AUTOMATED CURVE FITTING USING REGRESSION ANALYSIS FOR FINDING THE BEST MODEL

AYON ROY

NAFISA TABASSUM

TAUSIF AL ZUBAYER

A THESIS SUBMITTED FOR
THE DEGREE OF BACHELOR OF SCIENCE IN
COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
MILITARY INSTITUTE OF SCIENCE AND TECHNOLOGY

2020

SUPERVISOR'S APPROVAL

This thesis paper titled **"AUTOMATED CURVE FITTING USING REGRESSION ANALYSIS FOR FINDING THE BEST MODEL"**, submitted by Ayon Roy, ID: 201714018, Nafisa Tabassum, ID: 201714042 and Tausif Al Zubayer, ID: 201714064 has been accepted as satisfactory in partial fulfillment of the requirements for the degree B.Sc. in Computer Science and Engineering in 2020.

Md. Abdus Sattar
Associate Professor
Department of Computer Science and Engineering
Military Institute of Science and Technology

DECLARATION

This is to certify that the work presented in this thesis paper, titled, “AUTOMATED CURVE FITTING USING REGRESSION ANALYSIS FOR FINDING THE BEST MODEL”, is the outcome of the investigation and research carried out by the following students under the supervision of Md. Abdus Sattar, Associate Professor, Department of Computer Science and Engineering, Military Institute of Science and Technology.

It is also declared that neither this thesis paper nor any part thereof has been submitted anywhere else for the award of any degree, diploma or other qualifications.

Ayon Roy
Roll: 201714018
14 March 2021

Nafisa Tabassum
Roll: 201714042
14 March 2021

Tausif Al Zubayer
Roll: 201714064
14 March 2021

ABSTRACT

Researchers often need to find relationship among one or more independent variables and the dependent variable. Moreover, data analysts need to find patterns in a dataset. Usually, they use regression analysis to accomplish these tasks. But the process of fitting a regression model to a dataset can often be time consuming and inefficient. To solve this issue, this thesis aims to develop an automated system for curve fitting by regression analysis. Researchers can upload a dataset into the system, split the dataset into training and test set, select relevant features and label from the dataset and the system will return the best fit linear regression model for that dataset. So researchers with limited technical knowledge will also be able to find the best fit linear regression model for a dataset using this automated system. This system automates the process of regression analysis and helps to find out relationships in a dataset.

ACKNOWLEDGEMENT

We are thankful to Almighty Allah for his blessings for the successful completion of our thesis. Our heartiest gratitude, profound indebtedness and deep respect go to our supervisor, Md. Abdus Sattar, Associate Professor, Department of Computer Science and Engineering, Military Institute of Science and Technology, for his constant supervision, affectionate guidance and great encouragement and motivation. His keen interest on the topic and valuable advices throughout the study was of great help in completing thesis. We are especially grateful to the Department of Computer Science and Engineering (CSE) of Military Institute of Science and Technology (MIST) for providing their all out support during the thesis work.

Finally, we would like to thank our families and our course mates for their appreciable assistance, patience and suggestions during the course of our thesis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATION

HTML : Hypertext Markup Language
CSS : Cascading Style Sheet
UI : User Interface
VO2 : Volume of Oxygen
N-EX : Non-Exercise

LIST OF SYMBOLS

n	: No. of data points in the dataset
m	: No. of columns or independent variables in the dataset
S_t	: The sum of squared residuals around the mean for the dependent variable
S_r	: The sum of squared residuals around the regression line
r^2	: R2 score of a regression model
$a_0, a_1, ..., a_m$	: Parameters of a regression model
$\frac{\partial S_r}{\partial a_i}$	: Partial derivative of S_r with respect to parameter a_i
y	: Dependent variable
$x_1, x_2, ..., x_m$	: Independent variables
$\ln$	: Natural logarithm
e	: Exponent (Euler's constant)
θ	: Phase shift of sinusoid function
c_1	: Amplitude of sinusoid function

CHAPTER 1
INTRODUCTION

The chapter firstly presents the background and the motivation of the thesis. Then the problem statement is described, which is followed by high level objectives of the thesis. After that, an overview of the methodology followed is discussed and thesis scope is presented. Finally the organization of the thesis is described.

1.1 Research Background

In this era of science, information is of utmost importance. Information drives scientific discoveries and information comes from data. The field of data analysis and data visualization has gained momentum in recent decades. Data visualization is necessary for finding patterns in data. Curve fitting is one of many techniques used in the field of Pattern recognition. Curve fitting is the process of constructing a mathematical function that has the best fit to a series of data points. Curve fitting examines the relationship between one or more predictors (independent variables) and a response variable (dependent variable), with the goal of defining a 'best fit' model to describe that relationship. This type of relationships or patterns can be found in almost all types of situations. So, in almost all fields of science, be it physics, chemistry, biology or medical science, curve fitting plays an important role for finding patterns in data, for finding outliers or for extrapolating the curve to make predictions.

Regression analysis is one of the methods that is used for curve fitting. Regression analysis is a statistical technique to determine the correlations between two or more variables having cause-effect relations, and to make predictions for the topic using the relation [1]. Currently regression models are being applied widely in Linguistics, Sociology and History [2]. Linear regression model is one of the most frequently applied statistical procedures in observational astronomy [3]. These models have also been widely used in geography [4]. Almost every discipline is making use of regression analysis.

Since curve fitting by regression analysis is done in every research field continuously, an automated system will be of great help to the researchers who want to get insight from the data. This insight can then be used for various purposes like: for future events prediction, for business related decision making, for finding patterns in the data, for various other research purposes etc.

1.2 Problem Statement

Researchers are frequently confronted with the task of deciding which of several equations or models describe their dataset best. To find out the “best fitting model” for a dataset, a program can be written which will test the dataset against different models and finally choose a model which gives the least error. Alternatively, a tool like MATLAB can also be used to find out the best fitting model. But these types of approaches are time consuming, require technical knowledge and are often repetitive.

Keeping these in mind, if an automated system can be developed where the user will upload the dataset and the system will return the best fitting model, then it will save time for the researchers and also people without any technical background will be able to find patterns or insight in the data. Therefore this thesis focuses on the implementation of an automated curve fitting system.

1.3 Thesis Objectives

The objective of this thesis is to design an automated system which will help researchers as well as people with no technical background fit a dataset to several regression models and find the best fitting regression model. The user will also be able to know how well each model fits the dataset and as a result it will also give the user insight about the dataset. So the system will be able to help the users find patterns in the dataset automatically, get insights and make future predictions from the dataset.

1.4 Methodological Overview

To conduct this thesis, the following steps were followed. Firstly, the objectives of the thesis were defined and the related works were studied to find out any gap in the existing systems.

Secondly, a detailed study was conducted which included required descriptions and definitions on how the system will be developed, what technologies will be used and how the interface will be designed etc.

Thirdly, a web application system was developed so that users can upload their dataset and the system returns the best fit regression model along with the accuracy score. Also, a library named ‘curfi.js’ was published alongside the web application, so that any developer can use the library to create more robust automated curve fitting web applications in the future [5].

Finally, the system was evaluated by uploading different datasets and checking the results and accuracy scores for the models returned by the system.

1.5 Thesis Scope

The overall thesis work is based on Curve fitting by regression analysis, which is a common task in many different disciplines like machine learning, data mining, business intelligence, predicting future behavior of a system etc. This thesis is focused on automating the task of regression analysis so that novice users as well as expert users can get benefit from the automated system by saving time and effort.

1.6 Thesis Organization

The overall thesis is based on automating the curve fitting process by regression analysis. An automated system has been designed and developed so that researchers can directly upload a dataset and find the 'best fit' linear regression model. The next chapters elaborate on the thesis work in a step by step manner.

Chapter 2 has a theoretical background on curve fitting and regression analysis is discussed. A literature review on the related works in this case is also done to find out how regression methods are used in various research fields and how automating this process will help the researchers.

Chapter 3, walks through the methodology, which includes discussion on the focused design and development of the automated system. It also discusses the design and development of the user interface and the publication of the library.

Chapter 4 discusses the evaluation objectives, the procedure followed for evaluation, an in depth view of data analysis is done and results are obtained.

Chapter 5 shows the overall thesis outcomes and implications. It also shows the limitations of the work done and also what can be done in future working on this concept.

CHAPTER 2
THEORETICAL BACKGROUND AND LITERATURE REVIEW

This chapter firstly describes basic theory behind curve fitting. Then regression analysis along with various linear regression models are described. Then the chapter briefly discusses the literature focusing on the scope of the thesis.

2.1 Curve Fitting

Curve fitting is a way to model or represent a dataset by assigning a 'best fit' function along the entire range. It is the process of approximating a closed form function from a given dataset [6]. It captures the trend in the data and allows making predictions of how the data series will behave in the future. The best fitting curve is defined here to be that which minimizes the maximum absolute deviation between the fitted curve and the given data [7].

Curve fitting is a preliminary activity to many techniques used to model and solve production problems such as simulation, predictive modelling, and statistical inference [6]. Data are often given for discrete values along a continuum. Curve fitting techniques can be applied to fulfil two types of requirements: 1) to obtain estimates at points between the discrete values and 2) to obtain a simplified version of a complicated function. When a model is fit to a dataset, a simplified function is obtained which represents the trend of that dataset. This simplified function can be used to estimate points in between the discrete values or can be used to make future predictions by extrapolating the curve.

There are two general approaches for curve fitting that are distinguished from each other on the basis of the amount of error associated with the dataset-

(a) **Least-squares regression** : When the dataset exhibits a significant degree of error or "noise," the basic strategy is to derive a single curve that represents the general trend of the dataset. One approach of this nature is called 'least-squares regression'.

(b) **Interpolation** : When these data are known to be very precise, the basic approach is to fit a curve or a series of curves that pass directly through each of the points. This approach is known as 'interpolation'.

The same dataset can be fit to various functions or models by either using the 'least-squares regression' method or 'interpolation' method. Which approach will be used usually depends on the nature of the dataset and the application of the fit. In some applications, linear models will fit well and in some other applications, linear interpolation or curvilinear models will fit well.

In engineering practices, two types of applications are generally encountered when fitting experimental data-

(a) **Trend analysis** : Trend analysis represents the process of using the pattern of these data to make predictions. Trend analysis may be used to predict or forecast values of the dependent variable. This can involve extrapolation beyond the limits of the observed data or interpolation within the range of the data.

(b) **Hypothesis testing** : A second engineering application of experimental curve fitting is hypothesis testing. Here, an existing mathematical model is compared with measured data. Often, alternative models are compared and the "best" one is selected on the basis of empirical observations.

From this it is apparent that curve fitting is useful in various fields of research. And for this reason, this thesis focuses on curve fitting and automates it.

2.2 Regression Analysis

The most old multivariate technique used in science was the least square method, which later in the nineteenth century, came to be known as Regression Analysis [8]. Generally, it is used to predict one variable, given the value of other variable/s.

Regression analysis is a statistical technique for determining the relationship between a single dependent (criterion) variable and one or more independent (predictor) variables [9]. Mathematically speaking, it is a method for estimating the mathematical relationship of Y in terms of X i.e $Y = f(X)$, where X can be a number of variables called covariates or predictors that results in the value of Y, which is called an outcome [8].

The purpose for using regression analysis can be widely divided into two points-

1. Prediction, in the field of machine learning.
2. Determining the relationship between the dependent and independent variable/s.

Regression analysis can be of various types. A few of them are discussed below-

Linear regression is the best-known and most easily understood form of regression analysis [10]. Linear regression is a specific method used for determining the relationship between an unknown or scalar parameter (the dependent variable) and a known parameter (the independent variable). Here, a single unknown outcome variable, y and a single known predictor variable, x, forms this relationship [8].

Multiple linear regression is a method used for determining the relationship between a single unknown outcome variable y, and multiple known predictor parameters, $x_1, x_2, \ldots, x_m$ where, $m =$ an integer value [11]. It can be written as Eq. (2.1).

$$y = a_0 + a_1 x_1 + a_2 x_2 + \cdots + a_m x_m \tag{2.1}$$

where $a_i =$ regression coefficients.

Polynomial linear regression is the least square regression to fit the data to a higher order polynomial [11]. It is required because, often for higher degrees of equation, a straight line does not suffice the data points. It can be written as Eq. (2.2).

$$y = a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m \tag{2.2}$$

Logarithmic linear regression is used to handle the non-linear relationship that exists between the independent and dependent variables [12]. But, this makes the effective relationship non linear, while preserving the linear model. The equation can be written as Eq. (2.3).

$$y = a_0 + a_1 \ln x_1 + a_2 \ln x_2 + \cdots + a_m \ln x_m \tag{2.3}$$

Exponential regression models can be used for relationships among variables that are not inherently linear, but can be made linear after a transformation [13].The equation can be expressed as Eq. (2.4).

$$y = a_0 + a_1 e^{x_1} + a_2 e^{x_2} + \cdots + a_m e^{x_m} \tag{2.4}$$

Sinusoidal regression is a model that transforms a non-linear regression to a linear regression. This can be expressed as Eq. (2.5).

$$y = a_0 + c_1 \sin(x + \theta) \tag{2.5}$$

The briefly explained relations between the variables are all a form of linear regression analysis. There are several other forms of non-linear regression analysis, but our thesis focuses on the linear form of relationships between dependent and independent variables.

