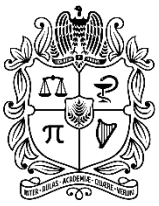


UNIVERSIDAD
NACIONAL
DE COLOMBIA

scikit-forecasts: Una librería en Python para el pronóstico de series de tiempo no lineales

María Alejandra Arango González

Universidad Nacional de Colombia

Facultad de Minas, Área Curricular de Sistemas e Informática

Medellín, Colombia

2021

scikit-forecasts: Una librería en Python para el pronóstico de series de tiempo no lineales

María Alejandra Arango González

Tesis de Maestría presentada como requisito parcial para optar al título de:

Magister en Ingeniería de Sistemas

Director:

Juan David Velásquez Henao, PhD

Línea de Investigación:

Analítica

Universidad Nacional de Colombia

Facultad de Minas, Área Curricular de Sistemas e Informática

Medellín, Colombia

2021

The saddest aspect of life right now is that science gathers knowledge faster than society gathers wisdom.

Isaac Asimov

Declaración de obra original

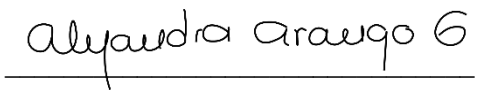
Yo declaro lo siguiente:

He leído el Acuerdo 035 de 2003 del Consejo Académico de la Universidad Nacional. «Reglamento sobre propiedad intelectual» y la Normatividad Nacional relacionada al respeto de los derechos de autor. Esta disertación representa mi trabajo original, excepto donde he reconocido las ideas, las palabras, o materiales de otros autores.

Cuando se han presentado ideas o palabras de otros autores en esta disertación, he realizado su respectivo reconocimiento aplicando correctamente los esquemas de citas y referencias bibliográficas en el estilo requerido.

He obtenido el permiso del autor o editor para incluir cualquier material con derechos de autor (por ejemplo, tablas, figuras, instrumentos de encuesta o grandes porciones de texto).

Por último, he sometido esta disertación a la herramienta de integridad académica, definida por la universidad.



María Alejandra Arango González

Fecha 30/04/2021

Agradecimientos

A Casa y todo el equipo, por su apoyo siempre; al profesor Juan David, por su guía y paciencia; a Sofía y a Lukas.

Resumen

scikit-forecasts: Una librería en Python para el pronóstico de series de tiempo no lineales

El interés en el uso de técnicas de aprendizaje de máquinas, y en general, de modelos no lineales para el pronóstico de series de tiempo ha crecido exponencialmente en las últimas dos décadas. Sin embargo, muchas de las librerías disponibles de aprendizaje de máquinas no contemplan el uso de modelos de series de tiempo, haciendo que el científico de datos consuma gran parte de su tiempo en la tarea de convertir sus datos a un formato de problema de regresión para poder utilizar estas librerías. Esto claramente evidencia la necesidad de contar con librerías especializadas en el pronóstico de series de tiempo usando técnicas de aprendizaje de máquinas y modelos no lineales en general. En esta tesis se presenta la librería de Python de código abierto llamada scikit-forecasts, la cual permite transformar y pronosticar series de tiempo usando técnicas de aprendizaje de máquinas, entre las que se incluyen los modelos autorregresivos, las redes neuronales artificiales, modelos neuro-difusos, modelos TAR y modelos SETAR, entre otros. La librería puede ser usada interactivamente en un libro de Jupyter, lo que facilita el desarrollo de modelos. Los métodos de pronóstico se encuentran implementados como clases que pueden ser utilizadas con muchas de las funciones disponibles en scikit-learn. La librería está diseñada para soportar los procesos de evaluación de distintos tipos de transformaciones, el pronóstico con diferentes tipos de modelos no lineales y la comparación de pronósticos obtenidos con modelos diferentes.

Palabras clave: predicción, series de tiempo, redes neuronales artificiales, no linealidad, aprendizaje de máquinas.

Abstract

scikit-forecasts: A Python package for nonlinear time series forecasting

The significance of using machine learning techniques and nonlinear models for time series forecasting has been grown exponentially in past two decades. Nevertheless, a lot of Python packages and libraries using machine learning methods are not taking in account time series models and making data scientists to consume a lot of their time converting data into a regression problem format in order to use these libraries. This kind of issue remarks a necessity of packages and libraries focused on time series forecasting and the use of machine learning techniques and nonlinear models in general. This thesis introduces an open-source Python package called scikit-forecasts, which allows time series transforming and forecasting by using machine learning techniques such as autoregressive modes, artificial neural networks, neuro-fuzzy models, TAR and SETAR models, among others. This package can be interactively used in Jupyter notebooks, which simplifies and promotes new models' development. Forecasting methods are implemented as classes that can be used with scikit-learn's available functions. This package has been designed to support evaluation processes of different types of transformations, forecasts of several types of nonlinear models and comparisons of forecasting results that have been obtained using different models.

Keywords: forecasting, time series, artificial neural networks, nonlinear, machine learning.

Contenido

Lista de figuras

Pág.

Lista de tablas

Pág.

Lista de símbolos y abreviaturas

Símbolos con letras latinas

Símbolo	Término	Unidad SI	Definición
c	Umbral en las funciones de transición	1	Anexo A
d	Diferenciación del modelo	1	Anexo A
$G(z_t)$	Función de transición del modelo	1	Anexo A
p	Orden de modelos auto regresivos	1	Anexo A
q	Orden de modelos de medias móviles	1	Anexo A
s	Retraso del modelo	1	Anexo A
x_t	Valor de la serie en el tiempo t	1	Anexo A
w_t	Ruido blanco	1	$w_t \sim N(0, \sigma^2)$
z_t	Variables de transición	1	Anexo A

Símbolos con letras griegas

Símbolo	Término	Unidad SI	Definición
α	Parámetro del modelo de suavizado exponencial	1	Anexo A
β	Parámetro del modelo de suavizado exponencial	1	Anexo A
γ	Parámetro del modelo de suavizado exponencial	1	Anexo A
μ	Media de los valores de la serie de tiempo	1	Anexo A
φ	Parámetros de los modelos auto regresivos	1	Anexo A
$\varphi(B)$	Operador auto regresivo	1	Anexo A
σ^2	Varianza de los valores de la serie de tiempo	1	Anexo A
θ	Parámetros de los modelos de medias móviles	1	Anexo A

Abreviaturas

Abreviatura	Término
AR	Auto Regressive
ARMA	Auto Regressive Moving Average
ARCH	Auto Regressive Conditional Heteroskedasticity
ARMA	Auto Regressive Moving Average
ARIMA	Auto Regrresive Integrated Moving Average

Abreviatura	Término
GARCH	Generalized Auto Regressive Conditional Heteroskedasticity
IEEE	Institute of Electrical and Electronics Engineers
MA	Moving Average
MLP	Multi Layer Perceptron
RBF	Radial Basis Function
SARIMA	Seasonal Auto Regressive Integrated Moving Average
SETAR	Self Exciting Threshold Auto Regressive
STAR	Smoothing Threshold Auto Regressive
TAR	Threshold Auto Regressive
TSF	Time Series Forecasting
VAR	Vector Auto Regressive

1. Introducción

El campo del análisis y pronóstico de series de tiempo está comprendido por todas las técnicas que aplicadas a las series de tiempo conducen a mejorar el conocimiento que se tiene de ellas, ya sea para determinar sus propiedades, explicar por qué ocurrió el pasado, o pronosticar sus valores futuros [1], y para lo cual, las herramientas computacionales juegan un papel fundamental. Particularmente, el análisis de series de tiempo se ha usado para el estudio de la estructura dinámica de un proceso, investigar la relación dinámica entre variables, realizar un ajuste temporal de cualquier conjunto de datos, mejorar el análisis de regresiones cuando los errores están correlacionados y generar predicciones de puntos e intervalos [2]. El pronóstico es usado fundamentalmente en procesos de toma de decisiones, como un mecanismo para obtener valores posibles que permitan analizar la robustez de las decisiones.

En el ámbito del desarrollo científico computacional, las nuevas técnicas que han logrado resolver varios de los problemas de altos volúmenes de datos y capacidad física han estado orientadas a la predicción y a la toma de decisiones. Por un largo período de tiempo, el pronóstico de series de tiempo ha sido transversal tanto a las nuevas técnicas – o aquellas reconocidas como nuevas en cada momento - como a las diferentes áreas del conocimiento. Dentro de las tendencias de investigación modernas, se hace difícil reconocer problemas que no estén relacionados con el tiempo o que no adquieran información a medida que este transcurre: aunque las series de tiempo tuvieron sus inicios en las ciencias sociales y físicas, los conceptos básicos han aparecido en cada área con una consecuente transferencia a la tecnología [1].

Principalmente, el pronóstico de series tiempo ha tenido un alto impacto en la predicción económica y financiera (crecimiento económico, mercado de acciones, control de ventas), pero también se aplica en las ciencias naturales (cambios climáticos, ritmos circadianos, control de plagas) y las ciencias sociales (crecimiento y estudios poblacionales, políticas públicas). Esta diversidad ha impulsado el desarrollo de una gran cantidad de software y paquetes estadísticos dedicados al pronóstico, al igual que algoritmos creados sólo con este fin. Gran parte de la teoría y la investigación

actual de las series de tiempo está orientada a solucionar los problemas relacionados con el pronóstico simultáneo de un alto volumen de series, las cuáles pueden tener un alto volumen de datos.

Aunque hay grandes avances en el desarrollo de algoritmos de pronóstico y en la capacidad computacional, tanto en hardware como en el software que aprovecha estos avances, la propagación de estos desarrollos está sujeta a muchas limitaciones. Una de las principales limitantes para entender cómo fueron obtenidos los resultados de un estudio y realizar su comparación con otros métodos, es que los algoritmos pueden estar implementados en diferentes lenguajes de programación o en herramientas computacionales, que inclusive no pueden estar disponibles actualmente. Además, muchos de los software usados en la investigación científica son privados o comerciales dificultando también su reproducibilidad [3]. Últimamente, muchos de los esfuerzos en el desarrollo computacional buscan reducir estas limitaciones. El desarrollo de código libre ha sido exponencial desde 1990 [4]. A su vez, este código se va perfeccionando y aceptando tanto en la industria como en las entidades gubernamentales, quienes son los principales promotores de la investigación.

Con lo anterior enunciado, esta tesis presenta un entorno integrado para el pronóstico de series de tiempo que facilite el pronóstico usando modelos predictivos y que permita agregar rápidamente nuevos modelos. Este entorno también busca facilitar la comparación entre diferentes modelos. La herramienta está desarrollada en Python 3 [5] y es una extensión de la librería scikit-learn, ajustándose a algunas de sus características y funcionalidades. Inicialmente cuenta con una serie de modelos entre los que se incluyen los modelos autorregresivos, las redes neuronales artificiales, modelos neuro-difusos, modelos TAR y modelos SETAR, entre otros, pero está principalmente diseñada para crear y usar modelos fácilmente dentro de ella. Cuenta también con un *pipeline* que permite ejecutar varios modelos sobre una o varias series de tiempo, además de una función de búsqueda para hallar el mejor modelo posible para cada una de las series de tiempo bajo estudio.

Este documento está organizado como sigue: en el Capítulo 1 se presenta la motivación y el significado de esta tesis; en el Capítulo 2 se presentan los modelos matemáticos implementados; en el Capítulo 3, la arquitectura de la librería y sus funcionalidades; en el Capítulo 4 se presenta un tutorial dónde se aplican las funcionalidades explicadas en el Capítulo 3 a una serie de tiempo y se remarcan las bondades de la librería; en el Capítulo 5 se muestran dos casos comparativos y de aplicación y, finalmente, en el Capítulo 6 se presentan las conclusiones de toda la investigación y desarrollo de la librería.

1. Motivación y significado

1.1 Motivación

1.1.1 Importancia del pronóstico de series de tiempo

Una serie de tiempo es un conjunto de observaciones sobre los valores que toma una variable (cuantitativa) a través del tiempo. Por tanto, una serie de tiempo es una forma estructurada de representar datos [6]. Estos datos pueden comportarse de manera diferente: pueden presentar tendencias, estacionalidades o incluso no tener una forma definida.

Una característica fundamental de las series de tiempo es que las variables no son independientes entre sí y el análisis debe tener en cuenta el orden temporal de las observaciones [7].

Usualmente las series de tiempo se utilizan para predecir el comportamiento futuro de la variable medida. Este pronóstico ha tenido un alto impacto en la predicción económica y financiera, pero también se usa en las ciencias naturales y las ciencias sociales. Las aplicaciones varían desde predicciones económicas y econométricas, de nuevos productos y ventas, financieras (incluyendo los mercados de acciones), de energía y medio ambiente (cambios climáticos, ritmos circadianos, control de plagas), demográficas y poblacionales, hasta tecnológicas y relacionadas a las políticas públicas como el crimen.

1.1.2 Diferencia entre modelado y pronóstico de series de tiempo

Las series de tiempo están compuestas por un componente básico que es posible de modelar (o representar en un modelo) y un componente aleatorio.

El modelado consiste en realizar un estudio descriptivo de la serie de tiempo el cual se basa en encontrar sus componentes. Los componentes que se consideran habitualmente son: la tendencia, la

cuál se puede definir como un cambio a largo plazo, un efecto estacional, el cuál es una periodicidad o variación que se presenta cada cierto período y la componente aleatoria que se describe con algún tipo de modelo probabilístico [7].

Cuando se observan los valores de una serie, se pretende normalmente no sólo explicar el pasado, sino también predecir el futuro. Para esta predicción, se combinan los componentes de la serie y se encuentran uno o más valores en el futuro. A esto se le llama pronóstico de series de tiempo.

Existen dos diferentes aproximaciones al modelado y pronóstico de series de tiempo. La primera de ellas es una aproximación estadística y la segunda, una aproximación desde el aprendizaje de máquinas.

1.1.3 Ventajas y limitaciones de las herramientas disponibles para modelado de series de tiempo

Los métodos estadísticos basados en la independencia de las observaciones no son válidos para el análisis de series de tiempo porque las observaciones en un instante de tiempo dependen de los valores de la serie en el pasado.

Desde una aproximación estadística, el modelo que representa la serie de tiempo debe cumplir unos supuestos. El primero de ellos es que la autocorrelación entre los puntos de la serie es cero (representado en la Ecuación 1). El segundo es que los errores siguen una distribución normal representado en la Ecuación 2.

$$\rho_j = \frac{cov(X_j, X_{j-k})}{\sqrt{V(X_j)}\sqrt{V(X_{j-k})}} = 0 \quad (1)$$

$$w_t \sim N(0, \sigma_w^2) \qquad (2)$$

Bajo estos supuestos, el mejor modelo es el que logra minimizar el error como se muestra en la Ecuación 3.

$$Min\ (MSE = \frac{1}{n}\sum_{i=1}^{n}\left(\widehat{X_i} - X_I\right)^2) \quad (3)$$

El formalismo que requiere esta aproximación puede dar información muy importante para la mejora del modelo (desde la matemática intentar entender cómo funciona el mundo), sin embargo, esto también conlleva algunas limitaciones. Para este formalismo se deben tener muy bien definidos los tipos de problemas y una estructura clara, por ejemplo, para el preprocesamiento de series de tiempo.

Además, sólo existen algunos tratamientos para datos atípicos y *outliers*, lo cual hace difícil el modelado y pronóstico cuando este tipo de datos se presentan.

