Implementation of Polynomial Regression using Least Squares and Gradient Descent 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 polynomial regression program using least squares and gradient descent in Python. Polynomial regression helps to predict the output data based on the features of the input data using a polynomial function of degree (*n*). Least squares is used to minimize the error between the observed and predicted data. Gradient descent is used to find the optimal solution that provides the minimum value of error function.

The basic steps of polynomial regression using least squares and gradient descent are explained: preparing observed data, computing matrix of features, initializing weights, computing predicted data, computing error function, computing partial derivatives, updating weights, computing final predictions, and plotting predicted data.

The developed program was tested on an experimental dataset. The program successfully performed the basic steps of polynomial regression using least squares and gradient descent and provided the required results.

Keywords
Artificial Intelligence, Machine Learning, Prediction, Polynomial Regression, Least Squares, Mean Squared Error, MSE, Gradient Descent, 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 and efficiency of computer programs [1-12].

Polynomial regression is one of the important applications in machine learning. It is sharing the knowledge between the related fields: machine learning, programming, data science, mathematics, statistics, and numerical methods [13-20].

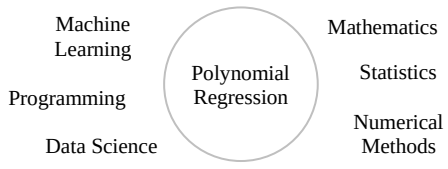

Fig 1: Field of Polynomial Regression

In this paper, polynomial regression is applied using least squares and gradient descent to predict the output data based on the features of the input data. Polynomial regression has a wide range of applications in different fields like industry, business, education, marketing, advertising, medicine, public health, agriculture, environment, climate change, etc.

2. LITERATURE REVIEW
The review of literature revealed the major contributions in the field of polynomial regression using least squares and gradient descent [21-32].

Polynomial regression is an important algorithm in machine learning. It helps to model the relationship between the independent variable (*x*) and the dependent variable (*y*) using a polynomial function of degree (*n*) in the following form:

$$y = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

where: (*y*) is the dependent variable, (*x*) is the independent variable, (a_0, a_1, …, a_n) are the coefficients (or weights) associated with the powers of the independent variable.

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

Linear: $f(x) = a_0 + a_1 x$
Quadratic: $f(x) = a_0 + a_1 x + a_2 x^2$
Cubic: $f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$

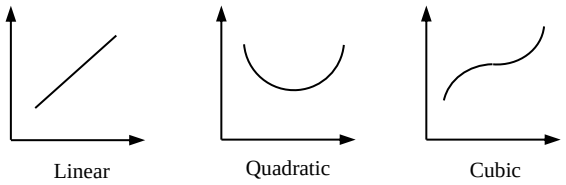

Fig 2: Models of Polynomial Regression

Polynomial Regression:
Polynomial regression is a prediction algorithm used to predict the output data based on the features of the input data. The concept of polynomial regression is illustrated in the following diagram:

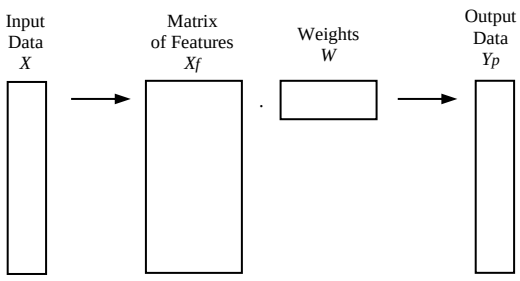

Fig 3: Explanation of Polynomial Regression

The input data (*X*) is transformed to compute the matrix of features (*Xf*). Then, it is processed using the weights (*W*) to predict the output data (*Yp*).

The matrix of features (*Xf*) holds the powers of the input data (*X*). It is represented in the following form:

$$Xf = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m-1} & x_{m-1}^2 & \cdots & x_{m-1}^{n-1} \end{bmatrix}$$

The fundamental concepts of polynomial regression using least squares and gradient descent are explained in the following section.