2.3 Related Works

So far the basic theoretical description of curve fitting and various types of regression analysis are given. From this discussion, it is apparent that regression analysis is an important method in the fields of pattern recognition, data mining, machine learning etc. Now some previous works will be discussed to understand the premise of this thesis work.

Nelson Fumo et al. has described how to model the residential energy consumption using linear regression [14]. In this study, Nelson Fumo et al. has conducted a simple linear regression, a multiple linear regression as well as a quadratic linear regression analysis to model the energy consumption of a house with daily and hourly data. Linear regression model has been proved to be useful in this case because of its simplicity and reasonable accuracy. The researchers then compared the accuracy of the models using various parameters such as 'coefficient of determination' and 'root mean square error'. The researchers anticipated that in the future, individual households will be able to develop their own energy consumption models by using the data coming from the smart meters. The regression analysis helped the researchers get this insight.

Abdulsalam Olaniyi et al. has demonstrated how to discover knowledge from databases [15]. Organizations collect massive amounts of data and store it in data warehouses. But most of these data remain unused and organizations don't often know how to extract useful information from massive amounts of data. So the researchers in this study applied regression analysis for stock price prediction to demonstrate that regression analysis is suitable for knowledge discovery from data. The values of certain variables were extracted from the database which were used by the regression model to predict the future values of other variables. This study has revealed how regression analysis can be proved valuable for discovering patterns in the data to make robust future predictions.

D.I. Bradshaw et al. has developed a multiple linear regression model to predict VO2max based on non-exercise (N-EX) data [16].VO2max is a measure of the maximum oxygen uptake of a person before or after intense exercise. More oxygen uptake means a healthy condition of the cardiorespiratory system. So VO2max is one of the parameters to determine a person's health condition. In this study, there were 100 participants, aged 18 - 65 years old. They were given a maximal graded exercise test to assess VO2max. The data were collected just before the exercise test. The data included the participant's age, gender, body mass index, perceived functional ability to walk, jog, or run given distances, and current physical activity level. Then a multiple linear regression equation is generated considering the previously mentioned 5 parameters as the independent variables and VO2max as the dependent variable. The authors noted that this multiple linear regression model will be applicable for any sample of adults aged 18 - 65 years old. So this study provides a robust regression model to predict VO2max in adult men and women.

Christine Heim et al. used multiple linear regression analysis to predict the level of neuroendocrine stress response in adult women who have experienced childhood abuse in the past or major life events recently [17]. Studies have shown that childhood trauma, trauma in adulthood and major life events are responsible for depression and anxiety disorders in adults. These types of disorders are also correlated with corticotropin-releasing factor (CRF) system which is a hormonal system associated with the central nervous system of humans. This system is also known as neuroendocrine stress response system because studies have shown in the past that the CRF system is heavily influenced in the presence of depression and anxiety disorder. So in this study, the researchers decided to measure the neuroendocrine stress response in adult women with respect to some predictors like demographic variables, childhood trauma, adulthood trauma, major life events in the past year and daily hassles in the past month etc. Then the researchers used this data to develop a multiple linear regression model that can reveal which predictor is the most influential factor in neuroendocrine stress response. This study reveals that multiple linear regression models can be helpful in medical science research too.

So from all these studies, it is clear that linear regression models are used intensively in multiple disciplines including public health science, medical science, stock market prediction, psychology, energy sector etc.

2.4 Chapter Summary

After summarising the related work, a few concerns have come out from the literature survey. Almost all studies have implemented simple linear regression and multiple linear regression models. But the researchers have implemented these models from scratch. This process is repetitive, time consuming and takes a lot of effort. So an automated system for regression analysis is necessary for the researchers in many disciplines.

CHAPTER 3
METHODOLOGY

This chapter firstly describes how the system was designed. Then it focuses on the development of the system along with the algorithms implemented in the system. Then the chapter discusses the design and development of the UI. Lastly, the chapter focuses on the library which was published in npm.

3.1 Designing the System

A web application system is proposed to automate the curve fitting process and to find the best fit model [18]. The system will help the researchers with little technical knowledge be able to easily find the best model that fits a dataset. The proposed web application system contains all the features needed for a user to navigate the system, select required features and target label from the dataset and find out which model fits the data best, the accuracy of that best fit model and also what other models might fit the dataset.

3.2 Developing the System

The system takes in a dataset as input from the user in csv(Comma Separated Values) format. The user then also selects the features and target label associated with that dataset for fitting the model. The system then converts that csv file and turns it into a two dimensional array like data-structure for calculation on the data points more conveniently. This converted dataset is then split into two datasets- 1) Training dataset, 2) Test dataset. The train-test split percentage can be set by the user from the user interface. The training dataset is then fit to the six different regression models : The simple linear regression model, the multiple linear regression model, the polynomial linear regression model, the logarithmic linear regression model, the exponential linear regression model and the sinusoidal regression model. The process of fitting the dataset with these six regression models is discussed in the following subsections.

3.2.1 Simple Linear Regression

The simple linear regression model is a straight line fit to a dataset. In simple linear regression there is only one dependent variable(y) and one independent variable(x). The mathematical expression of this model is shown in Eq. (3.1).

$$y = a_0 + a_1 x + error \tag{3.1}$$

Where a_0 and a_1 are the coefficients representing the intercept and slope respectively. Here $error$ is called residual which is the difference between the model and data points.

$$error = y - a_0 - a_1 x \tag{3.2}$$

The strategy for finding 'best fit' is to minimize the sum of the squared residuals for each data point which is described in Eq. (3.3) [11].

$$\sum_{i=0}^{n} {error_i}^2 = \sum_{i=0}^{n} (y_i - a_0 - a_1 x_i)^2 \tag{3.3}$$

where $n =$ number of data points.
The sum of squared residuals can be denoted by S_r. To minimize the error function, the derivative of S_r with respect to the parameters a_0 and a_1 can be derived respectively [11].

$$\frac{\partial S_r}{\partial a_0} = -2 \sum (y_i - a_0 - a_1 x_i) \tag{3.4}$$

$$\frac{\partial S_r}{\partial a_1} = -2 \sum [(y_i - a_0 - a_1 x_i) x_i] \tag{3.5}$$

After obtaining these two equations, we set them equal to 0, and then we will get Eqs. (3.6) and (3.7).

$$\sum y_i - \sum a_0 - \sum a_1 x_i = 0 \tag{3.6}$$

$$\sum y_i x_i - \sum a_0 x_i - \sum a_1 {x_i}^2 = 0 \tag{3.7}$$

By solving these two equations, we can derive a_0 and a_1 and we can use these two values to build our linear model.

$$a_0 = \overline{y} - a_1 \overline{x} \tag{3.8}$$

$$a_1 = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{n \sum {x_i}^2 - (\sum x_i)^2} \tag{3.9}$$

a_0 and a_1 were the two parameters for the simple linear regression model whose 'best values' will be calculated from the above two equations. These two values will give us the 'best fit' model which will closely follow the trend of the dataset.

The process to find the a_0 and a_1 is similar to multiple linear regression which is discussed in the next subsection.

3.2.2 Multiple Linear Regression

Multiple linear regression model is quite similar to a simple linear regression model. The main difference between these models is the number of features or independent variables. Simple linear regression model represents the relationship between one feature and one dependent variable. But multiple linear regression is a mathematical technique used to model the relationship between multiple independent predictor variables and a single dependent outcome variable [19].

So simple linear regression incorporates only one feature x but multiple linear regression incorporates multiple features $x_1, x_2, x_3, \ldots, x_n$ etc. The equation for multiple linear regression model looks like the following Eq. (3.10).

$$y = a_0 + a_1 x_1 + a_2 x_2 + error \tag{3.10}$$

Here only two variables x_1 and x_2 for this model are considered but multiple linear regression models can incorporate more than two variables. The parameters are denoted by a_0, a_1 and a_2. The goal is to find out the best values for these parameters to find the 'best fit' model for a given dataset.

The procedure for finding the best values for the parameters is quite the same as a simple linear regression model discussed in the previous subsection. So at first we have to know the equation of sum of squared residuals as shown in Eq. (3.11).

$$S_r = \sum_{i=1}^{n} (y_i - a_0 - a_1 x_{1i} - a_2 x_{2i})^2 \tag{3.11}$$

Then we have to take the derivative of S_r with respect to each of the parameters $a_0, a1$ and $a2$.

$$\frac{\partial S_r}{\partial a_0} = -2 \sum (y_i - a_0 - a_1 x_{1i} - a_2 x_{2i}) \tag{3.12}$$

$$\frac{\partial S_r}{\partial a_1} = -2 \sum x_{1i}(y_i - a_0 - a_1 x_{1i} - a_2 x_{2i}) \tag{3.13}$$

$$\frac{\partial S_r}{\partial a_2} = -2 \sum x_{2i}(y_i - a_0 - a_1 x_{1i} - a_2 x_{2i}) \tag{3.14}$$

After finding out the derivatives, we will set these equations equal to 0. So we will get 3 linear equations. We can solve this system of linear equations to find out the best values of the parameters. These values will give us the multiple linear regression model which will 'best fit' the data.

We can also express this system of linear equations in matrix notation like Eq. (3.15).

$$\begin{bmatrix} n & \sum x_{1i} & \sum x_{2i} \\ \sum x_{1i} & \sum {x_{1i}}^2 & \sum x_{1i}x_{2i} \\ \sum x_{2i} & \sum x_{1i}x_{2i} & \sum {x_{2i}}^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum x_{1i}y_i \\ \sum x_{2i}y_i \end{bmatrix} \tag{3.15}$$

If we consider $A = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}$ and $Z = \begin{bmatrix} n & \sum x_{1i} & \sum x_{2i} \\ \sum x_{1i} & \sum {x_{1i}}^2 & \sum x_{1i}x_{2i} \\ \sum x_{2i} & \sum x_{1i}x_{2i} & \sum {x_{2i}}^2 \end{bmatrix}$ and $Y = \begin{bmatrix} \sum y_i \\ \sum x_{1i}y_i \\ \sum x_{2i}y_i \end{bmatrix}$

Then Eq. (3.15) becomes like Eq. (3.16).

$$Z \cdot A = Y \tag{3.16}$$

If we bring Z to right side, essentially deriving its inverse matrix, and multiplying this inverse matrix with the matrix Y , then we can find the matrix A in the following way.

$$A = Z^{-1} \cdot Y \tag{3.17}$$

This equation also satisfies for an m dimensional multiple linear regression where m is the order.

$$y = a_0 + a_1x_1 + a_2x_2 + \cdots + a_mx_m + error \tag{3.18}$$

Now for finding the matrix A, we need to compute two matrices Z and Y. The algorithm described in Appendix A.2 is used for finding the Z and Y matrices and then finding the A matrix from Z and Y. For this algorithm, 1's must be stored in $x0$.

With this algorithm we can find the matrix A which is a column matrix of $a_0, a_1, \ldots, a_m$, model parameters for multiple linear regression. These model parameters provide relationships for each independent variable with the dependent variable, essentially providing the pattern of the dataset.

3.2.3 Polynomial Linear Regression

The polynomial linear regression model is similar to a simple linear regression model because of the fact that it has one independent variable and one dependent variable. But the main difference is that the polynomial linear regression model uses a higher order polynomial equation where the independent variable x has higher order power. For example, a second order polynomial linear regression model is denoted by the Eq. (3.19).

$$y = a_0 + a_1 x + a_2 x^2 + error \tag{3.19}$$

Here, the second order polynomial linear regression model is described but the polynomial linear regression model can be upto nth order where n is any integer. The parameters of this model are a_0, a_1 and a_2. The goal is to find the best values of these parameters which will give us the 'best fit' model for a given dataset.

The procedure for finding the best values for the parameters is quite the same as a simple linear regression model or a multiple linear regression model discussed in previous subsections. So first the sum of squared residuals will be calculated by the Eq. (3.20).

$$S_r = \sum_{i=1}^{n} (y_i - a_0 - a_1 x_i - a_2 {x_i}^2)^2 \tag{3.20}$$

Then the derivative of S_r will be calculated with respect to each of the parameters a_0, a_1 and a_2.

$$\frac{\partial S_r}{\partial a_0} = -2 \sum (y_i - a_0 - a_1 x_i - a_2 {x_i}^2) \tag{3.21}$$
$$\frac{\partial S_r}{\partial a_1} = -2 \sum x_i (y_i - a_0 - a_1 x_i - a_2 {x_i}^2) \tag{3.22}$$
$$\frac{\partial S_r}{\partial a_2} = -2 \sum {x_i}^2 (y_i - a_0 - a_1 x_i - a_2 {x_i}^2) \tag{3.23}$$

After finding out the derivatives, these three equations will be set equal to zero. So a system of three linear equations Eqs. (3.24) to (3.26) will be generated.

$$(n)a_0 + (\sum x_i)a_1 + (\sum {x_i}^2)a_2 = \sum y_i \tag{3.24}$$
$$(\sum x_i)a_0 + (\sum {x_i}^2)a_1 + (\sum {x_i}^3)a_2 = \sum x_i y_i \tag{3.25}$$
$$(\sum {x_i}^2)a_0 + (\sum {x_i}^3)a_1 + (\sum {x_i}^4)a_2 = \sum {x_i}^2 y_i \tag{3.26}$$

This system of linear equations has to be solved to find out the best values of the three parameters. The best values of the three parameters will give us the polynomial linear regression model which will 'best fit' the dataset.

