

Abstract

El artículo presenta una comparación de varias paqueterías de Python para series de tiempo. Su objetivo es presentar un panorama de diferentes cometidos y métodos implementados, esto incluye documentación, dependencias y sus comunidades. Fueron analizados 40 paqueterías, y las clasificaron de acuerdo a sus labores implementadas, los métodos para el preprocesamiento y licencias. Entre todos los objetivos, se encontró que el más frecuente son los pronósticos. La mitad de las paqueterías proveen acceso a bases de datos reales o simuladas.

1. Introducción

Una serie de tiempo es un conjunto de datos generados de medidas sucesivas a lo largo del tiempo. El análisis de este tipo de datos ha encontrado varias aplicaciones en diferentes campos de estudio, como lo es la finance, la salud e incluso el monitoreo de redes computacionales así como en el medio ambiente. Existe la tendencia de reducir los costos de medir y guardar dichos datos, lo que ha permitido el incremento del llamado *Big Data* y tecnologías de análisis como *machine learning* y *data mining*. Por tanto, a medida que el número de casos para análisis de series aumenta, también aumenta la cantidad de científicos de datos, ingenieros de datos, analistas, ingenieros de software que se ven en la necesidad de usar librerías para series de tiempo.

La cotejación sistemática propuesta en el artículo se basa en el lenguaje de programación Python, pues es el más usado entre científicos de datos. La meta del artículo no es evaluar la calidad de las implementaciones por sí mismas, sino para proveer un panorama de posibilidades en las herramientas del análisis de series de tiempo.

2. Publicaciones similares

El análisis de series temporales es un amplio campo de estudio que cubre demasiados dominios de aplicación. La literatura inspecciona en; ya sea métodos de análisis, clasificación y *clustering*, detección de anomalías, reconocimiento de patrones, o reducción de dimensionalidad o encuestas financieras, tecnologías de la información, industria o medicina.

Sin embargo, la existencia de implementaciones es en vez enlistada en una manera poco sistemática, como muchas citas dadas por el autor. El artículo se jacta de que no existe tal estudio estructurado y sistemático, por lo que es objetivo de dicho trabajo.

3. Metodología

Se conduce una revisión sistemática de acuerdo a Kitchenham et. al. en su trabajo *A systematic review of systematic review process research in software engineering*. Sin embargo, el autor menciona que dicho estudio se enfoca en la literatura en lugar de las paqueterías de software. La Metodología aplicada la muestra en la siguiente figura

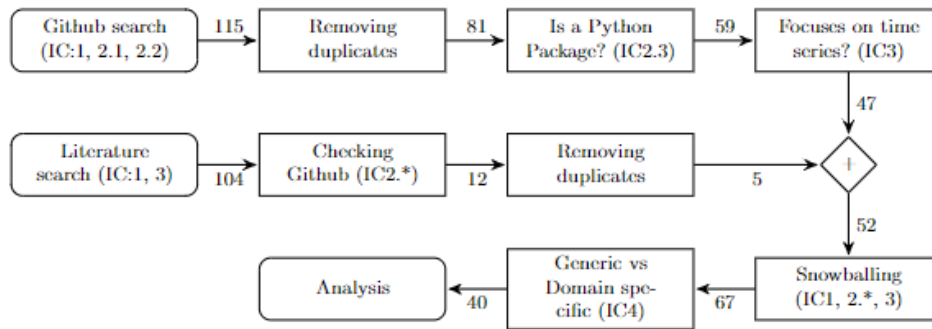


Figura 1: Búsqueda y filtrado de procesos, las etiquetas de las flechas indican el número de repositorios discriminados tras cada paso.

3.a. Preguntas de investigación

Se formaliza la meta del artículo. La idea es **análizar** las paqueterías de Python dedicadas a análisis de series de tiempo con el propósito de estructurar las implementaciones disponibles con respecto a las implementaciones desde el punto de vista del practicante. Los cuestionamientos son:

1. ¿Cuáles cometidos para análisis de series temporales existen? y ¿Cuáles de estas implementaciones están en alguna paquetería de Python?
2. ¿Cómo las paqueterías apoyan la evaluación del resultado producido?
3. ¿Cómo las paqueterías apoyan su uso, que percepción se dá para estimar la duración de dichas paqueterías ?

