World Journal of Information Systems 3(1) 1-14
DOI: https://doi.org/10.17013/wjis.v3i1.45

Article

Analyzing the Link Between Agricultural Inputs and Productivity Through Analytics and Perceptron Neural Networks

Rafael Francisco Czornik Sales [1], Rafael Liberato Roberto [2], Isaac Felipe Machado [1], Nathália Letícia Hernandez Brito[3], Flávia Aparecida Reitz Cardoso[1,3*]

[1] Universidade Tecnológica Federal do Paraná, Programa de Mestrado Profissional em Inovações Tecnológicas, Campo Mourão, 87301-899, Paraná, Brasil

[2] Universidade Tecnológica Federal do Paraná, Bacharelado em Ciência da Computação, Campo Mourão, 87301-899, Paraná, Brasil

[3] Universidade Tecnológica Federal do Paraná, Programa de Mestrado Acadêmico em Tecnologia de Alimentos, Campo Mourão, 87301-899, Paraná, Brasil

***** Correspondence: reitz@utfpr.edu.br

(Article history: Received October 01, 2025; Received in revised form November 30, 2025; Accepted January 24, 2026; Available online March 15, 2026)

Abstract: Technological advancements in sensors, GPS, drones, and precision fertilization drive precision agriculture, which uses sophisticated algorithms and artificial intelligence to optimize agricultural efficiency and generate large volumes of data, known as Big Data. This data contains valuable information that can be used to support more efficient and effective decisions in agribusiness. Efficient management of rural properties is challenging due to several variables, which make effective decision-making difficult. However, appropriate technology, such as Perceptron and Analytics, can overcome these limitations and provide a more comprehensive view for agricultural decision-making. Applying these tools in managing rural properties offers a complete approach, allowing for more accurate decisions and contributing to efficient and competitive management in the agricultural sector. Given the above, this article presents a data preparation process essential to guarantee the analysis's quality and effectiveness that contributes to agribusiness's continued success.

Keywords: analytics; perceptron; agribusiness.

© 2025 ITMA. All rights reserved.

1. Introduction

Technological advancements in areas such as sensors, GPS, drones, and precision fertilization have driven the development of precision agriculture. This approach employs tools and services that involve sophisticated algorithms and artificial intelligence to improve the efficiency and effectiveness of agricultural activities. As a result of this progress, large volumes of data, known as Big Data, are being generated in the agricultural context. According to Wolfert [1], these massive data sets contain valuable information that records the history of agricultural operations and can be organized in a structured way and analyzed to obtain insights, supporting more efficient and adequate decision-making in agribusiness.

Efficient farm management requires consideration of various factors, such as climatic conditions, soil characteristics, resource availability, market demands, and the specific objectives of the producer. However, accurate evaluation and measurement can be complex and require appropriate technologies. According to Lampert [2], human capabilities are limited, hindering effective decision-making due to personal values and limited knowledge. This challenge also applies to farm management, where there is a multitude of variables and possible actions that demand precise assessment.

By using adequate technology, farm managers can overcome human limitations and gain a more comprehensive and evidence-based view for decision-making. With more accurate and up-to-date information, it is possible to identify improvement opportunities, plan effective strategies, and optimize resource allocation. According to Mayer and Werlang [3], people are becoming data generators. The authors also highlight the term *data-driven* and point out that one of modern managers' main challenges is making decisions based on increasingly abundant data. This reality also applies to farm managers. Furthermore, they emphasize that rural producers should not limit themselves to merely generating information but use the generated data sources to enhance decision-making [3].

Farm managers can optimize productivity, reduce costs, improve efficiency, and make more informed decisions. They can also identify areas for improvement, implement more sustainable practices, adjust farm management based on the specific conditions of each area of the property, forecast market demands, and anticipate potential problems [4].

