Universidad Michoacana de San Nicolás de Hidalgo

División de Estudios de Posgrado de la Facultad de Ingeniería Eléctrica

MODELADO DEL ERROR DE PREDICCIÓN EN SERIES DE TIEMPO BASADO EN LA CALIDAD DE SUS DATOS

TESIS

Que para obtener el grado de

MAESTRO EN CIENCIAS EN INGENIERÍA ELÉCTRICA

Presenta

Víctor Manuel Téllez Velázquez

Director de Tesis
Dr. en Ciencias Computacionales Juan José Flores Romero

Morelia, Michoacán agosto del año 2019

MODELADO DEL ERROR DE PREDICCION EN SERIES DE TIEMPO BASADO EN LA CALIDAD DE SUS DATOS

Los Miembros del Jurado de Examen de Grado aprueban la **Tesis de Maestría en Ciencias en Ingeniería Eléctrica** de ***Víctor Manuel Téllez Velázquez***

Dr. Félix Calderón Solorio
Presidente del Jurado

Dr. Juan José Flores Romero
Director de Tesis

Dr. Jaime Cerda Jacobo
Vocal

Dr. José Antonio Camarena Ibarrola
Vocal

Dr. Héctor Rodríguez Rangel
Revisor Externo (Inst. Tecnológico de Culiacán Sinaloa)

Dr. Roberto Tapia Sánchez
Jefe de la División de Estudios de Posgrado de la Facultad de Ingeniería Eléctrica. UMSNH (Por reconocimiento de firmas)

UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO
Agosto 2019

Esta tesis es dedicada a mi director de tesis Juan José Flores Romero. A toda mi familia y amigos en especial a mis padres, Romulo Téllez Loeza, Elvira Velázquez Hernandez. A mis segundos padres Armando Guerrero Zavala y Ma. Elena Téllez Velázquez. A mi novia y futura esposa Ana Rosa Barriga Heredia y su familia.

Resumen

En esta tesis, se aborda el problema de evaluar la calidad de los datos de una serie de tiempo antes de pronosticar. El propósito de la evaluación de la calidad es determinar la precisión aproximada que obtendríamos en el pronóstico con los datos de la serie de tiempo. Para abordar este tema se propone la extracción de cuatro características de la serie de tiempo, las cuales son: porcentaje de datos faltantes, porcentaje de valores atípicos, nivel de ruido y magnitud del caos. Para esto se trabajó con series de tiempo sintéticas, de las cuales nueve son caóticas y una serie de tiempo de la función sinusoidal, usada como referencia. Las diez series de tiempo fueron perturbadas con la remoción de datos faltantes, inclusión de valores atípicos e inclusión de ruido; en total se analizaron 2,160 series de tiempo sintéticas. Después de haber generado las series de tiempo, se obtuvieron las características las cuales son una estimación de los niveles de perturbación que fueron inyectados a las series sintéticas. Las series perturbadas fueron modeladas mediante redes neuronales artificiales; se realizó el pronóstico y se obtuvo el error simétrico medio porcentual absoluto (SMAPE). Con las estimaciones de las características y el error obtenido se entrenó un regresor para determinar el porcentaje de error que puede llegar a tener el pronóstico de una determinada serie de tiempo, sin hacer un modelado de pronóstico. Se obtuvo un nivel de efectividad de este regresor del 85%, lo cual se considera una precisión satisfactoria.

Palabras clave: aprendizaje de máquina, series de tiempo, ruido, regresión, pronóstico.

Abstract

This thesis is about evaluating the quality of data from a time series before forecasting. The purpose of the quality evaluation is to determine the approximate accuracy that we would obtain in the forecast with the time series. To attend this topic, we propose the extraction of four characteristics of the time series, which are: percentage of missing data, percentage of outliers, noise level and magnitude of chaos. For this we worked with synthetic time series, of which nine are chaotic and a time series of the sinusoidal function, used as a reference. The ten time series were disturbed with the removal of missing data, inclusion of outliers and inclusion of noise; In total 2,160 synthetic time series were analyzed. After having generated the time series, the characteristics were obtained, which are an estimate of the perturbation levels that were injected into the synthetic series. The disturbed series were modeled using artificial neural networks; the forecast was made and the absolute percentage average symmetric error smape was obtained. With the estimates of the characteristics and the error obtained, a regressor was trained to determine the percentage of error that the forecast of a certain time series may have without making a forecast modeling. An effectiveness level of this 85% regressor was obtained, which is considered satisfactory accuracy.

Keywords: machine learning, time series, noise, regression, forecast

Índice general

Índice de figuras

Índice de tablas

Índice de algoritmos

Lista de símbolos

$\mathbb{R}^+$ Conjunto de los números reales positivos. 5

$\mathbb{R}^{nV}$ Conjunto de los números reales en un espacio de n_V dimensiones. 4

μ Media o promedio, es la suma de los datos dividida entre el número total de datos. 20

σ La desviación estándar indica qué tan dispersos están los datos con respecto a la media. 20

$^{\circ}C$ Unidad termométrica grado celsius. 21

Glosario

Keras Keras es una biblioteca de Redes Neuronales Artificiales de código abierto escrita en Python. 44–46

linear Es una función lineal usada por Keras. 49

pandas Pandas es una biblioteca de software escrita para el lenguaje de programación Python para la manipulación y análisis de datos. En particular, ofrece estructuras de datos y operaciones para manipular tablas numéricas y series de tiempo. 29

PyAstronomy PyAstronomy (PyA) es una colección de paquetes relacionados con la astronomía desarrollados en Python. 29, 31

Python Python es un lenguaje de programación interpretado su filosofía es la sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, soporta orientación a objetos, programación imperativa y, programación funcional. 16, 44

SciPy SciPy es una colección de algoritmos matemáticos y funciones basadas en la extensión NumPy de Python. Agrega un poder significativo comandos y clases de alto nivel para manipular y visualizar datos. 30

sigmoid Es una función matemática que tiene una curva característica en forma de "S". 49

TensorFlow TensorFlow es una biblioteca de aprendizaje automático de código abierto para investigación y producción. 44

Acrónimos

NaN Se usa NaN para denotar datos faltantes. 29

lrd Densidad de accesibilidad local (Local Reachability Density). 23, 24

2D Dos dimensiones. 10, 13

3D Tres dimensiones. 9–14

API Interfaz de programación de aplicaciones (Application Programming Interface). 44

ARIMA Modelo autorregresivo integrado de promedio móvil (AutoRegressive Integrated Moving Average). 6, 80

AWGN Función para agregar ruido blanco gaussiano a la señal(Add White Gaussian Noise). 32

BSEqSamp Funcionalidad para estimar el ruido a partir de datos muestreados de manera equidistante (Beta Sigma Equidistant Sampling). 31

COF Método de factor atípico basado en conectividad (Connectivity-based Outlier Factor). 22

dB Decibelio es una unidad que se utiliza para expresar la relación entre dos valores de presión sonora, o tensión y potencia eléctrica.. 14, 27, 31–33, 75

FastABOD Método de detección de valores atípicos basada en un ángulo rápido (Fast Angle-Based Outlier Detection). 22

INFLO Método de influencia de valores atípicos (Influenced Outlierness). 22

IQR rango intercuartil (interquartile range). 20

KDEOS Método (Kernel Density Estimation Outlier Score). 22

kNN Método de los k vecinos más cercanos (k nearest neighbors). 21

kNNW Método de los k vecinos más cercanos con pesos (kNN-Weight). 21

kW Kilovatio. Un kilovatio equivale a 1,000 vatios (Watts). 1

LDF Método de factor de densidad local. 22

LDOF Método de factor de valor atípico basado en la distancia local (Local Distance-based Outlier Factor). 21

LOCF Última observación arrastrada (Last Observation Carried Forward). 18

LOF Factor de valor atípico local (Local Outlier Factor). 5, 21, 22, 24

LoOP Método de probabilidades de valores atípicos locales (Local Outlier Probabilities). 21

LSTM Memoria a corto y largo plazo,(Long Short Term Memory). 80

MAE Error absoluto medio (Mean Absolute Error). 59, 63, 64, 79

MAPE Error medio porcentual absoluto (Mean Absolute Percentage Error). 3, 59

MATLAB Laboratorio de matrices es un software de computo numérico (MATrix LABoratory). 32

MLP Perceptrón Multicapa (MultiLayer Perceptron). 3, 52

MSE Error cuadrático medio (Mean Square Error). 46, 66

ODIN Método de detección de valores atípicos usando el número de grado (Outlier Detection using Indegree Number). 22

PSR Reconstrucción del espacio de fase (Phase Space Reconstruction). 45

PyA Python Astronomy es una biblioteca de astronomía desarrollada en Python. 28

relu Función de activación unidad lineal rectificada (Rectified Linear Unit)). 48, 49

RMSE Raíz cuadrada del error cuadrático medio (Root Mean Square Error). 59

RNA Red Neuronal Artificial. 44

SimplifiedLOF Método de LOF Simplificado. 21

SMAPE Error simétrico medio porcentual absoluto (Symmetric Mean Absolute Percentage error). 3, 48–51, 55, 58, 60, 65, 70–72, 76

SNR Relación señal a ruido (Signal to Noise Ratio). 14, 28–30, 32, 33, 39, 40, 70, 77

st Serie de Tiempo.. 1, 20

tanh Función de activación tangente hiperbólica. 49

TCP/IP Protocolo de control de transmisión/protocolo de internet (Transmission Control Protocol/Internet Protocol). 16

Capítulo 1

Introducción

Una serie tiempo es una secuencia de observaciones medidas en determinados momentos del tiempo (minutos, horas, días, semanas, meses, años, etc.), ordenados cronológicamente, espaciados entre sí de manera uniforme así los datos usualmente son dependientes entre sí la notación matemática de una serie de tiempo es; $st = \{x_1, x_2, \ldots, x_N\}$ donde st es la serie de tiempo y $N \geq 1$. El principal objetivo de una serie de tiempo, dentro de su análisis es hacer pronóstico y la razón por la que el pronóstico es tan importante es que la predicción de eventos futuros es un aporte crítico en muchos tipos de procesos de planificación y toma de decisiones. Algunos ejemplos de donde se pueden utilizar las series de tiempo para pronóstico son: Energía eléctrica (Generación, distribución, proyección del crecimiento de la red), Economía y marketing (proyecciones del empleo y desempleo, evolución del índice de precios de productos), demografía (número de habitantes por año, tasa de mortalidad infantil por año), medio ambiente (evolución horaria de niveles de óxido de azufre y de niveles de óxido de nitrógeno en una ciudad, temperatura media mensual, medición diaria del contenido en residuos tóxicos en un río), etc. George E. P. Box [George E. P. Box, 2015]. La Figura 1.1 muestra una serie de tiempo y el comportamiento que tiene, ésta contiene las mediciones de los kW (kilowatt) que son distribuidos por un circuito; las mediciones son hechas de manera horaria.

Generalmente las series de tiempo se encuentran llenas de perturbaciones las cuales impactan directamente en la calidad del pronóstico, es por esté motivo que se aborda en esta tesis la calidad de los datos en una serie de tiempo modelando el error de predicción.

Este capítulo presenta una breve descripción de los objetivos de esta tesis,

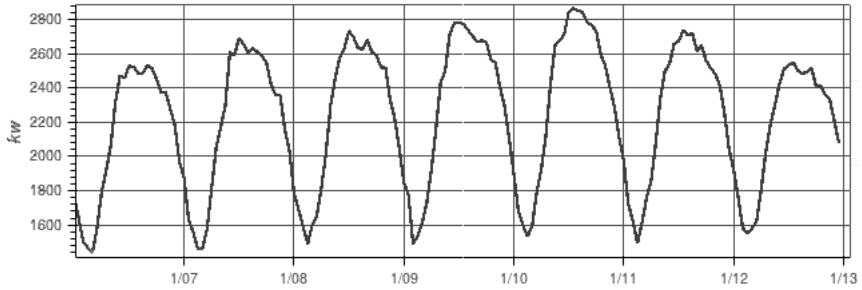


Figura 1.1: Serie de tiempo de la potencia real consumida en un circuito eléctrico de distribución, mediciones horarias en el periodo del 6 al 13 de enero de 2019

la revisión del estado del arte y antecedentes de este trabajo. Además, se presenta la justificación y metodología utilizada para el desarrollo de esta tesis. Finalmente se hace una descripción por capítulos.

1.1. Planteamiento del problema

En el análisis de las series tiempo se usan métodos que permiten extraer información de interés, principalmente se usan para el pronóstico el cual es una predicción de algún evento o eventos futuros. El pronóstico es un problema importante que abarca muchos campos, incluidos los negocios, la industria, el gobierno, la economía, las ciencias ambientales, la medicina, las ciencias sociales, la política y las finanzas. Los problemas de pronóstico a menudo se clasifican como a corto, mediano y largo plazo. Como lo sugiere Neils Bohr, hacer buenas predicciones no siempre es fácil; como ejemplo se pueden citar los siguientes pronósticos malos: “La población es constante en tamaño y seguirá siéndolo hasta el final de la humanidad”, “1930 será un año laboral espléndido (Departamento de trabajo de EE.UU)., pronóstico de año nuevo en 1929, justo antes de la caída del mercado el 29 de octubre”, “Las computadoras se están multiplicando a un ritmo rápido. Para el cambio de siglo, habrá 220,000 en los EE.UU. (Wall Street Journal, 1966)”estas citas son tomadas del libro [Douglas C. Montgomery and Kulahci, 2015].

El pronóstico de una serie de tiempo es afectado directamente por la calidad de los datos, esto se debe a diferentes factores como lo son (datos fal-

tantes, valores atípicos, ruido y caos). Estas características afectan de manera directa los pronósticos, porque estos pueden llegar a ser malos. Después de una revisión no se ha encontró un método que indique de manera aproximada el error de predicción antes de realizar el modelado de pronóstico; motivados por esta situación en esta tesis se aproxima el error de pronóstico de la serie de tiempo extrayendo cuatro características: datos faltantes, valores atípicos, ruido y caos. Una vez extraídas estas características, usar el perceptrón multicapa (MLP MultiLayer Perceptron) para realizar pronóstico y determinar el error SMAPE que es el error porcentual absoluto medio simétrico, el cual es una medida de precisión basada en errores porcentuales. La diferencia absoluta entre el valor real y_i y el valor pronosticado $\hat{y}_i$ se divide entre la mitad de la suma de los valores absolutos del valor real y_i y el valor pronosticado $\hat{y}_i$. El valor de este cálculo se suma para cada punto ajustado i y se divide nuevamente por el número de puntos ajustados N y se multiplica por 100% dado en (1.1).

$$SMAPE = \frac{100}{N}\sum_{i=1}^{N}\frac{|y_i - \hat{y_i}|}{(|y_i| + |\hat{y}_i|)/2}(\%) \tag{1.1}$$

Esta métrica de error expresa la calidad en la predicción. Con las características extraídas y con el error de predicción se crea un regresor con el algoritmo de bosques aleatorios Breiman [Breiman, 2001], la precisión del regresor se calcula con (1.2) donde el error porcentual absoluto medio MAPE, es una medida de la precisión como un porcentaje para problemas de regresión. Donde y_i es el valor real y $\hat{y}_i$ es el valor pronosticado. La diferencia entre y_i y $\hat{y}_i$ se divide por el valor real y_i. El valor absoluto en este cálculo se suma para cada punto en el tiempo i y se divide por el número de puntos ajustados N y se multiplica por 100% esta dado por (1.3).

$$precisionR = (100 - MAPE)\,\% \tag{1.2}$$

$$MAPE = \frac{100}{N}\sum_{i=1}^{N}|\frac{y_i - \hat{y}_i}{y_i}|(\%) \tag{1.3}$$

Esto ayuda a que simplemente extrayendo las características podamos tener una aproximación al error de pronóstico SMAPE y con ello la calidad de los datos de la serie de tiempo; la calidad se calcula como (100-SMAPE)%; si se encuentra entre (90% y 100%) se considera como la mejor calidad de datos

para realizar pronóstico, si se encuentra entre (70% y 90%) se considera como una buena calidad de datos, si se encuentra entre (60% y 70%) se considera como una regular calidad de datos y entre (0% y 60%) indica que la calidad de los datos es mala para realizar pronóstico. Esto ayudará a decidir de manera rápida si realizar o no el modelado de pronóstico.

El problema que resuelve esta tesis se puede definir de la siguiente manera. Sea X una serie de tiempo conformada por los datos $\{x_1, x_2, \ldots, x_N\}$.

La serie de tiempo X es dividida en dos segmentos como en (1.4): un conjunto de entrenamiento X_T que contiene el 70% de la serie de tiempo y un conjunto de validación X_V con el 30% restante.

$$X = X_T | X_V \tag{1.4}$$

Las longitudes de los conjuntos de entrenamiento y validación están dadas por (1.5) y (1.5), respectivamente.

$$n_T = |X_T| \tag{1.5}$$

$$n_V = |X_V| \tag{1.6}$$

Se extrae un conjunto de características de la serie de tiempo (1.7).

$$Car(X) = \{C_1, C_2, \ldots, C_k\} \tag{1.7}$$

Para la extracción de cada una de las características C_k se usa la función F_k en la serie de tiempo X, como en (1.8).

$$C_k = F_k(X) \tag{1.8}$$

Se define T_m como el conjunto de características C_k obtenidas aplicando la función F_k con (1.8).

$$T_m = \{(C_1, C_2, \ldots, C_k) | C_k = F_k(X)\} \tag{1.9}$$

Se genera un predictor neuronal $P(X_T, n_V)$ el cual recibe el conjunto de entrenamiento y el tamaño de la validación y regresa las predicciones en números reales con n_V dimensiones $\mathbb{R}^{n_V}$, este predictor se encuentra definido por (1.10).

$$P(X_T, n_V) : X \to \mathbb{R}^{n_V} \tag{1.10}$$

La medida del error de pronóstico, $Err(X)$, está definida por (1.11)

$$Err(X) = E(X_V, P(X_T, n_V)) \tag{1.11}$$

Se genera un ejemplo para cada serie de tiempo X, dado por $(Car(X), Err(X))$. Con esos ejemplos se entrena un regresor, con el conjunto de características T_m se estima el error $\mathbb{R}^+$ con (1.12)

$$R : T_m \to \mathbb{R}^+ \tag{1.12}$$

Dada una serie de tiempo Y se calculan sus características $Car(Y)$. El error estimado usando un predictor del mismo tipo, estará dado por (1.13).

$$Err(Y) = R(Car(Y)) \tag{1.13}$$

1.2. Antecedentes

Después de una revisión no se encontró una investigación previa a la que se propone en esta tesis. Uno de los pocos trabajos relacionados que se encontró fue una investigación sobre el método de control de calidad basado en datos de series de tiempo hidrológicas Zhao et al. [Zhao et al., 2018], el cual estudia los métodos de control de calidad de los datos y la consistencia de los datos.

Los datos faltantes se producen cuando no se almacena ningún valor para la serie de tiempo en una observación, estos generalmente pueden ser causados por problemas de grabación o transmisión, las investigaciones que se encontraron sobre este tema son encontrar los datos faltantes, después hacen un preprocesamiento y usan la imputación de datos que es el método de rellenar con valores que se obtienen mediante un proceso de estimación [Douglas C. Montgomery and Kulahci, 2015].

Dentro de las series de tiempo se encuentran datos atípicos (outliers), son observaciones que numéricamente son distantes del resto de los datos, los cuales afectan a la calidad del pronóstico, para la detección de estos valores atípicos se cuenta con el método Local Outlier Factor (LOF) Breunig et al. [Breunig et al., 2000].

En el estudio experimental hecho en Campos et al. [Campos et al., 2016], se abordó un desafío constante en la detección de valores atípicos no supervisados: la evaluación de algoritmos en términos de efectividad. Se discutió

la notoria falta de conjuntos de datos de referencia comúnmente aceptados y bien entendidos. También se elaboraron estudios sobre las medidas de evaluación utilizadas comúnmente, sus fortalezas y debilidades, y como se pueden usar varias medidas en combinación para proporcionar información sobre el desempeño relativo de los métodos.

En el desarrollo de esta tesis se realizó un extenso análisis experimental de un conjunto representativo de métodos de detección de valores atípicos sin supervisión, tanto clásicos como recientes, en una gran colección de conjuntos de datos. Los documentos que proponen un método novedoso a menudo justifican su rendimiento en función de una medida de evaluación específica, en pocos conjuntos de datos y para pocos parámetros de configuración. Al utilizar una colección diversa de conjuntos de datos, varias medidas de evaluación y una amplia gama de ajustes de parámetros, argumentamos aquí que, por lo general, no tiene sentido ni justificación establecer el comportamiento superior de cualquier método para el caso general.

En la investigación Flores et al. [Flores et al., 2016] se hace la comparación de las técnicas de pronóstico de series de tiempo con respecto a la tolerancia al ruido. Presentan un análisis y una comparación de cómo la presencia de ruido afecta las diferentes técnicas de pronóstico se menciona que el ruido es omnipresente en la producción de series de tiempo; no se puede asumir que los datos de origen están limpios, la mayoría de las veces están contaminados con ruido. Las técnicas de pronóstico incluidas en esta comparación son Nearest Neighbors, Artificial Neural Networks, ARIMA, Fuzzy Neural Networks y Nearest Neighbors combinados con Differential Evolution. Entre todos ellos, la técnica que funciona mejor y se ve menos afectada por el ruido es Vecinos más Cercanos combinados con Evolución Diferencial.

1.3. Objetivos

1.3.1. Objetivo general

El objetivo general que se trabaja en esta tesis es determinar la calidad de los datos de una serie de tiempo, modelando el error de predicción.

1.3.2. Objetivos particulares

Los objetivos particulares de esta tesis son:

- Generar 10 series de tiempo sintéticas y perturbarlas con varios niveles de datos faltantes, valores atípicos y ruido. En total se generarán 2,160 series de tiempo.
- Extraer las cuatro características (Datos Faltantes, valores atípicos, ruido y caos) y guardarlas en una tabla de datos llamada características y error .
- Diseñar modelos de perceptrones multicapa que son un tipo de red neuronal para hacer pronóstico de cada una de las series de tiempo sintéticas generadas, obtener el error de las predicciones y guardarlos en la tabla de datos llamada características y error.
- Con la tabla de datos características y error hacer un regresor del error, para que este regresor se ajuste a los datos y de una aproximación a la precisión del pronóstico sin necesidad de hacer todo el modelado y después ver el error.

1.4. Justificación

El problema de saber la calidad de los datos de una serie de tiempo es un tema que se plantea al momento de preguntar ¿Cual es la precisión del pronóstico que se obtendrá con sus datos? La mayoría de las veces se pide un compromiso de confiabilidad en el pronóstico de una serie de tiempo. Después de una revisión no se encontró un método que indique la calidad de los datos de una serie de tiempo. Otra razón que motivó hacer esta investigación es evitar realizar todo el proceso de modelado y de pronóstico que en cuestión de tiempo llevaría entre un día hasta semanas. Con este método que se propone se podrá obtener de manera automática y rápida una aproximación del error de predicción. Esta aproximación depende directamente de la calidad de los datos de la serie de tiempo.

1.5. Descripción de capítulos

El contenido de esta tesis se encuentra organizado en seis capítulos. El Capítulo 2 presenta técnicas para la extracción de características de una serie de tiempo. El Capítulo 3 describe los perceptrones multicapa, así como los

modelos que se usaron para realizar el pronóstico. El Capítulo 4 describe el regresor de bosques aleatorios y como se ajusto el número de árboles de decisión. El Capítulo 5 muestra los resultados obtenidos. El Capítulo 6 menciona las conclusiones y trabajos futuros.