Least Squares:

Least Squares is a mathematical method used to minimize the error between the observed and predicted data. The error function is defined by the Mean Squared Error (MSE). It is computed by the following formula:

$$\text{MSE} = \left(\frac{1}{m}\right) \sum (y - y_p)^2 \qquad (1)$$

where: (y) is the observed value, (y_p) is the predicted value, and (m) is the number of samples.

The concept of least squares is illustrated in the following diagram:

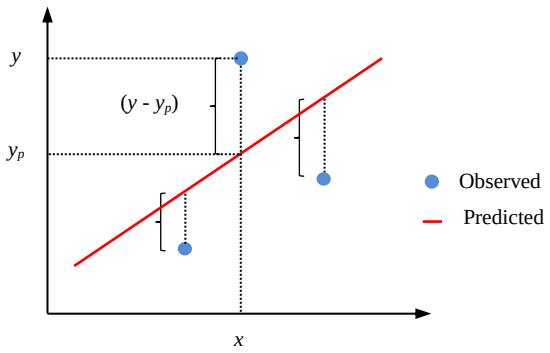

Fig 4: Explanation of Least Squares

Gradient Descent:

Gradient descent is an optimization method used to find the optimal solution that provides the minimum value of error function.

The concept of gradient descent is illustrated in the following diagram:

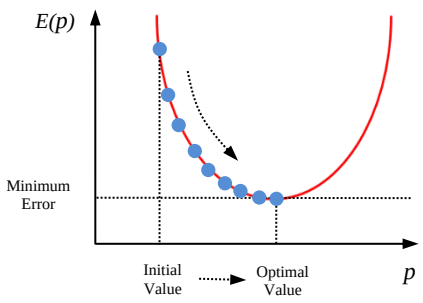

Fig 5: Explanation of Gradient Descent

Gradient descent is an iterative method that starts by giving initial values to the parameters. Then, the partial derivatives of error function with respect to parameters are used to update the parameters. This process continues until the optimal solution is reached to provide the minimal value of error function.

In general, the parameter (p) is updated by the following formula:

$$p_{new} = p_{old} - \alpha \left(\frac{\partial E}{\partial p}\right) \qquad (2)$$

where: (p_{new}) is the new value of parameter, (p_{old}) is the old value of parameter, (α) is the learning rate, ($\frac{\partial E}{\partial p}$) is the partial derivative of error function with respect to parameter.

By using formula (1), the partial derivative of error function (MSE) with respect to weight (w) is given by the following formula:

$$\frac{\partial \text{MSE}}{\partial w} = \left(\frac{-1}{m}\right) \sum (y - y_p)\, x^j \qquad (3)$$

Therefore, the weight (w) is updated by the following formula:

$$w_{new} = w_{old} - \alpha \left(\frac{\partial \text{MSE}}{\partial w}\right) \qquad (4)$$

The steps of gradient descent method are explained in the following algorithm:

