Characterization of load tests in web applications through polynomial regression

Caracterización de pruebas de carga en aplicaciones web mediante regresión polinomial

DOI: https://dx.doi.org/10.17981/ingecuc.21.1.2025.07

Original Research.
Date Received: 20/05/2022, Date Accepted: 24/03/2024.

Gabriel Elías Chanchí-Golondrino
Universidad de Cartagena - Research group DaTos. Cartagena, (Colombia)
gchanchig@unicartagena.edu.co

Manuel Alejandro Ospina-Alarcón
Universidad de Cartagena - Research group DaTos. Cartagena, (Colombia)
mospinaa@unicartagena.edu.co

Mónica Esther Ospino-Pinedo
Universidad de Cartagena - Research group E-soluciones. Cartagena, (Colombia)
mospinop@unicartagena.edu.co

To cite this paper
G. Chanchí-Golondrino, M. Ospina-Alarcón & M. Ospino-Pinedo "Characterization of load tests in web applications through polynomial regression," *INGE CUC*, vol. 21, no. 1, 2025. DOI: https://dx.doi.org/10.17981/ingecuc.21.1.2025.07

Resumen

Introducción: Uno de los retos de la infraestructura que soporta aplicaciones web es garantizar la calidad de servicio (QoS), por lo que es necesario caracterizar el desempeño de esta infraestructura para la planificación y configuración tanto de la capacidad del sistema como de la gestión de recursos. Es así como uno de los métodos más destacados para evaluar el rendimiento de la infraestructura de aplicaciones web son las pruebas de carga o estrés, que buscan determinar el comportamiento de un sitio web o servicio web en términos de la cantidad de solicitudes secuenciales o simultáneas que emulan conexiones de clientes reales.
Objetivo: En este artículo, proponemos una herramienta que aporta a la caracterización de pruebas de carga o esfuerzo de aplicaciones web mediante el uso de técnicas de machine learning y específicamente mediante el uso de modelos de aprendizaje supervisado asociados al método de regresión polinomial.
Metodología: Se desarrolló mediante el patrón iterativo de Pratt una herramienta basada en librerías de software libre y tecnologías del lenguaje Python, que admite tanto la ejecución de las pruebas de carga, como la determinación del grado del polinomio (2 o 3) permitiendo caracterizar la curva con el tiempo que tarda el servidor en responder o atender un determinado porcentaje de solicitudes, después de detectar valores atípicos en los datos capturados.
Resultados: Utilizando la herramienta propuesta, se realizó un caso de estudio en la página web de la Universidad de Cartagena para verificar la funcionalidad y utilidad del enfoque propuesto en el que se ejecutaron un total de 50 solicitudes secuenciales, a partir de las cuales la herramienta determinó un modelo polinomial de grado 3 que obtuvo como resultado para el conjunto de entrenamiento un valor de RMSE de 0.992 y un valor de R2 de 0.994, mientras que para el conjunto de prueba se obtuvo un valor de RMSE de 10.447 y un valor de R2 de 0.744.
Conclusiones: El algoritmo representa un aporte en relación con los trabajos del estado del arte revisados y respecto a diferentes herramientas para la ejecución de pruebas de carga, que en términos generales no utilizan métodos de aprendizaje automático para modelar el desempeño de la infraestructura que soporta las aplicaciones web, limitándose a la ejecución de pruebas secuenciales y concurrentes. Específicamente, el enfoque propuesto en este artículo permitió determinar un polinomio que caracteriza la relación matemática entre el porcentaje de solicitudes atendidas y el tiempo de respuesta obtenido por el servidor web.

Palabras clave
Pruebas de carga; Aprendizaje automático; Regresión polinomial; Aplicaciones web.

Abstract

Introduction: One of the challenges of the infrastructure that supports web applications is to guarantee the quality of service (QoS), so it is necessary to characterize the performance of this infrastructure for the planning and configuration of both system capacity and resource management. This is how one of the most outstanding methods to evaluate the performance of the web application infrastructure are load or stress tests, which seek to determine the behavior of a website or web service in terms of the number of sequential or simultaneous requests that it receives. emulate real client connections.
Objective: In this article, we propose a tool that contributes to the characterization of load or stress tests of web applications through the use of machine learning techniques and specifically through the use of supervised learning models associated with the polynomial regression method.
Method: Using Pratt's iterative pattern, a tool based on free software libraries and Python language technologies was developed, which supports both the execution of load tests and the determination of the degree of the polynomial (2 or 3), allowing the curve to be characterized with the time it takes for the server to respond or service a certain percentage of requests, after detecting outliers in the captured data.
Results: Using the proposed tool, a case study was carried out on the website of the University of Cartagena to verify the functionality and usefulness of the proposed approach in which a total of 50 sequential requests were executed, from which the tool determined a polynomial model of degree 3 that obtained as a result for the training set an RMSE value of 0.992 and an R2 value of 0.994, while for the test set an RMSE value of 10.447 and an R2 value of 0.744 were obtained.
Conclusions: The algorithm represents a contribution regarding to the reviewed state of the art works and respect to different tools for the execution of load tests, which in general terms do not use machine learning methods to model the performance of the infrastructure that supports the applications. web, limiting itself to the execution of sequential and concurrent tests. Specifically, the approach proposed in this article allowed to determine a polynomial that characterizes the mathematical relationship between the percentage of requests served and the response time obtained by the web server.

Keywords
Load tests; Machine learning; Polynomial regression; Web applications.

© The author; licensee Universidad de la Costa - CUC.