1.1.4 Ventajas y limitaciones de las herramientas disponibles en Machine Learning

Los modelos de aprendizaje de máquinas cuentan con las siguientes características:

- La filosofía fundamental es la precisión del modelo, la verificación de las propiedades estadísticas de los errores no es fundamental ni se acostumbra a hacerlo.
- El modelo de pronóstico se suele mirar como si fuera un problema de regresión (lineal o no lineal)
- La aproximación de regresión tiene unos elementos interesantes desarrollados en el aprendizaje de máquinas: métricas de error diferentes al MSE, modelos no lineales propios de AI (Fuzzy, Neuro-Fuzzy, ANN)

Estas características facilitan el modelado y pronóstico de las series de tiempo, sin embargo, las herramientas disponibles en aprendizaje de máquinas siguen teniendo la limitación de estar orientadas únicamente a modelos de regresión y clasificación, los cuáles son posibles de usar después de un largo proceso de desarrollo del modelo.

1.2 Formulación del problema

Los desarrollos en el análisis y pronóstico de series de tiempo no lineales han permitido pronosticar, modelar y comprender fenómenos naturales (clima, alimentación y desastres), económicos (las empresas y las políticas macroeconómicas) y del desarrollo –más recientemente ligados a la ecología, medio ambiente y consumo de energía–, pero siempre existe la necesidad de incorporar las

metodologías de análisis y pronóstico en nuevos campos, y a su vez, generar nuevas metodologías que permitan una comprensión más profunda de los fenómenos y que modelen mejor las relaciones entre las variables. Por otra parte, el investigador y el profesional se ven enfrentados a la necesidad de integrar los nuevos desarrollos y modelos propuestos en la literatura más reciente a las metodologías existentes, lo que no resulta tan fácil en muchos casos, debido a que estos se encuentran en diferentes etapas de investigación o implementados en diferentes entornos o lenguajes de programación (que en muchos casos son comerciales o que no son fácilmente accesibles), dificultando enormemente esta tarea.

Desde un punto de vista práctico, es necesario que los profesionales e investigadores que usan metodologías de series de tiempo tengan a su disposición herramientas y librerías que integren tanto las metodologías tradicionales para pronóstico de series de tiempo, como métodos menos explorados y más modernos (como los modelos más recientes que provienen de las áreas de *machine learning*, *deep learnin*g, y que incluyen modelos híbridos, lo cual puede dificultar el proceso de desarrollo de un modelo final). La integración de estos desarrollos, flexibilizando los entornos de trabajo para poder mezclar metodologías y modelos, es una necesidad primordial para alcanzar una mayor precisión en los pronósticos. Finalmente, es importante romper barreras para que cualquier investigador acceda a un ambiente integrado de trabajo y realice sus propias contribuciones de una manera más eficiente, sin tener que desgastarse en realizar desarrollos de modelos bien conocidos o integrar resultados producidos en entornos incompatibles.

La librería scikit-forecasts está desarrollada con el propósito de generar un entorno de trabajo con metodologías integradas y hacerlo fácilmente accesible, más amable y bajo una estructura en la que el investigador y el profesional del área pueda integrar un modelo (o usar diferentes modelos) para pronosticar una o más series de tiempo, tal como ocurre usualmente en la práctica.

1.3 Hipótesis

Hasta el momento no existen herramientas computacionales que incorporen simultáneamente técnicas de aprendizaje de máquinas y técnicas de estadística que permitan encontrar (obtener) los beneficios de ambas metodologías. Además, los principales *frameworks* de *machine learning* están orientados únicamente a modelos de regresión y clasificación, aunque si se puede utilizar, se debe pedalear el proceso de desarrollo de modelo.

Se requiere una herramienta en la que sea fácil construir modelos (desde cualquier aproximación), que permita fácilmente comparar los resultados de cada uno de estos modelos y además que tenga la facilidad de extenderse. Esta tendría el potencial de ser usada para la implementación de nuevas metodologías.

1.4 Objetivos

El objetivo general de esta librería es el desarrollo de un entorno integrado para el pronóstico de series de tiempo no lineales que facilite la inserción de nuevos modelos dentro de la herramienta.

Los objetivos específicos son:

1. Realizar el diseño, implementación y prueba de la librería bajo el esquema de scikit-learn.
2. Implementar el modelo ARIMA.
3. Implementar un modelo de redes neuronales artificiales.
4. Implementar un modelo híbrido neuro-difuso.

1.5 Importancia del software

El desarrollo en los campos de modelado y pronóstico avanza a un ritmo acelerado y cuenta con una gran comunidad, haciendo posible que el número de publicaciones que señalan avances, usos y aplicaciones crezca rápidamente. Sin embargo, muchas de las publicaciones presentan metodologías o casos de aplicación desarrollados en entornos de programación cuya configuración no es fácilmente reproducible (o imposible de reproducir), y en muchos casos, el código no es de libre acceso. Es así, que se hace necesario ir más allá de señalar un avance y una aplicación y generar entornos que permitan reproducir los resultados, usar las metodologías en nuevos casos, y descubrir errores.

Por otra parte, las nuevas técnicas para análisis de datos (*machine* y *deep learning*) están siendo desarrolladas principalmente en Python. Este lenguaje alcanzó el primer lugar de los lenguajes de programación más usados según el ranking de la IEEE en el 2017 [8] y se ha mantenido allí desde entonces. Aunque Python ha logrado grandes avances en el procesamiento de datos con librerías

como Pandas [9], las herramientas para el análisis y pronóstico de series de tiempo no han sido ampliamente desarrolladas.

Tanto en Python como en otro tipo de software (comercial, en línea), no ha sido posible la reproducción de los resultados, la replicabilidad y el análisis colaborativo en el campo de las series de tiempo, por las razones ya expuestas. Es por esto que la librería scikit-forecasts es desarrollada —sobre un lenguaje de programación libre, de amplio uso, con posibilidades en técnicas modernas– con el propósito de generar un entorno libre, flexible y accesible en este campo.

1.6 Contribuciones

scikit-forecasts es una librería en Python para el pronóstico de series de tiempo usando técnicas no lineales. La librería implementa varios de los métodos más clásicos y ampliamente utilizados para el pronóstico de series de tiempo, al igual que modelos de pronóstico más recientes y que han estado enfocados en solucionar otros problemas de regresión y clasificación. Esta librería está desarrollada como una extensión de scikit-learn, una librería en Python enfocada en modelos de aprendizaje de regresión y clasificación (aprendizaje supervisado y no supervisado), así que cuenta con muchas de sus funcionalidades y un flujo de trabajo similar que aplana la curva de aprendizaje de la librería desarrollada. Como extensión, scikit-forecasts está enfocada únicamente en el pronóstico (y no en modelado) de series de tiempo.

Además de contar con varios modelos de pronóstico, como modelos autorregresivos, redes neuronales artificiales, modelos neuro-difusos, modelos TAR y modelos SETAR, entre otros, es posible dentro de la librería:

- Cargar fácilmente múltiples series de tiempo a partir de un directorio.
- Transformar la(s) serie(s) de tiempo antes de ser ingresada(s) al modelo para evitar tendencias y desviaciones.
- Realizar un análisis exploratorio de los datos.
- Insertar fácilmente otros modelos dentro de la herramienta.
- Encontrar fácilmente los parámetros óptimos de un modelo.
- Correr diferentes modelos y transformaciones sobre múltiples series de tiempo y obtener métricas de error que permitan la elección del mejor modelo para cada serie.

scikit-forecasts permite, en un entorno libre y con muy pocas líneas de código, replicar pruebas, realizar comparaciones y desarrollar nuevos modelos para un análisis complejo y el pronóstico de una o múltiples series de tiempo.

1.7 Trabajos relacionados

En esta sección se describen las principales librerías que han sido utilizadas para el análisis y pronóstico de series de tiempo en los lenguajes Python, R, MATLAB, SAS, SPSS y STATA, y extensiones para Microsoft Excel.

En el proceso de investigación científica se puede notar claramente el uso de las diferentes herramientas dependiendo de la etapa en la que se esté aún cuando estas varíen un poco a través del tiempo. Esto evidencia el problema de desarticulación dentro de la investigación y su reproducción. Las etapas no son necesariamente consecutivas ni paralelas, actúan más como una descripción del tipo de trabajo que se lleva a cabo.

Inicialmente, se encuentra la etapa de trabajo individual exploratorio en el que se han usado diversos ambientes interactivos de trabajo. Entre los más comunes se encuentran Microsoft Excel, Matlab, Octave, Mathematica y algunos más especializados como SPSS, Statgraphics, STATA, SAS o R para estadística. Muchos de estos programas, aunque son interactivos, con gran capacidad para el análisis de datos y su visualización, cuentan con muchas limitaciones. La mayoría de ellos son comerciales y costosos, lo que los hace no solo difíciles de adquirir sino también de modificar en caso de que se desee hacerlo, impidiendo la replicabilidad y reproducibilidad de los resultados. Las principales herramientas que utilizan cada uno de estos programas para el análisis y pronóstico de series de tiempo se presentan en la Tabla 1-1: Herramientas de análisis o pronóstico de series de tiempo. y se describen a continuación.

En MATLAB, CAPTAIN Toolbox es una caja de herramientas para series de tiempo no estacionarias, identificación de sistemas, procesamiento de señales y pronóstico. Este Toolbox no está enfocado en las series de tiempo y se debe definir la variable temporal al momento de la estimación. Neural Network Toolbox, también de MATLAB, es una caja de herramientas que provee algoritmos, funciones y aplicaciones para crear, entrenar, visualizar y simular redes neuronales. Se pueden realizar clasificaciones, regresiones, agrupamientos, reducciones de dimensionalidad,

pronóstico de series de tiempo y modelado y control de sistemas dinámicos. Tampoco está enfocado en las series de tiempo y se deben presentar las variables de entrada como un problema de regresión.

En R, el paquete Forecast presenta un conjunto de métodos y herramientas para la visualización y el análisis de series de tiempo univariadas y su pronóstico. Es uno de los paquetes más completos dedicado específicamente a las series de tiempo. El paquete Neuralnet contiene métodos para el entrenamiento de redes neuronales usando propagación hacia atrás.

El sistema Time Series Forecasting de SAS desarrolla un ajuste del modelo y un pronóstico de manera automática. Selecciona el modelo que mejor se ajusta a la serie de tiempo. También identifica comportamientos y realiza diagnósticos. Es uno de los sistemas más completos para el análisis de series de tiempo con un tipo de licencia comercial.

STATA cuenta con el paquete Forecasting, este es una *suite* de comandos para obtener pronósticos a través de la solución de modelos o conjuntos de ecuaciones de una o más variables.

El módulo Forecasting de SPSS es un sistema para predecir tendencias y desarrollar pronósticos, muy similar a la *suite* de STATA.
En Python, existen varias librerías que contribuyen al pronóstico de las series de tiempo. Statsmodels es una librería que permite estimar modelos estadísticos y desarrollar pruebas. Tiene una gran cantidad de métodos para el análisis de series de tiempo. La librería McFly permite realizar una clasificación de series de tiempo utilizando métodos de *deep learning*. Por último, la librería TSFEL permite realizar un análisis exploratorio de las series de tiempo sin mucho esfuerzo en programación.

Por último, Microsoft Excel cuenta con la función FORECAST que permite realizar un pronóstico de una serie de tiempo definida en los libros. El entorno de Palisade, que funciona como una extensión de Microsoft Excel, permite implementar además modelos de redes neuronales para el pronóstico sobre datos incompletos y diferentes tipos de datos (categóricos y numéricos).

Tabla 1-1: Herramientas de análisis o pronóstico de series de tiempo.

Herramienta	Software	Tipo de licencia	Opciones de preprocesamiento / Modelos
CAPTAIN Toolbox	MATLAB	Comercial	No tiene opciones de preprocesamiento. Permite estimar los parámetros utilizando varios modelos de

[10]			optimización y realiza un análisis exhaustivo de errores.
Neural Network Toolbox [11]	MATLAB	Comercial	No tiene opciones de preprocesamiento. Permite obtener un pronóstico mediante un modelo de redes neuronales, sin embargo, se deben definir las variables de entrada.
Forecast Package [12]	R	Libre	Implementa algunas opciones de preprocesamiento como la transformación Box-Cox y diferenciación estacional. Cuenta con los modelos de pronóstico más ampliamente utilizados como regresión lineal, AR, ARIMA, Holt Winters y redes neuronales.
Neuralnet Package [13]	R	Libre	No tiene opciones de preprocesamiento. Se centra en el entrenamiento de redes neuronales artificiales.
Time Series Forecasting System [14]	SAS	Comercial	El sistema permite definir una tendencia de la serie como preprocesamiento, así como una descomposición y una estacionalidad. Cuenta además con un análisis exploratorio de la serie. SAS implementa los modelos más comunes de pronóstico: modelos ARIMA y de suavizado exponencial.
Forecasting [15]	STATA	Comercial	El sistema no realiza un preprocesamiento de la serie. Tiene integrados los modelos más comunes para pronóstico de series de tiempo: regresiones lineales, modelos ARIMA, VAR y ARCH.
Forecasting Module [16]	SPSS	Comercial	El módulo permite realizar una descomposición estacional en la parte de preprocesamiento. Los modelos que contiene son modelos ARIMA y de suavizado exponencial.
Statsmodels Package [17]	Python	Libre	La librería permite realizar una descomposición de la serie, remover tendencias y además tiene integrados varias pruebas para el análisis. Cuenta con modelos básicos de regresión lineal, ARIMA, SARIMAX, suavizado exponencial y modelos multivariados.
McFly Package [18]	Python	Libre	No cuenta con un preprocesamiento de la serie. Utiliza métodos de aprendizaje profundo dedicados a las series de tiempo, pero con la necesidad de definir un modelo de regresión.
TSFEL Package [19]	Python	Libre	Dedicada únicamente al preprocesamiento de la serie de tiempo, permite realizar transformaciones y pruebas.
Forecast [20]	Microsoft Excel	Comercial	Función individual que utiliza el método de suavizado exponencial para encontrar el pronóstico. El preprocesamiento debe realizarse mediante otras funciones.

NeuralTools [21]	Microsoft Excel	Comercial	La extensión no realiza un preprocesamiento. Implementa el método de redes neuronales artificiales para encontrar el pronóstico.

En muchos de estos paquetes se debe hacer manualmente el preprocesamiento de la serie, tal como transformaciones no lineales y remoción de tendencias y ciclos; por consiguiente, es responsabilidad del modelador llevar los pronósticos a las mismas unidades de la serie de datos original. Por otra parte, solo unas pocas herramientas (por ejemplo, el TSF de SAS) explícitamente presentan facilidades para la comparación y combinación de modelos. Otra falencia es que muchas de las herramientas están diseñadas para modelar una única serie de tiempo; esto quiere decir que: no es posible aplicar fácilmente una batería de modelos a un conjunto de series de tiempo, y que el modelador para realizar esta tarea debe escribir su propio código. Las herramientas comerciales usualmente no permiten la adición de nuevos modelos de pronóstico, por lo que el análisis está limitado a los modelos implementados, que suelen corresponder a los modelos más tradicionales y de uso más difundido como son los modelos ARIMA y los modelos de Suavizado Exponencial.

2. Descripción de los modelos matemáticos implementados

Los modelos clásicos de regresión son desarrollados para los llamados casos estáticos, es decir, que la variable dependiente es influenciada por variables independientes en el mismo momento del tiempo. Para el caso de las series de tiempo, se requiere que la variable dependiente sea influenciada por valores pasados de las variables independientes e incluso, por sus propios valores pasados. Si el presente puede ser modelado en términos de valores pasados, podemos decir que es posible realizar un pronóstico [22].

