

Proyect II report

Students:

* Agustín Kuskinen Gimeno (NRC: 8074)
* Eduardo Correa Marchant (NRC: 8074)
* Matthias Seiltgens Trouve (NRC: 8074)
* Yuan Huang Fan (NRC: 8074)

Teacher in charge:

* PhD. Pablo Schwarzenberg Riveros
* PhD. Billy Peralta Márquez

Career:

Informatic Civil Engineering

Content

[Introduction 2](#_Toc164964822)

[Description of the characteristics that constitute the input of the model. 3](#_Toc164964823)

[Description of the parameters used to train the model. 3](#_Toc164964824)

[Description of the architecture of the model used. 3](#_Toc164964825)

[Conclusions regarding work 3](#_Toc164964826)

[Limitations of the model and proposals about how to improve its performance in the future. 3](#_Toc164964827)

# Introduction

The challenge we face involves examining a vast database related to electricity consumption in different urban areas and distribution points during a specific period. This detailed information includes unique records, measurement schedules, quantities of energy consumed, data quality indicators, and the underlying electrical infrastructure.

To better grasp this landscape, images illustrating key aspects of the data have been provided, such as unique identifiers, timestamps, and energy consumption distribution. These visual representations are crucial for understanding the complexity of the electrical grid and the variability in consumption patterns.

It is suggested to enhance data visualization through restructuring to simplify interpretation. Breaking down the information into 10 columns would facilitate data comprehension and highlight relationships between different variables.

Data quality analysis will be an essential part of our research, along with the use of descriptive statistics and graphs to identify significant trends. Furthermore, we will propose a selection of variables for a machine learning model, as well as splitting the dataset into training and testing sets to evaluate the model's effectiveness.

Regarding deep learning techniques, we will explore which ones are most suitable for evaluating and selecting the best generated model, providing reasoned justifications for our choice.

Finally, we will conceptually describe the type of neural model that could be developed using the proposed inputs and outputs, offering a clear vision of how neural networks could effectively address this challenge.

## Description of the characteristics that constitute the input of the model.

Before describing the characteristics that constitute the model's input, it is absolutely essential to clarify that in our 'DATETIME' type data, the period 2018 – 2021 has been used for training, and 2022 for testing. The reason for this is that regardless of whether our dataset contains outliers or not, both the 2017 and 2023 data are incomplete, providing either unreliable information or information that is completely out of reach or context.

Summarizing our results, ARIMA was the winning model both with outliers and without outliers in our datasets. Nevertheless, we will give a special mention to the Recurrent Neural Network (RNN) using an ADAM optimizer, to explain other characteristics that, as a group, we believe can complement the information and analysis.

For both the original dataset with outliers and the dataset without outliers, ARIMA was considered the winning model. However, while ARIMA does not have an explicit analysis of its characteristics, it is fully considered based on the results obtained (analysis of results in Description of the parameters used to train the model (Justify choice)). It is important to mention that ARIMA is a model more oriented towards statistical modeling than a neural network model. At the same time, we will also give a special mention to using the Recurrent Neural Network (RNN) with the ADAM optimizer.

Regarding ARIMA, we could mention some of the data used (characteristics) in ARIMA for each substation.

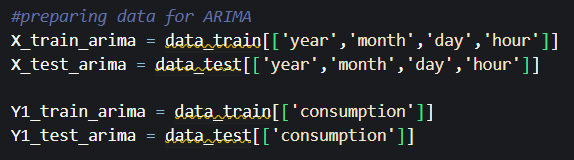


Figure 1: Input data for ARIMA.

As explicitly mentioned in the previous image, the input features used were the variables 'year', 'month', 'day', 'hour', both for the training and testing sets, with the 'consumption' variable used as the dependent variable for both training and testing.

It is also important to clarify that unlike MLP and RNN, ARIMA does not necessarily require, and in fact, it is not recommended to use it with dummy variables. Therefore, dummy variables are not used here.

At the same time, to verify which bar is better for training and prediction, it is ideal to work with all possible types of bars.

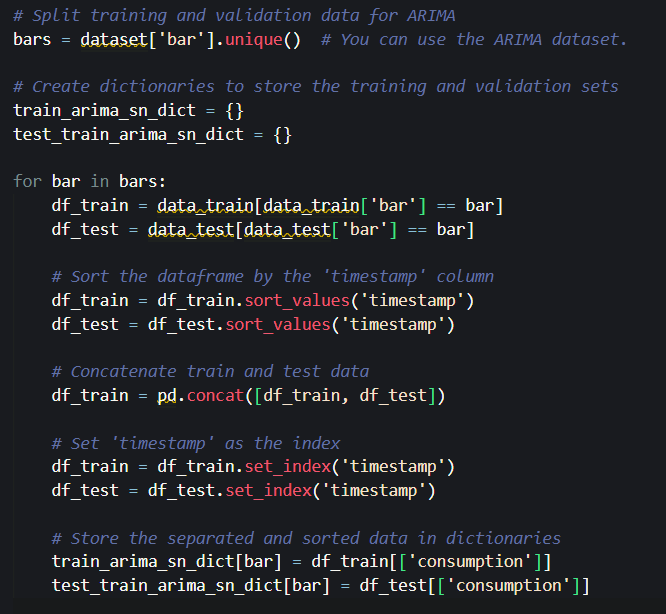


Figure 2: Separation by bar and sorting.

In the previous image, we considered separating the bars for training and validation for future use in ARIMA. Simultaneously, we decided to use the 'timestamp' variable as the INDEX for our dataset, both for the original dataset and the modified dataset.

It's important to remember that in an ARIMA model, the input features are the past values of the time series you're trying to predict. In our case, we use the past values of:

* Consumption: to predict future values.
* Year, month, day, hour, bar: these are used to split your data into training and validation sets and to organize your data in the correct order.

Regarding RNN with ADAM, if we follow the provided code:

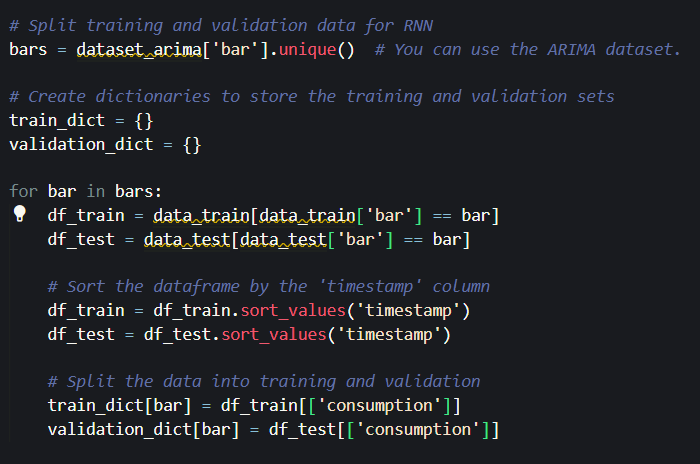


Figure 3: Splitting training and testing data for RNN, along with dictionary creation.

If we look at the previous image, we can still see that it uses an identical structure to that of ARIMA in the use of dictionaries and the importance of the substations (stored in bars).

