Implementation of Logistic Regression using 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 logistic regression program using gradient descent in Python. Logistic regression helps to classify data into categories based on the features of samples. Sigmoid function is used to transform values into probabilities and predict the required categories. Gradient descent is used to find the optimal solution that provides the minimum value of error function.

The basic steps of linear regression using gradient descent are explained: preparing actual data, initializing weights and bias, computing predicted data, applying sigmoid function, computing cost function, computing partial derivatives, updating weights and bias, computing final prediction, computing confusion matrix, and computing statistical measures.

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

Keywords

Artificial Intelligence, Machine Learning, Classification, Logistic Regression, Sigmoid Function, Gradient Descent, Confusion Matrix, 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 studying computer algorithms and methods to improve the performance of computer programs.

Logistic regression is one of the important applications of machine learning. It is a common area that is sharing knowledge from various fields such as machine learning, programming, data science, mathematics, statistics, and numerical methods [1-12, 13-15].

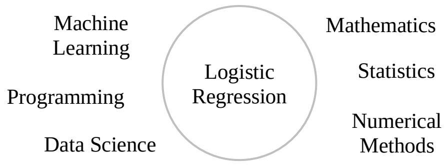

Fig 1: Field of Logistic Regression

In this research, logistic regression is applied to classify data into categories based on the features of samples. Gradient descent is used to find the optimal solution that provides the minimum value of error function. Logistic regression has a wide range of applications in many fields like: business, finance, marketing, education, medicine, public health, etc.

2. LITERATURE REVIEW

The review of literature revealed the major contributions in the field of logistic regression using gradient descent [16-28].

Logistic regression is an important algorithm in machine learning. It was first discovered by the Belgian mathematician Verhulst (1838) to predict the growth of population [29]. Then, it was rediscovered by Pearl and Reed in 1920 to model the growth of population in the United States [30].

Logistic regression is used to model the relationship between the independent variables (features) and the dependent variable (prediction). For example, it is used to predict whether patients are likely to have heart disease based on their features (age, sex, weight, blood pressure, cholesterol, etc).

The fundamental concepts of logistic regression using gradient descent are explained in the following section.

Logistic Regression:

Logistic regression is a prediction algorithm used to classify data into categories based on the features of samples. The process of logistic regression is performed iteratively as shown in the following explanation:

The input data (X), which contains the features of samples, is processed using weights (W) and bias (b). Then, sigmoid function is applied to transform the resulting values into probabilities in the range [0,1]. After that, the probabilities are used to predict the required categories of the output data (Yp).

The concept of logistic regression is illustrated as shown in the following diagram:

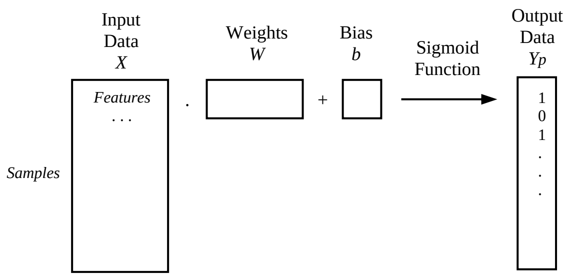

Fig 2: Concept of Logistic Regression

Sigmoid function is used to transform values into probabilities between (0) and (1). The sigmoid function is defined by the following formula:

$$f(x) = \frac{1}{1 + e^{-x}} \qquad (1)$$

The graph of sigmoid function is shown in the following diagram:

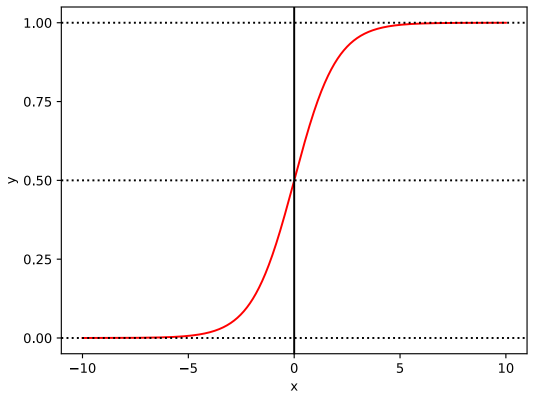

Fig 3: Graph of Sigmoid Function

The resulting values of sigmoid function are compared to a threshold (0.5). If the value is above threshold, it gives (1), and if the value is below threshold, it gives (0).