La librería scikit-forecasts cuenta con varios modelos integrados. A continuación se describe cada uno de ellos.

2.1 Modelos desde la aproximación estadística

2.1.1 Modelos lineales

Modelo autorregresivo

Los modelos autorregresivos están basados en la idea de que el valor presente en la serie de tiempo puede ser explicado como una función de *p* valores pasados de la misma serie, dónde *p* determina el número de pasos hacia atrás necesarios para pronosticar el valor presente.

Un modelo autorregresivo de orden *p*, abreviado AR(p), se expresa de la forma:

$$x_t = \varphi_1 x_{t-1} + \varphi_2 x_{t-2} + \cdots + \varphi_p x_{t-p} + w_t, \qquad (4)$$

dónde x_t es estacionaria, $\varphi_1, \varphi_2, \ldots, \varphi_p$ son constantes $(\varphi_p \neq 0)$. Se asume que w_t es un ruido blanco que sigue una distribución normal con media 0 y varianza σ_w^2. Además, la media de x_t es 0. En caso de que no sea 0, se reemplaza x_t por $x_t - \mu$, así:

$$x_t - \mu = \varphi_1(x_{t-1} - \mu) + \varphi_2(x_{t-2} - \mu) + \cdots + \varphi_p\big(x_{t-p} - \mu\big) + w_t, \qquad (5)$$

ó

$$x_t = \alpha + \varphi_1 x_{t-1} + \varphi_2 x_{t-2} + \cdots + \varphi_p x_{t-p} + w_t, \quad (6)$$

dónde

$$\alpha = \mu\left(1 - \varphi_1 - \cdots - \varphi_p\right). \quad (7)$$

Esta ecuación es similar a un modelo de regresión dónde los parámetros a estimar son $\varphi_1, \varphi_2, \ldots, \varphi_p$.

Dado que los regresores $x_{t-1}, \ldots, x_{t-p}$ son componentes aleatorios, una manera útil de expresar un modelo AR(p) es

$$\left(1 - \varphi_1 B - \varphi_2 B^2 - \cdots - \varphi_p B^p\right) x_t = w_t, \quad (8)$$

o más concisamente

$$\varphi(B) x_t = w_t. \quad (9)$$

$\varphi(B)$ es llamado el operador autorregresivo.

Modelo de medias móviles

Como una alternativa a los modelos autorregresivos en donde el x_t es una combinación lineal, el modelo de medias móviles de orden q, abreviado MA(q), asume que el ruido blanco w_t está combinado linealmente con los datos observados. El modelo se define como:

$$x_t = w_t + \theta_1 w_{t-1} + \theta_2 w_{t-2} + \cdots + \theta_q w_{t-q}, \quad (10)$$

dónde hay q retrasos en la media móvil y $\theta_1, \theta_2, \ldots, \theta_q \left(\theta_q \neq 0\right)$ son los parámetros. w_t es un ruido blanco que sigue una distribución normal.

Este modelo también puede ser escrito de la forma

$$x_t = \theta(B) w_t, \quad (11)$$

dónde $\theta(B)$ es el operador de medias móviles.

Modelo autorregresivo de medias móviles

Un modelo autorregresivo de medias móviles (ARMA) es un modelo mixto para series de tiempo estacionarias.

Una serie de tiempo se puede expresar como un modelo ARMA(p, q) si es estacionaria y sigue la siguiente ecuación:

$$x_t = \varphi_1 x_{t-1} + \cdots + \varphi_p x_{t-p} + w_t + \theta_1 w_{t-1} + \cdots + \theta_q w_{t-q}, \qquad (12)$$

dónde $\varphi_p \neq 0$, $\theta_q \neq 0$ y $\sigma_w^2 > 0$. Los parámetros p y q son llamados orden autorregresivo y de medias móviles respectivamente. Si x_t tiene una media μ diferente de 0, se define

$$\alpha = \mu(1 - \varphi_1 - \cdots - \varphi_p) \qquad (13)$$

y se escribe el modelo cómo

$$x_t = \alpha + \varphi_1 x_{t-1} + \cdots + \varphi_p x_{t-p} + w_t + \theta_1 w_{t-1} + \cdots + \theta_q w_{t-q}. \qquad (14)$$

Como es posible notar, cuando q = 0, el modelo es llamado un modelo autorregresivo de orden p, AR(p). Cuando p = 0, el modelo es un modelo de medias móviles de orden q, MA(q).

Más concisamente, este modelo puede escribirse de la forma

$$\varphi(B)x_t = \theta(B)w_t. \qquad (15)$$

Modelo autorregresivo integrado de medias móviles

En muchas ocasiones, las series de tiempo tienen dos componentes: uno de tendencia no estacionaria y uno de tendencia estacionaria de media cero. Un modelo integrado autorregresivo de medias móviles o ARIMA es un modelo ARMA que incluye una diferenciación.
Un proceso x_t es un modelo ARIMA(p,d,q) si

$$\nabla^d x_t = (1 - B)^d x_t \qquad (16)$$

es un modelo ARMA(p,q). En general el modelo puede ser escrito de la forma

$$\varphi(B)(1-B)^d x_t = \theta(B) w_t. \quad (17)$$

Si $E\left(\nabla^d x_t\right) = \mu$, el modelo se escribe como

$$\varphi(B)(1-B)^d x_t = \alpha + \theta(B) w_t, \quad (18)$$

dónde

$$\alpha = \mu\left(1 - \varphi_1 - \cdots - \varphi_p\right). \quad (19)$$

Modelo autorregresivo integrado de medias móviles estacional

En algunas ocasiones, ocurre una dependencia de un tiempo en el pasado con un retraso *s*. Por ejemplo, en datos económicos mensuales hay un fuerte componente anual debido a la actividad del año calendario. Esto se expresa como *s* = 12. En datos trimestrales, *s* = 4. En datos relacionados con los fenómenos naturales como la temperatura, este componente también es muy claro debido a las estaciones.

Un modelo autorregresivo de medias móviles con un componente estacional, ARMA(P,Q)s toma la forma

$$\Phi_P(B^S) x_t = \Theta_Q(B^S) w_t, \quad (20)$$

Con los operadores

$$\Phi_P(B^S) = 1 - \Phi_1 B^s - \Phi_2 B^{2s} - \cdots - \Phi_P B^{Ps} \quad (21)$$

y

$$\Theta_Q(B^S) = 1 + \Theta_1 B^s + \Theta_2 B^{2s} + \cdots + \Theta_Q B^{Qs} \quad (22)$$

Un modelo autorregresivo integrado de medias móviles estacional, SARIMA (definido por Box & Jenkins en 1970) está dado por

$$\Phi_P(B^S)\varphi(B)\nabla_S^D \nabla^d x_t = \alpha + \Theta_Q(B^S)\theta(B) w_t, \quad (23)$$

dónde w_t es un ruido blanco normalmente distribuido.

El modelo general se denota como ARIMA(p,d,q)(P,D,Q)s.

2.1.2 Modelos no lineales

Modelo autorregresivo de transición

Un modelo autorregresivo de transición ó TAR, es un modelo autorregresivo dónde los parámetros cambian dependiendo del valor de una variable exógena de umbral [23]. En este modelo, el umbral se convierte en otro de los parámetros a estimar.

Un modelo TAR está representado por

$$y_t = x_t\varphi^{(j)} + \sigma^{(j)}\epsilon_t \;\; si \;\; r_{j-1} < z_t \leq r_j \tag{24}$$

Dónde

$$x_t = \left(1, y_{t-1}, y_{t-2}, \dots, y_{t-p}\right), j = 1,2, \dots, k, \text{ y } -\infty = r_0 < r_1 < \cdots < r_k = \infty. \tag{25}$$

Y z_t es la variable exógena de umbral.

Los $k-1$ umbrales no triviales dividen el dominio de la variable z_t en k regímenes donde la serie de tiempo y_t sigue un modelo AR(p) diferente.

Modelo autorregresivo de transición suavizado

En el modelo autorregresivo de transición suavizado STAR, la discontinuidad en el umbral del modelo TAR es reemplazada por una función suavizada de transición. Esta función

$$0 < G(z_t) < 1 \tag{26}$$

depende de la variable de transición z_t y el modelo se representa

$$y_t = x_t\varphi^{(1)}\left(1 - G(z_t)\right) + x_t\varphi^{(2)}G(z_t) + \epsilon_t \tag{27}$$

Dos de las funciones de transición más comunes son la función logística y la función exponencial. La función logística puede ser representada como

$$G(z_t; \gamma, c) = \frac{1}{1 + e^{-\gamma(z_t - c)}}, \gamma > 0 \tag{28}$$

dónde el parámetro c representa un umbral y γ la velocidad de la transición.

Usando la función exponencial, la función de transición puede ser representada como

$$G(z_t; \gamma, c) = 1 - e^{-\gamma(z_t - c)^2}, \gamma > 0 \tag{29}$$

Modelo de suavizado exponencial

Los modelos de suavizado exponencial fueron propuestos por Brown (1959), Holt (1957) y Winters (1960). El pronóstico se encuentra usando un promedio ponderado de observaciones pasadas, donde el peso decae exponencialmente mientras más vieja sea la observación [24].
El modelo simple de suavizado exponencial se utiliza para series de tiempo que no tienen un patrón de tendencia ni estacionalidad y está dado de la forma

$$\hat{x}_{T|T-1} = \alpha x_{T-1} + \alpha(1-\alpha)x_{T-2} + \alpha(1-\alpha)^2 x_{T-3} + \cdots, \qquad (30)$$

dónde $0 \leq \alpha \leq 1$ es el parámetro de suavizado. El pronóstico en el tiempo t es una media ponderada de todas las observaciones controladas por el parámetro α.

Una representación alternativa es presentar un sistema con varios componentes. En el suavizado exponencial simple, los componentes serían una ecuación de pronóstico y una de suavizado, como se muestran a continuación:

Ecuación de pronóstico	$\hat{x}_{t+h\|t} = l_t$	(31)
Ecuación de suavizado	$l_t = \alpha x_t + (1-\alpha) l_{t-1}$	(32)

El modelo de Holt permite realizar el pronóstico con una tendencia siguiendo el siguiente sistema (incluyendo el componente de tendencia).

Ecuación de pronóstico	$\hat{x}_{t+h\|t} = l_t + hb_t$	(33)
Ecuación del nivel	$l_t = \alpha x_t + (1-\alpha)(l_{t-1} + b_{t-1})$	(34)
Ecuación de tendencia	$b_t = \beta^*(l_t - l_{t-1}) + (1-\beta^*)b_{t-1},$	(35)

Dónde α es el parámetro de suavizado de nivel y β^* el parámetro de suavizado de la tendencia. Ambos parámetros entre 0 y 1.

Finalmente, el método de Holt Winters logra capturar una estacionalidad añadiendo este componente. Este método tiene 2 variaciones. El método aditivo que se usa preferiblemente cuando las variaciones estacionales son constantes en la serie de tiempo. El método multiplicativo, usado

preferiblemente cuando las variaciones estacionales cambian proporcionalmente dependiendo del nivel en la serie.

Los componentes del método aditivo están descritos como

$$\hat{x}_{t+h|t} = l_t + hb_t + s_{t+h-m(k+1)} \quad (36)$$

$$l_t = \alpha(x_t - s_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1}) \quad (37)$$

$$b_t = \beta^*(l_t - l_{t-1}) + (1-\beta^*)b_{t-1} \quad (38)$$

$$s_t = \gamma(x_t - l_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m}, \quad (39)$$

dónde k es la parte entera de $\frac{(h-1)}{m}$ y γ sigue la restricción normal entre 0 y 1.

Los componentes del método multiplicativo están descritos como

$$\hat{x}_{t+h|t} = (l_t + hb_t)s_{t+h-m(k+1)} \quad (40)$$

$$l_t = \alpha\frac{x_t}{s_{t-m}} + (1-\alpha)(l_{t-1} + b_{t-1}) \quad (41)$$

$$b_t = \beta^*(l_t - l_{t-1}) + (1-\beta^*)b_{t-1} \quad (42)$$

$$s_t = \gamma\frac{x_t}{(l_{t-1} - b_{t-1})} + (1-\gamma)s_{t-m} \quad (43)$$

2.2 Modelos de aprendizaje de máquinas

Modelo de redes neuronales

Un modelo MLP (Multi Layer Perceptron) es un algoritmo de aprendizaje supervisado que aprende una función

$$f(\cdot) : R^m \rightarrow R^o \quad (44)$$

mediante el entrenamiento con un conjunto de datos, dónde m es el número de dimensiones de entrada y o el número de dimensiones de salida. Dado un conjunto de características

$$x = x_1, x_2, \ldots, x_m \quad (45)$$

y un objetivo y, el algoritmo puede aprender una función no lineal para encontrar un valor. Entre la capa de entrada y la de salida, puede haber más capas no lineales, llamadas capas ocultas.

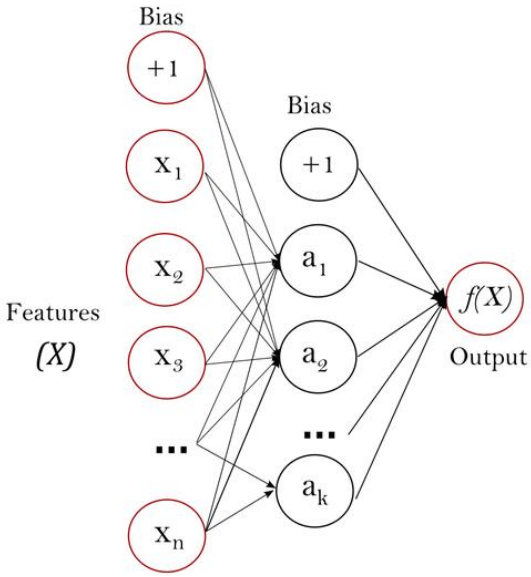
Para una serie de tiempo, el conjunto de características X está representado por los p valores pasados de la serie.

Cada neurona en la capa oculta transforma el valor de la capa anterior en una sumatoria linear ponderada $w_1x_1 + w_2x_2 + \cdots + w_mx_m$, seguida por una función de activación no lineal $g(\cdot) : R \to R$.

La capa de salida recibe los valores ponderados de la última capa oculta y los transforma en los valores de salida (o pronósticos).

La Figura 2-1: Modelo MLP con una capa oculta tomado de scikit-learn. muestra un modelo MLP con una capa oculta.

Figura 2-1: Modelo MLP con una capa oculta tomado de scikit-learn.



Modelo híbrido neuro difuso

El modelo AMNFIS presentado por Velásquez (2015) es un sistema neuro difuso adaptativo multidimensional para el pronóstico de series de tiempo no lineales. Consiste en un conjunto de L reglas difusas donde la i-ésima regla es expresada como

$$\text{Si } x_t \in S_i, \text{ entonces } p_i = f(\theta_i; x_t) \qquad (46)$$

dónde $x_t = [y_{t-1}, y_{t-2}, \ldots]'$ es un vector de valores previos de y_t, S_i es un conjunto difuso multidimensional, pi es el valor pronosticado para y_t usando la regla i y f(.) es una función lineal que toma la forma

$$p_i = f(\theta_i; x_t) = \theta_{io} + \sum_{p=1}^{P} \theta_{ip} y_{t-p} \tag{47}$$

Esta forma es un modelo autorregresivo de orden p con intercepto diferente de cero.