3.b. Criterio de inclusión

Para guiar la revisión y filtración de las paqueterías relevantes, los autores definieron los siguientes criterios de inclusión. El paquete debe ser de dominio público, escrito en Python, y disponible en GitHub. El paquete debe estar bajo activo mantenimiento. Debe tener más de 100 estrellas de GitHub. Debe estar listado en el PyPI (Python Package Index, un repositorio de software). Se debe poder instalar via pip o conda. El paquete explícitamente debe enfocarse en análisis de series temporales. Finalmente, se excluyen paqueterías que se enfocan en series de tiempo para un campo en particular, como finanzas, ciencias de la tierra, audio, etc.

3.c. Buscando repositorios de dominio público en GitHub

El artículo nos menciona que filtraron los repositorios de GitHub por lenguaje (Python), número de estrellas, y aquellos repositorios que se hayan actualizado después de Julio del 2020.

Con la finalidad de seleccionar una lista de temas relevantes, manualmente se seleccionó una **lista de ocho paqueterías** de Python conocidas para ser usadas en análisis de series temporales: pandas, numpy, scipy, statsmodel, ruptures, tsfresh, tslearn, and sktime. Se examinaron los temas usados en cada paquetería. En total se consideran 16 funciones o temas diferentes. De los 115 repositorios encontrados en GitHub, tras el proceso de inclusión-discriminación ilustrado en Fig.1, se llegó a que solo 40 repositorios son candidatos a examinar.

3.d. Buscando bases de datos en bibliografía científica

La búsqueda de paqueterías solo en un repositorio podría no ser suficiente para cubrir todas las paqueterías existentes. Como por ejemplo fue el caso de la paquetería *tsfresh*. Los autores usaron bases de datos como IEEE Xplore, ACM Digital Library, Web of Science, Scopus y Journal of Open Source Software (JOSS), Zenodo. Por ejemplo, para las primeras cuatro, usaron un query para filtrar dichas paqueterías. Se encontró que cinco de estos repositorios no se encontraban en GitHub, estas fueron: *tsfresh*, *neurodsp*, *EoN*, *nolds*, y *pastas*.

3.e. Snowballing

Con el propósito de extender la investigación, se usó una acercamiento *snowballing*. Manualmente, los autores revisaron cada paquetería para encontrar link a otras librerías. De igual forma, manualmente se revisó la documentación y los repositorios de GitHub para ver si incluían trabajos relacionados. 15 nuevas paqueterías se incluyeron en el análisis después de lo anteriormente descrito.

3.f. Genérico vs dominio específico

Como ya se habló antes, se seleccionaron los paquetes que no se enfocan en una disciplina en particular que no sea análisis de series temporales.