This system of linear equations can also be expressed in matrix notation like Eq. (3.27).

$$\begin{bmatrix} n & \sum x_i & \sum {x_i}^2 \\ \sum x_i & \sum {x_i}^2 & \sum {x_i}^3 \\ \sum {x_i}^2 & \sum {x_i}^3 & \sum {x_i}^4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum x_i y_i \\ \sum {x_i}^2 y_i \end{bmatrix} \tag{3.27}$$

If we consider $A = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}$ and $Z = \begin{bmatrix} n & \sum x_i & \sum {x_i}^2 \\ \sum x_i & \sum {x_i}^2 & \sum {x_i}^3 \\ \sum {x_i}^2 & \sum {x_i}^3 & \sum {x_i}^4 \end{bmatrix}$ and $Y = \begin{bmatrix} \sum y_i \\ \sum x_i y_i \\ \sum {x_i}^2 y_i \end{bmatrix}$

Then Eq. (3.27) becomes like Eq. (3.28).

$$Z \cdot A = Y \tag{3.28}$$

So by taking the Z on the right side, the following Eq. (3.29) is derived.

$$A = Z^{-1} \cdot Y \tag{3.29}$$

This equation can be easily extended for an m dimensional polynomial linear regression equation.

$$y = a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m + error \tag{3.30}$$

Now for finding the matrix A, we need to compute two matrices Z and Y. The algorithm described in Appendix A.3 is used for finding the Z and Y matrices and then finding the A matrix from Z and Y. For this algorithm, 1's must be stored in $x0$.

With this algorithm we can find the matrix A which is a column matrix of $a_0, a_1, \ldots, a_m$, model parameters for polynomial linear regression. These model parameters provide a relationship between the independent variable with the dependent variable, essentially providing the pattern of the dataset.

3.2.4 Logarithmic Linear Regression

Logarithmic linear regression model is quite similar to the previously described linear regression models. This regression model can incorporate multiple independent variables and one dependent variable. But the only difference is that the independent variables are incorporated into this model as the natural logarithm of x where x is an independent variable. The equation of this model considering only two independent variables x_1 and x_2 is shown in Eq. (3.31).

$$y = a_0 + a_1 \ln(x_1) + a_2 \ln(x_2) + error \tag{3.31}$$

The equation of the sum of squared residuals is shown in Eq. (3.32).

$$S_r = \sum_{i=1}^{n} (y_i - a_0 - a_1 \ln(x_{1i}) - a_2 \ln(x_{2i}))^2 \tag{3.32}$$

After taking the derivatives of S_r with respect to a_0, a_1 and a_2, the following Eqs. (3.33) to (3.35) can be written.

$$\frac{\partial S_r}{\partial a_0} = -2 \sum (y_i - a_0 - a_1 \ln(x_{1i}) - a_2 \ln(x_{2i})) \tag{3.33}$$

$$\frac{\partial S_r}{\partial a_1} = -2 \sum \ln(x_{1i})(y_i - a_0 - a_1 \ln(x_{1i}) - a_2 \ln(x_{2i})) \tag{3.34}$$

$$\frac{\partial S_r}{\partial a_2} = -2 \sum \ln(x_{2i})(y_i - a_0 - a_1 \ln(x_{1i}) - a_2 \ln(x_{2i})) \tag{3.35}$$

Setting the above equations equal to zero gives us a system of 3 linear equations. This system of linear equations must be solved to get the best values of a_0, a_1 and a_2.

$$(n)a_0 + (\sum \ln(x_{1i}))a_1 + (\sum \ln(x_{2i}))a_2 = \sum y_i \tag{3.36}$$

$$(\sum \ln(x_{1i}))a_0 + (\sum (\ln(x_{1i}))^2)a_1 + (\sum (\ln(x_{1i}))(\ln(x_{2i})))a_2 = \sum (\ln(x_{1i}))y_i \tag{3.37}$$

$$(\sum \ln(x_{2i}))a_0 + (\sum (\ln(x_{1i}))(\ln(x_{2i})))a_1 + (\sum (\ln(x_{2i}))^2)a_2 = \sum (\ln(x_{2i}))y_i \tag{3.38}$$

This system of linear equations can be expressed with the matrix notation like Eq. (3.39).

$$\begin{bmatrix} n & \sum \ln(x_{1i}) & \sum \ln(x_{2i}) \\ \sum \ln(x_{1i}) & \sum (\ln(x_{1i}))^2 & \sum \ln(x_{1i}) \ln(x_{2i}) \\ \sum \ln(x_{2i}) & \sum \ln(x_{1i}) \ln(x_{2i}) & \sum (\ln(x_{2i}))^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum \ln(x_{1i}) y_i \\ \sum \ln(x_{2i}) y_i \end{bmatrix} \tag{3.39}$$

If we consider $A = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}$ and $Z = \begin{bmatrix} n & \sum \ln(x_{1i}) & \sum \ln(x_{2i}) \\ \sum \ln(x_{1i}) & \sum (\ln(x_{1i}))^2 & \sum \ln(x_{1i}) \ln(x_{2i}) \\ \sum \ln(x_{2i}) & \sum \ln(x_{1i}) \ln(x_{2i}) & \sum (\ln(x_{2i}))^2 \end{bmatrix}$ and $Y = \begin{bmatrix} \sum y_i \\ \sum \ln(x_{1i}) y_i \\ \sum \ln(x_{2i}) y_i \end{bmatrix}$

Then Eq. (3.39) becomes like Eq. (3.40).

$$Z \cdot A = Y \tag{3.40}$$

So by taking the Z on the right side, the following Eq. (3.41) is derived.

$$A = Z^{-1} \cdot Y \tag{3.41}$$

This process can be extended upto m independent variables where m is any integer.

$$y = a_0 + a_1 \ln(x_1) + a_2 \ln(x_2) + \cdots + a_m \ln(x_m) + error \tag{3.42}$$

Now for finding the matrix A, we need to compute two matrices Z and Y. The algorithm described in Appendix A.4 is used for finding the Z and Y matrices and then finding the A matrix from Z and Y. For this algorithm, 1's must be stored in $x0$.

With this we can find the matrix A which is a column matrix of $a_0, a_1, \ldots, a_m$, model parameters for logarithmic linear regression. These model parameters provide relationships for each independent variable with the dependent variable, essentially providing the pattern of the dataset.

3.2.5 Exponential Linear Regression

Exponential linear regression model is similar to other linear regression models discussed so far. This regression model can incorporate multiple independent variables and one dependent variable. But the only difference from other models is that the independent variables are used in the model as the power of the exponent e. The equation of this model is Eq. (3.43).

$$y = a_0 + a_1 e^{x_1} + a_2 e^{x_2} + error \tag{3.43}$$

The sum of squared residuals is shown in Eq. (3.44).

$$S_r = \sum_{i=1}^{n} (y_i - a_0 - a_1 e^{x_{1i}} - a_2 e^{x_{2i}})^2 \tag{3.44}$$

After taking the derivatives of S_r with respect to the parameters a_0, a_1 and a_2, Eqs. (3.45) to (3.47) are derived.

$$\frac{\partial S_r}{\partial a_0} = -2\sum(y_i - a_0 - a_1 e^{x_{1i}} - a_2 e^{x_{2i}}) \tag{3.45}$$
$$\frac{\partial S_r}{\partial a_1} = -2\sum e^{x_{1i}}(y_i - a_0 - a_1 e^{x_{1i}} - a_2 e^{x_{2i}}) \tag{3.46}$$
$$\frac{\partial S_r}{\partial a_2} = -2\sum e^{x_{2i}}(y_i - a_0 - a_1 e^{x_{1i}} - a_2 e^{x_{2i}}) \tag{3.47}$$

Setting the above equations equal to zero gives us a system of 3 linear equations. This system of linear equations Eqs. (3.48) to (3.50) must be solved to get the best values of a_0, a_1 and $a2$.

$$na_0 + (\sum e^{x_{1i}})a_1 + (\sum e^{x_{2i}})a_2 = \sum y_i \tag{3.48}$$
$$(\sum e^{x_{1i}})a_0 + (\sum e^{2x_{1i}})a_1 + (\sum e^{x_{1i}+x_{2i}})a_2 = \sum e^{x_{1i}} y_i \tag{3.49}$$
$$(\sum e^{x_{2i}})a_0 + (\sum e^{x_{1i}+x_{2i}})a_1 + (\sum e^{2x_{2i}})a_2 = \sum e^{x_{2i}} y_i \tag{3.50}$$

This system of linear equations can be expressed with the matrix notation like Eq. (3.51).

$$\begin{bmatrix} n & \sum e^{x_{1i}} & \sum e^{x_{2i}} \\ \sum e^{x_{1i}} & \sum e^{2x_{1i}} & \sum e^{x_{1i}+x_{2i}} \\ \sum e^{x_{2i}} & \sum e^{x_{1i}+x_{2i}} & \sum e^{2x_{2i}} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum e^{x_{1i}} y_i \\ \sum e^{x_{2i}} y_i \end{bmatrix} \tag{3.51}$$

If we consider $A = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}$ and $Z = \begin{bmatrix} n & \sum e^{x_{1i}} & \sum e^{x_{2i}} \\ \sum e^{x_{1i}} & \sum e^{2x_{1i}} & \sum e^{x_{1i}+x_{2i}} \\ \sum e^{x_{2i}} & \sum e^{x_{1i}+x_{2i}} & \sum e^{2x_{2i}} \end{bmatrix}$ and $Y = \begin{bmatrix} \sum y_i \\ \sum e^{x_{1i}} y_i \\ \sum e^{x_{2i}} y_i \end{bmatrix}$

Then Eq. (3.51) becomes Eq. (3.52).

$$Z \cdot A = Y \tag{3.52}$$

So by taking the Z on the right side, the following Eq. (3.53) is derived-

$$A = Z^{-1} \cdot Y \tag{3.53}$$

This process can be extended upto m independent variables where m is any integer. Then the equation for exponential linear regression becomes like Eq. (3.54).

$$y = a_0 + a_1 e^{x_1} + a_2 e^{x_2} + a_3 e^{x_3} + \cdots + a_m e^{x_m} + error \tag{3.54}$$

Now for finding the matrix A, we need to compute two matrices Z and Y. The algorithm described in Appendix A.5 is used for finding the Z and Y matrices and then finding the matrix A from Z and Y. For this algorithm, 1's must be stored in $x0$.

With this we can find the matrix A which is a column matrix of $a_0, a_1, \ldots, a_n$, model parameters for exponential linear regression. These model parameters provide relationships for each independent variable with the dependent variable, essentially providing the pattern of the dataset.

3.2.6 Sinusoidal Regression

The sine and cosine functions are periodic functions and they can be used to construct regression models which are called "Sinusoidal regression" models. To construct these models, both sine and cosine functions can be used; there is no clear-cut convention for choosing either function and the results will be identical for both functions. The regression model equation for sine function is expressed as Eq. (3.55).

$$y = a_0 + c_1 \sin{(x + \theta)} + error \tag{3.55}$$

In the Eq. (3.55), x is the independent variable and y is the dependent variable. The parameters required to build the sinusoidal model are a_0, c_1 and θ. a_0 is the mean value which denotes the average height of the sinusoidal function above the x-axis. c_1 is the amplitude of the sine wave which denotes the height of oscillation. Finally, θ is called the phase shift which denotes the extent of shift of the wave horizontally.

But the Eq. (3.55) has non-linear characteristic because of the θ term. So Eq. (3.55) has to be converted in the form of a linear regression model like the previously described models. To achieve the linear regression form, the following trigonometric identity Eq. (3.56) can be applied.

$$sin(A + B) = \sin A \cos B + \cos A \sin B \tag{3.56}$$

So after applying this identity, the $c_1 \sin{(x + \theta)}$ part will be converted as described in Eq. (3.57).

$$\begin{aligned} c_1 \sin{(x + \theta)} &= c_1(\sin x \cos \theta + \cos x \sin \theta) \\ &= a_1 \sin x + a_2 \cos x \end{aligned} \tag{3.57}$$

where $a_1 = c_1 \cos \theta$ and $a_2 = c_1 \sin \theta$

So the final equation in the linear regression model form will look like Eq. (3.58).

$$y = a_0 + a_1 \sin x + a_2 \cos x + error \tag{3.58}$$

So now the equation Eq. (3.58) has 3 parameters a_0, a_1 and a_2. But the initial non-linear

Eq. (3.55) had 3 parameters a_0, c_1 and θ. We can find out c_1 and θ using a_1 and a_2 as described in Eq. (3.59) and Eq. (3.60).

$$c_1 = \sqrt{{a_1}^2 + {a_2}^2} \tag{3.59}$$

$$\theta = \tan^{-1}\frac{a_2}{a_1} \tag{3.60}$$

So after linearizing the nonlinear equation Eq. (3.55), we have got 3 parameters a_0, a_1 and a_2 and we can determine the original equation's parameters c_1 and θ from a_1 and a_2 using the above two formulas Eq. (3.59) and Eq. (3.60).