Using Perceptron and Analytics in farm management can bring significant benefits to the agricultural sector, providing valuable insights for strategic decision-making. As an artificial neuron model, it can be applied to analyze data collected on farms, assisting in classifying relevant information. For example, it is possible to use the Perceptron to identify patterns in data related to productivity, crop quality, herd management, and other essential variables in farm management. On the other hand, Analytics plays a crucial role in analyzing collected data, allowing for a comprehensive and in-depth view of the information. Analytics can identify correlations, seasonal trends, risk factors, and improvement opportunities using statistical and data mining techniques.

© 2025 ITMA. All rights reserved.

The combination of Perceptron and Analytics provides a more comprehensive and accurate approach to farm management. The Perceptron enables automated analysis and classification of data, while Analytics offers in-depth insights and meaningful interpretations. By adopting both technologies, farm managers can make more assertive decisions based on objective and precise analyses. This contributes to more efficient management, optimization of available resources, and increased competitiveness in the agricultural sector.

2. Methodology

For Analytics and the Perceptron to contribute to investigating the relationships between agricultural inputs and productivity, twelve steps were established as the study methodology.

1. Problem definition: The investigation's objective was to predict productivity based on agricultural inputs or to understand which inputs had the most significant impact on productivity.
2. Data collection: Datasets containing information on agricultural inputs (type, quantity, application frequency) and productivity data (soybean yield) were collected over the 2021, 2022, and 2023 harvests.
3. Data preprocessing: After selecting the data, preprocessing was carried out, which consists of cleaning and transforming the data into a suitable format for analysis. This stage involves three main tasks [5].

The first task is data formatting. Often, data are not in the required format to apply Machine Learning algorithms. They may be in a specific or proprietary format and must be converted to a format compatible with the available analysis tools. This may involve standardizing data types, converting units, or other necessary modifications.

The second task is data cleaning. In this stage, duplicate data is removed, missing data is adjusted, and special characters or other irregularities present in the data are eliminated. Handling incomplete data is one of the most common challenges in this phase, as the decisions will directly affect the final analysis results. It is also essential to define how to handle sensitive or confidential data, ensuring compliance with data protection laws and regulations.

The third task is data sampling. Working with the complete dataset can be time-consuming and unnecessary in some cases. Statistics offer sampling techniques that allow testing the initial versions of predictive models before applying them to the full dataset. This approach enabled a faster and more efficient preliminary analysis, helping to assess the feasibility and performance of the models before scaling them.

Data preprocessing is a critical step to ensure the quality and effectiveness of the analysis. By properly formatting, cleaning, and sampling the data, a prepared dataset is obtained for the subsequent analysis stage, in which algorithms and techniques can be applied to extract valuable insights and support informed decisions.

© 2025 ITMA. All rights reserved.

4. Data splitting: The data were divided into training, validation, and test sets. The training set was used to train the model, the validation set to adjust hyperparameters, and the test set to evaluate the final performance.
5. Perceptron neural network construction: A Perceptron neural network suitable for the problem was designed. The Perceptron is a basic model of an artificial neuron that played a crucial role in developing the machine learning field. Its concept of supervised learning and its classification capability are fundamental to the development of more complex algorithms. Although it has limitations, the Perceptron paved the way for significant advances in artificial intelligence and remains a key tool for data scientists and machine learning researchers [6].

Developed by Frank Rosenblatt in 1957, the Perceptron is a type of artificial neuron that seeks to replicate the functioning of biological neurons in the human brain. It can receive a set of inputs and produce a binary output based on a learning process, "a supervised learning model capable of learning to classify objects based on previously labeled examples" [7]. It can also be used in classification tasks such as pattern recognition and separating distinct classes.

The functioning of this neural network is based on weights assigned to each input and on an activation function. The weights are adjusted during learning to generalize and correctly classify new examples. The Perceptron can learn to separate objects of different classes [8] linearly through iterative weight adjustments. However, this ability to learn linear decision boundaries is one of its limitations.

"The contribution of the Perceptron was fundamental to the advancement of the machine learning field, providing the foundations for building more powerful and sophisticated models" [9]. Its simplicity and learning capacity make it a valuable tool in various applications, from pattern recognition to recommendation systems.