3. scikit-forecasts: Una librería en Python para el pronóstico de series de tiempo no lineales

scikit-forecasts es una librería en Python dedicada al análisis y pronóstico de series de tiempo no lineales. Esta librería cuenta, además de los modelos de análisis, con una serie de funcionalidades que permiten el preprocesamiento y descripción de los datos. También permite externamente transformar la serie de tiempo para que pueda ser ingresada al modelo sin ningún tipo de tendencia o desviación.

La librería permite desarrollar un flujo de trabajo con la variación de sus parámetros para una o varias series de tiempo (*pipeline*) y, además, elegir el mejor modelo entre una variedad definida (ya sea entre diferentes modelos o entre cambios en los parámetros del mismo modelo) a partir de su función de búsqueda. Es posible dividir el flujo de trabajo de las técnicas de aprendizaje en 4 pasos [10]:

- **Recolección de la información**: cómo obtener los datos necesarios para implementar modelo y su limpieza.
- **Representación de los datos:** características de los datos y preparación para ser ajustados al modelo.
- **Aprendizaje:** elección del método o algoritmo que será entrenado con los datos recolectados.
- **Evaluación del modelo:** resultados de la predicción del modelo comparados con datos reales para obtener métricas de rendimiento y concluir si es satisfactorio.

Existen cuatro grandes librerías en Python para implementar este flujo dedicadas al *machine* y *deep learning*.

- **TensorFlow:** Es una plataforma para desarrollar y entrenar modelos de aprendizaje automático. TensorFlow permite una compilación sencilla de modelos, implementación en diferentes ambientes y publicación rápida de modelos debido a su estructura (simple y flexible) [25].

- **Keras:** Es una librería de *deep learning* diseñada específicamente para implementar redes neuronales y está soportada en TensorFlow. El foco de Keras es la implementación rápida [26].

- **PyTorch:** Al igual que Keras, **PyTorch** está construida en TensorFlow (y basada en Torch). Este framework permite un crecimiento rápido de la investigación debido a su componente de acelaración [27].

- **scikit-learn:** Es una librería simple y eficiente de aprendizaje automático, enfocada en herramientas para la predicción y el análisis de datos [28]. A diferencia de las demás librerías, scikit-learn cuenta con diferentes funcionalidades específicas para el análisis de series de tiempo. Una de ellas es la capacidad de diferenciar los modelos entre clasificación y regresión. Otra, la implementación de técnicas de verificación de modelos como *cross-validation* enfocadas específicamente en series de tiempo (las cuáles tienen otro grado de complejidad en la validación del modelo).

Aún con esto, scikit-learn no ha desarrollado un componente predictivo completamente enfocado en las series de tiempo. Para la implementación de los modelos es necesario construir matrices con ciertas características y rehacer los modelos en torno a la estructura de la librería.

La extensión **scikit-forecasts** es una herramienta para insertar modelos exclusivos del análisis de series de tiempo y aprovechar las ventajas con las que cuenta scikit-learn.

3.1 Arquitectura del software

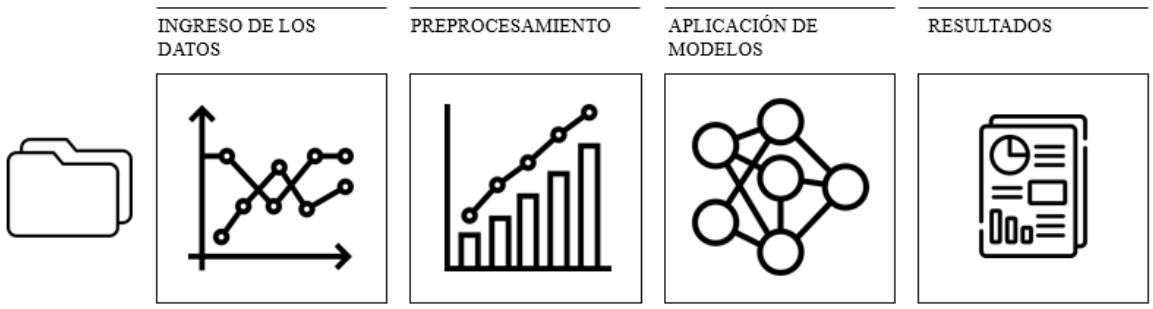
La librería scikit-forecasts está desarrollada en Python 3, y sigue los mismos principios de desarrollo usados en la arquitectura de scikit-learn. Esto implica que los modelos desarrollados en scikit-forecasts están representados como objetos cuya parametrización se realiza en el constructor de la clase; al igual que los modelos implementados en scikit-learn, los modelos implementados en scikit-

forecasts tienen la función fit para su entrenamiento y la función predict para su pronóstico. Por otra parte, los transformadores en scikit-forecasts usan las funciones fit y transform, de forma similar a sus análogos en scikit-learn. Esta característica reduce la curva de aprendizaje, particularmente para usuarios con conocimientos de scikit-learn, y permite que se puedan utilizar funciones ya implementadas en scikit-learn. Muchas de sus características y funcionalidades dependen de Numpy, SciPy, Pandas, Matplotlib, scikit-learn, entre otras. Los resultados obtenidos usando los métodos de la librería, están estandarizados para que sean compatibles con las demás librerías de análisis en Python.

La librería puede ser usada interactivamente en un libro de Jupyter, lo que facilita el desarrollo de modelos y la visualización de los resultados. También puede ser usada en cualquier entorno de programación que soporte Python 3.

La Figura 3-1 presenta el flujo de trabajo de la librería. Inicialmente se obtienen los datos. Estos deben ser series de tiempo de la clase *pandas.Series*. Estos datos son analizados y preprocesados, dónde pueden ser limpiados, transformados o divididos. En una tercera etapa, uno o varios modelos de pronóstico son estimados para la serie de tiempo y finalmente, se obtienen los resultados que pueden ser comparados con otros siguiendo este mismo flujo de trabajo.

Figura 3-1: Flujo de trabajo de scikit-forecasts.



En la Tabla 3-1 se presentan los metadatos del software.

Tabla 3-1: Metadatos del código.

Actual version del código	0.1.0
Link permanente al repositorio de esta versión	https://github.com/jdvelasq/scikit-forecasts

Licencia legal	MIT License
Versión del sistema usado	Git
Lenguaje de programación y herramientas	Python 3.7
Requerimientos y dependencias	NumPy, SciPy, Pandas, Matplotlib, Scikit-learn, Math, Random, Doctest, Statsmodels, Tabulate, Paver
Link al manual o documentación	https://jdvelasq.github.io/scikit-forecasts/
Correo electrónico de soporte y preguntas	maaarangogo@unal.edu.co

3.2 Funcionalidades

3.2.1 Ingreso de datos

Los modelos dentro de la librería reciben una serie de tiempo. Esta es un vector de tipo *pandas.Series* que tiene dos componentes, un índice y un valor (*index* y *values*). El índice corresponde a una referencia en momentos del tiempo. Aunque no es estrictamente necesario, el índice comúnmente es del tipo *pandas.DateTime*, el cual cuenta con características importantes para el pronóstico como lo es la frecuencia.

3.2.2 Datasets disponibles en la herramienta

La librería cuenta con una serie de datasets, entre los que se encuentran los presentados en la Tabla 3-2. Para el uso de estos datasets, es posible cargar la serie con las funciones especificadas.

Tabla 3-2: Descripción de los datasets de la librería y fuentes de los datos.

Dataset	Descripción del dataset	Fuente
The airline passenger data set	Este data set mide el número total de pasajeros en un vuelo internacional por mes desde Enero de 1949 hasta Diciembre de 1960. La gráfica de los datos muestra no linealidad y exhibe un comportamiento estacional multiplicativo.	[29]
The Canadian lynx data set	Este data set consiste en el número de linces Canadiences atrapados en el río Mackenzie en el norte de Canadá por año durante el período de 1821 a 1934 (114 observaciones).	[30]
The sunspot data set	Este data set es el récord de la actividad anual de manchas solares visibles de cara al sol y el número de grupos en las que se agrupan	[31]

	desde 1700. Este conjunto de datos es no lineal, no gaussiano y es comúnmente usado para medir la efectividad de modelos estadísticos no lineales.	
The internet data set	Este dataset contiene el número de usuarios loggeados en un servidor de internet cada minuto durante 100 minutos. Es un modelo que indica no estacionariedad.	[32]
The printing and writing paper data set	Este data set contiene las ventas de la industria del papel (en miles de francos) entre Enero de 1963 y Diciembre de 1972 (120 observaciones). Este conjunto de datos muestra estacionalidad y una creciente tendencia.	[33]
The pollution equipment data set	Este data set contiene los envíos mensuales de equipos de polución desde Enero de 1986 hasta Octubre de 1996 (en miles de francos). Consiste en 130 observaciones que muestran una "no estacionariedad en varianza".	[33]
Electricity contract prices data set	Este data set corresponde al precio promedio mensual de los contratos de electricidad en Colombia desde Mayo de 1996 hasta Junio de 2008 (146 observaciones) en pesos por kWh. Se caracteriza por tener una tendencia lineal y un ciclo estacional.	[34]
Electricity demand data set	Este data set con 177 observaciones corresponde a la demanda mensual de electricidad en el mercado Colombiano en miles de GWh desde Agosto de 1995 hasta Abril de 2010.	[34]
Champagne data set	Este data set corresponde al número de ventas mensuales de champaña desde Enero de 1964 hasta Septiembre de 1972 (105 observaciones).	[35]

3.2.3 Carga automática de archivos

En el pronóstico de series de tiempo, es común que se deban procesar muchas series simultáneamente. scikit-forecacts contiene la función `file_reader`, que permite cargar simultáneamente todos los archivos de datos ubicados en un directorio especificado por el usuario como series de tiempo, para que puedan ser procesadas por la librería. Los datos son convertidos a objetos del tipo *pandas.Series*. En el siguiente fragmento de código, se da un ejemplo de una carga automática de archivos desde un directorio específico.

```
001 >>> from skfore.file_reader import file_reader
002 >>> times = file_reader.read_dir('datasets/', '*')
003
```

```
[         Perrin
Month
1964-01    2815
1964-02    2672
1964-03    2755
1964-04    2721
1964-05    2946
...         ...
1972-05    4618
1972-06    5312
1972-07    4298
1972-08    1413
1972-09    5877

[105 rows x 1 columns],          Perrin
Month
1964-01    2815
1964-02    2672
1964-03    2755
```

3.2.4 Funcionalidades para el análisis exploratorio de los datos

La librería permite realizar el análisis exploratorio de los datos (o el análisis preliminar de la serie de tiempo) basándose en las librerías Statsmodels, SciPy y Pandas. Esta exploración proporciona rápidamente información sobre la estructura del problema a modelar. El análisis exploratorio de los datos se encuentra codificado en la clase series_viewer, la cual recibe una serie de tiempo a analizar y devuelve información sobre la serie mediante los métodos que se describen en la Tabla 3-3.

Tabla 3-3: Métodos para el análisis exploratorio de los datos.

Método	Descripción
Resumen estadístico	Proporciona una idea general de los datos con los que se está trabajando (como número de observaciones, mínimo y máximo, entre otros).
Gráfico de línea	Muestra generalizada de la serie de tiempo. Aquí se pueden notar estructuras o tendencias de la serie.
Histograma de frecuencias	Proporciona una mayor comprensión de la estructura de los datos. Es la representación gráfica de la frecuencia de los valores de la serie en forma de barras.
Gráfico de densidad	Proporciona una mayor comprensión de la estructura de los datos. Se representa como una línea que sigue el histograma de frecuencias de los datos.
Gráfico ACF	Gráfica de la función de autocorrelación. Proporciona una medida de correlación entre las observaciones de la serie separadas por algunas unidades de tiempo.
Gráfico PACF	Gráfica de la función de autocorrelación parcial. Proporciona una medida de correlación entre las observaciones de la serie separadas por algunas unidades de tiempo después de ajustarse.
Gráfico cuantil-cuantil	Proporciona un análisis gráfico de normalidad en una muestra. Analiza cuánto difiere la distribución de los datos con respecto a una distribución normal con la misma media y desviación estándar [36].
Test de normalidad Jarque Bera	El test de Jarque-Bera es una prueba de bondad de ajuste para comprobar si una muestra de datos tiene la asimetría y la curtosis de una distribución normal [37].

3.2.5 Transformación de una serie de tiempo

En este apartado se explica cómo la serie puede ser preliminarmente analizada y con base en esta información, transformada para remover cualquier tipo de nivel, tendencia y/o estacionalidad.

scikit-forecasts implementa las principales transformaciones que se aplican a las series de tiempo para su pronóstico. La primera transformación permite la aplicación de funciones matemáticas que modifican el nivel de la serie (log, sqrt, exp, etc); seguidamente, se puede realizar la remoción de la tendencia a largo plazo de la serie; y finalmente, se puede remover la componente estacional. La librería implementa en una clase todos los tipos de transformaciones para que sean aplicados secuencialmente. De igual forma, también permite realizar la transformación inversa con el fin de llevar los pronósticos a las mismas unidades de la serie original. Esta clase está basada en NumPy y SciPy y recibe como argumentos la transformación, tendencia y/o estacionalidad deseada. En la Tabla 3-4 se presentan las transformaciones posibles que pueden ser aplicadas.

Tabla 3-4: Tipos de transformación de las series de tiempo.