But the best values of the 3 parameters a_0, a_1 and a_2 must be determined at first. These parameters are of a linear regression model. So the method to determine the best values of these parameters is exactly the same as previously described linear regression models.

So, at first, the equation of the sum of squared residuals is shown in Eq. (3.61).

$$S_r = \sum_{i=1}^{n}(y_i - a_0 - a_1 \sin x_i - a_2 \cos x_i)^2 \tag{3.61}$$

Next, the derivatives of the sum of squared residuals will be derived with respect to each parameter.

$$\frac{\partial S_r}{\partial a_0} = -2\sum(y_i - a_0 - a_1 \sin x_i - a_2 \cos x_i) \tag{3.62}$$

$$\frac{\partial S_r}{\partial a_1} = -2\sum \sin x_i(y_i - a_0 - a_1 \sin x_i - a_2 \cos x_i) \tag{3.63}$$

$$\frac{\partial S_r}{\partial a_2} = -2\sum \cos x_i(y_i - a_0 - a_1 \sin x_i - a_2 \cos x_i) \tag{3.64}$$

After setting the above Eqs. (3.62) to (3.64) equal to zero, we will get a system of 3 linear equations Eqs. (3.65) to (3.67).

$$na_0 + (\sum \sin x_i)a_1 + (\sum \cos x_i)a_2 = \sum y_i \tag{3.65}$$

$$(\sum \sin x_i)a_0 + (\sum \sin^2 x_i)a_1 + (\sum(\sin x_i)(\cos x_i))a_2 = \sum(\sin x_i)y_i \tag{3.66}$$

$$(\sum \cos x_i)a_0 + (\sum(\sin x_i)(\cos x_i))a_1 + (\sum \cos^2 x_i)a_2 = \sum(\cos x_i)y_i \tag{3.67}$$

This system of linear equations Eqs. (3.65) to (3.67) can be expressed in the matrix notation like Eq. (3.68).

$$\begin{bmatrix} n & \sum \sin x_i & \sum \cos x_i \\ \sum \sin x_i & \sum \sin^2 x_i & \sum(\sin x_i)(\cos x_i) \\ \sum \cos x_i & \sum(\sin x_i)(\cos x_i) & \sum \cos^2 x_i \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum(\sin x_i)y_i \\ \sum(\cos x_i)y_i \end{bmatrix} \tag{3.68}$$

If we consider $A = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}$ and $Z = \begin{bmatrix} n & \sum \sin x_i & \sum \cos x_i \\ \sum \sin x_i & \sum \sin^2 x_i & \sum (\sin x_i)(\cos x_i) \\ \sum \cos x_i & \sum (\sin x_i)(\cos x_i) & \sum \cos^2 x_i \end{bmatrix}$ and $Y = \begin{bmatrix} \sum y_i \\ \sum (\sin x_i) y_i \\ \sum (\cos x_i) y_i \end{bmatrix}$

Then Eq. (3.68) becomes Eq. (3.69).

$$Z \cdot A = Y \tag{3.69}$$

So by taking the Z on the right side, Eq. (3.70) is derived.

$$A = Z^{-1} \cdot Y \tag{3.70}$$

The matrix A is a column matrix which contains the parameters a_0, a_1 and a_2. So the Eq. (3.70) will give the best values of these 3 parameters. Then from the parameters a_1 and a_2 the parameters c_1 and θ of the original equation Eq. (3.55) can be derived by applying the formulas previously described in Eq. (3.59) and Eq. (3.60). So in this way, by linearizing the original non-linear equation, it is possible to find out the 'best fit' sinusoidal regression model.

Now for finding the matrix A, we need to compute two matrices Z and Y. The algorithm described in Appendix A.6 is used for finding the Z and Y matrices and then finding the A matrix from Z and Y. For this algorithm, 1's must be stored in $x0$.

With this we can find the matrix A which is a column matrix of a_0, a_1 and a_2, model parameters for sinusoidal regression. These model parameters provide relationships of the independent variable with the dependent variable, essentially providing the pattern of the dataset.

3.2.7 Automated Model Selection

To automatically select the best model we first need to define a way to compare the models. The best way for comparison of several regression models is to calculate the accuracy of these models and then sort them to get the best fit model for a given dataset. For the accuracy we can compute the R squared (r^2) value for a regression model for a given dataset. To do this, we return to the original dataset and determine the total sum of the squares around the mean for the dependent variable (in our case, y). This quantity is designated S_t. This is the magnitude of the residual error associated with the dependent variable prior to regression [11]. After performing the regression, we can compute S_r, the sum of the squares of the residuals around the regression line. This characterizes the residual error that remains after the regression [11]. The difference between the two quantities, S_t and S_r, quantifies the improvement or error reduction due to conducting regression analysis rather than as an average value. Because the magnitude of this quantity is scale-dependent, the difference is normalized to S_t to give us Eq. (3.71).

$$r^2 = \frac{S_t - S_r}{S_t} \tag{3.71}$$

where r^2 is called the *coefficient of determination* and r is the *correlation coefficient*$(\sqrt{r^2})$. For a perfect fit, $S_r = 0$ and $r = r^2 = 1$, signifying that the model explains 100 percent of the variability of the data. For $r = r^2 = 0$, $S_r = S_t$ and the fit represents no improvement from training.

With the r^2 value we can compare the regression models for the training dataset. When r^2 is close to 1(or equals to 1) the model is good and represents significant improvement after training. We run the given dataset for all the regression models discussed in previous subsections and compute r^2 values for all these models for that dataset. After a sorting of these models based on their r^2 values, the regression model with the greater r^2 value is the best fit model. This best fit model is the model that best describes the relationship of the given dataset.

This automated process is run using the *AutoTrain* function that takes *trainX, trainY, testX, testY, highestOrder* as function parameters. Here, *trainX* is the training dataset's features list, namely the list of independent variable(x_i) values. *trainY* is the training dataset's label list, namely the list of dependent variable(y) values. Similarly *testX* and *testY* are the test dataset feature list and label list respectively. The *highestOrder* is the value for m which is the order and used by the polynomial regression model. The *AutoTrain* function first fit all the models using the *fitAllModels* function that takes *trainX, trainY, highestOrder* as parameters and returns a list of trained regression models along with their r^2 scores for both training and test datasets. *AutoTrain* function then sorts these models based on their r^2 scores and returns the best fit model.

This best fit model along with other models are then shown in the UI in sorted order to the user. The equations for these regression models along with the values of $a_0, a_1, a_2, \ldots, a_n$ are also shown in the UI so that users can easily get the insight and relationship from the dataset.

3.3 Design and Development of the User Interface(UI)

This section describes the design and development of the user interface(UI) for the web application system.

We designed a web based system (*CurFi*) so that users can use the automated regression tool without installing any softwares or programs. Any user can access the web application by using the address of the website from a browser [18].

The UI is designed in such a way that it takes a very little effort for any user to use the system. After going to the website, the user can upload a dataset using the 'Choose File' button. The user can also select the train-test split percentage in the UI. After that the user needs to choose the features and label from the UI for the automated regression analysis. After that the user can click the train button and then the system will automatically calculate and fit all the regression models. After fitting all the regression models and then computing and comparing these models the best fit model along with r^2 score and equation for the model is shown on the UI so that the user can better understand the relationships in the dataset. Some snapshots of UI are shown in Fig. 3.1 to Fig. 3.6.

Figure 3.1: UI for uploading the dataset

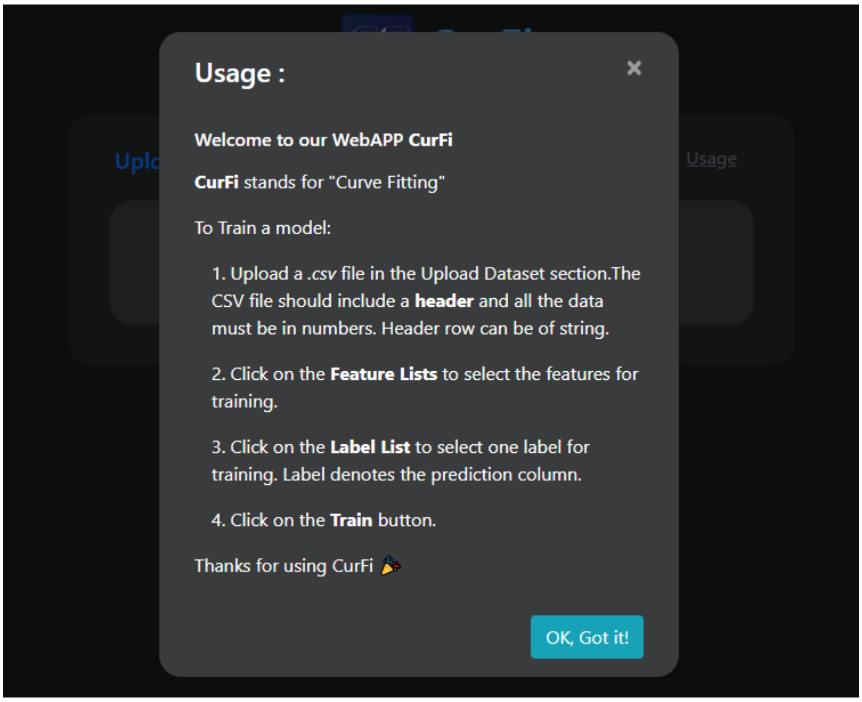

Figure 3.2: UI for describing the usage

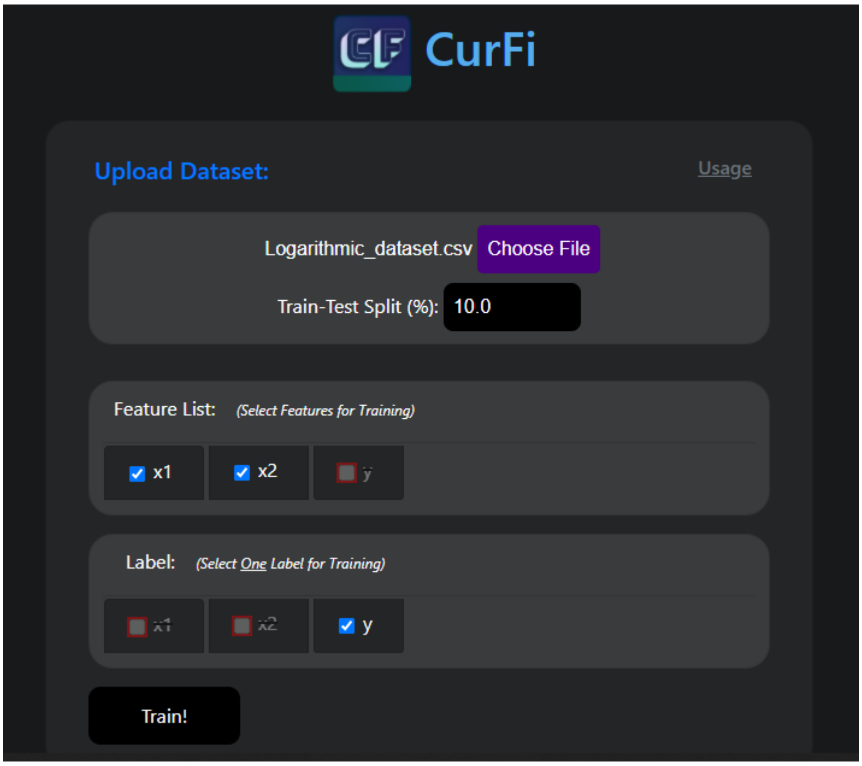

Figure 3.3: UI for selecting feature list and label

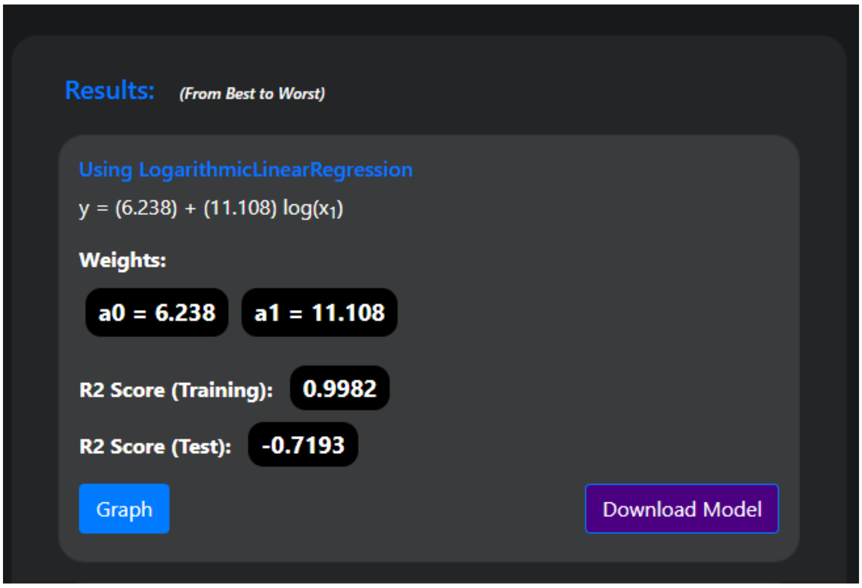

Figure 3.4: UI for showing the equation, parameters, and r^2 score of a regression model

