Implementation of Curve Fitting using Polynomial Regression in Python

Ahmad Farhan AlShammari
Department of Computer and Information Systems
College of Business Studies, PAAET
Kuwait

ABSTRACT
The goal of this research is to develop a curve fitting program using polynomial regression in Python. Curve fitting is an important application in machine learning. It helps to find the curve that best fits to the data points. The polynomial regression is used to model the relationship between the independent variable (x) and the dependent variable (y) using a polynomial function of degree (n). Polynomial regression can provide linear and non-linear models.

The basic steps of curve fitting using polynomial regression are explained: preparing observed points, computing matrix, computing transpose of matrix, multiplying by transpose, performing forward elimination, performing back substitution, finding out coefficients, making polynomial equation, computing predicted points, and plotting curve.

The developed program was tested on an experimental dataset from Kaggle. The program successfully performed the basic steps of curve fitting using polynomial regression and provided the required results.

Keywords
Artificial Intelligence, Machine Learning, Curve Fitting, Polynomial Regression, Numerical Methods, Gauss Method, Python, Programming.

1. INTRODUCTION
In recent years, machine learning has played a major role in the development of computer systems. Machine learning (ML) is a branch of Artificial Intelligence (AI) which is focused on the study of computer algorithms to improve the performance of computer programs [1-4, 5-7, 8-11].

Curve fitting is one of the important applications in machine learning. It helps to find the curve that best fits to the data points. Curve fitting is a common area that is sharing knowledge with many fields like: machine learning, programming, data science, mathematics, statistics, and numerical methods [12-14, 15-18].

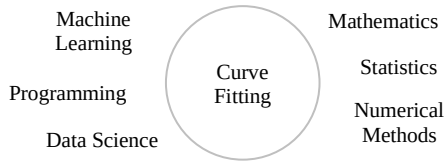

Fig 1: Field of Curve Fitting

In this paper, curve fitting is applied using polynomial regression to find the curve that best fits to the data points. Polynomial regression is extensively used in data analysis to explore the behavior of data. It has a wide range of applications in many fields like: engineering, business, education, medicine, public health, environment, climate change, etc.

2. LITERATURE REVIEW
The review of literature outlined the major contributions in curve fitting using polynomial regression [19-23, 24-29].

Polynomial regression is a well-known method in mathematics. It was first presented in (1815) by Gergonne [30]. Polynomial regression is used to model the relationship between the independent variable (x) and the dependent variable (y) using a polynomial function of degree (n) as shown in the following form:

$$f(x) = a_0 + a_1x + a_2x^2 + \ldots + a_n x^n$$

There are many functions that can be used to implement the models of curve fitting. For instance: polynomial, logarithmic (log), exponential (e^x), and trigonometric (sin, cos). The polynomial function is widely used because of its simplicity and flexibility to represent different types of curves.

In general, polynomial regression can provide linear and non-linear models for example: linear, quadratic, and cubic.

Linear: $f(x) = a_0 + a_1x$
Quadratic: $f(x) = a_0 + a_1x + a_2x^2$
Cubic: $f(x) = a_0 + a_1x + a_2x^2 + a_3x^3$

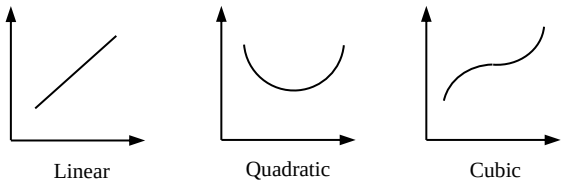

Fig 2: Models of Polynomial Regression

The fundamental concepts of curve fitting are explained in the following section:

Curve Fitting:
Curve fitting is the process of finding the curve that best fits to the given (observed) data points.

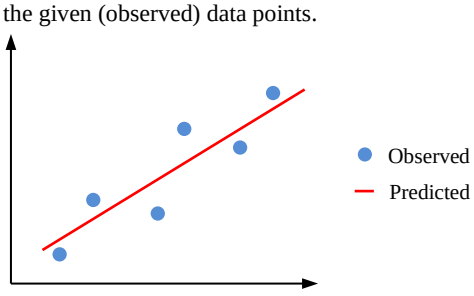

Fig 3: Explanation of Curve Fitting

Polynomial Regression:

The polynomial regression is a mathematical method used to model the relationship between the independent variable (x) and the dependent variable (y) as a polynomial function of degree (n) as shown in the following form:

$$y = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n \qquad (1)$$

where (x) is the independent variable, (y) is the dependent variable, and ($a_0, a_1, \ldots, a_n$) are the coefficients of the polynomial.