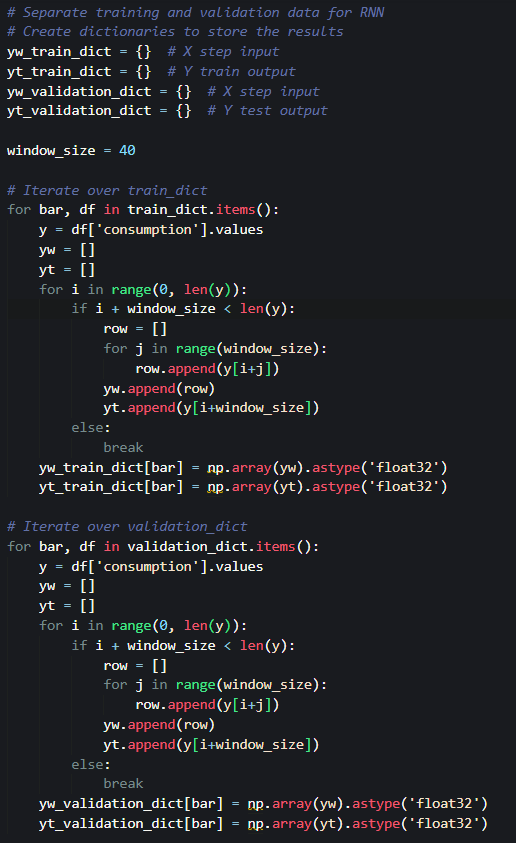


Figure 4: Preparing data for RNN for prediction.

The previous code prepares the training and validation data for our Recurrent Neural Network (RNN) and divides the time series into windows of size 'window\_size' to generate input and output sequences. Converting the data to type 'float32' is essential for the proper functioning of the RNN.

Given this information, we can mention that our input features would be:

* Time Windows (window\_size = 40): This means that our model uses the previous 40 'consumption' values to predict the current value. These time windows define how much temporal information the model is using to make predictions.
* Consumption: The target variable, historical consumption data is used to train the model and make future predictions.
* Bar: Used to expect different consumption patterns for each bar (substation name).

Now, it is important to consider that 'year', 'month', 'day', 'hour' are not used as direct input features to the RNN model, but they are implicitly included in our 'TIMESTAMP' index.

# Description of the parameters used to train the model

Before that, we should mention that in both our Dataset, with or without Outliers, it is important to clarify, as mentioned in the previous point, how our modified datasets are structured with a timestamp index from 2018 to 2022. Each column represents a substation in the format 'bar\_stationName, Substation belongs to {AJAHUEL, BUIN, CHENA, CNAVIA, ELSALTO, FLORIDA, LOSALME}'. The datasets designed for ARIMA model usage have the following structure:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Figure 5: Structure of datasets oriented for ARIMA modeling (Spanish Ver. Source)

As seen in the previous image, we will use the timestamp as the INDEX, and we will use the bars and consumption as our dependent variables, where we will predict possible future consumption values for each bar (substation name) based on our timestamp. Remember that as mentioned earlier, we will use the timestamp from 2018 to 2021 for training and 2022 for testing. For ARIMA, we will use our independent variables for training and testing using 'year', 'month', 'day', 'hour', and our dependent variable for training and testing will be 'consumption'.

As recalled in Part 1 of the assignment, the autocorrelation and partial autocorrelation plots have provided us with a value of for ARIMA. Additionally, by default, a value of was used. It's important to clarify the use of these values, as they were determined based on the graphical analysis of autocorrelation and partial autocorrelation seen in the code of Part 1.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Figure 6: ARIMA Plots for AJAHUEL

Gráfico

Descripción generada automáticamente

Figure 7: ARIMA Plots for BUIN

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente con confianza media

Figure 8: ARIMA Plots for CHENA

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figure 9: ARIMA plots for CNAVIA

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figure 10: ARIMA Plots for ELSALTO

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Figure 11: ARIMA plot for FLORIDA

Gráfico

Descripción generada automáticamente

Imagen 12: Gráficos de ARIMA para LOSALME

### By observing each autocorrelation and partial autocorrelation plot, we appreciate that the optimal values to use them as ARIMA parameters would be (p=1, q=1), those, taking d = 1 afterward, will be our parameters used for our predictions.

### Regarding the values that can be obtained with the statistical summaries with model.summary(), they also have their total importance for our parameter analysis that will be analyzed, by bar (substation name).

### Before going to the statistical analysis, it is important to mention some parameters that we will have in each SARIMAX Results table, and its relation with ARIMA:

### Log Likelihood: It is the logarithm of the likelihood function. The higher this value, the better the model fits the data.

### AIC (Akaike Information Criterion): It is a model selection criterion. Models with lower AIC are preferable.

### BIC (Bayesian Information Criterion): Like AIC, BIC is a model selection criterion. Models with lower BIC are preferable.

### Coef: These are the estimated coefficients for each term of the model.

### Std err: It is the standard error of the estimated coefficients. The smaller this value, the more reliable the coefficient estimation.

### Jarque-Bera (JB): It is a test for the normality of residuals. If the associated p-value (Prob(JB)) is small (usually less than 0.05), you can conclude that the residuals are not normal.

### These parameters will be considered for analysis of each table for each substation.

### Statistical Analysis with Dataset with Outliers (ARIMA).

For AJAHUEL, we have the following statistical summary:

Interfaz de usuario gráfica

Descripción generada automáticamente

*Image 13: Statistical Summary of AJAHUEL*

In AJAHUEL, we observe:

* Log Likelihood: -178581.090 (no expected fit for AJAHUEL).
* AIC: 357168.1
* BIC: 357194.1
* Std error: 0.265 (oriented to sigma2).
* Jarque-Bera: 22963182.2

For BUIN, we have the following statistical summary:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Image 14: Statistical Summary of BUIN

In BUIN, we observe:

* Log Likelihood: -71812.2 (no expected fit for BUIN).
* AIC: 143630.5
* BIC: 143656.6
* Std error: 0.000 (oriented to sigma2).
* Jarque-Bera: 607431297802.6

For CHENA, we have the following statistical summary:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

*Image 15: Statistical Summary of CHENA*

In CHENA, we observe:

* Log Likelihood: -182723.9 (no expected fit for CHENA).
* AIC: 365453.9
* BIC: 365480.03
* Std error: 0.286 (oriented to sigma2).
* Jarque-Bera: 23132270.91

For CNAVIA, we have the following statistical summary:

* Interfaz de usuario gráfica, Texto

  Descripción generada automáticamente

*Image 16: Statistical Summary of CNAVIA*

In CNAVIA, we observe:

* Log Likelihood: -212846.3 (no expected fit for CNAVIA).
* AIC: 425698.6
* BIC: 425698.6
* Std error: 1.539 (oriented to sigma2).
* Jarque-Bera: 8014114.06

For ELSALTO, we have the following statistical summary:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

*Image 17: Statistical Summary of ELSALTO*

In ELSALTO, we observe:

* Log Likelihood: -212354.3 (no expected fit for ELSALTO).
* AIC: 424714.6
* BIC: 424714.6
* Std error: 0.958 (oriented to sigma2).
* Jarque-Bera: 32694744.02

For FLORIDA, we have the following statistical summary:

Interfaz de usuario gráfica

Descripción generada automáticamente

*Image 18: Statistical Summary of FLORIDA*

In FLORIDA, we observe:

* Log Likelihood: -129488.175 (no expected fit for FLORIDA).
* AIC: 258982.35
* BIC: 259008.4
* Std error: 0.017 (oriented to sigma2).
* Jarque-Bera: 45537301.6

For LOSALME, we have the following statistical summary:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Image 19: Statistical Summary of LOSALME

In LOSALME, we observe:

* Log Likelihood: -177604.3 (no expected fit for LOSALME).
* AIC: 355214.7
* BIC: 355240.8
* Std error: 0.262 (oriented to sigma2).
* Jarque-Bera: 23670295.7