Algorithm 1: Gradient Descent Method

```
# initialize weights
W = [0, …]
# learning rate
α = 0.0001
# number of iterations
nt = 1000
for t = 0 to nt do
    # compute predicted data
    yp = ∑(w * x)
    # compute error function
    MSE = (1/m) ∑(y – yp)^2
    # compute partial derivative w.r.t. weights
    dw = (–1/m) ∑(y – yp) x^j
    # update weights
    w = w – α * dw
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:

$$R^2 = 1 - \left(\frac{\sum (y - y_p)^2}{\sum (y - \bar{y})^2}\right) \qquad (5)$$

where: (y) is the observed value, (y_p) is the predicted value, and ($\bar{y}$) is the average of the observed values.

R^2 can take values between (0) and (1), where: (1) indicates that the predicted data fully fits with the observed data and (0) indicates that the predicted data does not fit with the observed data.

Polynomial Regression System:

The polynomial regression system is explained in the following outline:

Input: Observed data (*X, Y*).
Output: Predicted data (*Yp*).
Processing: The observed data is prepared for processing. First, the matrix of features is computed and the weights are initialized to zeros. Then, the predicted data is computed for the current values of weights and the error function is computed. After that, the partial derivatives of error function are computed and the weights are updated. At last, the final predictions are computed and the predicted data is plotted.

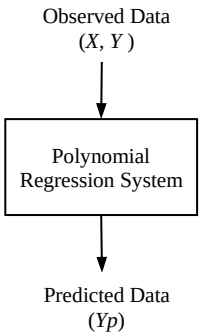

Fig 6: Diagram of Polynomial Regression System

Python:

Python [33] is a general high-level programming language. It is very simple, easy to learn, and powerful. It is the most popular programming language, especially for the development of machine learning applications.

Python provides many additional libraries for different purposes such as Numpy [34], Pandas [35], Matplotlib [36], NLTK [37], SciPy [38], and SK Learn [39].

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

3. RESEARCH METHODOLOGY

The basic steps of polynomial regression are: (1) preparing observed data, (2) computing matrix of features, (3) initializing weights, (4) computing predicted data, (5) computing error function, (6) computing partial derivatives, (7) updating weights, (8) computing final predictions, and (9) plotting predicted data.

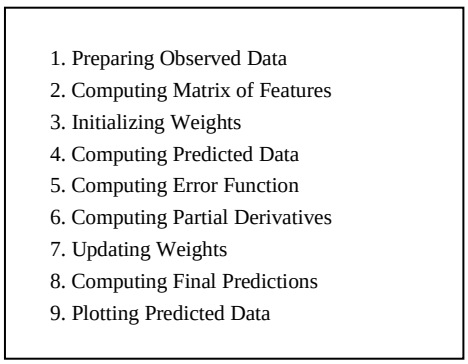

Fig 7: Steps of Polynomial Regression

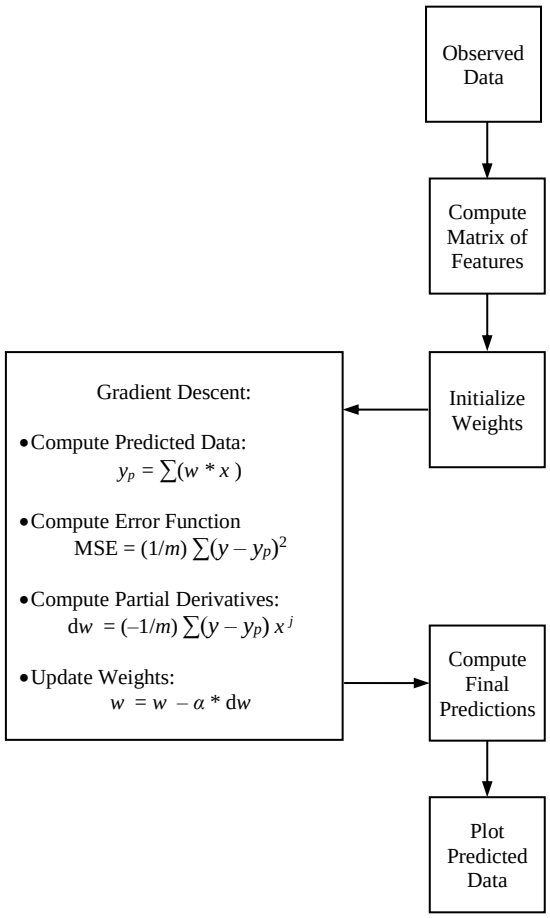

Fig 8: Flowchart of Polynomial Regression

The steps of polynomial regression using least squares and gradient descent are explained in the following section.

1. Preparing Observed Data:

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

$$X = [x_0, x_1, x_2, \ldots, x_{m-1}]$$
$$Y = [y_0, y_1, y_2, \ldots, y_{m-1}]$$

2. Computing Matrix of Features:

The matrix of features (*Xf*) is represented in the following form:

$$Xf = [[1, x_0, x_0^2, \ldots, x_0^{n-1}],$$
$$[1, x_1, x_1^2, \ldots, x_1^{n-1}],$$
$$\ldots$$
$$[1, x_{m-1}, x_{m-1}^2, \ldots, x_{m-1}^{n-1}]]$$

It is computed by the following code:

