



International Journal of Information and Decision Sciences

ISSN online: 1756-7025 - ISSN print: 1756-7017

<https://www.inderscience.com/ijids>

Artificial neural networks in the development of business analytics projects

Juan Bernardo Quintero, David Villanueva-Valdes, Bell Manrique-Losada

DOI: [10.1504/IJIDS.2024.10061743](https://doi.org/10.1504/IJIDS.2024.10061743)

Article History:

Received:	12 June 2020
Last revised:	29 June 2021
Accepted:	10 July 2021
Published online:	26 January 2024

Artificial neural networks in the development of business analytics projects

Juan Bernardo Quintero

Faculty of Engineering,
EAFIT University,
Medellín, Colombia
Email: jquinte1@eafit.edu.co

David Villanueva-Valdes and
Bell Manrique-Losada*

Faculty of Engineering,
University of Medellín,
Medellín, Colombia
Email: dvillanueva@udem.edu.co
Email: bmanrique@udem.edu.co

*Corresponding author

Abstract: The accelerated evolution of information and communication technologies, with an ever-growing increase in their access and availability, has become the foundation for the current big data age. Business analytics (BAs) has helped different organisations leverage the large volumes of information available today. In fact, artificial neural networks (ANNs) provide deep data mining facilities to organisations for identifying patterns, predict probable future states, and fully benefit from predictions/forecasts. This article describes three ANNs application scenarios for the development of BA projects, by using network learning for: 1) executing accounting processes; 2) time series forecasts; 3) regression-based predictions. We validate scenarios by implementing an application-case using actual data, thus demonstrating the full extent of the capabilities of this technique. The main findings exhibit the expressive power of the programming languages used in data analytics, the wide range of tools/techniques available, and the impact these factors may have on the BA development projects.

Keywords: artificial neural networks; ANNs; business analytics; data analytics; big data; deep data mining; network learning process; time series forecast; regression-based prediction; activity-based costing; supervised learning; decision making.

Reference to this paper should be made as follows: Quintero, J.B., Villanueva-Valdes, D. and Manrique-Losada, B. (2024) 'Artificial neural networks in the development of business analytics projects', *Int. J. Information and Decision Sciences*, Vol. 16, No. 1, pp.46–72.

Biographical notes: Juan Bernardo Quintero is a software architect and data scientist in a Colombian developer company. He obtained his PhD degree from University of Antioquia in 2015. He is a Professor in areas like artificial intelligence, software architecture and data analytics at the University of

Antioquia, University EAFIT, and University of Quindío. His current research interests and work areas include artificial neural network, distributed ledger technologies, and cloud computing.

David Villanueva-Valdes is an active consultant for several companies in Medellín, Colombia. Also, He is a Professor at the University of Medellín and his research includes machine learning, data science, and data engineering to resolve business questions or more specific for customer data behaviour analysis.

Bell Manrique-Losada is a Software Engineer and Researcher from ARKADIUS Research Group. He obtained her Master and PhD degrees from Universidad Nacional de Colombia in 2006 and 2015, respectively. She is a Professor in areas like software design, requirement engineering, and data analytics at the University of Medellín, Colombia. Her current research interests and work areas include natural language processing, text analytics, software engineering, and engineering education.

1 Introduction

For several decades, artificial neural networks (ANNs), a branch of artificial intelligence, have proposed analogies between brains and computers in terms of their function. For its part, ‘data analytics’ is the discipline responsible for exploring, discovering, and even interpreting data patterns with the purpose of drawing conclusions. However, whenever these activities are used to support business decisions that generate added value to an organisation, they are known as ‘business analytics’ (BAs) (Shmueli et al., 2016). According to Gartner, Inc.’s (2020) IT glossary, BAs includes solutions for designing analysis models, building simulation scenarios, understanding realities and predicting future states.

This article assesses the integration strategies between ANNs and BAs projects by means of application scenarios by using network learning for:

- 1 executing accounting processes
- 2 time series forecasts
- 3 regression-based predictions.

Based on such scenarios, we implement an application-case using actual data to validate and demonstrate the full extent of the capabilities of the strategies.

The article is structured as follows: Section 2 introduces and explains the methodologies used to develop data analytics projects; Section 3 proposes some key scenarios for using ANNs in BAs; Section 4 describes the implementation of these scenarios; Section 5 assesses scenario implementations; and Section 6 finally provides conclusions and proposes future work.

2 Analytical methodologies

Usually recognised as methodologies or processes used to discover hidden knowledge in the data, these methodologies are divided into phases, which usually include data contextualisation, preparation, processing, and analysis (Azevedo and Santos, 2008). Below are some of the most common analytical methodologies:

- *KDD – knowledge discovery in databases*: This methodology involves the following phases: selection, pre-processing, transformation, data mining, and interpretation. In practice, KDD fosters the selection of information collected in database tables. Then, its phases perform operations on these tables for the purpose of obtaining knowledge both iteratively and through an interactive process (Fayyad, 1996). Because it involves the evaluation and interpretation of patterns and models to make decisions, KDD requires a broad and deep knowledge of a study area.
- *SEMMA – sample, explore, modify, model and assess*: This methodology, created by the SAS Institute, according to Azevedo and Santos (2008), refers to a five-stage data mining process that samples, explores, modifies, models, and assesses information, as per its acronym. Although SEMMA is itself a methodology, it is commonly associated with the SAS Enterprise Miner tool (SEMMA SAS Institute, 2020). Based on its structure SEMMA allows the conception, creation, and evolution of data mining projects, helping to present solutions to business problems as well as to find business goals.
- *CRISP – DM cross industry standard process for data mining*: This methodology, developed by Dainmmler Chrysler, SPSS, and NCR, proposes a data life cycle based on a six-state cyclical process (Chapman et al., 2008). It is an iterative process starting with obtaining a business understanding of the problem, and then, an understanding of the dat. Next, such data is processed, and the result produced by the machine learning algorithm is then evaluated. Given the way its stages are related, CRIPS-DM can be considered more flexible when implemented in projects of different nature.

There is another methodology in the analytical trends called ASUM DM which complements CRISP DM with additional phases and details which we describe and compare in Table 1.

Table 1 Data mining methodologies

	<i>KDD</i>	<i>SEMMA</i>	<i>CRISP-DM</i>
Stage	Pre-KDD	---	Business understanding
	Selection	Sample	Data understanding
	Pre-processing	Exploration	
	Transformation	Modification	Data preparation
	Data mining	Modelling	Modelling
	Interpretation/evaluation	Assessment	Evaluation
	Post KDD	---	Deployment

Source: Translation by Azevedo and Santos (2008)

Table 1 compares the processing stages involved in the KDD, SEMMA, and CRISP DM methodologies.

Comparison of the stages of the three methodologies described above indicates that the business understanding stage in CRISP DM evidences the traceability of the information requirements, starting from the domain, passing through knowledge priorities, and ultimately leading to the objectives of the end users (Azevedo and Santos, 2008). KDD can be seen as an implementation of both SEMMA and CRISP DM. In addition, it may be argued that CRISP DM is more complete than SEMMA since the Sample stage should not be carried out without prior business knowledge. For these reasons, and based on the popular relationship between CRISP DM and big data, we use some of the CRISP DM stages to describe the scenarios in this article, as follows:

- 1 business understanding for describing scenarios in Section 3
- 2 data understanding and preparation, modelling, evaluation and deployment in Sections 4 and 5.

In addition to the analytical spectrum, there's another methodology called ASUM DM. It looks like CRISP DM but with extra detailed in some phases: business understanding with *analytics approach*, data understanding with *data requirements* and *data collection*, and deployment with *feedback*. However, such additional phases in ASUM DM do not contribute to explain the integration of ANNs and BAs. Therefore, in this paper we use simple CRISP DM.