Tipo de transformación	Transformación	Descripción
Transformación	*log*	Aplica la función de logaritmo natural en base *e* a la serie de tiempo.
	log10	Aplica la función logaritmo natural en base 10 a la serie de tiempo.
	sqrt	Aplica la función raíz cuadrada a la serie de tiempo.
	cbrt	Aplica la función raíz cúbica a la serie de tiempo.
	boxcox	Aplica la transformación Box-Cox a la serie de tiempo y almacena el parámetro λ.
Tendencia	*linear*	Estima una función lineal que se ajuste a la serie de tiempo y la remueve de esta.
	cuadratic	Estima una función cuadrática que se ajuste a la serie de tiempo y la remueve de esta.
	cubic	Estima una función cúbica que se ajuste a la serie de tiempo y la remueve de esta.
	diff1	Realiza una diferenciación de grado 1 en el tiempo.
	diff2	Realiza una diferenciación de grado 2 en el tiempo.
Estacionalidad	*poly2*	Estima un polinomio de grado 2 que se ajuste a la serie de tiempo y lo remueve de esta.
	diff	Realiza una diferenciación de la frecuencia de la serie de tiempo.

scikit-forecasts implementa la clase `Tranformer` para el preprocesamiento de la serie de tiempo, el cual permite aplicar las transformaciones presentadas en la Tabla 3-4 a una serie de tiempo. La implementación sigue la misma convención utilizada por scikit-learn en los transformadores. De esta forma, la especificación de la transformación se realiza en el constructor de la clase; luego debe entrenarse el transformador mediante la función `fit`, y finalmente aplicarse mediante la función `transform`. En el siguiente fragmento de código se carga la serie Airline Passengers y se aplican las siguientes transformaciones: se aplica la función log para hacer aproximadamente constante el ciclo estacional; seguidamente se remueve la tendencia mediante diferenciación simple; y finalmente se remueve la componente cíclica mediante la diferenciación estacional de período 12. En la línea 006 se aplica la transformación inversa para llevar la serie transformada a las unidades de la serie original.

```
001 >>> from skfore.preprocessing.Transformer import transformer
002 >>> from skfore.datasets import load_passenger()
003 >>> ts = load_passenger()
004 >>> print(ts)
005
     Month
     1949-01    112
     1949-02    118
     1949-03    132
     1949-04    129
     1949-05    121
                ...
     1960-08    606
     1960-09    508
     1960-10    461
     1960-11    390
     1960-12    432
     Name: passengers, Length: 144, dtype: int64
006 >>> my_transform = transformer(
007 ...     trans='log',
008 ...     trend='diff1',
009 ...     seasonal='diff'
010 ... )
011 >>> t_ts = my_transform.fit(ts=ts)
012 >>> print(t_ts)
013
    Month
    1949-01    0.000000
    1949-02    0.000000
    1949-03    0.000000
    1949-04    0.000000
    1949-05    0.000000
                  ...
    1960-08   -0.045934
    1960-09    0.012024
    1960-10    0.031830
    1960-11   -0.050082
    1960-12   -0.009964
    Length: 144, dtype: float64
```

```
014 >>> original_ts = my_transform.inverse_transform(ts=t_ts)
015 >>> print(original_ts)
016
    Month
    1949-01    112
    1949-02    118
    1949-03    132
    1949-04    129
    1949-05    121
               ...
    1960-08    606
    1960-09    508
    1960-10    461
    1960-11    390
    1960-12    432
    Name: passengers, Length: 144, dtype: int64
```

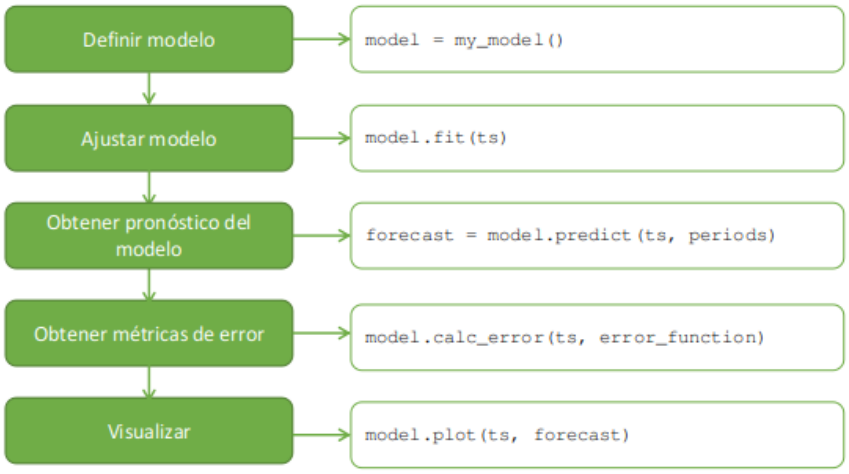
3.2.6 Modelos implementados para el pronóstico de series de tiempo

scikit-forecasts implementa los siguientes modelos para el pronóstico de series de tiempo:

- ***AR:*** Modelo autorregresivo.
- ***MA*:** Modelo de medias móviles.
- ***ARMA:*** Modelo autorregresivo de medias móviles.
- ***ARIMA:*** Modelo autorregresivo integrado de medias móviles.
- ***SARIMA:*** Modelo autorregresivo integrado de medias móviles estacional.
- ***TAR:*** Modelo autorregresivo de transición.
- ***STAR:*** Modelo autorregresivo de transición estacional.
- ***Holt-Winters:*** Modelo de suavizado exponencial.
- ***MLP:*** Modelo de redes neuronales.
- ***AMNFIS:*** Modelo híbrido neuro difuso.

El flujo de trabajo para obtener pronósticos de la serie de tiempo mediante un modelo se describe en la Figura 3-2. Inicialmente se define un modelo (el cuál es llamado `my_model` como ejemplo). Luego se ajusta el modelo mediante la función `fit`. El método `predict`, a continuación, permite obtener el pronóstico de la serie. Finalmente, los métodos `calc_error` y `plot`, permiten obtener las métricas de error y visualizaciones, respectivamente.

Figura 3-2: Flujo de trabajo para aplicación de modelos de pronóstico.



El método `fit` permite encontrar los parámetros óptimos del modelo. Recibe una serie de tiempo a ajustar, una función de error (o métrica de error) y los parámetros del optimizador; Retorna el modelo con sus parámetros óptimos estimados. El método `fit` puede usar diferentes tipos de optimizadores que están implementados en SciPy o scikit-learn.

El método `predict` permite el pronóstico de varios períodos hacia adelante a partir de un punto en el tiempo. Este método devuelve un *DataFrame* con las predicciones del modelo y otras características elegidas.

Inicialmente, el método `predict` recibe una serie de tiempo y un número de períodos a pronosticar, pero existen diferentes parámetros dentro de la función que pueden ser elegidos para obtener el pronóstico. El parámetro `blind` define si el pronóstico es o no ciego, es decir, si pronostica tantos períodos a futuro como fueron requeridos solo con usar la simulación del modelo o por el contrario, recibe una serie comparada que alimenta período a período y pronostica solo un paso a la vez.

En caso de que el parámetro `blind` sea definido como False, el usuario deberá proporcionar una serie de tiempo adicional, con tantas observaciones como períodos requeridos, de la cual se extraerán los datos reales para el pronóstico de un solo paso (parámetro `tsp`).

El usuario podrá definir el método de muestreo de su preferencia. Bajo esta estructura se encuentran los métodos *Bootstrap* y la simulación de un error normal con media y desviación conocidos (por la serie de tiempo dada). Para estas simulaciones, también es posible elegir el intervalo de confianza (entre 0 y 1) y el número de iteraciones de la simulación. En caso de que ninguno de estos parámetros sea definido (o alguno de ellos), el método por defecto es un muestreo *Bootstrap*, con un intervalo de confianza del 95% o 0.95 y 300 iteraciones.

El parámetro `random_state` permite definir un número de semilla determinado.

El resultado será el *DataFrame* compuesto por el índice correspondiente al período de tiempo, los intervalos de confianza, el pronóstico bajo el método requerido y en caso de que el pronóstico no sea ciego, los valores reales de la serie.

Tabla 3-5: DataFrame resultado de aplicar el método predict.

Columna	Descripción
ci_inf	Intervalo de confianza inferior del pronóstico.
ci_sup	Intervalo de confianza superior del pronóstico.
bootstrap	Valor del pronóstico generado por el método bootstrap.
normal	Valor del pronóstico generado por el muestreo de error normal.
series	Valor del pronóstico generado por el método simulate.
forecast	Valor del pronóstico generado por el muestreo de errores elegido (normal o bootstrap).
real	Valor de la serie de tiempo real que se puede compar (blind = False).

El método `filter_ts` del modelo recibe una serie de tiempo y retorna los residuales obtenidos al calcular los pronósticos un paso adelante. Adicionalmente, se implementó el método `forecast`, que computa el pronóstico un paso adelante dada una serie de tiempo. `filter_ts` retorna la diferencia entre el valor real y el pronóstico un paso adelante obtenido con `forecast`.

El método `simulate` recibe una serie de tiempo y retorna una serie simulada, dónde cada valor corresponde al pronóstico usando los valores previos de la serie más un error. Este método permite generar series artificiales que tienen las mismas propiedades de la serie real.

3.2.7 Uso de modelos de scikit-learn para pronóstico de series de tiempo

scikit-forecasts implementa la clase `Reg2TS` la cual permite utilizar un modelo de regresión de la librería scikit-learn para realizar pronósticos de una serie de tiempo. Esta clase recibe como parámetros de entrada el orden del modelo autorregresivo y el modelo con el cuál se quiere pronosticar. Permite además aplicar todas las funcionalidades de los modelos dentro de la librería.

Un ejemplo de esta clase se presenta en la Sección 4.4.

3.2.8 Uso de diferentes métricas de error

El modelo puede ser estimado utilizando diferentes métricas de error como el error cuadrático medio o el error medio absoluto (o cualquiera definido por el usuario).

El método `calc_error` recibe la serie de tiempo a ajustar y una función de error (o métrica). En caso de que esta última no sea definida, la métrica por defecto será el error cuadrático medio (*MSE - Mean Squared Error*).

El método retorna un valor correspondiente al cálculo del error de la métrica elegida.

3.2.9 Actualización de la configuración de los modelos

La librería permite actualizar los parámetros que definen la configuración del modelo, como por ejemplo, el orden de un modelo autorregresivo. Esta capacidad es necesaria por compatibilidad con los modelos de scikit-learn, particularmente para la búsqueda de hiper-parámetros óptimos tal como se hace con la función `GridSearchCV`.

3.2.10 Funcionalidades para el desarrollo de nuevos modelos

La librería permite la especificación de modelos de usuario; para ello, todos los nuevos modelos deben ser derivados de la clase `BaseModel`. En la especificación del nuevo modelo, el usuario solamente debe implementar los métodos `fit`, `forecast` y `predict`, ya que el método `filter` se encuentra especificado en el modelo base y depende de los tres métodos anteriores.

Para construir un modelo AR desde cero, inicialmente se define la clase del modelo derivada de `BaseModel`. Esta clase recibe los parámetros del modelo: el parámetro `p` que corresponde al orden y los parámetros `intercept` y `phi`, correspondientes a los parámetros propios del modelo y los cuáles se incluyen aquí en caso de que el usuario final quiera especificarlos.

```
001 >>> from skfore.models.BaseModel import BaseModel
002 >>> class AR(BaseModel):
003 >>>    """Autoregressive model
004 >>>    Parameter optimization method: scipy's minimization
005 >>>    Args:
006 >>>        p (int): order
007 >>>        intercept (boolean or double): False for set intercept to
008 ... 0 or double
009 >>>        phi (array): array of p-length for set parameters without
010 ... optimization
011 >>>    Returns:
012 >>>        AR model structure of order p
013 >>>    """
014 >>>    def __init__(self, p=None, intercept=None, phi=None):
015 >>>        if p == None:
016 >>>            self.p = 0
017 >>>        else:
018 >>>            self.p = p
019 >>>        if intercept == False:
020 >>>            self.phi0 = 0
021 >>>        else:
022 >>>            self.phi0 = intercept
023 >>>        self.phi = phi
```

Se crea inicialmente una función llamada `forecast` donde se especifica el modelo. Esta función recibe la serie de tiempo y devuelve el pronóstico de ella un paso adelante.

```
001 >>>     def forecast(self, ts):
002 >>>        """ Next step
003 >>>        Args:
004 >>>            ts (pandas.Series): Time series to find next value
005 >>>        Returns:
006 >>>            Value of next time stamp
007 >>>        """
008 >>>        if len(self.phi) != self.p:
009 >>>            self.phi = numpy.random.rand(self.p)
010 >>>        y = ts.values
011 >>>        lon = len(y)
012 >>>        if lon <= self.p:
013 >>>            y_last = y[0:lon]
014 >>>            result = self.phi0 + numpy.dot(y_last, self.phi[0:lon])
015 >>>        else:
016 >>>            y_last = y[lon-self.p:lon]
017 >>>            result = self.phi0 + numpy.dot(y_last, self.phi)
018 >>>        return result
```

A continuación, se definirá el método `fit`. Este método debe encargarse de optimizar los parámetros del modelo y devolver el mismo modelo. Para la optimización de parámetros, se implementará una función de minimización del error dependiente de SciPy. La función de error puede ser definida utilizando el método `calc_error` de la clase `BaseModel`.

```
001 >>>    def fit(self, ts, error_function = None):
002 >>>         """ Finds optimal parameters using a given optimization 003 ...
function
004 >>>        Args:
005 >>>            ts (pandas.Series): Time series to fit
006 >>>            error_function (function): Function to estimates error
007 >>>        Return:
008 >>>            self
009 >>>        """
010 >>>        def f(x):
011 >>>            self.vector2params(x)
012 >>>            return self.calc_error(ts, error_function)
013 >>>        x0 = self.params2vector()
014 >>>        optim_params = scipy.optimize.minimize(f, x0)
015 >>>        self.vector2params(vector = optim_params.x)
016 >>>        return self
```

Las funciones auxiliares `params2vector` y `vector2params`, devuelven los parámetros en forma de vector para aplicar la función de optimización y actualizan los parámetros dentro del modelo recibidos en forma de vector de la optimización respectivamente.

3.2.11Graficación de los pronósticos

El método `plot` genera una gráfica general de la serie. Tiene la posibilidad de ser generada mediante 2 funciones.

La primera de ellas recibe los mismos parámetros de la función `predict` (explicada en **¡Error! No se encuentra el origen de la referencia.**). La segunda, recibe como parámetros una serie de tiempo y un *DataFrame* que sale como resultado de aplicar la función `predict`.

Ambas funciones generan una gráfica basada en la librería Matplotlib. En esta gráfica se encuentran señalados la gráfica de línea real de la serie, la gráfica de línea de la serie ajustada, la gráfica de línea del pronóstico de los períodos requeridos y los intervalos de confianza generados para este pronóstico.

3.2.12 Otros métodos incluidos

La clase base implementa además otros métodos que facilitan el pronóstico. Entre ellos se encuentra el análisis (exploratorio) de residuales. Este conjunto de métodos (al igual que los descritos en 2.3), permite generar un análisis exploratorio de los residuales. Este análisis da una visión del comportamiento de los residuales del modelo.

El método `set_residuals` genera una clase de tipo `series_viewer`, la cual permite realizar el análisis exploratorio utilizando las funciones explicadas con anterioridad.

3.2.13 Búsqueda de los mejores parámetros de un modelo

La librería cuenta con una clase de búsqueda llamada `GridSearchCV` la cual permite mediante una función (igual que su homóloga en scikit-learn), encontrar los mejores parámetros entre una selección para un modelo dado. Los parámetros del modelo son optimizados a través de una búsqueda exhaustiva en un *grid* de parámetros. Un ejemplo de la aplicación de esta función puede verse en la sección 4.4.

3.2.14 Comparaciones entre modelos

Después de obtener una estimación de la serie aplicando uno de los modelos definidos en la librería o por el usuario, es posible replicar el proceso iterativamente para obtener resultados comparativos entre modelos fácilmente usando las funciones `Pipeline` y `MultiPipeline`. Los métodos que se describen en este apartado permiten llevar a cabo esta tarea.

scikit-forecasts cuenta con una clase `Pipeline`, la cual permite aplicar una serie de transformaciones y modelos a una serie de tiempo y retorna una métrica de error dada por cada combinación de transformación/modelo.

La clase recibe una lista de transformaciones, definidas con anterioridad y una lista de modelos, también definidos. Un ejemplo de esto puede verse en la Sección 4.3.

La clase `MultiPipeline` aplica la misma funcionalidad a múltiples series de tiempo que pueden ser cargadas de manera automática.

4. Tutoriales

En este capítulo se presentarán varios ejemplos significativos del pronóstico de una serie de tiempo. Cada uno de los tutoriales está enfocado en la presentación de una o más características particulares de la librería.

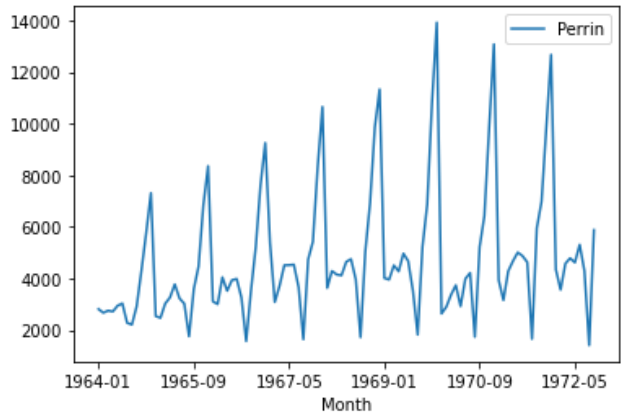
4.1 Tutorial 1: Pronóstico básico de una serie de tiempo usando una red neuronal

En este tutorial se presenta un ejemplo completo de pronóstico de una serie de tiempo, y corresponde al pronóstico de las ventas mensuales de champaña en una región de Francia. El conjunto de datos contiene el total de ventas mensuales en millones de francos de champaña desde enero de 1964 hasta septiembre de 1972, y contiene 105 observaciones.