By substituting the observed points (x_i, y_i) into formula (1), the following equations are obtained:

$$\begin{aligned} y_0 &= a_0 + a_1 x_0 + a_2 x_0^2 + \ldots + a_n x_0^n \\ y_1 &= a_0 + a_1 x_1 + a_2 x_1^2 + \ldots + a_n x_1^n \\ y_2 &= a_0 + a_1 x_2 + a_2 x_2^2 + \ldots + a_n x_2^n \\ &\vdots \\ y_m &= a_0 + a_1 x_m + a_2 x_m^2 + \ldots + a_n x_m^n \end{aligned}$$

The system of equations can be written in the matrix notation as shown in the following form:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_m^2 & \cdots & x_m^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}$$

which is equivalent to the following equation:

$$Y = M A \qquad (2)$$

where (Y) is the matrix of y-values, (M) is the matrix of x-values powers, and (A) is the matrix of coefficients.

Then, equation (2) is multiplied by the transpose of matrix M:

$$(M^T Y) = (M^T M) A$$

$$L = U A \qquad (3)$$

where (M^T) is the transpose of matrix M, (L) is the result of multiplying M^T by Y, and (U) is the result of multiplying M^T by M.

To solve equation (3), Gauss method is applied to find out the coefficients of matrix A.

Gauss Method:

Gauss method is a famous method in Algebra used to solve a system of equations with (n) unknown variables.

For example, assume the following system:

$$A X = B$$

which can be represented in the following form:

$$\begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & \cdots & a_{0,n} \\ a_{1,0} & a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,0} & a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n,0} & a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

where (A) is the matrix of coefficients, (X) is the matrix of unknown variables, and (B) is the matrix of constants.

Gauss method is performed in two steps: (1) forward elimination and (2) back substitution.

(1) Forward Elimination:

The row operations are applied to reach the upper triangular form as shown in the following form:

$$\begin{bmatrix} \alpha_{0,0} & \alpha_{0,1} & \alpha_{0,2} & \cdots & \alpha_{0,n} \\ 0 & \alpha_{1,1} & \alpha_{1,2} & \cdots & \alpha_{1,n} \\ 0 & 0 & \alpha_{2,2} & \cdots & \alpha_{2,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \alpha_{n,n} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix}$$

The forward elimination is performed by the following algorithm:

Algorithm 1: Forward Elimination

```
for i = 0 to n do
    for j = i+1 to n do
        factor = α_{j,i} / α_{i,i}
        for k = i to n do
            α_{j,k} = α_{j,k} – factor * α_{i,k}
        end for
        β_j = β_j – factor * β_i
    end for
end for
```

(2) Back Substitution:

The unknown variables ($x_0, x_1, \ldots, x_n$) are solved by back substitution using the following formula:

$$x_i = \left(\beta_i - \sum_{j=i+1}^{n} \alpha_{i,j} x_j \right) \Big/ \alpha_{i,i} \qquad (4)$$

The back substitution is performed by the following algorithm:

Algorithm 2: Back Substitution

```
for i = n to 0 do
    sum = 0
    for j = i+1 to n do
        sum = sum + α_{i,j} * x_j
    end for
    x_i = (β_i – sum) / α_{i,j}
end for
```

R-squared:

R-squared (R^2) is a statistical measure used to evaluate the polynomial regression model. It is computed by the following formula:

$$R^2 = 1 - \left(\sum \left(y_i - y_{pi} \right)^2 \Big/ \sum \left(y_i - \bar{y} \right)^2 \right) \qquad (5)$$

where (y_i) is the observed y-value, (y_{pi}) is the predicted y-value, and ($\bar{y}$) is the average of the observed y-values.

The R^2 can take values in the range [0,1], where (0) indicates that the curve does not fit to the data points and (1) indicates that the curve fits to the data points.

Curve Fitting System:

In the curve fitting system:
Input: Observed points (*X*,*Y*).
Output: Predicted Points (*X*,*Yp*).
Processing: The observed points are substituted into the polynomial function to obtain the corresponding equations. Then, matrix (*M*) is computed and the transpose (M^T) is multiplied by the system equation. After that, the system is solved by Gauss method to find out the coefficients and make the polynomial equation. Finally, the predicted points are computed and the curve is plotted.