Therefore, sigmoid function is used to predict the required category, for example: category (0) or category (1). Prediction is performed by the following formula:

$$f(x) = \begin{cases} 1, & y > 0.5 \\ 0, & y \le 0.5 \end{cases} \qquad (2)$$

The logistic regression is using gradient descent method to minimize the error function. It is explained in the following section.

Gradient Descent:

Gradient descent is an optimization method used to find the optimal solution that provides the minimum value of error (or cost) function. The cost function is defined as shown in the following formula:

$$\text{Cost} = \left(\frac{-1}{m}\right) \sum \left(y \log(y_p) + (1 - y) \log(1 - y_p) \right) \qquad (3)$$

Where: (y) is the actual y-value, (y_p) is the predicted y-value, and (m) is the number of samples.

The partial derivative of cost function with respect to weights (W) is defined as shown in the following formula:

$$\frac{\partial \text{Cost}}{\partial W} = \left(\frac{-1}{m}\right) \sum (y - y_p)\,(x) \qquad (4)$$

The partial derivative of cost function with respect to bias (b) is defined as shown in the following formula:

$$\frac{\partial \text{Cost}}{\partial b} = \left(\frac{-1}{m}\right) \sum (y - y_p) \qquad (5)$$

Updating weights (W) and bias (b) is an iterative process. The weights (W) is updated as shown in the following formula:

$$W_{new} = W_{old} - \alpha \left(\frac{\partial \text{Cost}}{\partial W}\right) \qquad (6)$$

Where: (W_{new}) is the new value of weights, (W_{old}) is the old value of weights, (α) is the learning rate, and ($\frac{\partial \text{Cost}}{\partial W}$) is the partial derivative of cost function with respect to weights.

The bias (b) is updated as shown in the following formula:

$$b_{new} = b_{old} - \alpha \left(\frac{\partial \text{Cost}}{\partial b}\right) \qquad (7)$$

Where: (b_{new}) is the new value of bias, (b_{old}) is the old value of bias, (α) is the learning rate, and ($\frac{\partial \text{Cost}}{\partial b}$) is the partial derivative of cost function with respect to bias.

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

Algorithm 1: Logistic Regression using Gradient Descent

```
# initialize weights
$W$ = [0, 0, …]
# initialize bias
$b = 0$
# learning rate
$\alpha = 0.0001$
# number of iterations
$Nt = 1000$
for $t = 0$ to $Nt$ do
    # compute predicted data
    $Yp = X \,.\, W + b$
    # compute sigmoid function
    $Yp = 1/(1 + exp(- y_p))$
    # compute cost function
    Cost = $(-1/m)\ \Sigma\ ((y \log(y_p) + (1 - y) \log(1 - y_p))$
    # compute partial derivative w.r.t. weights
    d$W$ = $(-1/m) * (Y - Yp) \,.\, X$
    # compute partial derivative w.r.t. bias
    db = $(-1/m) * (Y - Yp)$
    # update weights
    $W = W - \alpha *$ (d$W$)
    # update bias
    $b = b - \alpha *$ (db)
end for
```

Confusion Matrix:

Confusion matrix is a statistical table used to summarize the results of the logistic regression model. It is represented as shown in the following diagram.

		Predicted	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

Fig 4: Representation of Confusion Matrix

Where:
TP: is the number of true positives.

TN: is the number of true negatives.
FP: is the number of false positives.
FN: is the number of false negatives.