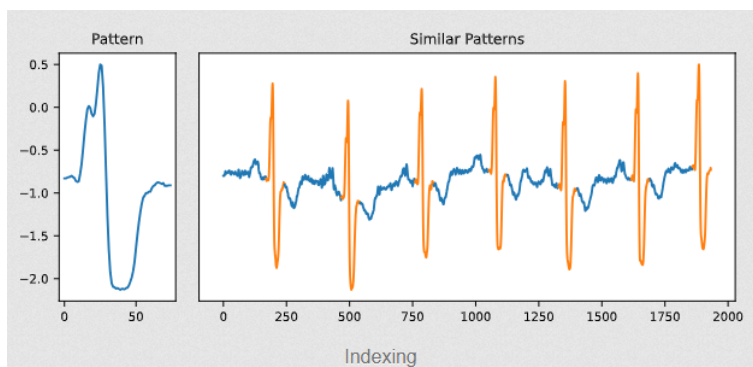
4. Resultados

4.a. Implementación de cometidos para análisis de series temporales

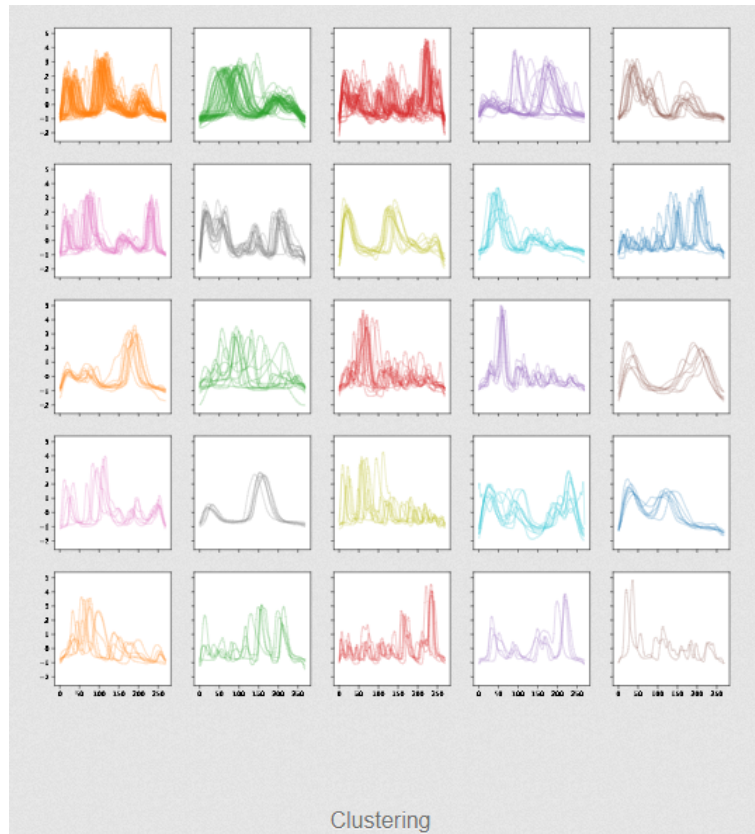
Para contestar la primera pregunta de investigación, primeramente se revisaron los cometidos presentes en la literatura y luego se analizaron en cada uno de los 40 paquetes clasificando cual es capaz de procesar dicho cometido.

Definición de los cometidos

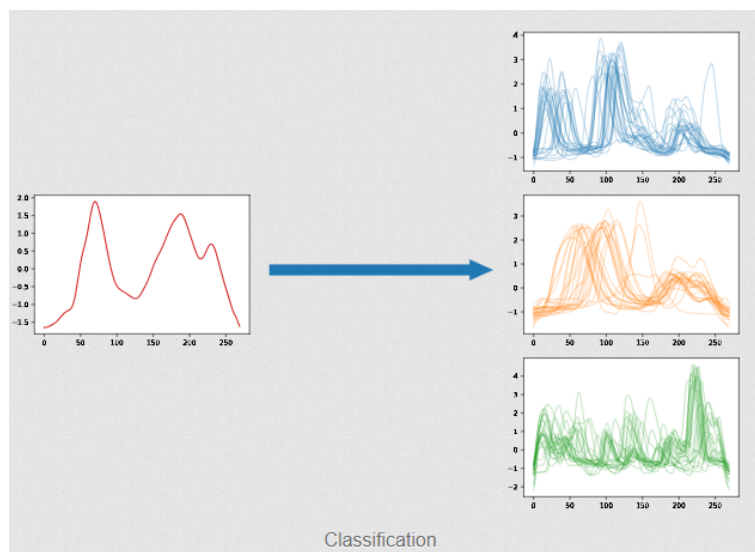
1. **Indexing:** Dada una serie de tiempo y algunas similitudes de medida, encuentra la serie de tiempo más cercana.



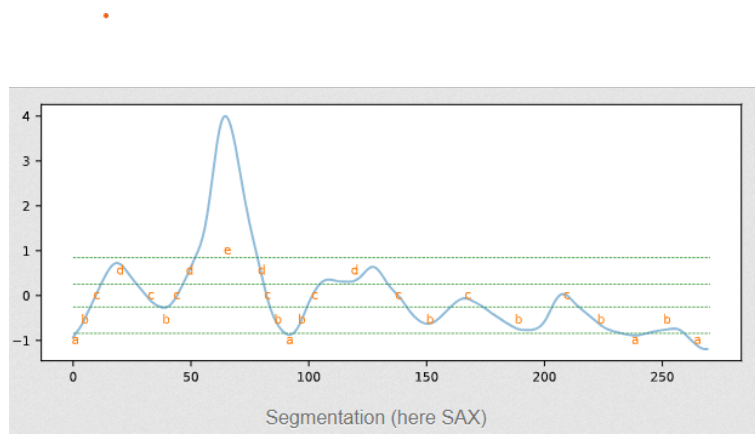
2. **clustering:** Encuentra grupos de series temporales similares.