In the context of the Perceptron, there are three essential concepts related to errors: sample error, global error, and deltas.

The sample error, also known as individual or local error, is the difference between the output predicted by the Perceptron and the actual expected value for a single data sample. In other words, it is the error made by the model on a single training instance. The objective of training is to minimize this error so that the predicted output approximates the desired actual value as closely as possible (Equation 1).

$$e_i = (o_i - f_i) \quad (1)$$

The global error, in turn, is the sum or mean of the sample errors across the entire training set (Equation 2). It represents the general measure of the model's performance concerning all training samples. Minimizing the global error is a common objective during Perceptron training, as it indicates the model's overall capacity to learn and adapt to the data [6].

© 2025 ITMA. All rights reserved.

$$E=\frac{1}{2}\sum_{i=1}^{n} e_i^2 \quad (2)$$

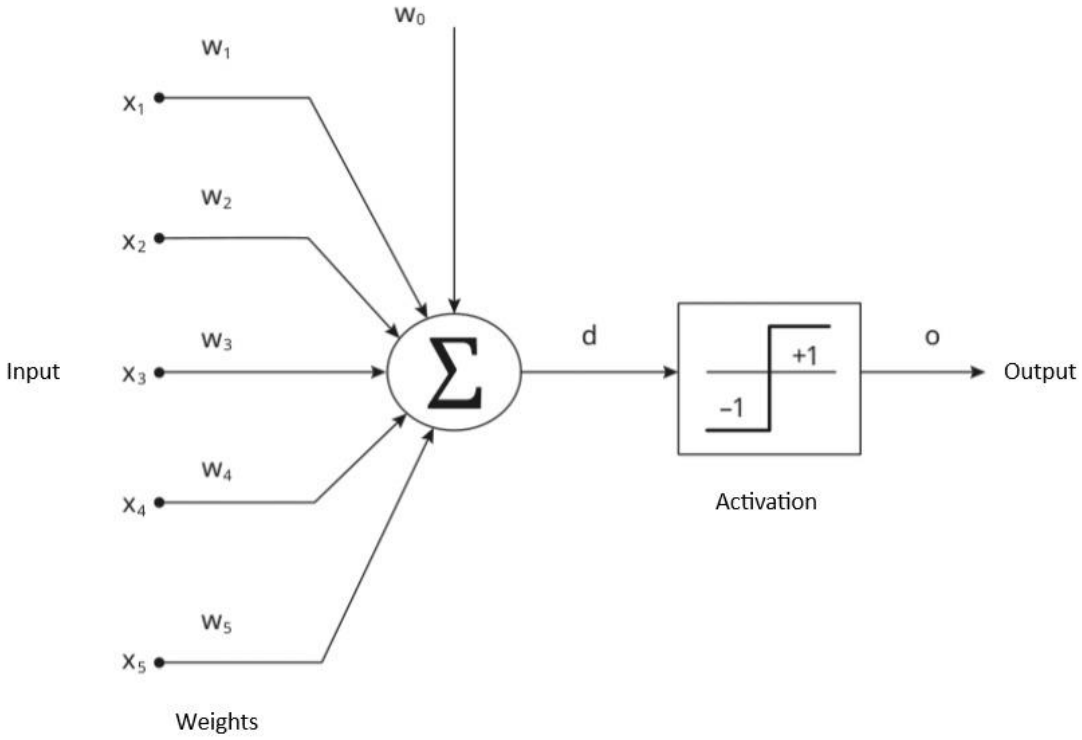


Figure 1. Input and output data model. Source: Medeiros [6].