Statistical Measures:

The statistical measures: Accuracy, Precision, Recall, and F1-Score are used to evaluate the logistic regression model. They are defined as the following:

Accuracy: is the ratio of true predictions to the total predictions.
Precision: is the ratio of true positive predictions to the positive predictions.
Recall: is the ratio of true positive predictions to the actual positive instances.
F1-Score: is the harmonic mean of precision and recall.

They are computed by the following formulas:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (8)$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (9)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (10)$$

$$\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (11)$$

Logistic Regression System:

In the logistic regression system:
Input: Actual data (X, Y).
Output: Predicted data (Yp).
Processing: The actual data is obtained and prepared for processing. Then, the weights and bias are used to compute the predicted data and cost function. After that, the partial derivatives of cost function with respect to wights and bias are used to update the values of weights and bias. The final weights and bias are used to compute the final prediction.

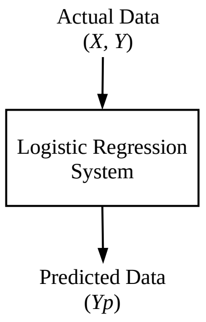

Fig 5: Diagram of Logistic Regression System

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 many additional libraries for different purposes such as: Numpy [32], Pandas [33], Matplotlib [34], NLTK [35], SciPy [36], and SK Learn [37].

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

3. RESEARCH METHODOLOGY

The basic steps of logistic regression are: (1) preparing actual data, (2) initializing weights and bias, (3) computing predicted data, (4) applying sigmoid function, (5) computing cost function, (6) computing partial derivatives, (7) updating weights and bias, (8) computing final prediction, (9) computing confusion matrix, and (10) computing statistical measures.

1. Preparing Actual Data (X, Y)
2. Initializing Weights and Bias
3. Computing Predicted Data (Yp)
4. Applying Sigmoid Function
5. Computing Cost Function
6. Computing Partial Derivatives
7. Updating Weights and Bias
8. Computing Final Prediction
9. Computing Confusion Matrix
10. Computing Statistical Measures

Fig 6: Steps of Logistic Regression

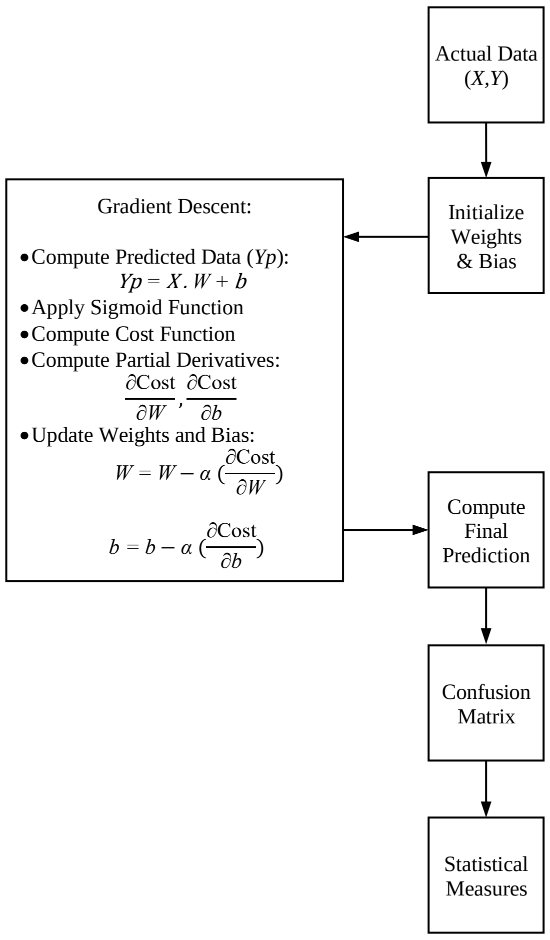
Actual Data (X,Y)
Initialize Weights & Bias
Gradient Descent:
• Compute Predicted Data (Yp): Yp = X . W + b
• Apply Sigmoid Function
• Compute Cost Function
• Compute Partial Derivatives: ∂Cost/∂W , ∂Cost/∂b
• Update Weights and Bias: W = W − α (∂Cost/∂W)
b = b − α (∂Cost/∂b)
Compute Final Prediction
Confusion Matrix
Statistical Measures

Fig 7: Flowchart of Logistic Regression

The steps of logistic regression are explained in details in the following section.