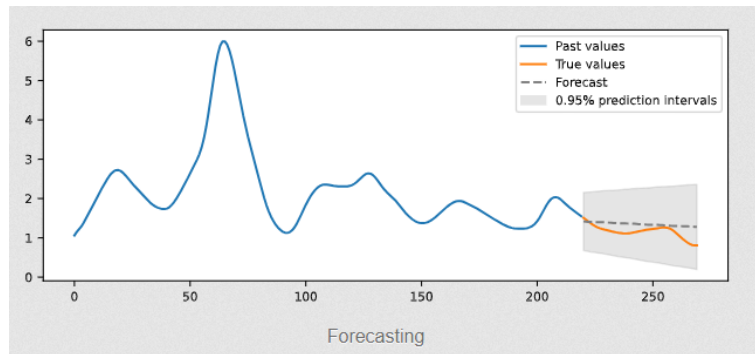
3. **clasificación:** Asigna a cada serie temporal una clase predefinida.



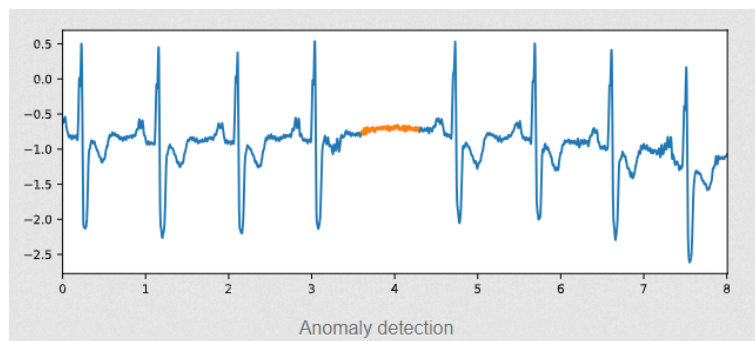
4. **Segmentación:** Crea una aproximación precisa de una serie temporal por reducción de su dimensión mientras retiene características especiales.



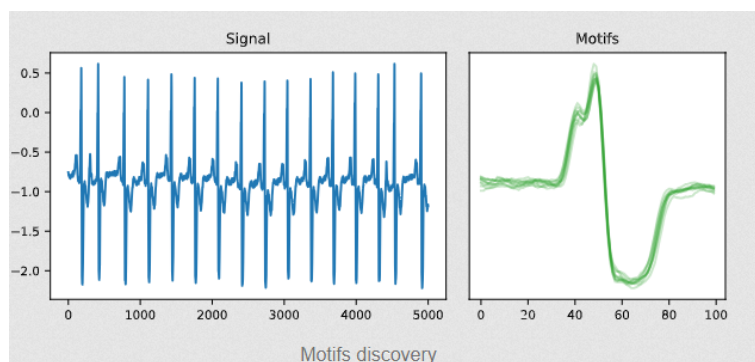
5. **Pronóstico:** Dada una serie de tiempo hasta un tiempo t_n , pronóstico los siguientes valores.



6. **Detección de anomalías:** Encuentra subsecuencias de la serie de tiempo, también llamadas discords.



7. **Motif Discovery:** Encuentra cada subsecuencia (llamada Motif) que parece gobernar la serie temporal recurrentemente.



8. **Rules Discovery:** Encuentra las reglas que podrían gobernar la asociación entre conjuntos de sub-secuencias de series temporales.