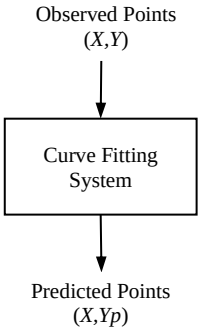

Fig 4: Diagram of Curve Fitting System

Python:

Python [31] is a general high-level programming language. It is simple, easy to learn, and powerful. It is the most preferred programming language by the developers of machine learning applications.

Python provides additional libraries such as: Numpy [32], Pandas [33], Matplotlib [34], NLTK [35], and SK Learn [36].

In this research, the standard functions of Python are applied without using any additional library.

3. RESEARCH METHODOLOGY

The basic steps of curve fitting are: (1) preparing observed points, (2) computing matrix, (3) computing transpose of matrix, (4) multiplying by transpose, (5) performing forward elimination, (6) performing back substitution, (7) finding out coefficients, (8) making polynomial equation, (9) computing predicted points, and (10) plotting curve.

1. Preparing Observed Points (*X*,*Y*)
2. Computing Matrix
3. Computing Transpose of Matrix
4. Multiplying by Transpose
5. Performing Forward Elimination
6. Performing Back Substitution
7. Finding out Coefficients
8. Making Polynomial Equation
9. Computing Predicted Points (*X*,*Yp*)
10. Plotting Curve

Fig 5: Steps of Curve Fitting

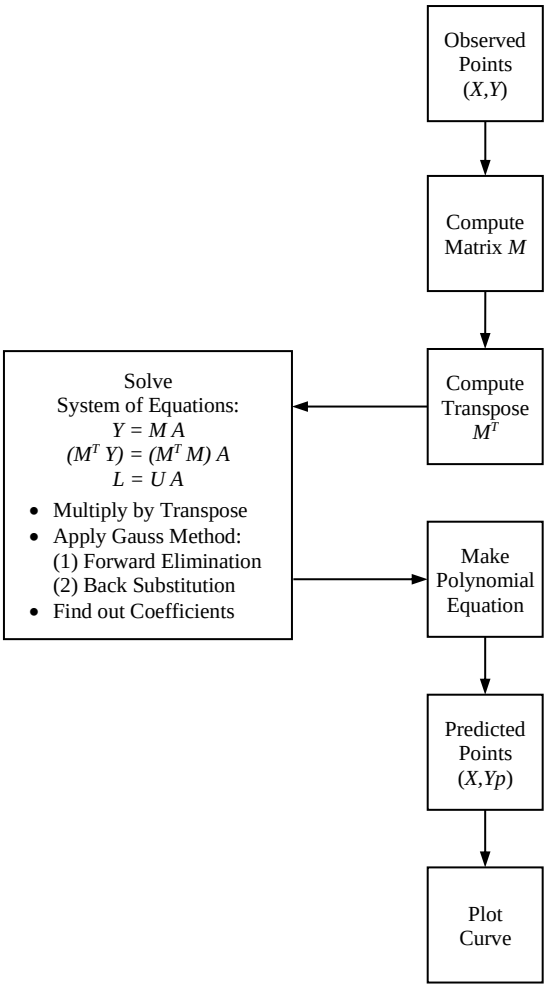

Fig 6: Flowchart of Curve Fitting

The steps of curve fitting are explained in the following section.

1. Preparing Observed Points:

The observed points (*X*,*Y*) are obtained from the original source and converted into lists in the following form:

```
X = [x0, x1, x2, ..., xm]
Y = [y0, y1, y2, ..., ym]
```

2. Computing Matrix:

Matrix (*M*) holds the powers of *x*-values. It is computed by the following code:

```
for i in range(m):
    for j in range(n):
        M[i][j] = X[i]**j
```

3. Computing Transpose of Matrix:

The transpose of matrix *M* (M^T) is computed by the following code:

```
for i in range(n):
    for j in range(m):
        Mt[i][j] = M[j][i]
```

4. Multiplying by Transpose:

Multiplying equation (2) by the transpose to obtain the resulting matrices (*L* and *U*). The matrix multiplication is done by the following code:

```
for i in range(len(a)):
    for j in range(len(b[0])):
        sum = 0
        for k in range(len(a[0])):
            sum += a[i][k]*b[k][j]
        c[i][j] = sum
```