4.1.1 Carga y graficación de la serie de tiempo

A continuación se carga el dataset y se grafica la serie de tiempo.

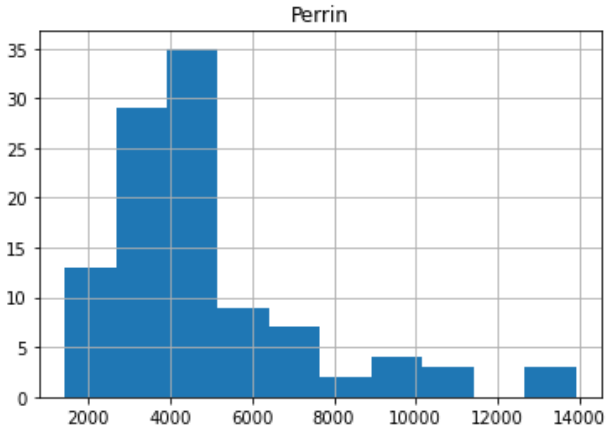
```
001 >>> from skfore.datasets import load_champagne
002 >>> ts = load_champagne()
003 >>> ts.plot()
004
```



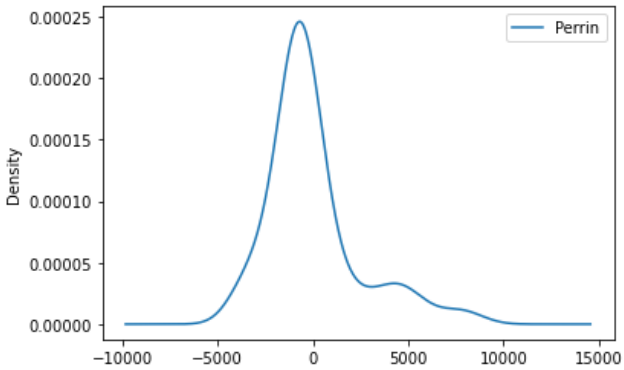
4.1.2 Análisis exploratorio de la serie

En el código presentado a continuación se realiza el análisis exploratorio de la serie. En la línea 003, se llama la función describe para describir estadísticamente la serie; en la línea 005, se visualiza el histograma de la serie; en la línea 007, se construye el gráfico de densidad de la serie; en las líneas 009 y 011, se obtienen los gráficos de autocorrelación y autocorrelación parcial respectivamente. En la línea 013, se construye un gráfico de normalidad y finalmente en la línea 015, se realiza el test de normalidad de Jarque-Bera.

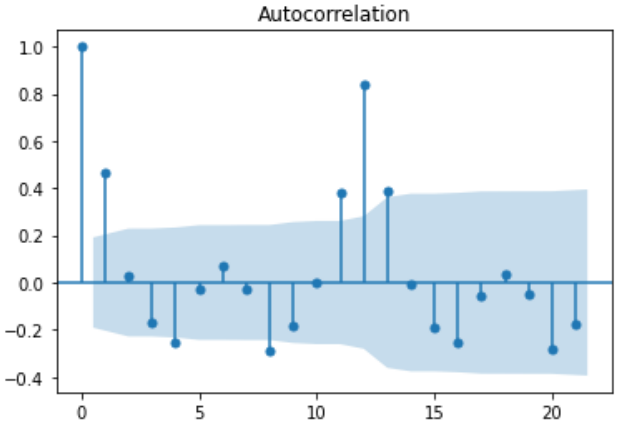
```
001 >>> from skfore.skfore import series_viewer
002 >>> viewer = series_viewer(ts)
003 >>> viewer.describe()
004
      count      105.000000
      mean      4761.152381
      std       2553.502601
      min       1413.000000
      25%       3113.000000
      50%       4217.000000
      75%       5221.000000
      max      13916.000000
      Name: Perrin, dtype: float64
005 >>> viewer.histogram()
006
```



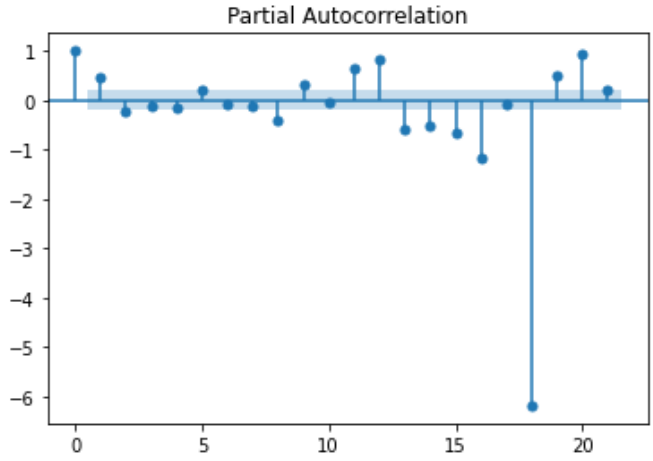

```
007 >>> viewer.density_plot()
008
```



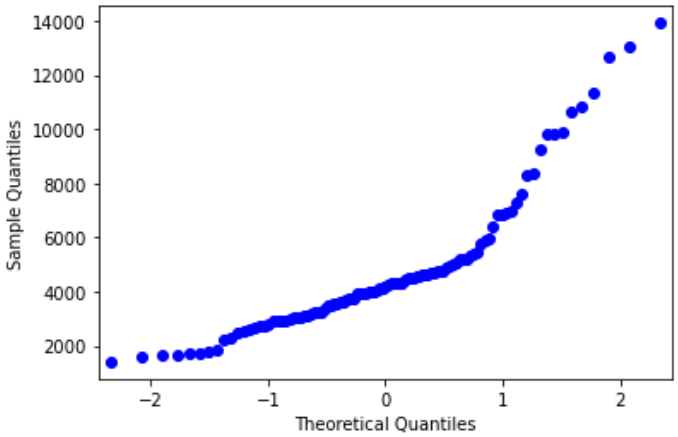

```
009 >>> viewer.ACF_plot()
010
```




```
011 >>> viewer.PACF_plot()
012
```




```
013 >>> viewer.qq_plot()
014
```




```
015 >>> viewer.normality()
016
```

Jarque Bera normality test	
Test statistics value	73.4391
p value	1.11022e-16

4.1.3 Especificación y entrenamiento del modelo

Inicialmente se dividirá la serie en un conjunto de entrenamiento y otro de prueba.

```
001 >>> y_train = ts[0:84]
002 >>> y_test = ts[84:]
```

A continuación, se especifica el modelo en la línea 002 el cuál será una red neuronal con 13 regresores. En la línea 003 se entrena el modelo para el conjunto de entrenamiento.

```
001 >>> from skfore.models.MLP import MLP
002 >>> model = MLP(p=13)
003 >>> model.fit(y_train)
```

4.1.4 Evaluación del modelo

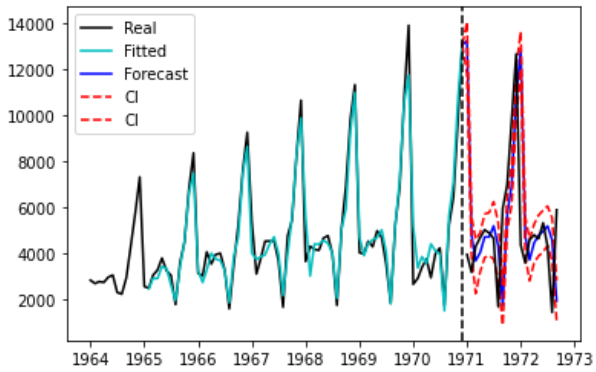
Se evalúa el modelo para el conjunto de entrenamiento en las líneas 001 y 002 y para el conjunto de prueba en las líneas 003 y 004.

```
001 >>> model.calc_error(y_train)
002 441556.9264069677
003 >>> model.calc_error(y_test)
004 60525.8760605172
```

4.1.5 Pronóstico un paso adelante

Este método compara el valor real de la serie con el pronóstico y grafica ambas series de tiempo. Pronostica un paso adelante a la vez. En la línea 001 se pronostica la serie 21 períodos hacia adelante y se define además la serie comparada (que permite tomar el último valor real). En la línea 003 se grafica el pronóstico.

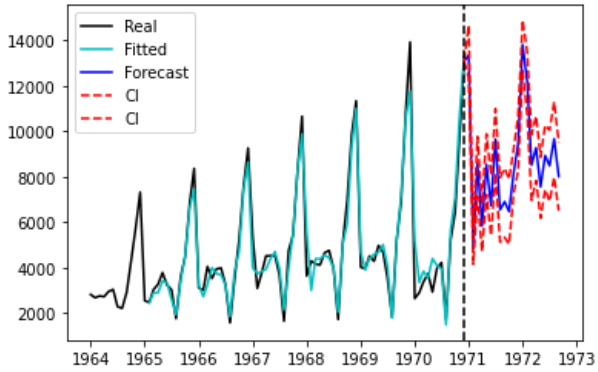
```
001 >>> forecast = model.predict(ts=y_train, periods=21, tsp=y_test, blind=False)
002 >>> model.plot(tse, forecast)
003
```



4.1.6 Pronóstico varios pasos hacia adelante

Este método pronostica una cantidad de períodos deseada hacia el futuro sin tener en cuenta valores pasados. En la línea 001 se especifica el pronóstico y la cantidad de períodos que se quieren obtener. En la línea 002 se grafica este pronóstico.

```
001 >>> forecast = model.predict(ts=y_train, periods=21)
002 >>> model.plot(y_train, forecast)
003
```



4.2 Tutorial 2: Búsqueda de la mejor parametrización de un modelo ARIMA

4.2.1 Carga de los datos

Para este tutorial se utilizará la misma serie utilizada en el ejemplo anterior la cual es cargada mediante el siguiente código.

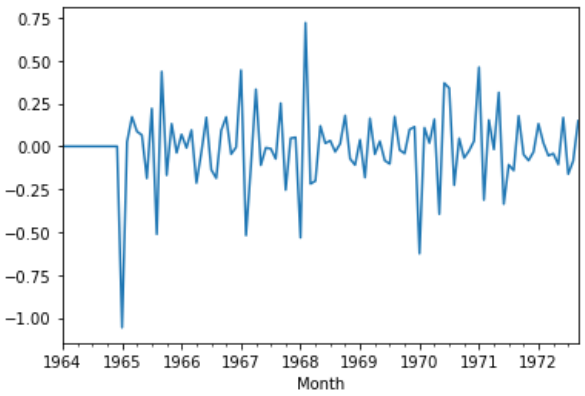
```
001 >>> from skfore.datasets import load_champagne
002 >>> y = load_champagne()
```

4.2.2 Transformación de la serie de tiempo

Después de evaluar esta serie, se especificará una transformación logarítmica, con diferenciación simple y estacionalidad. La serie de tiempo transformada puede verse en la línea 006.

```
001 >>> from skfore.preprocessing.Transformer import transformer
002 >>> my_transform = transformer(trans='log', trend='diff1',
003 ... seasonal='diff')
```

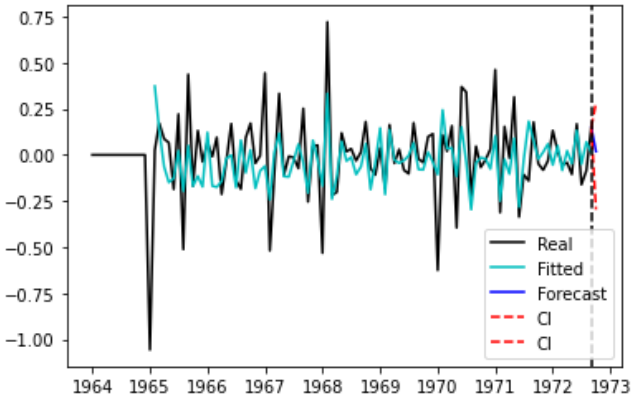
```
004 >>> tyt = my_transform.fit(ts=y)
005 >>> tyt.plot()
006
```



4.2.3 Especificación del modelo base y métricas de error

Se especifica un modelo ARIMA(13,1,1) y se ajusta el modelo para la serie transformada.

```
001 >>> from skfore.models.SARIMA import SARIMA
002 >>> model = SARIMA(p= 13, d=1, q=1, P=0, D=0, Q=0, s=0)
003 >>> model.fit(tyt)
004 >>> forecast = model.predict(ts=tyt, periods=1)
005 >>> model.plot(tyt, forecast)
006
```



4.2.4 Búsqueda del mejor modelo

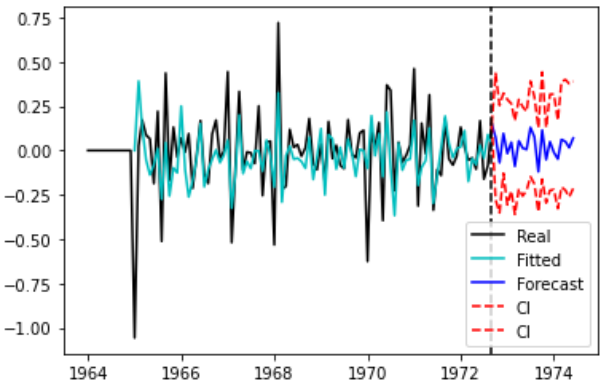
Para la búsqueda del mejor modelo, se utiliza un `GridSearchCV` el cuál recibe un modelo y los parámetros entre los que puede elegir. Dado que se quiere obtener el mejor modelo ARIMA, se especifican los parámetros p, d y q del modelo SARIMA y los demás se ajustan a cero.

```
001 >>> from skfore.pipeline._search import GridSearchCV
002 >>> model = SARIMA()
003 >>> parameters = {'p' : list(range(1,15)),
004 ...              'd' : list(range(3)),
005 ...              'q' : list(range(1,15)),
006 ...              'P' : [0], 'D' : [0], 'Q' : [0], 's' : [0]}
007 >>> my_gs = GridSearchCV(model, parameters)
008 >>> result = my_gs.fit(tyt)
009 >>> print(result.best_score_)
010 >>> print(result.best_model_)
011 0.026908093135437385
012 SARIMA(p = 12, d = 1, q = 13, P = 0, D = 0, Q = 0, s = 0)
```

4.2.5 Pronóstico varios períodos hacia adelante

Utilizando el mejor modelo encontrado en el paso anterior, se pronostican varios períodos hacia adelante en la serie de tiempo.

```
001 >>> best_model = result.best_model_
002 >>> best_model.fit(tyt)
003 >>> forecast = result.best_model_.predict(ts=tyt, periods=21)
004 >>> best_model.plot(tyt, forecast)
005
```



4.3 Tutorial 3: Especificación de un pipeline básico

Si se quieren aplicar varias transformaciones y varios modelos a una serie de tiempo, basta con usar la clase `Pipeline`. Esta recibe como parámetros una lista de transformaciones y una lista de modelos y realiza una combinatoria de la cuál devuelve el resultado de la métrica de error que se elija.