Capítulo 2

Extracción de características de una serie de tiempo

La extracción de características es uno de los pasos preliminares en los algoritmos de aprendizaje automático, para identificar atributos relevantes fuertes y débiles. Las características usadas en esta tesis para medir la calidad de una serie de tiempo son: remoción de datos faltantes, valores atípicos y ruido; el caos no se agregó porque es parte de la naturaleza de las series de tiempo. Para la extracción de estas características se generaron series de tiempo sintéticas, una vez generadas las series de tiempo y perturbadas se determina un estimador de estas características.

2.1. Series de tiempo sintéticas

Las series de tiempo sintéticas se usan porque no presentan perturbaciones, se pueden generar y agregar diferentes tipos de perturbaciones, son llamadas sintéticas porque los datos no son obtenidos por la medición directa de un proceso real. Las series de tiempo que se utilizaron en esta tesis aparecen en la Tabla 2.1; la primera columna series de tiempo contiene el nombre de las series de tiempo que se utilizaron. La segunda columna sistema de ecuaciones muestra los sistemas que modelan las series de tiempo caóticas y la función seno. La tercera columna parámetros usuales son los valores con los cuales fueron generadas las series de tiempo. La cuarta columna condiciones iniciales son los valores iniciales de las variables que forman los sistemas de ecuaciones. De las 10 series de tiempo que se muestran 9 son caóticas y

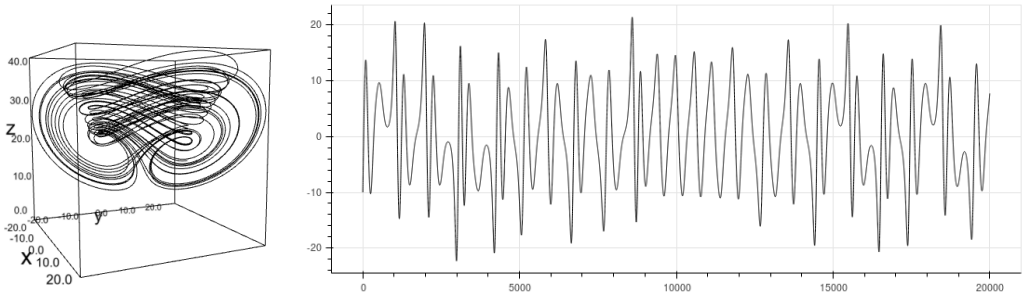
una función seno.

Tabla 2.1: Tabla de las series de tiempo sintéticas.

Serie de tiempo	Sistema de ecuaciones	Parámetros usuales	Condiciones iniciales
Sistema Chen	$dx/dt = a(y-x)$ $dy/dt = (c-a)x - xz - cy$ $dz/dt = xy - bz$	$a = 35$ $b = 3$ $c = 28$	$x_0 = -10$ $y_0 = 0$ $z_0 = 37$
Oscilador Duffing	$dx/dt = y$ $dy/dt = -x^3 + x - by + A\ sin\omega t$	$b = 0.25$ $A = 0.4$ $\omega = 1$	$x_0 = 0.2$ $y_0 = 0$ $t_0 = 0$
Atractor cíclico simétrico de Halvorsen	$dx/dt = -ax - 4y - 4z - y^2$ $dy/dt = -ay - 4z - 4x - z^2$ $dz/dt = -az - 4x - 4y - x^2$	$a = 1.27$	$x_0 = -5$ $y_0 = 0$ $z_0 = 0$
Atractor de Lorenz	$dx/dt = \sigma(y-x)$ $dy/dt = -xz + rx - y$ $dz/dt = xy - bz$	$\sigma = 10$ $r = 28$ $b = \frac{8}{3}$	$x_0 = 0$ $y_0 = -0.01$ $z_0 = 9$
Atractor de Rössler	$dx/dt = -y - z$ $dy/dt = x + ay$ $dz/dt = b + z(x-c)$	$a = 0.2$ $b = 0.2$ $c = 5.7$	$x_0 = -9$ $y_0 = 0$ $z_0 = 0$
Atractor de Rucklidge	$dx/dt = -\kappa x + \lambda y - yz$ $dy/dt = x$ $dz/dt = -z + y^2$	$\kappa = 2$ $\lambda = 6.7$	$x_0 = 1$ $y_0 = 0$ $z_0 = 4.5$
Oscilador Shawn-van der Pol	$dx/dt = y + A\ sin\omega t$ $dy/dt = -x + b(1-x^2)y$	$b = 1$ $A = 1$ $\omega = 2$	$x_0 = 1.3$ $y_0 = 0$ $t_0 = 0$
Flujo cúbico más simple	$dx/dt = y$ $dy/dt = z$ $dz/dt = -az + xy^2 - x$	$a = 2.028$	$x_0 = 0$ $y_0 = 0.96$ $z_0 = 0$
Flujo lineal por partes más simple	$dx/dt = y$ $dy/dt = z$ $dz/dt = -az - y + \lvert x\rvert - 1$	$a = 0.6$	$x_0 = 0$ $y_0 = -0.7$ $z_0 = 0$
Seno	$sin(2\pi ft)$	$f = 1000$	$t_0 = 0$

La primer serie de tiempo que aparece es el sistema Chen, Chen y Ueta [Chen and Ueta, 1999]. La Figura 2.1(a) muestra el comportamiento en 3D del sistema de Chen y la Figura 2.1(b) muestra la serie de tiempo de la

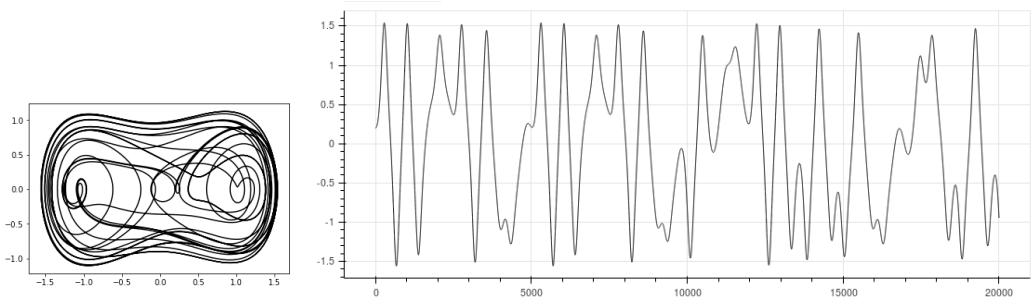
variable X.



(a) Trayectoria en 3D (b) Serie de tiempo de la variable x

Figura 2.1: Sistema Chen.

La segunda serie de tiempo es el oscilador de Duffing (llamado así por Georg Duffing), un famoso sistema dinámico no lineal. Desde el siglo pasado, los sistemas dinámicos no lineales de tipo Duffing se han investigado en muchos campos mediante numerosas investigaciones. Este sistema es capaz de exhibir un comportamiento caótico. La Figura 2.2(a) muestra el comportamiento en 2D y la Figura 2.2(b) muestra la serie de tiempo de la variable X.

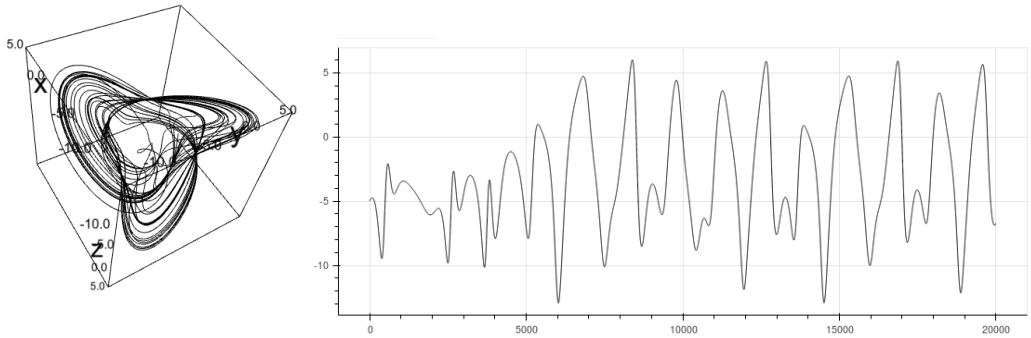


(a) Trayectoria en 2D (b) Serie de tiempo de la variable x

Figura 2.2: Oscilador Duffing.

La tercera serie de tiempo es del atractor cíclico simétrico de Halvorsen Azar y Vaidyanathan [Azar and Vaidyanathan, 2016], que es simétrico con respecto al intercambio cíclico de x, y y z. Halvorsen mostró que es caótico

cuando $a = 1.27$. La Figura 2.3(a) muestra el comportamiento en 3D y la Figura 2.3(b) muestra la serie de tiempo de la variable X.

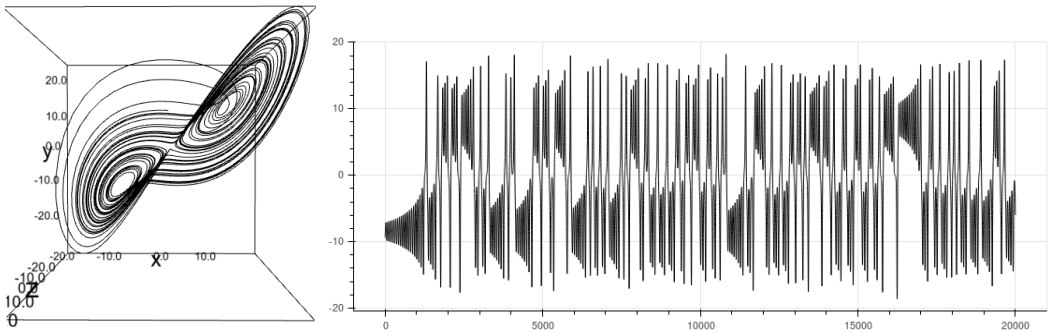


(a) Trayectoria del sistema en espacio de fase

(b) Serie de tiempo de la variable x

Figura 2.3: El atractor cíclico simétrico de Halvorsen.

La cuarta serie de tiempo es el atractor de Lorenz; este es un sistema de ecuaciones diferenciales ordinarias estudiado por Edward Lorenz. Las soluciones a este sistema son caóticas para ciertos valores iniciales; el "efecto mariposa"se deriva de las implicaciones reales del atractor de Lorenz Sprott [Sprott, 2003]. La Figura 2.4(a) muestra el comportamiento en 3D y la Figura 2.4(b) muestra la serie de tiempo de la variable X.



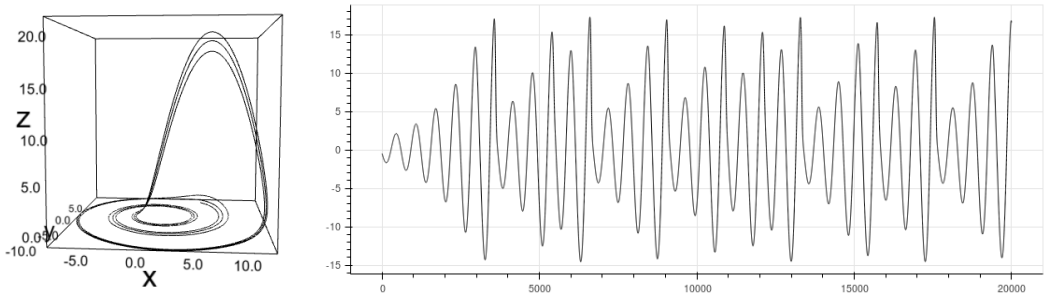
(a) Trayectoria en 3D

(b) Serie de tiempo de la variable x

Figura 2.4: Atractor de Lorenz

La quinta serie de tiempo es el atractor de Rössler es un sistema de tres

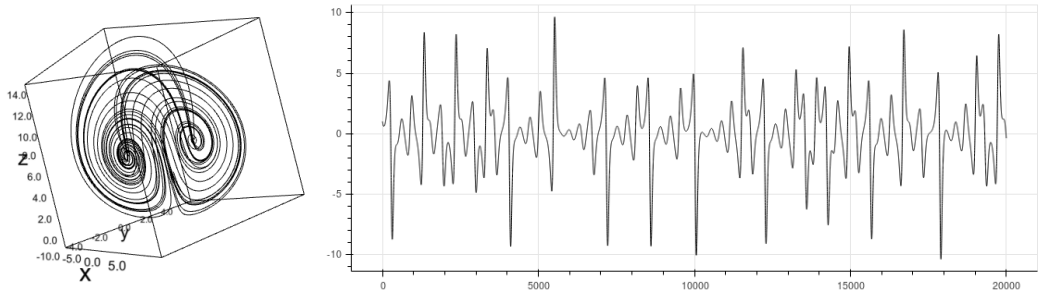
ecuaciones diferenciales ordinarias no lineales estudiadas originalmente por Otto Rössler. Estas ecuaciones diferenciales definen un sistema dinámico de tiempo continuo que exhibe dinámicas caóticas Rössler [Rössler, 1976]. La Figura 2.5(a) muestra el comportamiento en 3D y la Figura 2.5(b) muestra la serie de tiempo de la variable X de este atractor.



(a) Trayectoria en 3D (b) Serie de tiempo de la variable x

Figura 2.5: Atractor de Rössler.

La sexta serie de tiempo, el atractor Rucklidge, también es caótico Sprott [Sprott, 2003]. La Figura 2.6(a) muestra el comportamiento en 3D y la Figura 2.6(b) muestra la serie de tiempo de la variable X de este atractor.

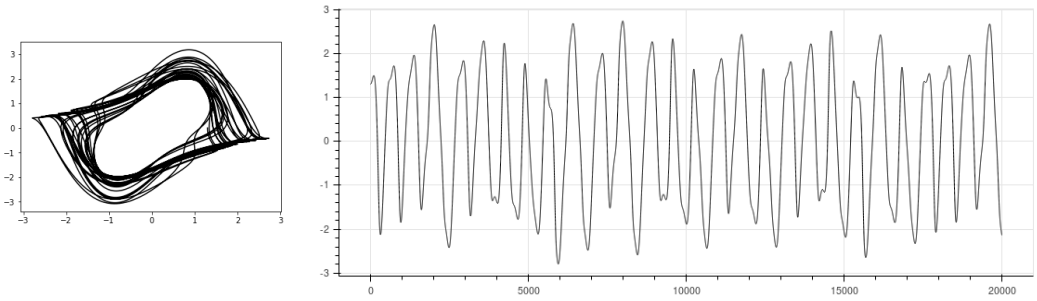


(a) Trayectoria en 3D (b) Serie de tiempo de la variable x

Figura 2.6: Atractor de Rucklidge.

La séptima serie de tiempo es el oscilador de Van der Pol, descrito por el ingeniero y físico Balthasar Van der Pol, encontró oscilaciones estables a las que llamó oscilaciones de relajación, conocidas en la actualidad como ciclos límite. Este sistema fue uno de los primeros descubrimientos experimentales

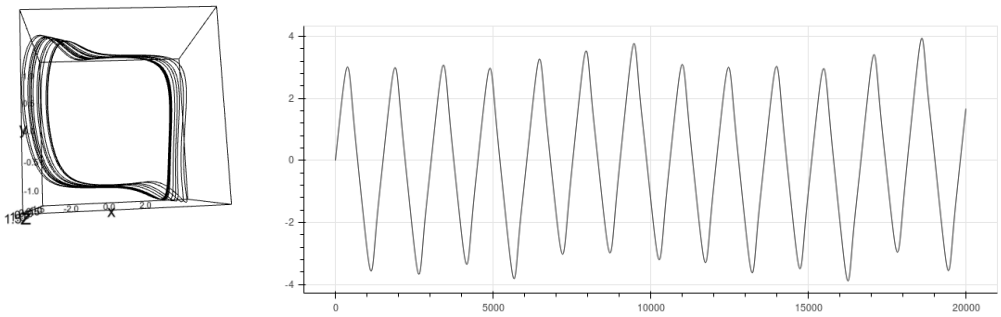
de la teoría del caos. La Figura 2.7(a) se muestra el comportamiento en 2D y la Figura 2.7(b) muestra la serie de tiempo de la variable X.



(a) Trayectoria en 2D (b) Serie de tiempo de la variable x

Figura 2.7: Oscilador Shawn-van der Pol.

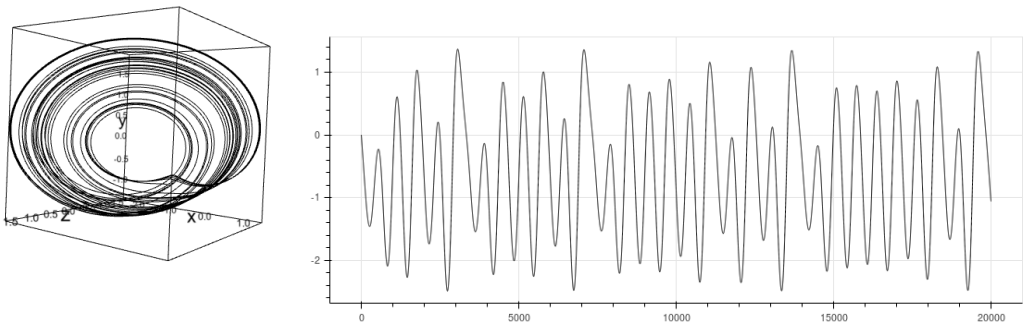
La octava serie de tiempo, flujo cúbico, más simple, Sprott encontró una variedad de funciones caóticas con no linealidades cúbicas Sprott y J Linz [Sprott and J Linz, 2000], la Figura 2.8(a) muestra el comportamiento en 3D y la Figura 2.8(b) muestra la serie de tiempo de la variable X.



(a) Trayectoria en 3D (b) Serie de tiempo de la variable x

Figura 2.8: Flujo cúbico más simple.

La novena serie de tiempo, representa el flujo lineal por partes más simple Sprott y J Linz [Sprott and J Linz, 2000] la Figura 2.9(a) muestra el comportamiento en 3D y la Figura 2.9(b) muestra la serie de tiempo de la variable X.



(a) Trayectoria en 3D (b) Serie de tiempo de la variable x

Figura 2.9: Flujo lineal por partes más simple.

La décima serie de tiempo, de una función seno Figura 2.10. Esta serie de tiempo fue incluida como referencia en los experimentos, para visualizar de una manera fácil el comportamiento al momento de perturbar las serie de tiempo y extraer las características.

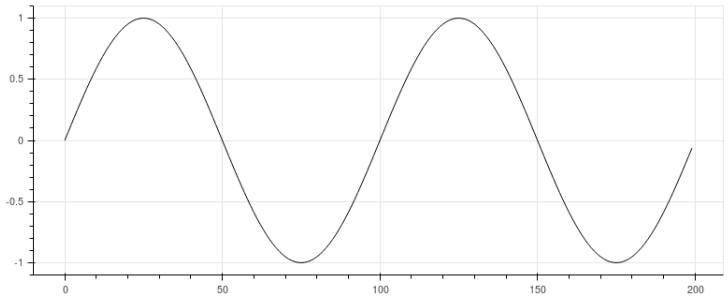


Figura 2.10: Seno.

Todas las series de tiempo se perturbaron por ese motivo se diseño una codificación para referirse cuando una serie se encuentra perturbada con que porcentaje o nivel. La Tabla 2.2, la columna datos faltantes % indica el porcentaje de perturbación de esta característica, la columna valores atípicos % indica el porcentaje de perturbación con esa característica, y la columna ruido SNR dB indica la perturbación en niveles de ruido. La codificación para las diferentes perturbaciones que se tienen es la siguiente: f_a_r donde f indica el porcentaje de datos faltantes, a indica el porcentaje de valores atípicos y

r indica el nivel de ruido. Por lo tanto esa misma codificación indica que la serie de tiempo no se va a perturbar pero es importante generarla para tenerla como referente dentro del análisis. Las codificaciones (0_0_5, 0_0_10, 0_0_15, 0_0_20, 0_0_25) indican que se perturbaron unicamente con niveles de ruido 5, 10, 15, 20, 25 respectivamente. La codificación 0_5_0, indican que se perturbó un 5% de los datos solamente con valores atípicos. La codificación 0_5_5, indica que se perturbó la serie con 5% de valores atípicos y con un nivel de ruido de $5dB$ y así sucesivamente donde los $\vdots$ indican que se generaron todas las perturbaciones hasta terminar con la última codificación 25_25_25, donde se perturba con 25% de datos faltantes, 25% de valores atípicos y un nivel de ruido de $25dB$

Tabla 2.2: Porcentajes y niveles de perturbaciones.

Datos Faltantes %						Valores Atípicos %						Ruido SNR dB					
0	5	10	15	20	25	0	5	10	15	20	25	0	5	10	15	20	25
*						*						*					
*						*							*				
*						*								*			
*						*									*		
*						*										*	
*						*											*
*							*					*					
*							*						*				
*							*							*			
*							*								*		
*							*									*	
*							*										*
*								*				*					
$\vdots$						$\vdots$						$\vdots$					$\vdots$
					*						*						*

El resto del capítulo describe las características que esta tesis propone como importantes para determinar la calidad de la información contenida en una serie de tiempo.

2.2. Datos faltantes

En una serie de tiempo es casi imposible tener un conjunto de datos completo; esto es, donde todas las observaciones se encuentren registradas. Los datos faltantes es un problema para toda serie de tiempo, ya que una de las premisas para considerar una secuencia de datos como una serie de tiempo es que necesita tener un orden cronológico como pueden ser datos registrados en resoluciones (minutales, horarias, diarias, semanales, mensuales, bimestrales, etc.). Si faltan datos en la serie de tiempo se convierte en el primer obstáculo para el diseño de modelos de predicción. Para esto es importante detectar los valores faltantes.

Los valores faltantes son aquellos datos que no aparecen con un valor numérico dentro de la serie de tiempo. La Figura 2.11 muestra la ausencia de valores; esta ausencia puede ser causada por diferentes circunstancias como lo pueden ser: El sistema se encuentra en mantenimiento, por tal motivo se encuentra apagado y no genera datos, mal funcionamiento en los equipos de adquisición de datos, errores en la comunicación TCP/IP, adquisición de datos manualmente y se perdió la bitácora o se olvidó tomar las mediciones, etc. Por lo tanto si la serie de tiempo contiene valores faltantes no se podrá realizar el pronóstico porque los pronosticadores son sensibles a valores faltantes.

Todos los algoritmos que se usaron e implementaron en esta tesis fueron implementados en Python 3.6.

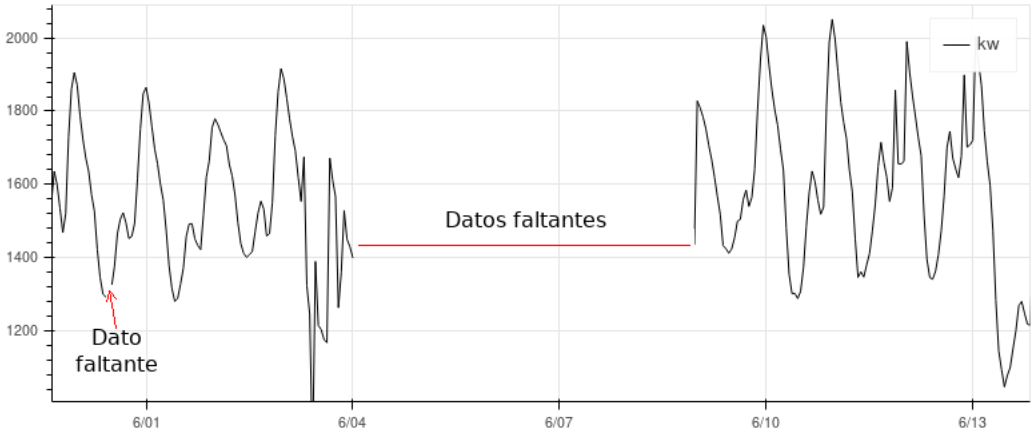


Figura 2.11: Datos faltantes

2.2.1. Detección de datos faltantes

El Algoritmo 1 llamado *datosFaltantes* encuentra el porcentaje de datos faltantes en la serie de tiempo. Recibe como argumento una serie de tiempo como un *DataFrame* de pandas McKinney et al. [McKinney et al., 2010], se obtiene el tamaño de la serie de tiempo y se obtienen los índices de los valores faltantes marcados como NaN. Una vez que se recorrió toda la serie de tiempo y guardados los índices estos se cuentan, y el resultado se divide entre el tamaño total de la serie de tiempo. Al final el algoritmo regresa el porcentaje de datos faltantes.