1. Preparing Actual Data:
The actual data (*X*, *Y*) is obtained from the original source and converted into lists in the following form:

```
X = [[x0,0, x0,1, x0,2, ..., x0,n],
     [x1,0, x1,1, x1,2, ..., x1,n],
     [x2,0, x2,1, x2,2, ..., x2,n],
     ...
     [xm,0, xm,1, xm,2, ..., xm,n]]

Y = [y0, y1, y2, ..., ym]
```

2. Initializing Weights and Bias:
The weights (*W*) and bias (*b*) are initialized to zeros by the following code:

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

3. Computing Predicted Data:
The predicted data (*Yp*) is computed by the following code:

```
for i in range(m):
    Yp[i] = dot(X[i], W) + b
```

4. Applying Sigmoid Function:
The sigmoid function is computed by the following code:

```
def sigmoid(x):
    return 1/(1 + math.exp(-x))
```

5. Computing Cost Function:
The cost function is computed by the following code:

```
def compute_cost(Y, Yp):
    sum = 0
    for i in range(m):
        sum += (Y[i]*math.log(Yp[i]) +
               (1 - Y[i])*math.log(1 - Yp[i]))
    return (-1/m)*sum
```

6. Computing Partial Derivatives:
The partial derivative of cost function with respect to weights is computed by the following code:

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

The partial derivative of cost function with respect to bias is computed by the following code:

```
def compute_db(Y, Yp):
    delta = subtract(Y, Yp)
    return (-1/m)*sum(delta)
```

7. Updating Weights and Bias:
The weights (*W*) and bias (*b*) are updated by the following code:

```
W = subtract(W, multiply(alpha, dW))
b = b - alpha * db
```

8. Computing Final Prediction:
The final prediction is computed by the following code:

```
for i in range(m):
    if (Yp[i] > 0.5):
        Yp[i] = 1
    else:
        Yp[i] = 0
```

9. Computing Confusion Matrix:
The confusion matrix is computed by the following code:

```
def confusion(Y, Yp):
    tp = 0
    tn = 0
    fp = 0
    fn = 0
    for i in range(m):
        if (Y[i] == 1 and Yp[i] == 1):
            tp += 1
        elif (Y[i] == 0 and Yp[i] == 0):
            tn += 1
        elif (Y[i] == 0 and Yp[i] == 1):
            fp += 1
        elif (Y[i] == 1 and Yp[i] == 0):
            fn += 1
    return tp, tn, fp, fn
```

10. Computing Statistical Measures:
The statistical measures: Accuracy, Precision, Recall, and F1-Score are computed by the following code:

```
accuracy  = (tp + tn)/(tp + tn + fp + fn)
precision = tp/(tp + fp)
recall    = tp/(tp + fn)
f1_score  = (2*precision*recall)/(precision +
             recall)
```

4. RESULTS AND DISCUSSION
The developed program was tested on an experimental dataset from Kaggle [38]. The program performed the basic steps of logistic regression using gradient descent and provided the required results. The program output is shown in the following section.