3 Scenarios for using ANNs in BAs

There are many types of topologies (architectures) that describe the structure and behaviour of the ANN algorithms, with each one exhibiting features for the solution or representation of different machine learning models (James et al., 2020). Table 2 lists the most common topologies with their corresponding name or acronym.

Table 2 List of neural network topologies

<i>Network name</i>	<i>Acronym</i>	<i>Network name</i>	<i>Acronym</i>
<i>Multi-layer perceptron</i>	<i>(MLP)</i>	Deep convolutional inverse graphics networks	(DCIGN)
Radial basis function	(RBF)	Generative adversarial networks	(GAN)
Hopfield network	(HN)	Recurrent neural networks	(RNN)
Markov chains	(MC)	<i>Long/short-term memory</i>	<i>(LSTM)</i>
Boltzmann machines	(BM)	Gated recurrent units	(GRU)
Restricted Boltzmann machines	(RBM)	Neural Turing machines	(NTM)
Autoencoders	(AE)	Bidirectional recurrent neural networks	(BiRNN)
Sparse autoencoders	(SAE)	Deep residual networks	(DRN)
Variational autoencoders	(VAE)	Echo state networks	(ESN)
Denoising autoencoders	(DAE)	Extreme learning machines	(ELM)
Deep belief networks	(DBN)	Liquid state machines	(LSM)
Convolutional neural networks	(CNN)	Support vector machines	(SVM)

The scenarios implemented will use two of these topologies as follows:

- 1 Analogy: Propagation in an MLP network for the execution of accounting processes.
- 2 Forecasts: Using the learning processes of an LSTM network to process time series.
- 3 Predictions: Using the learning processes of an MLP network to implement regressions.

For scenario 1, there is a latent justification, based on the analogy between an ANN and an *activity-based costing (ABC)* model; however, scenarios 2 and 3 require further argumentation. Although there are many tools, languages, and utilities for using time series and regression techniques, the two scenarios in which they are implemented with neural networks are motivated by the deep learning boom, in which artificial intelligence engines streamline massive paralleling and processing works (Siami-Namini et al., 2019), thus taking advantage of the *graphics processing unit (GPU)* to perform tasks such as matrix multiplication. As deep learning exhaustively uses deep and convolutional neural networks, the integration of time series and regression techniques in environments of this nature provides two notorious advantages (Dong et al., 2017):

- a Developers do not require making great efforts to leverage the good performance provided by the execution environments of artificial intelligence engines.
- b The same techniques, tools, and work environments can be used in the development of BAs projects, which, by their nature, already deliver high levels of heterogeneity.

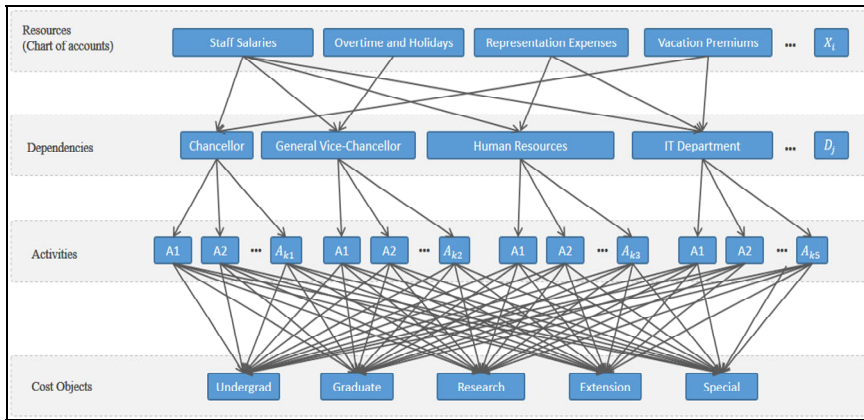
3.1 *Analogy: propagation in an MLP network to process activity-based costing costs*

The ABC system helps companies determine the actual cost of their products or services in order to improve their decision making and solve their indirect cost problems (Jara et al., 2003). Through this system, companies may assess the costs of their products or services with a greater degree of accuracy, taking into account the limitations of traditional methodologies, such as indirect cost distributions or cost allocation, throughout the entire product life cycle.

The main feature elements under this model for this cost structure are the collector, cost levels, and links between them. At the behavioural level, the most important feature is cost distribution, which uses the elements proposed in the structure to generate cost objects. In this model we found the following:

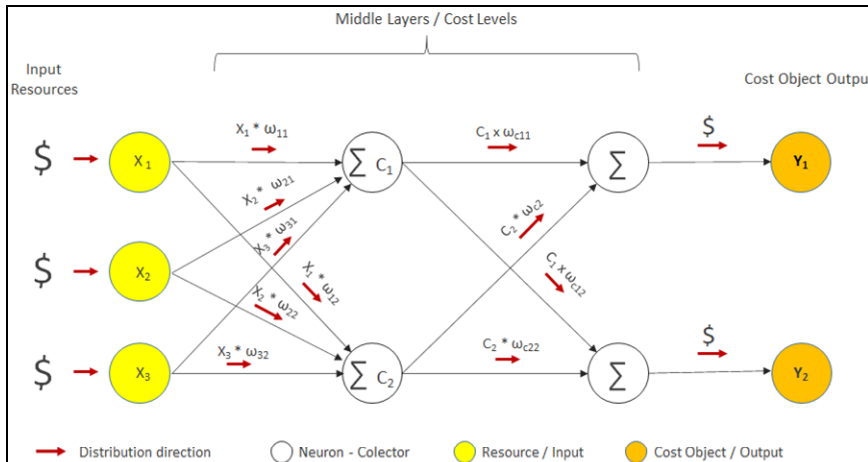
- Each object requires several activities (e.g., design, engineering, purchasing, and production).
- Each activity consumes resources of different categories (e.g., working time of the manager).
- Cost drivers are measures of the activities performed (e.g., number of units produced).

For this scenario, we used actual cost model data from different universities, compiled into an ABC model such as the one provided in Figure 1.

Figure 1 ABC model for a test university (see online version for colours)

Cost distribution is the most important ABC model operation since it realises the main purpose of the cost model. Based on the similarities between an artificial neuron and a cost structure collector from the ABC model, the Forward phase of an MLP network is ideal for distributing costs if:

- 1 the network is modified so that each neuron becomes a collector (collecting neuron), thus eliminating their activation function
- 2 the data preparation cost model is reduced to elements that coincide with the matrix algebra, which also involves the implementation of the forward phase in an MLP ANN, as shown in Figure 2.

Figure 2 Forward phase distribution in a modified MLP (see online version for colours)

The analysis of the data information available within the scope of ABC costs is known as ‘cost analysis’ and, for the particular case of ABC costs, analytics answer questions such as the following: on what is the company spending? (Resources) why does the company spend as much as it does? What does the company do? (Activities) why does the

company do what it does and how much does it cost? What are the returns of our activities? (Cost objects) (Cokins, 2001).

According to Cokins, if a cost methodology, such as ABC, provides greater accuracy in cost calculation, then it also permeates company profit reports and analytics, which positively improve estimates by providing better insight and improving decision making. ABC can monitor the hidden losses and profits of the traditional costing methods. By using the ABC system, activities can be classified as value-added and non-value-added activities, aversely to traditional cost accounting systems where direct materials and labour are the only costs that can be traced directly to the product.

Learning on a MLP network optimises processing costs in a cost distribution network. As a potential contribution, they can also be used in other similar structures in different areas, for instance with algorithm of exploration of a tangle in an Iota network, which is a directed acyclic graph that serves to apply the principles of blockchain in the field of internet of things (<https://www.iota.org/>).