Algoritmo 1: Detección de datos faltantes

```
1 datosFaltantes(st)
2   tamano ← len(st)
3   faltantes ← st.isnull().sum() / tamano
4   return 100 * faltantes
```

2.2.2. Remoción de datos faltantes

La remoción de datos se hace directamente en las series de tiempo sintéticas que se utilizaron para realizar los experimentos. Por remoción se entiende la eliminación de datos de una serie de tiempo. Esta eliminación está dada por el porcentaje de datos faltantes que se desea para la perturbación de la serie de tiempo. En esta tesis se usaron porcentajes de (0, 5, 10, 15, 20, 25) para los experimentos. El Algoritmo 2 *eliminarDatos* lleva acabo la remoción de datos faltantes en una serie de tiempo. Recibe dos argumentos el primero *st* que es la serie de tiempo que se desea perturbar y el segundo argumento es el porcentaje con el cual se desea perturbar. Se calcula cuantos elementos se van a eliminar basado en el porcentaje. Posteriormente de manera aleatoria se eligen los indices que se van a eliminar. Una vez que se tienen los índices se marcan con el valor NaN el cual indica dato faltante.

Al finalizar el algoritmo regresa la serie de tiempo *st* perturbada con el porcentaje de datos faltantes que se indicó.

Algoritmo 2: Remoción de datos faltantes

```
1  eliminarDatos(st, porcentaje)
2    indices ← []
3    tamano ← len(st)
4    elementos ← round(porcentaje/100 * tamano)
5    for i en range(elementos) do
6      k ← 0
7      while k == 0 do
8        r ← randint(tamano)
9        ifr not en indices
10         indices.append(r)
11         st[r] ← Nan
12         k ← 1
13   return st
```

2.2.3. Imputación

La imputación de datos es el proceso de corregir los datos faltantes o reemplazar los valores atípicos con un proceso de estimación. La imputación reemplaza los valores faltantes o erróneos con un valor "probable"basado en otra información disponible. Esto permite que el análisis funcione con técnicas estadísticas diseñadas para manejar los conjuntos de datos completos Douglas C. Montgomery and Kulahci, [Douglas C. Montgomery and Kulahci, 2015]. En nuestro caso vamos a usar la imputación solo para tener los datos completos para esto se optó por el método última observación arrastrada LOCF (Last Observation Carried Forward), lo que hace esté método es rellenar el valor faltante con el valor de la última observación $x_t = x_{t-1}$ y en caso de que el dato faltante sea el primero de la serie se rellena con la media $x_0 = media(X)$ de la señal, se utiliza la media para no agregar información que no se conoce y preservar la misma.

El Algoritmo 3 *imputar* rellena los datos faltantes en la serie de tiempo, funciona de la siguiente manera: recibe una serie de tiempo st. Se obtienen los índices donde se encuentran los valores faltantes. Cuando se detecta que $ts[0] = NaN$ se le asigna el valor de la media de la serie de tiempo, y para cualquier indice diferente de cero, se rellena el valor faltante con el valor de la última observación se regresa la serie de tiempo ya rellenada y completa.

Algoritmo 3: Imputación (Rellenar datos faltantes)

```
1 imputar(st)
2   indicesFaltantes ←st.isnull().nonzero()[0]
3   for i en indicesFaltantes do
4     if i > 0
5       st[i] ← st[i − 1]
6     else
7       st[i] ← st.mean()
8   return st
```

2.3. Valores atípicos

Al igual que los valores faltantes los valores atípicos impactan directamente en el pronóstico, por tal motivo es muy importante detectarlos. Los datos de las series de tiempo se pueden ver afectados por eventos aislados, perturbaciones o errores que crean inconsistencias y dan lugar a datos inusuales que no son consistentes con el comportamiento general de la serie de tiempo. Estos datos inusuales son llamados valores atípicos. Estos pueden ser el resultado de eventos externos inusuales, como por ejemplo cambios repentinos en un sistema físico, económicos, eventos climáticos inusuales, etc., o simplemente debido a un registro o errores graves en la medición George E. P. Box [George E. P. Box, 2015].

Definición de Hawkins [Hawkins, 1980] "Un valor atípico es una observación que se desvía tanto de las otras observaciones como para despertar sospechas de que fue generado por un mecanismo diferente". La eficiencia y la eficacia de la detección de valores atípicos son de gran interés. Existen dos tipos de valores atípicos los globales y los locales, motivo por el cual se tienen diferentes métodos para detectarlos. A continuación se muestra que la detección de valores atípicos se realizó en 2 etapas la primera usa un método para la detección de valores atípicos globales y la segunda usa un método para la detección de valores atípicos locales.

2.3.1. Valores atípicos globales

En una serie de tiempo un dato se considera como valor atípico global si se desvía significativamente del resto del conjunto de datos Han et al.

[Han et al., 2012]. Los valores atípicos globales a veces se llaman anomalías puntuales, y son el tipo más simple de valores atípicos la Figura 2.13 muestra marcados con cruces de color rojo a dos valores atípicos globales.

Existen diferentes métodos para detección de valores atípicos globales, Z-score, Z-score modificado, rango intercuartil rango intercuartil (interquartile range) (IQR) (interquartile range), regla de tres sigma (three-sigma rule), etc. Han et al. [Han et al., 2012]. Los valores atípicos son datos que tienen una baja probabilidad de ser generados por la distribución general (por ejemplo, desviarse más de 3 veces la desviación estándar de la media), el método que mejor detecta los valores atípicos globales es la regla de tres sigma.

En estadística la regla de la tres sigma, es utilizada para recordar el porcentaje de valores que se encuentran dentro de una banda alrededor de la media en una distribución normal. Se puede expresar de la siguiente manera, donde x_k es una observación de una variable aleatoria normalmente distribuida, μ es la media de la distribuciónx y σ es la desviación estándar. Es una heurística convencional donde se considera que casi todos los valores se encuentran dentro de tres desviaciones estándar de la media, por lo tanto en la práctica es útil tratar el 99.7 % de probabilidad Pr como certeza, en (2.1) se muestra la probabilidad de esta regla.

$$Pr(\mu - 3\sigma \leq x_k \leq \mu + 3\sigma) \approx 0.9973 \tag{2.1}$$

La utilidad de esta heurística depende significativamente de la pregunta bajo consideración. La regla de tres sigma, que establece que incluso para las variables no distribuidas normalmente, al menos el 88.8 % de los casos deben estar dentro de los intervalos de tres sigma calculados adecuadamente.

El Algoritmo 4 *deteccionGlobal*, implementa la regla de tres sigma, usada para detectar los valores atípicos globales. Esta regla funciona de la siguiente manera: recibe como argumento la st, después obtiene la μ y la σ de la serie de tiempo, calcula el límite superior $limSup$ con (2.2) y el limite inferior $limInf$ con (2.3).

$$limSup = \mu + 3\sigma \tag{2.2}$$

$$limInf = \mu - 3\sigma \tag{2.3}$$

Una vez calculados los limites se revisan los datos de la serie de tiempo, y se obtienen los índices de los valores que son mayores al límite superior y

los indices de los valores que son menores al límite inferior, una vez obtenidos se devuelve la lista de índices con valores atípicos globales y termina el algoritmo.

Algoritmo 4: Detección de valores atípicos globales

```
1  deteccionGlobal(st)
2    indices ← []
3    mu ← media(st)
4    sigma ← desEst(st)
5    limSup ← mu + (3 * sigma)
6    limInf ← mu − (3 * sigma)
7    for i en range(len(st)) do
8      if st[i] > limSup or st[i] < limInf
9        indices.append(i)
10   return indices
```

El Algoritmo 4 *deteccionGlobal* regresa una lista con los índices donde se detectaron valores atípicos globales, esta lista se utilizará en la detección de valores atípicos locales para evitar que se cuente un valor atípico 2 veces.

2.3.2. Valores atípicos locales

Los valores atípicos locales son más difíciles de detectar que los globales, ya que para encontrarlos es necesario conocer la naturaleza de la serie de tiempo por ejemplo: "La temperatura actual es de $27°C$. ¿Es un valor atípico?"Depende, por ejemplo, de la hora y la ubicación. Si es el mes de diciembre por la noche, en Morelia, sí es un valor atípico. Si es el mes de mayo por el día en Morelia, entonces es un caso normal. A diferencia de la detección de valores atípicos globales, en este caso, para decir si el valor de la temperatura de hoy es un valor atípico o no, depende del contexto: la fecha, la hora, la ubicación y posiblemente algunos otros factores Han et al.[Han et al., 2012].

Existe una investigación muy detallada acerca de los métodos de detección de valores atípicos titulada: En la evaluación de la detección de valores atípicos no supervisados: medidas, conjuntos de datos y un estudio empírico Campos et al. [Campos et al., 2016]. En ese trabajo de investigación ponen a prueba doce métodos: kNN (k nearest neighbors), kNNW (kNN-Weight), LOF (local outlier factor), SimplifiedLOF, LoOP (Local Outlier Probabilities),

LDOF (Local Distance-based Outlier Factor), ODIN (Outlier Detection using Indegree Number), KDEOS (Kernel Density Estimation Outlier Score), COF (Connectivity-based Outlier Factor), FastABOD (Fast Angle-Based Outlier Detection), LDF (Local Density Factor), INFLO (Influenced Outlierness). Se analizó la efectividad de la detección de valores atípicos; se usó una colección de conjuntos de datos que se había usado para la evaluación de los métodos, y el resultado que arrojó ese estudio fue que en general el método LOF tuvo un mejor rendimiento Campos et al. [Campos et al., 2016]. Por esa razón en esta tesis se ha usado el método LOF Breunig et al. [Breunig et al., 2000] para la detección de valores atípicos locales.

El método factor de valor atípico local (LOF), compara los valores atípicos con sus vecindarios locales, en lugar de la distribución global de datos. Es una detección basada en la densidad alrededor de un valor atípico, por lo tanto es significativamente diferente la densidad alrededor de sus vecinos. (LOF) utiliza la densidad relativa de un punto contra sus vecinos como indicador del grado de que los objetos son valores atípicos. Este método funciona de la siguiente manera:

1. Calcula todas las distancias entre cada dos puntos de datos $d(p, o)$, se pueden usar diferentes cálculos de distancias como lo son; distancia Manhattan, distancia euclidiana entre otras. En esta tesis se utiliza la distancia euclidiana entre dos puntos $d_E(p, o)$ dada en (2.4) donde el punto $p = (p_1, p_2)$ y el punto $o = (o_1, o_2)$.

$$d_E(p, o) = \sqrt{(p_1 - o_1)^2 + (p_2 - o_2)^2} \tag{2.4}$$

2. Se calcula la distancia entre p y su k-ésimo vecino $kDistancia(p)$. Por ejemplo si $k = 3$ significa que se debe encontrar el tercer vecino más cercano a cada punto. Por ejemplo en la Figura 2.12 con $k = 3$ el k-ésimo vecino más cercano a p es el tercer vecino que tiene la distancia máxima d_{max}.

3. Cálculo de vecinos más cercanos, encuentre los k vecinos, por ejemplo $k = 3$ indica que se van a encontrar los tres vecinos más cercanos o que pertenezcan al conjunto de puntos D de la serie de tiempo, además se verifica que la distancia del punto p a o sea menor o igual a la distancia de su k-ésimo vecino, este proceso se hace para cada punto p con (2.5).

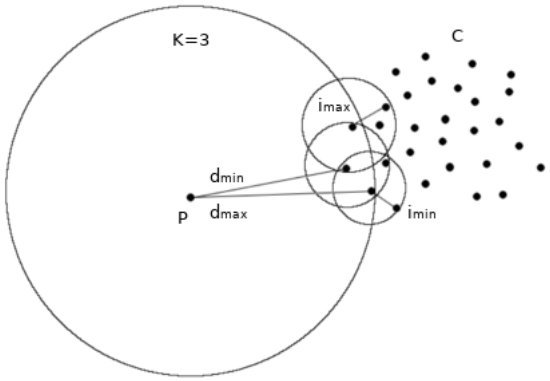


Figura 2.12: Calculo de LOF

La Figura 2.12 muestra encerrados en un círculo los k vecinos para el punto p.

$$N_k(p) = \{o | o \in D, d(p,o) \leq kDistancia(p)\} \tag{2.5}$$

4. Calcula la distancia de alcance del punto p con respecto al punto o; primero se obtiene la distancia calculada entre el punto p y el k-ésimo vecino, después se obtiene la distancia calculada entre el punto p y el punto o, una vez obtenidas estas distancias se comparan y se toma la mayor la cual será la distancia de alcance del punto p con respecto al punto o. Ésta distancia de alcance se define por (2.6).

$$distanciaAlcance_k(p,o) = max\{kDistancia(p), d(p,o)\} \tag{2.6}$$

5. Calcula la densidad de accesibilidad local lrd (local reachability density) con (2.7); donde $|N_k(p)|$ es el número de vecinos más cercanos que se encontraron, no necesariamente el número de vecinos cercanos son igual a k porque existe la posibilidad que vecinos se encuentren a la misma distancia. El denominador suma las distancias de alcance del

punto p con respecto a cada vecino encontrado o.

$$lrd_k(p) = \frac{|N_k(p)|}{\sum_{o \in N_k(p)} distanciaAlcance_k(p, o)} \tag{2.7}$$

6. Calcula el factor de valor atípico local LOF de un punto p con (2.8). Donde $lrd_k(o)$ calcula la densidad de accesibilidad local para cada vecino o del punto p, y es dividida entre $lrd_k(p)$ la densidad de accesibilidad local de p, se hace la sumatoria para cada vecino o que pertenezca al conjunto de vecinos cercanos de p, y se divide entre el número de vecinos $|N_k(p)|$.

$$LOF_k(p) = \frac{\sum_{o \in N_k(p)} \frac{lrd_k(o)}{lrd_k(p)}}{|N_k(p)|} \tag{2.8}$$

Una vez calculados todos los LOF, si $LOF(k) \approx 1$ (densidad similar a los vecinos) se etiqueta como un valor normal, si $LOF(k) < 1$ (mayor densidad que sus vecinos) se etiqueta como valor normal y si $LOF(k) > 1$ (menor densidad que sus vecinos) se etiqueta como valor atípico.

En la implementación de esta tesis se utilizó la biblioteca de scikit-learn la cual incluye el método LOF, Pedregosa et al. [Pedregosa et al., 2011].

El Algoritmo 5, es el método LOF usado para detectar los valores atípicos locales, el cual funciona de la siguiente manera: la función $LOF(st, k)$ recibe 2 argumentos: st es la serie de tiempo y k es el número de vecinos. Al inicio del algoritmo se asigna $-\infty$ a la variable lof. Se calculan todas las distancias euclidianas entre todos los pares de puntos de la serie de tiempo. Se recorre cada punto p que se encuentra en la serie de tiempo, se calcula la distancia entre p y su k-ésimo vecino y también se calculan los vecinos más cercano. Una vez calculadas estas distancias se pasan como argumentos a la función *distanciaAlcance* que calcula la distancia de alcance del punto p con respecto a su vecino o, con el número de vecinos dividido por la sumatoria de las distancias de alcance se calcula la densidad de accesibilidad local lrd. Se calcula el lof temporal con la sumatoria de las densidades de accesibilidad local del punto p y sus vecinos o divididos entre el número de vecinos. Se comparan el lof temporal el lof y se guarda el mayor este proceso se hace para cada punto p de la serie de tiempo. Al final se regresan los valores LOF ordenados de mayor a menor.

Algoritmo 5: Detección de valores atípicos locales

```
1  LOF(st, k)
2    lof ← −∞
3    distancias ← d_E(st)
4    for cada punto p en st do
5      kDist ← kDistancia(p, k)
6      veCerc ← N_k(p, k)
7      lrd ← len(veCerc) / Σ_{o∈veCerc} distanciaAlcance_k(kDist, distancias[p,o])
8      templof ← (Σ_{o∈veCerc} lrd[o,k]/lrd[p,k]) / len(veCerc)
9      lof ← max(lof, templof)
10   return sort(lof)
```

Para ejemplificar el concepto de valores atípicos locales se muestra gráficamente como son detectados. En la Figura 2.13. Se observan 3 valores atípicos locales representados con una linea pequeña de color rojo, que se encuentran dentro de los límites superiores e inferiores de la serie de tiempo y son detectados de manera correcta, porque son valores que se separan de sus vecinos y no continúan con la trayectoria que lleva la serie de tiempo.

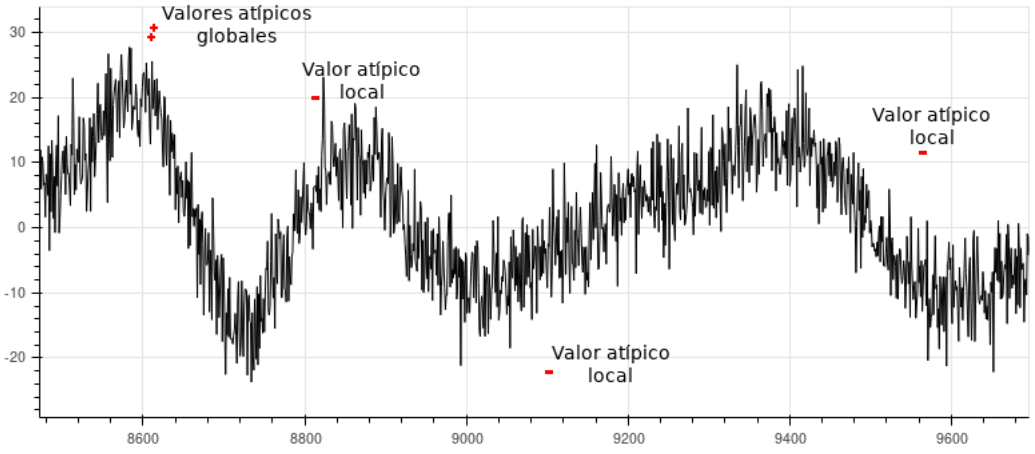


Figura 2.13: Valores atípicos globales y locales

Para obtener el porcentaje total de valores atípicos; se suman los valores atípicos globales y locales. Para llevar acabo esta suma, una vez que termina el Algoritmo 5, la implementación de sklearn neighbors LocalOutlierFactor etiqueta los valores atípicos con un valor de -1 y 1 a los valores normales, se extraen los índices de los elementos marcados como valores atípicos, estos se agregan a la lista de los índices de valores atípicos globales para sumarlos y regresar el porcentaje extraído.

2.3.3. Inclusión de valores atípicos

La inclusión de valores atípicos globales y locales en las series de tiempo sintéticas se refiere a modificar el valor de los datos de la serie de tiempo. Esta modificación se hace en base al porcentaje de valores atípicos que se desean incluir para la perturbación de la serie de tiempo. En esta inclusión de valores atípicos globales y locales, al igual que en la inclusión de datos faltantes se hicieron con los porcentajes de (0, 5, 10, 15, 20, 25). El Algoritmo 6 $agregarValoresAtipicos$ lleva acabo esta inclusión de valores atípicos; función $agregarValoresAtipicos$ recibe dos argumentos: st es la serie de tiempo que se desea perturbar y el porcentaje de la serie de tiempo que se desea perturbar. En la línea 2 se crea una lista de nombre índices el cual se utiliza para guardar los índices de los datos que se van a eliminar, en la línea 3 la variable tamano obtiene el tamaño total de la serie de tiempo, en la línea 4 se calcula cuantos datos van a cambiar su valor, de la línea 5 a la 12 se ejecuta un ciclo for, en la línea 6 se asigna el valor 0 a k, la línea 7 comienza un $while$ que mientras k sea igual a 0 se ejecutara de lo contrario no, este ciclo sirve para que un índice marcado no se pueda marcar nuevamente, en la línea 8 r toma un valor entero de manera aleatoria entre 0 y el tamaño de la st, en la línea 9 se comprueba que el dato con indice r no haya sido perturbado previamente y que además no sea un dato faltante. En la línea 10 se agrega a la lista índices el índice que ha sido perturbado, en la línea 11 se cambia el valor de la st en el índice r por dos veces el valor del dato con esto se agregan tanto valores atípicos locales así como globales. En la línea 12 k toma el valor de 1 y sale del while, en la línea 13 se regresa la serie de tiempo con valores atípicos incluidos.

Algoritmo 6: Inclusión de valores atípicos

```
1  agregarValoresAtipicos(st, porcentaje)
2    indices ← []
3    tamano ← len(st)
4    elementos ← round(porcentaje/100 * tamano)
5    for i en range(elementos) do
6      k ← 0
7      while k == 0 do
8        r ← randint(tamano)
9        if r not en indices and st[r] != Nan
10         indices.append(r)
11         st[r] ← st[r] * 2
12         k ← 1
13   return st
```

2.4. Ruido

El ruido está definido como la información que no es de interés, degrada o contamina a los datos e impide hacer un buen uso de la serie de tiempo. Al igual que los datos faltantes y los valores atípicos impactan de manera directa en la calidad del pronóstico. La mayoría de las series de tiempo contienen una contribución estocástica producida por el ruido, ya sea por la medición u otras fuentes aleatorias que se agregan a los datos. Por lo general, ni la señal, ni el ruido se conocen con precisión antes de la medición, por lo que ambos deben estimarse después.

La razón señal ruido por sus siglas en inglés (SNR Signal to Noise Ratio) es una medida que se define como la razón entre la potencia de la señal y la potencia de ruido, a menudo expresada en decibeles (dB). En series de tiempo se define como la razón entre la potencia de una señal (información significativa o datos limpios) y la potencia del ruido de fondo (señal no deseada o datos contaminados):

Debido a que muchas señales tienen un rango dinámico muy amplio, las señales a menudo se expresan usando la escala logarítmica de decibeles. Usando la definición de SNR que muestra en (2.9). Donde P es la potencia media.

Tanto para la señal como la de ruido.

$$SNR_{dB} = 10\ log_{10}\left(\frac{P_{señal}}{P_{ruido}}\right) \tag{2.9}$$

Cuando los datos de una serie de tiempo están limpios (SNR es infinito) y cuando los datos están totalmente contaminados (SNR es negativo cuando: Potencia de la señal < Potencia del ruido).

En el análisis de los datos, a menudo se requiere una estimación razonable de la amplitud del ruido para llevar a cabo, por ejemplo, una prueba de bondad de ajuste. Así surge el problema de estimar la amplitud (por ejemplo, la varianza o la desviación estándar) de la contribución de ruido que carece de conocimiento sobre el ruido y la señal real.

Una técnica ampliamente utilizada para estimar la amplitud del ruido es el procedimiento $\beta\sigma$ desarrollado por Czesla et al. [Czesla et al., 2018]. Esta estimación del ruido no se limita a la astronomía, en ella se estudió un procedimiento para estimar la desviación estándar de la contribución estocástica asumiendo normalidad e independencia. Este algoritmo se encuentra implementado en la biblioteca de PyAstronomy (PyA), colección de paquetes que cumplan con un cierto estándar tanto en código como en calidad de documentación PyA [PyA, 2019].