I. INTRODUCTION

With the evolution of the Internet, the number of applications available in the cloud and the demand for them by end users is increasing, so that one of the challenges in web applications and technological infrastructure that supports these applications is the performance and reliability of the applications [1, 2, 3, 4, 5]. In this sense, understanding the capacity of the server which hosts a web application is fundamental for the planning and configuration of both system capacity and resource management, taking into account the quality of service (QoS) [6, 7, 8]. Poor performance of a web system can negatively affect the profitability and reputation of companies that rely on such infrastructure [9], [10]. An important measure to ensure the quality of web applications is through the use of performance tests [2]. Performance tests collect information related to memory and CPU usage, as well as server response time to sequential and concurrent requests [11].

Among the methods used to evaluate the performance of the infrastructure available in the cloud, load or stress tests stand out, which allow determining or predicting the behavior of a website or web service in response to a different number of requests that emulate the behavior of real clients [12]-[14]. Thus, the objective of performance tests based on load simulation is to identify performance bottlenecks in order to optimize the operation and reliability of the system. Therefore, the load capacity is an important indicator of the tests performed on web applications in order to determine their performance [2], [15].

In the same sense, with respect to the tools that are responsible for performing load tests on web applications, their accuracy is essential, since they can lead the company to overestimate or underestimate the capabilities of the system [2]. Although the existence of different tools to execute load tests such as Apache Benchamark, JMeter, BlazeMeter, Blitz and Gatling, which allow the simulation of sequential and concurrent requests to web applications, has been evidenced, these tools do not use machine learning methods supported by the polynomial regression approach to characterize the curve obtained from the response times of web applications [16], in order to identify the mathematical or polynomial equation that represents the behavior or load capacity of the evaluated web application. Likewise, the aforementioned tools do not use outlier detection methods for the accurate estimation of the load of a web server.

In this paper, we propose as a contribution the characterization of load tests performed on web applications using machine learning techniques and specifically using the supervised learning model supported by polynomial regression. For the above, this article developed as a contribution an automated tool that allows both the execution of sequential and concurrent load tests, and the analysis of the results obtained in the test by applying polynomial regression on the test results, in order to determine the polynomial of degree 2 or 3 that characterizes the behavior of web applications as a function of the percentage of requests served. The tool also performs the evaluation of the polynomial regression model obtained for the training and test set, after automatic detection of outliers. As a means of validation of the approach proposed in this article, a case study was carried out in which load tests were performed on the web portal of the Universidad de Cartagena in Colombia. The application approach of supervised learning techniques and the automated tool proposed are intended to serve as a reference for the development and replication of research focused on the characterization of the results of a load and/or stress test, in order to obtain a diagnosis of the infrastructure that supports the hosting of web portals.

The paper is organized as follows: Section II presents a set of related works that were considered as a reference for the development of this research. Section III describes the different phases of the methodology that guided the development of this research. Section IV presents the results obtained from this work, which includes the description of the functional modules that make up the proposed tool, as well as the description of the graphical interface of the implemented tool. This section also presents a case study developed from the tool built on the web portal of the Universidad de Cartagena. Section V presents the discussion based on the results obtained and considering the works mentioned in the state of the art. Finally, Section VI presents the conclusions and future work derived from this research.

II. RELATED WORKS

Different works have been carried out in the literature about load tests. Thus in [13] the conceptualization of load tests is presented and a scenario for the execution of load tests in web applications is proposed, which allows determining parameters such as: response time, CPU usage of the web server, memory used by the server and memory used by the web application. In [11], a model for the execution and automation of performance tests in web services is proposed as a contribution, which is based on load balancing strategies in order to obtain the processing capacity and the tolerance capacity of a given web service. In the same sense, in [17] a method for performance evaluation in web services based on distributed agents is proposed, which are responsible for executing different connection threads with the web service and reporting the results with a manager agent. On the other hand, taking into account that load models do not usually match real scenarios, an improved user group model is proposed in [2] to improve the simulation of real load in performance tests on web applications. Likewise, [18] proposes as a contribution an improvement to the traditional performance tests on web applications, in such a way that the number of simulated concurrent users decreases or increases automatically depending on the application response times, until the number of users with which the application responds adequately is determined. In [19] a review of the types, indicators and methods used in performance tests on web applications, as well as a set of strategies to be considered in the optimization of performance tests on this type of applications, is carried out. In [20] a tool for load and/or stress evaluation in web applications based on cognitive metrics is proposed as a contribution, in order to analyze potential risks at the performance level in web services and reduce unclear errors in the distributed environment. In [21] an automated technique for the execution of load tests in web applications is proposed as a contribution, which creates a synthetic workload with specific characteristics and with dependence between requests, considering that traditional methods do not consider this dependence. In [6], an approach for measuring stress in web applications based on low-level hardware performance metrics, such as instruction execution rate and cache access behavior, is proposed as a contribution, which define the inner workings of the server and are used by machine learning techniques to obtain the performance level of the web application. In [16] a tool for the execution of sequential and concurrent load tests on web applications is proposed, which allows a statistical and graphical analysis of the results, as well as the application of linear regression methods on the response times of the requests by the server.

The above works show the growing need to characterize and automate the development process of load and/or stress tests on web applications and services, in order to improve the availability and quality of the services offered to end users on the web. In this sense, it has been evidenced that most of the works have focused on improving the technique or the process for the execution of the tests, without focusing on the characterization of the response curves of the load test results through machine learning techniques. In this sense, although the proposal presented in [16] performs a characterization of the curve with the server response times, making use of a linear regression model, it does not consider the detection of outliers, so that the equation representing the server behavior can be optimized by means of a curve that better represents the model through polynomial regression and outlier filtering. Thus, this paper proposes as a contribution the characterization of load tests in web applications by using polynomial regression methods and outlier detection, which according to the results obtained provide a closer approximation to the behavior of web server response times.