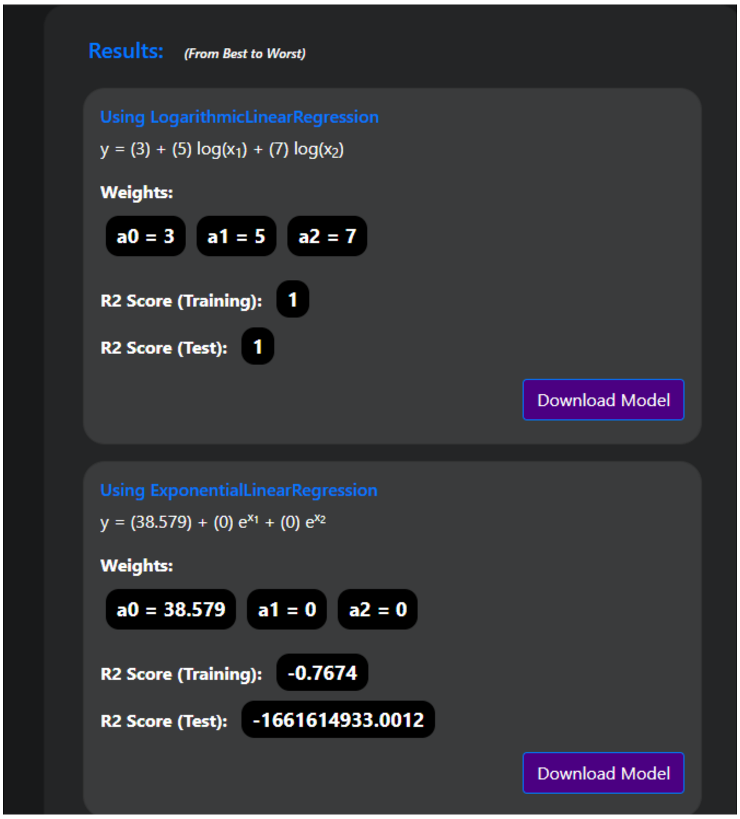

Figure 3.5: UI for describing the regression models sorted from best to worst

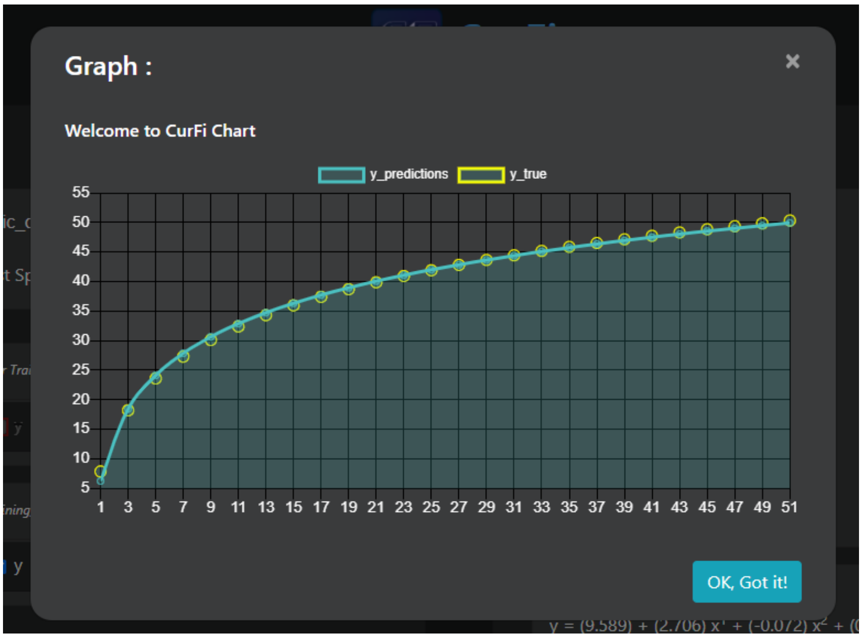

Figure 3.6: UI for showing the graphical visualization of a regression model

The UI is designed using the HTML markup language. And the site is designed using the CSS language. For interactions within the website like button clicks, dataset uploading etc Javascript is used. The website is then deployed to a cdn server that sends the site to any user who requests for the website.

3.4 Publishing the Library

Apart from developing a web application system we also published a library named '*curfi.js*' [5]. This library is written completely in javascript language. This is also the same library that we used for developing the web application system. We published the library on npm which is a javascript library distribution platform so that anyone can use our library to build more robust and automated systems for the web.

CHAPTER 4
EVALUATING THE SYSTEM

This chapter discusses the overall evaluation study. Firstly, the chapter highlights the objectives of the evaluation. Then the evaluation procedure is discussed. Finally, the data analysis and results are highlighted.

4.1 Evaluation Objectives

The developed system is evaluated based on three objectives. The first one is to evaluate the effectiveness of the system to find out an accurate relationship between one or more independent variables and a dependent variable. Then the ability of the system to find patterns in a real world dataset is also evaluated. Then the ability of finding the 'best fit' model automatically is evaluated.

4.2 Evaluation Procedure

For performing the evaluation of the system, some datasets were created according to the linear regression functions that are implemented in the system. These datasets were then uploaded in the system and trained to find out if the system is able to find the 'best fit' linear regression model accurately. At least one dataset was created for each of the linear regression models present in the system. In this way, it became clear that the system is able to find the 'best fit' model accurately.

Moreover a real world dataset named "Breast Cancer Wisconsin" is used to evaluate the system [20]. The dataset has the independent variables 'Sample code number', 'Clump Thickness', 'Uniformity of Cell Size', 'Uniformity of Cell Shape', 'Marginal Adhesion', 'Single Epithelial Cell Size', 'Bare Nuclei', 'Bland Chromatin', 'Normal Nucleoli' and 'Mitoses'. Among these independent variables, we have excluded 'sample code number' because it is just an ID number which will not contribute to the training process. Each independent variable except 'sample code number' has the domain 1 - 10. The dependent variable is 'class' which is a binary variable representing Breast Cancer type. It has values 2 and 4 where 2 indicates 'benign' and 4 indicates 'malignant'. After this dataset is uploaded and trained, the system was able to find the accurate value of the dependent variable with a very high r^2 score. This proves that the system is capable of finding patterns in real world dataset and making predictions as well.

4.3 Analysis and Results

Some datasets were prepared based on some predefined equations. Then these datasets were uploaded to the system for finding the accuracy of the system. In most of the cases, the system reproduced the equations with a very high r^2 score, almost close to 1. This means that the system can accurately find out the relationship between one or more independent variables and the dependent variable. The Table 4.1 shows the performance of the system for these prepared datasets.

Regression Type	**Proposed Relationship (Equation)**	**Equation by System**	**Training R2 Score**	**Test R2 Score**
Linear Regression	$2 + 3x$	$2 + 3x$	1.00	1.00
Multiple Linear Regression	$15 + 9x_1 - 6x_2$	$14.805 + 8.874x_1 - 5.842x_2$	0.99	0.987
Exponential Regression	$2 + 3e^{x_1} + 8e^{x_2}$	$2 + 3e^{x_1} + 8e^{x_2}$	1.00	1.00
Polynomial Regression	$3 + 4x + 8x^2$	$3 + 4x + 8x^2$	1.00	1.00
Logarithmic Regression	$-1.57+4.4 \ln(x_1)+ 3.6 \ln(x_2)$	$-1.569 + 4.36 \ln(x_1) + 3.59 \ln(x_2)$	0.975	0.957
Sinusoidal Regression	$3 + 4 \sin(x + 5)$	$3 + 4 \sin(x + 5)$	1.00	1.00

Table 4.1: Performance Analysis for the System

As discussed in the previous subsection, the system is also evaluated against a real world dataset named “Breast Cancer Wisconsin”. The dataset was uploaded and the system automatically found the relationship in that dataset. For predicting the cancer type, the logarithmic linear regression model represents the best relationship. The results are shown in Fig. 4.1 and Fig. 4.2.

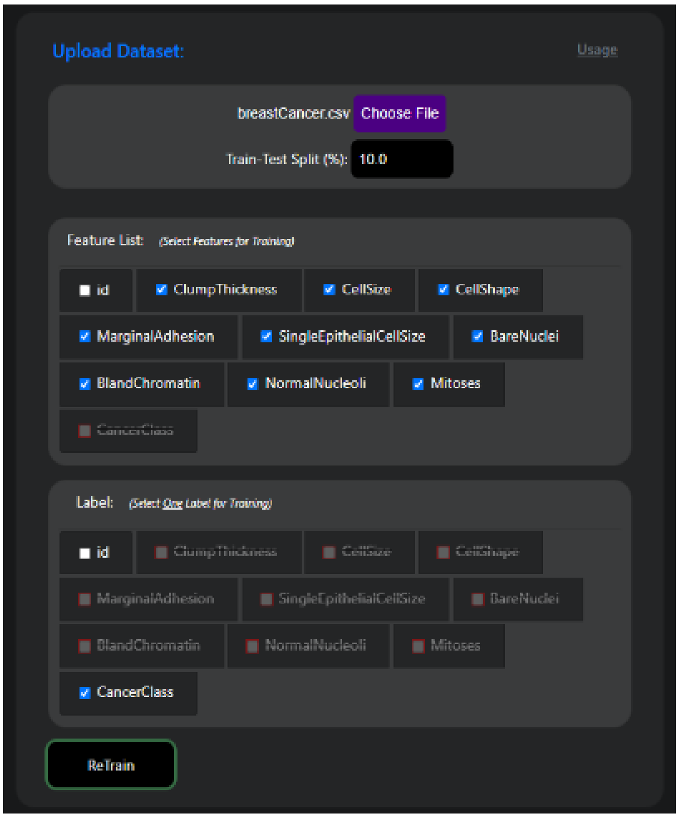

Figure 4.1: Selecting the feature list and label of the dataset

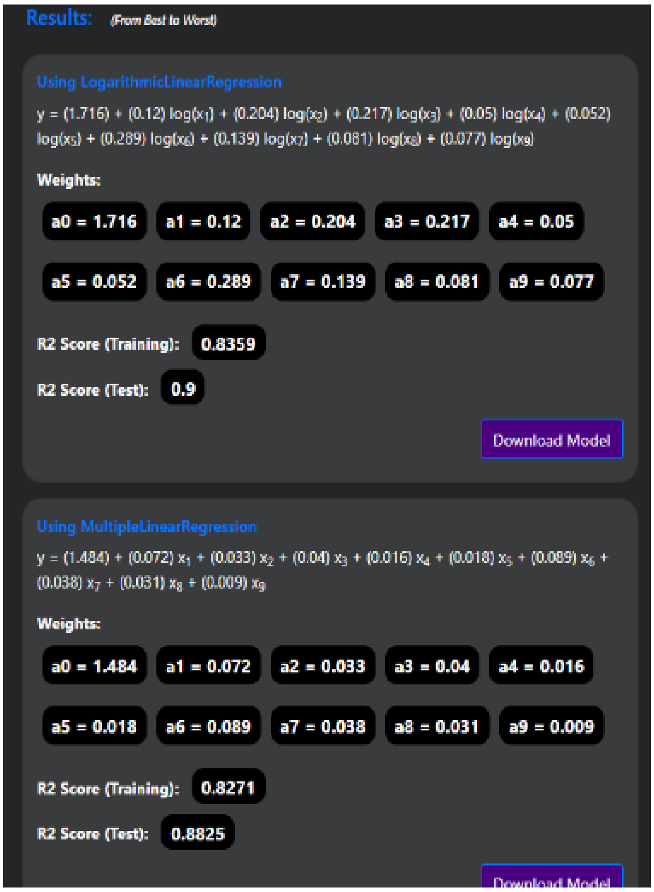

Figure 4.2: The resulting regression models sorted from best to worst

CHAPTER 5
DISCUSSION AND CONCLUSION

The chapter initially discusses the outcomes and the implications of the thesis. Then the limitations and the possible future work is also discussed.

5.1 Thesis Outcomes

In the modern world finding relationships among data points is a very important task for the researchers in various fields. Data analysts also need to find patterns in the dataset. For this they use regression analysis in most of the cases. But this process is repetitive and time consuming. This thesis focuses on solving this problem by automating the process of regression analysis.

The developed system discussed in this thesis, finds the relationship for a given dataset automatically by finding the 'best fit' model. The system finds the 'best fit' model by fitting a curve to the dataset. The best curve with the lowest residual error represents the 'best fit' model.

The developed system tries to find the relationship for a given dataset by fitting six different regression models to the dataset. The six regression models are : simple linear regression, multiple linear regression, polynomial linear regression, exponential linear regression, logarithmic linear regression and sinusoidal regression. The model with the highest r^2 score represents the best model that best defines the relationship among the data points.