El procedimiento $\beta\sigma$ se basa en el análisis de la distribución de derivadas numéricas descritas en Czesla et al. [Czesla et al., 2018]. Si la señal se puede aproximar por un polinomio de grado N, su derivada $N + 1$ desaparece, dejando sólo los términos de ruido para contribuir al valor de la derivada numérica, si y solo si es ruido aditivo.

En la implementación del procedimiento ($\beta\sigma$) se supone que el ruido es gaussiano con desviación estándar σ_0, en todos los análisis y contribuciones de ruido independientes en mediciones individuales. En la práctica, una distribución gaussiana a menudo proporciona una aproximación razonable de las propiedades del ruido. No se puede esperar que el ruido sea exactamente gaussiano. Una de las desventajas de este procedimiento es que es sensible a las series de tiempo donde predominan las altas frecuencias.

Otro método que se utilizará para calcular SNR son las medias móviles centradas son una variación de las medias móviles simples. En (2.10) se muestran las medias móviles simples, donde T representa el índice del dato actual, M representa el número de observaciones o datos que se van a promediar (tamaño de ventana), con ellas se pretende hacer un filtro a la serie

de tiempo para separar la información del ruido. En (2.11) se define las medias móviles centradas; se asume que M es impar. En estadística, una media móvil es un cálculo utilizado para analizar un conjunto de datos para crear series de promedios. Así las medias móviles son una lista de números en la cual cada uno es el promedio de un subconjunto de los datos originales.

$$MA_T = \frac{y_T + y_{T-1} + \cdots + y_{T-M}}{M} = \frac{1}{M} \sum_{j=T-M}^{T} y[j] \tag{2.10}$$

$$MA_T = \frac{\sum_{j=1}^{(M-1)/2} y_{T-j} + y_T + \sum_{j=1}^{(M-1)/2} y_{T+j}}{M} \tag{2.11}$$

La biblioteca de pandas contiene el método de medias móviles. La Figura 2.14 muestra el funcionamiento de las medias móviles centradas; en la columna llamada valores de la serie de tiempo se muestran los primero 15 datos de la serie de tiempo del sistema de Chen, la columna medias móviles ventana = 2, significa que se toma dos datos y se promedian, al inicio solamente se tiene el valor -11.200589 en la posición 0 y antes de él no existen datos, al momento de hacer el promedio se detecta que falta un valor y se marca con NaN, cuando avanza a la posición 1 el promedio toma el valor de $\frac{-11.200589+(-7.116236)}{2} = -9.158412$ y así sucesivamente hasta que recorre toda la serie de tiempo. La misma figura muestra la columna medias móviles ventana = 7; significa que se toman 7 datos y se promedian, al igual que cuando se inicio con la ventana 2 no se tienen datos antes de la posición 0 y se marca con NaN, avanza a la posición 1 ahora se tiene un dato atrás y otro adelante en total son tres datos pero la ventana es de 7 se marca con NaN, avanza a la posición 2 ahora se tienen dos datos atrás y dos datos adelante en total son cinco pero la ventana es de 7 se marca como NaN, avanza a la posición 3 ahora se tienen tres datos atrás y tres datos adelante en total 7 con el valor del centro de la ventana, se hace el promedio y da un valor de -10.834262 y así sucesivamente hasta que recorre la serie de tiempo.

El procedimiento del algoritmo que se propone para encontrar el ruido SNR es el siguiente;

1. Encuentra la desviación estándar del ruido usando el procedimiento $\beta\sigma$ que se encuentra en la biblioteca de PyAstronomy.

2. Calcula la varianza del ruido la cual es el cuadrado de la desviación estándar σ^2.

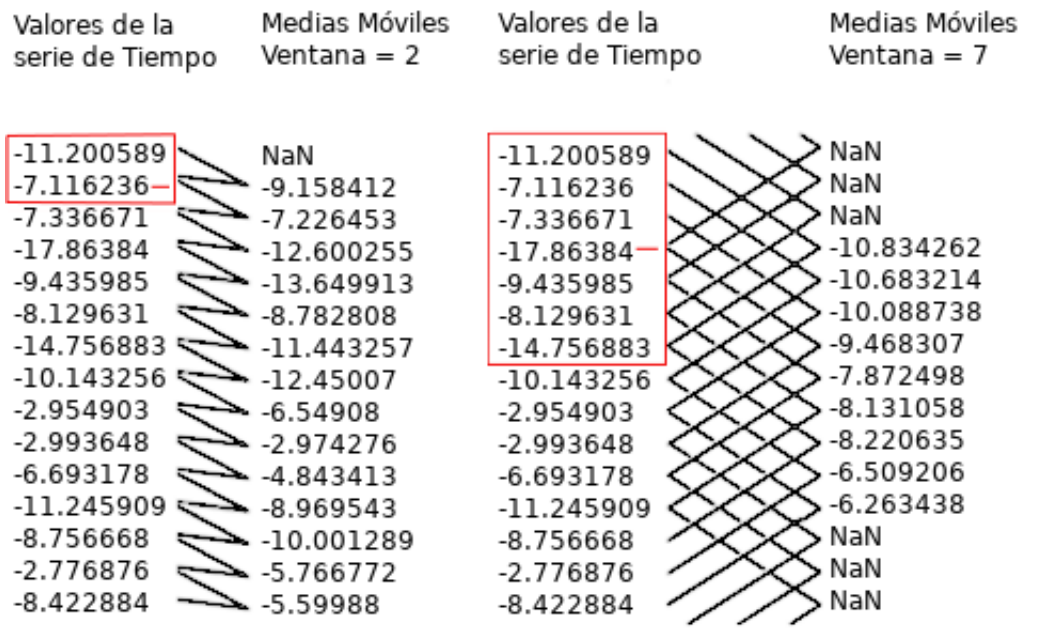


Figura 2.14: Funcionamiento de medias móviles centradas con una ventana de 2 y 7

3. Se inicia un ciclo el cual inicialmente tiene una ventana M igual a 2 y se incrementa en uno cada vez hasta encontrar el ruido (ver punto 7) o hasta que llega a un valor de 19.

4. Se usa el método de medias móviles centradas para separar la señal del ruido, la diferencia entre la serie de tiempo y la señal extraída genera residuos los cuales se consideran como ruido.

5. Se calcula la varianza de los residuos que son el ruido extraído de la serie de tiempo.

6. Se extrae la potencia de la señal y la potencia del ruido con el periodograma que viene incorporado en la librería de SciPy.

7. Si la diferencia entre la varianza de los residuos y la varianza del ruido es menor a 0.001, se usa (2.9) para calcular el valor del ruido (esta condición indica) que se encontró el nivel de SNR.

8. Si termino el ciclo y la diferencia entre las varianzas no fue menor a 0.001 se calcula el ruido con (2.9), la cual devuelve una aproximación.

El Algoritmo 7 *detectarSNR* se usa para la detección de ruido SNR en dB.

Algoritmo 7: Detección de ruido SNR

```
1  detectarSNR(st)
2    beq ← pyasl.BSEqSamp()
3    nstd, nstdstd ← beq.betaSigma(st)
4    varRuido ← np.power(nstd, 2)
5    for ventana in range(2,20) do
6      senal ← st.rolling(window = ventana, center = True).mean()
7      residuos ← st − senal
8      residuos ← residuos.dropna()
9      varResiduos ← np.var(residuos)
10     pxxDenSenal ← periodogram(senal[ np.isnan(senal)])
11     pxxDenRuido ← periodogram(residuos[ np.isnan(residuos)])
12     pxxDenSenal ← pxxDenSenal[ np.isnan(pxxDenSenal)]
13     pxxDenRuido ← pxxDenRuido[ np.isnan(pxxDenRuido)]
14     if np.abs(varRuido − varResiduos) < 1e − 3
15       if ventana == 2
16         snrCalculado ← ∞
17         break
18       else
19         snrCalculado ← 10 ∗ np.log10(np.mean(pxxDenSenal) / np.mean(pxxDenRuido))
20         break
21     if ventana == 19
22       snrCalculado ← 10 ∗ np.log10(np.mean(pxxDenSenal) / np.mean(pxxDenRuido))
23   return snrCalculado
```

Recibe una serie de tiempo, de la biblioteca PyAstronomy se crea un objeto beq de la clase BSEqSamp el cual hace la llamada al método *betaSigma*, se pasa la serie de tiempo como argumento y el método *betaSigma* devuelve la estimación de la desviación estándar del ruido en la serie de tiempo. Se calcula la varianza a partir de la desviación estándar obtenida del ruido. Para la separación de la señal y el ruido se crean dos variables; *senal* que va

a contener la señal filtrada, y *residuos* que va contener los residuos los cuales se consideran como el ruido. La variable *snrCalculado* guarda el valor del ruido calculado. Para la detección del ruido se usan medias móviles centradas para separar la señal y el ruido, las medias móviles utilizan ventanas para ir haciendo la separación se encontró que una ventana mayor a 20 no mejorá la detección del ruido. La señal separada del ruido es el resultado de aplicar las medias móviles, y el ruido son los residuos que se generan cuando a la serie de tiempo original se le resta la señal obtenida con medias móviles centradas. Se eliminan los valores NaN de los residuos, se obtiene la varianza de los residuos. Se obtiene la potencia para la señal y para los residuos. Después se comparan las varianzas, si la diferencia entre la varianza obtenida con el método $\beta\sigma$ y la varianza del ruido extraído es menor a 1e-3, indica que se encontró aproximadamente el nivel de ruido SNR y se devuelve en dB, $snrCalculado$ es igual a $10log_{10}(\frac{P_{senal}}{P_{ruido}})$ donde P es la potencia media tanto de la señal como del ruido. Si el tamaño de ventana es 19 indica que no se obtuvo la mejor aproximación del ruido y devuelve $snrCalculado$.

2.4.1. Inclusión de ruido

La inclusión de ruido se hace directamente en las series de tiempo sintéticas que se utilizaron para realizar los experimentos. El nivel de ruido inyectado a una serie de tiempo se expresa como SNR. Para la inclusión de ruido se implementó en Python la función AWGN (Añadir ruido gaussiano blanco a la señal por sus siglas en inglés Add White Gaussian Noise) que viene dentro de la caja de herramientas de MATLAB [Matlab, 2015]. Esta función puede agregar ruido blanco gaussiano aditivo para obtener la relación señal/ruido deseada (SNR). El uso principal de esta función es agregar ruido a una señal limpia (SNR infinita) para obtener una señal resultante con una SNR determinada. La función $y = AWGN(x, SNR)$ mide la potencia del vector de señal de la serie de tiempo y agrega ruido gaussiano blanco a la serie de tiempo para el nivel de SNR. Se garantiza en [Matlab, 2015] que la señal resultante tiene la SNR especificada.

A diferencia de las inclusiones de datos faltantes y de valores atípicos la inclusión de ruido no se especifica como un porcentaje si no nivel de ruido SNR en dB. Para incluir la perturbación de la serie de tiempo en esta tesis usamos niveles de ruido (5, 10, 15, 20, 25dB) para los experimentos, en el Algoritmo 8 *agregarRuido* lleva acabo la inclusión de ruido. Recibe dos argumentos, *st* que es la serie de tiempo que se desea perturbar y el segundo

argumento es el nivel de ruido SNR en dB. Se obtiene el tamaño de la serie de tiempo, el nivel de ruido SNR se cambia a una escala lineal. Se calcular la energía de la serie de tiempo y con ella se encuentra la densidad espectral de ruido. Se calcula la desviación estándar para el ruido y con ella se genera el ruido de manera aleatoria en toda la serie de tiempo y finalmente se le agrega el ruido a la serie de tiempo y se regresa la *st* perturbada con ruido SNR.

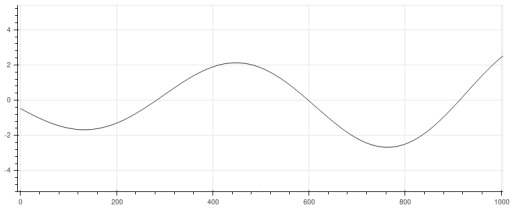
Algoritmo 8: Agregar ruido SNR

```
1 agregarRuido(st, SNR)
2    n ← len(st)
3    lsnr ← 10^(SNR/10)
4    E ← (1/n) Σ_{i=0}^{n-1} |st[i]|^2
5    ED ← E/lsnr
6    ruidoSigma ← √ED
7    ruido ← np.random.normal(0, ruidoSigma, n)
8    return st + ruido
```

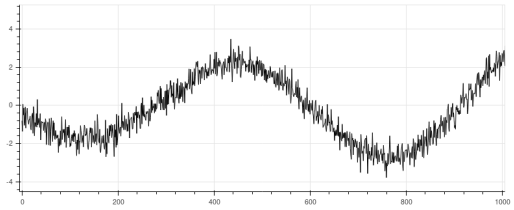
La Figura 2.15(a) muestra la serie de tiempo sintética Rössler Sprott [Sprott, 2003] sin inclusión de ruido SNR. La Figura 2.15(b) muestra la misma serie pero ahora con la inclusión de ruido $SNR = 25dB$ y la Figura 2.15(c) muestra la serie de tiempo con inclusión de ruido $SNR = 5dB$, cuando el ruido SNR tiende a ∞ se dice que la señal esta limpia y cuando SNR tiende a 0 se dice que se encuentra contaminada o distorsionada.

La Figura 2.15(d) muestra la serie de tiempo atractor de Rössler con ruido SNR = $25dB$ la cual se procesa con el Algoritmo 7 $detectarSNR(st)$ de color rojo se muestra la serie de tiempo ruidosa y de color negro la serie de tiempo filtrada con medias móviles centradas. Además el Algoritmo 7 entrega el ruido detectado SNR = $25.25dB$ con un error de 0.25.

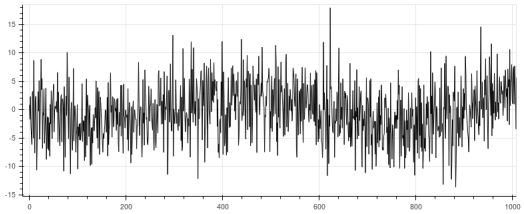
La Figura 2.15(e) muestra la serie de tiempo Rössler con ruido SNR $\approx$ $5dB$ la cual se procesa con el Algoritmo 7 $detectarSNR(st)$ de color rojo se muestra la serie de tiempo ruidosa y de color negro la serie de tiempo filtrada con medias móviles centradas. Además el Algoritmo 7 entrega el ruido detectado SNR = $5.16dB$ con un error absoluto de 0.16. Se puede observar que esta serie de tiempo se encuentra más contaminada que la serie de tiempo con SNR = $25dB$; el nivel de ruido se detectó de manera correcta.



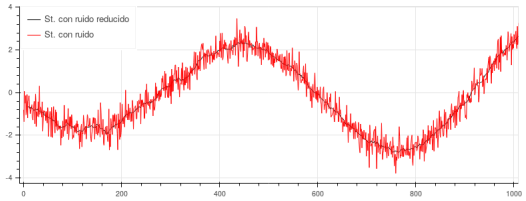
(a) 0_0_0



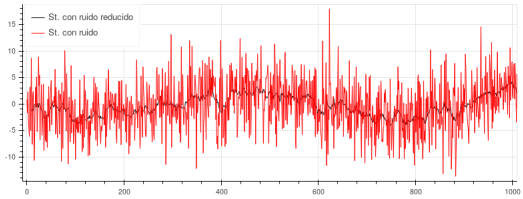
(b) 0_0_25



(c) 0_0_5



(d) 0_0_25 filtrada (ruido reducido) linea color negro



(e) 0_0_5 filtrada (ruido reducido) linea color negro

Figura 2.15: Segmento de la serie de tiempo atractor de Rössler.

Con estos experimentos donde las series de tiempo se perturbaron con diferentes niveles de ruido se prueba que el Algoritmo 7, detecta de manera aproximada la característica del ruido aún cuando la serie de tiempo presenta mucha distorsión en los datos.

2.5. Caos

Lo fundamental de la ciencia es la consideración de que los experimentos son predecibles y repetibles. Así sorprendió a la mayoría de los científicos cuando se encontró un sistema determinista simple que no era predecible ni repetible. En cambio, exhibieron un caos en el que el cambio más pequeño en

las condiciones iniciales produce un resultado muy diferente, incluso cuando las ecuaciones se conocen exactamente Sprott [Sprott, 2003].

El caos ha sido estudiado durante mucho tiempo. Se suele creer que Jules Henri Poincaré matemático, físico y filósofo francés fue el primero que estudió el caos. A fines del siglo XIX, Poincaré estudió el problema de los tres cuerpos restringidos. Poincaré descubrió que la solución de este sistema simple es muy complicada y no se puede dar con precisión. Luego, en 1963, Lorenz reveló el "efecto mariposa"al estudiar la predicción del clima; por su trabajo, Lorenz es reconocido como el padre del caos. Pero el uso formal del caos es de la investigación de Li y Yorke [Li and Yorke, 1975]. Después de eso, el caos ha sido ampliamente estudiado y se han introducido muchos conceptos importantes, como las dimensiones, los exponentes de Lyapunov, Transformada de Fourier y transformada de Hilbert, y reconstrucción del atractor Liu [Liu, 2010].

Determinar la presencia de caos en una serie de tiempo es un problema que se resuelve con el cálculo de los exponentes de Lyapunov. El exponente de Lyapunov de un sistema dinámico, es una cantidad que caracteriza el grado de separación de dos trayectorias infinitesimalmente cercanas. Los exponentes de Lyapunov determinan la divergencia entre las trayectorias del espacio inicialmente cercanas $d(0)$, si las trayectorias se separan en el tiempo exponencialmente rápido se dice que el sistema es caótico, mientras que para un sistema estable las trayectorias se acercan exponencialmente rápido. El exponente de Lyapunov estima la magnitud del caos, la Figura 2.16 muestra la divergencia de las trayectorias.

Los exponentes de Lyapunov se denotan con la letra griega lambda (λ); $\lambda < 0$ quiere decir que el sistema en estudio no presenta caos, cuanto más negativo sea el exponente, mayor será su predictibilidad. Los puntos fijos superstables y los puntos periódicos superstables tienen un exponente de Lyapunov de $-\infty$, si $\lambda = 0$ indica que el sistema está en estado estable sin caos y $\lambda > 0$ quiere decir que existe caos. Wolf et al. [Wolf et al., 1985] presentaron los primeros algoritmos que permiten la estimación de exponentes de Lyapunov no negativos a partir de una serie de tiempo experimental. Posteriormente se han desarrollado nuevos algoritmos que obtienen el mayor exponente de Lyapunov. Rosenstein et al. [Rosenstein et al., 1993] presentan un método para calcular el mayor exponente de Lyapunov a partir de una serie de tiempo experimental. El método de Rosenstein sigue directamente la definición del mayor exponente de Lyapunov y es preciso porque aprovecha todos los datos disponibles. El algoritmo es rápido, fácil de implementar y

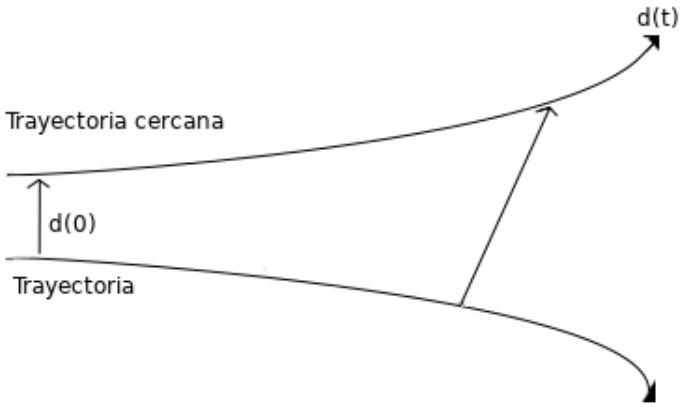


Figura 2.16: Divergencia de trayectorias

robusto a los cambios de dimensión de integración, tamaño del conjunto de datos, retardo de reconstrucción y nivel de ruido. La Figura 2.16 muestra dos trayectorias, la distancia inicial $d(0)$ entre ellas es corta y conforme pasa el tiempo las trayectorias y sus distancias divergen en $d(t)$.

Cuando se presenta divergencia de las trayectorias indica que el sistema pronto se comportará de manera muy diferente y la capacidad predictiva se pierde rápidamente. Cualquier sistema que contenga al menos un exponente de Lyapunov positivo se define como caótico.

El algoritmo de Rosenstein está implementado en Nolds de Python Scholzel [Scholzel,]; es una pequeña biblioteca basada en números que proporciona una implementación y un recurso de aprendizaje para medidas no lineales para sistemas dinámicos basados en series de tiempo unidimensionales.

El algoritmo de Rosenstein [Rosenstein et al., 1993] es capaz de recuperar el mayor exponente de Lyapunov, el cual esta definido en (2.12) donde $d(t)$ es la divergencia promedio en el tiempo t y C es una constante que normaliza la separación inicial.

$$d(t) = C\, e^{\lambda_1 t} \tag{2.12}$$

El primer paso es reconstruir la dinámica del atractor de la serie de tiempo. Para ello, se implementa el método de los retardos Takens [Takens, 1981], la trayectoria reconstruida, ahora llamada X, se puede expresar como una

matriz en la que cada fila es un vector de fase-espacio dado por (2.13).

$$X = (X_1, X_2, \ldots, X_M)^T \tag{2.13}$$

De manera que cada valor X_i es el estado del sistema en tiempo discreto $i = 1, 2, \ldots, M$. Para una serie de tiempo de N puntos $x_1, x_2, \ldots, x_N$ cada X_i es dado por (2.14), donde τ es el retardo y m es la dimensión de embebido (es la dimensión mínima del espacio en el que reconstruye la fase a partir de sus mediciones y en el que la trayectoria no se cruza).

$$X_i = [x_i, x_{i+\tau}, x_{i+2*\tau}, \ldots, x_{i+(m-1)*\tau}] \tag{2.14}$$

Por lo tanto, X es una matriz de tamaño (M x m) y las constantes m, M, τ y N se relacionan como en (2.15)

$$M = N - (m-1)\tau \tag{2.15}$$

En el método de Rosenstein, el retardo se aproxima para igualar el retardo cuando la función de autocorrelación de la serie de tiempo cae a $1-1/e$ de su valor inicial. El cálculo de este τ se puede lograr utilizando la transformada rápida de Fourier.