III. METHODOLOGY

For the development of this research, the iterative research pattern proposed by Pratt [22] was taken into consideration, which consists of four methodological phases: “observe the application”, “identify the problem”, “develop the solution”, and “test the solution” (see Fig.1).

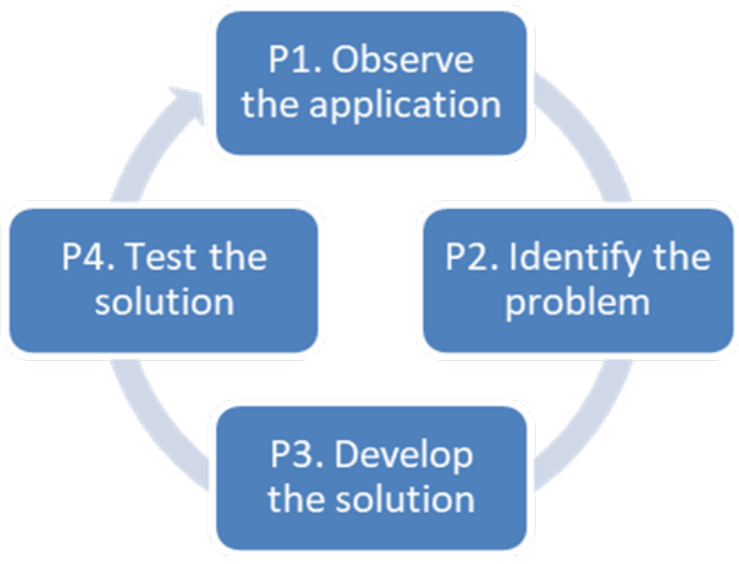

Fig. 1 Methodology considered.
Source: self-made

In phase 1 of the methodology, the load and stress tests developed on web applications were characterized in order to determine their scope. Likewise, different tools for the execution of these load tests were explored in order to identify different deficiencies in the execution and analysis of the test results. Based on the above, it was identified that the tools explored do not include the detection of outliers in response times and do not make use of polynomial regression models for the characterization of the curve that represents the response times of the infrastructure that supports the evaluated web application.

In phase 2 of the methodology, based on the deficiencies identified in phase 1, the functional modules and high-level interfaces of an automated tool for the execution of load and/or stress tests in web applications were designed, considering the inclusion of outlier detection and polynomial regression processes. Polynomial regression is a special case of multiple linear regression in which the aim is to obtain the prediction of a response variable from a predictor variable (both quantitative), so that the relationship of these variables is modeled as a polynomial type function. Thus, from the polynomial regression it is possible to obtain a polynomial of degree P such as the one presented in Eq. (1).

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \ldots + \beta_p X^p + \varepsilon \quad (1)$$

Thus, the polynomial of Eq. (1) can be expressed in terms of the linear model of Eq. (2) in which X=X1, X2=X2, X3=X3…….Xp=XP.

$$Y = \beta_0 + \beta_1 X + \beta_2 X_2 + \beta_3 X_3 + \ldots + \beta_p X_p + \varepsilon \quad (2)$$

Based on the above, to solve Eq. (2) by means of multiple linear regression, it is possible to use the matrix approach proposed in Eq. (3) for a total of n samples.

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1\mathrm{p}} \\ 1 & x_{21} & \cdots & x_{2\mathrm{p}} \\ 1 & x_{31} & \cdots & x_{3\mathrm{p}} \\ \vdots & \vdots & \cdots & \vdots \\ 1 & x_{\mathrm{n}1} & \cdots & x_{\mathrm{np}} \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix} + \begin{pmatrix} \varepsilon_0 \\ \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_p \end{pmatrix} \quad (3)$$

The matrix in Eq. (3) can be summarized or expressed as presented in Eq. (4).

$$Y = X\beta + \varepsilon \quad (4)$$

Considering that the objective of Eq. (4) is to determine the vector of coefficients , it is possible by matrix operations (matrix transpose and the inverse matrix) to clear Eq. (4), as shown in Eq. (5).

$$\beta = \left(X^T X\right)^{-1} X^T \left(Y - \varepsilon\right) \quad (5)$$

Within phase 3 of the methodology, an automated tool was developed in the Python language, which allows in the first instance the execution of sequential and concurrent load tests on web applications using the free apache benchmark tool in the background. Secondly, the proposed tool allows the analysis of the response times obtained from the load tests through the use of the polynomial regression model described in phase 2, prior detection of outliers. For the application of the polynomial regression models and the outlier detection method, the Python machine learning library scikit-learn was used. Finally, in phase 4 of the methodology, the functionality of the tool was verified through the development of a case study in which load tests were performed on the portal of the Universidad de Cartagena, in order to determine the polynomial that characterizes the performance behavior of the infrastructure that supports this portal.

IV. RESULTS

This section describes the results obtained through the development of this research, which includes the specification through a flowchart of the functional processes developed by the tool, as well as the description of the functional modules that make up the tool. Similarly, this section presents the different views of the tool and a case study in which the functionality of the proposed tool was verified.

Based on the above, Fig. 2 shows a flowchart describing the different processes developed by the automated tool for the execution and analysis of load tests, using polynomial regression models.