So far, with outliers in ARIMA, we have FLORIDA as the ideal bar for training and testing

### Statistical Analysis with Dataset without Outliers (ARIMA).

For AJAHUEL, we have the following statistical summary:

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

Imagen20: Statistical Summary of AJAHUEL

In AJAHUEL, we observe:

* Log Likelihood: -178353.1 (no expected fit for AJAHUEL).
* AIC: 356712.2
* BIC: 356738.3
* Std error: 0.271 (oriented to sigma2).
* Jarque-Bera: 19443947.31

For BUIN, we have the following statistical summary

:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Image 21: Statistical Summary of BUIN

In BUIN, we observe:

* Log Likelihood: -234428.41 (no expected fit for BUIN).
* AIC: 468862.83
* BIC: 468888.8
* Std error: 6.061 (oriented to sigma2).
* Jarque-Bera: 496473.67

For CHENA, we have the following statistical summary:

:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

mage 22: Statistical Summary of CHENA

In CHENA, we observe:

* Log Likelihood: -237385.3 (no expected fit for CHENA).
* AIC: 474776.6
* BIC: 474802.6
* Std error: 7.016 (oriented to sigma2).
* Jarque-Bera: 603068.59

For CNAVIA, we have the following statistical summary:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Image 23: Statistical Summary of CNAVIA

In CNAVIA, we observe:

* Log Likelihood: -218026.2 (no expected fit for CNAVIA).
* AIC: 436058.559
* BIC: 436084.613
* Std error: 1.920 (oriented to sigma2).
* Jarque-Bera: 5546041.53

For ELSALTO, we have the following statistical summary:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Image 24: Statistical Summary of ELSALTO

in **ELSALTO**, the statistical results are as follows:

* Log Likelihood: -219819.569 (no expected fit for ELSALTO).
* AIC: 439645.139
* BIC: 439671.192
* Std error: 1.872 (oriented to sigma2).
* Jarque-Bera: 7588057.31

For **FLORIDA**, we have the following statistical summary:

Interfaz de usuario gráfica

Descripción generada automáticamente

Image 25: Statistical Summary of FLORIDA

in **FLORIDA**, the statistical results are as follows:

* Log Likelihood: -129488.175 (no expected fit for FLORIDA).
* AIC: 258982.351
* BIC: 259008.404
* Std error: 0.017 (oriented to sigma2).
* Jarque-Bera: 45537301.61

For **losalme**, we have the following statistical summary:

Interfaz de usuario gráfica

Descripción generada automáticamente

Image 26: Statistical Summary of LOSALME

in LOSALME , the statistical results are as follows:

* Log Likelihood: -178097.691 (no expected fit for FLORIDA).
* AIC: 356201.382
* BIC: 356227.436
* Std error: 0.270 (oriented to sigma2).
* Jarque-Bera: 22049415.72

So far, without outliers in ARIMA, FLORIDA seems to be the best choice for both training and testing.

## **Parameters of RNN with ADAM (Special Mention Model**).

As mentioned, we will also analyze our Recurrent Neural Network (RNN) with ADAM, as an honorary model to provide a more complete analysis and complement ARIMA information with RNN.

About the parameters mentioned, Because it will be further analyzed [Description of the architecture of the model used](file:///C:\Users\agust\Downloads\Nogger_best_helado%20(3).docx#_Descripción_de_la), we'll give a reference to the basics of the code seen in the previous point:

* **window\_size**: This is the size of the window used to create input sequences for the RNN model. In this case, the window size is 40.
* **Input shape**: The input shape for the model is (window\_size, 1). This means the model expects sequences of length 40, and each element of the sequence is a vector of dimension 1.
* **SimpleRNN**: This is the type of recurrent layer used in the model. In this case, a SimpleRNN layer is used with an output size equal to the window size.
* **Dense**: This is the output layer of the model. It has a single unit with a linear activation function.
* **Optimizer**: The optimizer used to train the model is Adam.
* **Metrics**: The metrics used to evaluate the model's performance during training are Mean Squared Error (MSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE).
* **Epochs**: The number of epochs for which the model is trained is 36.

Now, for the statistical analysis with **summary()** in RNN, it's important to explain the meaning of each one:

* Layer (type): It's the type of layer in the model, where there are two types:
  + SimpleRNN: describing it is a layer where the outputs of certain neurons are fed back to the input to help the network 'remember' information as time progresses
  + Dense: Capa densa, capa de red neuronal donde cada neurona recibe entrada de todas las neuronas de la capa anterior.
* Output Shape: It's the shape of the output of each layer.
* Param: It's the number of parameters that can be learned for each stage.
* **Total Params**: This is the total trainable parameters in the model.

We have the following values to consider for our summary en RNN:

### RNN summary with outliers:

Texto

Descripción generada automáticamente

Image 27: Summary of RNN with outliers

Given the above image, we appreciate the following information:

* In Layer (type): As mentioned, there are 2 layers: SimpleRNN and dense\_3.
* In Output Shape, it has an output layer of (None,40), meaning given a number of input batches, it will produce an output of size 40, while (None, 1) means given a number of input batches it will produce an output of size 1.
* Param: SimpleRNN has 1680 parameters, and dense\_3 has 41 parameters.
* Total Params: In total, there are 1721 parameters.

This analysis provides a basic understanding of the parameters and results obtained in the RNN model with outliers.

### **Summary of RNN with Outliers:**:

Texto

Descripción generada automáticamente

Image 28: Summary of RNN with Outliers

Given the above image, we can observe the following information:

* **Layer (type):** As mentioned, there are 2 layers: SimpleRNN and dense\_19.
* **Output Shape:** The output layer has a shape of (None,40), meaning that given a number of input batches, it will produce an output of size 40. Meanwhile, (None, 1) indicates that the layer will produce an output of size 1 given a number of input batches.
* **Param:** SimpleRNN has 1680 parameters, and dense\_3 has 41 parameters.
* **Total Params:** In total, there are 1721 parameters.

This summary provides insight into the structure and parameters of the RNN model with outliers.

# Description of the architecture of the model

## used Algorithmic and mathematical architecture of ARIMA (winning model)

### with data and graphs..

Mathematical analysis. As we remember, ARIMA aims to analyze and predict time series to forecast the future based on past values. If we recall mathematically, ARIMA can be expressed as:

Where is the time series, is the lag operator, are the model parameters, is the order of differencing, are the parameters of the moving average model, and is the error at time , assumed to follow a normal distribution with mean 0 and constant variance.

In our case, ∈

This means, represents our time series that we are modeling, oriented to our substations.

As mentioned, represents our lag operator, in practice, this means using past values of our time series to make predictions. In our code, it is handled internally in the ARIMA function when we specify as our ARIMA parameters..

Regarding , these are the parameters of the autoregressive model. In this case, with **model.fit()**, they are automatically learned. In our case, since for each substation, we would have only one autoregressive parameter which can be interpreted as the influence the immediately previous value of the time series could have on the current prediction.

The differencing order , is part of our moving average model parameters. Since its value is 1, it means we are performing a first-order differencing in our series, with a lag of 1, where is our differencing result, is the current value of the time series at time , and is the previous value at time

The values, are the parameters of the moving average model. They are learned automatically like the parameters, In this case, their function is to help determine the influence of each past prediction error on the current prediction. Since *q*=1, we can mention a moving average model function as where *μ* is the mean of the time series, and the rest of the function has been explained before. However, as we know, refers to our time error, also calculated with **model.fit()** where it is computed as

.

where is the real value and, is the value predicted by the time model.