3.2 *Forecasts: learning in an LSTM network to process time series*

From our area of interest, time series are economic variable observations constantly generated by organisations during their business activities (Franses, 1998). Conversely, a forecast model must be associated to an analytical objective, thus turning this forecast model into a means to achieve the corresponding business objective within the organisation (Constancioara et al., 2009).

This scenario proposes the use of a longitudinal cost characterisation over time as a supplement to the previous scenario report. In addition, the process of taking a single cost object, studying its historical cost behaviour and forecasting its future cost represents an important decision-making tool insofar as it provides the future costs of a product or service, thus supplementing business objectives. For example, determining potential losses or profits based on forecasts.

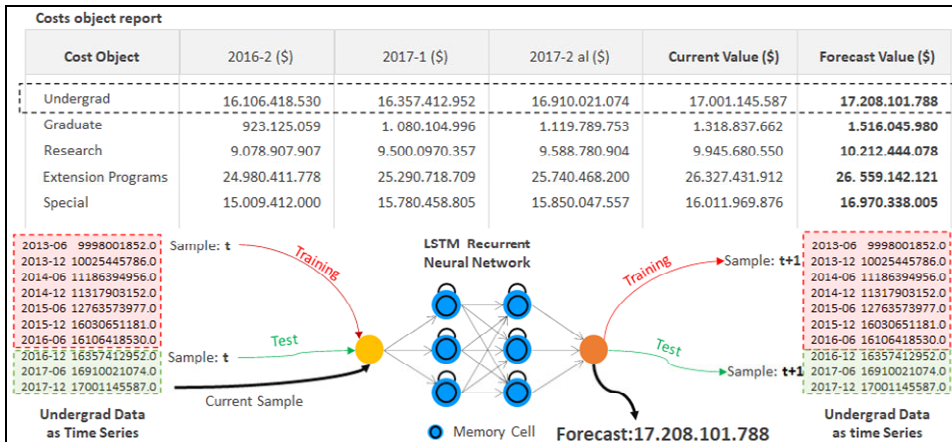
In the university context, and based on the previous scenario, an example of this type of decision would be pricing. The cost of an academic program at a university over semester-long interest periods is a time series in which the costs for the following period are forecasted in a cost report.

ANNs are mathematical models based upon the functioning of the human brain, and are composed of three different layers input, hidden and output layers, composed a set of neurons. Forecasting with ANNs involves two steps: training and learning. Training of feedforward networks – normally performed in a supervised manner. Further, *recurrent neural networks (RNNs)* are a type of neural network topology widely used for the prediction of time series (Bone et al., 2008; Siami-Namini et al., 2019). As RNNs learn from temporary input data contexts to make better predictions, a particular type of RNN that adequately handles the temporality of time series is the LSTM network. Figure 3 illustrates important elements for this scenario:

- 1 unveiling the full potential of the cost report, from cost temporality to a possible future decision-making value
- 2 evidencing cost object data organised as the time series they represent
- 3 the structure of the LSTM network trained using a cost object value as an input parameter to obtain a value forecast for the next period.

We can say that using a LSTM network to forecast a time series is novel in the context of cost analysis. Likewise, it can also be used to take advantage of the availability of libraries optimised for deep learning such as TensorFlow and Keras and thus explore more precise and efficient forecasting mechanisms. This network is able to model the temporal properties of the data and improve on the results obtained from traditional techniques could also be extended to the task of predicting the case outcome.

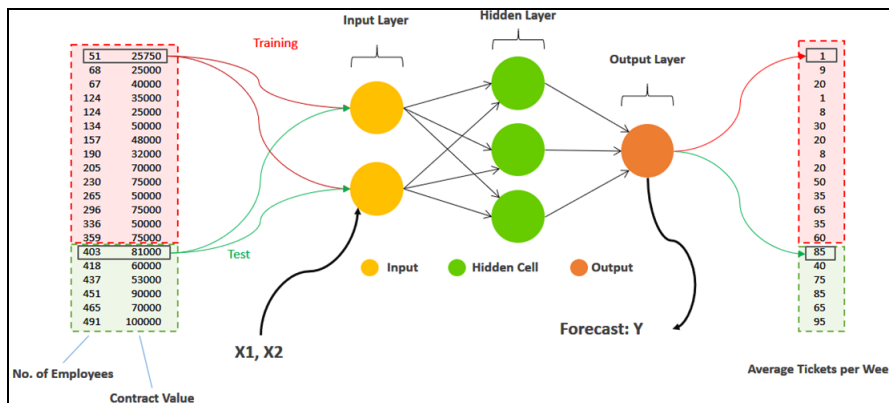
Figure 3 LSTM network with undergraduate dataset for training and forecasting (see online version for colours)



3.3 Predictions: MLP network learning for the implementation of regressions

The word ‘forecast’ suggests ‘prior knowledge’ of a situation; therefore, in this context, it is used to propose a future state based on past states, as it happens with time series. In contrast, the word ‘prediction’ suggests ‘foretelling’ a situation; therefore, in this context, this word proposes the value of a variable based on the values of other variables, as in the case of regression.

Figure 4 Network MLP with a *helpdesk* dataset for training and predicting (see online version for colours)



Regression is a statistical technique used to unveil the existing relationship between several variables; the value of one of these variables (the dependent or target variable) is determined based on the values of the other variables (the independent or predictor variables). In this work, a regression is implemented using an MLP network as an ANN, as we show in Figure 4. Here, the number of neurons in the input layer matches the number of independent or predictor variables, while the single neuron in the output layer corresponds to the dependent or target variable. In the single hidden layer, the number of neurons affects the complexity of the MLP network learning process, which is used to refine the network weights and thus determine the relationship between the input and the output variables. The outputs obtained for each input are compared against the outputs observed to determine the level of correction needed and, therefore, train the network. The experiments show that the simple MLP often outperforms more sophisticated learning models in prediction tasks.

In this context, an MLP network is useful when implementing a multiple linear regression, while other types of regressions, such as those using categorical variables or logistic regression, must use variants of this technique or other typical ANN architectures suitable for said purposes.

A *helpdesk* case study will be used to exemplify regression in the framework of BAs, where the dependent variable is the average number of weekly tickets generated by a client and the independent variables are the number of employees engaged by the client and the total contract value.

In addition to making predictions in a regression, an MLP network makes it possible to take advantage of the potential of GPU arrays for matrix multiplication and to add cheaper processing capacity and diverse uses to those formulated in the field of deep learning.

Table 3 ANN analysis in business analytics scenarios

<i>Technique description</i>	<i>Business case to which it applies</i>	<i>Example of business case</i>	<i>Artificial neural network setup</i>
<i>Analogy:</i> propagation in an MLP network for the execution of accounting processes	Composition or causation relationships between elements of the same type that assign values to each other.	Accounting records, composition of articles in manufacturing, hierarchical relationships.	Components or elements of the same type constitute neurons; the relationships among elements define links and synapses; neurons can be classified into levels.
<i>Forecasts:</i> learning from an LSTM network to process time series	Succession of data measured at certain times and sorted chronologically.	Variation of stock market indicators; profits or expenses by time periods; rainfall in a region.	A neuron in the input layer with the value of the variable to be measured and the time it occurs, a neuron in the output layer with the values of the next period to train and forecast.
<i>Predictions:</i> learning from an MLP network to implement regressions	Affecting a dependent variable based on changes to one or more independent variables.	Price analysis; time for a process; productivity analysis; population growth analysis.	A neuron in the input layer for each independent variable and a neuron in the output layer to represent the dependent variable for training and predicting.

In summary, in Table 3 we present a comparison of the scenarios for each topology, which includes some application suggestions in specific contexts.