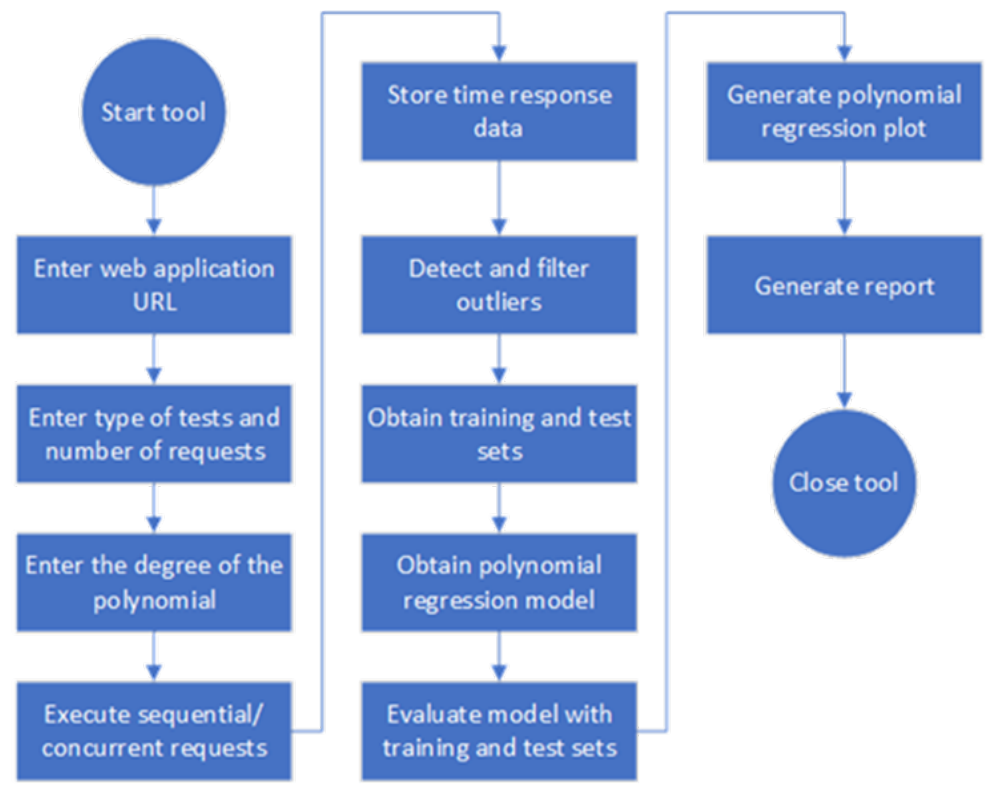

Fig. 2 Flowchart of the proposed tool.
Source: self-made

Once the automated tool is started, it is necessary to enter the URL of the web application to be evaluated, as well as the type of test to be performed (sequential or concurrent), the number of requests and the degree of the polynomial to be obtained. From the aforementioned data, the tool allows the execution of sequential and concurrent load tests, so that the response times obtained for different percentages of requests are stored in floating type arrays. Using the array data, the tool performs outlier detection and filters these time values in order to improve the accuracy of the polynomial regression model. Based on the data set without outliers, the tool divides the data into training and test sets to prevent overfitting of the polynomial regression model. On the training set data, the polynomial regression model is obtained for the defined degree, which is validated with both sets (training and test) using the R2 and RMSE metrics in order to evaluate the accuracy and effectiveness of the model. In the same way, the proposed tool allows the generation of a curve that presents the values of the response times as a function of the percentage of functions served, contrasted with the polynomial regression curve. Finally, the tool allows generating a report in .txt format with

the polynomial obtained and the results of the evaluation metrics (R2 and RMSE) for the given model.

The processes presented in Fig. 2 and described above can be grouped into 4 functional modules (see Fig 3): GUI, Load Tests, Outlier Detection, Polynomial Regression, Report Generation. The "GUI" module is in charge of displaying the tool's graphical interface and managing the different graphical components (labels, panels, buttons, text fields and text areas), as well as handling the different events that allow the test engineer to interact with the tool's interface. Thus, through the graphical interface generated by this module it is possible to enter the configuration data of a load test (URL of the web application, type of test, number of requests, degree of the polynomial) to obtain the polynomial regression model and its graphical visualization in the interface. The GUI module was implemented using the Tkinter library of the Python language.

