International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 11(2019) pp. 178-191
© MIR Labs, www.mirlabs.net/ijcisim/index.html

Submitted: 4 Jan, 2019; Accepted: 18 March, 2019; Publish: 26 July, 2019

Model Optimisation for Server Loading Forecasting with Genetic Algorithms

Cláudio A.D.Silva[1], Carlos Grilo[1,2] and Catarina Silva [1,3]

[1]School of Technology and Management, Polytechnic Institute of Leiria,
Portugal
klaudio.ads@gmail.com

[2]CIIC, Polytechnic Institute of Leiria,
Portugal
{*carlos.grilo, catarina*}*@ipleiria.pt*

[3]Center for Informatics and Systems of the University of Coimbra,
Portugal
catarina@dei.uc.pt

Abstract: **Server load prediction has different approaches and applications, with the general goal of predicting future load for a period of time ahead on a given system. Depending on the specific goal, different methodologies can be defined. In this paper, we study the use of temporal factors, along with its manual or optimised application using genetic algorithms. The main steps involved are data transformations, a novel pre-processing method based on enrichment of data through the inducing of temporal factors and a genetic algorithms wrapper that optimises all the variables in our approach. The created model was tested on a short-term load forecasting problem, with the use of data from single and combined months, regarding real data from Wikipedia servers. The learning methods used for creating the different models were linear regression, neural networks, and support vector machines. A basic dataset, as well as an enriched dataset, were the core elements for the two scenarios studied. Results show that it is possible to tune the dataset features, e.g., granularity and time window to improve prediction results.**

Keywords: Load Forecasting, Linear Regression, Artificial Neural Networks, Support Vector Machines, Server Load Prediction, Wikipedia.

I. Introduction

Load prediction, also known as load forecasting, emerged as a support to predict and prevent resource imbalance [1], rapidly proving its importance in the industry. Organisations with inappropriate forecasts or no forecasts at all have a high chance of being negatively affected. Incoherent forecasts can also have devastating effects in an organisation. Underestimation may result in revenue loss or in the incapability to fulfil the organisations goals. On the other hand, overestimation may lead to overstock, which can be impossible to resell/use with profit. When forecasting load, it is of the utmost importance to keep a balance between resources and demand, thus avoiding stock outs or over stocking.

Load forecasting is a crucial element in various areas, such as, electric load forecasting, sales forecasting, dam water levels forecasting, traffic forecasting and load forecasting in computer servers, commonly known as server load prediction [2, 3, 4, 5], which is the area of study of this paper. The goal in server load prediction is usually to determine the load that will be put on the servers by users, either in terms of data volume or number of connections. An efficient resource management is essential for the viability of a data center [6]. Adequate resource availability to workloads should always be guaranteed without breaching service level agreements. The resource allocation process can be supported and eased through prediction algorithms. These algorithms use incoming load and/or resources to forecast adequate resource allocation, thus guaranteeing the performance of the data center for every kind of specific load [2].

In server load prediction, and in load forecasting in general, one of the first steps is the identification of the load forecasting type. These types are categorised according to the forecast time horizon, which is usually problem dependent. It can be set by the specific goal of the prediction to achieve, or it can be constrained by the availability and accuracy of data. Therefore, each forecast horizon is used for different purposes and/or restrictions. According to Gross [7], these horizons can be divided as:

- Very-Short Term Load Forecasting (VSTLF), which has a forecast horizon lower than an hour.
- Short-Term Load Forecasting (STLF), which has a fore-

cast horizon of up to 30 days.

- Medium-Term Load Forecasting (MTLF), with a forecast horizon of up to one year.

- Long-Term Load Forecasting (LTLF), with a forecast horizon higher than a year.

The case study presented in this paper is of hourly load prediction, which falls in the STLF category. Along with the distinct types of load forecasting, a large variety of methods exist, each one with different levels of suitability to the different types and goals of the load forecasting problems. The dynamic nature of load in servers restrains the performance of certain prediction algorithms, such as, linear models, e.g., linear regression. The main cause for this performance deficit is the non-linearity and non-stationarity in data [8]. Nevertheless, there are other viable models that handle better the non-linearity and non-stationarity in data. For instance, Artificial Neural Networks (ANNs) [8, 9, 10, 11, 12] and Support Vector Machines (SVM) [13].
ANN, when compared to traditional linear models, tend to have better performance when handling nonlinear and non-stationary data. However, ANN have some chronic problems with them, e.g., local minima [14], a problem that SVM do not share. SVM can guarantee a global minimum by using a quadratic optimisation approach.
SVM use statistical learning theory, along with a structural risk minimisation principle. This leads to better generalisation and it is superior to the empirical risk minimisation used by ANN.
In this paper, we present a case study involving real data from the Wikipedia servers, which extends previous work [15]. The present case study shows an improvement in model performance through the application of a multiple step approach. This approach is divided between a manual application and an optimised application using genetic algorithms. The main steps involved are data transformations, such as, attribute creation, selection and modification. A novel preprocessing method based on enrichment of data through the inducing of temporal factors into the data, and lastly a genetic algorithm wrapper that optimises the combination of variables to be included as inputs of the model. The created model was tested by predicting the next immediate hour of load, with a single and combined months approaches. The learning methods used for creating the different models were linear regression, neural networks, and support vector machines.
The rest of the paper is organised as follows. Section 2 presents the learning methods applied in the paper. Section 3 describes the areas of application for server load prediction. In Section 4 the proposed approach is presented, along with the server load prediction problem. The results are presented in Section 5. Each tested scenario is described, along with the details of the test environment. Finally, Section 6 presents conclusions and future lines of research.

II. Learning methods

The learning methods used in this work are linear regression, ANN and SVM, for the creation of our prediction model, and genetic algorithms (GAs) for the optimisation of the various variables found in our approach.

A. Linear Regression

Linear regression [16] uses the relationship between an independent variable or variables, and the dependent variable, to build its model. Its applicability is mainly in regression problems. However, it can be used in classification problems to identify the correlation between attributes. Linear regression estimates the coefficients for a hyperplane or line that best fits the training data. In order to find which line better fits the training data the following equation is used:

$$\hat{y}_i = \alpha + \beta x_i \tag{1}$$

where $\hat{y}_i$ is the predicted response or fitted value for input x_i. The forecast value usually has an error, commonly denoted as prediction error or residual error, associated to it, which can be defined as:

$$\xi_i = y_i - \hat{y}_i \tag{2}$$

The line that best fits the training data is the one for which the n prediction errors, one per observed data point, has the smallest overall value. This goal can be achieved by using the least squares criterion, which minimises the sum of the squared prediction and can be calculated using:

$$Q = \sum_{i=1}^{n} (y - \hat{y})^2 \tag{3}$$

The squaring of the prediction error is done to avoid that the positive and negative prediction errors cancel each other out when summed and therefore yield 0. The problem that arises when using Equation 3 is that it just gives one value per regression line, i.e., in order to have the best possible result, the implementation of Equation 3 must be done in multiple lines. This problem can be tackled by using Equation 1 in conjunction with Equation 3, which leads to the following equation:

$$Q = \sum_{i=1}^{n} (y_i - (\alpha + \beta x_i))^2 \tag{4}$$

However, in order to compute Equation 4, the values of α and β must be calculated. This can be done through the equations:

$$\begin{gathered} \beta = \frac{\sum_i^n (x_i - \bar{x})(y_i - \bar{y}))}{\sum_1^i (x_i - \bar{x})^2} \\ \alpha = \bar{y} - \beta\bar{x} \end{gathered} \tag{5}$$

where $\bar{x}$ and $\bar{y}$ are the average values of the observation in *x* and *y*, respectively. Since β and α are derived using the least squares criterion, the resulting Equation 1 is often referred to as the least squares line or least squares regression line.

B. Artificial Neural Networks

ANN have several applications on short-term prediction. Its application is not exempt from some constraints and challenges, either due to intrinsic limitations of the methodology itself or to difficulty in defining the model's architecture [17].

An ANN thus consists of an interconnection of several processing units (neurons), with a configuration vaguely similar to that of the human brain. Figure 1 depicts an artificial neuron.