Para cada vector X_i, se encuentra el vecino más cercano X_j donde $d_i(O)$ es la distancia inicial desde el punto i hasta su vecino más cercano j utilizando la distancia euclidiana en (2.16).

$$d_i(0) = \min_{x_j} d_E(X_i, X_j) \tag{2.16}$$

El algoritmo de Rosenstein impone la restricción adicional de que los vecinos más cercanos deben tener una separación temporal mayor que el periodo medio $i - j > periodo\ medio$. Esto permite que se considere cada par de vecinos como condiciones iniciales cercanas para diferentes trayectorias. El mayor exponente de Lyapunov se estima como la tasa media de separación de los vecinos más cercanos. El algoritmo de Rosenstein se basa principalmente en el trabajo Sato et al [Sato et al., 1987], en el que el máximo exponente de Lyapunov λ se estima con (2.17), donde k es el período de muestreo de la serie de tiempo, y $d_j(i)$ es la distancia entre el j-ésimo par de vecinos más cercanos después de i pasos de tiempo discreto.

$$\lambda = \frac{1}{k(M-i)} \sum_{j=1}^{M-i} ln(\frac{d_j(i)}{d_j(0)}) \tag{2.17}$$

A medida que se siguen las trayectorias de X_i y X_j en el tiempo de un sistema caótico, las distancias entre X_{i+k} y X_{j+k} indicadas como $d_i(k)$ aumentarán de acuerdo con la ley de potencia en (2.18). λ es una buena aproximación del máximo exponente de Lyapunov, porque la expansión exponencial a lo largo del eje asociado con este exponente dominará rápidamente la expansión o contracción a lo largo de otros ejes.

$$d_i(k) = ce^{(\lambda k)} \tag{2.18}$$

Para calcular λ, observamos el logaritmo de la trayectoria de la distancia, porque con (2.19) da un conjunto de líneas (una para cada índice i) cuya pendiente es una aproximación de λ. Por lo tanto, se extrae la trayectoria media del registro $d'(k)$ tomando la media del registro $(d_i(k))$ sobre todos los vectores de órbita X_i. Luego se ajusta una línea recta a la gráfica de $d'(k)$ contra k. La pendiente de la línea da el parámetro deseado λ.

$$log(d_i(k)) = log(c) + \lambda k \tag{2.19}$$

En (2.19) se representa un conjunto de líneas aproximadamente paralelas, (para $i = 1, 2, ..., m$), cada una con una pendiente aproximadamente proporcional a λ_1. El mayor exponente de Lyapunov se calcula de forma fácil y precisa utilizando un ajuste de mínimos cuadrados en la línea "promedio"definida por (2.20). Donde $\langle\ \rangle$ denota el promedio sobre todos los valores de i. Este proceso de promedio es la clave para calcular valores precisos de λ_1, utilizando una serie de tiempo con pocos datos y ruidosos.

$$y(i) = \frac{1}{\Delta t}\langle ln\ d_k(i)\rangle \tag{2.20}$$

En la Figura 2.17 se muestra el diagrama de flujo del método de Rosenstein para la estimación del mayor exponente de Lyapunov.

El algoritmo de Rosenstein implementado por Nolds Scholzel [Scholzel,] determina el valor del exponente de Lyapunov (si es positivo indica que la serie de tiempo presenta un comportamiento caótico). La extracción de esta característica se espera aporte información de interés para generar el regresor del error.

En este Capítulo se presentaron 10 series de tiempo sintéticas, de las cuales 9 son caóticas y 1 función seno que se usa como referencia para los experimentos, estas series de tiempo se generaron porque es importante partir

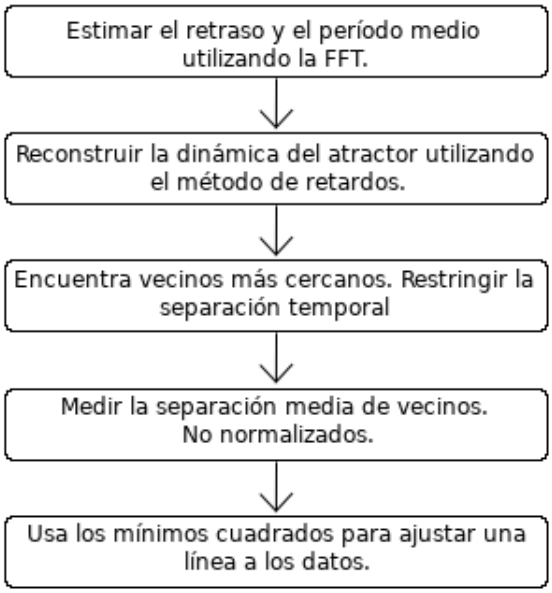


Figura 2.17: Diagrama de flujo del método de Rosenstein

con series de tiempo que contengan datos limpios, por tal motivo se utilizan en esta tesis. También se presentaron las cuatro características que se desean extraer de la serie de tiempo; la primera característica son los datos faltantes los cuales son detectados con el Algoritmo 1 $datosFaltantes$, para perturbar la serie de tiempo con esta característica se usa el Algoritmo 2 $eliminarDatos$ la cual marca los datos como faltantes, y para completar la serie de tiempo se usa el Algoritmo 3 $imputar$. La segunda característica son los valores atípicos, para la detección de estos se hace en dos pasos primero se detectan los valores atípicos globales con el Algoritmo 4 $deteccionGlobal$, después se detectan los valores atípicos locales con el Algoritmo 5 LOF, para perturbar las series de tiempo con valores atípicos se usa el Algoritmo 6 $agregarValoresAtipicos$. La tercera característica es el nivel de ruido SNR detectado por el Algoritmo 7 $detectarSNR$ el cual se creó en esta tesis, para perturbar las series de tiempo con el ruido SNR se usa el Algoritmo 8 $agregarRuido$. La cuarta característica el caos el cual es determinado por el Algoritmo de Rosenstein, las series de tiempo no pueden ser perturbadas con caos porque el caos es parte de la naturaleza de las series de tiempo.

Capítulo 3

Pronóstico de las series de tiempo sintéticas perturbadas

El pronóstico de una serie de tiempo es un problema importante que abarca muchos campos. Los problemas de pronóstico son clasificados a corto, mediano y largo plazo. Los problemas a corto plazo implican predecir eventos en períodos cortos de tiempo (minutos, horas, días, semanas y meses) en el futuro. Los pronósticos a mediano plazo se extienden de 1 a 2 años en el futuro, y los problemas de pronóstico a largo plazo pueden extenderse más de 2 años. Generalmente los pronósticos a corto y mediano plazo se usan para actividades como administración de operaciones, presupuestos y la selección de nuevos proyectos de investigación y desarrollo. Los pronósticos a largo plazo se usan en temas como la planificación estratégica Douglas C. Montgomery and Kulahci [Douglas C. Montgomery and Kulahci, 2015]. Éstas son solo algunas de las muchas situaciones diferentes en las que se requieren pronósticos para tomar buenas decisiones.

A pesar de la amplia gama de situaciones problemáticas que requieren pronósticos, solo hay dos tipos generales de técnicas de pronóstico: métodos cualitativos y métodos cuantitativos. Las técnicas de pronóstico cualitativo son a menudo de naturaleza subjetiva y requieren un juicio por parte de los expertos. Los pronósticos cualitativos se usan cuando hay pocos o ningún dato histórico en el que basar el pronóstico. Un ejemplo es la introducción de un nuevo producto, para el cual no hay un historial relevante, por lo tanto pueden hacer un pronóstico con ayuda de opiniones de expertos. Por otro lado las técnicas de pronóstico cuantitativo hacen uso formal de datos históricos y un modelo de pronóstico. El modelo resume formalmente los

patrones en los datos entre los valores actuales y anteriores de la variable. El modelo se utiliza para proyectar los patrones en los datos hacia el futuro. En otras palabras, el modelo de pronóstico se utiliza para extrapolar el comportamiento pasado y actual en el futuro Douglas C. Montgomery and Kulahci [Douglas C. Montgomery and Kulahci, 2015]. en esta tesis se usa el perceptrón multicapa, que es un tipo de red neuronal, para realizar el modelo de pronóstico.

3.1. Redes neuronales MLP

Las redes neuronales artificiales están inspiradas en la organización y el funcionamiento del cerebro. Las neuronas biológicas responden lentamente $10^{-3}s$ comparado con el $10^{-9}s$ que proporciona un circuito eléctrico. La ventaja del cerebro es que usa $\approx 10^{11}$ neuronas y cada neurona tiene $\approx 10^4$ conexiones Demuth et al. [Demuth et al., 2014], es por esto que el ser humano puede realizar tareas complejas.

Las redes neuronales se han aplicado en cientos de campos, algunas de las aplicaciones: En el campo aeroespacial; pilotos automáticos de aeronaves de alto rendimiento, sistemas de control de aeronaves, mejoras de piloto automático, detectores de falla de componentes de aeronaves. La electrónica; control de procesos, análisis de fallas de chips, visión artificial, la síntesis de voz, modelado no lineal. En entretenimiento; animación, efectos especiales, previsión de mercado. En lo financiero; análisis financiero corporativo, predicción del precio de la moneda. En robótica; control de trayectoria, robot montacargas, sistemas de visión, vehículos autónomos. En voz; reconocimiento de voz, compresión de voz, clasificación de vocales, síntesis de texto a voz.

El perceptrón multicapa (MLP Multilayer Perceptron) es una red neuronal artificial que se encuentra formado por múltiples capas ocultas. Esta característica le proporciona la capacidad de resolver problemas que no son linealmente separables, problema que tiene el perceptrón o perceptrón simple Demuth et al. [Demuth et al., 2014]. El perceptrón multicapa puede estar totalmente o localmente conectado. El perceptrón multicapa es un aproximador universal, puede aproximar relaciones no lineales entre los datos de entrada y salida. Esta red se ha convertido en una de las arquitecturas más utilizadas en el momento.

Las capas pueden clasificarse en tres: la primera es la capa de entrada, recibe los valores de entrada en la red, las siguientes son las capas ocultas

donde la salida de la primer capa es la entrada a la segunda capa y la salida de la segunda capa es la entrada a la tercera capa y así hasta las nh capas que tenga el perceptrón, cada capa puede estar formada por diferente número de neuronas. La última capa está formada por neuronas que producen los resultados de la red, la arquitectura de un perceptrón multicapa se muestra en la Figura 3.1.

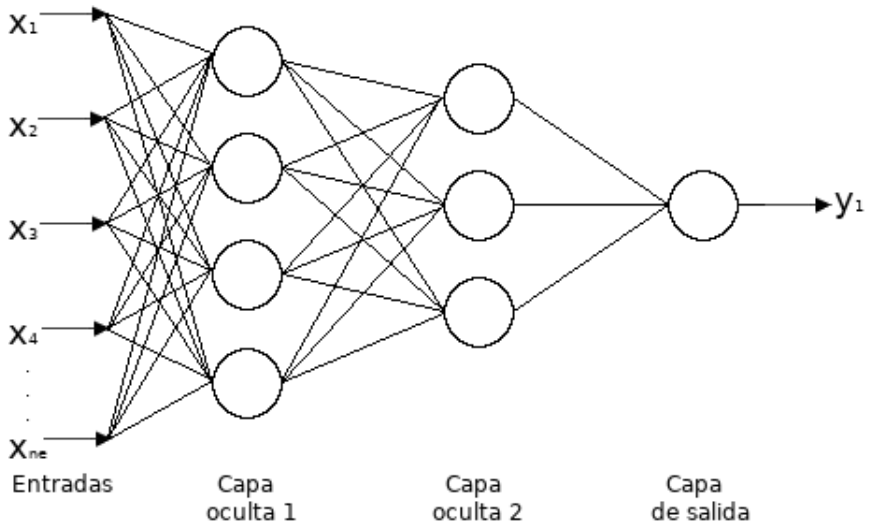


Figura 3.1: Arquitectura del perceptrón multicapa tiene la entrada desde x_1 hasta x_{ne}, dos capas ocultas la primera con 4 neuronas la segunda con 3 neuronas y la capa de salida con 1 neurona, la cual produce el resultado de la red.

Las neuronas biológicas están activas (excitadas) o inactivas (no excitadas) se dice que tiene un "estado de activación". Las neuronas artificiales también tienen estados de activación conocidos como funciones de activación o de transferencia, éstas pueden ser lineales o no lineales. Existen diferentes funciones de activación; la Tabla 3.1 muestra las funciones que se usaron en los perceptrones multicapa para el pronóstico, la columna nombre muestra el nombre de las funciones, la columna relación entradas/salidas muestra como entrada la variable h y como salida la variable a, la columna f. en keras muestra el nombre de la función en la biblioteca de Keras, y la columna figura muestra el comportamiento de cada función gráficamente. La primer función

es la positiva lineal cuando el valor de entrada $h < 0$ tiene un valor de salida $a = 0$, si $h \geq 0$ el valor de salida $a = h$ por lo tanto los posibles valores de salida van desde 0 hasta h. La función sigmoide $a = \frac{1}{1+e^{-h}}$ puede dar valores de salida en un rango que va de 0 a 1. La función lineal $a = h$ toma el valor de h y por último la función tangente hiperbólica calcula el valor de salida de la siguiente manera $a = \frac{e^h - e^{-h}}{e^h + e^{-h}}$, el valor de salida se encuentra en un rango que va de -1 a 1. Estas funciones de activación no son las únicas existen muchas más que se pueden consultar en el Capítulo 2 del libro "Neural Network Design"Demuth et al. [Demuth et al., 2014].

Tabla 3.1: Tabla de funciones de activación que se usaron en el modelado.

Nombre	Relación entradas/salidas	F. en Keras	Figura
Positiva lineal	a = 0 $h < 0$ a = h $0 \leq$ h	relu	
Sigmoide	$a = \frac{1}{1+e^{-h}}$	sigmoid	
Lineal	$a = h$	linear	
Tangente hiperbólica	a = $\frac{e^h - e^{-h}}{e^h + e^{-h}}$	tanh	

Por otro lado no se tiene una fórmula que indique cuantas capas necesitamos, cuantas neuronas necesita cada capa, tampoco que función de activación utilizar. Por eso que el diseño de una red neuronal artificial RNA se hace a prueba y error modificando cada parámetro obtener el mejor resultado posible.

3.1.1. Modelos

El diseño de modelos para los perceptrones multicapa son creados para poder llevar acabo el pronóstico de la serie de tiempo. Para crear estos modelos se uso la librería Keras, es una API de redes neuronales de alto nivel,

escrita en Python y capaz de ejecutarse sobre TensorFlow. Keras Fue desarrollado con un enfoque que permite la experimentación rápida. Pasar de la idea al resultado con el menor retraso posible es la clave para hacer una buena investigación Chollet et al. [Chollet et al., 2015]. En esta tesis se usó Keras porque permite la creación de los perceptrones multicapa de manera eficiente (a través de la facilidad de uso, la modularidad y la extensibilidad).

En la Sección 3.1 se explicó la arquitectura del perceptrón multicapa. Las series de tiempo que se usaron en esta tesis son univariadas (análisis de una sola variable), la estructura de datos de la serie de tiempo se necesita transformar a una estructura de tabla, esto se aplicá a todas las series de tiempo. Las tablas de datos fueron creadas con la biblioteca PSR (reconstrucción del espacio de fase) Kennel et al. [Kennel et al., 1992], la cual determina la dimensión de embebido m que corresponde al número de entradas del perceptrón multicapa, también calcula el τ tiempo de retardo óptimo el cual es utilizado como la predicción (objetivo).

Por ejemplo la Tabla 3.2 muestra los primeros 9 valores de la serie de tiempo sistema Chen no perturbada; esta estructura de datos necesita ser transformada para poder usar el perceptrón multicapa.

Tabla 3.2: Primeros 9 valores de la serie de tiempo sistema Chen no perturbada.

X	-10	-9.65	-9.30	-8.94	-8.58	-8.22	-7.86	-7.49	-7.13	...

Transformando la serie de tiempo del sistema de Chen, la Figura 3.2 muestra los primeros 9 datos, encerradas con un cuadrado rojo están las entradas $m = 6$ que tendrá el perceptrón multicapa del sistema Chen y $\tau = 1$ indicado con una flecha. Se recorre toda la serie de tiempo y se construye una tabla de datos la cual esta diseñada para esta serie de tiempo con una capa de entrada de 6 valores y el $\tau = 1$ es el objetivo y esto se puede observar en la Tabla 3.3.

La transformación a tabla se aplicó a todas las series de tiempo tanto las perturbadas como las no perturbadas, en la Tabla 3.4 se muestran algunas de las series de tiempo perturbadas a las cuales se les calcularon los parámetros m y τ con la biblioteca PSR, con los cuales fue creada cada tabla para estas series de tiempo. Nota la Tabla 3.4 solo muestra los parámetros de algunas series de tiempo, en total son 2,160, el tiempo para generar las tablas de

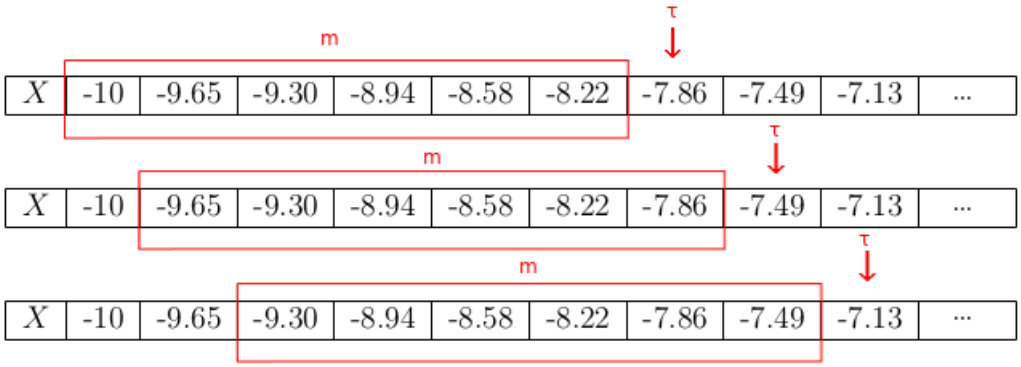


Figura 3.2: Transformación de la serie de tiempo a tabla de datos con $m = 6$ y $\tau = 1$

Tabla 3.3: Tabla de datos construida con $m = 6$ y $\tau = 1$.

X						y
-10.00	-9.65	-9.30	-8.94	-8.58	-8.22	-7.86
-9.65	-9.30	-8.94	-8.58	-8.22	-7.86	-7.49
-9.30	-8.94	-8.58	-8.22	-7.86	-7.49	-7.13
⋮	⋮	⋮	⋮	⋮	⋮	⋮

datos y encontrar los parámetros m y τ fue de aproximadamente ≈ 20 horas.

Después de generar las tablas de datos se encontraron los mejores modelos de perceptrón multicapa para las 10 series de tiempo sin perturbación.

El Algoritmo 9 *pronosticoMLP*; recibe 6 argumentos el primero nombre de la serie de tiempo, el segundo los datos de entrenamiento como entradas, el tercero los datos de entrada para la validación, el cuarto son los datos del objetivo de entrenamiento, el quinto son los datos del objetivo de la validación, el sexto son los mejores modelos encontrados para cada serie de tiempo sin perturbaciones. Al inicio se define el tipo de modelo del perceptrón multicapa que es secuencial el cual es una pila lineal de capas diseñado en Keras. En la línea 3 se toma el modelo de la serie de tiempo. En la línea 4 se asigna el modelo de la serie de tiempo y Keras construye la arquitectura. Después se configura el modelo de entrenamiento en este caso "rmsprop"es el optimizador que presenta un buen desempeño y es rápido MSE es la función

Tabla 3.4: m y τ calculados para crear las tablas de datos.

Perturbaciones	0_0_0		0_0_5		0_5_0		5_0_0		5_5_5	
Serie de tiempo	m	τ	m	τ	m	τ	m	τ	m	τ
Sistema Chen	6	1	5	1	6	1	6	1	5	4
Oscilador Duffing	6	1	5	1	6	1	6	1	5	1
Atractor cíclico simétrico de Halvorsen	6	1	5	2	6	1	6	1	5	1
Atractor de Lorenz	6	17	5	1	6	17	6	8	5	1
Atractor de Rössler	6	1	5	1	6	1	6	1	5	2
Atractor de Rucklidge	6	1	5	1	6	1	6	1	5	3
Oscilador Shawn-van der Pol	6	1	5	1	6	1	6	1	6	1
Flujo cúbico más simple	6	6	5	2	6	6	6	6	5	2
Flujo lineal por partes más simple	6	1	5	1	6	1	6	1	5	1
Seno	6	1	3	2	6	1	6	1	4	3

objetivo dada por (3.1) donde y_i es el valor real, $\hat{y}_i$ es el valor pronosticado y N es el numero de puntos, esta función objetivo hace que el valor de pérdida se minimice. Se crea un objeto para detener el entrenamiento cuando el valor de pérdida ha dejado de mejorar con una paciencia igual a cinco. Se crea una devolución de llamada que puede ser una función o un conjunto de funciones que se aplicarán en determinadas etapas del procedimiento de entrenamiento. Después de configurar el modelo con los parámetros, se realiza el entrenamiento del modelo pasando como argumentos los datos de entrenamiento tanto las entradas X como los objetivos Y, se indica que se tendrán

a lo más 500 épocas para encontrar el mejor modelo se le pasa la lista de la devolución de llamadas y también le indicamos que muestre la barra de progreso con verbose = 1. En la línea 9 se hace la predicción con los datos de entrenamiento. En la línea 10 se hace la predicción ahora con los datos de validación. En la línea 11 se obtiene el error SMAPE del entrenamiento usando (1.1) donde y es el valor real y $\hat{y}$ es el valor pronosticado, en la línea 12 se obtiene el error SMAPE en la validación, en la línea 14 se regresa el error SMAPE de entrenamiento y validación termina el algoritmo.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{3.1}$$

Algoritmo 9: pronosticoMLP

```
1  pronosticoMLP(nombreTs, entreX, validaX, entreY, validaY, modelos)

2    modeloMLP ← Sequential()
3    modeloSt ← modelos[nombreTs]
4    modeloMLP.modelos(modeloSt)
5    modeloMLP.compile(optimizer='rmsprop',loss='mse')
6    detener ← paradaTemprana(monitor='loss', patience=5)
7    listaLlamadas ← [detener]
8    modeloMLP.fit(entreX, entreY, epochs=500,
     callbacks=listaLlamadas,verbose=1)
9    predicEntre ← modeloMLP.predict(entreX)
10   predicValida ← modeloMLP.predict(validaX)
11   smapeEntre ← smape(entreY, predicEntre)
12   smapeValidation ← smape(validaY, predicValida)
13   return smapeEntre, smapeValidation
```

La Figura 3.3 muestra la arquitectura del modelo perceptrón multicapa para la serie de tiempo sistema Chen.

Para determinar la arquitectura del mejor modelo para cada una de las series de tiempo no perturbadas; fue necesario hacerlo a prueba y error, por ejemplo para la serie de tiempo del sistema Chen se usaron diferentes números de capas ocultas, comenzando con una capa oculta y otra a la salida intercambiando las funciones de activación y el número de neuronas no se logró que el modelo capturara el comportamiento de la serie de tiempo. El mejor modelo con esa arquitectura fue la capa oculta tenía 1,000 neuronas,

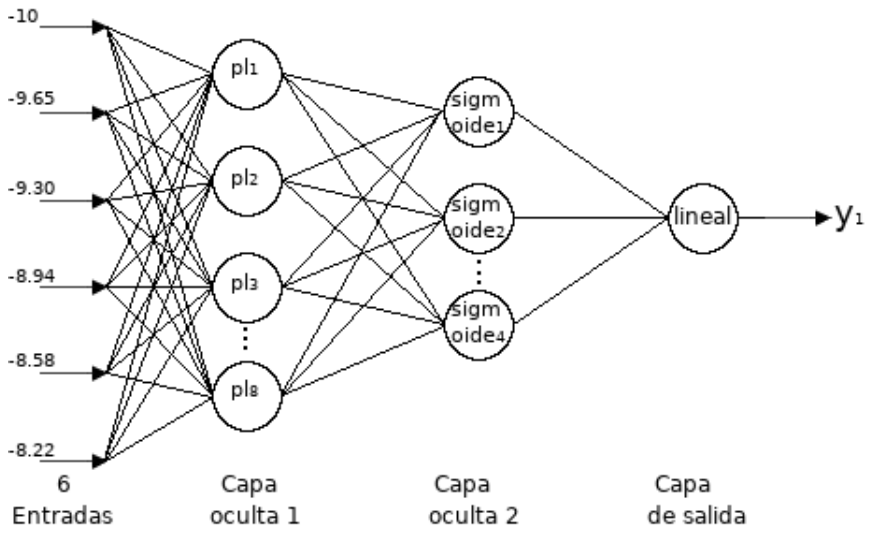


Figura 3.3: Modelo del perceptrón multicapa para el sistema Chen tiene 6 entradas, dos capas ocultas la primera con 8 neuronas que se activan con la función positiva lineal (pl) la segunda con 4 neuronas que se activan con la función sigmoide y la capa de salida tiene 1 neurona que se activa con la función lineal.

una función de activación relu y la capa de salida tenía una neurona con la función de activación tangente hiperbólica; el error SMAPE fue de 122.90. Como esa arquitectura no sirvió entonces se modificó y se agregó una capa oculta, cambiando el número de neuronas e intercambiando las funciones de activación se encontró un buen modelo el cual capturó el comportamiento de la serie de tiempo, la capa uno tenía 8 neuronas y una función de activación relu, la capa dos tenía 4 neuronas y una función de activación sigmoid y la capa de salida con una neurona y la función de activación linear; el error SMAPE fue de 1.28. Pensando en mejorar el modelo aún más se agregó otra capa oculta, la capa uno tenía 8 neuronas y una función de activación tanh, la capa dos tenía 8 neuronas y una función de activación sigmoid, la capa tres tenía 8 neuronas y una función de activación relu y la capa de salida con una neurona y la función de activación linear, el error SMAPE fue de 1.42. Teniendo en cuenta estos resultados se optó por la arquitectura más simple que mostrará el mejor desempeño. Este fue el procedimiento que se realizó

para obtener el mejor modelo para las series de tiempo sin perturbar.