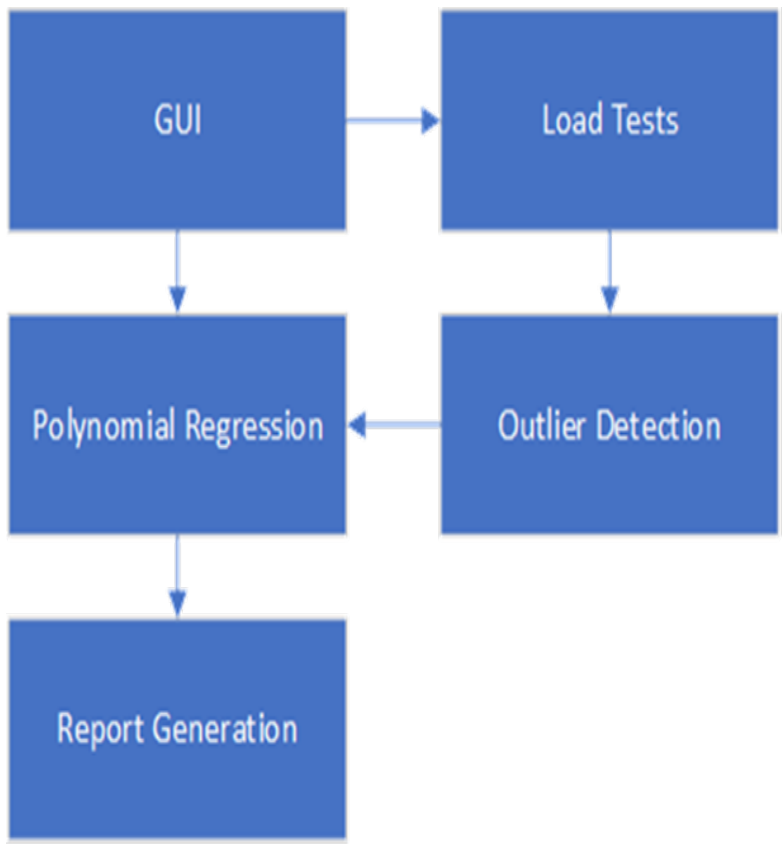

Fig 3. The Block diagram of the tool.
Source: self-made

The "Load Tests" module is responsible for the execution of sequential and concurrent tests on a given web application, considering the parameters entered in the graphical interface of the tool and using the free apache benchmark library in the background, which allows to emulate remote clients and to obtain a report of the response times of different progressive percentages of requests. The "Outlier Detection" module is in charge of detecting and filtering out-of-range values or "outliers" from the response times obtained in the load tests, using the "ElipcticEnvelope" method of the scikit-learn Python library. The above, in order to improve the accuracy and effectiveness of the polynomial regression model to be obtained. The "Polynomial Regression" module is in charge of determining the polynomial regression model starting from the data without outliers, as well as evaluating the model obtained with the training and test sets using the RMSE and R2 metrics. This module also plots the model obtained with respect to the complete load test data. The polynomial regression model was implemented by using the "PolynomialFeatures" method of Python's scikit-learn library, while the model chart was generated through Python's matplotlib library. Finally, the "Report Generation" module is in charge of generating a .txt report with the polynomial regression model obtained and its evaluation through the RMSE and R2 metrics, using the methods provided for file management in the Python language.

Fig. 4 shows the graphical interface of the tool for the execution of load tests, which implements the modules and processes previously described.

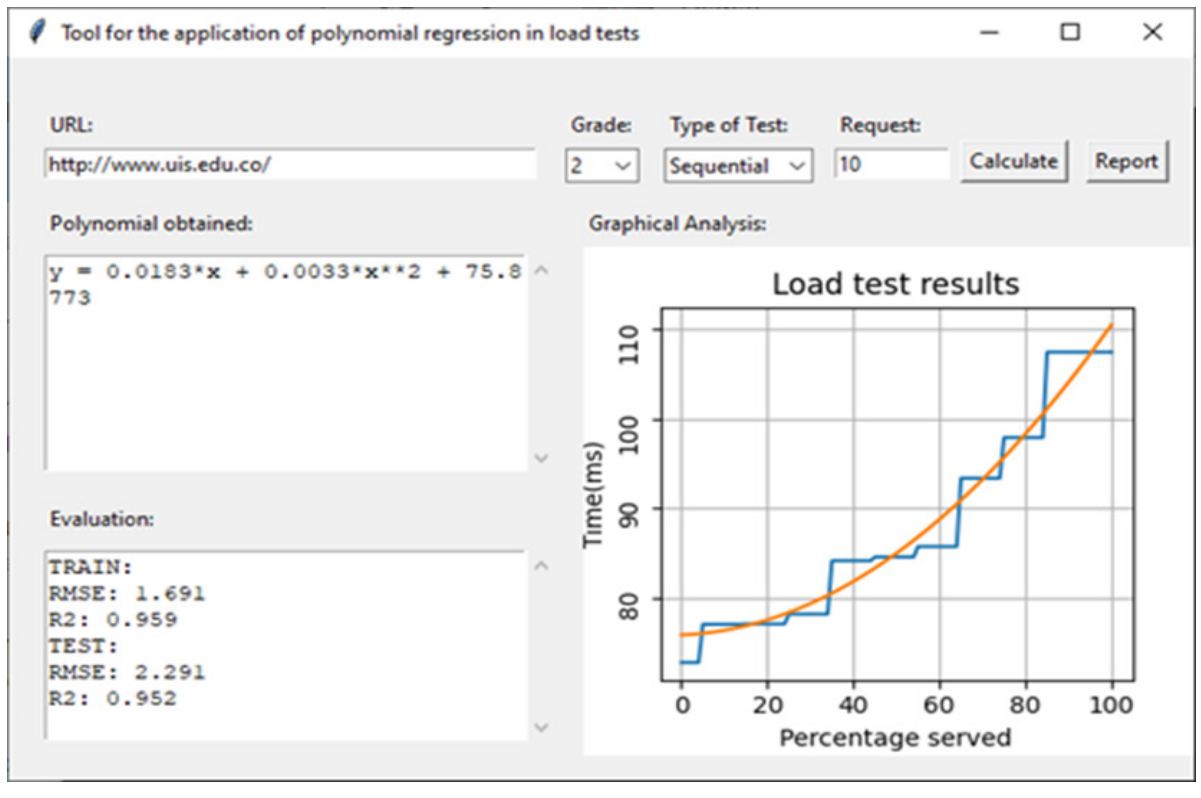

Fig. 4 Proposed tool.
Source: self-made

As shown in Fig. 4, once the test engineer has entered the load test configuration data (URL of the web application, degree of the polynomial to be determined, type of test and number of requests), it is possible to run the sequential and/or concurrent load tests on the web application specified through the URL by pressing the “Calculate” button, so that when the response times of the web application are obtained for different percentages of requests, the outliers are filtered and the polynomial regression model or polynomial equation that represents the curve with the response times with the training set (80% of the data) is determined, which is presented in the upper left part of the graphic interface of the tool. Likewise, once the tool obtains the polynomial regression model, it performs the validation of the model for the training and test sets using the RMSE and R2 metrics, so that the results are presented in the lower left part of the tool interface. On the other hand, the proposed tool allows generating a graph on the right side of the interface, in which the data of the test response times are contrasted with respect to the polynomial regression model obtained. Also, by pressing the “Report” button, it is possible to generate a report in a .txt file with the polynomial model obtained and the validation results for the training and test sets. As an example, Fig. 4 shows a working example of the tool in which load tests were performed with 10 sequential requests on the portal: http.//www.uis.edu.co, in order to obtain as a result a polynomial equation of degree 2. Thus, the tool obtained as a result the polynomial presented in Eq. (6), which relates the percentage of requests made or served (x) with respect to the response time (y).

$$y = 0.0183x + 0.0033x^2 + 75.8773 \quad (6)$$

Similarly, the validation of the above model with the validation set obtained an RMSE value of 1.691 and an R2 value of 0.959, while with the test set an RMSE value of 2.291 and an R2 value of 0.952 was obtained.