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

```
x0      x1      x2      x3      x4      x5      Y
-------------------------------------------------
0       1       3       2       108     1.5     0
0       1       2       3       129     2.6     0
0       0       2       2       160     3.6     0
0       0       1       3       147     1.4     0
0       1       0       3       155     3.1     0
2       1       1       1       142     0.6     0
1       0       0       3       168     1       0
1       0       0       2       160     1.8     0
2       0       2       3       173     3.2     0
...
```

Processing Gradient Descent:
The gradient descent method is processed (1,000) iterations. For each iteration, the value of cost function is printed as shown in the following view:

```
t        Cost
--------------------------------
0        0.693147180559946
```

```
100       0.690158485943933
200       0.6872036584683299
300       0.6842823066923432
400       0.6813940413812531
500       0.6785384756042125
600       0.6757152248273213
700       0.6729239070020548
800       0.6701641426491037
900       0.6674355549377053
```

Cost Function Plot:

The cost function is plotted as shown in the following chart:

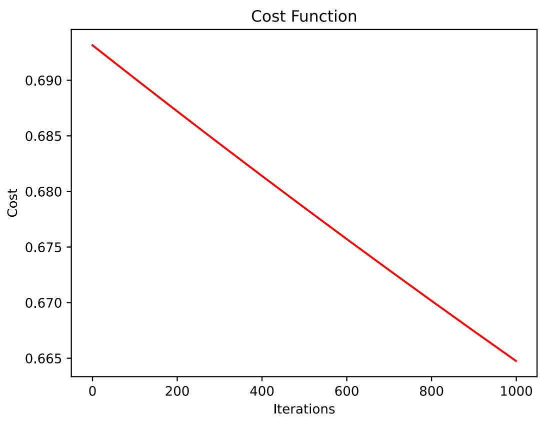

Fig 8: Cost Function Plot

The plot shows that the cost function is decreasing with iterations and the logistic regression model is converging to the optimal solution.

Final Weights and Bias:

The final values of weights (*W*) and bias (*b*) are shown in the following view:

```
Weights (W) =
0         0.025679475502280467
1         -0.018380288886279864
2         -0.024324854979565193
3         -0.024930746578287415
4         0.018749045520454904
5         -0.016679616059594417

Bias (b) = -1.446357367098013e-07
```

Final Prediction:

The final prediction is plotted as shown in the following chart:

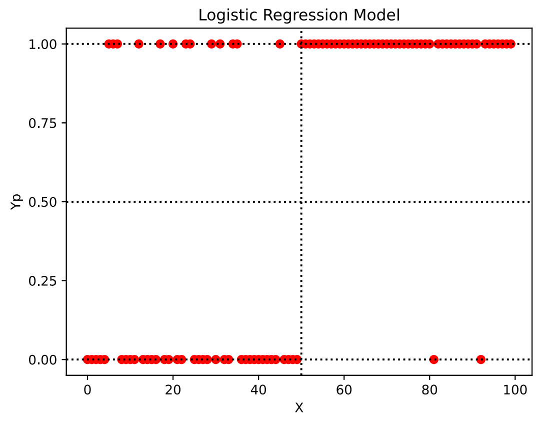

Fig 9: Logistic Regression Model

The plot shows that the logistic regression model has successfully classified data into two categories: (0) and (1).

Confusion Matrix:

The values of confusion matrix are printed as shown in the following view:

```
TP    = 48
TN    = 37
FP    = 13
FN    = 2
Total = 100
```

Then, the confusion matrix of the logistic regression model is represented as shown in the following diagram:

		Predicted	
		Positive	Negative
Actual	Positive	48	2
	Negative	13	37

Fig 10: Confusion Matrix of Model

Statistical Measures:

The statistical measures: Accuracy, Precision, Recall, and F1-Score are printed as shown in the following view:

```
Accuracy  = 0.85
Precision = 0.7868852459016393
Recall    = 0.96
F1-Score  = 0.8648648648648649
```

The accuracy of the logistic regression model is (0.85) which means that the logistic regression model has correctly predicted (85%) of the samples.

In summary, the program output clearly shows that the program has successfully performed the basic steps of logistic regression using gradient descent and provided the required results.