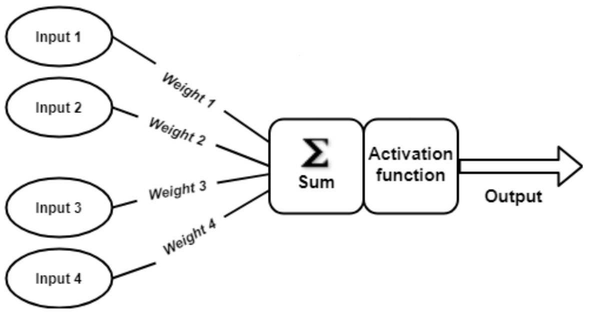


Figure. 1: Biological neuron vs Artificial neuron

The entries and exits of a neuron correspond, similarly, to the synapses and axons of the human brain neuron. Each new input presented is therefore weighted by a synaptic weight, which is indicative of the strength of the bond. Connection strengths are adjusted by a learning algorithm, in an iterative learning process. Each neuron performs very simple operations, starting with the weighted sum of the inputs by the weights of the respective connections, followed by the computation of the neuron activation value using an activation function, which receives the weighted sum as input. Usually, the activation function is non-linear and differentiable.
In load forecasting, the most common used type of ANN are multilayer feed-forward networks, trained with the backpropagation algorithm [18]. This approach is also applied in this work. Backpropagation can be divided into two distinct phases, a forward and a backward passing phase. In the forward phase, the output of the network is computed for each input pattern. The backward phase consists in updating the network weights in the direction of negative derivative of the output error. Forward and backward phases operate on the same data points and only proceed to the next data points after both phases are completed.

C. Support Vector Machines

SVM are supervised learning models that use learning algorithms for solving classification and regression problems. They were introduced by Vapnik and its co-workers in 1990 [19]. SVM are focused on two class discriminant functions. SVM make use of a combination of convex optimisation [20] and statistical learning [21] to find a suitable plane/margin in a data space that can separate the data into two different classes.
SVM represent their data as points in a high or infinite dimensional space. A hyperplane is created to correctly divide the data into different categories. SVM can then predict to which category new data will belong. A hyperplane should have the largest margin possible, which is the minimal distance between the hyperplane separating the two classes and the closest data points to the hyperplane (support vectors). Figure 2 depicts a hyperplane with its support vectors and respective margin.
The problem of finding the optimal hyperplane is often tackled by optimisation techniques. One of the most common techniques used is the Lagrange multiplier. Lagrange multipliers, sometimes referred as dual variables, are scalar variables that calculate function extremes in mathematical optimisation with constraints [22]. The optimisation algorithm for the hyperplane generates the weights of the output so that only the support vectors are taken into consideration when calculating the weights.

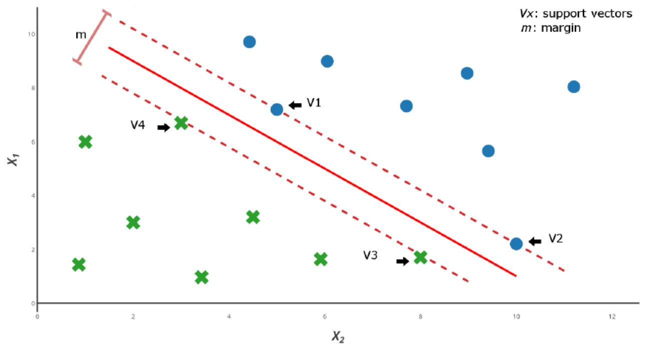


Figure. 2: Hyperplane with respective support vectors

Support Vector Classification (SVC) and Support Vector Regression (SVR) are the current nomenclature used by researchers to denominate SVM for classification and regression problems, respectively. In this paper, due to the nature of the problem, we use a SVM model adapted to linear regression functions, which categorises it as SVR. The operation of SVRs closely resembles that of ANN since they use multilayer sensors and Kernel functions. As in linear regression, the basic idea is to find a function that closely approximates the points of training while trying to reduce the error.

D. Genetic Algorithms

Genetic algorithms (GA) are parallel and stochastic search algorithms inspired by the natural selection of species theory and molecular biology, which allow the evolution of a set of potential solutions to a problem [23].
A GA starts by creating an initial population *P(0)* of size n of potential solutions to the problem to be solved. Then, it evaluates this population using a fitness function that returns a value for each individual indicating its quality as a solution to the problem. After these two steps, it proceeds iteratively while a stop condition is not met. In each iteration, the first step consists of the creation of a new intermediate population *P1(t)*, also of size *n*, by stochastically selecting the best individuals from *P(t)*. All selection methods must obey to the following two principles: 1) The best individuals have more possibilities of being chosen; 2) Selection is done with reposition so that individuals can be chosen more than once. After the selection step, a second intermediate population *P2(t)* is created by stochastically applying genetic operators to the individuals of *P1(t)*. There are mainly two types of genetic operators: recombination and mutation. Recombination mixes genetic material of two solutions, generating two new ones. This operator is applied hoping that the resulting solutions combine the best of their parents. Mutation undergoes minor changes on the solutions and it has the role of replacing or adding genetic material that does not exist in the current population, either because it has been lost or because it has

never existed in previous populations. The process usually involves the choice of a random point of an individual and the validation of a probability of mutation, which switches an existing gene with a randomly generated one. The last step of each iteration consists in evaluating the new population *P(t + 1)*. This sequence of operations stops usually after a predefined number of iterations after which the best individual found is returned [24].

III. Server load prediction

In server load prediction, the most common goal is to determine the load that will be put on the servers by users, either in terms of connections or data volume. To efficiently allocate resources, prediction algorithms are used. In addition, load forecasting lets data centers minimise resource consumption, reducing energy consumption, costs, and, as a result, reducing carbon emissions [25].

A. Areas of application

Server load prediction can be found in a variety of areas, such as, cloud computing, grid computing, utility computing, virtualisation or energy load prediction [25, 26, 27, 28]. The National Institute of Standards and Technology (NIST), defines cloud computing as a model that enables on-demand network access to a shared pool of configurable computing resources, such as, servers, networks, applications, services and storage, that can be rapidly supplied and/or released with minimal service provider interaction [26]. Cloud computing uses the predictions of load of different components to adjust its resources to an optimal state in a proactive approach. For example, the predictions of CPU load, enables the system to adjust the number of cores to an optimal state, releasing or allocating further resources when needed.
Grid computing uses server load prediction to coordinate network and computational resources to achieve a single load. Various scientific applications use grid computing in their solutions, due to its usually computationally intensive tasks. Load prediction lets researchers, analyse the resources needed to run, and adjust the resources accordingly. Although sharing many similarities and challenges, grid computing and cloud computing differ. The difference between cloud computing and grid computing relies mainly in the goal. Grid computing tries to use its resources for a single goal/load, such as, calculating a complex equation. Cloud computing tries to use its resources to answer to multiple goals, e.g., adjusting resources to answer the many different requests made to a server.
Server load prediction is also found in utility computing. Utility computing provides resources on-demand and charges customers based on usage, instead of the typical hardware renting process. Server load prediction enables service providers to truly maximise resource utilisation and minimise their operating costs. Virtualisation is another area that makes use of server load prediction. In virtualisation the physical hardware is overlooked, providing instead virtualized re-sources for applications to use. A virtualised server, which is frequently referred to as virtual machine (VM), provides the capability of merging computing re-sources from clusters of servers and dynamically allocate or re-allocate virtual resources to applications on-demand [26]. Through the use of load prediction, it is possible to test the optimal hardware configuration needed for a VM. Another area of application for server load prediction is autonomic computing, which aims at implementing computing systems capable of self-management, able to react to internal and external actions without human intervention. The goal is to overcome the complexity of managing computer systems and avoid human inter-action that may cause failures to the system. IBM's mainframes are a prime ex-ample in which autonomic computing is applied. Through the analysis of future load, the system independently readjusts the resources for upcoming load. The process of load prediction in autonomic computing involves usually predicting the number of application instances required to handle demand at each particular level, periodically predict the future demand and determining resource requirements, and, lastly, automatically allocating resources using the predicted re-source requirements [29].
Energy efficiency in data centers is another major area in server load prediction. It has been estimated that the cost of powering and cooling accounts for 53% of the total operational expenses of data centers [30]. Companies such as, Google or Amazon are under enormous pressure to reduce energy consumption. The goal is not only to cut down energy cost in data centers, but also to meet government regulations and environmental standards. Energy-aware job scheduling [31] is one way to do this, as well as reducing power consumption by turning off unused machines. For any of these solutions, load prediction is indispensable since, by foreseeing future load, it is possible to adequate schedule energy-aware jobs or turn of machines with no load running on them. Server load prediction became also an indispensable tool for the commercial viability of products like Amazon's web service[1], Google App Engine[2], Microsoft Windows Azure platform[3], or IBM's Bluemix[4].

IV. Proposed approach

The server load prediction problem studied in this paper uses Wikipedia's traffic logs as its research study source. An appropriate solution was defined and created containing automated data gathering, analysis and cleaning processes, data subset configuration and transformation modules and solution optimisation through the application of GA.