Tambien definen los compnentes para la Implementación

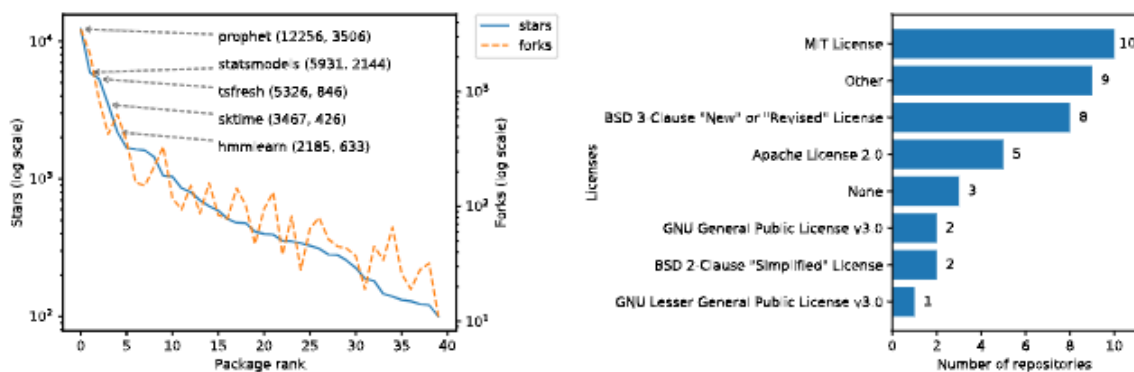
1. **Preprocesamiento:** Filtración de ruido, remover valores atípicos, imputación de valores faltantes.
2. **Representación:** Reducción de dimensionalidad, encontrar formas características fundamentales.
3. **Medidas de semejanza:**
4. **Esquemas de indentación:**

Cometidos implementados Se encontró que las paqueterías explicitamente mencionan los cometidos correspondientes al de la literatura. Se encontró que 6 paquetes prooven de pronosticos, 6 paquetes prooven clasificación, 6 prooven metodos de clustering, 6 prooven detección de anomalías, y 4 prooven métodos de segmentación; 4 que prooven reconocimiento de patrones; y abarcando la indentación con motif Discovery en una categoria, hay cinco paquetes que prooven dichos cometidos.

Luego, para las implementacionesd de componentes, se encontro que 4 paquetes ecplícitamente prooven reducción de dimensionalidad, 17 paquetes prooven imputación de valores faltantes, 24 paquetes prooven métodos de transformación y generación de características. y 7 paquetes prooven vétodos para computar medidas de similaridad. La tabla 4 en el articulo resume cual paquete es capaz de ciertos cometidos.

4.b. Uso de paquetes y comunidades

Para responder a la tercer pregunta de investigación, se extrajo información acercad e la documentación, las dependencias, y la comunidad que apoya dichos paquetes. Por ejmple, GitHub proove de varios repositorios de estadística que puede ser usados para obtener una idea de la esperanza de vida de lso paquetes. Se usó el numero de estrellas de GitHub y los *forks* para estimar la comunidad detras de cada paquete. Las siguiente figura ilustra la cantidad de de estrellas y *forks* para estimar la comunidad.



(a) Distribution of stars and forks for all 40 repositories (log scale). The repositories are ranked by the number of stars, in descending order.

(b) Distribution of licenses (number of repositories per license). None means that no license information was available from GitHub directly.

Casi todos los paquetes depende en numpy. Las paquetes con mayor dependencia son numpy, scipy, pandas, scikit-learn, y matplotlib.

5. Discusión y amenazas para la validación

En esta sección se discuten las elecciones que hicieron los autores que puedan afectar la validación del artículo. Esta revisión se centra en GitHub. Se limitaron a repositorios con al menos 100 estrellas. Esto es algo arbitrario sobre todo para paquetes con cerca de 100 estrellas.

Varios paquetes no pudieron ser encontrados de forma automática. Por ejemplo, el paquete *cesium* no lista en GitHub, sino que fue encontrado tras el *snowballing*. Trataron de automatizar algunos de los cometidos, usando PyPI y la API de GitHub, o la herramienta *johnnydep*. Hubo varios falsos positivos. Por lo que fue necesario checar cada uno manualmente. De igual forma, saber si un paquete se concentra en un dominio específico es algo ambiguo o difuso, entonces decidieron tomar un acercamiento conservativo para mantener la revisión concentrada en series temporales.

La clasificación de los paquetes está en la tabla 4 del artículo, que vemos en la siguiente figura