5. Performing Forward Elimination:

The forward elimination is performed by the following code:

```
for i in range(n):
    for j in range(i+1, n):
        factor = a[j][i]/a[i][i]
        for k in range(i, n):
            a[j][k] = a[j][k] - factor*a[i][k]
        b[j] = b[j] - factor*b[i]
```

6. Performing Back Substitution:

The back substitution is performed by the following code:

```
for i in range(n-1, -1, -1):
    sum = 0
    for j in range(i+1, n):
        sum += a[i][j]*x[j]
    x[i] = (b[i] - sum)/a[i][i]
```

7. Finding out Coefficients:

By applying Gauss method, the coefficients are obtained and shown in the following form:

```
A = [a0, a1, a2, ..., an]
```

8. Making Polynomial Equation:

The polynomial equation is obtained and shown in the following form:

```
f(x)= a0 + a1x + a2x^2 + ... + an x^n
```

9. Computing Predicted Points:

The predicted points (*X*,*Yp*) are computed by the following code:

```
for i in range(len(X)):
    Yp[i] = f(X[i])
```

10. Plotting Curve:

The curve is plotted to show the observed points and the predicted curve using the "matplotlib" library. It is done by the following code:

```
import matplotlib.pyplot as plt

plt.scatter(X, Y)
plt.plot(X, Yp, color="red")
```

4. RESULTS AND DISCUSSION

The developed program was tested on an experimental dataset from Kaggle [37]. The program performed the basic steps of curve fitting using polynomial regression and provided the required results. The program output is shown in the following section.

Observed Points:

The observed points (*X*,*Y*) are printed as shown in the following view:

```
        X       Y
------------------------
0       63.456  156.4
1       63.974  172.883
2       64.304  163.108
3       64.732  177.549
4       64.766  167.127
5       64.783  165.612
6       65.117  165.717
7       65.237  181.012
8       65.27   168.618
9       65.279  155.25
10      65.807  163.852
...
```

Matrix:

Matrix (*M*) is printed as shown in the following view:

```
Matrix (M):
0       1.0     63.456
1       1.0     63.974
2       1.0     64.304
3       1.0     64.732
4       1.0     64.766
5       1.0     64.783
6       1.0     65.117
7       1.0     65.237
8       1.0     65.270
9       1.0     65.279
10      1.0     65.807
...
```

Transpose of Matrix:

The transpose of matrix *M* (M^T) is printed as shown in the following view:

```
Transpose of Matrix (Mt):
0       1.0     1.0     1.0     1.0     ...
1       63.456  63.974  64.304  64.732  ...
```

Coefficients:

The coefficients of matrix (*A*) are printed as shown in the following view:

```
Coefficients (A):
-217.201        5.851
```

Polynomial Equation:

The equation of the "linear" polynomial (degree=1) is printed as shown in the following view:

```
f(x)= -217.201 + 5.851x
```

Predicted Points:

The predicted points (*X*,*Yp*) are computed and printed as shown in the following view:

```
        X       Yp
------------------------
0       63.456  154.112
1       63.974  157.143
```

```
2       64.304  159.073
3       64.732  161.576
4       64.766  161.777
5       64.783  161.872
6       65.117  163.832
7       65.237  164.531
8       65.27   164.726
9       65.279  164.779
10      65.807  167.868
...
```

Curve Plot:

The curve of the linear polynomial is plotted as shown in the following chart:

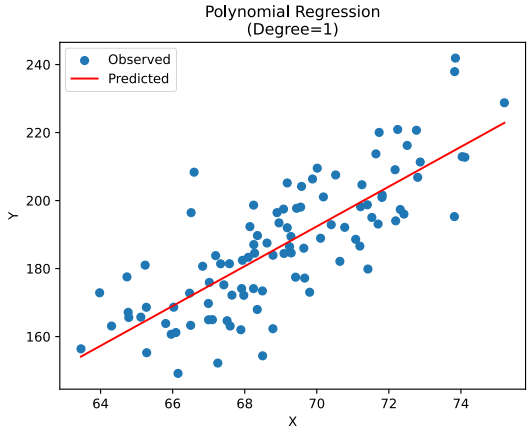

Fig 7: Linear Polynomial Model

The R^2 value is (0.621426) which indicates that the line fits to the data points.