Tabla 3.5: Modelos de los perceptrones multicapa para cada serie de tiempo.

Serie de tiempo	Número de entradas	Capas ocultas número de neuronas y función de activación	Función de activación capa de salida
Sistema Chen	6	Capa 1: 8 relu Capa 2: 4 sigmoid	linear
Oscilador Duffing	6	Capa 1: 24 relu Capa 2: 4 sigmoid	linear
Atractor cíclico simétrico de Halvorsen	6	Capa 1: 24 relu Capa 2: 4 sigmoid	linear
Atractor de Lorenz	6	Capa 1: 17 sigmoid Capa 2: 17 sigmoid Capa 3: 17 sigmoid	linear
Atractor de Rössler	6	Capa 1: 8 relu Capa 2: 4 sigmoid	linear
Atractor de Rucklidge	6	Capa 1: 24 relu Capa 2: 8 sigmoid	linear
Oscilador Shawn-van der Pol	6	Capa 1: 24 relu Capa 2: 4 sigmoid	linear
Flujo cúbico más simple	6	Capa 1: 24 sigmoid Capa 2: 12 sigmoid	linear
Flujo lineal por partes más simple	6	Capa 1: 24 sigmoid Capa 2: 12 sigmoid	linear
Seno	6	Capa 1: 64 relu Capa 2: 32 sigmoid	linear

Para determinar los mejores modelos en esta tesis; Se comenzó variando el número de capas desde una hasta cinco. Después se vario el número de neuronas desde una hasta treinta en pasos de uno y después con cien, doscientas, quinientas y mil neuronas. Se variaron las funciones de activación entre las capas, los modelos que dieron el menor error SMAPE fueron con los que se trabajó. Fue así como se llegó a la Tabla 3.5 donde muestra los mejores

modelos encontrados para las series sintéticas, se describen de la siguiente manera; en la primer columna se encuentra el nombre de la serie de tiempo en la segunda columna número de entradas detectadas al momento de generar las tablas de datos con el parámetro m, en la tercera columna se encuentran las capas ocultas así como las neuronas que utiliza cada una y la función de activación, en la cuarta columna se encuentra la capa de salida; en ella solo se muestra el nombre de la función de activación porque todas las capas de salida en esta tesis solamente requieren de una neurona que representa el pronóstico $\hat{y}$.

Los tiempos de entrenamiento de los modelos del perceptrón multicapa para las series de tiempo son: sistema Chen $\approx$ 54.13 segundos. Oscilador Duffing $\approx$ 27.99 segundos. Atractor cíclico simétrico de Halvorsen $\approx$ 24.69 segundos. Atractor de Lorenz $\approx$ 55.66 segundos. Atractor de Rössler $\approx$ 39.68 segundos. Atractor de Rucklidge $\approx$ 22.88 segundos. Oscilador Shawn-van der Pol $\approx$ 53.96 segundos. Flujo cúbico más simple $\approx$ 56.35 segundos. Flujo lineal por partes más simple $\approx$ 55.21 segundos y seno $\approx$ 2.61 segundos.

3.2. Errores de pronóstico

La validación del modelo consiste en una evaluación del modelo de pronóstico para determinar que tan bueno es el desempeño en la aplicación prevista. No solo se debe evaluar el entrenamiento del modelo a los datos históricos también se deben examinar los errores de validación que se experimentarán cuando el modelo se use para pronosticar datos no usados en el entrenamiento. Los errores de entrenamiento generalmente son más pequeños que los errores de validación. Un método ampliamente utilizado para validar un modelo de pronóstico antes de entregarlo al cliente, es emplear algún tipo de división de datos, donde los datos se dividen en dos segmentos: un conjunto de entrenamiento y un conjunto de validación. El modelo se ajusta solo al conjunto de datos de entrenamiento, y luego se simulan los pronósticos de ese modelo para las observaciones en el conjunto de validación. Esto puede proporcionar una guía útil sobre como se desempeñará el modelo de pronóstico cuando se exponga a datos nuevos.

La Tabla 3.6 muestra los errores de pronóstico SMAPE, la columna serie de tiempo contiene el nombre de las series de tiempo, la columna 0_0_0 indica que las series no fueron perturbadas, la columna E contiene los errores de entrenamiento de cada modelo y V contiene los errores de validación. Se

observa que los errores de entrenamiento generalmente son menores a los de validación, eso depende de la cantidad de datos que se utilicen.

Tabla 3.6: Errores SMAPE en el entrenamiento (E) y en la validación (V).

	0_0_0	
Serie de tiempo	E	V
Sistema Chen	1.28	1.12
Oscilador Duffing	3.40	4.71
Atractor cíclico simétrico de Halvorsen	2.45	3.18
Atractor de Lorenz	6.59	7.06
Atractor de Rössler	1.07	1.01
Atractor de Rucklidge	2.36	2.04
Oscilador Shawn-van der Pol	2.91	2.96
Flujo cúbico más simple	2.71	2.71
Flujo lineal por partes más simple	3.22	3.30
Seno	9.18	8.62

En este capítulo se realizó la predicción de las series de tiempo con el perceptrón multicapa MLP es una red neuronal. Para encontrar los mejores modelos se variaron el número de capas, el número de neuronas, y las funciones que se muestran en la Tabla 3.1. Los mejores modelos encontrados se muestran en la Tabla 3.5 corresponden a las series de tiempo no perturbadas y la predicción para las series de tiempo perturbadas se hicieron con los mismos modelos de las no perturbadas esto con la intensión de reafirmar

que la calidad de los datos de una serie es muy importante para realizar su pronóstico.

Capítulo 4

Regresión de error de pronóstico

El antropólogo británico, Sir Francis Galton (1822-1911), parece ser el primero en introducir la palabra "regresión"en su estudio sobre la herencia. Encontró que, en promedio, las alturas de los niños no tienden hacia las alturas de los padres, sino hacia el promedio en comparación con los padres Galton [Galton, 1886] describió cómo determinar la relación entre la altura de los niños usando la altura de los padres. Hoy el análisis de Galton se llama "análisis de correlación", un término del cual Galton también es responsable.

El análisis de regresión es un conjunto de procesos estadísticos que sirven como base para extraer inferencias sobre las relaciones entre variables. Dado que estas técnicas son aplicables en casi todos los campos de estudio, incluidas las ciencias sociales, físicas y biológicas, negocios e ingeniería, el análisis de regresión es quizás el más utilizado de todos los métodos de análisis de datos. La regresión hoy en día utiliza métodos para predecir el valor de una variable de respuesta (dependiente) a partir de una o más variables predictoras (independientes), donde las variables son numéricas. Existen varias formas de regresión: lineal, múltiple lineal, ponderada, polinomial, no paramétrica y aprendizaje máquina, entre otras. Específicamente, el análisis de regresión ayuda a comprender como cambia el valor típico de la variable dependiente (o "variable de criterio") cuando varía una de las variables independientes, mientras que las otras variables independientes se mantienen fijas. Galton [Galton, 1886]. Dado que el análisis de regresión no es solo ajustar ecuaciones a los datos, el punto crucial es que para cada problema en varios campos de la ciencia es necesario aclarar el objetivo del problema y los métodos nece-

sarios para el análisis de regresión. En esta tesis se usa el método de regresión con bosques aleatorios que pertenece al área de aprendizaje máquina.

A partir de la extracción de las 4 características de una serie de tiempo (datos faltantes, valores atípicos, ruido y caos) el problema es obtener una estimación del error en la predicción. Esta parte es importante porque con el regresor se obtiene el error de predicción, sin la necesidad de hacer un modelado de pronóstico, el cual puede tardar de días hasta semanas para dar resultados y decidir si la calidad de los datos es buena.

4.1. Regresor de bosques aleatorios

Un árbol de decisión es un modelo que predice el valor de una variable dependiente basado en varias variables independientes. Cada nodo interior corresponde a una variable, con ramas para cada posible valor o rango de valores que la variable puede tener. Las hojas del árbol representan el valor calculado para la variable. El procedimiento para determinar la estructura del árbol independiente. Primero determina qué variable proporciona la mejor partición, usando ganancia de información o el índice Gini Raileanu y Stoffel [Raileanu and Stoffel, 2004]. La base de datos de ejemplos se divide de acuerdo a la decisión que el nodo representa y el procedimiento se aplica recursívamente.

La Figura 4.1 muestra una porción del árbol de decisión que se será presentado en la Sección 5.4. El nodo raíz selecciona la variable 1/SNR y decisión ≤ 0.1. Por la rama izquierda se procede si esto se cumple y el procedimiento de formación del árbol se efectúa recursívamente, con los subconjuntos del conjunto de entrenamiento para los cuales $1/\text{SNR} \leq 0.1$ y $1/\text{SNR} > 0.1$, en las ramas izquierda y derecha respectivamente. La hoja de la izquierda indica el valor de la función de regresión, en este caso 12.1.

El algoritmo de bosques aleatorios Breiman [Breiman, 2001] genera un conjunto de árboles de decisión y aplica la técnica de boosting Schapire [Schapire, 1990] para calcular el resultado final. Esto es, promedia los resultados de regresión de los arboles en el bosque aleatorio.

El algoritmo de bosques aleatorios Breiman [Breiman, 2001] se puede usar tanto para problemas de clasificación como de regresión. La diferencia entre estos es que en la clasificación se utiliza para separar el conjunto de datos en clases (Categorizar). Por otro lado la regresión se usa cuando la respuesta es numérica o continua, por lo tanto se aplica para los problemas de (predic-

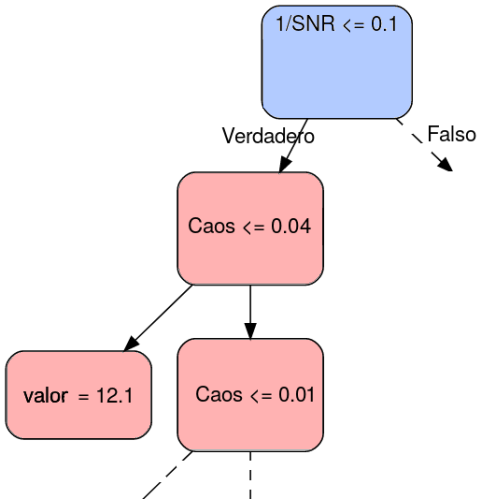


Figura 4.1: Parte del árbol de decisión

ción). Los bosques aleatorios son una combinación de árboles de decisión o (predictores), de modo que cada árbol depende de los valores de un vector aleatorio muestreado de forma independiente y con la misma distribución para todos los árboles del bosque. El error de generalización para los bosques converge hasta un límite a medida que aumenta el número de árboles en el bosque. El error de generalización de un bosque de clasificadores de árboles depende de la fuerza de los árboles individuales en el bosque y de la correlación entre ellos. Las estimaciones internas supervisan el error, la fuerza y la correlación, y se utilizan para mostrar la respuesta al aumento del número de características utilizadas en la división. Las estimaciones internas también se utilizan para medir la importancia variable. Estas ideas también son aplicables a la regresión. El elemento común en todos estos procedimientos es que para el k-ésimo árbol, se genera un vector aleatorio Θ_k, independiente de los vectores aleatorios pasados $\Theta_1, \ldots, \Theta_{k-1}$ pero con la misma distribución; se entrena un árbol usando el conjunto de entrenamiento Θ_k, lo que da como resultado un regresor $h(\mathbf{x}, \Theta_k)$, donde $\mathbf{x}$ es un vector de entrada. Después de que se genera una gran cantidad de árboles, se llama a estos procedimientos bosques aleatorios.

Los bosques aleatorios para la regresión se forman al crecer árboles dependiendo de un vector aleatorio, de modo que el predictor de árbol $h(\mathbf{x}, \Theta)$ tome valores numéricos en lugar de las etiquetas de clase. Los valores de sa-

lida son numéricos y se supone que el conjunto de entrenamiento se extrae independientemente de la distribución del vector aleatorio Y, $\mathbf{X}$. El error de generalización cuadrático medio para cualquier predictor numérico $h(\mathbf{x})$ es (4.1).

$$E_{\mathbf{X},Y}(Y - h(\mathbf{X}))^2 \tag{4.1}$$

El predictor de bosque aleatorio se forma tomando el promedio sobre los k árboles $\{h(\mathbf{x}, \Theta_k)\}$. De manera similar al caso de clasificación.

El algoritmo del regresor de bosques aleatorios se encuentra dentro de la biblioteca de sklearn Pedregosa et al. [Pedregosa et al., 2011]. Para ajustar el modelo del regresor se utilizó el Algoritmo 10 *regBosquesAleatorios*; recibe cinco parámetros el primero *numAr* es el número de árboles en el bosque aleatorio. El segundo parámetro *entreCa* es el 70% del conjunto de datos de las características que se utiliza como entrenamiento. El tercer parámetro *entreOb* es el 70% del conjunto de datos de los errores de predicción SMAPE que corresponden a las predicciones de los usados como objetivo de entrenamiento. El cuarto parámetro *validaCa* corresponde al restante 30% del conjunto de datos de las características que se usará como validación. El quinto parámetro *validaOb* corresponde al 30% restante del conjunto de datos de los errores de predicción SMAPE usados como objetivo de validación. En la línea 2 se crea el objeto que contendrá el modelo de bosques aleatorios recibe cuatro parámetros; número de árboles, la función para medir la calidad de una división, el parámetro $n\_jobs = -1$ indica que el proceso de ajuste se haga en paralelo. En la línea 3 se hace el ajuste del modelo con los datos de entrenamiento características y objetivos. En la línea 4 se hacen las predicciones con el modelo que se ajusto, usando los datos de las características de validación. En la línea 5 se calcula el error MAE como en (4.2).

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i| \tag{4.2}$$

Se calcula el error MAPE con (1.3). En la línea 7 se calcula la *precisionR* dada por (1.2) que obtuvo del modelo en la predicción con el conjunto de datos de validación. En la línea 8 se devuelve el regresor de bosque aleatorio,

los errores MAE, MAPE y la precisión del regresor.

Algoritmo 10: Regresor de bosques aleatorios

```
1 regBosquesAleatorios(numAr, entreCa, entreOb, validaCa, validaOb)

2   regresorBA ← RandomForestRegressor(n_estimators =
    numAr, criterion ="mse", n_jobs = −1)
3   regresorBA.fit(entreCa, entreObjetivo);
4   predicciones ← regresorBA.predict(validaCaracte)
5   mae ← MAE(predicciones, validaOb)
6   mape ← MAPE(predicciones, validaOb)
7   precisionR ← 100 − mape
8   return regresorBA, mae, mape, precisionR
```

El regresor de bosques aleatorios se formó variando el número de árboles; se hicieron pruebas iniciando con un bosque generado por 10 árboles donde la precisión fue de 75.72%. Se aumentó el número de árboles a 100 y la precisión fue de 80.33%. Se incremento nuevamente el número de árboles a 200 y la precisión fue de 82.33%, así se incremento el número de árboles hasta llegar a 1,000 árboles con una precisión de 85.13% donde se presentó la mejor precisión, para un bosque con más de 1,000 árboles ya no se presentó una mejora; con 2,000 árboles la precisión bajo a 84.98%.

En este capítulo se presentó el regresor de bosques aleatorios el cual pertenece a los algoritmos de aprendizaje máquina. El regresor de bosques aleatorios se puede usar tanto para problemas de clasificación como de regresión. La diferencia es que en clasificación se utiliza para separar el conjunto de datos en clases (categorizar). Por otro lado en la regresión se usa cuando la respuesta es numérica o continua, por lo tanto se aplica para los problemas de (predicción). El regresor de bosques aleatorios genera un conjunto de árboles de decisión, se comprobó que el número de árboles depende del problema, esto porque cuando se usaron 10 árboles la precisión fue de 75.72 %, con 1,000 árboles la precisión aumento a 85.13 y con 2,000 árboles la precisión bajo a 84.98. Cuando se usaron 2,000 bajo la precisión esto ocurre cuando el modelo se encuentra sobre ajustado y es incapaz de generalizar el problema, y hace una mala predicción con datos nuevos.

Capítulo 5

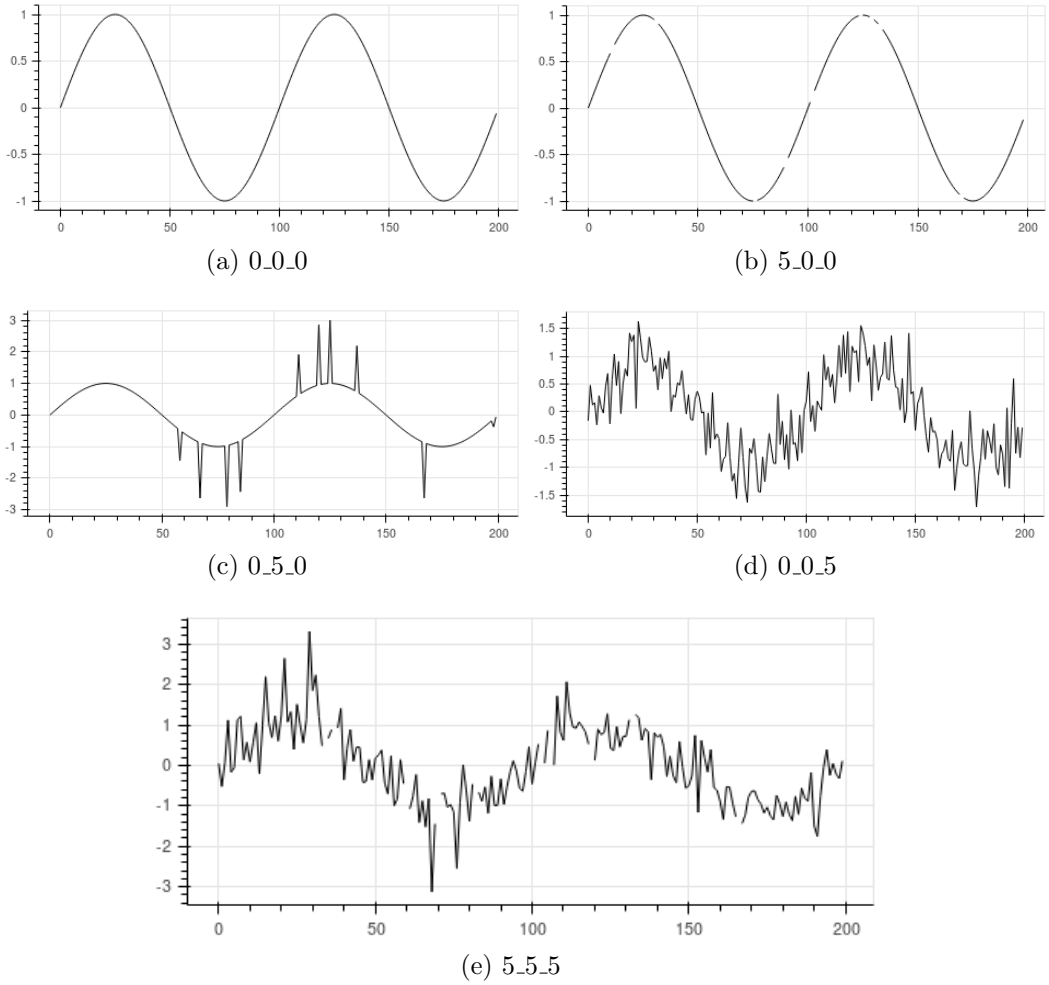
Resultados

Los resultados que se presentan en este capítulo se encuentran ordenados por las diferentes etapas del proceso para obtener la calidad de los datos. El primer resultado presenta la perturbación de las series de tiempo que se llevó a cabo con las cuatro características. El segundo resultado que se presenta es la extracción de las cuatro características. El tercer resultado es el pronóstico realizado con el perceptrón multicapa y el error SMAPE obtenido. El cuarto resultado es la regresión del error SMAPE diseñado con el regresor de bosques aleatorios, para generar el modelo del regresor se usó las características obtenidas y el error de pronóstico, finalmente se obtuvo la $PrecisionR$ del regresor en las predicciones.

5.1. Perturbación de las series de tiempo

Las series de tiempo de la Tabla 2.1 son las que se perturbaron con diferentes porcentajes de datos faltantes, valores atípicos y niveles de ruido. Para poder perturbar las series de tiempo se necesita la inclusión de datos faltantes la cual se lleva acabo con ayuda del Algoritmo 2. La inclusión de valores atípicos se lleva acabo con el Algoritmo 6, y finalmente para la inclusión de ruido se lleva acabo con el Algoritmo 8.