Inicialmente se cargan los datos tal como se muestra el ejemplo anterior.

```
001 >>> from skfore.datasets import load_champagne
```

```
002 >>> y = load_champagne()
```

Al igual que en el ejemplo anterior, en el siguiente código se aplica una transformación de los datos en un modelo ARIMA.

```
001 >>> from skfore.preprocessing.Transformer import transformer
002 >>> from skfore.models.SARIMA import SARIMA
003 >>> from skfore.pipeline.Pipeline import Pipeline
004 >>> my_transform = transformer(trans='log', trend='diff1', seasonal = 'diff')
005 >>> model = SARIMA(p = 13, d = 1, q = 1, P=0, D=0, Q=0, s =0)
006 >>> my_pipeline = Pipeline(ts=y, transformation=[my_transform], model=[model])
007 >>> my_pipeline.fit()
008 transformer(trans = log, trend = diff1, seasonal = diff)
009 SARIMA(p = 13, d = 1, q = 1, P = 0, D = 0, Q = 0, s = 0)
010 0.03816507898760296
```

Utilizando las métricas de error proporcionadas por el pipeline, es posible encontrar los parámetros óptimos del modelo. El pipeline recibe los modelos que determine el usuario y para cada combinación encuentra una métrica de error. Este valor permitirá elegir el modelo que más se ajusta.

```
001 >>> from skfore.preprocessing.Transformer import transformer
002 >>> from skfore.models.SARIMA import SARIMA
003 >>> from skfore.pipeline.Pipeline import Pipeline
004 >>> my_transform = transformer(trans='log', trend='diff1', seasonal = 'diff')
005 >>> model1 = SARIMA(p = 12, d = 1, q = 13, P=0, D=0, Q=0, s =0)
006 >>> model2 = SARIMA(p = 13, d = 1, q = 13, P=0, D=0, Q=0, s =0)
007 >>> model3 = SARIMA(p = 14, d = 1, q = 13, P=0, D=0, Q=0, s =0)
008 >>> my_pipeline = Pipeline(ts=y, transformation=[my_transform], model=[model1,
009 ... model2, model3])
010 >>> my_pipeline.fit()
011 transformer(trans = log, trend = diff1, seasonal = diff)
012 SARIMA(p = 12, d = 1, q = 13, P = 0, D = 0, Q = 0, s = 0)
013 0.026908093135437385
014 transformer(trans = log, trend = diff1, seasonal = diff)
015 SARIMA(p = 13, d = 1, q = 13, P = 0, D = 0, Q = 0, s = 0)
016 0.02830741782363902
017 transformer(trans = log, trend = diff1, seasonal = diff)
018 SARIMA(p = 14, d = 1, q = 13, P = 0, D = 0, Q = 0, s = 0)
019 0.026969274082634637
```

En la línea 013 se presenta el menor error que corresponde al mismo modelo presentado en el ejemplo anterior.

4.4 Tutorial 4: Búsqueda de los hiper-parámetros de diferentes tipos de modelos usando GridSearchCV

En los casos en que no se tenga claro cuál es el mejor modelo para realizar pronósticos de una serie de tiempo, es posible usar la función de búsqueda, la cual recibe un modelo y unos parámetros y devuelve una métrica de error elegida para cada combinación de parámetros.

Inicialmente se cargan los datos tal como se muestra en los anteriores tutoriales.

```
001 >>> from skfore.datasets import load_champagne
002 >>> y = load_champagne()
```

A continuación, para cada tipo de modelo se ejecuta la búsqueda de los mejores parámetros.

En el primer bloque se busca la mejor configuración de parámetros de un modelo autorregresivo.

```
001 >>> from skfore.pipeline._search import GridSearchCV
002 >>> from skfore.models.AR import AR
003 >>> model = AR()
004 >>> parameters = {'p' : list(range(10,14))}
005 >>> my_gs = GridSearchCV(model, parameters)
006 >>> result = my_gs.fit(y)
007 >>> print(result.best_score_)
008 >>> print(result.best_model_)
009 489196.19211170037
010 AR(p = 13, intercept = 1363.0104671132099, phi = [-0.13448383  0.90562808
0.05443729 -0.07849188  0.06473445 -0.12622993 0.05777489 -0.07094086  0.04010152
-0.10263322  0.04614906 -0.09732988 0.21275987])
```

En el segundo bloque se busca la mejor configuración de parámetros de un modelo ARIMA tal como se ejecutó en el tutorial anterior.

```
001 >>> from skfore.pipeline._search import GridSearchCV
002 >>> from skfore.models.SARIMA import SARIMA
003 >>> model = SARIMA()
004 >>> parameters = {'p' : list(range(10,14)),
005 ...               'd' : list(range(1,3)),
006 ...               'q' : list(range(10,14)),
007 ...               'P' : [0], 'D' : [0], 'Q' : [0], 's' : [0]}
008 >>> my_gs = GridSearchCV(model, parameters)
009 >>> result = my_gs.fit(y)
010 >>> print(result.best_score_)
011 >>> print(result.best_model_)
012 481193.5993673794
013 SARIMA(p = 11, d = 1, q = 13, P = 0, D = 0, Q = 0, s = 0)
```

En el último bloque se busca la mejor configuración de parámetros de un modelo de redes neuronales.

```
001 >>> from skfore.pipeline._search import GridSearchCV
002 >>> from skfore.models.MLP import MLP
003 >>> model = MLP()
004 >>> parameters = {'p' : list(range(10,14))}
005 >>> my_gs = GridSearchCV(model, parameters)
006 >>> result = my_gs.fit(y)
007 >>> print(result.best_score_)
008 >>> print(result.best_model_)
009 345983.4229859572
010 MLP(p = 13, model = MLPRegressor())
```

Según los resultados obtenidos en cada bloque, el mejor estimador es una red neuronal con 13 valores pasados de entrada.

4.5 Tutorial 5: Uso de la librería con modelos de scikit-learn

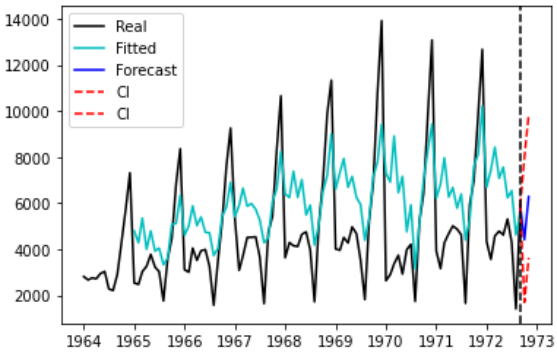
Mediante la clase `Reg2TS` es posible utilizar cualquier modelo de pronóstico de scikit-learn sin necesidad de convertir la serie de tiempo a un modelo de regresión.

Inicialmente se cargan los datos tal como se muestra en los ejemplos anteriores.

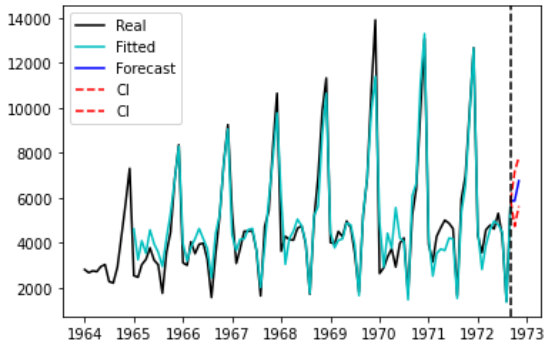
```
001 >>> from skfore.datasets import load_champagne
002 >>> y = load_champagne()
```

A continuación, se aplican un modelo de redes neuronales, un modelo lineal Lasso y un Random Forest bajo la misma estructura.

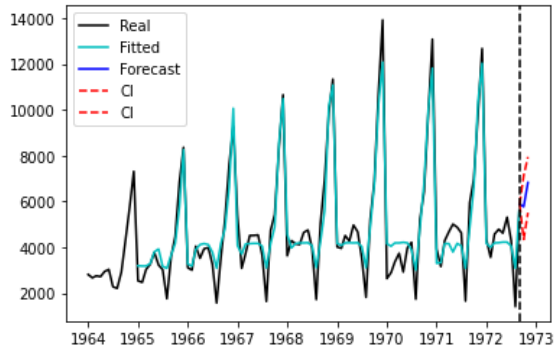
```
001 >>> from skfore.models.super_model import Reg2TS
002 >>> from sklearn.neural_network import MLPRegressor
003 >>> model1 = Reg2TS(p=12, model=MLPRegressor, hidden_layer_sizes=(100, ))
004 >>> model1.fit(y)
005 >>> forecast1 = model1.predict(y, 2)
006 >>> model1.plot(y, forecast1)
007
```




```
001 >>> from skfore.models.super_model import Reg2TS
002 >>> from sklearn.linear_model import Lasso
003 >>> model2 = Reg2TS(p=12, model=Lasso, alpha=0.1)
004 >>> model2.fit(y)
005 >>> forecast2 = model2.predict(y, 2)
006 >>> model2.plot(y, forecast2)
007
```




```
001 >>> from skfore.models.super_model import Reg2TS
002 >>> from sklearn.ensemble import RandomForestRegressor
003 >>> model3 = Reg2TS(p=12, model=RandomForestRegressor, max_depth=2,
004 ... random_state=0)
005 >>> model3.fit(y)
006 >>> forecast3 = model3.predict(y, 2)
007 >>> model3.plot(y, forecast3)
008
```



Las métricas de error para estos modelos se presentan a continuación.

```
001 >>> print(model1.calc_error(y))
002 >>> print(model2.calc_error(y))
003 >>> print(model3.calc_error(y))
004 5126994.231604024
005 626812.4001122902
006 535073.15345921
```

Siguiendo estas métricas, es posible determinar el modelo de Random Forest como el mejor pronóstico para esta serie.

El modelo `Reg2TS` permite además incluir parámetros dentro de la clase `GridSearchCV` y buscar su mejor combinación.

5. Comparación de técnicas de aprendizaje de máquinas para el pronóstico de serie de tiempo

5.1 Introducción

La predicción de series de tiempo no lineales ha sido un área de estudio ampliamente investigada. En los últimos años, las metodologías de aprendizaje de máquinas más utilizadas para cualquier tipo de predicción también han sido implementadas para el pronóstico de series de tiempo. Entre estas técnicas se encuentran los modelos de redes neuronales artificiales o perceptrón multicapa (*MLP*) y los modelos híbridos neuro-difusos [38].

Las redes neuronales artificiales se han utilizado como un aproximador funcional universal [39]. Estas utilizan comúnmente la aproximación de *feed-forward*, *back-propagation* (FFBP). Los parámetros del modelo tales como las variables de entrada, el número de capas ocultas, el número de nodos en cada capa oculta y la elección de las funciones de transferencia dependen del problema y es necesario la prueba y error para encontrar el mejor modelo para un problema en específico.

Los modelos híbridos neuro-difusos han sido más recientemente propuestos. La mayor parte de ellos están basados en el uso de conjuntos difusos unidimensionales en los antecedentes de las reglas, aunque también se han reportado algunas experiencias en el uso de conjuntos multidimensionales para la predicción de series de tiempo no lineales [38].

En esta sección se comparan dos series de tiempo ampliamente estudiadas utilizando diferentes variaciones de los modelos de redes neuronales e híbridos neuro-difusos mediante el uso de la librería scikit-forecasts. Entre las variaciones presentadas, se encuentra también la transformación de la serie de tiempo.

5.2 Metodologías utilizadas

Para el pronóstico de las series de tiempo se utilizan un modelo SARIMA tradicional, un modelo de redes neuronales artificiales y un modelo adaptativo multidimensional neuro-difuso.

El modelo SARIMA es un modelo autorregresivo integrado de medias móviles estacional con una dependencia de un tiempo en el pasado con un retraso *s*. Este modelo fue definido por Box & Jenkins en 1970.

El modelo de redes neuronales artificiales representado por el modelo MLP (*Multi Layer Perceptron)* es un algoritmo de aprendizaje supervisado que aprende una función mediante el entrenamiento con un conjunto de datos, con *m* dimensiones de entrada y *o* dimensiones de salida. Dado un conjunto de características y un objetivo (el cual es comúnmente minimizar la función de error), el algoritmo puede aprender una función no lineal para encontrar un valor. Entre la capa de entrada y la de salida, puede haber más capas no lineales, llamadas capas ocultas. Cada neurona en la capa oculta transforma el valor de la capa anterior en una sumatoria linear ponderada seguida por una función de activación no lineal. La capa de salida recibe los valores ponderados de la última capa oculta y los transforma en los valores de salida (o pronósticos).

Para una serie de tiempo, el conjunto de características de entrada está representado por los *p* valores pasados de la serie.

El modelo híbrido neuro-difuso, es un modelo AMNFIS (*Adaptive Multidimensional Neuro-Fuzzy Inference System*) desarrollado en [38] bajo un contexto de control predictivo. En AMNFIS, las funciones de pertenencia multidimensionales son calculadas como una función de la distancia del vector de entrada a los centros de los clústeres que representan cada conjunto difuso, mientras que los consecuentes de las reglas son funciones poligonales de orden uno, que en el contexto de las series de tiempo corresponden a modelos autorregresivos. Entre las ventajas de este modelo se encuentran: su arquitectura es muy simple en comparación con otros sistemas neuro-difusos, hay un número inferior de parámetros que deben ser ajustados por el algoritmo de optimización y la cantidad y posición de los clústeres puede ser fácilmente encontrada por el algoritmo de especificación.

Además de estos modelos, se desarrolló un comparativo utilizando los modelos de Huber y Random Forest tomados de scikit-learn.

5.3 Información utilizada

La serie *WWWUsage* es un dataset que contiene el número de usuarios loggeados en un servidor de internet cada minuto durante 100 minutos. Es un modelo que indica no estacionariedad. Los primeros 80 datos son usados para la estimación del modelo, y los 20 restantes para su predicción.

La serie *Paper* es un dataset que contiene las ventas de la industria del papel (en miles de francos) entre Enero de 1963 y Diciembre de 1972 (120 observaciones). Este conjunto de datos muestra estacionalidad y una creciente tendencia. Los primeros 100 datos se utilizan para el ajuste de los modelos, y los 20 finales para evaluar la precisión de la predicción.

5.4 Resultados obtenidos

5.4.1 The internet data set: WWWUsage

Para los primeros modelos de referencia, se pronostica la serie original sin ningún tipo de transformación. Se comparan los modelos híbridos neuro-difuso, de redes neuronales y un modelo ARIMA.

El modelo AMNFIS presentado en el documento de referencia usa como entradas los rezagos 1 al 4, y se encontró que un modelo con cuatro conjuntos difusos multidimensionales presenta el mejor balance la precisión del ajuste para las muestras de calibración y predicción.

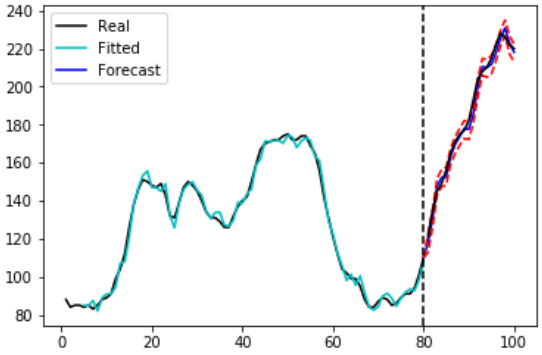
El modelo de redes neuronales presentado en el documento de referencia usa como entradas los rezagos usa como entradas los rezagos 1, 2, 3 y 4.

Tabla 5-1: Resultados comparados de pronóstico para la serie WWWUsage con AMNFIS.