```
def compute_Xf(X):
    Xf = []
    for i in range(m):
        row = []
        for j in range(n):
            row.append(X[i]**j)
        Xf.append(row)
    return Xf
```

3. Initializing Weights:

The weights (*W*) are initialized to zeros by the following code:

```
W = [0, ...]
```

4. Computing Predicted Data:

The predicted data (*Yp*) is computed by the following code:

```
def compute_Yp(Xf, W):
    Yp = []
    for i in range(m):
        Yp.append(dot(Xf[i], W))
    return Yp
```

5. Computing Error Function:

The error function (MSE) is computed by the following code:

```
def compute_MSE(Y, Yp):
    sum = 0
    for i in range(m):
        sum += (Y[i] - Yp[i])**2
    return sum/m
```

6. Computing Partial Derivatives:

The partial derivatives of error function with respect to weights (*dW*) are computed by the following code:

```
def compute_dW(Xf, Y, Yp):
    Xf_t = transpose(Xf)
    delta = subtract(Y, Yp)
    dW = []
    for i in range(n):
        dW.append((-1/m)*dot(delta, Xf_t[i]))
    return dW
```

7. Updating Weights:

The weights (*W*) are updated by the following code:

```
def update_W(W, alpha, dW):
    W = subtract(W, multiply(alpha, dW))
    return W
```

8. Computing Final Predictions:

The final predicted data (*Yp*) is computed by the following code:

```
Yp = compute_Yp(Xf, W)
```

9. Plotting Predicted Data:

The predicted data (*Yp*) is plotted using the "matplotlib" library. It is done by the following code:

```
import matplotlib.pyplot as plt

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

4. RESULTS AND DISCUSSION

The developed program was tested on an experimental dataset from Kaggle [40]. The program output is explained in the following section.

Observed Data:

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

```
          X              Y
-------------------------------------
0 :      63.45649398    156.3996764
1 :      63.97432572    172.8834702
2 :      64.30418789    163.1080171
3 :      64.7319256     177.5492634
4 :      64.76632913    167.1274611
5 :      64.78258298    165.6116262
6 :      65.11748489    165.7171122
7 :      65.23704952    181.0119732
8 :      65.27034552    168.6177462
9 :      65.27930021    155.2504207
...
```

Matrix of Features:

The matrix of features (*Xf*) is printed as shown in the following view:

```
          Xf
---------------------------
0 :     1.0     63.45649398
1 :     1.0     63.97432572
2 :     1.0     64.30418789
3 :     1.0     64.7319256
4 :     1.0     64.76632913
5 :     1.0     64.78258298
6 :     1.0     65.11748489
7 :     1.0     65.23704952
8 :     1.0     65.27034552
9 :     1.0     65.27930021
...
```

Processing Gradient Descent:

The gradient descent method is processed (1,000) iterations. For each iteration; the current value of error function (MSE) is printed as shown in the following view:

```
t         MSE
---------------------------
0         35280.73598330422
100       139.632191285907
200       139.6321664935802
300       139.63216649358026
400       139.63216649358026
500       139.63216649358026
600       139.63216649358026
700       139.63216649358026
800       139.63216649358026
900       139.63216649358026
```

Error Function Plot:

The error function (MSE) is plotted as shown in the following chart:

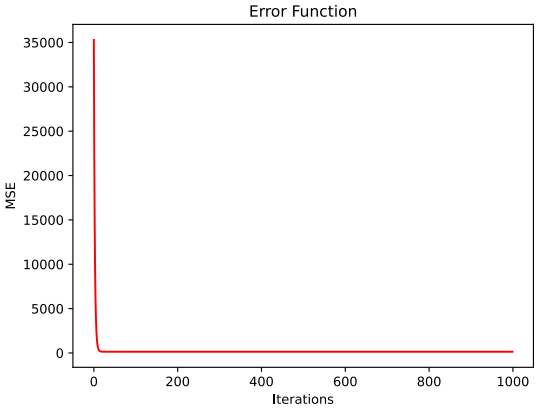

Fig 9: Error Function Plot

The plot shows that the error function is decreasing with iterations which indicates that the polynomial regression model is converging to the optimal solution.