A. Solution architecture overview

In this paper we propose a solution that tackles the load prediction problem. The initial steps involve downloading, formatting, and cleaning the data available from the Wikimedia foundation, which is then stored locally. With the stored data, a transformation module is applied to each dataset, which creates new training and test sets. Figure 3 presents the architecture of the approach taken.
The newly created train and test sets are submitted to the model creation and testing module. This module creates

[1]https://aws.amazon.com

[2]https://cloud.google.com/appengine

[3]https://azure.microsoft.com

[4]https://www.ibm.com/cloud-computing/bluemix

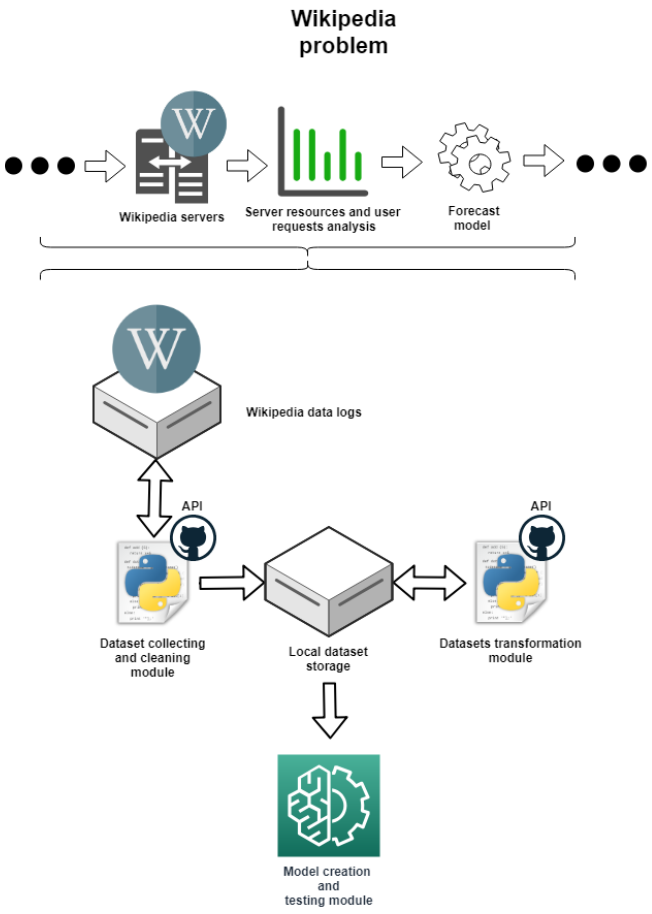

Wikipedia problem
Wikipedia servers
Server resources and user requests analysis
Forecast model
Wikipedia data logs
API
Dataset collecting and cleaning module
Local dataset storage
Datasets transformation module
Model creation and testing module

Figure. 3: Wikipedia solution architecture diagram

models based on linear regression, artificial neural networks and support vector machines.

1) Data gathering, analysis and cleaning

In the data gathering process, the initial step is to treat and collect the data from single or multiple sources and store locally. This can be achieved through a script. After the gathering process, the data is analysed and cleaned. Data analysis tools such as WEKA[5] or RapidMiner[6] can be used. In the cleaning process, the methodologies used in this work were a Pearson correlation coefficient analyses to eliminate irrelevant or repetitive data, direct manipulation in the dataset, e.g., wrongly formatted data, correctly re-formatted and the use of average values to fill in missing values and the application of a cleaning script, which automates the previous mentioned steps, along with adding surgical functions, increasing the versatility and utility of the overall script.
Subsequently to the cleaning step, the configuration values for different train and test sets are defined. The subset configuration values used in this paper were defined in accordance with the human interaction patterns/intervals. Different granularity and time window values were used, being granularity the level of detail each record contains and time window, the quantity of information back in time that is stored in each record. Once the configuration values are defined, the transformation module follows, involving a set of mandatory transformations to the dataset. Figure 4 depicts the steps involved in a single transformation cycle.

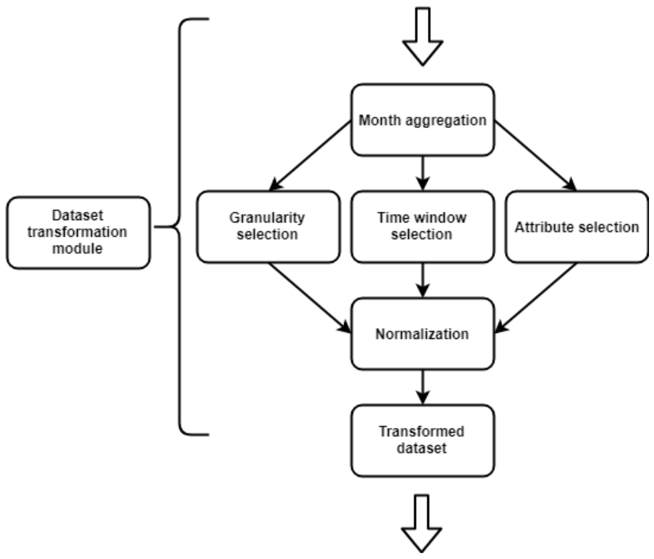


Figure. 4: Diagram depicting the different steps involved in the data transformation module [15].

The first step is applied when more than one dataset worth of data is available. The module uses the different possible combinations of data sets, to create a new aggregate dataset. A resulting dataset could be, for example, the combination of the months May, June, July and August.
Then, the next module transforms the data in accordance to the granularity level. Granularity defines the level of detail in the data, which in this work is always in an hourly scale. Through the application of a granularity value, the initial dataset is transformed into a dataset with records of G granularity, i.e., if we have a granularity of 3, and each original record contains data from an hour, the resulting record will contain the aggregate of 3 hours after the application of the granularity step. Following the granularity step, the time window step is applied. In this step the number of time window units (granularity units) back in time that a record will contain is defined. Since the granularity step was applied before, the created history/time-window shares the same scale. In the attribute selection step, the different sets of possible/desired combinations of attributes are used, creating different combinations datasets. The last step consists in applying normalisation on the resulting train and test set. For each train and test set that is normalised, there is another one that remains unaltered. This is done in order to evaluate how normalization affects the performance of the created model.

V. Experiments and results

This section focus on the solution for the Wikipedia load problem. Two variations of the dataset are used: a basic and an enriched one. In the test scenario the best general model is tested against the best absolute model. The general model is created by using the configuration for time window and granularity that has the best results on average (across all months), while the absolute model is the best model found. This is done to test the overall generalisation capabilities of the approach. The last test scenario is the application of a GA, for dataset optimisation. A final summary and discus-

[5]https://www.cs.waikato.ac.nz/ml/weka
[6]https://rapidminer.com

sion among all test scenarios are presented at the end of this section.

A. Dataset overview

The data used in this paper was collected from the Wikimedia foundation[7]. Whenever a request of a page is made to Wikipedia, whether for reading or editing, it must pass through one of Wikipedia's internal hosts. From this request, the project name, the size of the page, and the title of the page is collected. The resulting information is written into a page view raw dataset where one record is the hourly aggregate of the pageviews/requests and size of an individual page.
Figure 5 exemplifies five records of data of the initial dataset:

```
en.b 19th_Century_Literature 1 8422
en.b 360_Assembly/360_Instructions/BC 1 12217
en.b 360_Assembly/360_Instructions/LM 2 0
en.b 360_Assembly/Branch_Instructions 1 53068
en.b 360_Assembly/Pseudo_Instructions 1 8853
```

Figure. 5: Record structure example

Each record has four attributes separated by spaces. The first attribute refers to the language the page is written in. In the example above, all records start with *en* followed by a dot and as second sub-parameter, e.g., the first line has *en.b*. The *en* refers to the Wikipedia language, which in this example is English. The second parameter after the dot indicates what type of page was requested. Attributes that do not contain any second parameter correspond to Wikipedia projects. The second attribute is the name of the retrieved page. In the example above, for the first record this would be *19_Century_Literature*. The third attribute indicates the number of times a specific page was requested within that hour. These request numbers are not solely unique page visits. The last attribute is the size of the re-turned content. In the given example, the first record had just one request which accounted in total for 8422 response bytes. Each hourly dataset is composed of millions of records with this structure. Although the raw data presents an enormous quantity of data, the resulting dataset was significantly reduced by aggregating the records in an hourly basis instead of using unique page requests. A final single month dataset can have between 672 and 744 records, which is the equivalent of all the hours in February and all the hours in a month with 31 days.
Two different variations of datasets were tested. The first variation contained only two different attributes, aggregate number of generated load and aggregate number of requests. Each record contained the aggregate value of an hour. The second dataset variation was enriched with seven additional attributes. The at-tributes are the average number of requests, the average generated load, the standard deviation of the number of requests, the standard deviation of the load and the number of unique requested wiki pages. These attributes were also hourly aggregated. All the data used in the datasets was selected to contain English only requests. Figure 6 presents the steps involved in the data transformation process, from the data logs into a valid dataset record.
The data logs contain data regarding the hourly aggregates of each wiki page request. As such, this first step is to read line by line the data logs. During this procedure, an algorithm tries to find and correct anomalies in a line, e.g., the number of requests and/or load may contain text information. In such situations the algorithm identifies and removes the incorrect data, re-using, if possible, the remaining information. Each line is then divided by white spaces, creating a total of four different attributes.