Referencia	**Entrenamiento MSE (MAD)**	**Prueba MSE (MAD)**
AMNFIS [38]	5.87 (1.84)	8.06 (2.45)
AMNFIS (scikit-forecasts)	7.44 (2.19)	11.14 (2.85)
Modelo ANN en [38]	7.00 (2.10)	9.25 (2.25)
scikit-forecast MLP	9.54 (2.44)	105.73 (8.60)

HuberRegressor (Reg2TS)	11.74 (2.72)	21.07 (3.79)

Figura 5-1: Estimación y pronóstico de WWWUsage mediante el modelo AMNFIS.



5.4.2 The printing and writing paper data set: Paper

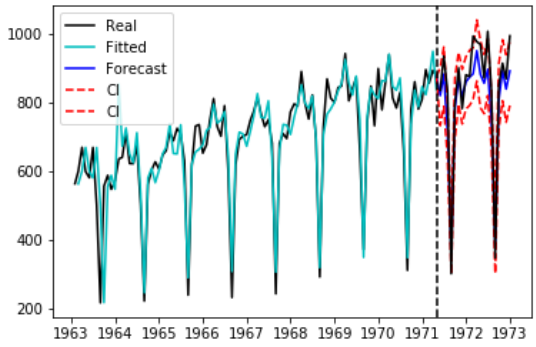
Para los primeros modelos de referencia, se pronostica la serie original sin ningún tipo de transformación. Se comparan los modelos híbridos neuro-difuso, de redes neuronales y un modelo SARIMA.

El modelo AMNFIS presentado en el documento de referencia usa como entradas los rezagos 1, 12 y 13, y L = 4. El modelo de redes neuronales presentado en el documento de referencia usa como entradas los rezagos usa como entradas los rezagos 1, 7 y 12. Esta serie se pronostica con un modelo ARIMA(0, 1, 1)(0, 1, 1)$_{12}$ en [38], el cuál en realidad es un modelo estacional (SARIMA).

Tabla 5-2: Resultados comparados de pronóstico para la serie Paper con AMNFIS.

Referencia	**Entrenamiento MSE (MAD)**	**Prueba MSE (MAD)**
Modelo AMNFIS en [38]	1605.10 (31.10)	4222.5 (53.19)
scikit-forecast AMNFIS	1417.73 (29.64)	26188.92 (135.73)
Modelo ANN en [38]	1951.5 (36.72)	5874.8 (62.62)
scikit-forecast MLP	835.09 (23.42)	44321.19 (149.47)
Modelo ARIMA en [38]	1715.5 (34.04)	4791.7 (54.72)
scikit-forecast SARIMA	4464.11 (42.82)	4086.08 (51.85)
RandomForestRegressor (Reg2TS)	1745.22 (28.50)	44466.74 (163.11)

Figura 5-2: Estimación y pronóstico de la serie Paper mediante un modelo SARIMA.



6. Conclusiones y trabajo futuro

6.1 Cumplimiento de objetivos

6.1.1 Cumplimiento de objetivo general

Como se propuso en el Objetivo General 1.1, se desarrolla una librería en Python que permite la inserción de nuevos modelos dentro de la herramienta de una forma fácil y con solo unas cuantas líneas de código. Esta librería es de código libre (distribuida bajo la licencia MIT) y está específicamente diseñada para el pronóstico de series de tiempo no lineales. Está considerada además, como una extensión de scikit-learn exclusivamente para series de tiempo.

Entre otras de sus funcionalidades se encuentran: cargar fácilmente múltiples series de tiempo desde un directorio, realizar un análisis exploratorio de las series y los residuales, transformaciones de las series y búsqueda del mejor modelo para una o múltiples series de tiempo.

6.1.2 Cumplimiento de los objetivos específicos

Como se propuso en el Objetivo Específico 1, se realiza el diseño, implementación y prueba de la librería bajo el esquema de scikit-learn. La librería cuenta con una estructura similar, lo cual aplana la curva de aprendizaje y utiliza muchas de sus funcionalidades para el desarrollo de nuevos modelos de pronóstico.

La librería implementa un modelo ARIMA, tal como se propuso en el Objetivo Específico 2. El modelo está basado en la librería Statsmodels y es un modelo autorregresivo integrado de medias móviles estacional con una dependencia de un tiempo en el pasado con un retraso *s*.

La librería implementa un modelo de redes neuronales artificiales, tal como se propuso en el Objetivo Específico 3. Este modelo está construido bajo el esquema de scikit-learn. El modelo MLP (*Multi Layer Perceptron)* es un algoritmo de aprendizaje supervisado que aprende una función mediante el

entrenamiento con un conjunto de datos, con *m* dimensiones de entrada y *o* dimensiones de salida. Las dimensiones de entrada son representadas por una matriz de regresores de dimensión *p* (valores pasados de la serie u orden del modelo)

La librería implementa un modelo híbrido neuro-difuso, tal como se propuso en el Objetivo Específico 4. Este es un modelo AMNFIS (*Adaptive Multidimensional Neuro-Fuzzy Inference System*) se construye desde cero basándose en [38]. En AMNFIS, las funciones de pertenencia multidimensionales son calculadas como una función de la distancia del vector de entrada a los centros de los clústeres que representan cada conjunto difuso, mientras que los consecuentes de las reglas son funciones poligonales de orden uno, que en el contexto de las series de tiempo corresponden a modelos autorregresivos.

Adicionalmente a los objetivos propuestos, se implementó una clase que permite elegir cualquiera de los modelos de regresión disponibles en scikit-learn. Esta funcionalidad permite encontrar los mejores parámetros del modelo y pronosticar varios períodos hacia adelante sin necesidad de convertir la serie de tiempo en una matriz de regresión.

6.2 Conclusiones

La creación de un entorno integrado de análisis de series de tiempo que facilite la inserción de nuevos modelos dentro de la herramienta se ocupa de muchas de las restricciones en la replicabilidad y reproducibilidad de los resultados. Que este entorno sea creado bajo una herramienta libre y que cumpla con las características adecuadas, fomenta el uso de los entornos colaborativos y el trabajo conjunto.

La librería scikit-forecasts está desarrollada con el propósito de generar un entorno de trabajo con metodologías integradas y hacerlo fácilmente accesible, más amable y bajo una estructura en la que el investigador y el profesional del área pueda integrar un modelo (o usar diferentes modelos) para analizar o pronosticar una o más series de tiempo. Además, cuenta tanto con modelos de pronóstico de series de tiempo tradicionales, como con metodologías más nuevas en análisis y pronóstico de datos, como las metodologías de aprendizaje de máquinas, enfocadas a las series de tiempo.

Con esta librería, se pueden obtener resultados y predicciones fácilmente, sin necesidad de recurrir a una transformación de los datos en términos de una regresión. Los resultados obtenidos mediante el uso de la librería logran ser comparables con los obtenidos en publicaciones académicas usando solo unas pocas líneas de código.

La librería hace posible además una comparación fácil y rápida entre modelos, lo cual permite al investigador elegir fácilmente entre ellos haciendo que los pronósticos sean cada vez más acertados.

6.3 Trabajo futuro

La librería permite integrar fácilmente modelos desde scikit-learn. En un trabajo futuro, es posible realizar una integración similar para otras librerías como TensorFlow, específicamente centradas en modelos de *deep learning*. Esta extensión permitiría la implementación de modelos de redes neuronales de aprendizaje profundo como las LSTM.

También es posible realizar otra extensión de la librería para modelos de volatilidad como los son los modelos ARCH, GARCH, CHARMA, modelos estocásticos de volatilidad, entre otros definidos en [40]. La volatilidad es un factor importante en la compra y venta de opciones. En este caso, la volatilidad significa la variación condicional del rendimiento del activo subyacente.

A. Anexo: Ruta de acceso e instalación de la librería scikit-forecasts

scikit-forecasts es un librería de código libre (distribuida bajo la licencia MIT) desarrollada en Python para el análisis y pronóstico de series de tiempo no lineales. Está considerada como una extensión de scikit-learn diseñada exclusivamente para series de tiempo.

Esta librería fue desarrollada en la versión 3.7 y puede ser instalada usando el siguiente comando.

```
$ pip install scikit-forecasts
```

scikit-forecasts está desarrollada para el análisis y pronóstico de series de tiempo no lineales y se puede utilizar para:

- Incluir fácilmente modelos dentro de la herramienta
- Cargar fácilmente múltiples series de tiempo desde un directorio
- Análisis exploratorio de series de tiempo
- Transformaciones de series de tiempo
- Pronóstico de series de tiempo
- Encontrar el mejor modelo para una o múltiples series de tiempo

La documentación completa de esta librería puede ser encontrada en https://jdvelasq.github.io/scikit-forecasts/

Bibliografía

[1] D. R. Brillinger, “Time Series: General,” 2000.

[2] R. S. Tsay, “Time series and forecasting: Brief history and future research,” in *Statistics in the 21st Century*, CRC Press, 2001, pp. 121–131. doi: 10.2307/2669408.

[3] F. Perez, B. E. Granger, U. C. Berkeley, C. Poly, and S. L. Obispo, “AN OPEN SOURCE FRAMEWORK FOR INTERACTIVE, COLLABORATIVE AND REPRODUCIBLE SCIENTIFIC COMPUTING AND EDUCATION.”

[4] C. M. Kelty, *Two bits the cultural significance of free software*. 2008.

[5] “Top Programming Languages 2020 - IEEE Spectrum.” https://spectrum.ieee.org/at-work/tech-careers/top-programming-language-2020?referrer=%2F (accessed Apr. 20, 2021).

[6] “Series de Tiempo : Pricing.” https://www.pricing.cl/conocimiento/series-de-tiempo/ (accessed Aug. 18, 2022).

[7] “Series Temporales Introducción”.

[8] “The 2017 Top Programming Languages - IEEE Spectrum.” https://spectrum.ieee.org/computing/software/the-2017-top-programming-languages (accessed Apr. 20, 2021).

[9] “pandas - Python Data Analysis Library.” https://pandas.pydata.org/ (accessed Apr. 20, 2021).

[10] “Matlab CAPTAIN Toolbox for time series analysis and forecasting.” http://www.es.lancs.ac.uk/cres/captain/default.htm (accessed Apr. 20, 2021).

[11] “Deep Learning Toolbox - MATLAB.” https://www.mathworks.com/products/deep-learning.html (accessed Apr. 20, 2021).

[12] “Package ‘forecast’ Title Forecasting Functions for Time Series and Linear Models Description Methods and tools for displaying and analysing univariate time series forecasts including exponential smoothing via state space models and automatic ARIMA modelling,” 2021.

[13] “Package ‘neuralnet’ Title Training of Neural Networks,” 2019.

[14] “SAS/ETS Time Series Forecasting System.” https://support.sas.com/rnd/app/ets/cap/ets_forecasting.html (accessed Apr. 20, 2021).

[15] “Title stata.com forecast-Econometric model forecasting.”

[16] “SPSS Software | IBM.” https://www.ibm.com/analytics/spss-statistics-software (accessed Apr. 20, 2021).

[17] “Introduction — statsmodels.” https://www.statsmodels.org/stable/index.html (accessed Apr. 20, 2021).

[18] D. van Kuppevelt, C. Meijer, F. Huber, A. van der Ploeg, S. Georgievska, and V. T. van Hees, “Mcfly: Automated deep learning on time series,” *SoftwareX*, vol. 12, Jul. 2020, doi: 10.1016/j.softx.2020.100548.

[19] “Welcome to TSFEL documentation! — TSFEL 0.1.4 documentation.” https://tsfel.readthedocs.io/en/latest/ (accessed Apr. 20, 2021).

[20] “Software de hojas de cálculo Microsoft Excel | Microsoft 365.” https://www.microsoft.com/es-es/microsoft-365/excel (accessed Apr. 20, 2021).

[21] “NeuralTools para análisis de predicción usando redes neuronales inteligentes - Palisade.” https://www.palisade-lta.com/neuraltools/ (accessed Jul. 02, 2021).

[22] *Time Series Analysis and Its Applications - With R Examples | Robert H. Shumway | Springer*. 2006.

[23] “Nonlinear Time Series Models 18.1 Introduction.”

[24] R. J. Hyndman and G. Athanasopoulus, “Forecasting: Principles and Practice (2nd ed).” https://otexts.com/fpp2/ (accessed Apr. 20, 2021).

[25] “TensorFlow.” https://www.tensorflow.org/?hl=es-419 (accessed Apr. 20, 2021).

[26] “Keras: the Python deep learning API.” https://keras.io/ (accessed Apr. 20, 2021).

[27] “PyTorch.” https://pytorch.org/ (accessed Apr. 20, 2021).

[28] “scikit-learn: machine learning in Python — scikit-learn 0.24.1 documentation.” https://scikit-learn.org/stable/ (accessed Apr. 20, 2021).

[29] “Time Series Analysis: Forecasting and Control - George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, Greta M. Ljung - Google Libros.” https://books.google.com.co/books?hl=es&lr=&id=rNt5CgAAQBAJ&oi=fnd&pg=PR7&dq=Time+Series+Analysis,+Forecasting+and+Control.+box+jenkins&ots=DK28BOj_QG&sig=HkqPSrBFGGDao_IztPkTUtXxxH0#v=onepage&q=Time Series Analysis%2C Forecasting and Control. box jenkins&f=false (accessed Apr. 30, 2021).

[30] M. G. Bulmer, “A Statistical Analysis of the 10-Year Cycle in Canada,” *The Journal of Animal Ecology*, vol. 43, no. 3, p. 701, Oct. 1974, doi: 10.2307/3532.

[31] “SILSO | World Data Center for the production, preservation and dissemination of the international sunspot number.” http://sidc.oma.be/silso/home (accessed Apr. 30, 2021).

[32] “UCI Machine Learning Repository: Internet Usage Data Data Set.” https://archive.ics.uci.edu/ml/datasets/Internet+Usage+Data (accessed Apr. 30, 2021).

[33] C. Beaumont, S. Makridakis, S. C. Wheelwright, and V. E. McGee, “Forecasting: Methods and Applications,” *J Oper Res Soc*, vol. 35, no. 1, p. 79, Jan. 1984, doi: 10.2307/2581936.

[34] J. D. V. HENAO, C. O. Z. PEREZ, and C. J. F. CARDONA, “A COMPARISON OF EXPONENTIAL SMOOTHING AND NEURAL NETWORKS IN TIME SERIES PREDICTION,” *DYNA*, vol. 80, no. 182, pp. 66–73, Nov. 2013.

[35] “Time Series Forecast Study with Python: Monthly Sales of French Champagne.” https://machinelearningmastery.com/time-series-forecast-study-python-monthly-sales-french-champagne/ (accessed Apr. 20, 2021).

[36] “RPubs - Análisis de normalidad.” https://rpubs.com/PAVelasquezVasconez/354989 (accessed Apr. 20, 2021).

[37] C. M. Jarque and A. K. Bera, “Efficient tests for normality, homoscedasticity and serial independence of regression residuals,” *Economics Letters*, vol. 6, no. 3, pp. 255–259, 1980, doi: 10.1016/0165-1765(80)90024-5.

[38] J. D. Velásquez, “Adaptive Multidimensional Neuro-Fuzzy Inference System for Time Series Prediction,” *IEEE Latin America Transactions*, vol. 13, no. 8, pp. 2694–2699, Aug. 2015, doi: 10.1109/TLA.2015.7332151.

[39] M. Ghiassi, H. Saidane, and D. K. Zimbra, “A dynamic artificial neural network model for forecasting time series events,” *International Journal of Forecasting*, vol. 21, no. 2, pp. 341–362, Apr. 2005, doi: 10.1016/j.ijforecast.2004.10.008.

[40] R. S. Tsay, “Analysis of Financial Time Series Financial Econometrics.”