Polynomials of Higher Degrees:

The same work is done for the polynomials of higher degrees. The polynomials of degree (2) and (3) are explained in the following section.

The equation of the "quadratic" polynomial (degree=2) is printed as shown in the following view:

```
f(x)= 1104.672 - 32.402x + 0.276x^2
```

The curve of the quadratic polynomial is plotted as shown in the following chart:

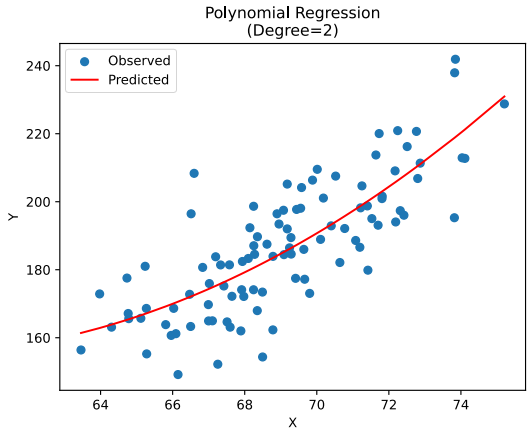

Fig 8: Quadratic Polynomial Model

The R^2 value is (0.634395) which indicates that the quadratic curve fits to the data points better than the linear model.

The equation of the "cubic" polynomial (degree=3) is printed as shown in the following view:

```
f(x)= 9225.315 - 385.296x + 5.382x^2 - 0.025x^3
```

The curve of the cubic polynomial is plotted as shown in the following chart:

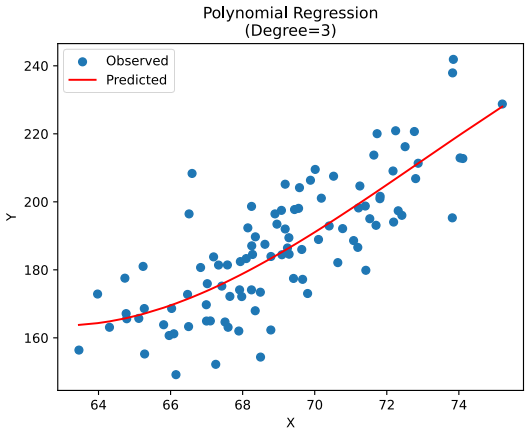

Fig 9: Cubic Polynomial Model

The R^2 value is (0.635398) which indicates that the cubic curve fits to the data points better than the linear and quadratic models.

A comparison of R^2 between the three models of polynomial regression is shown in the following table:

Table 1: Comparison of R^2 between The Three Models

	Polynomial Regression Model		
	Linear (Degree=1)	**Quadratic** (Degree=2)	**Cubic** (Degree=3)
R^2	0.621426	0.634395	0.635398

In summary, the program output clearly shows that the program has successfully performed the basic steps of curve fitting using polynomial regression and provided the required results.

5. CONCLUSION

Curve fitting is one of the important applications in machine learning. It helps to find the curve that best fits the data points. Polynomial regression is used to model the relationship between the independent variable (x) and the dependent variable (y) using a polynomial function of degree (n). Polynomial regression can provide linear and non-linear models.

In this research, the author developed a program to perform curve fitting using polynomial regression in Python. The developed program performed the basic steps of curve fitting: preparing observed points, computing matrix, computing transpose of matrix, multiplying by transpose, performing forward elimination, performing back substitution, finding out coefficients, making polynomial equation, computing predicted points, and plotting curve.

The program was tested on an experimental dataset from Kaggle and provided the required results: matrix, transpose of

matrix, coefficients, polynomial equation, predicted points, and curve plot.

In future work, more research is needed to improve and develop the methods of curve fitting. In addition, they should be more investigated on different fields, domains, and datasets.

6. REFERENCES

[1] Sammut, C., & Webb, G. I. (2011). "Encyclopedia of Machine Learning". Springer Science & Business Media.

[2] Jung, A. (2022). "Machine Learning: The Basics". Singapore: Springer.

[3] Badillo, S., Banfai, B., Birzele, F., Davydov I., Hutchinson, L., Kam-Thong, T., Siebourg-Polster, J., Steiert B., Zhang, J. "An Introduction to Machine Learning". Clinical Pharmacology & Therapeutics, 107(4), 871-885.