In order to verify the performance and relevance of the approach proposed in this article and of the automated tool implemented, a case study was conducted on the web portal of the Universidad de Cartagena, in which load tests were executed with a total of 50 sequential requests and with a polynomial of degree 3, as shown in Fig. 5.

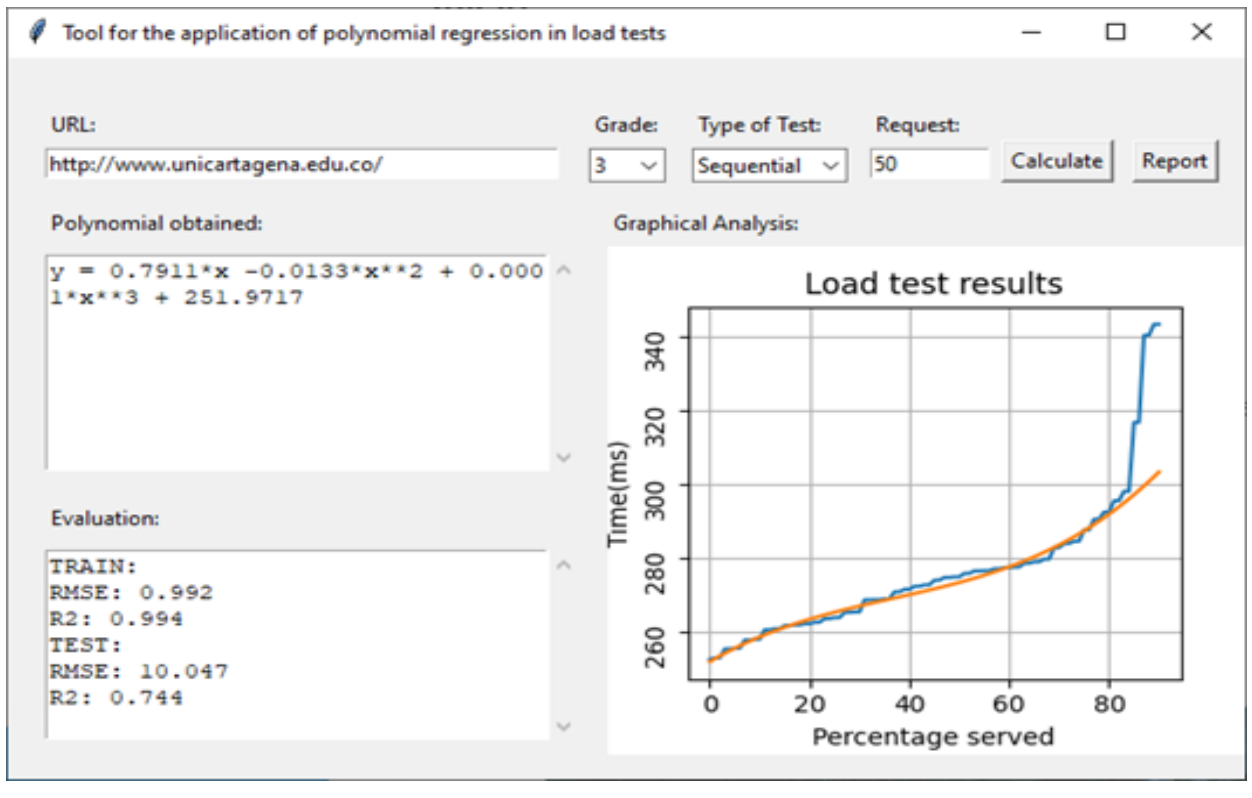

Fig. 5 Results of the case study with polynomial degree 3.
Source: self-made

The results obtained in the case study regarding the model determined and its validation with the training and test sets are presented in Table I. As mentioned in the previous section, the model obtained is in terms of the variables x and y, where x represents the percentage of sequential requests served, while y represents the response time. It can be seen that the correlation coefficient in the training set is close to 1, while in the test set it is above 0.7, indicating that the polynomial regression model has good accuracy in characterizing web server load. In the same way, through the model obtained it is possible to determine, by means of the first derivative, the curve that represents the variation of the response time as a function of the number of sequential requests, which is very useful to evaluate the infrastructure that supports the web application.

TABLE I CASE STUDY RESULTS.