The developed system shows the six models in the sorted order based on the r^2 score. For each model, the system shows the equation representing the relationship between independent variable and dependent variable. The system also shows a graphical plot of the data points and the curve representing that model.

In this way, the researchers can find and visualize the relationship in the dataset automatically and without any difficulty from the developed system.

5.2 Thesis Implications

This thesis focuses on automating regression analysis for curve fitting. The fields on which this thesis has implications are as follows :

Machine Learning: The automated system developed in this thesis has huge implications in the field of machine learning because machine learning focuses on finding patterns in datasets and making future predictions.

Data Mining: Data mining is about knowledge discovery from data. The field of data mining is mainly based on getting insights from data. So the developed automated system for regression analysis will be very useful in this field.

Data Visualization: The developed automated system for curve fitting by regression analysis can show graphical visualization of the best fit model when there is one independent variable and one dependent variable. So this system focuses on data visualization as well.

5.3 Thesis Limitations

The system can perfectly train datasets that have only numeric variables. Datasets with both numeric and categorical variables can not be trained directly but the categorical variables can easily be converted to numerical variables using different encoding techniques. So if a user wishes to train a dataset with one or more categorical variables, then encoding techniques can be used to convert the categorical variables to numerical ones. After that the user can successfully train the dataset with the developed system.

The system contains only the six linear regression models that are frequently used by researchers. But other types of regression models can be easily added to the system.

The system can provide graphical visualization of the models when the input dataset has only one independent and one dependent variable. But if the dataset contains multiple independent variables, then it is not possible to provide two dimensional plots of the models. Apart from the graphical visualization, equations for multiple independent variables can be found from the proposed system.

5.4 Future Work

The proposed system can automatically find the 'best fit' model among the six frequently used linear regression models for a given dataset. More regression models can be added to the system in the future to make the system more powerful and robust. Moreover, support for categorical variables can be added in the future so that users don't have to manually encode the categorical variables into numerical ones.

REFERENCES

[1] G. K. Uyanık and N. Güler, "A study on multiple linear regression analysis," *Procedia - Social and Behavioral Sciences*, vol. 106, pp. 234–240, 2013, 4th International Conference on New Horizons in Education. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877042813046429

[2] D. F. Andrews, "A robust method for multiple linear regression," *Technometrics*, vol. 16, no. 4, pp. 523–531, 1974. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/00401706.1974.10489233

[3] T. Isobe, E. D. Feigelson, M. G. Akritas, and G. J. Babu, "Linear Regression in Astronomy. I." , vol. 364, p. 104, Nov. 1990.

[4] M. A. Poole and P. N. O'Farrell, "The assumptions of the linear regression model," *Transactions of the Institute of British Geographers*, no. 52, pp. 145–158, 1971. [Online]. Available: http://www.jstor.org/stable/621706

[5] curfi-npm. [Online]. Available: https://tinyurl.com/dw8yhpfa

[6] M. GULSEN, A. E. SMITH, and D. M. TATE, "A genetic algorithm approach to curve fitting," *International Journal of Production Research*, vol. 33, no. 7, pp. 1911–1923, 1995. [Online]. Available: https://doi.org/10.1080/00207549508904789

[7] R. Bellman and R. Roth, "Curve fitting by segmented straight lines," *Journal of the American Statistical Association*, vol. 64, no. 327, pp. 1079–1084, 1969. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/01621459.1969.10501038

[8] S. Ranganathan, K. Nakai, and C. Schönbach, *Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics*, 01 2019.

[9] P. Palmer and D. O'Connell, "Regression analysis for prediction: Understanding the process," *Cardiopulmonary physical therapy journal*, vol. 20, pp. 23–6, 09 2009.

[10] R. J. Leatherbarrow, "Using linear and non-linear regression to fit biochemical data," *Trends in Biochemical Sciences*, vol. 15, no. 12, pp. 455–458, 1990. [Online]. Available: https://www.sciencedirect.com/science/article/pii/096800049090295M

[11] S. C. Chapra and R. P. Canale, *Numerical methods for engineers*. McGraw Hill, 2006.

[12] K. Benoit, "Linear regression models with logarithmic transformations," 2011.

[13] Exponential linear regression — real statistics using excel. [Online]. Available: https://tinyurl.com/385rmay4

[14] N. Fumo and M. Biswas, "Regression analysis for prediction of residential energy consumption," *RenewableandSustainableEnergyReviews*, vol. 47, pp. 332–343, 03 2015.

[15] A. Olaniyi, A. S., and J. G, "Stock trend prediction using regression analysis – a data mining approach," *ARPN Journal of Systems and Software*, vol. 1, pp. 154–157, 01 2011.

[16] D. I. Bradshaw, J. George, A. Hyde, M. LaMonte, P. Vehrs, R. Hager, and F. Yanowitz, "An accurate vo2max nonexercise regression model for 18–65-year-old adults," *Research Quarterly for Exercise and Sport*, vol. 76, pp. 426 – 432, 2005.

[17] D. M.D, D. M.S, M. B.A, A. M.D, P. M.D, C. Heim, D. J. Newport, D. Wagner, M. Wilcox, and A. Miller, "The role of early adverse experience and adulthood stress in the prediction of neuroendocrine stress reactivity in women: A multiple regression analysis," *Depression and Anxiety*, vol. 15, pp. 117 – 125, 01 2002.

[18] Curfi — ayon. [Online]. Available: https://curfi.netlify.app/

[19] K. A. Marill, "Advanced statistics: Linear regression, part ii: Multiple linear regression," *Academic Emergency Medicine*, vol. 11, no. 1, pp. 94–102, 2004. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1197/j.aem.2003.09.006

[20] C. Blake, "Uci repository of machine learning databases," 1998.

APPENDIX A
ALGORITHMS

A.1 Calculate the coefficient matrix A

Algorithm 1 Calculate the coefficient matrix A

Input: the matrix a, an array of length $order + 2$
Output: the coefficient matrix A, or $error$

```
procedure CALCULATE_A(a)
    Z ← an empty array
    Y ← an empty array
    for z ← 1 to (length of a)−1 do
        z3 ← an empty array
        y3 ← an empty array
        for z2 ← 1 to (length of a[z])−1 do
            if z2 ≠ (length of a[z]) − 1 then
                insert a[z][z2] into z3
            end if
            if z2 == (length of a[z]) − 1 then
                insert a[z][z2] into y3
            end if
        end for
        insert z3 into Z
        insert y3 into Y
    end for
    if matrix Z is invertible then
        Zinv ← inverse matrix of Z
        A ← matrix multiplication of Zinv and Y
        return A
    end if
    return error
end procedure
```

A.2 Multiple Linear Regression Algorithm

Algorithm 2 Calculate the matrix A for Multiple Linear Regression

Input: $trainX$, the feature data points and
$trainY$, the label data points of the dataset

Output: the coefficient matrix A, or $error$

```
procedure FIT_MULTIPLE_LINEAR_REGRESSION(trainX, trainY)
    n ← no of data points
    x0 ← an array of length n filled with 1
    x ← trainX
    y ← trainY
    insert x0 at the beginning of x
    order ← length of trainX
    a ← an array of length order + 2 filled with 0
    for i ← 1 to order + 1 do
        for j ← 1 to i do
            sum ← 0
            for l ← 0 to n − 1 do
                sum ← sum + x[i − 1][l] ∗ x[j − 1][l]
            end for
            a[i][j] ← sum
            a[j][i] ← sum
        end for
        sum ← 0
        for l ← 0 to n − 1 do
            sum ← sum + y[0][l] ∗ x[i − 1][l]
        end for
        a[i][order + 2] ← sum
    end for
    A ← CALCULATE_A(a)
    return A
end procedure
```

A.3 Polynomial Linear Regression Algorithm

Algorithm 3 Calculate the matrix A for Polynomial Linear Regression

Input: $trainX$, the feature data points and
$trainY$, the label data points of the dataset

Output: the coefficient matrix A, or $error$

procedure FIT_POLYNOMIAL_LINEAR_REGRESSION($trainX, trainY, order$)
    $n \leftarrow$ no of data points
    $x0 \leftarrow$ an array of length n filled with 1
    $x \leftarrow trainX$
    $y \leftarrow trainY$
    insert $x0$ at the beginning of x
    $a \leftarrow$ an array of length $order + 2$ filled with 0
    for $i \leftarrow 1$ to $order + 1$ **do**
        for $j \leftarrow 1$ to i **do**
            $k \leftarrow i + j - 2$
            $sum \leftarrow 0$
            for $l \leftarrow 0$ to $n - 1$ **do**
                $sum \leftarrow sum + (x[1][l])^k$
            end for
            $a[i][j] \leftarrow sum$
            $a[j][i] \leftarrow sum$
        end for
        $sum \leftarrow 0$
        for $l \leftarrow 0$ to $n - 1$ **do**
            $sum \leftarrow sum + y[0][l] * (x[1][l])^{i-1}$
        end for
        $a[i][order + 2] \leftarrow sum$
    end for
    $A \leftarrow$ CALCULATE_A(a)
    return A
end procedure

A.4 Logarithmic Linear Regression Algorithm

Algorithm 4 Calculate the matrix A for Logarithmic Linear Regression

Input: $trainX$, the feature data points and
$trainY$, the label data points of the dataset

Output: the coefficient matrix A, or $error$

```
procedure FIT_LOGARITHMIC_LINEAR_REGRESSION(trainX, trainY)
    n ← no of data points
    x0 ← an array of length n filled with 1
    x ← an empty array
    for each element in trainX do
        if element ≤ 0 then return error
        else
            insert log_e (element) into x
        end if
    end for
    y ← trainY
    insert x0 at the beginning of x
    order ← length of trainX
    a ← an array of length order + 2 filled with 0
    for i ← 1 to order + 1 do
        for j ← 1 to i do
            sum ← 0
            for l ← 0 to n − 1 do
                sum ← sum + x[i − 1][l] ∗ x[j − 1][l]
            end for
            a[i][j] ← sum
            a[j][i] ← sum
        end for
        sum ← 0
        for l ← 0 to n − 1 do
            sum ← sum + y[0][l] ∗ x[i − 1][l]
        end for
        a[i][order + 2] ← sum
    end for
    A ← CALCULATE_A(a)
    return A
end procedure
```

A.5 Exponential Linear Regression Algorithm

Algorithm 5 Calculate the matrix A for Exponential Linear Regression

Input: $trainX,$ the feature data points and
$trainY,$ the label data points of the dataset
Output: the coefficient matrix A, or $error$

procedure FIT_EXPONENTIAL_LINEAR_REGRESSION($trainX, trainY$)
  $n \leftarrow$ no of data points
  $x0 \leftarrow$ an array of length n filled with 1
  $x \leftarrow$ an empty array
  for each $element$ in $trainX$ **do**
    if $e^{element} == \infty$ or $e^{element} == -\infty$ **then return** $error$
    else
      insert $e^{element}$ into x
    end if
  end for
  $y \leftarrow trainY$
  insert $x0$ at the beginning of x
  $order \leftarrow$ length of $trainX$
  $a \leftarrow$ an array of length $order + 2$ filled with 0
  for $i \leftarrow 1$ to $order + 1$ **do**
    for $j \leftarrow 1$ to i **do**
      $sum \leftarrow 0$
      for $l \leftarrow 0$ to $n - 1$ **do**
        $sum \leftarrow sum + x[i-1][l] * x[j-1][l]$
      end for
      $a[i][j] \leftarrow sum$
      $a[j][i] \leftarrow sum$
    end for
    $sum \leftarrow 0$
    for $l \leftarrow 0$ to $n - 1$ **do**
      $sum \leftarrow sum + y[0][l] * x[i-1][l]$
    end for
    $a[i][order + 2] \leftarrow sum$
  end for
  $A \leftarrow$ CALCULATE_A(a)
  return A
end procedure

A.6 Sinusoidal Regression Algorithm

Algorithm 6 Calculate the matrix A for Sinusoidal Regression

Input: $trainX$, the feature data points and
$trainY$, the label data points of the dataset

Output: the coefficient matrix A, or $error$

```
procedure FIT_SINUSOIDAL_REGRESSION(trainX, trainY)
    n ← no of data points
    x0 ← an array of length n filled with 1
    x ← an empty array
    for each element in trainX do
        insert sin (element) into x
        insert cos (element) into x
    end for
    y ← trainY
    insert x0 at the beginning of x
    order ← (length of x) − 1
    a ← an array of length order + 2 filled with 0
    for i ← 1 to order + 1 do
        for j ← 1 to i do
            sum ← 0
            for l ← 0 to n − 1 do
                sum ← sum + x[i − 1][l] ∗ x[j − 1][l]
            end for
            a[i][j] ← sum
            a[j][i] ← sum
        end for
        sum ← 0
        for l ← 0 to n − 1 do
            sum ← sum + y[0][l] ∗ x[i − 1][l]
        end for
        a[i][order + 2] ← sum
    end for
    A ← CALCULATE_A(a)
    return A
end procedure
```