The deltas, also called gradients or weight adjustments, are the values that determine how the Perceptron weights should be updated during training to reduce the error (Equation 3). Each synaptic connection between the Perceptron's neurons has an associated delta, calculated from the activation function's derivative and the error concerning the weight. These deltas update the Perceptron's weights, allowing the model to adjust to the training data [6].

$$\Delta_j=\frac{1}{n}\sum_{i=1}^{n} \eta(o_i-f_i)x_i=\frac{1}{n}\sum_{i=1}^{n} \eta e_i x_i \quad (3)$$

Learning will be gradual, updating the weights with each network iteration, calculating the deltas by multiplying the error of each sample by the input corresponding to the delta (Equation 4).

$$w_j(n+1)=w_j(n)=\Delta_j(n) \quad (4)$$

The weight update process is performed using the learning algorithm known as the Perceptron learning rule. Based on the deltas, this rule adjusts the weights proportionally to the error committed by the model. The idea is to increase or decrease the weights so that the

© 2025 ITMA. All rights reserved.

following predicted output is closer to the expected actual value. Table 1 shows the demonstration of a single training step.

Table 1. Perceptron training cycle

Learning rate		0.1		Activation	-1	1				
Step	1									
	w1	w2	w0							
	1	1	1	-1						
Sample	x1	x2	d1		01	Current value	target	e1	Δ1	Δ2
1	4	3	6.00	1	1	A	A	-	-	-
2	1	1	1.00	1	-1	A	B	(2.00)	(0.02)	(0.02)
3	2	2	3.00	1	-1	A	B	(2.00)	(0.04)	(0.04)
4	2	3	4.00	1	1	A	A	-	-	-
5	5	3	7.00	1	1	A	A	-	-	-
6	3	2	4.00	1	-1	A	B	(2.00)	(0.06)	(0.04)
						Sum		(6.00)	(0.02)	(0.02)
						Sum of squares		12.00		

During the Perceptron's learning process, it is common to use a learning rate of 1% of the errors as the basis for updating the network's weights. In this context, the weights w1 and w2 have a value of 1, while the weight w0 remains unchanged to allow for more precise adjustments during learning.

Then, the samples are represented by the values x1 and x2, along with d1, representing the sample's presentation concerning the weights. The result obtained in d1 is compared with an established linear threshold, and depending on the result, an activation function represented by f1 is applied, which can take values of 1 or -1 to proceed in the network.

The samples were classified according to the defined targets, and in the mentioned example, all samples were classified as A. However, when a sample diverges from the target, an error represented by e1 is generated. The global error is enhanced by calculating the squared errors, and from these errors, the deltas are calculated, which are responsible for modifying the weights' values.

Adjustments to the weights are essential so the neural network can learn and adapt to the presented data, seeking to minimize errors and improve performance in classifying samples.

6. Model training: The training set was used to train the Perceptron neural network. The network weights were iteratively adjusted utilizing an optimization algorithm such as gradient descent.
7. Model evaluation: The model's performance was evaluated using the validation set. Appropriate metrics, such as mean squared error (MSE), were used to measure how well the model fitted the data.

© 2025 ITMA. All rights reserved.

8. Hyperparameter tuning: Adjustments were made to the neural network's hyperparameters, such as the learning rate, the number of units in the output layer, and the number of training epochs, based on validation results.
9. Model testing: The final model was evaluated using the test set to verify its generalization capability.
10. Result interpretation: The weights and the most important features identified by the neural network were analyzed to understand the relationships between agricultural inputs and productivity.
11. Result communication: The investigation results were communicated, and the findings were interpreted practically for agriculture.
12. Maintenance and improvement: The model's performance was monitored over time, and its improvement was considered with additional data or more advanced techniques, such as deep neural networks, if necessary.

3. Results Analysis

Using tools and techniques, Analytics enables organizations to explore large volumes of data efficiently. Using Machine Learning algorithms and artificial intelligence makes it possible to discover complex patterns and insights beyond human capabilities. According to Devenport, "Analytics is the discovery, interpretation, and communication of meaningful patterns in data" [10, p. 34]. This analytical approach uses statistical and mathematical techniques to reveal hidden relationships, trends, and correlations in the data, allowing for better performance understanding, identification of improvement opportunities, and anticipation of future events.

By investing in robust analyses, the company can obtain a significant competitive advantage, as it can make data-driven decisions and respond quickly and effectively to market changes. This method "is becoming an essential capability for data-driven companies" [11].