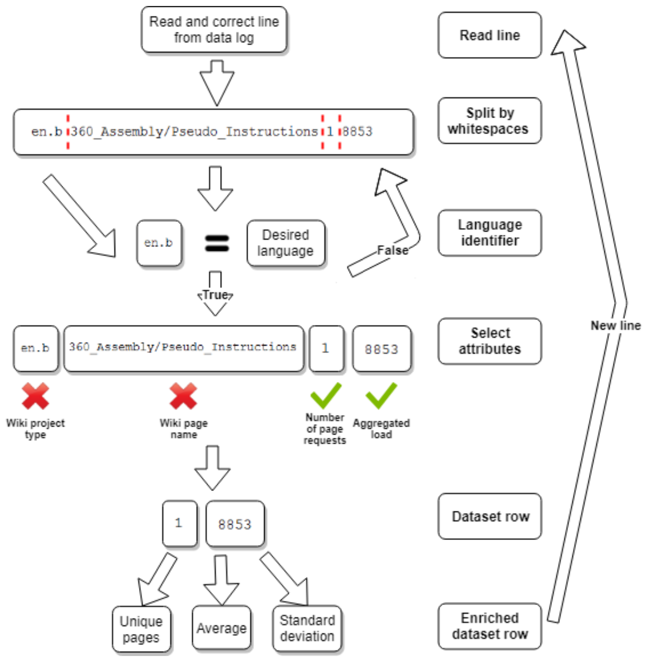


Figure. 6: Data transformation: data log row transformed into a dataset record.

From these four attributes, the attribute project type is used to identify the language, which in this paper is English. If the row is not English, the row is discard-ed without further treatment. Project type and page name are later discarded from the final dataset. This is mainly because each row from the final dataset refers to the hourly aggregate of the desired language and these attributes refer to singular pages.
When the process of creating a simple dataset row is finished, a data row enrichment process is initiated. This process is responsible for creating attributes that will be used in the dataset enrichment phase, which, as mentioned above, is the second dataset variation. When all data of a single data log file is successfully treated, the algorithm passes to the next hour, repeating the process until all the data logs of a month are treated. Figure 7 exemplifies both the baseline dataset and the enriched dataset.

B. The load problem

Server load prediction can be used in many different scenarios. However, scenarios with high activity, either in terms of requests/connections or generated load, tend to stand out. Wikipedia falls into such category. Each day, an enormous quantity of load is generated in the servers, which of course takes a toll on the servers. Correct adjustment of resources is utmost important when creating a reliable solution for the users. An overview of the problem that servers face every day, and which falls in the cloud computing application area, is presented in Figure 8. The example presented uses Wikipedia as a case study. Nevertheless, this problem can be

[7]https://dumps.wikimedia.org/other/pageviews/2016

found on a variety of load forecasting applications.

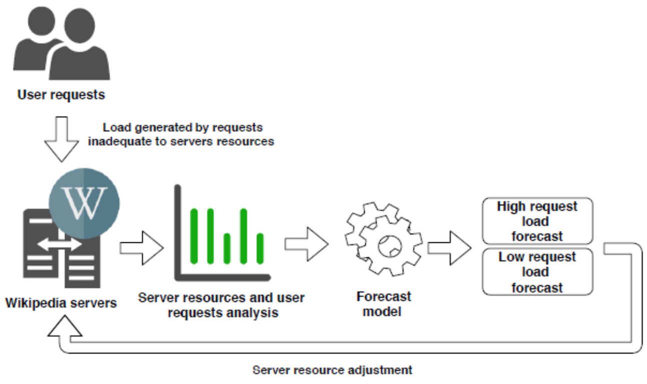

User requests
Load generated by requests inadequate to servers resources
Wikipedia servers
Server resources and user requests analysis
Forecast model
High request load forecast
Low request load forecast
Server resource adjustment

Figure. 8: Wikipedia problem overview.

Wikipedia servers set an initial resource configuration in accordance with the foreseen incoming load. This type of configuration must however be adapted throughout the day. This is done to guarantee the Service Level Agreements (SLA), to optimise the servers from an economical perspective and to increase hardware's lifespan.

By analysing the load patterns of previous data, a model can be constructed that predicts incoming load. Although, since the load is generated mainly by human interaction, past load patterns might not be enough to create a reliable model. The inclusion of extra variables, such as, statistical metrics, can increase the model's accuracy. With a created model, resources can be adjusted upfront for load peaks and low points, thus maintaining the adequate resources for the SLA.

C. Experimental setup

The tests were conducted using Wikipedia's data from January 2016 to August 2016[8], which were the most recent data at the time of this study. Time-window values of 1, 2, 4, 6, 8, 24 and 168 hours were applied along with granularity values of 1, 2 and 7 for experiments where GA where not used.

Linear regression, ANN and SVM were used to create predictive models. GA were chosen for optimisation of the feature values of the datasets. The configurations of the learning methods were the same on all the experiments, including the ones were the GA was used. The difference relied in the features used for training each learning method.

On a first experimental setup, the models were designed to predict the next hour of the incoming load for the Wikipedia servers using only a single month worth of data. Later on, the models were adapted to predict the upcoming hour using a combination of multiple months. The models were created using a split ratio of 70/30, i.e., 70% of the data was used for training the model, while 30% was used for testing. The record order after the splitting remained unaltered.

1) Evaluation metrics

In classical regression models the methodology adopted when choosing a model is the minimisation of the empirical estimation of the Mean Square Error (MSE). Other measures such as the Mean Absolute Error (MAE) or the Huber loss can also be used for robustness reasons. In this paper we opted however to use the MAPE as a measure of quality for regression models.

MAPE measures the accuracy of a forecasting method. It expresses the accuracy as a percentage. MAPE is calculated as follows:

$$MAPE = \sum_{t=1}^{n} \frac{|PE_t|}{n} \tag{6}$$

where A_t is the actual value, and F_t is the forecasted value. When using MAPE, it is possible to compare results between models, since it is scale independent. MAPE is also frequently used due to its very intuitive interpretation in terms of relative error. An example of this is the use of the MAPE in finance. Gains and losses are often measured in relative values. Real life examples of MAPE are frequently used when the value to predict is recurrently positive, i.e., above zero, which is the case for the data used in this study. In [34] the authors advocate the use of MAPE for forecasting problems, especially in situations where enough data is available.

2) Learning methods parameterization

The configuration adopted for each learning method, tried to achieve the best results from an average point of view. After some preliminary experiments, the following parameterizations were chosen.

The linear regression model uses the Akaike criterion [35] for model selection, with the ability to deal with weighted instances. The M5 selection method is used. This method steps through the attributes removing the one with the smallest standardised coefficient until no improvement is observed in the estimate of the error given by the Akaike information criterion. For the ridge parameter a value of 1.0e-8 was chosen.

The ANN model was created with a Multilayer Perceptron architecture and us-es backpropagation to classify the instances. The number of neurons was defined dynamically since our dataset varies in the total number of attributes according to the used variant. The hidden layers were defined by the formula:

$$Hiddenlayerneurons = ((NA + NoN))/2, \tag{7}$$

where *NA* is the number of input attributes and *NoN* is the number of output neurons. Only a linear output unit with no threshold is used. A learning rate of 0.3 was chosen. The learning rate is a proportionality constant that defines the amount of change of the weights. The use of 0.3 prevented the network from diverging from the target output, as well as improved the general performance. A training time of 500 iterations was defined. Lastly, a momentum of 0.2 for weights updating was applied during the learning process.

The SVM model implemented was based on the *SMOreg* variation. This SVM variation implements the support vector machine for regression. The *RegOpti-mizer*, which is the learning algorithm, was defined as *RegSMOImproved* with an e of 1.0e-12 and tolerance of 0.001. The *C* parameter, which defines a margin of tolerance where no penalty is given to errors, was set to 1. With a high parameter value, SVM will try to choose a smaller-margin hyperplane with fewer support vectors. On the contrary, a very small value of *C* causes the

[8]https://dumps.wikimedia.org/other/pageviews/2016

SVM to look for a larger-margin separating hyperplane, even if that hyperplane classifies more points. For the SVM kernel, the Polynomial kernel was used, since it is mostly used in non-linear modelling.