Final Weights:

The final values of weights (*W*) are printed as shown in the following view:

```
Weights (W):
w0 :     186.84726033099986
w1 :     15.139521908242594
```

Final Predictions:

The final predicted data (*Yp*) is printed as shown in the following view:

```
          Y                Yp
-----------------------------------------
0 :       156.3996764      154.1124867702328
1 :       172.8834702      157.14256355063677
2 :       163.1080171      159.0727420716572
3 :       177.5492634      161.57563635769336
4 :       167.1274611      161.77694757399104
5 :       165.6116262      161.87205648615847
6 :       165.7171122      163.83172489351398
7 :       181.0119732      164.53135371333212
8 :       168.6177462      164.72618425129866
9 :       155.2504207      164.77858234926907
...
```

Predicted Data Plot:

The final predicted data (*Yp*) is plotted as shown in the following chart:

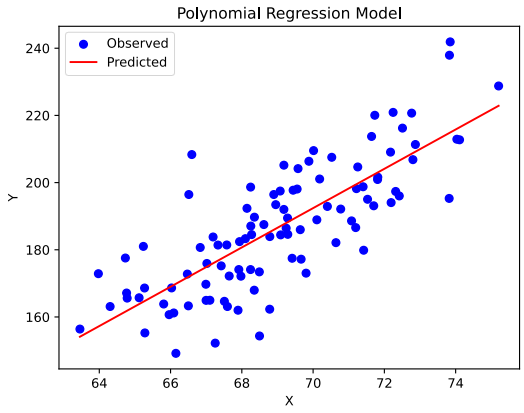

Fig 10: Polynomial Regression Model (Degree=1)

The plot shows that the predicted data (*Yp*) is strongly related to the observed data (*Y*).

R-Squared:

The R-squared (R^2) of the polynomial regression model is printed as shown in the following view:

```
R2 = 0.621426
```

The R^2 value is accepted which indicates that the predicted data fits with the observed data in about (%62) of the samples.

Plotting Higher Degrees:

The plots of higher degrees (quadratic and cubic) are shown in the following charts:

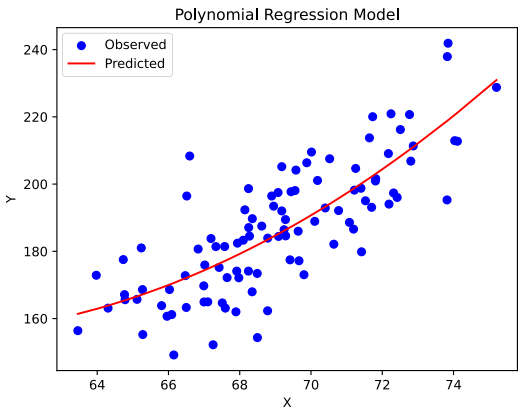

Fig 11: Polynomial Regression Model (Degree=2)

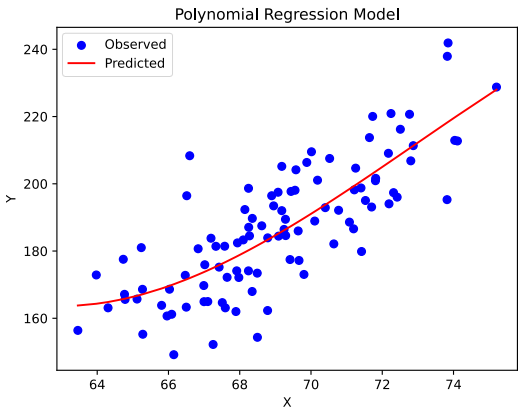

Fig 12: Polynomial Regression Model (Degree=3)

In summary, the program output shows that the program successfully performed the basic steps of polynomial regression using least squares and gradient descent and provided the required results.

5. CONCLUSION

Machine learning is playing a major role in the development of computer systems. It helps to improve the performance and efficiency of computer programs.