# Limitations of the model and proposals about how to improve its performance in the future.

Important clarification: the explanations for MSE and MAE can be found in the section "What are MSE and MAE?" to provide an understanding when discussing these values.

Additionally, for a more comprehensive analysis, the graphs can be compared with the statistical results obtained with ARIMA.

For AJAHUEL, the following graph was obtained to predict 2022.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Figure 12: Real values vs. predictions for 2022 in AJAHUEL with ARIMA

In the previous graph, it's noticeable that despite the biases, the predictions manage to cover a significant portion of the real values. However, there is a noticeable discrepancy between March 2022 and May 2022.

Additionally, the following values were obtained:

* MAE: 0.11106657372352198
* MSE: 1.5690193730856743
* AIC: 143630.5687315037

In BUIN, the following graph was obtained to predict 2022.

Gráfico, Histograma

Descripción generada automáticamente

Figure 13: Real values vs. predictions for 2022 in BUIN with ARIMA

In the previous graph, there is not a perfect balance in March 2022, both for real values and predictions. However, for the rest of the months, a good fit is observed.

Additionally, the following values were obtained:

* MAE: 8.612872117551614
* MSE: 252.6910514107488
* AIC: 365453.9844477476

In CHENA, the following graph was obtained to predict 2022.

Gráfico

Descripción generada automáticamente

Figure 14: Real values vs. predictions for 2022 in CHENA with ARIMA

It can be observed that, similar to AJAHUEL, there is a good correspondence between real values and predictions in CHENA, despite the discrepancy between March 2022 and May 2022.

Additionally, the following values were obtained:

* MAE: 8.612872117551614
* MSE: 252.6910514107488
* AIC: 365453.9844477476

In CHENA, the following graph was obtained to predict 2022.

For CNAVIA, the following graph was obtained to predict 2022:

Gráfico

Descripción generada automáticamente

Figure 15: Real values vs. predictions for 2022 in CNAVIA with ARIMA

We can observe from the previous image that, similar to AJAHUEL and CHENA, we have a nearly identical structure as mentioned earlier.

Additionally, the following values were obtained:

* MAE: 19.122630174875386
* MSE: 1003.6770766618824
* AIC: 425698.6492081627

For ELSALTO, the following graph was obtained to predict 2022.

Gráfico

Descripción generada automáticamente

Figure 16: Real values vs. predictions for 2022 in ELSALTO with ARIMA

We can observe from the previous image that, similar to AJAHUEL, CHENA, and CNAVIA, we have a nearly identical structure as mentioned earlier, although with a greater bias in March 2022.

Additionally, the following values were obtained:

* MAE: 17.54472937275069
* MSE: 979.7530138569809
* AIC: 424714.6714744745

For FLORIDA, the following graph was obtained to predict 2022.

Gráfico, Histograma

Descripción generada automáticamente

Figure 17: Real values vs. predictions for 2022 in FLORIDA with ARIMA

This time, we observe that the behavior is not the same as with the previous substations. However, it is still noticeable that the predictions follow the trend of the real values.

Additionally, the following values were obtained:

* MAE: 1.6953630055610456
* MSE: 22.01202381204552
* AIC: 258982.35080836996

For LOSALME, the following graph was obtained to predict 2022.

Gráfico

Descripción generada automáticamente

Figure 18: Real values vs. predictions for 2022 in LOSALME with ARIMA

We observe good performance in the previous image; however, there is significant discrepancy in March 2022 and May 2022, which undermines confidence.

Additionally, the following values were obtained:

* MAE: 8.210784909277896
* MSE: 200.46075057900393
* AIC: 355214.7373781807

We note that the substations with the lowest possible error index in both MAE and MSE are AJAHUEL and FLORIDA, with AJAHUEL being the preferred choice and FLORIDA as the second option.

Regarding the analysis of whether the series are stationary or not, the KPSS test was conducted:

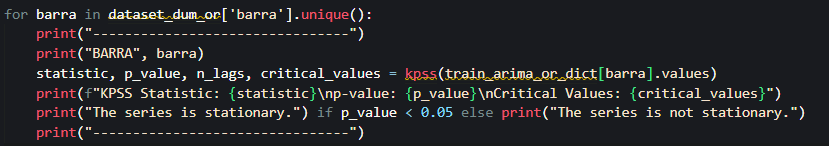


Figure 19: KPSS test on ARIMA, algorithm implemented.

--------------------------------

BARRA AJAHUEL

KPSS Statistic: 2.938176669701572

p-value: 0.01

Critical Values: {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739}

The series is stationary.

--------------------------------

--------------------------------

BARRA BUIN

KPSS Statistic: 2.5546466699244013

p-value: 0.01

Critical Values: {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739}

The series is stationary.

--------------------------------

--------------------------------

BARRA CHENA

KPSS Statistic: 7.129680832611945

p-value: 0.01

Critical Values: {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739}

The series is stationary.

--------------------------------

--------------------------------

BARRA CNAVIA

KPSS Statistic: 6.212401899408551

p-value: 0.01

Critical Values: {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739}

The series is stationary.

--------------------------------

--------------------------------

BARRA ELSALTO

KPSS Statistic: 3.5247620942521523

p-value: 0.01

Critical Values: {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739}

The series is stationary.

--------------------------------

--------------------------------

BARRA FLORIDA

KPSS Statistic: 1.4560520749754136

p-value: 0.01

Critical Values: {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739}

The series is stationary.

--------------------------------

--------------------------------

BARRA LOSALME

KPSS Statistic: 0.8370722283844385

p-value: 0.01

Critical Values: {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739}

The series is stationary.

--------------------------------

It can be observed that in most cases, the p-value is 0.01, concluding that all series are stationary.

## ARIMA-generated plots, MSE-MAE values, and stationary/non-stationary in outlier-free dataset:

For AJAHUEL, the following graph was obtained to predict 2022:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

It yields a result similar to the reference in image 29, with the same imbalance between March 2022 and June 2022.

Additionally, the following values were obtained:

* MAE: 8.297201255604781
* MSE: 206.70822250295922
* AIC: 356712.28298433626

For BUIN, the following graph was obtained to predict 2022

Gráfico, Histograma

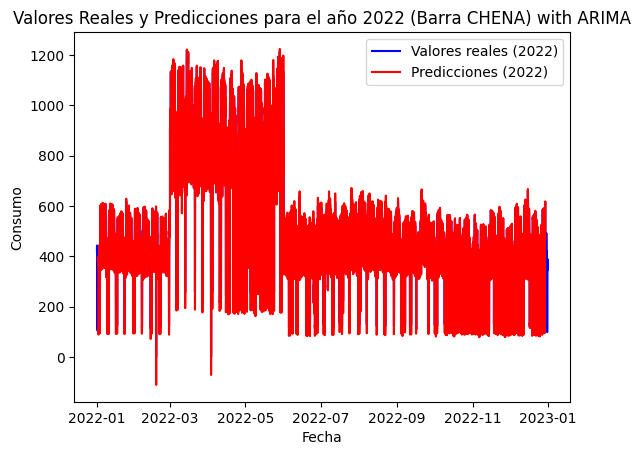
Descripción generada automáticamente

In comparison with the dataset with outliers, this one could fit better to the real values, although it still exhibits significant imbalances at different time intervals.