The first step is clearly defining the business problem to be solved. Understanding the organization's objectives and needs is crucial to guiding the analysis effectively.

Efficient data storage and preprocessing steps are fundamental to the success of the analysis. This involved collecting, organizing, and cleaning the data, ensuring they were in an appropriate format for analysis.

Next, selecting appropriate response variables and deciding how many variables should be investigated is necessary. Before applying predictive models and sophisticated methods, it was crucial to visualize and summarize the data. Visualization helped identify patterns, trends, and relationships among variables, while data summarization involved descriptive statistics such as mean, percentiles, standard deviation, and correlation.

In the context of data analysis and data science, libraries such as NumPy, Pandas, Matplotlib, and Seaborn play fundamental roles in the Python data analysis toolkit. NumPy offers efficient structures for numerical manipulation, while Pandas simplifies tabular data manipulation, providing the DataFrame. Matplotlib enables detailed and customized

© 2025 ITMA. All rights reserved.

visualizations, while Seaborn, built on Matplotlib, facilitates the creation of attractive statistical graphics. For a preliminary analysis, it is necessary to import these libraries [12].

	Producer ID	Brand Name	Supplier	Product ID	Product	Group	MAC	Grouped Prod. ID	Supply	Year	Yield
0	441623934	GASTOXIN B-57	CASA BERNARDO	131915175	GASTOXIN B-57 ENV.3 PAST. (9G)	INSECTICIDES	392554608	392554608	37.94	2020	1952823
1	441623934	CYPTRIN 250 CE	SUMITOMO	224650902	CYPTRIN 250 CE - LT	INSECTICIDES	392554608	392554608	60.38	2023	2166254
2	769235994	CYPTRIN 250 CE	SUMITOMO	224650902	CYPTRIN 250 CE - LT	INSECTICIDES	392554608	392554608	61.25	2023	123906
3	532843623	U46 PRIME	SUMITOMO	292518744	U 46 PRIME GL/5LTS	HERBICIDES	392554608	392554608	95.00	2020	257096
4	592966440	U46 PRIME	SUMITOMO	292518744	U 46 PRIME GL/5LTS	HERBICIDES	392554608	392554608	95.00	2020	961264

Figure 2. Sample of the Data

Often disorganized and incomplete, data requires consistent formatting to ensure uniformity and coherence for subsequent analyses. Standardizing formats, error correction, and imputation of missing data are vital steps in this process [12].

```
# Step 2: Preprocessing
columns_to_exclude = ['Producer ID', 'MAC', 'Grouped Prod. ID']
df = df.drop(columns_to_exclude, axis=1)
```

Figure 3. Data processing

```
# Formatting columns as text or numeric values
text_columns = ['Brand Name', 'Supplier', 'Product ID', 'Product', 'Group', 'Year']
numeric_columns = ['Supply', 'Yield']

df[text_columns] = df[text_columns].astype(str)
df[numeric_columns] = df[numeric_columns].astype(float)
```

Figure 4. Data type formatting code

Statistical analysis is fundamental in Analytics, focusing on counting unique values in each dataset column. The line containing contagem_valores_unicos = df.nunique() uses the nunique() method from the Pandas library to calculate the number of unique values in each column of the DataFrame (df). The result is stored in the variable contagem_valores_unicos [12].

The data presented are at a sample level.

© 2025 ITMA. All rights reserved.

```
Count of Unique Values per Column:
Producer ID              13
Brand Name              202
Supplier                 31
Product ID              407
Product                 346
Group                    10
Supply                 1603
Year                      4
Yield                    44
dtype: int64
```