APPENDIX B
CODES

B.1 Code of the 'CurFi' web app

We use this JavaScript code to develop 6 linear regression models along with their r^2 score.

```
// Failed to inverse '-111'

class Curfi {
    constructor() {
        this.modelName = '';
        this.weights = [];
        this.weightsLen = 0;
        this.inputShape = [];
        this.outputShape = [];
        this.accuracy = {
            r2_score: 0,
        };
        this.additionalParams = {};
    }

    modelParams(modelName, weights, inputShape, outputShape) {
        this.weights = [...weights];
        this.modelName = modelName;
        this.weightsLen = weights.length;
        this.inputShape = inputShape;
        this.outputShape = outputShape;
    }

    modelAccuracy(accuracy) {
        this.accuracy = { ...accuracy };
    }

    modelAdditionalParams(params) {
        this.additionalParams = { ...params };
    }

    loadModel(modelc) {
        this.modelParams(modelc.modelName, modelc.weights, modelc.inputShape
, modelc.outputShape);
```

```
        this.modelAdditionalParams(modelc.additionalParams);
        this.modelAccuracy(modelc.accuracy);
        return this;
    }

    saveModel(exportName = 'model') {
        var dataStr = "data:text/json;charset=utf-8," + encodeURIComponent(JSON.stringify({ ...this }));
        var downloadAnchorNode = document.createElement('a');
        downloadAnchorNode.setAttribute("href", dataStr);
        downloadAnchorNode.setAttribute("download", exportName + ".json");
        document.body.appendChild(downloadAnchorNode); // required for firefox
        downloadAnchorNode.click();
        downloadAnchorNode.remove();
    }

    fit_AllModels(trainX, trainY, highestOrder = 3) {
        let models = {
            singleX: {},
            multiX: {}
        };
        // For Multiple X variables
        if (trainX.length >= 1) {
            models.multiX.MLR = { ...this.fit_MLR(trainX, trainY) };
            models.multiX.EXP = { ...this.fit_EXP(trainX, trainY) };
            models.multiX.LogLR = { ...this.fit_LogLR(trainX, trainY) };
        }
        // For Single X variable
        if (trainX.length == 1) {
            models.singleX.LR = { ...this.fit_LR(trainX, trainY) };
            models.singleX.PLR = { ...this.fit_PLR(trainX, trainY, highestOrder) };
            models.singleX.SinLR = { ...this.fit_SinLR(trainX, trainY) };
        }

        function clean(obj) {
            for (let propNameo in obj) {
                for (let propName in obj[propNameo]) {
                    if (Object.keys(obj[propNameo][propName]).length === 0 && obj[propNameo][propName].constructor === Object || obj[propNameo][propName] === null || obj[propNameo][propName] === undefined) {
                        delete obj[propNameo][propName];
                    }
                    else if (Number.isNaN(obj[propNameo][propName].weights[0][0])) {
                        delete obj[propNameo][propName];
                    }
```

```
                }
            }
            return obj
        }

        clean(models);
        return { ...models };
    }


    // Linear Regression functions
    fit_LR(trainX, trainY) {
        return this.fit_LinearRegression(trainX, trainY);
    }
    fit_LinearRegression(trainX, trainY) {
        let obj = this.fit_MultipleLinearRegression(trainX, trainY);
        let A = [...obj.weights];
        this.modelParams("LinearRegression", [...A], trainX.length, trainY.length);
        let accuracy = {
            r2_score: obj.accuracy.r2_score,
        };
        this.modelAccuracy(accuracy);
        return this;
    }
    // MultipleLinearRegression Functions
    fit_MLR(trainX, trainY) {
        return this.fit_MultipleLinearRegression(trainX, trainY);
    }

    fit_MultipleLinearRegression(trainX, trainY) {
        // no of data points, no of rows
        let n = trainX[0].length;
        // all 1s array
        let x0 = new Array(n).fill(1);
        let x = [...trainX];
        let y = [...trainY];
        // inserting xo(all 1s) at the beginning
        x.unshift(x0);
        // order = m = columns in the dataset
        let order = trainX.length;
        let a = [...Array(order + 2)].map(e => Array(order + 2).fill(0));
        // let a = [...aa];

        // Coefficient matrix calculation A[]
        for (let i = 1; i <= order + 1; i++) {
            for (let j = 1; j <= i; j++) {
                let sum = 0;
```

```
                for (let l = 0; l < n; l++) {
                    sum += x[i - 1][l] * x[j - 1][l];
                }
                a[i][j] = sum;
                a[j][i] = sum;
            }
            let sum = 0;
            for (let l = 0; l < n; l++) {
                sum += y[0][l] * x[i - 1][l];
            }
            a[i][order + 2] = sum;
        }

        // Only the coefficients and not the Y part
        let Z = [];
        let Y = [];
        for (let z = 1; z < a.length; z++) {
            let zzz = [];
            let yyy = [];
            for (let zz = 1; zz < a[z].length; zz++) {
                if (zz != a[z].length - 1) {
                    zzz.push(a[z][zz]);
                }
                // order+2 are the Y matrix
                if (zz == a[z].length - 1) {
                    yyy.push(a[z][zz]);
                }
            }
            Z.push(zzz);
            Y.push(yyy);
        }
        // console.log(a, Z, Y)
        // calculation for A[] from A =  INV(Z) * Y
        let Zinv = this.matrix_invert(Z);
        if (Zinv === -111) { return {}; }
        // console.log(Zinv)
        let A = this.matrix_multiply(Zinv, Y);
        // console.log(A)

        // set this models parameters
        this.modelParams("MultipleLinearRegression", [...A], trainX.length, trainY.length);
        let accuracy = {
            r2_score: this.r2_score(trainY[0], this.predict(trainX)[0]),
        };
        this.modelAccuracy(accuracy);
        return this;
    }
```

```
// Exponential LinearRegression Functions
fit_EXP(trainX, trainY) {
    return this.fit_ExpLinearRegression(trainX, trainY);
}

fit_ExpLinearRegression(trainX, trainY) {
    // no of data points, no of rows
    let n = trainX[0].length;
    // all 1s array
    let x0 = new Array(n).fill(1);
    let isNaN = 0;
    let x = [...trainX].map(el1 => el1.map(el2 => { if (Math.exp(el2) ==
Number.POSITIVE_INFINITY || Math.exp(el2) == Number.NEGATIVE_INFINITY) {
isNaN = 1; } return Math.exp(el2); }));
    if (isNaN) {
        return {};
    }
    let y = [...trainY];
    // inserting xo(all 1s) at the beginning
    x.unshift(x0);
    // order = m = columns in the dataset
    let order = trainX.length;
    let a = [...Array(order + 2)].map(e => Array(order + 2).fill(0));
    // let a = [...aa];

    // Coefficient matrix calculation A[]
    for (let i = 1; i <= order + 1; i++) {
        for (let j = 1; j <= i; j++) {
            let sum = 0;
            for (let l = 0; l < n; l++) {
                sum += x[i - 1][l] * x[j - 1][l];
            }
            a[i][j] = sum;
            a[j][i] = sum;
        }
        let sum = 0;
        for (let l = 0; l < n; l++) {
            sum += y[0][l] * x[i - 1][l];
        }
        a[i][order + 2] = sum;
    }

    // Only the coefficients and not the Y part
    let Z = [];
    let Y = [];
    for (let z = 1; z < a.length; z++) {
```

```
            let zzz = [];
            let yyy = [];
            for (let zz = 1; zz < a[z].length; zz++) {
                if (zz != a[z].length - 1) {
                    zzz.push(a[z][zz]);
                }
                // order+2 are the Y matrix
                if (zz == a[z].length - 1) {
                    yyy.push(a[z][zz]);
                }
            }
            Z.push(zzz);
            Y.push(yyy);
        }
        // console.log(a, Z, Y)
        // calculation for A[] from A =  INV(Z) * Y
        let Zinv = this.matrix_invert(Z);
        if (Zinv === -111) { return {}; }
        // console.log(Zinv)
        let A = this.matrix_multiply(Zinv, Y);
        // console.log(A)

        // set this models parameters
        this.modelParams("ExponentialLinearRegression", [...A], trainX.length, trainY.length);
        let accuracy = {
            r2_score: this.r2_score(trainY[0], this.predict(trainX)[0]),
        };
        this.modelAccuracy(accuracy);
        return this;
    }

    // Polynomial LinearRegression Functions
    fit_PLR(trainX, trainY, order = 3) {
        return this.fit_PolynomialLinearRegression(trainX, trainY, order);
    }

    fit_PolynomialLinearRegression(trainX, trainY, order = 3) {
        // no of data points, no of rows
        let n = trainX[0].length;
        // all 1s array
        let x0 = new Array(n).fill(1);
        let x = [...trainX];
        let y = [...trainY];
        // inserting xo(all 1s) at the beginning
        x.unshift(x0);
        // order = m = columns in the dataset
        //let order = trainX.length;
```

```
        let a = [...Array(order + 2)].map(e => Array(order + 2).fill(0));
        // let a = [...aa];
        // Coefficient matrix calculation A[]
        for (let i = 1; i <= order + 1; i++) {
            for (let j = 1; j <= i; j++) {
                let k = i + j - 2;
                let sum = 0;
                for (let l = 0; l < n; l++) {
                    sum += Math.pow(x[1][l], k);
                }
                a[i][j] = sum;
                a[j][i] = sum;
            }
            let sum = 0;
            for (let l = 0; l < n; l++) {
                sum += y[0][l] * Math.pow(x[1][l], i - 1);
            }
            a[i][order + 2] = sum;
        }
        // Only the coefficients and not the Y part
        let Z = [];
        let Y = [];
        for (let z = 1; z < a.length; z++) {
            let zzz = [];
            let yyy = [];
            for (let zz = 1; zz < a[z].length; zz++) {
                if (zz != a[z].length - 1) {
                    zzz.push(a[z][zz]);
                }
                // order+2 are the Y matrix
                if (zz == a[z].length - 1) {
                    yyy.push(a[z][zz]);
                }
            }
            Z.push(zzz);
            Y.push(yyy);
        }
        // console.log(a, Z, Y)
        // calculation for A[] from A =  INV(Z) * Y
        let Zinv = this.matrix_invert(Z);
        if (Zinv === -111) { return {}; }
        // console.log(Zinv)
        let A = this.matrix_multiply(Zinv, Y);
        // set this models parameters
        this.modelParams("PolynomialLinearRegression", [...A], trainX.length
    , trainY.length);
        this.modelAdditionalParams({ order });
        let accuracy = {
```

```
            r2_score: this.r2_score(trainY[0], this.predict(trainX)[0]),
        };
        this.modelAccuracy(accuracy);
        return this;
    }

    // Logarithmic LinearRegression Functions
    fit_LogLR(trainX, trainY) {
        return this.fit_LogarithmicLinearRegression(trainX, trainY);
    }

    fit_LogarithmicLinearRegression(trainX, trainY) {
        // no of data points, no of rows
        let n = trainX[0].length;
        // if any zero in x then log will be NaN
        // In general, the function y=logbx where b is base,x>0 and  b 1  is a continuous and one-to-one function. Note that the logarithmic functionis not defined for negative numbers or for zero. The graph of the function approaches the y -axis as x tends to      , but never touches it .

        let isZero = 0;
        // all 1s array
        let x0 = new Array(n).fill(1);
        let x = [...trainX].map(el1 => el1.map(el2 => { if (el2 <= 0) { isZero = 1; } return Math.log(el2); }));
        if (isZero) {
            return {};
        }
        let y = [...trainY];
        // inserting xo(all 1s) at the beginning
        x.unshift(x0);
        // order = m = columns in the dataset
        let order = trainX.length;
        let a = [...Array(order + 2)].map(e => Array(order + 2).fill(0));
        // let a = [...aa];

        // Coefficient matrix calculation A[]
        for (let i = 1; i <= order + 1; i++) {
            for (let j = 1; j <= i; j++) {
                let sum = 0;
                for (let l = 0; l < n; l++) {
                    sum += x[i - 1][l] * x[j - 1][l];
                }
                a[i][j] = sum;
                a[j][i] = sum;
            }
            let sum = 0;
```

```
            for (let l = 0; l < n; l++) {
                sum += y[0][l] * x[i - 1][l];
            }
            a[i][order + 2] = sum;
        }

        // Only the coefficients and not the Y part
        let Z = [];
        let Y = [];
        for (let z = 1; z < a.length; z++) {
            let zzz = [];
            let yyy = [];
            for (let zz = 1; zz < a[z].length; zz++) {
                if (zz != a[z].length - 1) {
                    zzz.push(a[z][zz]);
                }
                // order+2 are the Y matrix
                if (zz == a[z].length - 1) {
                    yyy.push(a[z][zz]);
                }
            }
            Z.push(zzz);
            Y.push(yyy);
        }
        // console.log(a, Z, Y)
        // calculation for A[] from A =  INV(Z) * Y
        let Zinv = this.matrix_invert(Z);
        if (Zinv === -111) { return {}; }
        // console.log(Zinv)
        let A = this.matrix_multiply(Zinv, Y);
        // console.log(A)

        // set this models parameters
        this.modelParams("LogarithmicLinearRegression", [...A], trainX.
    length, trainY.length);
        let accuracy = {
            r2_score: this.r2_score(trainY[0], this.predict(trainX)[0]),
        };
        this.modelAccuracy(accuracy);
        return this;
    }

    // Sinusoidal Regression Functions
    fit_SinLR(trainX, trainY) {
        return this.fit_SinusoidalRegression(trainX, trainY);
    }

    fit_SinusoidalRegression(trainX, trainY) {
```

```
        // no of data points, no of rows
        let n = trainX[0].length;
        // all 1s array
        let x0 = new Array(n).fill(1);
        let x = [...trainX].map(el1 => el1.map(el2 => Math.sin(el2 * (Math.PI / 180))));
        let y = [...trainY];
        // inserting xo(all 1s) at the beginning
        x.unshift(x0);
        x.push(trainX[0].map(el => Math.cos(el * (Math.PI / 180))));
        // order = m = columns in the dataset
        let order = x.length - 1;
        let a = [...Array(order + 2)].map(e => Array(order + 2).fill(0));
        // let a = [...aa];

        // Coefficient matrix calculation A[]
        for (let i = 1; i <= order + 1; i++) {
            for (let j = 1; j <= i; j++) {
                let sum = 0;
                for (let l = 0; l < n; l++) {
                    sum += x[i - 1][l] * x[j - 1][l];
                }
                a[i][j] = sum;
                a[j][i] = sum;
            }
            let sum = 0;
            for (let l = 0; l < n; l++) {
                sum += y[0][l] * x[i - 1][l];
            }
            a[i][order + 2] = sum;
        }

        // Only the coefficients and not the Y part
        let Z = [];
        let Y = [];
        for (let z = 1; z < a.length; z++) {
            let zzz = [];
            let yyy = [];
            for (let zz = 1; zz < a[z].length; zz++) {
                if (zz != a[z].length - 1) {
                    zzz.push(a[z][zz]);
                }
                // order+2 are the Y matrix
                if (zz == a[z].length - 1) {
                    yyy.push(a[z][zz]);
                }
            }
            Z.push(zzz);
```

```
            Y.push(yyy);
        }
        // console.log(a, Z, Y)
        // calculation for A[] from A =  INV(Z) * Y
        let Zinv = this.matrix_invert(Z);
        if (Zinv === -111) { return {}; }
        // console.log(Zinv)
        let A = this.matrix_multiply(Zinv, Y);
        // console.log(A)
        let A1 = Math.sqrt(Math.pow(A[1][0], 2) + Math.pow(A[2][0], 2)); // the amplitude
        let delta = Math.atan(A[2][0] / A[1][0]) * (180 / Math.PI); // phase shift
        A[1][0] = A1;
        A[2][0] = delta;

        // So the final sinusoidal model will be in this form:
        // y = A0 + A1 sin (x + delta) where A0, A1 and delta are the three parameters

        // set this models parameters
        this.modelParams("SinusoidalRegression", [...A], trainX.length, trainY.length);
        let accuracy = {
            r2_score: this.r2_score(trainY[0], this.predict(trainX)[0]),
        };
        this.modelAccuracy(accuracy);
        return this;
    }

    AutoTrain(trainX, trainY, testX = null, testY = null, highestOrder = 3) {
        let models = this.fit_AllModels(trainX, trainY, highestOrder);
        let multimodels = { ...models.multiX };
        let singlemodels = { ...models.singleX };
        let allmodels = { ...multimodels, ...singlemodels };

        let sortedModel = [];
        Object
            .keys(allmodels).sort(function (a, b) {
                return allmodels[b].accuracy.r2_score - allmodels[a].accuracy.r2_score;
            })
            .forEach(function (key) {
                if (testY === null) {
                    let obj = new Curfi();
                    obj.loadModel(allmodels[key]);