Additionally, the following values were obtained:

* MAE: 20.281763076291867
* MSE: 2690.13135331007
* AIC: 468862.83658613433

For CHENA, the following graph was obtained to predict 2022:



Similar to the previous one with the dataset containing outliers, this one also exhibits an imbalance between March 2022 and June 2022, but it still fits well to the real values.

Additionally, the following values were obtained:

* MAE: 28.677905213181383
* MSE: 3082.606762608241
* AIC: 474776.63603023964

For CNAVIA, the following graph was obtained to predict 2022:

Gráfico, Histograma

Descripción generada automáticamente

Similar to the previous one with the dataset containing outliers, this one also exhibits an imbalance between March 2022 and June 2022, but it still fits well to the real values.

Additionally, the following values were obtained:

* MAE: 20.754829447741425
* MSE: 1271.623770474885
* AIC: 436058.5589394837

For ELSALTO, the following graph was obtained to predict 2022:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Similar to the previous one with the dataset containing outliers, this one also exhibits an imbalance between March 2022 and June 2022, but it still fits well to the real values.

Additionally, the following values were obtained:

* MAE: 20.876823846181058
* MSE: 1378.5956163559124
* AIC: 439645.13883211475

For FLORIDA, the following graph was obtained to predict 2022:

Gráfico, Histograma

Descripción generada automáticamente

Similar to the previous ones with the dataset containing outliers, this one also exhibits an imbalance between March 2022 and June 2022, but it still fits well to the real values.

Additionally, the following values were obtained:

* MAE: 1.6953630055791638
* MSE: 22.012023812044585
* AIC: 258982.35080836833

For LOSALME, the following graph was obtained to predict 2022:

Gráfico

Descripción generada automáticamente

Similar to the previous ones with the dataset containing outliers, this one also exhibits an imbalance between March 2022 and June 2022, but it still fits well to the real values.

Additionally, the following values were obtained:

* MAE: 8.264317675812723
* MSE: 205.0145630240507
* AIC: 356201.3821182895

We can conclude that the substations FLORIDA and AJAHUEL had better results in terms of MAE and MSE for this ARIMA model.

Regarding the analysis of stationary/non-stationary series using the KPSS test:

For BARRA AJAHUEL:

* KPSS Statistic: 0.9794760789016098
* p-value: 0.01
* Critical values: {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739}
* The series is stationary.

For BARRA BUIN:

* KPSS Statistic: 0.8747997965945181
* p-value: 0.01
* Critical values: {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739}
* The series is stationary.

For BARRA CHENA:

* KPSS Statistic: 2.152343848551646
* p-value: 0.01
* Critical values: {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739}
* The series is stationary.

For BARRA CNAVIA:

* KPSS Statistic: 4.292372256555857
* p-value: 0.01
* Critical values: {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739}
* The series is stationary.

For BARRA ELSALTO:

* KPSS Statistic: 0.9528920240212954
* p-value: 0.01
* Critical values: {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739}
* The series is stationary.

For BARRA FLORIDA:

* KPSS Statistic: 1.2942786316813313
* p-value: 0.01
* Critical values: {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739}
* The series is stationary.

For BARRA LOSALME:

* KPSS Statistic: 4.612052789120463
* p-value: 0.01
* Critical values: {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739}
* The series is stationary.

All substations are stationary with a p-value of 0.01.

## The algorithmic and mathematical architecture of RNN with Adam (model with special mention) with data and graphics.

### Mathematical Analysis.

A recurrent neural network (RNN) has one or more layers of input, hidden, and output. The particularity of the RNN is that the hidden layers have feedback connections (meaning, the output "feeds back" into the same layer as time progresses).

In an RNN, an input and a previous output are taken to produce a new output using the following formula:

Where is the weight matrix for the feedback connections, is the weight matrix for the input connections, is the bias vector, and is the activation function, where , and ​ are my RNN parameters.

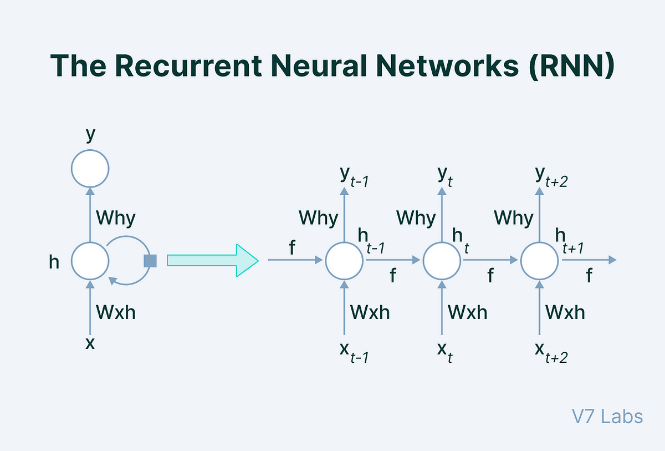


Figure 20: Reference of an RNN, with the parameters explained previously mathematically

Within the code, represents the input elements corresponding to an element in the input sequence, which are contained in the lists yw. Concerning , , and, they are internal parameters of the SimpleRNN layer that are learned during training, which is automatically handled by Keras. Regarding σ, it is the activation function used by SimpleRNN, which by default employs the activation='linear' function in this case. As for , it represents the hidden state at time t-1, which is used to "feed" the input sequence to the RNN. This occurs using model.compile(). Meanwhile, corresponds to the current output layer using SIMPLERNN, where, with the previously seen calculation function and parameters, is calculated, subsequently becoming in the next time step.

## Parameters oriented towards the architecture of RNN.

The parameters covered in this report will be as follows:

* Input: Data/information provided to the RNN.
* Encoding: Formatting the data to be received by the RNN.
* Hidden layers: Layers between the input and output layer.
* Number of neurons: Quantity of neurons in the RNN.
* Output layer dimension: Number of neurons in the RNN's output layer.
* Activation function: Function used for learning and processing in the RNN.
* Cost function: Function for analyzing the model's performance, understanding how the model works, and error analysis for error reduction.

## What are MSE and MAE?

Also, at the same time, much consideration will be given to the following errors that may be encountered:

* MSE: Mean Squared Error, a measure of error used to compare the estimates or predictions of a model with the actual values using a square, calculated as:
* MAE: Mean Absolute Error, a measure of error used to compare the estimates or predictions of a model with the actual values using absolute value, calculated as:

Here, n is the total number of observations, is the actual value given observation i, and is the predicted value given observation i.

Both error measures will be used, which will be important for understanding the choice of the best model (ARIMA vs RNN) in the future.

## Graphs provided by RNN, MSE-MAE values, and epoch vs loss graphs in dataset with outliers:

For AJAHUEL, the following results were obtained:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Figure 21: Plot of predictions vs real values for AJAHUEL with RNN

We will notice a certain resemblance in behavior with the graph of AJAHUEL obtained with ARIMA; we will indeed observe this with all the bars (substation names). The MAE and MSE values are:

* Final MAE: 17.747880935668945
* Final MSE: 1421.915771484375

Regarding the epoch vs loss graph obtained for AJAHUEL, thanks to tensorboard:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Figure 22: Epoch vs Loss graph for AJAHUEL

From epoch 10 onwards, the error starts decreasing very slowly, suggesting that the best epoch to stop training is between epochs 10 and 13.

Our model's architecture is as follows:

* Input: The input provided was a time sequence with a batch size of 30.
* Encoding: It is encoded in float32 format.
* Hidden Layers: It has only one hidden layer of type simple RNN.
* Number of Neurons: 40.
* Output Layer Dimension: 1.
* Activation Function: Linear.
* Loss Function: MSE (Mean Squared Error).