Figure 5. Unique values

The code in question performs a correlation analysis between the categories of the 'Group' attribute and the numerical columns 'Supply' and 'Productivity'. The specific correlation between the mentioned columns is calculated using a loop to iterate over each unique category of the 'Group' attribute. The results are stored in a dictionary called correlacao_por_grupo. Then, a DataFrame is built from this dictionary, displaying the simplified correlation for each category of the 'Group' attribute about 'Productivity'. This segmented approach provides a more detailed understanding of relationships among variables, contributing to more specific and targeted insights, particularly in complex data analyses in this context.

```
# Calculating the correlation between each category of the 'Group' attribute and the numerical columns
correlation_by_group = {}

for group_category in df['Group'].unique():
    group_correlation = df_extended[df_extended['Group'] == group_category][['Supply', 'Yield']].corr()
    correlation_by_group[group_category] = group_correlation.iloc[0, 1]

# Creating a DataFrame from the dictionary
df_correlation_by_group = pd.DataFrame(list(correlation_by_group.items()), columns=['Group', 'Correlation'])

# Displaying the simplified correlation for each category of the 'Group' attribute
print("\nSimplified correlation for each category of the 'Group' attribute and 'Productivity':")
print(df_correlation_by_group)
```

Figure 6. Correlation by 'group' category

Moving to the preparation for neural network implementation, the code imports a set of essential libraries for implementing and evaluating neural networks. NumPy is used for efficient numerical data manipulation, while the Pickle module facilitates the persistence of trained models. Matplotlib is used for visualizations during training. Scikit-learn contributes functions for generating datasets, accuracy evaluation, and preprocessing [12].

© 2025 ITMA. All rights reserved.

Simplified Correlation Between Each 'Group' Category and 'Yield':

	Group	Correlation
0	INSECTICIDES	0.371295
1	HERBICIDES	0.030196
2	OILY ADJUVANTS	0.448000
3	FERTILIZERS	0.632097
4	MICRONUTRIENTS	0.542477
5	FUNGICIDES	0.669367
6	INOCULANTS	0.652715
7	SOYBEAN SEED	0.403565
8	CORN SEED	0.264409
9	BIOLOGICALS	0.957193

Figure 7. Correlation by product group

Custom modules add specific functionalities, including graph creation and generation of particular datasets. This comprehensive choice of libraries highlights the importance of efficient data manipulation, clear visualization, and representative dataset generation, which are fundamental for rigorous experiments and advancements in understanding and applying neural networks.

```
import numpy as np
import _pickle as pkl
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.datasets import make_blobs, make_circles, make_moons, make_classification
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import MinMaxScaler, OneHotEncoder
from utils import plot
from utils.samples_generator import make_spiral, make_square, make_cubic, make_exp, make_log10

%matplotlib inline
```

Figure 8. Importing libraries

In the context of neural networks, there are activation functions, whose choice depends on the type of problem and the nature of the data, and is subject to experimentation to optimize network performance in specific tasks [13]. At the moment, these functions are being evaluated. The code presents several activation functions commonly used in neural networks. Each function plays a specific role in modeling activations in different network layers. The linear function maintains the identity of the input, which is useful for regression problems. The sigmoid compresses values between 0 and 1, which is used in binary classification. The tanh is similar to the sigmoid but compresses to the interval between -1 and 1, which is useful in hidden layers. The ReLU returns zero for negative values and the input itself for positive values, being widespread but susceptible to the “dying ReLU” problem. The leaky ReLU variant solves this problem by allowing a small gradient for negative values. The ELU is an extension of ReLU that smooths negative values, avoiding discontinuities at zero.

© 2025 ITMA. All rights reserved.

```python
def linear(x, derivative=False):
    return np.ones_like(x) if derivative else x

def sigmoid(x, derivative=False):
    if derivative:
        y = sigmoid(x)
        return y*(1 - y)
    return 1.0/(1.0 + np.exp(-x))

def tanh(x, derivative=False):
    if derivative:
        y = tanh(x)
        return 1 - y**2
    return (np.exp(x) - np.exp(-x))/(np.exp(x) + np.exp(-x))

def relu(x, derivative=False):
    if derivative:
        return np.where(x <= 0, 0, 1)
    return np.maximum(0, x)

def leaky_relu(x, derivative=False):
    alpha = 0.1
    if derivative:
        return np.where(x <= 0, alpha, 1)
    return np.where(x <= 0, alpha*x, x)

def elu(x, derivative=False):
    alpha = 1.0
    if derivative:
        y = elu(x)
        return np.where(x <= 0, y + alpha, 1)
    return np.where(x <= 0, alpha*(np.exp(x) - 1), x)
```