```

```
                    sortedModel.push(obj);
                } else if (testX !== null && testY !== null) {
                    let obj = new Curfi();
                    obj.loadModel(allmodels[key]);
                    obj.accuracy.r2_score_test = obj.r2_score(testY[0], obj.predict(testX)[0]);

                    sortedModel.push(obj);
                }
            });
        this.loadModel(sortedModel[0]);
        return sortedModel;
    }


    // textX is in columnwise
    predict(testX) {
        let wt = [...this.weights];
        let testY = [];
        for (let r = 0; r < testX[0].length; r++) {
            let sum = wt[0][0];
            if (this.modelName === "PolynomialLinearRegression") {
                for (let c = 0; c < this.additionalParams.order; c++) {
                    sum += wt[c + 1][0] * this.coefficientFunction(testX[0][r], c + 1);
                }
            } else {
                for (let c = 0; c < testX.length; c++) {
                    sum += wt[c + 1][0] * this.coefficientFunction(testX[c][r], c + 1);
                }
            }
            testY.push(sum);
        }
        return [[...testY]];
    }

    coefficientFunction(val, pos) {
        switch (this.modelName) {
            case "LinearRegression":
                return val;
                break;
            case "PolynomialLinearRegression":
                return Math.pow(val, pos);
                break;
            case "MultipleLinearRegression":
                return val;
                break;
```

```
            case "ExponentialLinearRegression":
                return Math.exp(val);
                break;
            case "LogarithmicLinearRegression":
                return Math.log(val);
                break;
            case "SinusoidalRegression":
                return Math.sin((val + this.weights[pos + 1][0]) * Math.PI / 180);
                break;

            default:
                return val;
                break;
        }
    }

    // r2 value function
    r2_score(y_true, y_pred) {
        let numOr0 = n => isNaN(n) ? 0 : n;
        let y_true_Sum = y_true.reduce((a, b) => numOr0(a) + numOr0(b));
        let y_true_Mean = y_true_Sum / y_true.length;

        let St = 0;
        let Sr = 0;
        for (let yi = 0; yi < y_true.length; yi++) {
            St += (y_true[yi] - y_true_Mean) * (y_true[yi] - y_true_Mean);
            Sr += (y_true[yi] - y_pred[yi]) * (y_true[yi] - y_pred[yi]);
        }
        return (St - Sr) / St;
    }

    // Round upto digits after decimal
    round(num, digits) {
        return Math.round((num + Number.EPSILON) * Math.pow(10, digits)) / Math.pow(10, digits);
    }
    // Round up to 3 digits after decimal
    round3(num) {
        return Math.round((num + Number.EPSILON) * 1000) / 1000;
    }
    // Round up to 2 digits after decimal
    round2(num) {
        return Math.round((num + Number.EPSILON) * 100) / 100;
    }

    // Matrix Functions
    matrix_multiply(a, b) {
```

```
        var aNumRows = a.length, aNumCols = a[0].length,
            bNumRows = b.length, bNumCols = b[0].length,
            m = new Array(aNumRows);  // initialize array of rows
        for (var r = 0; r < aNumRows; ++r) {
            m[r] = new Array(bNumCols); // initialize the current row
            for (var c = 0; c < bNumCols; ++c) {
                m[r][c] = 0;             // initialize the current cell
                for (var i = 0; i < aNumCols; ++i) {
                    m[r][c] += a[r][i] * b[i][c];
                }
            }
        }
        return m;
    }

    matrix_transpose(a) {

        // Calculate the width and height of the Array
        var w = a.length || 0;
        var h = a[0] instanceof Array ? a[0].length : 0;

        // In case it is a zero matrix, no transpose routine needed.
        if (h === 0 || w === 0) { return []; }

        /**
         * @var {Number} i Counter
         * @var {Number} j Counter
         * @var {Array} t Transposed data is stored in this array.
         */
        var i, j, t = [];

        // Loop through every item in the outer array (height)
        for (i = 0; i < h; i++) {

            // Insert a new row (array)
            t[i] = [];

            // Loop through every item per item in outer array (width)
            for (j = 0; j < w; j++) {

                // Save transposed data.
                t[i][j] = a[j][i];
            }
        }

        return t;
    }

```

```
    // Returns the inverse of matrix `M`.
    matrix_invert(M) {
        // I use Guassian Elimination to calculate the inverse:
        // (1) 'augment' the matrix (left) by the identity (on the right)
        // (2) Turn the matrix on the left into the identity by elemetry row ops
        // (3) The matrix on the right is the inverse (was the identity matrix)
        // There are 3 elemtary row ops: (I combine b and c in my code)
        // (a) Swap 2 rows
        // (b) Multiply a row by a scalar
        // (c) Add 2 rows

        //if the matrix isn't square: exit (error)
        if (M.length !== M[0].length) { return -111; }

        //create the identity matrix (I), and a copy (C) of the original
        var i = 0, ii = 0, j = 0, dim = M.length, e = 0, t = 0;
        var I = [], C = [];
        for (i = 0; i < dim; i += 1) {
            // Create the row
            I[I.length] = [];
            C[C.length] = [];
            for (j = 0; j < dim; j += 1) {

                //if we're on the diagonal, put a 1 (for identity)
                if (i == j) { I[i][j] = 1; }
                else { I[i][j] = 0; }

                // Also, make the copy of the original
                C[i][j] = M[i][j];
            }
        }

        // Perform elementary row operations
        for (i = 0; i < dim; i += 1) {
            // get the element e on the diagonal
            e = C[i][i];

            // if we have a 0 on the diagonal (we'll need to swap with a lower row)
            if (e == 0) {
                //look through every row below the i'th row
                for (ii = i + 1; ii < dim; ii += 1) {
                    //if the ii'th row has a non-0 in the i'th col
                    if (C[ii][i] != 0) {
                        //it would make the diagonal have a non-0 so swap it
```

```
                    for (j = 0; j < dim; j++) {
                        e = C[i][j];        //temp store i'th row
                        C[i][j] = C[ii][j];//replace i'th row by ii'th
                        C[ii][j] = e;       //repace ii'th by temp
                        e = I[i][j];        //temp store i'th row
                        I[i][j] = I[ii][j];//replace i'th row by ii'th
                        I[ii][j] = e;       //repace ii'th by temp
                    }
                    //don't bother checking other rows since we've swapped
                    break;
                }
            }
            //get the new diagonal
            e = C[i][i];
            //if it's still 0, not invertable (error)
            if (e == 0) { return -111; }
        }

        // Scale this row down by e (so we have a 1 on the diagonal)
        for (j = 0; j < dim; j++) {
            C[i][j] = C[i][j] / e; //apply to original matrix
            I[i][j] = I[i][j] / e; //apply to identity
        }

        // Subtract this row (scaled appropriately for each row) from ALL of
        // the other rows so that there will be 0's in this column in the
        // rows above and below this one
        for (ii = 0; ii < dim; ii++) {
            // Only apply to other rows (we want a 1 on the diagonal)
            if (ii == i) { continue; }

            // We want to change this element to 0
            e = C[ii][i];

            // Subtract (the row above(or below) scaled by e) from (the
            // current row) but start at the i'th column and assume all the
            // stuff left of diagonal is 0 (which it should be if we made this
            // algorithm correctly)
            for (j = 0; j < dim; j++) {
                C[ii][j] -= e * C[i][j]; //apply to original matrix
                I[ii][j] -= e * I[i][j]; //apply to identity
            }
        }
```

```
        }

        //we've done all operations, C should be the identity
        //matrix I should be the inverse:
        return I;
    }

    modelEqnnHTML(model = this, rnd = this.round3) {
        this.loadModel(model);
        let ystr = '';
        switch (this.modelName) {
            case "LinearRegression":
                ystr = `y = (${rnd(this.weights[0][0])})`;
                for (let a = 1; a < this.weightsLen; a++) {
                    ystr += ` + (${rnd(this.weights[a][0])}) x`;
                }
                return ystr;
                break;
            case "PolynomialLinearRegression":
                ystr = `y = (${rnd(this.weights[0][0])})`;
                for (let a = 1; a < this.weightsLen; a++) {
                    ystr += ` + (${rnd(this.weights[a][0])}) x<sup>${a}</sup>`;
                }
                return ystr;
                break;
            case "MultipleLinearRegression":
                ystr = `y = (${rnd(this.weights[0][0])})`;
                for (let a = 1; a < this.weightsLen; a++) {
                    ystr += ` + (${rnd(this.weights[a][0])}) x<sub>${a}</sub>`;
                }
                return ystr;
                break;
            case "ExponentialLinearRegression":
                ystr = `y = (${rnd(this.weights[0][0])})`;
                for (let a = 1; a < this.weightsLen; a++) {
                    ystr += ` + (${rnd(this.weights[a][0])}) e<sup>x<sub>${a}</sub></sup>`;
                }
                return ystr;
                break;
            case "LogarithmicLinearRegression":
                ystr = `y = (${rnd(this.weights[0][0])})`;
                for (let a = 1; a < this.weightsLen; a++) {
                    ystr += ` + (${rnd(this.weights[a][0])}) log(x<sub>${a}</sub>)`;
                }
```

```
                return ystr;
                break;
            case "SinusoidalRegression":
                ystr = `y = (${rnd(this.weights[0][0])}) + (${rnd(this.weights[1][0])}) Sin(x + (${rnd(this.weights[2][0])}))`;
                return ystr;
                break;

            default:
                return ystr = `Couldn't Create Equation      `;
                break;
        }
    }
}
```