Model/Evaluation	Results
Model	y=0.7911x-0.0133x2+0.0001x3+251.9717
Training set evaluation	RMSE=0.992 R2=0.994
Test set evaluation	RMSE=10.047 R2=0.744

Source: self-made

V. DISCUSSION

Given the need evidenced in the state of the art to automate the process of developing load tests in web applications, this work proposed as a contribution the development of an automated tool for the execution of load tests and the analysis of the curve obtained with the response times of the server that hosts the evaluated web application, so that the tool has as a contribution the application of machine learning methods and specifically the obtaining of polynomial regression models for the characterization of the curve that relates the percentage of requests served with the response time of the server. Thus, the tool proposed in this paper represents a contribution with respect to the state-of-the-art works reviewed and with respect to different tools for the execution of load tests, which in general terms do not use machine learning methods to model the performance of the infrastructure that supports web applications, limiting themselves to the execution of sequential and concurrent tests. Specifically, the approach proposed in this article allows determining a polynomial of degree 2 or 3 that characterizes the mathematical relationship between the percentage of requests served (x) and the response time obtained by the web server (y), which were captured through the background use of the apache benchmark library, prior detection and filtering of out-of-range values or outliers. Likewise, as an added value and in order to replicate the approach proposed in this paper, free software technologies and libraries were used, such as the Python library Tkinter, which allows the management and control of the graphical interface of the

tool, and the machine learning library scikit learn, which allows the automatic detection of outliers, the separation of the training/test sets and the obtaining of the polynomial regression model. Thus, the proposed tool can be extrapolated and considered to enrich the execution of performance tests based on other types of parameters different from those considered in this research.

VI. CONCLUSIONS

Based on the need for tools that allow the execution and analysis of load tests on web applications, in this paper, we proposed as a contribution the development of an automated tool which not only allows the execution of sequential and concurrent load tests, but also the application of machine learning models based on supervised learning and specifically supported in polynomial regression, in order to obtain the polynomial equation that characterizes the responses of the server that hosts the web application to load tests. Thus, the results obtained from the approach considered are intended to support the test engineer to make decisions about improving the performance or configuration of the infrastructure supporting a given web application. Likewise, the proposed approach can be considered to be extrapolated to characterize the performance of web applications in terms of other related parameters.

The approach proposed in this paper is not only a contribution with respect to other state-of-the-art works regarding the application of machine learning models, but also with respect to the identification and filtering of outliers associated with the response times obtained for different percentages of requests served, which directly influences the effectiveness of the model obtained. In this sense, the tool proposed in this paper use the “ElipcticEnvelope” method of the scikit-learn Python library, which allows the automatic detection of out-of-range values within a data array with server response times. The approach considered in this paper is intended to serve as a reference to replicate the outlier detection method in the implementation of tools for the execution and analysis of performance tests based on different parameters than those considered in this research.

The tool presented in this paper was implemented using free software tools, in such a way that for the execution of the sequential and concurrent load tests, the free apache benchmark library was used in the background. Similarly, for the detection of outliers in server response times, the scikit-learn Python library was used through the “ElipcticEnvelope” method. In the same wat, to obtain the polynomial regression model or the polynomial equation that characterizes the server response curve, the “PolynomialFeatures” method of the scikit-learn Python library was used. Finally, the Python Tkinter library was used to implement the graphical interface of the proposed tool. The aforementioned libraries are intended to serve as a reference for the implementation in the academic and business context of automated tools for the evaluation and performance analysis of web applications.

In order to verify the relevance of the proposed approach and the constructed tool, a case study was developed on the web portal of the Universidad de Cartagena, in which a total of 50 sequential requests were executed, from which the tool determined a polynomial model of degree 3 that obtained as a result for the training set, an RMSE value of 0. 992 and an R2 value of 0.994, while for the test set, an RMSE value of 10.447 and an R2 value of 0.744 were obtained.

Finally, as a future work derived from the present research, it is intended to complement the analysis of the results by incorporating unsupervised learning or clustering methods for the analysis of the results obtained in the load tests performed on a web application. Likewise, it is intended to include models based on time series for the characterization of the response curve obtained in the load tests.

ACKNOWLEDGEMENTS

We thank the Universidad de Cartagena, Faculty of Engineering, Systems Engineering School for the support received in the development of this research.

CRediT AUTHORSHIP CONTRIBUTION STATEMENT

Gabriel E. Chanchí G.: Conceptualization; Data Curation; Investigation; Methodology; Writing original draft. Manuel A. Ospina A.: Research, Software, Writing - Edit and review. Mónica E. Ospino P.: Conceptualization, Research, Writing - Edit and review.

REFERENCES

[1] L. Germine, K. Nakayama, B. C. Duchaine, C. F. Chabris, G. Chatterjee, and J. B. Wilmer, "Is the Web as good as the lab? Comparable performance from Web and lab in cognitive/perceptual experiments," *Psychon. Bull. Rev.*, vol. 19, no. 5, pp. 847–857, Oct. 2012, doi: 10.3758/S13423-012-0296-9/FIGURES/3.

[2] H. Zhu and H. Wu, "Research on web application load testing model," in Proceedings of the *2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference, ITNEC 2017*, Feb. 2018, vol. 2018-Janua, pp. 1175–1178, doi: 10.1109/ITNEC.2017.8284961.

[3] R. Khan and M. Amjad, "Performance testing (load) of web applications based on test case management," *Perspect. Sci.*, vol. 8, pp. 355–357, Sep. 2016, doi: 10.1016/J.PISC.2016.04.073.

[4] R. A. Garita-Araya, "*Tecnología Móvil: desarrollo de sistemas y aplicaciones para las Unidades de Información*," E-Ciencias la Inf., vol. 3, no. 2, pp. 1–14, 2013.

[5] A. Yu and S. Yang, "Research on web server cluster load balancing algorithm in web education system," *J. Supercomput.* 2018 765, vol. 76, no. 5, pp. 3364–3373, Sep. 2018, doi: 10.1007/S11227-018-2573-5.

[6] J. Rao and C. Z. Xu, "Online capacity identification of multitier websites using hardware performance counters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 3, pp. 426–438, 2012, doi: 10.1109/TPDS.2010.92.