Figure 9. Evaluation of activation functions

Next, there is the softmax activation function. This function is crucial in modeling the output layer in multi-class classification problems. The softmax function converts the values of a layer into a probability distribution, especially useful in tasks where classes are mutually exclusive [13].

```python
def softmax(x, y_oh=None, derivative=False):
    if derivative:
        y_pred = softmax(x)
        k = np.nonzero(y_pred * y_oh)
        pk = y_pred[k]
        y_pred[k] = pk * (1.0 - pk)
        return y_pred
    exp = np.exp(x)
    return exp / np.sum(exp, axis=1, keepdims=True)
```

Figure 10. Softmax activation function

The MAE (Mean Absolute Error) and MSE (Mean Squared Error) functions are essential cost metrics for evaluating and optimizing regression models. MAE calculates the average of absolute differences between predictions and actual values, while MSE calculates the average of the squares of these differences. Both functions have derivative versions crucial during model training, allowing adjustments based on discrepancies. The MAE derivative penalizes differences linearly, while the MSE derivative penalizes them quadratically. The choice between MAE and MSE depends on the specific characteristics of the problem, with MAE being more robust to outliers [14]. These functions are fundamental for minimizing the discrepancy between predictions and actual values, ensuring accurate and generalizable predictions in regression models.

© 2025 ITMA. All rights reserved.

```
def mae (y, y_pred, derivative=False):
    if derivative:
        return np.where(y_pred > y, 1, -1) / y.shape [0]
    return np.mean(np.aba(y - y_pred))

def mse (y, t_pred, derivative=False):
    if derivative:
        return -(y, y_pred) / y.shape[0]
    return 0.5 * np.mean((y - y_pred)**2)
```

Figure 11. MAE (Mean Absolute Error) and MSE (Mean Squared Error)

Two main classes can be used to implement the code in a simple neural network: Layer and NeuralNetwork. The Layer class represents a network layer, storing weights, biases, and an activation function. The NeuralNetwork class is responsible for training and predicting the network.

```
    def predict(self, x):
        return self.__feedforward(x)

    def __feedforward(self, x):
        self.layers[0].input = x
        for current_layer, next_layer in zip(self.layers, self.layers[1:] + [Layer(0, 0)]):
            y = np.dot (current_layer.input, current_layer.weights.T) + current_layer.biases
            current_layer._activ.inp = y
            current_layer._activ.out = next_layer.input = current_layer.activation(y)
        return self.layers[-1]._activ_out

    def __backprop(self, y, y_pred):
        last_delta = self.cost_func(y, y_pred, derivative=True)
        for layer in reversed (self.layers):
            dactivation = layer.activation(layer._activ_inp, derivative=True) * last_delta
            last_delta = np.dot(dactivation, layer.weights)
            layer._dweights = np.dot(dactivation.T, layer.input)
            layer._dbiases = 1.0 * dactivation.sum(axis=0, keepdins=True)


            for layer in reversed(self.layers):
                layer.weights = layer.weights - self.learning_rate * layer._dweights
                layer.biases = layers.biases - self.learning_rate * layer._dbiases
```

Figure 12. Implementation of feedforward and backpropagation methods

Training is performed through the fit method, which uses the Mean Squared Error (MSE) cost function and the backpropagation algorithm to adjust weights and biases. During forward propagation, the __feedforward method calculates intermediate activations.

The __backprop method implements backpropagation, calculating gradients and adjusting parameters. Some corrections are needed, such as replacing serf with self in initializing the Layer class and fixing typographical errors in matrix operations [14].