D. Baseline scenario case study

Wikipedia's data from January 2016 to August 2016 was used in this test scenario. Time window values of 1, 2, 4, 6, 8, 24 and 168 hours were applied and granularity values of 1, 2 and 7 were used. A grid search approach was used and the different possible combinations of these attributes throughout different months were tested with the three different learning methods.

1) Single month

For the single month prediction tests, only datasets containing data from a single month were used. The prediction scope lied always in the next hour. The average MAPE values per learning method across all months are presented in the table below.

	Linear Regression			ANN			SVM		
	Average	Smallest	Highest	Average	Smallest	Highest	Average	Smallest	Highest
Jan.	2.15	1.33	4.24	2.60	1.36	5.36	1.97	1.20	3.23
Feb.	1.68	0.97	3.81	2.34	1.16	9.75	1.56	0.96	3.81
Mar.	2.54	1.05	5.22	2.98	1.22	5.79	2.60	1.16	5.17
Apr.	3.52	1.37	10.78	3.50	1.30	12.79	2.30	1.48	10.64
May	2.65	1.33	5.32	2.66	1.61	5.06	2.26	1.32	4.66
Jun.	1.72	1.02	10.48	2.20	1.11	3.76	1.74	1.02	4.63
Jul.	4.99	2.06	5.60	6.54	1.63	8.42	4.97	2.13	5.59
Aug.	2.18	0.93	6.58	2.60	1.21	8.06	2.10	1.04	6.41

Table 1: Average, Smallest, and Highest MAPE values for all months for Linear Regression, ANN and SVM

Table 1 shows that most average errors fall between 2% and 3%. On average, the best performing month was February with the lowest average errors being 1.68% and 1.56%, with linear regression and SVM respectively. An error lower than 2% was also achieved for January and June. The worst results were obtained for July, with errors between 4.97% and 6.54%. In short, most of our runs led to very appealing results, with average errors all below the 7% mark. April was the worst performing month from the absolute point of view, achieving errors above the 10% mark. The best performing month was February when using SVM, with a configuration of granularity 2 and time window 8.
Figure 9 depicts the resulting chart for the month of February when putting prediction and real value side by side. The two depicted red arrows represent the peak values that occurred in that month. It is possible to confirm that, through the application of our methodology, the created solution could successfully predict these peak values.
Through the various tests that were taken, it was possible to verify which configurations achieved the best results on average across all months. We used these configurations to create a general model and compared the results with the best achieved model. For linear regression the general model had granularity 7 and time window 8, for ANN granularity 1 and time window 8 was used and, lastly, for SVM granularity 7 and time window 8 were used. Table 2 shows a comparison of the results achieved with these models and the models that obtained the best results for each month.

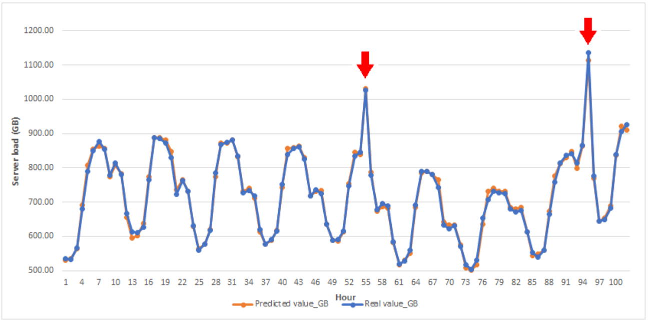


Figure. 9: Graphical representation of predicted and real values resulting from the SVM model regarding the month of February:a) first half of the month with regular data pattern; b) second half of the month with irregular data pattern

	Linear Regression			ANN			SVM		
	General	Smallest	Difference	General	Smallest	Difference	General	Smallest	Difference
Jan.	1.33	1.33	0	1.36	1.36	0	1.38	1.2	0.18
Feb.	1.21	0.97	0.24	1.18	1.16	0.02	1.24	0.96	0.28
Mar.	1.28	1.05	0.23	1.53	1.22	0.31	1.35	1.16	0.19
Apr.	1.6	1.37	0.23	1.53	1.3	0.23	1.56	1.48	0.08
May	1.92	1.33	0.59	2.06	1.61	0.45	1.94	1.32	0.62
Jun.	1.98	1.02	0.96	1.32	1.11	0.21	1.57	1.02	0.55
Jul.	2.39	2.06	0.33	1.63	1.63	0	2.42	2.13	0.29
Aug.	1.39	0.93	0.46	2.22	1.21	1.01	1.42	1.04	0.38

Table 2: General model MAPE values for linear regression, ANN and SVM using the basic dataset

From these results it is possible to verify that the general models achieve satisfying results when compared to the best ones. All results have a MAPE difference less than 1%. The MAPE difference measures the variance between the smallest MAPE value, which had specific granularity and time window configuration for each month, and the model with the general MAPE configuration. Most importantly, low MAPE differences indicate that the model is versatile enough so that less adjustment is needed when applying to different datasets.

2) Multiple months

For the multiple month's test, datasets containing the aggregate of sequential months were used, i.e., the first month combination is the combination of the months July and August, while the second combination is the aggregate of the month June, July and August, creating a total of seven combinations containing all months. The prediction scope was also for the next hour. The average MAPE values across all months are presented in the table below.

	Linear Regression			ANN			SVM		
	Average	Smallest	Highest	Average	Smallest	Highest	Average	Smallest	Highest
1st combi	1.94	1.26	3.28	2.62	1.33	4.99	1.76	1.21	3.20
2nd combi	1.88	1.21	4.90	3.06	1.32	5.71	2.40	1.16	5.04
3rd combi	1.90	1.13	7.17	3.35	1.18	5.26	2.71	1.12	6.64
4th combi	1.73	1.10	4.78	2.66	1.22	5.52	2.13	1.09	4.31
5th combi	2.24	1.16	11.38	2.22	1.26	5.89	1.95	1.18	15.16
6th combi	2.04	1.21	4.10	3.23	1.33	14.21	2.78	1.22	3.73
7th combi	2.04	1.20	4.40	3.55	1.24	8.94	2.88	1.22	3.81

Table 3: Average MAPE values for all month combinations for linear regression, ANN and SVM

Table 3 shows that most average errors fall between 1.7% and 3.5%. On aver-age, the best performing month combination was August and July for the SVM with the lowest errors being 1.76%, and for linear regression with a MAPE of 1.73% with the months of August, July, June, May and April. The highest average error was found when using ANN and combining all months, the MAPE was of 3.55%. It is possible to verify that the variance across the different combinations is less than the one in the single month counterpart.

Regarding the learning models, all learning methods performed well, with linear regression being the best performing method. ANN were the worst performing method. It was the only learning method that achieved average errors higher than 3%, with the highest average error being 3.55%. When comparing the learning models, with the single month variation, it is possible to conclude that, on both variations, ANN are the worst performing learning method. Linear regression and SVM are very similar in results on both variations. From an absolute point of view, the average values and the absolute lowest values, scored fairly near values.

In the same way as in the single month variation, in the combined month variation a general model was created using the average best configurations. For linear regression the general model had granularity 1 and time window 24, for ANN granularity 1 and time window 4 and for SVM granularity 1 and time window 24. Table 4 presents the synthesis of these results.

	Linear Regression			ANN			SVM		
	General	Smallest	Difference	General	Smallest	Difference	General	Smallest	Difference
1st combi.	1.26	1.26	0	3.23	1.33	1.90	1.21	1.21	0
2nd combi.	1.21	1.21	0	1.62	1.32	0.30	1.16	1.16	0
3rd combi.	1.13	1.13	0	1.59	1.18	0.41	1.12	1.12	0
4th combi.	1.10	1.10	0	1.22	1.22	0	1.09	1.09	0
5th combi.	1.16	1.16	0	1.26	1.26	0	1.18	1.18	0
6th combi.	1.21	1.21	0	1.40	1.33	0.07	1.22	1.22	0
7th combi.	1.20	1.20	0	1.31	1.24	0.07	1.22	1.22	0

Table 4: General model MAPE values for linear regression, ANN and SVM using basic dataset with month combination

The results achieved in this test indicate that a general model is almost always the best resulting model. Linear regression and SVM presented MAPE difference of 0 in all instances. This means that the lowest MAPE is also the general model. These results induce us to believe that through the use of month combination a general model is able to be created, with a generalisation high enough to be used in other test scenarios.