4 Scenario implementation

4.1 Analogy: distribute ABC costs with forward phase of a modified MLP network

To implement the distribution scenario, the cost data must be prepared for processing as an MLP ANN (Van Veen, 2020) and meet its propagation needs. Based on this, a collecting neuron, a common artificial neuron without its activation function, was conceived, as shown in Figure 5.

Figure 5 Modification of the artificial neuron for cost distribution, (a) artificial neuron (b) modified artificial neuron (see online version for colours)

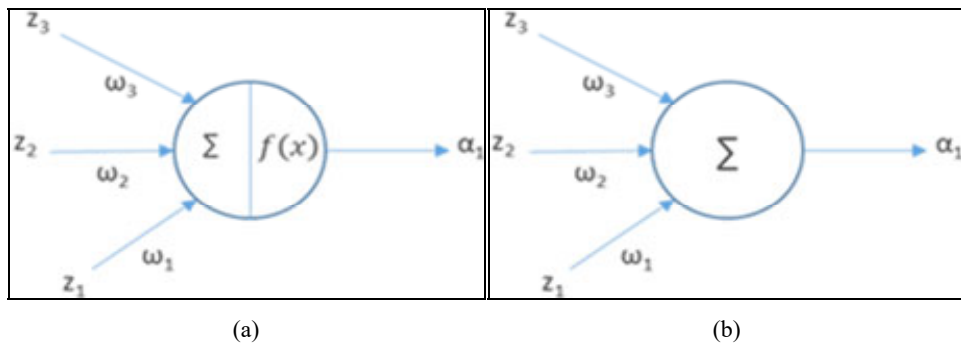


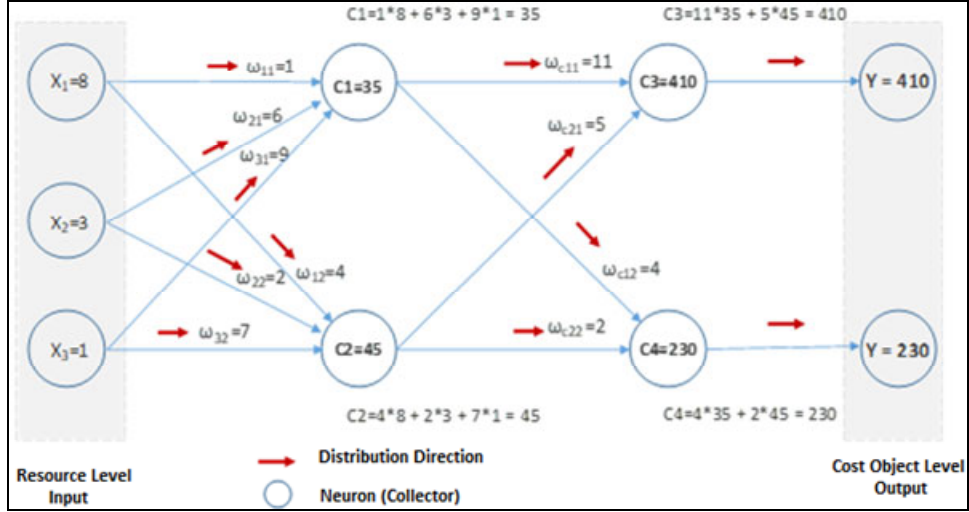
Figure 5 displays a step-by-step example for an MLP architecture, using a simple cost structure with a modified transfer function to propagate forward instead of learning. Please note the calculation of the net function for each neuron or, to be more specific, each collecting neuron. At the end, the business objective requires a report that may list the cost objects from our university example during a given period using the resource tables and the period criteria (links) as input.

Table 4 Characteristics of the ABC cost structure for a fictional university

Name	Total no. of elements	Name	Total elements
Resources	85	Level 1 to level 2 links	237
Dependencies	4	Level 2 to level 3 links	103
Activities	103 (23 – 12 – 49 – 19)	Level 3 to level 4 links	515
Cost objects	5	Criteria types	Weighted (N1 to N2)
MOTP (N2 to N3)			
TAO (N3 to N4)			

Then, the data are prepared by converting the data from Oracle tables to *datasets* that can be processed by our new special cost propagation network. Table 4 provides a summary of the cost structure features. It should be noted that these features also define our network. This data comes from a real business exercise in which we find a common ABC cost structure among seven universities in Colombia, South America.

Figure 6 Cost distribution based on an MLP without the activation function (see online version for colours)



For the distribution process shown in Figure 6, our propagation algorithm defines the calculation of each collecting neuron using equations (1) and (2) since they have no additional activation function.

$$nC_1 = x_1 \times w_{11} + x_2 \times w_{21} + x_3 \times w_{31} \quad (1)$$

$$nC_2 = x_1 \times w_{12} + x_2 \times w_{22} + x_3 \times w_{32} \quad (2)$$

This suggests that ‘propagating’ is an operation that can be written as a matrix algebra operation in which the set of values from the collecting neurons of each level is an array given by equation (3).

$$\begin{bmatrix} c_1 & c_2 \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \times \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix} \quad (3)$$

Therefore, the data must be available in four *datasets*: a 1×85 dataset containing the resource values, an 85×4 dataset for the level 1 to level 2 links, a 4×103 dataset for the level 2 to level 3 links, and a last 103×5 dataset for level 3 to level 4 links.

When the data preparation has been completed, there must be four files: one containing the input data and three matrices relating each element from each previous level with its corresponding element in the next level. These three matrices are named

W_1 , W_2 , and W_3 and will be the weight matrices required within the model adjusted for an MLP network without activation function. Table 5 presents the data preparation results. All data records on these datasets are money in Colombian pesos (COP) and units in billions.

Table 5 Summary of data preparation result elements

#	Matrix	Dataset name	Dimension	Origin
1	X	recursos.csv	1×85	Movements table
2	W_1	enlacesn1n2.csv	85×4	Links table
3	W_2	enlacesn2n3.csv	4×103	Links table
4	W_3	enlacesn3n4.csv	103×5	Links table

This scenario is validated by comparing execution times from a traditional implementation in imperative PL/SQL language against two Python-based implementations using the proposed model, with one implementation consuming data directly from Oracle and the other from *datasets* in CSV format.

To implement the cost distribution scenario, some common tools were selected within the scope of the current analytics, specifically using Spyder as a Python development tool; TensorFlow as a library for high-performance numerical calculations capable of being deployed in different platforms (CPUs, GPUs and TPUs) and from desktop computers to server *clusters* and mobile devices, among others; and finally, Keras, a high-level API, for the implementation of the ANN model. The algorithm is also designed to run on other libraries, such as TensorFlow and Theano, increasing its processing capacity while improving model simplicity.

Figure 7 Distribution results using Spyder (see online version for colours)

```
In [6]: runfile('C:/Omorok/learning/udem/maestria/_ann/distributionCost_using_MLP_TF.py', wdir='C:/Omorok/learning/udem/maestria/_ann')
[[ 1.80512410e+09  1.82723878e+09  1.83030720e+09  1.61263680e+09
  1.91215987e+09]]
```

The results from the execution of the Python script on the described infrastructure can be seen in Figure 7 and it is the result of the data manipulation provided by Table 5 in CSV datasets.

4.2 Cost forecasting: processing time series using an LSTM network

As per the proposed third scenario, cost object data organised period by period may be seen as a period-value pair sequence or time series, defined as statistical data collected, observed or recorded in regular time intervals (Hamilton, 1994). The above allows us to select one of the cost objects from our case study (the cost objects of our test university are undergraduate, postgraduate, research, outreach, and special) to create a *dataset* that meets the suggested period-value shape. For our purposes, undergraduate data will be used to create a forecast model to predict future costs.