5. CONCLUSION

Logistic regression is one of the important applications in machine learning. It helps to classify data into categories based on the features of samples. Sigmoid function is used to transform values into probabilities and predict the required categories. 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 logistic regression using gradient descent in Python. The developed program performed the basic steps of logistic regression using gradient descent: preparing actual data, initializing weights and bias, computing predicted data, applying sigmoid function, computing cost function, computing partial derivatives, updating weights and bias, computing final prediction, computing confusion matrix, and computing statistical measures.

The program was tested on an experimental dataset from Kaggle and provided the required results: predicted data, cost function, partial derivatives, updated weighs and bias, final prediction, confusion matrix, and statistical measures.

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

6. REFERENCES

[1] Sammut, C., & Webb, G. I. (2017). "Encyclopedia of Machine Learning and Data Mining". New York: Springer.

[2] Shalev-Shwartz, S., & Ben-David, S. (2014). "Understanding Machine Learning: From Theory to Algorithms". Cambridge University Press.

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

[4] 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.

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

[6] Muller, A. C., & Guido S. (2017). "Introduction to Machine Learning with Python". O'Reilly Media.

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

[8] Giussani, A. (2019). "Applied Machine Learning with Python". EGEA spa.

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

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

[11] Mathur, P. (2019). "Machine Learning Applications using Python: Cases Studies from Healthcare, Retail, and Finance". Apress.

[12] Hetland, M. L. (2014). "Python Algorithms: Mastering Basic Algorithms in the Python Language". Apress.

[13] Dangeti, P. (2017). "Statistics for Machine Learning". Packt Publishing.

[14] VanderPlas, J. (2017). "Python Data Science Handbook". O'Reilly Media.

[15] McKinney, W. (2018). "Python for Data Analysis". O'Reilly Media.

[16] Kleinbaum, D. G., & Klein, M. (2010). "Logistic Regression: A Self-Learning Text". New York: Springer.

[17] Hilbe, J. M. (2009). "Logistic Regression Models". CRC Press.

[18] Menard, S. (2002). "Applied Logistic Regression Analysis". Sage Publications.

[19] Hosmer Jr, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). "Applied Logistic Regression". John Wiley & Sons.

[20] Menard, S. W. (2010). "Logistic Regression: From Introductory to Advanced Concepts and Applications. Sage.

[21] Nick, T. G., & Campbell, K. M. (2007). "Logistic Regression". Topics in Biostatistics, 273-301.

[22] Osborne, J. W. (2014). "Best Practices in Logistic Regression". Sage Publications.

[23] Garson, G. D. (2014). "Logistic Regression: Binary and Multinomial". Asheboro, NC.

[24] Kleinbaum, D. G., & Klein, M. (2010). "Logistic Regression, Statistics for Biology and Health".

[25] Harris, J.K. (2021). "Primer on Binary Logistic Regression". Family Medicine and Community Health, 9(1).

[26] Boateng, E.Y., & Abaye, D.A. (2019). "A Review of the Logistic Regression Model with Emphasis on Medical Research". Journal of Data Analysis and Information Processing. 7, 190-207.

[27] Shipe, M.E., Deppen, S.A., Farjah, F., & Grogan, E.L. (2019). "Developing Prediction Models for Clinical Use using Logistic Regression: An Overview". Journal of thoracic disease, 11(4), S574-S584 .

[28] Phillips, J.M. (2021). "Gradient Descent". In: Mathematical Foundations for Data Analysis. Springer Series in the Data Sciences. Cham: Springer.

[29] Verhulst, P. F. (1838). "Notice sur la loi que la population poursuit dans son accroissement". Correspondance Mathématique et Physique. 10: 113–121.

[30] Pearl, R.; Reed, L. J. (1920). "On the Rate of Growth of the Population of the United States since 1790 and Its Mathematical Representation". Proceedings of the National Academy of Sciences. 6 (6): 275–288.

[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] SciPy: https://scipy.org

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

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