E. Enriched dataset scenario

Just as in the basic scenario, data from January 2016 to August 2016 was used. History values of 1, 2, 4, 6, 8, 24 and 168 hours and granularity values of 1, 2 and 7 were also used. The additional attributes used in the enhanced version were average load, number of requests, standard deviation for the load and number of request and the unique page visits.

1) Single month

The testing approach taken in the enhanced scenario is equal to the basic one, only differing in the dataset used. The average MAPE values per learning method across all months are presented in the table below.

	Linear Regression			ANN			SVM		
	Average	Smallest	Highest	Average	Smallest	Highest	Average	Smallest	Highest
Jan.	2.73	0.89	5.23	1.09	0.76	1.37	1.31	0.81	1.71
Feb.	3.15	0.87	7.05	1.07	0.79	1.21	1.46	0.84	1.69
Mar.	3.65	1.50	8.41	1.50	0.77	1.84	1.73	0.87	2.02
Apr.	3.02	0.94	5.33	1.24	0.85	1.40	1.50	0.98	1.84
May	1.97	0.88	3.56	1.38	0.85	1.71	1.29	0.87	1.43
Jun.	1.87	0.88	3.44	1.38	0.98	1.53	1.06	0.87	1.21
Jul.	2.55	0.93	4.78	1.63	0.99	1.95	1.57	0.95	1.87
Aug.	3.64	1.04	4.10	1.24	0.89	1.56	2.43	1.08	3.65

Table 5: Average MAPE values for all months for linear regression, ANN and SVM

Table 5 shows that highest average errors fall between 2% and 3% and with the linear regression model. On average, the best performing learning methods were ANN and SVM, having the ANN a slight advantage. The best performing months were January and February for the ANN and June for the SVM, with all values being around the 1% mark. From an absolute point of view, all learning methods had models with errors below the 0.90% mark. The best result was found in January by using ANN and a configuration with granularity of 2 and time window of 24. The worst result was found on the month of March and using linear regression. Linear regression had an overall worse performance when compared to the remaining learning methods. As for the baseline scenario, a general model was created using the average best configurations. For linear regression the general model had granularity 1 and time window 24, for ANN granularity 1 and time window 4 and for SVM granularity 7 and time window 8. Table 6 presents the synthesis of these results.

	Linear Regression			ANN			SVM		
	General	Smallest	Difference	General	Smallest	Difference	General	Smallest	Difference
Jan.	1.07	0.89	0.18	0.83	0.76	0.07	1.01	0.81	0.20
Feb.	1.03	0.87	0.16	0.87	0.79	0.08	1.03	0.84	0.39
Mar.	1.64	1.50	0.14	0.78	0.77	0.01	1.11	0.87	0.24
Apr.	0.94	0.94	0	0.89	0.85	0.04	0.98	0.98	0
May	0.93	0.88	0.05	1.01	0.85	0.16	0.96	0.87	0.09
Jun.	1.09	0.88	0.21	1.03	0.98	0.05	1.05	0.87	0.18
Jul.	0.98	0.93	0.05	1.07	0.99	0.08	1.01	0.95	0.06
Aug.	1.18	1.04	0.14	1.02	0.89	0.13	1.21	1.08	0.13

Table 6: General model MAPE values for linear regression, ANN and SVM

The results presented, show that the general model achieves very satisfying and promising results. The use of the enhanced dataset evens out the MAPE difference. All results have a MAPE difference lower than 0.39%.

2) Multiple months

As mentioned in the previous section, the tests taken in this scenario only varied in the dataset used. The average MAPE values per learning method across all months are presented in the Table below.

	Linear Regression			ANN			SVM		
	Average	Smallest	Highest	Average	Smallest	Highest	Average	Smallest	Highest
1st combi	4.03	2.43	10.07	1.54	1.31	5.46	1.63	1.22	4.13
2nd combi	3.67	2.93	9.92	1.61	1.18	4.81	1.86	1.41	6.85
3rd combi	4.13	2.57	8.01	1.47	1.08	8.33	1.42	1.42	5.78
4th combi	3.12	1.81	14.45	1.70	1.35	6.78	1.64	1.36	5.11
5th combi	3.07	1.79	13.57	1.39	1.1	5	1.32	1.2	6.88
6th combi	3.13	2.52	6.14	1.42	1.08	9.57	1.39	1.33	4.06
7th combi	2.89	1.89	7.86	1.43	1.05	9.21	1.59	1.51	5.28

Table 7: Average MAPE values for month combinations: linear regression, ANN and SVM

Table 7 shows that most average errors fall around the 1.5%

mark for the learning methods ANN and SVM. Linear regression shows the worst results when using an enriched dataset. On average, the best performing month combination was the 5^{th} and 7^{th}, with the lowest errors being 1.32%, 1.39% and 2.89%, for the SVM, ANN and linear regression respectively. Regarding the learning algorithms, ANN presented the best results, with occasionally SVM performing better then ANN. Linear regression was the worst performing learning method, achieving average errors higher than the 4% mark with the 1st and 3rd month combination. For the combined month variation with the enriched dataset, a general model was created using the average best configurations. For linear regression the general model had granularity 1 and time window 24, for ANN granularity 1 and time window 8 and for SVM granularity 1 and time window 24. Table 8 presents the synthesis of these results.

	Linear Regression			ANN			SVM		
	General	Smallest	Difference	General	Smallest	Difference	General	Smallest	Difference
1st combi.	3.76	2.43	1.33	1.31	1.31	0	1.22	1.22	0
2nd combi.	3.04	2.93	0.11	1.18	1.18	0	1.41	1.41	0
3rd combi.	2.57	2.57	0	1.08	1.08	0	1.42	1.42	0
4th combi.	1.81	1.81	0	1.35	1.35	0	1.36	1.36	0
5th combi.	1.99	1.79	0.2	1.10	1.10	0	1.20	1.20	0
6th combi.	2.52	2.52	0	1.08	1.08	0	1.33	1.33	0
7th combi.	2.51	1.89	0.62	1.05	1.05	0	1.51	1.51	0

Table 8: General model MAPE values using basic dataset with month combination

The results show that most average errors fall between 1 and 1.5%, for the learning methods ANN and SVM. Linear regression was the worst learning method, as for the basic dataset scenario. On average, the best performing month combination was the 7^{th}, with the lowest errors being 1.05%, and using ANN. The conclusions achieved in this test run, are identical to the basic dataset scenario. The learning methods were better able to generalise when using a data source of multiple months.

F. Genetic algorithm scenario

The GA applied in this paper tries to optimise the dataset from the perspective of its features. Each individual consists of various genes, containing the variables to optimise. This process is referred as coding of individuals. The variables of the problem are the configuration values of granularity, time window, the use or not of the number of requests attribute, the use or not of normalization and the months of data used. The evaluation of individuals is done through a fitness/objective function. The objective function considered is the Mean Absolute Percentage Error (MAPE), resulting from the created models using either linear regression, ANN or SVM. The MAPE function is defined in Equation 6. The GA in this paper tries to minimise the value of MAPE.

1) Encoding of individuals

Each individual is composed by five genes. Figure 10 presents an example of an encoded individual.

Each gene represents a possible value when configuring the features in a dataset. Each individual in the initial population is generated with five random gene values, to ensure greater diversity. Although the value for each gene is random, the values are always within a predefined range to ensure that the

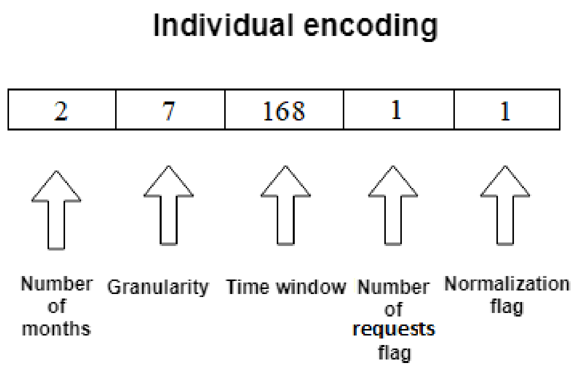


Figure. 10: Example of the encoding of an individual.

created individuals have valid configuration values. For the number of months, the range falls between 1 and 8, since the study uses a total of 8 different months. The normalisation flag and request flag vary between 0 and 1. In the event were a flag is activated, normalization and/or the use of the number of requests attribute are triggered. Granularity and time window vary between 1 and 7 and 1 and 168, respectively. These two ranges could have a higher upper limit. However, this range was defined in order to replicate the same limits as the ones used in the non-GA approach.