[4] Forsyth, D. (2019). "Applied Machine Learning". Cham: Springer International Publishing.

[5] Chopra, D., & Khurana, R. (2023). "Introduction to Machine Learning with Python". Bentham Science Publishers.

[6] Raschka, S. (2015). "Python Machine Learning". Packt Publishing Ltd.

[7] Richert, W., & Coelho, L. (2013). "Building Machine Learning Systems with Python". Packt Publishing Ltd.

[8] Jordan, M. I., & Mitchell, T. M. (2015). "Machine Learning: Trends, Perspectives, and Prospects". Science, 349(6245), 255-260.

[9] Das, S., Dey, A., Pal, A., & Roy, N. (2015). "Applications of Artificial Intelligence in Machine Learning: Review and Prospect". International Journal of Computer Applications, 115(9), 31-41.

[10] Dhall, D., Kaur, R., & Juneja, M. (2020). "Machine Learning: A Review of the Algorithms and its Applications". Proceedings of ICRIC 2019: Recent Innovations in Computing, 47-63.

[11] Sarker, I. H. (2021). "Machine Learning: Algorithms, Real-world Applications and Research Directions". SN Computer Science, 2(3), 160.

[12] Brandt, S. (2014). "Data Analysis: Statistical and Computational Methods for Scientists and Engineers". Cham: Springer.

[13] VanderPlas, J. (2017). "Python Data Science Handbook: Essential Tools for Working with Data". O'Reilly Media.

[14] Johansson, R. (2015). "Numerical Python: A Practical Techniques Approach for Industry". Apress.

[15] Atkinson, K. (1989). "An Introduction to Numerical Analysis". John Wiley & Sons.

[16] Chapra, S. C. (2010). "Numerical Methods for Engineers". McGraw-Hill.

[17] Golub, G. H., & Van Loan, C. F. (2013). "Matrix Computations". JHU Press.

[18] Cassel, K. W. (2021). "Matrix, Numerical, and Optimization Methods in Science and Engineering". Cambridge University Press.

[19] Arlinghaus, S. (2023). "Practical Handbook of Curve Fitting". CRC Press.

[20] Zielesny, A. (2016). "From Curve Fitting to Machine Learning". Berlin Heidelberg: Springer.

[21] Lancaster, P., & Salkauskas, K. (1986). "Curve and Surface Fitting: An Introduction". London: Academic Press.

[22] Guest, P. G. (2012). "Numerical Methods of Curve Fitting". Cambridge University Press.

[23] Motulsky, H., & Christopoulos, A. (2004). "Fitting Models to Biological Data using Linear and Nonlinear Regression: A Practical Guide to Curve Fitting". Oxford University Press.

[24] Fahrmeir, L., Kneib, T., Lang, S., & Marx, B. D. (2021). "Regression: Models, Methods, and Applications". Berlin: Springer.

[25] Peckov, A. (2012). "A Machine Learning Approach to Polynomial Regression". Ljubljana, Slovenia, URL: https://kt.ijs.si/wp-content/uploads/2021/11/phd_ aleksandar_peckov.pdf.

[26] Ostertagová, E. (2012). "Modeling using Polynomial Regression". Procedia Engineering, 48, 500-506.

[27] Gupta, A., Sharma, A., & Goel, A. (2017). "Review of Regression Analysis Models". International Journal of Engineering Research & Technology, 6(08), 58-61.

[28] Kumar, S., & Bhatnagar, V. (2022). "A Review of Regression Models in Machine Learning". Journal of Intelligent Systems and Computing, 3(1), 40-47.

[29] Choksi, B., Venkitaraman, A., & Mali, S. (2017). "Finding Best Fit for Hand-drawn Curves using Polynomial Regression". International Journal of Computer Applications, 174(5), 20-23.

[30] Stigler, S. (1974). "Gergonne's 1815 Paper on The Design and Analysis of Polynomial Regression Experiments". Historia Mathematica, 1(4), 431–439.

[31] Python: https://www.python.org

[32] Numpy: https://www.numpy.org

[33] Pandas: https:// pandas.pydata.org

[34] Matplotlib: https://www. matplotlib.org

[35] NLTK: https://www.nltk.org

[36] SK Learn: https://scikit-learn.org

[37] Kaggle: https://www.kaggle.com