Package Name	Tasks							Data Preparation					Evaluation			Data		Documentation				
	T1	T2	T3	T4	T5	T6	T7	DP1	DP2	DP3	DP4	DP5	E1	E2	E3	D1	D2	Do1	Do2	Do3	Do4	Do5
arch	+												+			+	+	+	*	+	+	+
atspy	+								+	+	+		+	+	+				+		+	+
banpei				+			+														+	+
cesium											+							+	*	+	+	+
darts	+								+	+	+			+	+	+	+	+		+	+	+
deeptime	+	+						+		+	+			+	+	+		+	+		+	+
deltapy	+	+						+		+	+	+						+	+	+	+	+
dtadistance			+									+			+			+	+	+	+	+
EMD-signal										+					+			+	+	+	+	+
flood-forecast	+								+		+			+	+			+	+	+	+	+
gluonts	+								+		+			+	+	+	+	+		+	+	+
hcrystalball	+								+				+	+	+	+	+		*	+	+	+
hmmlearn	+										+					+		+	*	+	+	+
hypertools			+					+	+		+				+			+	*	+	+	+
linearmodels														+		+		*			+	+
luminaire	+			+			+		+	+	+		+					+			+	
matrixprofile		+	+	+	+	+			+			+			+	+				+	+	+
mcily		+											+		+			+			+	+
neuralprophet	+								+	+	+				+			+	*	+	+	+
nolds									+					+		+	+	+		+	+	+
pmdarima	+									+	+		+	+	+		+		*		+	+
prophet	+						+			+			+	+	+			+	*	+	+	+
pyaf	+								+	+	+		+	+	+		+	+		+	+	+
pycwt										+						+	+	+	+	+	+	+
pydlm	+									+			+	+	+			+	*	+	+	+
pyFIS	+										+		+	+	+	+	+		+	+	+	+
pyodds				+									+	+	+			+	*	+	+	+
pytorchts	+								+		+				+	+	+	+	+	+	+	+
pyts		+			+	+			+	+	+	+		+	+	+	+	+	*	+	+	
PyWavelets										+	+					+	+	+	*	+	+	+
ruptures							+						+	+		+		+	+	+	+	+
scikit-multiflow		+		+			+		+		+		+	+	+	+		+	*	+	+	
seglearn											+					+	+	+	*	+	+	+
sktime	+	+		+	+				+	+	+	+	+	+	+	+	+	+	*	+	+	+
sktime-dl	+	+											+	+				+			+	+
statsmodels	+							+	+	+			+	+	+	+	+	+	*	+	+	+
stumpy					+	+						+						+	*	+	+	+
tftb										+	+		+	+		+		+	+	+	+	+
tsfresh									+		+		+	+		+	+			+	+	+
tslearn		+	+			+					+	+	+			+	+	+			+	+
Total	20	6	6	6	4	4	5	4	17	16	24	7	13	23	25	16	19	34	30	28	40	37
	T1	T2	T3	T4	T5	T6	T7	DP1	DP2	DP3	DP4	DP5	E1	E2	E3	D1	D2	Do1	Do2	Do3	Do4	Do5
	Tasks							Data Preparation					Evaluation			Data		Documentation				

Figura 2: Clasificación de paqueterías. Tasks: T1(Pronóstico),T2 (clasificación), T3 (clustering), T4 (detección de anomalías), T5 (Segmentación), T6 (reconocimiento de patrones), T7(Cambio del putn de detección). Preparación de los datos(componentes de implementación): DP1 (reducción de dimensionalidad), DP2 (imputación de valor ausente), DP3 (descomposición), DP4 (preprocesamiento), DP5 (medidas de similaridad). Evaluaciones: E1 (selección de modelos, búsqueda hiperparamétrica, selección de características), E2 (metricas y pruebas estadísticas), E3 (visualización). Bases de datos: D1 generación sintética de datos, D2 (continen bases de datos). Documentación: Do1 (documentación dedicada), Do2 (notebook ejecutable), Do3 (referecia API), Do4 (guia de instalación), Do5 (guia al usuario).

6. Conclusión

En el artículo se presentó una revisión sistematica de 40 paqueterías de Python dedicadas a series temporales. Los autores propusieron una categorización de los paquetes según el análisis implementado. Tambien discutieron el procedimiento de búsqueda de tales paqueterías, los posibles sesgos y retos encontrados mientras se hacia dicha actividad. Como los paquetes iran evolucionando, los autores mantienen una lista actualizada en su pagina de GitHub en <https://siebert-julien.github.io/time-series-analysis-python/overview.html>.

Referencias

- [1] Siebert, J., Groß, J., and Schroth, C. (2021). A systematic review of python packages for time series analysis. arXiv preprint arXiv:2104.07406.
- [2] Brockwell, P. J., and Davis, R. A. (Eds.). (2002). Introduction to time series and forecasting. New York, NY: Springer New York.