2) Algorithm configuration

The initial population consists in a set of randomly generated individuals so that the genes values are inside the respective ranges. Regarding the selection process, the binary tournament method was used [24]. This method repeats n times the following procedure, where n is the size of the population: two individuals are chosen randomly from the current population. The individual with the best fitness is selected and inserted in the intermediary population *P1(t)*.
After the selection process, for each pair of individuals in *P1(t)*, the recombination operator is applied with some probability previously defined. In this work, the single cut recombination operator is used: For each pair of individuals in *P1(t)* that should be recombined, a random cut-off point is defined. The genes to the right (or left) of the cut-off point are exchanged between the parents thus giving rise to two descendants.
The mutation operator is applied after the recombination one. The implementation of the mutation involved an individual-wise verification of the occurrence of mutation, according to a given probability. In cases were mutation is to be applied, a gene is randomly selected. The selected gene value is then replaced by a randomly generated one bound to its valid range.
The selection of individuals to be part of the next population is the last step of each iteration. The selection of individuals is made by considering the initial population of that iteration, *P(t)*, and its descendants *P2(t)*. Several individuals are then chosen, so that the new population is made up of a pre-defined percent-age of the best individuals of that iteration. This percentage defines the elitism level of the algorithm. For example, 75% of elitism means that 75% of the best individuals are selected to be part of the new population. The remaining individuals are chosen at random.
After some preliminary experiments, the GA parameters

were established as follows: population size equal to 100 individuals, number of generations equal to 100, probabilities of recombination and mutation equal to 80% and 10%, respectively, and elitism equal to 100%. The chosen elitism value means that the next population is composed of the best individuals among the parents, *P(t)*, and their descendants, *P2(t)*.

3) GA results vs non-GA results

In order to test the GA approach, the basic dataset variation was used. This dataset variation was the lightest version from the computational perspective and did not present a large difference in results when compared to the enriched version, hence why it was used in the GA. The computational time taken by each variation, however, differed enormously. Being the GA approach for its own already heavy from a computational perspective, it was opted to choose the lighter dataset version.

Table 9 presents, for comparison purposes, a synthesis of the tests using the basic dataset, the enhanced dataset and the basic dataset using GA, for single months.

	Basic dataset				Enriched dataset				GA model			
	MAPE	Gran	TW	Alg.	MAPE	Gran	TW	Alg.	MAPE	Gran	TW	Alg.
Jan.	1.20	7	4	SVM	0.76	2	24	ANN	0.73	4	97	SVM
Feb.	0.96	2	8	SVM	0.79	2	8	ANN	0.66	6	92	SVM
Mar.	1.05	7	8	LR	0.77	1	4	ANN	0.81	3	28	LR
Apr.	1.37	2	24	LR	0.85	2	8	ANN	0.76	2	153	ANN
May.	1.32	7	1	SVM	0.85	7	4	ANN	0.78	7	61	SVM
Jun.	1.02	1	4	LR/SVM	0.87	1	8	SVM	0.83	2	75	LR
Jul.	2.01	1	4	ANN	0.93	1	24	LR	0.84	4	79	ANN
Aug.	0.93	7	8	LR	0.89	1	24	ANN	0.87	7	36	SVM

Table 9: Minimum MAPE values with respective pre-processing configuration using single

As it is possible to verify, the basic scenario only achieved a MAPE lower than 1% for February and August, with 0.96% and 0.93%, respectively. When using the enhanced dataset, all results were under the 1% mark; the best result was for January with 0.76%. When using the GA approach, the results outperformed the basic and enhanced test scenario. The best results were achieved for February with granularity of 6, time window of 92 and the SVM learning method, with a MAPE of 0.66%. When comparing the general model results, with the created GA model, it is possible to compare which learning method requires more parameterization optimisation, in order to increase the model's performance. Table 10 shows the summary of the general model results, along with GA model results.

	Basic dataset		Enriched dataset		GA model	
	MAPE	Alg.	MAPE	Alg.	MAPE	Alg.
Jan.	1.33	LR	0.83	ANN	0.73	SVM
Feb.	1.18	ANN	0.87	ANN	0.66	SVM
Mar.	1.28	LR	0.78	ANN	0.81	LR
Apr.	1.53	ANN	0.89	ANN	0.76	ANN
May.	1.92	LR	0.93	LR	0.78	SVM
Jun.	1.32	ANN	1.03	SVM	0.83	LR
Jul.	1.63	ANN	0.98	LR	0.84	ANN
Aug.	1.39	LR	1.02	SVM	0.87	SVM

Table 10: Minimum MAPE values using single month test variation

As can be seen, when trying to create the best suitable model for each month variation and considering all three learning models, the GA model still achieves the best results. The enriched dataset version comes second in terms of lowest MAPE and, finally, the basic dataset comes last. Although having the highest MAPE from all three variations tested, the basic scenario is still able to achieve models with MAPE of lower than 2%. From the learning method perspective, the results seem to indicate that linear regression and ANN create the better models. When taking into consideration just the enriched dataset, ANN is the clear winner. Nevertheless, when taking the whole case study in perspective, SVM seems to achieve the best results, although this is mainly due to the optimisation that the GA model does. In summary, when creating a generalised model, one can take three different variations, with all achieving results under the 2% mark. The factor that most varies between these three variations is the computation time. By increasing the complexity of the learning process and dataset, better results are achieved. However, the computational time is directly proportional to these factors.

VI. Conclusions and Future Work

This paper described an approach to a server load prediction problem on Wikipedia's load. The approach taken involved a set of data transformations, pre-processing and a genetic algorithm wrapper to optimise all dataset variables. The results achieved indicate that the approach applied has a high suitability for the data used. The viability of the methodology is also validated due to the use of real-life data. Lastly, the use of different training and test sets across multiple months leads us to conclude that the methodology achieves good generalisation on the overall problem, without creating an overfitted model. A basic and an enhanced scenario were tested, showing that linear regression and SVM achieved better results in the basic scenario and ANN achieved the best results when using the enhanced dataset. The best results however were achieved using the basic dataset in conjunction with GA.

Since the approach is applied on different training and test sets across multiple months, results indicate that this methodology increases the generalisation capabilities of the models. We would like to stress that the proposed pre-processing approach is general and can thus be applied in different scopes that involve time series, not only server load forecast.

Future work is suggested in the use of data from different years, the addition of external data, increasing the granularity and time window test range and extending the GA application to the model parameter optimisation.

References

[1] Di Persio, L., Cecchin, A. and Cordoni, F., 2017, "Novel approaches to the energy load unbalance forecasting in the Italian electricity market", Journal of Mathematics in Industry, 7(1), 5.

[2] Ayub, N., Javaid, N., Mujeeb, S., Zahid, M., Khan, W. Z. and Khattak, M. U., 2019, "Electricity Load Forecasting in Smart Grids Using Support Vector Machine", International Conference on Advanced Information Networking and Applications, 1-13.

[3] Cantón Croda, R. M., Gibaja Romero, D. E. and Caballero Morales, S. O., 2019, "Sales Prediction through Neural Networks for a Small Dataset", International Journal of Interactive Multimedia & Artificial Intelligence, 5(4).

[4] Hipni, A., El-shafie, A., Najah, A., Karim, O. A., Hussain, A. and Mukhlisin, M., 2013, "Daily forecasting of dam water levels: comparing a support vector machine (SVM) model with adaptive neuro fuzzy inference system (ANFIS)", Water resources management, 27(10), 3803-3823.

[5] Pukach, P. and Hladun, V., 2018, "Using dynamic neural networks for server load prediction.", Computational linguistics and intelligent systems (2), 157-160.

[6] Lorido-Botran, T., Miguel-Alonso, J., and Lozano, J. A., 2014, "A review of auto-scaling techniques for elastic applications in cloud environments", Journal of grid computing, 12(4), 559-592.

[7] Gross, G. and Galiana, F. D., 1987, "Short-term load forecasting", Proceedings of the IEEE, 75(12), 1558-1573.

[8] Pukach, P. and Hladun, V., 2018, "Using dynamic neural networks for server load prediction", Computational linguistics and intelligent systems (2), 157-160.

[9] Gebreyohans, G. and Gandhi, N., 2018, "Analyzing Children's Data Using Machine Learning: A Case Study in Ethiopia", International Journal of Computer Information Systems and Industrial Management Applications, 10, 154-163.

[10] Ashok, K. and Karamjit B., 2018, "Multiple Classifier System for Writer Independent Offline Handwritten Signature Verification using Hybrid Features", International Journal of Computer Information Systems and Industrial Management Applications, 10, 252-260.

[11] Herbst, N., Amin, A., Andrzejak, A., Grunske, L., Kounev, S., Mengshoel, O. J. and Sundararajan, P., 2017, "Online Workload Forecasting", Self-Aware Computing Systems, 529-553.