For BUIN, the following results were obtained:

Gráfico

Descripción generada automáticamente

Figure 23: Prediction vs. actual values plot for BUIN with RNN

Here, however, we can see a plot with significant imbalance and mismatch between real values and predictions, so it's not a confidence-inspiring graph.

The MAE and MSE values are:

* Final MAE: 1.0272787809371948
* Final MSE: 5.204549789428711

Regarding the epoch vs loss graph obtained through tensorboard:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Figure 24: Epoch vs Loss graph for BUIN

In this image, we can see that our best epoch with the best possible values for MSE and MAE was epoch 17.

Our model's architecture is as follows:

* Input: The input provided was a time sequence with a batch size of 30.
* Encoding: It is encoded in float32 format.
* Hidden Layers: It has only one hidden layer of type simple RNN.
* Number of Neurons: 40.
* Output Layer Dimension: 1.
* Activation Function: Linear.
* Loss Function: MSE (Mean Squared Error).

Principio del formulario

For CHENA, the following results were obtained:

Gráfico

Descripción generada automáticamente

Figure 25: Graph of predictions vs actual values in CHENA with RNN

We observe, similar to ARIMA, a good pattern tracking in the graph. The MAE and MSE values are as follows:

* Final MAE: 36.221317291259766
* Final MSE: 5499.35888671875

Regarding the epoch vs loss graph, we obtained the following insights, thanks to tensorboard:

Gráfico

Descripción generada automáticamente con confianza media

Figure 26: Epoch vs Loss graph for CHENA

It can be observed that the error does not decrease much after the 4th epoch.

Our model's architecture is as follows:

* Input: The input provided was a time sequence with a batch size of 30.
* Encoding: It is encoded in float32 format.
* Hidden Layers: It has only one hidden layer of type simple RNN.
* Number of Neurons: 40.
* Output Layer Dimension: 1.
* Activation Function: Linear.
* Loss Function: MSE (Mean Squared Error).

For CNAVIA, the following results were obtained:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Figure 27: Graph of predictions vs actual values in CNAVIA with RNN

We observe a behavior quite similar to ARIMA. The MAE and MSE values are:

* Final MAE: 74.43292236328125
* Final MSE: 22694.056640625

And regarding the epoch vs loss graph obtained through TensorBoard:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Figure 28: Epoch vs loss graph for CNAVIA

In this case, the error stopped decreasing drastically from epoch 5 onwards.

Our model's architecture is as follows:

* Input: The input provided was a time sequence with a batch size of 30.
* Encoding: It is encoded in float32 format.
* Hidden Layers: It has only one hidden layer of type simple RNN.
* Number of Neurons: 40.
* Output Layer Dimension: 1.
* Activation Function: Linear.
* Loss Function: MSE (Mean Squared Error).

For ELSALTO, the following results were obtained:

Gráfico, Histograma

Descripción generada automáticamente

Figure 29: Graph of predictions vs actual values in ELSALTO with RNN

We observe a behavior quite similar to ARIMA. The MAE and MSE values are as follows:

* **Final MAE:** 87.69651794433594
* **Final MSE:** 29617.1015625

Regarding the epoch vs loss graph, we obtained the following information from TensorBoard:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Figure 30: Epoch vs loss graph for ELSALTO

There isn't much improvement from the beginning to the end, as it follows almost a horizontal line. Our element is:

* Input: The input provided was a time sequence with a batch of 30.
* Encoding: It comes in float32 format.
* Hidden layers: It only has one hidden layer of simple RNN type.
* Number of neurons: 40.
* Output Layer Dimension: 1.
* Activation Function: Linear.
* Cost Function: MSE

For FLORIDA, the following results were obtained:

Gráfico, Gráfico de líneas, Histograma

Descripción generada automáticamente

Figure 31: Graph of predictions vs actual values in FLORIDA with RNN

We observe a behavior very similar to ARIMA; in particular, it fits better than the other substations. The MAE and MSE values are:

* Final MAE: 4.054068565368652
* Final MSE: 92.0399169921875

And regarding the epoch vs loss graph, we obtain it thanks to tensorboard:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Figure 32: Epoch vs loss graph for FLORIDA

Here, it can be observed that the error decreases during training but does not decrease for validation. Our elements are:

* Input: The input provided was a time sequence with a batch size of 30.
* Encoding: It enters in float32 format.
* Hidden layers: It only has one hidden layer of simple RNN type.
* Number of neurons: 40.
* Output Layer Dimension: 1.
* Activation function: Linear.
* Cost function: MSE

For LOSALME, the following results were obtained:

Gráfico

Descripción generada automáticamente

Figure 33: Graph of predictions vs actual values in LOSALME with RNN

We observe a behavior similar to ARIMA, but like the others, it does not fit well between the 1500-3000 range of the sample. The MAE and MSE values are:

* Final MAE: 13.163113594055176
* Final MSE: 636.6898803710938

And regarding the epoch vs loss graph obtained through TensorBoard:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Figure 34: Epoch vs loss graph for LOSALME

There isn't much improvement after epoch 3. Our setup is as follows:

* Input: Time series data with a batch size of 30.
* Encoding: Input data is encoded in float32 format.
* Hidden Layers: Only one hidden layer of simple RNN type.
* Number of Neurons: 40.
* Output Layer Dimension: 1.
* Activation Function: Linear.
* Cost Function: MSE.

## Graphs provided by RNN, MSE-MAE values, and epoch vs loss graphs in dataset without outliers:

For AJAHUEL, the following results were obtained:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Figure 35: Predictions vs real values plot of AJAHUEL with RNN

We can observe a similar behavior to the plot of AJAHUEL taken with RNN with outliers. The MAE and MSE values are:

* Final MAE: 17.461612701416016
* Final MSE: 1447.413818359375

Regarding the epoch vs loss graph taken for AJAHUEL, thanks to TensorBoard:

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

Figure 36: Epoch vs loss graph for AJAHUEL

Here, it can be observed that from epoch 10 onwards, there is not much improvement in its error, which means it reaches its best point at epoch 10. Our setup is as follows:

* Input: The input provided was a time sequence with a batch size of 30.
* Encoding: Entered in float32 format.
* Hidden layers: It only has one hidden layer of a simple RNN type.
* Number of neurons: 40.
* Output layer dimension: 1.
* Activation function: Linear.
* Cost function: MSE

For BUIN, the following results were obtained:

Gráfico, Histograma

Descripción generada automáticamente

Figure 37: Predictions vs real values plot of BUIN with RNN

Here, however, one can appreciate a graph with too much imbalance and mismatch between actual values and predictions, hence, it might not be a reliable graph.

The MAE and MSE values are as follows:

* Final MAE: 1.0272787809371948
* Final MSE: 5.204549789428711

Regarding the epoch vs loss graph obtained through TensorBoard:

Gráfico

Descripción generada automáticamente con confianza baja

Figure 38: Epoch vs loss graph for BUIN

There isn't much improvement from the beginning to the end. Our setup is as follows:

* Input: The input provided was a time sequence with a batch size of 30.
* Encoding: Entered in float32 format.
* Hidden layers: Only has one hidden layer of a simple RNN type.
* Number of neurons: 40.
* Output layer dimension: 1.
* Activation function: Linear.
* Cost function: MSE.

