	0.746043
0.295082	0.20716
0.508197	0.448728
0.819672	0.80179
0.836066	0.820372
0.47541	0.411564
0.770492	0.746043
0.836066	0.820372
0.442623	0.3744
0.737705	0.708879
0.606557	0.560222