[12] Sahu, P., Mohapatra, P. and Parvathi, S. P. K., 2017, "Neural Network training using FFA and its variants for Channel Equalization", International Journal of Computer Information Systems and Industrial Management Applications, 257-264.

[13] Vladimir, V. N., and Vapnik, V., 1995, "The nature of statistical learning theory", Springer Heidelberg.

[14] Silva, C., Grilo, C. and Silva, C., 2018, "Server load prediction on Wikipedia traffic: influence of granularity and time window", Proceedings of the Tenth International Conference of Soft Computing and Pattern Recognition, SoCPaR 2018.

[15] Gori, M. and Tesi, A., 1992, "On the problem of local minima in backpropagation", IEEE Transactions on Pattern Analysis & Machine Intelligence, (1), 76-86.

[16] Seber, G. A. and Lee, A. J., 2012, "Linear regression analysis", 329, John Wiley & Sons.

[17] Khotanzad, A., Afkhami-Rohani, R., Lu, T. L., Abaye, A., Davis, M. and Maratukulam, D. J., 1997, "ANNSTLF-a neural-network-based electric load forecasting system", IEEE Transactions on Neural networks, 8(4), 835-846.

[18] Rojas, I. and Pomares, H., 2016, "Time series analysis and forecasting: selected contributions from the ITISE Conference", Springer.

[19] Boyd, S. and Vandenberghe, L., 2004, "Convex optimization", Cambridge university press.

[20] Vapnik, V. N., 1999, "An overview of statistical learning theory", IEEE transactions on neural networks, 10(5), 988-999.

[21] Bertsekas, D. P., 2014, "Constrained optimization and Lagrange multiplier methods", Academic press.

[22] Forrest, M., M. and S., 1994, "Genetic algorithms and artificial life", Artificial life 1.3, 267-289.

[23] Mitchell, M., 1998, "An introduction to genetic algorithms", MIT press.

[24] Zhang, Q., Cheng, L. and Boutaba, R., 2010, "Cloud computing: state-of-the-art and research challenges", Journal of internet services and applications, 7–18.

[25] Cheng, H. B. and Yang, G., 2005, "Improved AR-based model of host load prediction in computing grid [J]", Computer Applications, 11.

[26] Padala, P., Shin, K. G., Zhu, X., Uysal, M., Wang, Z., Singhal, S. and Salem, K., 2007, "Adaptive control of virtualized resources in utility computing environments", ACM SIGOPS Operating Systems Review, 41, 289-302.

[27] Shao-juna, W., Jun-jiea, C. H. E. N. and Taob, G. U. O., 2012, "Optimized virtual machine deployment mechanism in cloud platform", Computer Engineering, 11.

[28] Ghelardoni, L., Ghio, A. and Anguita, D., 2013, "Energy load forecasting using empirical mode decomposition and support vector regression", IEEE Transactions on Smart Grid, 4(1), 549-556.

[29] Imperato, M., 1991, "An introduction to Z", Bromley, Kent, UK: Chartwell-Bratt.

[30] Hamilton, J., 2009, "Cooperative expendable micro-slice servers (CEMS): low cost, low power servers for internet-scale services", Conference on Innovative Data Systems Research, CIDR'09.

[31] Vasić, N., Barisits, M., Salzgeber, V. and Kostic, D., 2009, "Making cluster applications energy-aware", Proceedings of the 1st workshop on Automated control for datacenters and clouds, 37-42.

[32] Lee Rodgers, J. and Nicewander, W. A., 1988, "Thirteen ways to look at the correlation coefficient", The American Statistician, 42(1), 59-66.

[33] Miltin Mboh, C., Montzka, C., Baatz, R. and Vereecken, H., 2014, "A novel partial grid search approach for handling complex multi-dimensional parameter estimation and state improvement at the catchment scale", EGU General Assembly Conference Abstracts, 16.

[34] Armstrong, J. S. and Collopy, F., 1993, "Error measures for generalizing about forecasting methods: Empirical comparisons: International Journal of Forecasting", 8 (1), 69–80.

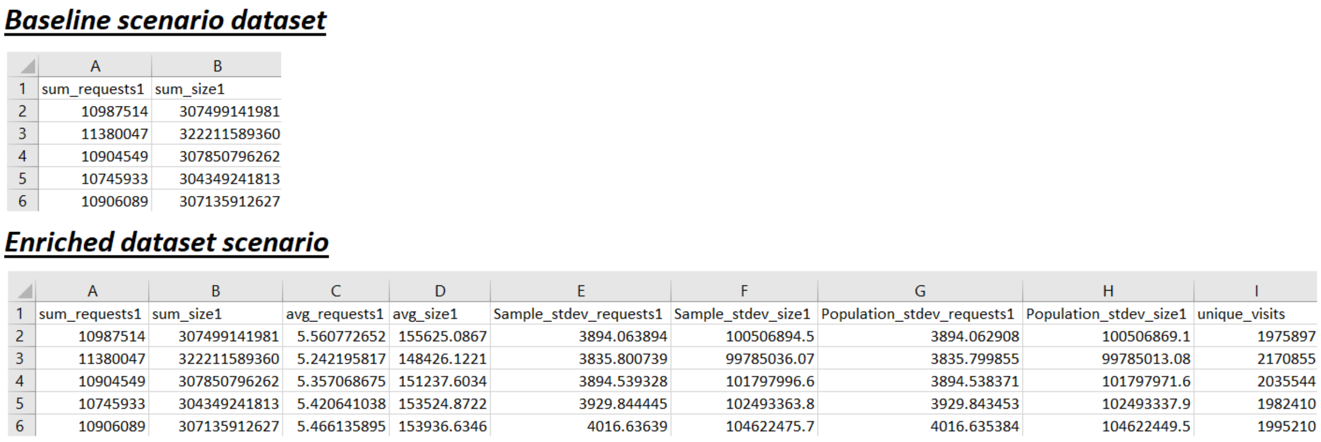
[35] Sakamoto, Y., Ishiguro, M. and Kitagawa, G., 1986, "Akaike information criterion statistics", Dordrecht, The Netherlands: D. Reidel, 81.

Author Biographies

Cláudio A. D. Silva received his B.Sc (2014) in Computer Engineering from the Polytechnic Institute of Leiria (Portugal). He is currently concluding his M.Sc in Computer Engineering - Mobile Computing. His professional career splits between freelancing as a full stack developer and Software Engineer at IBM Deutschland R&D, where among his developer tasks he submitted a patent application for sign language translation framework. His research areas are load forecasting, optimisation through genetic algorithms and sign language translation based on machine learning.

Carlos Grilo graduated in 1997 and received his M.Sc in Computer Engineering in 2003 from the University of Coimbra. He received his Ph.D in Computer Engineering in 2011 from the University of Lisbon. He teaches at the Polytechnic Institute of Leiria, Portugal, since 1997. He is also a researcher at the Computer Science and Communication Research Centre, Polytechnic Institute of Leiria. He has published about 35 articles in both international conferences and journals. His research interests include artificial intelligence, machine learning, evolutionary computation and the evolution of cooperation.

Catarina Silva graduated in Electrical Engineering and received the M.Sc. and Ph.D. in Computer Science from the University of Coimbra, Coimbra, Portugal, in 1997, 2000, and 2009, respectively. She teaches at the Polytechnic Institute of Leiria, Portugal since 1997. She is also a Researcher in the Adaptive Computation Group of the Centre for Informatics and Systems, University of Coimbra. She is the author and co-author of several books, circa 15 journal articles and 80 conference papers. Her research interests include intelligent systems, machine learning and their applications, especially text classification, and mobile application development.



Baseline scenario dataset

	A	B
1	sum_requests1	sum_size1
2	10987514	307499141981
3	11380047	322211589360
4	10904549	307850796262
5	10745933	304349241813
6	10906089	307135912627

Enriched dataset scenario

	A	B	C	D	E	F	G	H	I
1	sum_requests1	sum_size1	avg_requests1	avg_size1	Sample_stdev_requests1	Sample_stdev_size1	Population_stdev_requests1	Population_stdev_size1	unique_visits
2	10987514	307499141981	5.560772652	155625.0867	3894.063894	100506894.5	3894.062908	100506869.1	1975897
3	11380047	322211589360	5.242195817	148426.1221	3835.800739	99785036.07	3835.799855	99785013.08	2170855
4	10904549	307850796262	5.357068675	151237.6034	3894.539328	101797996.6	3894.538371	101797971.6	2035544
5	10745933	304349241813	5.420641038	153524.8722	3929.844445	102493363.8	3929.843453	102493337.9	1982410
6	10906089	307135912627	5.466135895	153936.6346	4016.63639	104622475.7	4016.635384	104622449.5	1995210

Figure. 7: Example of Wikipedia dataset variations.