Figure 8 *Dataset before and after data preparation*

Cost Object: Undergrad			
Semester	Cost	X	Y
2013-06	9.998.001.852	-1	-0.9921624
2013-12	10.025.445.786	-0.9921624	-0.66061153
2014-06	11.186.394.956	-0.66061153	-0.62305463
2014-12	11.317.903.152	-0.62305463	-0.21019124
2015-06	12.763.573.977	-0.21019124	0.72284036
2015-12	16.030.651.181	0.72284036	0.74447845
2016-06	16.106.418.530	0.74447845	0.81615895
2016-12	16.357.412.952	0.81615895	0.97397611
2017-06	16.910.021.074	0.97397611	1.
2017-12	17.001.145.587		
Original Dataset		Dataset after Data Preparation	

The *dataset* mentioned above, and for which data are available within the business context, is displayed in Figure 8. However, the type of model implementation requires preparing the data in terms of machine learning and, more specifically, in terms of a model within supervised learning. This implies that the *dataset* used was modified to meet the following features:

- *Scale*: In order to be assimilated by the transfer function of each neuron in the LSTM network. In this case, a sigmoid activation function was used, so the data must be on a scale of 1 to -1 .
- *Stationary*: Convert the series to stationary, after determining whether the series is non stationary, because the series being non-stationary means that its projected mean, and sometimes even its variance, will be changing, which makes the model more difficult to adjust. In general, a series that does not exhibit this behaviour will always be preferable as a model. That is, the use of stationary data series is highly preferred to ensure that the model may be better adjusted.

Table 6 LSTM ANN model parameter set

Parameter	# value
Sample	Dataset obj. costs
Time step	1
Features	1
Batch size	1
No. of epochs	1,500
No. of neurons	54
No. of hidden layers	4
Activation function	Hyperbolic tangent
Loss function	Mean square error (MSE)
Optimisation algorithm	ADAM
Learning/testing ratio	60/40%

The test stationary state was performed using the Dickey-Fuller method augmented through a Python script and the following hypotheses:

- *Null hypothesis (H0)*: If accepted, it implies that the time series has a unit root, which means that it is non-stationary. In other words, it has a time-dependent structure.
- *Alternative hypothesis (H1)*: The null hypothesis is rejected, which suggests that the time series does not have a unit root and, therefore, it is stationary. It does not have a structure which is dependent on time.

The test results may be interpreted through the p-value. A p-value below the threshold (such as 5% or 1%) suggests that the null (stationary) hypothesis is rejected; otherwise, a p-value above the threshold suggests that the null (non-stationary) hypothesis is accepted.

- p-value > 0.05: Accepts the null hypothesis (H0); the data have a unit root and are not stationary.
- p-value <= 0.05: Rejects the null hypothesis (H0); the data do not have a unit root and are stationary.

The result of the script is presented in Figure 9, considering the test values provided in the previous code. One way to interpret the results is to determine whether the ADF value, when compared with the critical values, defines the probability of rejecting the H0 hypothesis. In this case, a value of 0.0 (or positive) is greater than all critical values. Therefore, it is less possible to reject the H0 hypothesis, and the data series is then accepted as having a single root and being non-stationary.

Figure 9 Augmented Dickey-Fuller (ADF) test results

```

ADF Test Statistic  0.000000
P-Value: 0.958532
Critical Values:
1%: -7.355
5%: -4.474
10%: -3.127

```

For the particular case of time series, such as *supervised learning* problems, the ‘lag’ statistical method, which consists of using the same series as input and output data and trying to ‘offset’ the input data to create an appropriate matrix for learning, may be used. Figure 8 shows the original *dataset* and the prepared dataset with scaling operations, conversion to stationary series, and disposition for supervised learning.

For this scenario, we propose using the recurrent LSTM neural network since, for our purposes, this type of network exhibits certain advantages over other methods, such as their neurons being able to ‘remember’ throughout different epochs, and turning this ability into a tool for adding useful information to the transfer function of each neuron in the following epoch. In this case, unlike the common regression prediction model, time series add complexity due to their dependence on the sequentiality between the input variables. Here is where an LSTM network, commonly used in deep learning, may be successfully trained for long and complex architectures.

For the successful implementation of this model, the following elements must be considered:

- An LSTM network expects an input defined by three parameters:
 - 1 samples
 - 2 number of time steps
 - 3 features.

The first is the input data; the second is the measurement of individual steps for a single variable throughout an observation time; and the third are the individual features measured during the observation time.

- Because the data are sequential, a single time step and feature may be used as model parameters; this implies that a single step in the time series may be taken as an individual sample.
- To take advantage of the main LSTM feature, that is, its ability to remember states between epochs, it is necessary to also define the number of epochs that must be saved before forgetting.

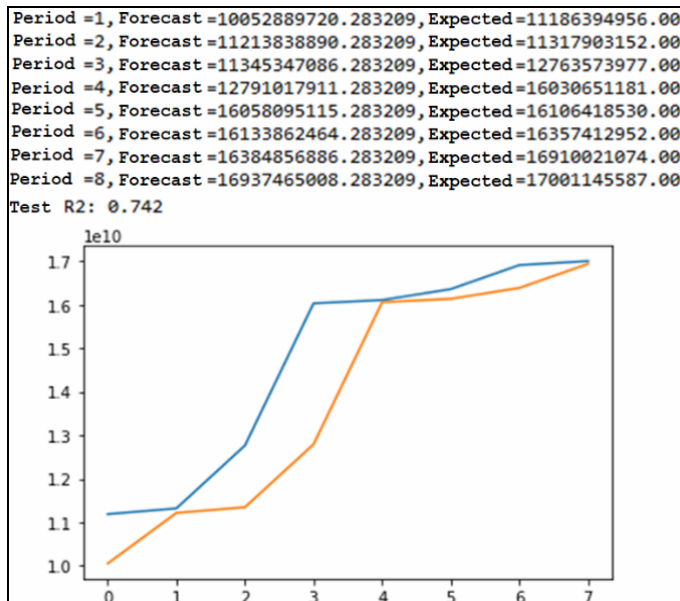
Finally, the batch size, the total number of times, and the number of neurons in the hidden layer are defined. Table 6 summarises model configuration.

The model was assessed through the coefficient of determination (R^2 value), which demonstrates how close the predicted values are to the actual values of the series, and whose general definition is as follows:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \quad (4)$$

The R^2 value is expressed as a number between 0 and 1, where 0 means that the model does not fit at all and 1 means that it fits perfectly.

Figure 10 Summary results for the LSTM network model (see online version for colours)



The *script* execution results can be seen in Figure 10. These results show a goodness of fit or R^2 value of 0.742 (74.2%), in addition to plotting the original values of the series against the predicted values for each one.

4.3 Cost prediction: implement multiple linear regression using an MLP network

The proposed scenario requires an implementation for the solution of a supervised learning problem, where the business objective is achieved through regression by relating the independent variables of the number of employees and contract value to the number of tickets that were generated. Consequently, the data must be prepared to create two *datasets*: one to feed the MLP network and another to validate the outputs of each interaction from the *backpropagation* process, as we show in Table 7. All data records on these datasets are money in COP and have units around billions.

Table 7 List of *datasets* after the data preparation

#	Dataset name	Dimension
1	X	20×2
2	y	20×1
3	train_X	13×2
4	train_y	7×1

Table 8 List of MLP parameters

Parameter	# value
Batch size	1
No. of epochs	200
No. of neurons	3
No. of hidden layers	1
Activation function	ReLu
Loss function	Mean square error (MSE)
Optimisation algorithm	ADAM
# data (learning/tests)	13/7

In addition, the model was configured using an MLP network with consumption parameters for the described data and with an internal configuration similar to the one presented in Figure 4. Table 8 shows the described parameter configuration, which is consistent with this type of neural network.