La serie de tiempo seno contiene 200 datos la cual se muestra en la Figura 5.1. En la Figura 5.1(a) se observa la serie de tiempo correspondiente a la codificación 0_0_0 la cual indica que no se encuentra perturbada y se ve claramente el comportamiento. La Figura 5.1(b) corresponde a la codificación 5_0_0 esta indica que un 5% de datos han sido eliminados un total de 10 datos



(a) 0_0_0 (b) 5_0_0

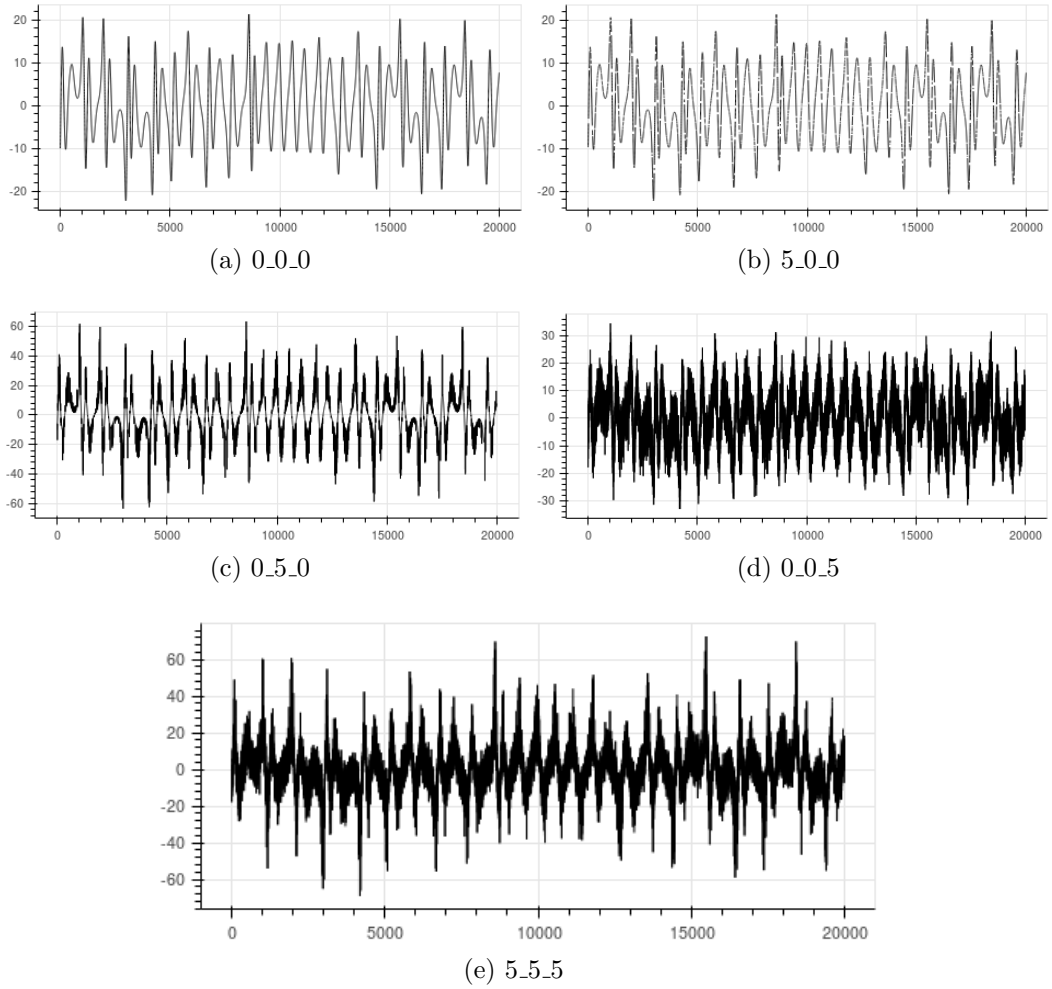
(c) 0_5_0 (d) 0_0_5

(e) 5_5_5

Figura 5.1: Serie de tiempo seno perturbada

es por eso que se ven huecos. La Figura 5.1(c) corresponde a la codificación 0_5_0 que indica que un 5% de los datos fueron alterados para generar los valores atípicos en total 10 datos por tal motivo se observan picos altos. La Figura 5.1(d) corresponde a la codificación 0_0_5 (nivel de ruido de $5dB$) éste es el nivel más alto de ruido que se usó en este trabajo por tal motivo se observa una fuerte perturbación. La Figura 5.1(e) corresponde a la codificación de 5_5_5 la cual contiene 5% de datos faltantes y valores atípicos y ruido de $5dB$ por consecuencia se observa una fuerte perturbación. La Figura 5.2

corresponde a la serie de tiempo sistema Chen en la cual se puede observar el mismo comportamiento cuando es perturbada, la diferencia es que esta serie contiene 20,000 datos y las gráficas son más densas.



(a) 0_0_0

(b) 5_0_0

(c) 0_5_0

(d) 0_0_5

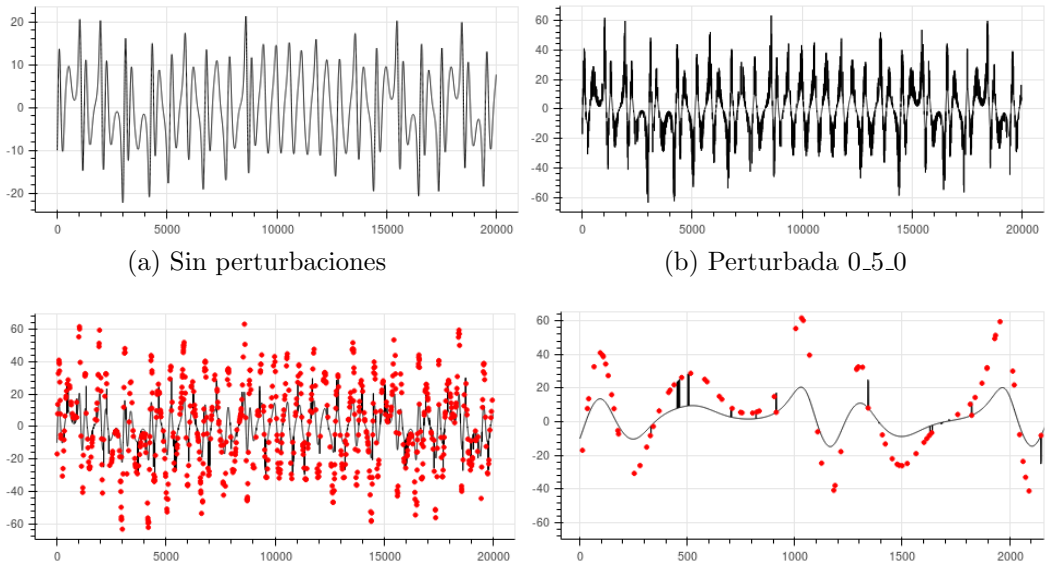
(e) 5_5_5

Figura 5.2: Serie de tiempo sistema Chen

5.2. Extracción de las características

La extracción de características se realizó después de haber perturbado las series de tiempo. Para extraer la primer característica que es el porcentaje

de los datos faltantes se utilizo el Algoritmo 1. Para el análisis de los datos faltantes solo se cuentan los índices donde no se encuentra un valor y al final se obtiene el porcentaje en la Figura 5.2(b) se pueden observar los huecos que se generan. La segunda característica que se extrae es el porcentaje de valores atípicos se realiza en dos etapas la primera es contar los atípicos globales con el Algoritmo 4 y para la segunda etapa es contar los atípicos locales con el Algoritmo 5 y finalmente sumarlos y obtener el porcentaje. En la Figura 5.3 se muestra la serie de tiempo sistema Chen. La Figura 5.3(a) no contiene perturbaciones y se encuentra graficada en un rango de (-22 a 22). La Figura 5.3(b) se encuentra perturbada con 5% de valores atípicos y ahora la gráfica aumento su rango de (-62 a 62). La Figura 5.3(c) muestra los valores atípicos extraídos como puntos rojos. En la Figura 5.3(d) se hace un acercamiento para mostrar los primeros 2000 datos para observar a detalle y darse cuenta que algunos valores que son atípicos no se reconocen y esto es porque los valores son un conjunto y se encuentran cerca uno del otro por lo tanto el Algoritmo 5 no los detecta como valores atípicos ya que se basa en densidad.



(a) Sin perturbaciones

(b) Perturbada 0_5_0

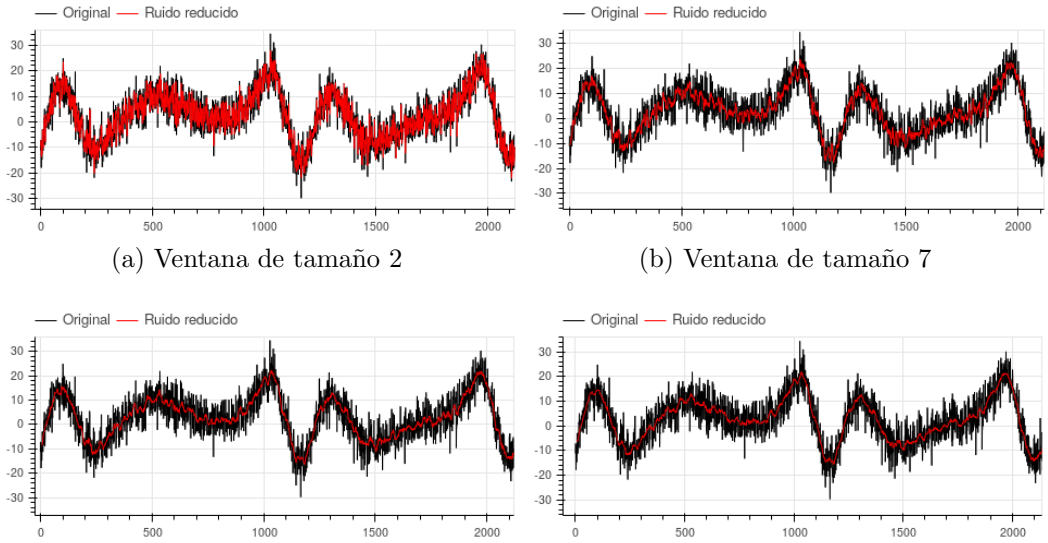
(c) Extraccion de los valores atípicos los puntos rojos son los valores atípicos

(d) Muestra de los primero 2000 datos

Figura 5.3: Serie de tiempo sistema Chen

La tercera característica es la extracción del nivel de ruido SNR con el

Algoritmo 7. En la Figura 5.4 se puede observar el proceso que se lleva acabo para detectar el nivel de ruido. La Figura 5.4(a) muestra que la extracción del ruido es $8.65dB$ con una ventana de tamaño 2. La Figura 5.4(b) extrajo ruido igual $5.87dB$ con una ventana de 7. La Figura 5.4(c) se extrajo ruido igual a $5.43dB$ con una ventana de 14 y finalmente la Figura 5.4(d) se extrajo $5.31dB$ con una ventana de tamaño 19.



(a) Ventana de tamaño 2

(b) Ventana de tamaño 7

(c) Ventana de tamaño 14

(d) Ventana de tamaño 19

Figura 5.4: Extracción del ruido de la serie de tiempo sistema Chen

La comprobación de que la extracción de las características se lleve a cabo correctamente se muestra en la Tabla 5.1, la cual muestra el desempeño de la extracción de la característica de datos faltantes. En la columna serie de tiempo se tienen los nombres de las series que se usaron. En la columna (5_0_0) corresponde a la perturbación del con el 5% de datos faltantes y así sucesivamente con las siguientes columnas 10_0_0, 15_0_0, 20_0_0 y 25_0_0. Por ejemplo para la serie de tiempo sistema Chen se detectó exactamente el 5% de datos faltantes al igual para las demás series de tiempo en la columna MAE se muestra el error que es 0 lo que indica que se extraen perfectamente el porcentaje de los datos faltantes para todas las series de tiempo y porcentajes.

En la Tabla 5.2 se muestra el desempeño de la extracción de la característica de valores atípicos. Las columnas (0_5_0, 0_10_0, 0_15_0, 0_20_0 y 0_25_0) corresponden a las series de tiempo perturbadas con ese porcentaje

Tabla 5.1: Extracción de datos faltantes de las series de tiempo perturbadas con 5, 10, 15, 20 y 25% de datos faltantes y cálculo del error MAE

Serie de tiempo	5_0_0	10_0_0	15_0_0	20_0_0	25_0_0	MAE
Sistema Chen	5%	10%	15%	20%	25%	0
Oscilador Duffing	5%	10%	15%	20%	25%	0
Atractor cíclico simétrico de Halvorsen	5%	10%	15%	20%	25%	0
Atractor de Lorenz	5%	10%	15%	20%	25%	0
Atractor de Rössler	5%	10%	15%	20%	25%	0
Atractor de Rucklidge	5%	10%	15%	20%	25%	0
Oscilador Shawn-van der Pol	5%	10%	15%	20%	25%	0
Flujo cúbico más simple	5%	10%	15%	20%	25%	0
Flujo lineal por partes más simple	5%	10%	15%	20%	25%	0
Seno	5%	10%	15%	20%	25%	0

de valores atípicos. Por ejemplo, para la serie de tiempo sistema Chen se detectaron (3%, 5%, 4%, 4%, 3%) de valores atípicos respectivamente en la columna MAE se muestra el error igual a 11.2%.

La Tabla 5.3 muestra el desempeño de la extracción de la característica del ruido para las columnas (0_0_5, 0_0_10, 0_0_15, 0_0_20 y 0_0_25) correspondientes a las series de tiempo perturbadas con ese nivel de ruido. Por ejemplo para la serie de tiempo sistema Chen se detectó ($5.31dB$, $10.27dB$, $15.18dB$, $20.23dB$, $25.15dB$) de ruido respectivamente en la columna MAEs e muestra el error de $0.23dB$.

Tabla 5.2: Extracción de los valores atípicos de las series de tiempo perturbadas con 5, 10, 15, 20 y 25% de valores atípicos y cálculo del error MAE

Serie de tiempo	0_5_0	0_10_0	0_15_0	0_20_0	0_25_0	MAE
Sistema Chen	3%	5%	4%	4%	3%	11.2
Oscilador Duffing	2%	2%	2%	2%	1%	13.2
Atractor cíclico simétrico de Halvorsen	3%	5%	5%	4%	3%	11
Atractor de Lorenz	4%	5%	6%	6%	5%	9.8
Atractor de Rössler	3%	5%	5%	4%	3%	11
Atractor de Rucklidge	2%	3%	3%	3%	3%	12.2
Oscilador Shawn-van der Pol	2%	3%	3%	3%	2%	12.4
Flujo cúbico más simple	3%	4%	4%	3%	2%	11.8
Flujo lineal por partes más simple	2%	3%	3%	3%	2%	12.4
Seno	2%	2%	0%	0%	0%	14.2

Se puede observar de las tablas 5.1, 5.2 y 5.3 que el mejor desempeño para extraer características es el de datos faltantes ya que esta se detecta con un error MAE de 0%. El ruido es otra característica que se extrae de forma aproximada con un error MAE de $0.23dB$ para el sistema Chen y el promedio de las 10 series es de $0.69dB$. Los valores atípicos no tuvieron un buen desempeño porque para el sistema Chen tiene un error MAE de 11.2% y el promedio de las 10 series es de 11.92%. Finalmente para la estimación del caos, esta medida solo se calcula no se puede obtener el error porque esta característica no fue incluida de manera manual a la serie de tiempo, para

Tabla 5.3: Extracción del ruido de las series de tiempo perturbadas con 5, 10, 15, 20 y 25dB de ruido y cálculo del error MAE

Serie de tiempo	0_0_5	0_0_10	0_0_15	0_0_20	0_0_25	MAE
Sistema Chen	5.31dB	10.27dB	15.18dB	20.23dB	25.15dB	0.23
Oscilador Duffing	5.29dB	10.29dB	15.26dB	20.60dB	26.68dB	0.62
Atractor cíclico simétrico de Halvorsen	3.45dB	8.30dB	13.37dB	18.33dB	23.34dB	1.63
Atractor de Lorenz	4.77dB	9.56dB	13.92dB	17.55dB	19.75dB	1.88
Atractor de Rössler	5.16dB	10.21dB	15.20dB	20.18dB	25.25dB	0.20
Atractor de Rucklidge	5.28dB	10.18dB	15.14dB	20.06dB	24.99dB	0.13
Oscilador Shawn-van der Pol	5.24dB	10.18dB	15.21dB	20.24dB	25.78dB	0.33
Flujo cúbico más simple	5.32dB	10.19dB	15.16dB	20.23dB	25.29dB	0.24
Flujo lineal por partes más simple	4.00dB	8.87dB	13.81dB	18.86dB	24.87dB	0.91
Seno	4.94dB	10.65dB	16.80dB	21.22dB	25.00dB	0.74

estimar la magnitud del caos se usa el algoritmo de Rosenstein el cual calcula el máximo exponente de Lyapunov.

5.3. Pronóstico

Para realizar el pronóstico para cada una de las series de tiempo, con los diferentes niveles de perturbación, se utilizaron los modelos perceptrón multicapa. La Tabla 5.4 muestra los errores SMAPE del entrenamiento (E) y la validación (V), se observa que las series de tiempo sin perturbaciones pre-

sentan los errores más pequeños, por ejemplo los errores de entrenamiento se encuentran en el intervalo $[1.07, 9.18]$ y los errores de validación se encuentran en el intervalo $[1.01, 8.62]$, Para las mismas series de tiempo pero perturbadas con un nivel de ruido $5dB$ los errores son mayores, por ejemplo los errores del entrenamiento se encuentran en el intervalo $[72.02, 100.58]$, y los errores de validación se encuentran en el intervalo $[72.56, 93.72]$. Con base a estos errores la perturbación del ruido es la que más afecta al pronóstico.

Tabla 5.4: Errores SMAPE en el entrenamiento (E) y en la validación (V).

	Perturbaciones							
	0_0_0		0_0_5		0_5_0		5_0_0	
Serie de tiempo	E	V	E	V	E	V	E	V
Sistema Chen	1.28	1.12	77.68	77.95	16.98	16.03	3.98	3.43
Oscilador Duffing	3.40	4.71	74.01	76.18	18.86	19.34	1.63	1.87
Atractor cíclico simétrico de Halvorsen	2.45	3.18	73.89	78.06	13.21	14.91	1.54	2.23
Atractor de Lorenz	6.59	7.06	81.90	83.71	24.26	25.53	5.19	5.84
Atractor de Rössler	1.07	1.01	83.15	77.18	20.24	20.11	2.24	2.03
Atractor de Rucklidge	2.36	2.04	100.58	93.72	22.74	20.09	5.41	4.76
Oscilador Shawn-van der Pol	2.91	2.96	72.02	72.56	16.11	16.59	1.28	1.27
Flujo cúbico más simple	2.71	2.71	74.57	73.53	17.37	17.79	2.48	2.57
Flujo lineal por partes más simple	3.22	3.30	79.25	78.25	17.46	17.94	1.55	1.44
Seno	9.18	8.62	80.18	74.68	33.98	43.92	11.31	14.00

La Figura 5.5 muestra el progreso del ajuste del modelo para el pronóstico

de la serie de tiempo del sistema Chen sin perturbaciones el cual usa como función objetivo minimizar el error MSE con (3.1). En la primera época se ajustó el modelo teniendo un error de 58.5593; en la segunda época se obtuvo un error de 46.6422. Se puede observar que bajó el error de manera notable, en la tercera época se obtuvo un error de 38.4495 y se observa que continua bajando. Así continua hasta llegar a la época 88, en la cual tiene un error de 0.0153. En las épocas 89, 90, 91, 92 y 93 no se mejoró el modelo es aquí donde se usa la parada temprana que tiene la función detener después de una paciencia de 5 intentos para mejorar el modelo.

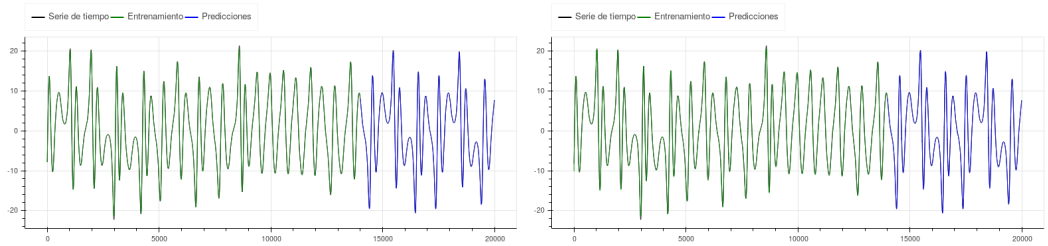
```
Epoch 1/500
13995/13995 [=======] - 1s 54us/step - loss: 58.5593
Epoch 2/500
13995/13995 [=======] - 1s 38us/step - loss: 46.6422
Epoch 3/500
13995/13995 [=======] - 0s 30us/step - loss: 38.4495
*           *           *              *
*           *           *              *
*           *           *              *
Epoch 88/500
13995/13995 [=======] - 1s 38us/step - loss: 0.0153
Epoch 89/500
13995/13995 [=======] - 1s 39us/step - loss: 0.0158
Epoch 90/500
13995/13995 [=======] - 1s 37us/step - loss: 0.0166
Epoch 91/500
13995/13995 [=======] - 1s 36us/step - loss: 0.0159
Epoch 92/500
13995/13995 [=======] - 1s 38us/step - loss: 0.0165
Epoch 93/500
13995/13995 [=======] - 1s 39us/step - loss: 0.0161
```

Figura 5.5: Progreso del ajuste del modelo, para el pronóstico de la serie de tiempo de Chen sin perturbaciones comienza la primera época con un error MSE de 58.5593 y se ajustó el modelo hasta llegar a un error MSE de 0.0161 en 93 épocas.

La Figura 5.6(a) muestra la serie de tiempo del sistema Chen, sin perturbaciones, de color negro, el entrenamiento en color verde y el pronóstico en color azul. Se puede observar que el modelo se ajustó de buena manera en el entrenamiento, porque a simple vista no se logra distinguir el color negro que pertenece a la serie de tiempo. Esto significa que el modelo puede seguir la

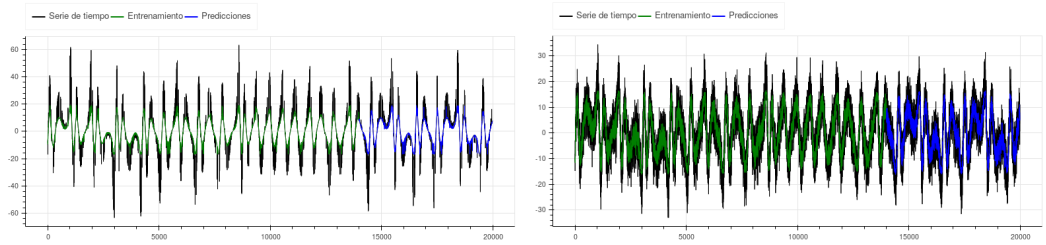
naturaleza de la serie de tiempo, por otra parte la validación del pronóstico de un paso adelante también se ajustó de forma satisfactoria.

La Figura 5.6(b) muestra la serie de tiempo del sistema Chen, perturbada con el 5% de datos faltantes, de color negro. El entrenamiento en color verde y el pronóstico en color azul. Se puede observar que el modelo se ajustó de buena manera en el entrenamiento porque a simple vista no se logra distinguir el color negro, que pertenece a la serie de tiempo. Por otra parte la validación del pronóstico también se ajustó adecuadamente.



(a) 0_0_0 (b) 5_0_0

Figura 5.6: Pronóstico de la serie de tiempo sistema Chen.



(a) 0_5_0 (b) 0_0_5

Figura 5.7: Pronóstico de la serie de tiempo sistema Chen.

La Figura 5.7(a) muestra la serie de tiempo del sistema Chen, perturbada con el 5% de valores atípicos, de color negro, el entrenamiento en color verde y el pronóstico en color azul. Se puede observar que el modelo se ve afectado porque ahora se puede ver el color negro de la serie de tiempo. El ajuste es malo para el entrenamiento por lo tanto también la validación se ve afectada ya que el pronóstico es malo y no se ajusta a la serie de tiempo.

La Figura 5.7(b) muestra la serie de tiempo del sistema Chen, perturbada con un nivel de ruido de $5dB$ de color negro, el entrenamiento en color verde y

el pronóstico en color azul. Se puede observar que el modelo se ve seriamente afectado por el ruido, porque no se logró ajustar de buena manera en el entrenamiento, porque a simple vista se logra ver el color negro que pertenece a la serie de tiempo. Esto significa que el modelo no consiguió seguir la naturaleza de la serie de tiempo. Por otra parte, al igual que el entrenamiento, la validación del pronóstico no logró ajustar.

Los errores de pronóstico SMAPE encontrados en las series de tiempo indican, que entre más ruido SNR se presenta en la serie de tiempo el pronóstico tiende a ser malo (no hace buenas predicciones), dependiendo del nivel de ruido que presente. También se encontró que la presencia de los valores atípicos afecta al pronóstico en menor manera que el ruido, y por último el que existan valores faltantes afecta en menor manera. La combinación de las perturbaciones afecta en diferente medida al pronóstico, porque depende de la misma naturaleza de la serie de tiempo; por ejemplo para la misma serie de tiempo sistema Chen perturbada con 25% de datos faltantes, 25% de valores atípicos y un nivel de ruido de $5dB$, da un error SMAPE de entrenamiento de 114.52 y en la validación un error SMAPE de 89.42. Estos son los porcentajes y niveles máximos de perturbación y se espera que con ellos se tengan los errores más grandes pero no es así; se encontró que con las perturbaciones 0% de datos faltantes, 25% de valores atípicos y un nivel de ruido de $5dB$, da un error SMAPE de entrenamiento de 149.65 y en la evaluación un error SMAPE de 96.96. Estos errores son mayores con niveles menores de perturbación en datos faltantes. Así como este ejemplo existen muchos, es por eso que la combinación de las perturbaciones da diferentes resultados dependiendo la naturaleza de la serie.