[7] P. P. do Nascimento, P. Pereira, J. M. Mialaret, I. Ferreira, and P. Maciel, "A methodology for selecting hardware performance counters for supporting non-intrusive diagnostic of flood DDoS attacks on web servers," *Comput. Secur.*, vol. 110, p. 102434, Nov. 2021, doi: 10.1016/J.COSE.2021.102434.

[8] R. D. van der Mei, R. Hariharan, and P. K. Reeser, "Web Server Performance Modeling," *Telecommun. Syst.* 2001 163, vol. 16, no. 3, pp. 361–378, 2001, doi: 10.1023/A:1016667027983.

[9] T. Ahmad, D. Truscan, and I. Porres, "Identifying worst-case user scenarios for performance testing of web applications using Markov-chain workload models," *Futur. Gener. Comput. Syst.*, vol. 87, pp. 910–920, Oct. 2018, doi: 10.1016/J.FUTURE.2018.01.042.

[10] R. Ramakrishnan and A. Kaur, "An empirical comparison of predictive models for web page performance," *Inf. Softw. Technol.*, vol. 123, p. 106307, Jul. 2020, doi: 10.1016/J.INFSOF.2020.106307.

[11] X. Y. Guo, X. S. Qiu, Y. H. Chen, and F. Tang, "Design and implementation of performance testing model for web services," in *CAR 2010 - 2010 2nd International Asia Conference on Informatics in Control, Automation and Robotics*, 2010, vol. 1, pp. 353–356, doi: 10.1109/CAR.2010.5456825.

[12] C. Vögele, A. van Hoorn, E. Schulz, W. Hasselbring, and H. Krcmar, "WESSBAS: extraction of probabilistic workload specifications for load testing and performance prediction—a model-driven approach for session-based application systems," *Softw. Syst. Model.*, vol. 17, no. 2, pp. 443–477, May 2018, doi: 10.1007/S10270-016-0566-5/TABLES/10.

[13] B. Vani, R. Deepalakshmi, and S. Suriya, "Web based testing-An optimal solution to handle peak load," in Proceedings of the *2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering, PRIME 2013*, 2013, pp. 5–10, doi: 10.1109/ICPRIME.2013.6496439.

[14] A. Minaya Cubillas, "Una Revisión de los Métodos de Pruebas para Aplicaciones Web," *Rev. Investig. Sist. e Informática*, vol. 5, no. 2, pp. 46–59, 2008.

[15] J. C. C. Shaw, C. G. Baisden, and W. M. Pryke, "Performance Testing — A Case Study of a Combined Web/Telephony System," *BT Technol. J.*, vol. 20, no. 3, pp. 76–86, Jul. 2002, doi: 10.1023/A:1020899610791.

[16] G. E. Chanchí, J. L. Vivas, and W. Y. Campo, "Propuesta de una herramienta portable para la ejecución de pruebas de carga en aplicaciones web," *Rev. Ibérica Sist. e Tecnol. Informação*, no. E29, pp. 159–172, 2019.

[17] D. Hao, Y. Chen, F. Tang, and F. Qi, "Distributed agent-based performance testing framework on Web Services," in Proceedings *2010 IEEE International Conference on Software Engineering and Service Sciences, ICSESS 2010*, 2010, pp. 90–94, doi: 10.1109/ICSESS.2010.5552290.

[18] G. Jiang and S. Jiang, "A quick testing model of web performance based on testing flow and its application," *2009 6th Web Inf. Syst. Appl. Conf. WISA 2009*, pp. 57–61, 2009, doi: 10.1109/WISA.2009.16.

[19] K. Zhu, J. Fu, and Y. Li, "Research the performance testing and performance improvement strategy in web application," in *ICETC 2010 - 2010 2nd International Conference on Education Technology and Computer*, 2010, vol. 2, pp. 328–332, doi: 10.1109/ICETC.2010.5529374.

[20] R. Srinivasa Perumal and P. Dhavachelvan, "Performance analysis of distributed web application: A key to high perform computing perspective," in Proceedings - *1st International Conference on Emerging Trends in Engineering and Technology, ICETET 2008*, 2008, pp. 1140–1145, doi: 10.1109/ICETET.2008.246.

[21] D. Krishnamurthy, J. A. Rolia, and S. Majumdar, "A synthetic workload generation technique for stress testing session-based systems," *IEEE Trans. Softw. Eng.*, vol. 32, no. 11, pp. 868–882, Nov. 2006, doi: 10.1109/TSE.2006.106.

[22] K. Pratt, "*Design Patterns for Research Methods: Iterative Field Research*." 2009, [Online]. Available: http://kpratt.net/wp-content/uploads/2009/01/research_methods.pdf.

Gabriel Elías Chanchí Golondrino Electronics and Telecommunications Engineer, Master in Telematics Engineering, PhD in Telematics Engineering. Assistant Professor Universidad de Cartagena. Cartagena, Colombia. Director of the Technological Development for Society Research Group, (DaToS), Associate Researcher. https://orcid.org/0000-0002-0257-1988

Manuel Alejandro Ospina Alarcón Control Engineer, Master in Engineering, PhD of Engineering. Assistant Professor Universidad de Cartagena. Cartagena, Colombia. Member of the Technological Development for Society Research Group, (DaToS), Associate Researcher. https://orcid.org/0000-0003-4510-0753

Mónica Ether Ospino Pinedo Systems Engineer, Esp in Educational management, Master in Strategic Management Information Technology Specialty Orientation Software Companies, PhD Student. Assistant Professor Universidad de Cartagena. Cartagena, Colombia. Member of the E-soluciones research Group. Associate Researcher https://orcid.org/0000-0003-3357-6316