Finally, a Python script is developed using the Skit Learn Library, which includes models like the MLPRegressor for the deployment of the MLP network and its subsequent configuration based on scenario needs. Below, we provide a part of the code that simply illustrates the configuration, training, and prediction phases for the illustrated data.

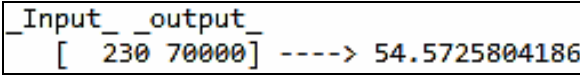
```

#Creation of the MLP neural network
regMLP=MLPRegressor(hidden_layer_sizes=(1,3), max_iter=150)
#Training the network
regMLP.fit(X,y)
#Next predicted value
entrada=np.array ((230, 70,000)), dtype=int)
#Prediction
predicción=regMLP.predict(entrada)
print(predicción)

```

The execution result yields 54.57 tickets for 230 employees and a contract value of 70,000. Figure 11 displays the Python console with these results.

Figure 11 Spyder console prediction results



```

_ Input_ _output_
[ 230 70000] ----> 54.5725804186

```

5 Scenario evaluation

5.1 Analysis of the distribution scenario

For the distribution scenario analysis, our implementation used actual data from a cost structure at an actual university, which gave us permission to extract a part of their cost structure for this research.

5.1.1 Operation in an imperative language (PL/SQL)

PL/SQL is an imperative language (it focuses on the ‘how’), which implies an implementation that literally uses some cycles for the external structure (level cycles) and other cycles for the internal structure (collection cycles). Within these two cycles, each collector queries the corresponding criteria and features to make the corresponding cost distribution with the linked collectors. For example, for the distribution of 1,000, 1,020, 2,400, and 5,300 dependencies within a given period of time, a total of 38,145 records are obtained, which are then consolidated into 855 records, thus suggesting that the PL/SQL distribution handles and processes more information to achieve its goal.

On the other hand, Python, which is a declarative language (focusing on the ‘what’), does not require a consolidation process because the calculation of the matrix operation consolidates without saving any trace of money discriminated in the links, thus guaranteeing the matrix operation. For these purposes, one must only be careful to place zeros in the matrix positions where there is no cost transfer from a collecting neuron from a given original level to another collecting neuron in a target level. At the end, the PL/SQL cost distribution process only required a total of 557 lines of code to perform the cost distribution.

5.1.2 Operation in a declarative language (Python)

As a declarative language, Python focuses on the ‘what’, which implies that, instead of cycles, as in the case of PL/SQL, a matrix algebra-based implementation was performed, as already described in 0. When performing the scenario implementation using Python, the results from the *script* evidence a difference of billions over the values from the PL/SQL-based implementation. Based on this, we decided to perform two implementations instead of one to verify if there were any differences in terms of ingestion. These two implementations and their features are described below:

- 1 Python-based implementation and ingestion from Oracle: A direct connection was made to the Oracle database where the ABC cost model data is stored using the `cx_Oracle` library. Then, the data preparation stage converted the data into matrix form within Python using the Pandas and Numpy libraries. Finally, the cost distribution was performed, successfully obtaining the five cost objects required.
- 2 Implementation exporting from Oracle and ingestion with Python: The movements and links tables were exported in CSV format. Then, the data were prepared in Google Spreadsheet to convert them into matrix form. Finally, the data were ingested to Python using the Pandas library through its `read_csv()` function. Finally, the cost distribution was performed, successfully obtaining the five cost objects required.

An advantage that became evident when performing both implementations using the proposed scenario in Python is that error handling is simplified once the distribution to matrix algebra issue is reduced. In this case, multiplication, since the support of null links replaced by zeros, order correspondence in the matrix dimensions and the order of operands must all basically be guaranteed. We must also emphasise the power offered by the Pandas and Numpy libraries for data preparation, since they provide practically a different function or procedure for each desired case cleaning, organisation, form, etc. At the end, the distribution process in Python provides a total of 55 lines of code for the implementation with ingestion from Oracle and 36 lines of code for ingestion from flat files in CSV format.

5.1.3 Procedure execution analysis

Three implementations were simultaneously executed in the same computer for the cost distribution scenario. Then, their execution times were measured, considering two situations:

- 1 after restarting the Oracle service and the Python kernel
- 2 without restarting the Oracle service and the Python kernel.

Table 9 provides the runtime behaviour for the three implementations, taking into account the situations described. In addition, separate times are recorded for the data ingestion and the processing time of the distribution. Times on these datasets are from two steps, data ingestion from SQL and Oracle engines at first and then from datasets allocated on Google Spreadsheet and then processing, when the cost (money) was distributed accordingly to ABC.

Table 9 Time comparison between distribution implementations

<i>Implementation</i>	<i>PL/SQL</i>	<i>Python + Oracle</i>	<i>Python + TensorFlow</i>
Time in seconds (with kernel/service restart)	Distribution: 3.907	Ingestion: 2.405	Ingestion: 0.024
	Consolidation: 0.438	Processing: 0.379	Processing: 0.126
Time in seconds (no kernel/service restart)	Distribution: 1.328	Ingestion: 0.071	Ingestion: 0.019
	Consolidation: 0.125	Processing: 0.001	Processing: 0.055

The results from Table 9 suggest an advantage for Python while using TensorFlow for both the ingestion and the distribution process. Conversely, PL/SQL exhibits more modest times in general with or without its service being restarted; however, it is clear that the manipulation of distribution data takes more time than for its rival implementations.

In turn, connecting directly to Oracle will still be an alternative despite requiring more time after restarting the Python kernel, since once the kernel has cached the necessary elements, the difference between times with the TensorFlow-based implementation becomes similar. It should also be considered that, in practical terms, the time spent preparing data outside of Python was much greater than the time dedicated to the Oracle implementation, plus adding an accuracy matter that will be discussed in the following chapter.

Table 10 Implemented cost structure level code

<i>Level code</i>	<i>Level name</i>
10	Resources – chart of accounts
20	Dependencies
30	Activities
40	Cost objects

Next, in Table 10 we provide the results obtained by the distribution implementations whose features were listed in the previous section. Labelling codes are an accounting standard in Colombia and we use it to know where the money comes from, where it is and where it needs to be in the way the cost distribution makes sense. However, it should be noted that these comparisons are based not only on cost object results (last level) but also on their dependency level, which evidences a resulting behaviour that enriches this analysis.

This analysis excludes the first level since it is not affected by the cost distribution as it is part of the data input consumed by the distribution. In the same way, the level of activities is excluded because it would yield too many results, and these results would be very similar to each other and would not denote the accuracy behaviour required for these three implementations.

As we shown in Table 11, the level of accuracy of the three implementations is almost indiscernible when comparing their absolute deviations. On the other hand, as per Table 12, the level of accuracy decreases when increasing absolute deviations from hundreds of thousands in Python + Oracle to millions in Python + TF (TensorFlow). To find the reason behind this behaviour, the data were reviewed from their source in the Oracle tables and printed after being consumed in the Python + Oracle implementation,

and the files were reviewed in the Google Spreadsheet (where the data preparation was completed) before and after being exported to CSV format.