For CHENA, the following results were obtained:

Gráfico, Histograma

Descripción generada automáticamente

Figure 39: Predictions vs real values plot of CHENA with RNN

We observe a similar pattern tracking as in ARIMA, except for the range between 1500-3000. The MAE and MSE values are as follows:

* Final MAE: 97.60198211669922
* Final MSE: 32242.763671875

Regarding the epoch vs loss graph obtained through tensorboard:

Gráfico, Gráfico de líneas, Gráfico de dispersión

Descripción generada automáticamente

Figure 40: Epoch vs loss graph for CHENA

The training error stops decreasing after epoch 3, but its validation error decreases significantly until epoch 5. Our setup includes:

* Input: The input provided was a time sequence with a batch size of 30.
* Encoding: It enters in float32 format.
* Hidden layers: It only has one hidden layer of type simple RNN.
* Number of neurons: 40.
* Output layer dimension: 1.
* Activation function: Linear.
* Cost function: MSE

For CNAVIA, the following results were obtained:

Gráfico, Histograma

Descripción generada automáticamente

Figure 41: Predictions vs real values plot of CNAVIA with RNN

In this case, compared to the other RNN predictions, it deviates significantly between the range of 1500-3000 without being able to follow the pattern. The MAE and MSE values are:

* Final MAE: 88.85162353515625
* Final MSE: 30633.515625

Regarding the epoch vs loss graph obtained through TensorBoard:

Gráfico

Descripción generada automáticamente con confianza media

Figure 42: Epoch vs loss graph for CNAVIA

The improvement in the training error for ELSALTO is minimal; however, the validation error fluctuates slightly. Our model configuration is as follows:

* Input: Sequential time series data with a batch size of 30.
* Encoding: Encoded in float32 format.
* Hidden Layers: Only one hidden layer of a simple RNN type.
* Number of Neurons: 40.
* Output Layer Dimension: 1.
* Activation Function: Linear.
* Cost Function: MSE.

Now, let's discuss the results for ELSALTO:

Gráfico

Descripción generada automáticamente

Figure 43: Predictions vs real values plot of ELSALTO with RNN

The same phenomenon as observed in CNAVIA but with even more pronounced discrepancies in the same range. The MAE and MSE values are:

* Final MAE: 77.2448501586914
* Final MSE: 21839.796875

Regarding the epoch vs loss graph obtained from TensorBoard:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Figure 44: Epoch vs loss graph for ELSALTO

Similarly to the previous case, there isn't much improvement in the training error, but from epoch 32 onwards, a significant drop in validation error can be observed.

Our setup is as follows:

* Input: Time sequence data with a batch size of 30.
* Encoding: Input data in float32 format.
* Hidden layers: Only one hidden layer of Simple RNN type.
* Number of neurons: 40.
* Output layer dimension: 1.
* Activation function: Linear.
* Cost function: MSE.

The results obtained for FLORIDA are as follows:

Gráfico, Histograma

Descripción generada automáticamente

Figure 45: Predictions vs real values plot of FLORIDA with RNN

We observe a behavior very similar to ARIMA, and in this particular case, the model fits better than the other substations.

The MAE and MSE values are as follows:

* Final MAE: 3.005073308944702
* Final MSE: 54.2440299987793

Regarding the epoch vs loss graph obtained from TensorBoard:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Figure 46: Epoch vs loss graph for FLORIDA

The training error stops growing from epoch 6 onwards, and its validation error begins to oscillate as the epochs progress.

Our setup is as follows:

* Input: The input provided was a time sequence with a batch size of 30.
* Encoding: The input is in float32 format.
* Hidden layers: There is only one hidden layer of type simple RNN.
* Number of neurons: 40.
* Output layer dimension: 1.
* Activation function: Linear.
* Cost function: MSE.

For LOSALME, the following results were obtained:

Gráfico, Histograma

Descripción generada automáticamente

Figure 47: Predictions vs real values plot of LOSALME with RNN

Same observation with the same model using outlier data.

The MAE and MSE values are:

* Final MAE: 14.747123718261719
* Final MSE: 859.4706420898438

And regarding the epoch vs loss graph obtained through TensorBoard:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Figure 48: Epoch vs loss graph for LOSALME

It can be observed that the validation error starts to increase from the beginning of training, but around epoch 32 or 33, it decreases slightly below the initial error we had. However, it starts to increase again afterward.

Our element is:

* Input: The input provided was a time sequence with a batch size of 30.
* Encoding: Entered in float32 format.
* Hidden Layers: Only has one hidden layer of type simple RNN.
* Number of Neurons: 40.
* Output Layer Dimension: 1.
* Activation Function: Linear.
* Cost Function: MSE.

# Explanation of how the winning model (ARIMA) was determined using graphs:

To explain the winning model (as mentioned, ARIMA), the analysis was conducted using MSE and MAE:

A lower MSE and lower MAE among the models indicate that the model has better predictions compared to other models or a previous version of the same model. To recall some concepts:

* A lower MSE indicates that the model's predictions tend to be closer to the actual values, as MSE measures the average of the squares of the differences between the model's predictions and the actual values.
* A lower MAE indicates that the model's predictions have smaller absolute errors, meaning the absolute differences between the predictions and the actual values are smaller.

So, we will compare ARIMA with RNN with ADAM, and implicitly compare with MLP (Multilayer Perceptron). Although we obtained better results in AJAHUEL and FLORIDA based on MSE and MAE values, we will consider FLORIDA as the better substation for training and testing. This decision is because the behavior of the graph with both ARIMA and RNN provides more confidence compared to AJAHUEL.

Additionally, although MLP was not considered for the report, it is implemented in the code. Hence, it will be implicitly mentioned for comparison among "MLP vs ARIMA vs RNN."

## Comparison with outliers.

In FLORIDA, given the dataset with outliers, we obtained the following values with MLP, ARIMA, and RNN:

mse\_mlp = 1072.09

mse\_arima = 22.012

mse\_rnn = 92.04

mae\_mlp = 16.99

mae\_arima = 1.69

mae\_rnn = 4.05

mape\_mlp = 224.03

mape\_arima = 1.12

mape\_rnn = 1.35

Where, if we also calculated the values of MAPE, we see that the most convenient one is using ARIMA. When we compare the graphs using a bar chart:

Gráfico, Gráfico de barras

Descripción generada automáticamente

Figure 49: Comparison of model performance using MSE

Gráfico, Gráfico de barras

Descripción generada automáticamente

Figure 50: Comparison of model performance using MAE

We observe that ARIMA explicitly yields better results with lower MAE and MSE.

## Comparison without outliers.

The same analysis performed without outliers, observing the values obtained in MLP vs ARIMA vs RNN:

mse\_mlp = 1106.525

mse\_arima = 22.012

mse\_rnn = 54.244

mae\_mlp = 17.517

mae\_arima = 1.695

mae\_rnn = 3.005

mape\_mlp = 255.696

mape\_arima = 1.124

mape\_rnn = 1.158

We also observe that ARIMA has the lowest values for MSE, MAE, and MAPE.

Gráfico

Descripción generada automáticamente

Figure 51: Comparison of model performance using MSE

Gráfico, Gráfico de barras

Descripción generada automáticamente

Figure 52: Comparison of model performance using MAE

We observe the same trend, with ARIMA exhibiting fewer errors in both MAE and MSE. FLORIDA appears to be the preferred option for training and testing. For complete MAE and MSE values, they can be found in the Description of the Model Architecture section.