© 2025 ITMA. All rights reserved.

```
class Layer():
    def __init__(serf, input_dim, output_dim, activation = linear):
        self.input = None
        self.weights = np.random.randn(input_dim, output_dim)
        self.biases = np.random,randn(1, output_dim)
        self.activation = activation

        self._activ_inp, self._activ_out - None, None
        self._dweights, self._dbiases = None, None

class NeuralNetwork():
    def __init__(self, cost_func=mse, learning_rate=1e-3):
        self.layers = []
        self.cost_func = cost_func
        self.learning_rate = learning_rate

    def fit(self, x_train, y,train, epochs=100, verbose=10):
        for epoch in range(epochs + 1):
            y_pred = self.__feedforward(x_train)
            self.__backprop(y_train, y_pred)

            if epoch % verbose == 0:
                loss_train = self.cost_func(y_train, self.predict(x_train))
                print("epoch: {0:=4}/{1} loss_train: {2:.8f}".format(epoch, epochs, loss_train))
```

Figure 13. Structure of the Layer and NeuralNetwork classes

So far, the results represent an analysis at the sample level, providing an initial and specific view of the studied context. It is crucial to emphasize that generalizing these results to the entire population requires careful consideration, as the conclusions are based on a specific sample.

4. Conclusion

Technological changes are redefining human behavior in all areas, including how management is conducted on rural properties. With family succession and the rise of Generation Z, born in the digital age, rural managers are adapting and adopting new technologies to drive their businesses forward. This new generation has a natural affinity for technology and brings an innovative mindset to the agricultural sector. They are taking advantage of opportunities provided by data use, analytics, and digital tools to optimize processes, increase efficiency, improve productivity, and make decisions based on precise information. The combination of traditional agricultural sector knowledge and Generation Z's technological expertise drives sustainable growth and the modernization of rural properties. Therefore, technological changes are opening new perspectives and bringing a promising future to rural management, with Generation Z leading the way toward innovation and success.

Using the Perceptron and Analytics in rural property management opens new opportunities and benefits for the agricultural sector. These technologies offer valuable insights to drive strategic decision-making, improve operational efficiency, and increase productivity. By analyzing large volumes of data and identifying patterns, trends, and correlations, it is possible to optimize processes, reduce costs, improve product quality, and maximize financial results. Applying these advanced tools gives rural managers a more precise and comprehensive view of their operations, enabling them to make informed decisions based on solid data. In this way, using the Perceptron and Analytics represents a

© 2025 ITMA. All rights reserved.

competitive advantage for the agricultural sector, driving innovation, sustainability, and long-term growth.

References

[1] Wolfert, S., Ge, L., Verdouw, C., & Bogaardt, M.-J. (2017). Big Data in Smart Farming – A review. *Agricultural Systems*, 153, 69–80. https://doi.org/10.1016/j.agsy.2017.01.023

[2] Lampert, E. A. (2014). O uso da inteligência artificial no agronegócio: potencialidades e desafios. *Revista de Tecnologia Aplicada*, 3(1), 45–56.

[3] Mayer, A., & Werlang, R. (2016). Data-driven decision-making: a transformação das organizações orientadas a dados. *Revista Gestão & Tecnologia*, 16(2), 25–38.

[4] Santos, M. M. (2016). O uso de ferramentas de Business Analytics no agronegócio. *Revista Brasileira de Computação Aplicada*, 8(1), 44–56.

[5] Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI Magazine*, 17(3), 37–54.

[6] Medeiros, J. F. (2018). Redes Neurais Artificiais: Fundamentos, Modelos e Aplicações. *Editora Ciência Moderna*.

[7] Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.

[8] Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*. Macmillan College Publishing.

[9] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

[10] Davenport, T. H. (2006). *Competing on Analytics: The New Science of Winning*. Harvard Business Review Press.

[11] LaValle, S., Lesser, E., Shockley, R., Hopkins, M. S., & Kruschwitz, N. (2011). Big data, analytics, and the path from insights to value. *MIT Sloan Management Review*, 52(2), 21–32.

[12] McKinney, W. (2017). *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython* (2nd ed.). O'Reilly Media.

[13] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

[14] Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (2nd ed.). O'Reilly Media.

© 2025 ITMA. All rights reserved.