Table 11 Comparison of distribution actual at level 20 (billions COP)

<i>Dependence</i>	<i>Accrual X implementation</i>			<i>Absolute deviations</i>		
	<i>PL/SQL</i>	<i>Python + Oracle</i>	<i>Python + TF</i>	<i>PL/SQL</i>	<i>Python + Oracle</i>	<i>Python + TF</i>
1,000	3,639,082,020	3,639,082,020	3,639,082,020	0	0	0.18
1,020	384,107,576	384,107,576	384,107,576	0	0	0.06
2,400	3,151,707,849	3,151,707,849	3,151,707,850	0	0	0.85
5,300	1,968,836,929	1,968,836,929	1,968,836,930	0	0	0.53

Table 12 Comparison of distribution accruals at level 40 (billions COP)

<i>Cost object</i>	<i>Result implementation X</i>			<i>Absolute deviations</i>		
	<i>PL/SQL</i>	<i>Python + Oracle</i>	<i>Python + TF</i>	<i>PL/SQL</i>	<i>Python + Oracle</i>	<i>Python + TF</i>
1	1,829,955,277	1,829,823,583	1,805,124,100	0	131,694	24,831,177
2	1,843,178,998	1,843,076,888	1,827,238,780	0	102,109	15,940,218
3	1,870,132,432	1,870,035,020	1,830,307,200	0	97,412	39,825,232
4	1,655,262,545	1,655,152,527	1,612,636,800	0	110,019	42,625,745
5	1,945,205,122	1,945,098,039	1,912,159,870	0	107,083	33,045,252

Finally, it can be deduced that this accuracy behaviour is manifested when passing from one level to another, with two possible causes:

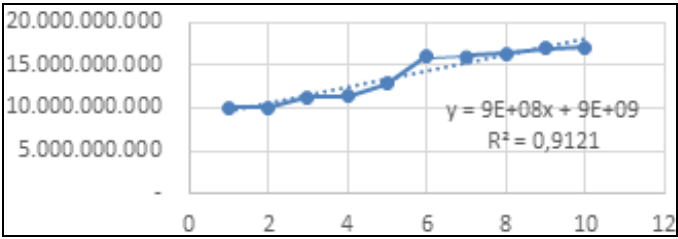
- 1 The rounding when performing operations is different in each implementation and also the number of operations is low in early levels; therefore, each implementation has a different level of accuracy.
- 2 The PL/SQL-based implementation increases the possibility of inaccuracy because the traceability implied in using temporary tables increases the number of operations throughout the level distribution, as seen in the number of records involved before and after consolidating.

Due to the above, it is very possible that the implementation that adds more accuracy is the Python + Oracle, since it eliminates the need for the additional processing required by the PL/SQL-based implementation and as well as the dependence on data preparation outside Python, as with the Python + TensorFlow implementation.

5.2 Analysis of the time series scenario

For the analysis of this scenario, the different time-series forecasting methods were compared with each other, based on the same data described in the implementation chapter of this scenario. Figure 12 displays these ten time series datasets, with their equation and trend line.

Figure 12 Cost vs. semester chart, considering undergraduates (see online version for colours)



5.2.1 Comparison of time series results

Our comparative chart displays the behaviour of several regression methods for the LSTM network, in an attempt to provide conclusions based on the results. Next, Table 13 shows the aforementioned methods and some of the data required for their execution.

$$F_t = F_{t-1} + \alpha(A_{t-1} - F_{t-1}) \quad (5)$$

Once each method is applied to the source data series, their behaviour is compared to the LSTM network, as shown in Table 14.

It is important to mention that the blank spaces evidenced in Table 14 are generated by the amount of data required by the method to start making forecasts. For the LSTM network, they are generated by converting data to a supervised learning problem. This implies that, as exposed in the scenario implementation, the lag required to produce an apt *dataset* disables two samples.

Table 13 Methods and data used for comparative analysis

Method/data	Description
MMS 2	Simple moving average 2 periods
MMS 3	Simple moving average 3 periods
SE	Exponential smoothing with $\alpha = 0.5$ (5)
MMP	Weighted moving average with 50% for t , 30% for $t - 1$, and 20% for $t - 2$
REG	Regression
Intercept	8,612,762,440
Pending	937,624,448

Finally, Table 15 presents the accrued values, where evidently both the regression accruals and the LSTM network obtain close results and are considered the best results. However, the LSTM network is a more robust implementation. Further, when assessing the values from period to period, it may be concluded that, with a greater number of data, the network may achieve greater accuracy in learning and therefore in forecasting.

Table 14 Forecast comparison table for each method (billions COP)

<i>Data</i>		<i>Analysis method</i>				
<i>Semester</i>	<i>Value</i>	<i>MMS2</i>	<i>MMS3</i>	<i>SE</i>	<i>MMP</i>	<i>LSTM</i>
1 13-06	9,998,001,852			9,998,001,852		9,550,386,888
2 13-12	10,025,445,786			9,998,001,852		10,488,011,336
3 14-06	11,186,394,956	10,011,723,819		10,011,723,819		11,425,635,784
4 14-12	11,317,903,152	10,605,920,371	10,403,280,865	10,599,059,388	10,600,431,584	12,363,260,232
5 15-06	12,763,573,977	11,252,149,054	10,843,247,965	10,958,481,270	11,019,959,220	13,300,884,681
6 15-12	16,030,651,181	12,040,738,565	11,755,957,362	11,861,027,623	12,014,436,925	14,238,509,129
7 16-06	16,106,418,530	14,397,112,579	13,370,709,437	13,945,839,402	14,107,978,414	15,176,133,577
8 16-12	16,357,412,952	16,068,534,856	14,966,881,229	15,026,128,966	15,415,119,415	16,113,758,025
9 17-06	16,910,021,074	16,231,915,741	16,164,827,554	15,691,770,959	16,216,762,271	17,051,382,473
10 17-12	17,001,145,587	16,633,717,013	16,457,950,852	16,300,896,017	16,583,518,129	17,989,006,921
11 18-06		16,955,583,331	16,756,193,204	16,651,020,802	16,845,061,706	18,926,631,370
						16,937,465,008

Table 15 Absolute deviations per period and accrued for each method (billions COP)

Semester number	Absolute deviations					
	MMS2	MMS3	SE	MMP	REG	LSTM
1	--	--	--	--	--	--
2	--	--	--	--	--	--
3	--	--	--	--	--	--
4	711,982,781	914,622,287	718,843,765	717,471,568	1,045,357,080	104,064,262
5	1,511,424,923	1,920,326,012	1,805,092,707	1,743,614,757	537,310,704	1,418,226,891
6	3,989,912,617	4,274,693,819	4,169,623,558	4,016,214,256	1,792,142,052	3,239,633,270
7	1,709,305,951	2,735,709,093	2,160,579,128	1,998,440,116	930,284,953	48,323,415
8	288,878,097	1,390,531,723	1,331,283,986	942,293,537	243,654,927	223,550,488
9	678,105,333	745,193,520	1,218,250,115	693,258,803	141,361,399	525,164,188
10	367,428,574	543,194,735	700,249,570	417,627,458	987,861,334	63,680,579
11	9,257,038,275	12,524,271,190	12,103,922,829	10,528,920,495	5,677,972,450	5,622,643,093

5.3 Analysis of scenario 3

For the analysis of scenario 3, a comparison was made between the results from a multiple linear regression and the results from the regression using an MLP network. Figure 11 denotes the results of the MLP regression in the Spyder console, indicating that for the 230 employees and 70,000 contract value, the prediction yields 54.5 tickets. However, the prediction improves after feeding the network with the arrangement of all the available X values. Figure 13 shows this result considering that only the last record will be used for the comparison.