Principio del formulario

# Conclusions regarding the work, model limitations, and proposals for improving its performance in the future.

## ConclusionsPrincipio del formulario

The comparative analysis between prediction models, particularly ARIMA and RNN with the Adam optimizer, provides a detailed insight into the capability of these models to forecast energy demand in electrical substations. The results highlight the overall superiority of ARIMA over RNN in terms of accuracy and stability, especially evidenced in the FLORIDA substation. This superiority is reflected in the lower MSE and MAE values obtained with ARIMA compared to RNN, both in datasets with outliers and without them. The superior performance of ARIMA suggests its utility as a reference model for predicting energy demand in electrical substations. However, it is important to consider the limitations and considerations associated with the ARIMA model, as well as potential areas for improvement for future studies and applications.

## Model Limitations:

Despite its effectiveness, the ARIMA model has certain limitations that can affect its utility and applicability in real-world environments. These limitations include:

1. Sensitivity to data quality: The performance of the ARIMA model can be affected by the quality of the input data, including the presence of outliers, missing data, or measurement errors. Incomplete and inaccurate historical data may limit the model's ability to capture significant temporal patterns and make accurate predictions.
2. Dependency on hyperparameter selection: The effectiveness of the ARIMA model is influenced by the proper selection of parameters *p*, *d*, and *q*, which determine the amount of autocorrelation and seasonality accounted for in the model. Manual selection of these parameters can be subjective and require domain expertise, while automatic hyperparameter selection approaches may be computationally intensive and not always guarantee the best configuration.
3. Inability to capture complex patterns: ARIMA relies on linear and stationary assumptions about the data, which can limit its ability to capture nonlinear or complex patterns present in some datasets. This limitation can lead to inaccurate or biased predictions, especially in environments with dynamic and variable behaviors.

## Proposals to Improve Future Performance:

To address these limitations and improve the performance of the ARIMA model in the future, the following proposals can be considered:

1. Investigation of hybrid models: Explore the possibility of combining the strengths of different models, such as ARIMA and neural networks, into a hybrid approach. This approach could leverage ARIMA's trend and seasonality modeling capabilities while using deep learning techniques to capture more complex and nonlinear patterns in the data. Combining models could allow for better adaptation to the complexity and variability of energy demand data.
2. Data quality improvement: Perform proper data cleaning and preprocessing to enhance the model's performance. This involves identifying and correcting anomalies, outliers, and missing data before training the model, which can significantly improve prediction accuracy. Additionally, collecting additional data and improving the frequency and granularity of the data can provide additional insights for modeling and prediction.
3. Hyperparameter optimization: Explore the use of automated hyperparameter optimization techniques to find optimal values of p, d, and q in ARIMA, as well as other parameters in deep learning models. This can help improve the accuracy and generalization of the model, ensuring it is well-tuned to the data. Utilizing advanced optimization approaches such as Bayesian optimization or grid search can efficiently explore the hyperparameter space and find optimal configurations.
4. Continuous model performance evaluation: Regularly evaluate the model's performance using appropriate metrics and validate the model with new and updated data. This will allow for the identification of any performance degradation and corrective measures as needed, ensuring the model is robust and reliable in different scenarios. Additionally, implementing real-time monitoring techniques and continuous feedback can help keep the model updated and adapted to changes in data and environmental conditions.
5. Exploration of alternative models: In addition to ARIMA and RNN, explore other prediction models such as state space models, more advanced deep learning models (e.g., LSTM or convolutional neural networks), or approaches based on artificial intelligence techniques such as generative adversarial networks (GANs) to assess their suitability and performance in predicting energy demand in electrical substations. Diversifying approaches can provide new insights and additional perspectives on time series modeling and prediction in this domain.

## Summary of Conclusion, Limitations, and Proposals:

In summary, although the ARIMA model has proven to be an effective tool for predicting energy demand in electrical substations, it is important to recognize its limitations and explore strategies to improve its performance in the future. Combining traditional approaches like ARIMA with more advanced machine learning techniques and adopting rigorous data preprocessing practices can help overcome some of the identified limitations and enhance the accuracy and robustness of the model. Furthermore, ongoing research in this field is crucial to staying abreast of advances in prediction models and data analysis techniques, and to adapting these innovations to the specific needs of the electrical sector. Collaboration among researchers, industry professionals, and policymakers can also contribute to identifying key challenges and effective solutions for managing energy demand and planning electrical infrastructure. Ultimately, the successful application of prediction models like ARIMA in real-world environments depends not only on the quality and performance of the model itself but also on its effective integration with energy management systems and users' ability to interpret and act on the generated predictions. By addressing the limitations of the model and exploring new opportunities to enhance its performance, we can move towards more efficient and sustainable management of electrical energy demand, thereby contributing to the building of a more resilient and sustainable energy future.

# BIBLIOGRAPHY AND HELP MATERIAL:

* Código máquina. (2023, 26 de junio): Análisis Exploratorio de Series de Tiempo con Gráficas de Autocorrelación y Retardo usando Python (Video), YouTube: <https://www.youtube.com/watch?v=U-vdLHdyDLI>
* Ritvikmath. (2020, 20 de marzo): ACF & PACF code example: Time Series Talk (Video), YouTube: <https://www.youtube.com/watch?v=y8opUEd05Dg>
* Schwarzenberg, Pablo: (2023): Repositorio de GitHub (CINF104): <https://github.com/pabloschwarzenberg/CINF104>
* HackersRealm. (2023, 11 de julio): Detect and Remove Outliers in the Data| Machine Learning| Python (Página educativa): <https://www.hackersrealm.net/post/detect-and-remove-outliers-python>
* SuperDataScience Team, María Santos, Juan Gabriel Molina. (noviembre 2023): Machine Learning de A a la Z: R y Python para Data Science (Udemy): [https://www.udemy.com/course/machinelearning-es/learn/lecture/14334826?start=60#overview](https://www.udemy.com/course/machinelearning-es/learn/lecture/14334826?start=60%23overview)
* Santiago Hernández. (): Inteligencia Artificial y Deep Learning desde cero en Python (Udemy): [https://www.udemy.com/course/deep-learning-desde-cero-en-python/learn/lecture/23889650?start=150#overview](https://www.udemy.com/course/deep-learning-desde-cero-en-python/learn/lecture/23889650?start=150%23overview)
* Sebastián Maldonado (2011, 21 de diciembre): Preprocesamiento de Datos (pptx): <https://www.u-cursos.cl/diplomados/2011/2/DIP78J/1/material_docente/bajar?id_material=401889>
* OpenAI. (s.f.). ChatGPT (GPT-3.5). OpenAI: <https://openai.com/chatgpt>
* Joanby (2023, agosto): Repositorio de GitHub (Deep Learning de A a la Z: Redes Neuronales al Completo): <https://github.com/joanby/deeplearning-az>
* BitBoss (2020, 27 de febrero): Redes neuronales: De la neurona al perceptrón multicapa en 9 minutos. El problema XOR (Video), YouTube: <https://www.youtube.com/watch?v=MU3cLsSfnME&t=280s>
* InformatiCódigo (2022, 4 de agosto): SimpleRNN, GRU y LSTM| Redes neuronales RECURRENTES Python Keras 2022 (Video), YouTube:

<https://www.youtube.com/watch?v=bGcxrZMlJ2M>

* Joaquín Amat Rodrigo, Javier Escobar Ortiz (2023, septiembre): Modelos ARIMA y SARIMAX con Python (Página educativa): <https://cienciadedatos.net/documentos/py51-modelos-arima-sarimax-python>

# Our GITHUB:

The github where all the files are located is at the following link: <https://github.com/FenixCompany/Tarea-1-CINF-104>