Polynomial regression is one of the important applications in machine learning. It helps to predict the output data based on the features of the input data. Polynomial regression is applied

using least squares and gradient descent. Least squares is used to minimize the error between the observed and predicted data. Gradient descent is used to find the optimal solution that provides the minimum value of error function.

In this research, the author developed a program to perform polynomial regression using least squares and gradient descent in Python. The basic steps of polynomial regression are: preparing observed data, computing matrix of features, initializing weights, computing predicted data, computing error function, computing partial derivatives, updating weights, computing final predictions, and plotting predicted data.

The program was tested on an experimental dataset and provided the required results: matrix of features, predicted data, error function, weights, and final predictions.

In future work, more research is needed to improve and develop the current methods of polynomial regression using least squares and gradient descent. 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] Kubat, M. (2021). "An Introduction to Machine Learning". Cham, Switzerland: Springer.

[4] Mohammed, M., Khan, M. B., & Bashier, E. B. M. (2016). "Machine Learning: Algorithms and Applications". Crc Press.

[5] Dey, A. (2016). "Machine Learning Algorithms: A Review". International Journal of Computer Science and Information Technologies, 7 (3), 1174-1179.

[6] Bonaccorso, G. (2018). "Machine Learning Algorithms: Popular Algorithms for Data Science and Machine Learning". Packt Publishing.

[7] Jo, T. (2021). "Machine Learning Foundations: Supervised, Unsupervised, and Advanced Learning". Springer.

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

[9] Müller, A. C., & Guido, S. (2016). "Introduction to Machine Learning with Python: A Guide for Data Scientists". O'Reilly Media.

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

[11] Forsyth, D. (2019). "Applied Machine Learning". Cham, Switzerland: Springer.

[12] Sarkar, D., Bali, R., & Sharma, T. (2018). "Practical Machine Learning with Python". Apress.

[13] Holmes, M. H. (2023). "Introduction to Scientific Computing and Data Analysis". Springer Nature.

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

[15] Igual, L., & Seguí, S. (2024). "Introduction to Data Science: A Python Approach to Concepts, Techniques and Applications". Springer Nature.

[16] Qamar, U., & Raza, M. S. (2023). "Data Science Concepts and Techniques with Applications". Berlin/Heidelberg, Germany: Springer.

[17] Aggarwal, C. C. (2024). "Probability and Statistics for Machine Learning: A Textbook". Cham, Switzerland: Springer.

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

[19] James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). "An Introduction to Statistical Learning: With Applications in Python". Springer Nature.

[20] Kong, Q., Siauw, T., & Bayen, A. (2020). "Python Programming and Numerical Methods: A Guide for Engineers and Scientists". Academic Press.

[21] Peckov, A. (2012). "A Machine Learning Approach to Polynomial Regression". Ljubljana, Slovenia.

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

[23] Groß, J. (2003). "Linear Regression". Springer Science & Business Media.

[24] Olive, D. J. (2017). "Linear Regression". Berlin, Germany: Springer.

[25] Yan, X., & Su, X. (2009). "Linear Regression Analysis: Theory and Computing". World Scientific.

[26] Schroeder, L. D., Sjoquist, D. L., & Stephan, P. E. (2016). "Understanding Regression Analysis: An Introductory Guide". Sage Publications.

[27] Montgomery, D.C., Peck, E.A., Vining G. G. (20012). "Introduction to Linear Regression Analysis". Wiley Series in Probability and Statistics: John Wiley & Sons.

[28] Kutner, N., Nachtsheim, C., & Neter, J. (2004). "Applied Linear Regression Models". McGraw-Hill/Irwin Series: Operations and Decision Sciences.

[29] Seber, G. A., & Lee, A. J. (2003). "Linear Regression Analysis". John Wiley & Sons.

[30] Leemis, L.M. (1991). "Applied Linear Regression Models". Journal of Quality Technology, 23, 76-77.

[31] Weisberg, S. (2005). "Applied Linear Regression". John Wiley & Sons.

[32] Massaron, L., & Boschetti, A. (2016). "Regression Analysis with Python". Packt Publishing.

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

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

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

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

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

[38] SciPy: https://scipy.org

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

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