Figure 13 Prediction for each value pair from arrangement X

Input	output
[51 25750]	----> 16.4318859512
[68 25000]	----> 15.7927976233
[67 40000]	----> 25.8213961154
[124 35000]	----> 22.0183150337
[124 25000]	----> 15.3379969988
[134 50000]	----> 31.9575776889
[157 48000]	----> 30.4347209683
[190 32000]	----> 19.4782046015
[205 70000]	----> 44.7415915385
[230 75000]	----> 47.8787145629
[265 50000]	----> 30.8936690851
[296 75000]	----> 47.3426995411
[336 50000]	----> 30.3170468648
[359 75000]	----> 46.8310488385
[403 81000]	----> 50.4818963117
[418 60000]	----> 36.3314068424
[437 53000]	----> 31.5008768632
[451 90000]	----> 56.1043534364
[465 70000]	----> 42.6300172103
[491 100000]	----> 62.4598138824
[230 70000]	----> 44.5385555454

5.3.1 Comparison of regression results

Table 16 shows the results of both regressions after the implementation, evidencing consistency after providing values that are quite close to each other.

Table 16 Comparison of regression predictions

Entry	Multiple linear regression	Regression with MLP
230, 70,000	45.9	44.5

6 Conclusions and future work

The main conclusion of this article refers to the possibilities represented, for example, by using techniques and methods that may enrich BAs and their current application. We worked with three scenarios that responded to our business objectives within the analytics

environment by using the different tools available. Conclusions pertaining to each scenario are presented below:

6.1 Cost distribution scenario using the MLP Forward phase

A similarity between the cost model and a neural network structure provides an opportunity to optimise critical business processes and, by methodologically demanding a culture of business and data understanding, open the door to generate a foundation for an analytical environment where other models may be developed to cover more and better business objectives.

For this scenario, the implementation differences are very promising and, while the stored procedures require additional intrinsic processing, the Python scripts are cleaner and more versatile in terms of data management and the distribution process itself.

The execution times are also promising, although the TensorFlow multiprocessing could be much more differentiating in an implementation that requires kernel selection from an available GPU multiprocessor. This does not mean that the advantage is not evident with or without restarting the kernel, but it does raise a test scenario for future study works.

Regarding the results, it is clear that the data themselves provoke or require special handling of the approximations, especially since the cost objects are measured in billions. In the PL/SQL-based implementation, after many data inspections in different parts of the levels, it became evident that accuracy was being affected by the number of transactions that the temporary tables drag when the distribution approaches the lower levels. The same cannot be said for Python-based implementations because matrix algebra, in this case multiplication, does not differentiate the portions of money throughout the distribution, but instead it simply consolidates level by level and, thus, the trace is lost.

6.2 Time series scenario using an ANN

The main conclusion for this scenario is expressed from the point of view of the network's capability of providing decent learning with a small amount of training data, as shown in the comparison of absolute deviations. However, another point of view would be a very robust implementation associated with a relatively simple data preparation offering the same simple regression features. Still, in spite of the less optimistic point of view, the small amount of data available for training is evident and therein lies the opportunity for better accuracy with a largely trained ANN.

An additional observation implies that a more adjusted implementation to business requires exploring a multivariable forecast to include all cost object values.

Finally, current business demands, exemplified in a continuous use of information as a competitive advantage, have created reasonable conditions of knowledge, tools, and methodologies, so that certain disciplines, such as machine learning, may contribute in finding scenarios where techniques, such as ANNs, may represent an analytical opportunity.

The use of *big data* tools in business scenarios, from data processing platforms used exclusively for analytics to TensorFlow mathematical libraries, can provide companies access to multiprocessing data by leveraging different and relatively inexpensive infrastructure. TensorFlow, in particular, allows us to think beyond physical CPU

arrangements and start thinking about GPU arrangements that are more common today, such as cryptocurrency mining.

The integration strategies between ANNs and BAs have demonstrated that learning on a MLP network optimises processing costs in cost distribution and help to forecast in time series. As a potential contribution, they can also be used in other similar structures in different areas such as internet of things, for instance with algorithms of exploration of a tangle in an Iota network, improving the availability of libraries optimised, or adding processing capacity in the deep learning field.

In terms of future work, a second method for the implementation of the *backpropagation* algorithm may be formulated; a complete and possibly convenient way in terms of costs. This method would require a more sophisticated neuron structure that could store more information, transfer cost information ‘between epochs’ and be used not only at the cost level but also at cost object level.

We also believe that a classification exercise may improve the selection of cost-inducing criteria, which would allow us to decide which criteria to use and at what level, based on the different costs between them.

References

- Azevedo, A. and Santos, M.F. (2008) ‘KDD, SEMMA and CRISP-DM: a parallel overview’, in *IADIS 2008: Proc. Int’l Assoc. Development of the Information Soc., European Conf. Data Mining*, Amsterdam, The Netherlands, pp.182–185.
- Bone, R., Assaad, M. and Cardot, H. (2008) ‘A new boosting algorithm for improved time-series forecasting with recurrent neural networks’, *Information Fusion*, Vol. 9, No. 1, pp.45–55.
- Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C. and Wirth, R. (2008) *CRISP-DM 1.0 – Step-by-Step Data Mining Guide*, Technical Report CRISPMWP-1104, SPSS Inc., USA [online] <https://www.the-modeling-agency.com/crisp-dm.pdf> (accessed 28 February 2020).
- Cokins, G. (2001) *Activity-Based Cost Management: An Executive’s Guide*, John Wiley & Sons, USA.
- Constangioara, A., Bodog, S., Laszlo, F. and Petrica, D. (2009) ‘Forecasting in business’, *Journal of Electrical and Electronics Engineering*, Vol. 2, No. 211, pp.211–214.
- Dong, X., Qian, L. and Huang, L. (2017) ‘Short-term load forecasting in smart grid: a combined CNN and K-means clustering approach’, in *BigComp 2017: Proceedings of the IEEE International Conference on Big Data and Smart Computing*, Jeju, Korea, pp.119–125.
- Fayyad, U.M. (1996) ‘Data mining and knowledge discovery: making sense out of data’, *IEEE Expert: Intelligent Systems and Their Applications*, Vol. 11, No. 5, pp.20–25.
- Franses, P.H. (1998) *Time Series Models for Business and Economic Forecasting*, Cambridge University Press, London.
- Gartner, Inc. (2020) *Gartner IT Glossary*, Technical Report, Gartner, Inc., Stamford [online] <https://www.gartner.com/it-glossary/business-analytics> (accessed 28 February 2020).
- Hamilton, J.D. (1994) *Time Series Analysis*, 2nd ed., Princeton University Press, New Jersey.
- James, M., Tom, M., Groeneveld, P. and Kibardin, V. (2020) ‘Physical mapping of neural networks on a wafer-scale deep learning accelerator’, in *ISPD 2020: Proceedings of the International Symposium on Physical Design*, Taipei, Taiwan, pp.145–149.
- Jara, G.N., Castañeda, J. and Gómez, L.F. (2003) ‘Sistema de costeo basado en actividades como herramienta del presupuesto inteligente para el distrito capital, Santa Fé de Bogotá – Colombia-Secretaria de Hacienda Distrital Bogotá DC’, *Contaduría Universidad de Antioquia*, Vol. 1, No. 43, pp.179–204.

- SEMMA SAS Institute (2020) [online] http://www.sas.com/en_us/software/analytics/enterprise-miner.html (accessed 20 February 2020).
- Shmueli, G., Patel, N. and Bruce, P. (2016) *Data Mining for Business Analytics*, John Wiley & Sons, New Jersey.
- Siami-Namini, S., Tavakoli, N. and Namin, A.S. (2019) ‘The performance of LSTM and BiLSTM’, in *Big Data 2019: Proceedings IEEE International Conference on Big Data, Forecasting Time Series*, Los Angeles, California, USA, pp.3285–3292.
- Van Veen, F. (2020) *Neural Network Zoo Prequel: Cells and Layers* [online] <https://www.asimovinstitute.org/author/fjodorvanveen/> (accessed 28 February 2020).