5.4. Regresión

Para diseñar la regresión de error SMAPE del pronóstico se utilizaron las características extraídas y el error SMAPE que se obtuvo del pronóstico con cada serie. Una parte importante que se debe mencionar es que el valor de SNR cuando su valor es $5dB$ perturba más fuerte a la serie de tiempo que cuando su valor es de $25dB$; para mantener el sentido con las otras variables que la perturbación crece cuando el porcentaje es mayor, ahora se cambia el valor SNR por 1/SNR. Por ejemplo, cuando el valor es de $5dB$ ahora tendrá un valor de 0.2 y cuando valga 25 tendrá un valor de 0.04; con este cambio se mantiene el sentido con las otras variables. En la Tabla 5.5 se muestra la

Tabla 5.5: Características extraídas y el error de pronóstico

Serie de tiempo	Extracción de características				Error de pronóstico
No perturbadas	Datos faltantes	Valores atípicos	1/SNR	Caos	SMAPE pronóstico
Sistema Chen	0	0	0.013	0.001	1.12
Oscilador Duffing	0	0	0	0.0004	4.71
Atractor cíclico simétrico de Halvorsen	0	0	0	0.0001	3.17
Atractor de Lorenz	0	0	0.019	0.0085	7.05
Atractor de Rössler	0	0	0.012	0.0006	1.00
Atractor de Rucklidge	0	2	0	0.0009	2.04
Oscilador Shawn-van der Pol	0	0	0	0.0004	2.95
Flujo cúbico más simple	0	0	0	0.0002	2.70
Flujo lineal por partes más simple	0	0	0	0.00003	3.30
Seno	0	0	0	0.0721	8.62

extracción de las características y el error SMAPE para las series de tiempo sin perturbaciones (0_0_0). En la serie de tiempo del sistema Chen se extrajo el 0% de datos faltantes lo cual es correcto porque no se perturbo, el 0% de valores atípicos también es correcto, el 0.013 de 1/SNR es muy cercano a 0, el caos es igual a 0.001 el cual indica la presencia de caos en la serie de tiempo; finalmente el error SMAPE 1.12 corresponde al pronóstico. En la serie de tiempo Atractor de Rucklidge se detectó el 2% de los datos como valores atípicos, esto es porque la serie de tiempo presenta picos y estos son

los detectados como valores atípicos. Por otro lado el mejor modelo que se encontró fue el que predice la serie de tiempo del Atractor de
Rössler con un error SMAPE de 1 y por otro lado el modelo con más error es el que predice la serie de tiempo seno con 8.62.

La Tabla 5.5 solo muestra las características y error de predicción de las series de tiempo sin perturbación, pero para generar el regresor de bosques aleatorios se utilizaron todas las series de tiempo tanto perturbadas como no perturbadas, las cuales suman un total de 2,160 series de tiempo. La tabla de datos completa contiene las 4 columnas de las características y otra columna para el error SMAPE y 2,160 renglones de las series de tiempo, los cuales se intercambian de manera aleatoria. Para entrenar al regresor de bosques aleatorios el primer 70% de los datos se usan para el entrenamiento y el restante 30% se usa para la validación del modelo.

Utilizando el Algoritmo 10 con parámetros de tamaño del bosque aleatorio de 1,000 árboles de decisión, que se hiciera el análisis de la importancia de las variables y se indicó que se usara el 70% de los datos para el entrenamiento y el 30% restante para validación. Una vez ejecutado se obtuvieron los siguientes resultados: en la Figura 5.8 se muestran los primeros 10 renglones de la tabla de datos de las características y el error cuando se cargan se puede observar que los índices se encuentran ordenados desde 0 hasta 9. Posteriormente se intercambian los índices y se vuelven a mostrar los primeros 10 renglones de la tabla de datos. Se puede observar que ahora los índices no se encuentran ordenados (el primer índice es el 1,095, el segundo índice es el 1,966 etc.) Después se muestran las dimensiones de los conjuntos el primero es de entrenamiento el cual tiene 1,512 renglones y 4 columnas de las características. El segundo conjunto es el objetivo del entrenamiento el cual tiene 1,512 renglones y una columna el error "SMAPE pronóstico". El tercer conjunto son la características de validación el cual tiene 648 renglones y las 4 columnas de las características. El cuarto y último conjunto es el objetivo de la validación el cual tiene 648 renglones y una columna el error "SMAPE pronostico".

Después que se prepararon los datos se hace el ajuste del modelo del regresor de bosque aleatorio el cual para arroja los siguientes errores al momento de validación.

Error absoluto medio: 5.969
Error rmse 8.479
PrecisionR: 85.134 %.

```
Base de datos de caracteristicas y error
    Datos Faltantes  Valores Atípicos     1/SNR      Caos  smape pronostico
0                 0                 0  0.012627  0.001054          1.123105
1                 0                 0  0.039752  0.000761         15.959387
2                 0                 0  0.049423 -0.000214         24.011332
3                 0                 2  0.065870 -0.000283         36.160149
4                 0                 5  0.097306  0.000244         55.047501
5                 0                 6  0.188165  0.000052         77.953079
6                 0                 3  0.120126 -0.000276         16.034263
7                 0                 3  0.125379 -0.001319         26.786634
8                 0                 4  0.125767 -0.000345         31.806376
9                 0                 6  0.134303 -0.000545         44.889877
Base de datos de caracteristicas y error con intercambio de indices
       Datos Faltantes  Valores Atípicos     1/SNR      Caos  smape pronostico
1095                0                 3  0.180795  0.000847         63.266472
1966                0                 3  0.386113 -0.000502         74.554953
1886               20                 3  0.164008  0.000266         36.655863
192                25                 4  0.158461  0.001595         27.546259
990                15                 4  0.184357  0.001452         35.856020
845                25                 6  0.451388  0.004613         84.156365
1162               10                 2  0.160315 -0.000220         72.382831
832                25                 2  0.103054  0.003120         54.558102
938                10                 0  0.048353  0.000173         21.728850
122                15                 4  0.166635 -0.000212         39.501453
Entrenamiento caracteristicas dimensiones: (1512, 4)
Entrenamiento objetivo dimensiones: (1512,)
Validacion caracteristicas dimensiones: (648, 4)
Validacion objetivo dimensiones: (648,)
```

Figura 5.8: Ejecución del programa.

Para el análisis de la importancia de las variables reportó los siguientes resultados.

Variable: 1/SNR Importance: 0.82
Variable: Caos Importance: 0.1
Variable: Valores Atípicos Importance: 0.05
Variable: Datos Faltantes Importance: 0.02

De la Figura 5.9 se puede observar que 2 características son las más importantes las cuales son 1/SNR y Caos. Como parte del análisis se creo un nuevo modelo de regresor de bosques aleatorios con los mismos parámetros que el anterior pero ahora con tan solo estas 2 características los errores que arroja este modelo en la validación son.

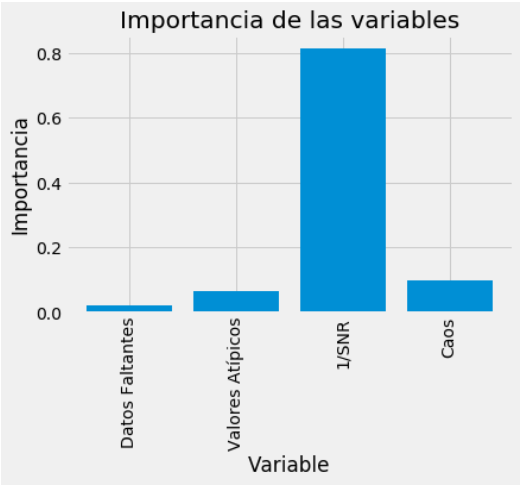


Figura 5.9: Variables importantes

Error absoluto medio: 7.36
PrecisionR: 82.93 %.

Se observa que aumentó el error absoluto medio de 5.9685 a 7.36 y por lo tanto bajo la precisión de 85.1336% a 82.93%, lo que corresponde a un 2.2036%. Esto indica que si se desea se puede prescindir de las características de datos faltantes y de valores atípicos. Los resultados arrojados por los modelos de bosques aleatorios indican que se puede tener un buen regresor con solo dos variables.

En este capítulo se mostraron los resultados obtenidos, se mostró como fueron realizadas las perturbaciones en las series de tiempo, con diferentes porcentajes de datos faltantes, valores atípicos y niveles de ruido. Después que se perturbaron las series de tiempo, se realizó la extracción de cada una de las características con los diferentes algoritmos, en la extracción el mejor desempeño lo tuvo la detección de datos faltantes, la detección de nivel de ruido mostró un buen desempeño, y la detección de valores atípicos mostró un bajo desempeño porque no es fácil detectar los valores atípicos locales. Después de extraer las características se realizó el pronóstico con los mejores modelos de los perceptrones multicapa, se observo que el mejor pronóstico se obtiene cuando los datos no presentan perturbaciones, el pronóstico se ve afectado fuertemente cuando el nivel de ruido es de 5dB. Por último el regresor de bosques aleatorios con 1,000 árboles de decisión, fue entrenado con las

características extraídas y los errores SMAPE de pronóstico, la precisión que se obtuvo con el regresor fue de 85.133 usando las cuatro características, y una precisión de 82.93 % usando las características de ruido y caos. Los resultados obtenidos en esta tesis son satisfactorios porque se logró el objetivo de determinar la calidad de los datos de la serie de tiempo, modelando el error de pronóstico. Los datos de características, errores de predicción y parámetros para generar las tablas se encuentran disponibles en la siguiente liga https://drive.google.com/open?id=1xWFdtCORcWex4yaqdrfsEVJIvjnxv7a9.

Capítulo 6

Conclusiones y trabajos futuros

De las 10 series de tiempo que se eligieron para hacer este análisis 9 son caóticas con un tamaño de 20,000 datos estas series de tiempo se encuentran en la Tabla 2.1. Esta elección se hizo con el conocimiento previo de que las series caóticas son de las series de tiempo más difíciles de predecir por su comportamiento irrepetible. La serie de tiempo de la función seno se eligió para poder observar de una manera más simple el comportamiento que estaban teniendo los diferentes algoritmos que se utilizaron para el análisis de la calidad de los datos.

La perturbación de las series de tiempo se hizo en 3 características: datos faltantes, valores atípicos y ruido. Para perturbar la serie con datos faltantes se cambiaron los valores de manera aleatoria por el valor NaN que indica dato faltante. Para la perturbación con valores atípicos se uso el Algoritmo 6 $agregarValoresAtipicos$ lo que hace es multiplicar por 2 el valor del dato que se va a perturbar. Para la perturbación con ruido SNR se implemento en Python el Algoritmo 8 $agregarRuido$ que implementa la función AWGN de la caja de herramientas de MATLAB [Matlab, 2015].

Para las series de tiempo perturbadas con las diferentes características se creó la siguiente codificación (f_a_r; 0_5_0, 0_0_5), donde f corresponde al porcentaje de los datos faltantes, a corresponde al porcentaje de los valores atípicos y r corresponde al nivel de ruido en dB con el que se perturbó la serie de tiempo. De las Figuras 5.1 y 5.2 se muestra como afecta de diferentes formas las perturbaciones. Para la perturbación con los datos faltantes se observa como las Figuras 5.1(b) y 5.2(b) tienen huecos, para la perturbación con valores atípicos se observa que las Figuras 5.1(c) y 5.3(b) presentan picos, para la perturbación con ruido se observa en las Figuras 5.1(d) y 5.4 que es

muy fuerte la perturbación.

La extracción de características se llevó a cabo en todas las series de tiempo tanto las perturbadas como en las no perturbadas. Las características que se extraen son cuatro: datos faltantes, valores atípicos, ruido y caos. La característica que se extrae de manera sencilla y sin error es la de datos faltantes y esto es porque simplemente se revisa que se encuentre el dato y si no está se cuenta. El ruido es otra característica que se extrae de forma correcta ya que los errores mostrados en la Tabla 5.3 son pequeños y entre más pequeño sea el error mejor es la extracción. Los valores atípicos son muy difíciles de extraer porque basta que un conjunto de datos atípicos se encuentren cerca para no ser detectados. Por este motivo es que se vio afectado el desempeño, como lo muestra la Tabla 5.2, donde los errores son grandes. No se puede analizar el desempeño de la extracción del caos porque esta característica no se agregó manualmente a la serie de tiempo. La extracción de estas cuatro características para una serie de tiempo de tamaño 20,000 toma aproximadamente 110 segundos. por lo cual para extracción de las características de las 2,160 series de tiempo tomó aproximadamente 66 horas.

Se encontró que en todas las series que presentan ruido los modelos producen un mal pronostico, dependiendo del nivel de ruido que presente. También se encontró que la presencia de los valores atípicos afecta al pronóstico en menor manera que el ruido, y por último el que existan valores faltantes afecta en menor manera. La combinación de las perturbaciones afecta en diferente medida el pronóstico, porque depende de la misma naturaleza de la serie de tiempo, por ejemplo para la misma serie de tiempo (sistema Chen) perturbada con 25% de datos faltantes, 25% de valores atípicos y un nivel de ruido de $5dB$, da un error MSE de entrenamiento de 112.08 y en la validación un error MSE de 114.53. Estos son los porcentajes y niveles máximos de perturbación que se usaron se espera que con ellos se tengan los errores más grandes pero no es así, se encontró que con las perturbaciones 0% de datos faltantes, 25% de valores atípicos y un nivel de ruido de $5dB$, da un error MSE de entrenamiento de 143.54 y en la validación un error MSE de 149.66. Estos errores son mayores con un nivel bajo de perturbación en datos faltantes, y así como este ejemplo existen muchos, por eso la combinación de las perturbaciones da diferentes resultados dependiendo la naturaleza de la serie.

La regresión se llevó a cabo usando un método de aprendizaje máquina llamado regresor de bosques aleatorios. El modelo de este regresor contiene 1,000 árboles de decisión que conforman el bosque aleatorio. El regresor se

ajusto utilizando el 70% de la tabla de datos de las características y el error como entrenamiento y se validó con el 30% restante. Después que el modelo del regresor se ajustó se llevó a cabo un análisis de la importancia de las variables. En la Figura 5.9 se observa que dos de las cuatro características son las más importantes: 1/SNR y Caos. Como parte del análisis se diseño un nuevo modelo de regresor de bosques aleatorios con los mismos parámetros que el anterior, pero ahora con solo las 2 características más importantes. Comparando los resultados el regresor con las cuatro características presenta un error MAE de 5.9685 en la validación y una precisión del 85.1336%. El regresor con las dos características más importantes presenta un error MAE de 7.36 en la validación y una precisión de 82.93. De este análisis se concluye que extrayendo las dos características más importantes se puede tener una buena aproximación al error que puede presentar una serie de tiempo en su pronóstico. Pero si se extraen las cuatro características se obtiene una mejor aproximación al error.

Para conocer la calidad de los datos de una serie de tiempo con esta tesis, se requiere de un tiempo de 110s aproximadamente para la extracción de características (para una serie de tiempo con 20,000 datos). Adicionalmente se tiene el tiempo de 0.5s aproximadamente para la regresión del error en total se necesita 1 minuto y 50.5 segundos para obtener una estimación del error que un modelo neuronal produciría. El tiempo para estimar la precisión del pronóstico sin hacer modelado es muy poco comparado con los días o semanas que se requieren para realizar el modelado y hacer las pruebas con diferentes métodos y posteriormente obtener la precisión del pronóstico y decidir si la calidad de los datos ha sido buena o mala.

6.1. Trabajos futuros

Las mejoras que se pueden realizar al trabajo presentado en esta tesis es utilizar series de tiempo reales de diferentes áreas; la energía eléctrica en esta área se encuentran diferentes tipos de series de tiempo (viento, solar, generación, distribución, crecimiento de red, etc.). En economía y marketing (empleo, desempleo, precios de productos, ventas, etc.). En demografía (número de habitantes, tasa de mortalidad, etc.), medio ambiente (temperatura, lluvia, etc.). Teniendo estas diferentes series de tiempo, amplía el espacio para calcular la calidad de alguna serie de tiempo.

Agregar la normalización de los datos es otra mejora que puede aportar

más precisión en el error de pronóstico.

Ampliar la combinación de las perturbaciones es otra mejora, en esta tesis se usaron perturbaciones con niveles de (0, 5, 10, 15, 20 y 25), y se puede incrementar el número de estos niveles, se plantea tener niveles comenzando en 1 hasta 40 con incrementos de 1. estos son 40 niveles de perturbación y combinándolos se generarían 40x40x40 = 640,000 diferentes series de tiempo. Seguir el mismo proceso hasta ajustar el nuevo regresor, con lo cual se espera que el error del pronóstico mejore.

Agregar nuevas características es otra mejora que se puede realizar, agregar cuatro características; el valor máximo, el valor mínimo, la media y la desviación estándar. Para tener un total de 8 características, se espera que proporcionen más información y ajustar mejor el modelo del regresor, hacer un análisis de importancia de las variables y ver como impacta en el error del regresor.

Incluir diferentes métodos de pronóstico de series de tiempo es otra mejora que se puede realizar, incluir los métodos; redes neuronales recurrentes LSTM, vecinos cercanos y ARIMA. Se espera que agregando estos métodos se mejore la predicción para cualquier tipo de serie de tiempo.

Agregar algoritmos evolutivos para encontrar los mejores parámetros para el diseño de los modelos de perceptrones multicapa, con esto se garantiza tener el mejor modelo para cada serie de tiempo.

Hacer un preprocesamiento de los datos, para todas las características y comparar el pronóstico de la serie de tiempo sin preprocesamiento y con preprocesamiento. Se espera que con este preprocesamiento la calidad de la serie de tiempo mejore y por lo tanto la precisión del pronóstico.

Referencias

[Azar and Vaidyanathan, 2016] Azar, A. T. and Vaidyanathan, S. (2016). *Advances in Chaos Theory and Intelligent Control*. Springer Publishing Company, Incorporated, 1st edition.

[Breiman, 2001] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

[Breunig et al., 2000] Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander. (2000). Lof: Identifying density-based local outliers. *SIGMOD Rec.*, 29(2):93–104.

[Campos et al., 2016] Campos, G. O., Zimek, A., Sander, J., Campello, R. J. G. B., Micenková, B., Schubert, E., Assent, I., and Houle, M. E. (2016). On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. *Data Mining and Knowledge Discovery*, 30(4):891–927.

[Chen and Ueta, 1999] Chen, G. and Ueta, T. (1999). Yet another chaotic attractor. *International Journal of Bifurcation and Chaos - IJBC*, 9:1465–1466.

[Chollet et al., 2015] Chollet, F. et al. (2015). Keras. `https://keras.io`.

[Czesla et al., 2018] Czesla, S., Molle, T., and Schmitt, J. H. M. M. (2018). A posteriori noise estimation in variable data sets. With applications to spectra and light curves. *aap*, 609:A39.

[Demuth et al., 2014] Demuth, H. B., Beale, M. H., De Jess, O., and Hagan, M. T. (2014). *Neural Network Design*. Martin Hagan, USA, 2nd edition.

[Douglas C. Montgomery and Kulahci, 2015] Douglas C. Montgomery, C. L. J. and Kulahci, M. (2015). *Introduction to Time Series Analysis and Forecasting, 2nd Edition*. Wiley Series in Probability and Statistics.

[Flores et al., 2016] Flores, J. J., Calderon, F., Gonzalez, J. R. C., Ortiz, J., and Farias, R. L. (2016). Comparison of time series forecasting techniques with respect to tolerance to noise. In *2016 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, pages 1–6.

[Galton, 1886] Galton, F. (1886). Regression towards mediocrity in hereditary stature. *The Journal of the Anthropological Institute of Great Britain and Ireland, Vol. 15, pp. 246-263.*

[George E. P. Box, 2015] George E. P. Box, Gwilym M. Jenkins, G. C. R. G. M. L. (2015). *Time Series Analysis: Forecasting and Control, 5th Edition.* Wiley Series in Probability and Statistics.

[Han et al., 2012] Han, J., Kamber, M., and Pei, J. (2012). 12 - outlier detection. In Han, J., Kamber, M., and Pei, J., editors, *Data Mining (Third Edition)*, The Morgan Kaufmann Series in Data Management Systems, pages 543 – 584. Morgan Kaufmann, Boston, third edition edition.

[Hawkins, 1980] Hawkins, D. (1980). *Identification of outliers.* Chapman and Hall.

[Kennel et al., 1992] Kennel, M. B., Brown, R., and Abarbanel, H. D. I. (1992). Determining embedding dimension for phase-space reconstruction using a geometrical construction. *Phys. Rev. A*, 45:3403–3411.

[Li and Yorke, 1975] Li, T.-Y. and Yorke, J. A. (1975). Period three implies chaos. *The American Mathematical Monthly*, 82(10):985–992.

[Liu, 2010] Liu, Z. (2010). Chaotic time series analysis. *Mathematical Problems in Engineering.*

[Matlab, 2015] Matlab (2015). *MATLAB version 8.5.0.197613 (R2015a).* The Mathworks, Inc., Natick, Massachusetts.

[McKinney et al., 2010] McKinney, W. et al. (2010). Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot,

M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 12:2825–2830.

[PyA, 2019] PyA (2019). Pyastronomy documentation. Technical report.

[Raileanu and Stoffel, 2004] Raileanu, L. and Stoffel, K. (2004). Theoretical comparison between the gini index and information gain criteria. *Annals of Mathematics and Artificial Intelligence*, 41:77–93.

[Rosenstein et al., 1993] Rosenstein, M. T., Collins, J. J., and Luca, C. J. D. (1993). A practical method for calculating largest lyapunov exponents from small data sets. *Physica D: Nonlinear Phenomena*, 65(1):117 – 134.

[Rössler, 1976] Rössler, O. (1976). An equation for continuous chaos. *Physics Letters A*, 57(5):397 – 398.

[Sato et al., 1987] Sato, S., Sano, M., and Sawada, Y. (1987). Practical Methods of Measuring the Generalized Dimension and the Largest Lyapunov Exponent in High Dimensional Chaotic Systems. *Progress of Theoretical Physics*, 77(1):1–5.

[Schapire, 1990] Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, 5(2):197–227.

[Scholzel,] Scholzel, C. *Nolds Nonlinear measures for dynamical systems*.

[Sprott, 2003] Sprott, J. C. (2003). *Chaos and Time-Series Analysis*. Oxford University Press, Inc., New York, NY, USA.

[Sprott and J Linz, 2000] Sprott, J. C. and J Linz, S. (2000). Algebraically simple chaotic flows. *International Journal of Chaos Theory and Applications*, 5:1–20.

[Takens, 1981] Takens, F. (1981). Detecting strange attractors in turbulence. In Rand, D. and Young, L.-S., editors, *Dynamical Systems and Turbulence, Warwick 1980*, pages 366–381, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Wolf et al., 1985] Wolf, A., Swift, J. B., Swinney, H. L., and Vastano, J. A. (1985). Determining lyapunov exponents from a time series. *Physica D: Nonlinear Phenomena*, 16(3):285 – 317.

[Zhao et al., 2018] Zhao, Q., Zhu, Y., Wan, D., Yu, Y., and Cheng, X. (2018). Research on the data-driven quality control method of hydrological time series data. *Water*, 10(12).