César Isaí García Cornejo
cesar.cornejo@cimat.mx
Series de Tiempo

Abstract

El articulo 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 análizados 40 paqueterías, y las clasificaron de acuerdo a sus labores implementadas, los métodos para el preprocesamiento y licencias. Entre todos los objetivos, se encontro que el más frecuente son los pronosticos. La mitas de las paqueterías prooven acceso a bases de datos reales o simuladas.

1. Introducción

Una serie de tiempo es un conjunto de datos genereados 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 computaciones así como en el medio ambiente. Exite la tenedecia de redicir los costos de medir y guardar dichos datos, lo que ha permitido el incremento del llamado *Big Data* y tecnologias de análisis como *machine learning* y *data mining*. Por tanto, a medida que el número de casos para análisis de series aumenta, tambien aumenta la cantidad de cientificos de datos, ingenieros de datos, análistas, ingenieros de software que se ven en la necesidad de usar librerias para series de tiempo.

La cotejación sistematica propuesta en el articulo se basa en el lenguaje de programación Python, pues es el más usado entre cientificos de datos. La meta del articulo no es evaluar la calidad de las implementaciones por si mismas, sino para proover 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 anomalias, reconocimiento de patrones, o reducción de dimensionalidad o encuentas financieras, tecnologias de la información, industria o medicina.

Sin embargo, la existencia de implentaciones es en vexes enlistada en una manera poco sistematica, como muchas citas dadas por el autor. El articulo se jacta de que no exite tal estudio estructurado y sistematico, por lo que es objetivo de dicho trabajo.

3. Metodología

Se conduce una revisión sistematica de acuerdo a Kitchenham et. al. en su trabajo *A systematic review of systematic review process rearch in software engineering*. Sin embargo, el autore 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

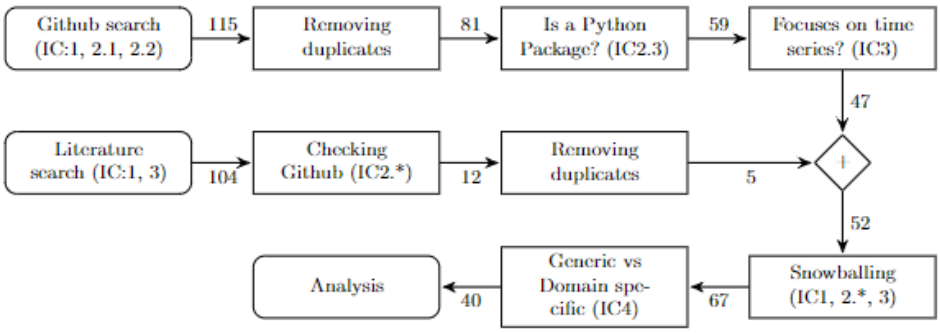


Figura 1: Busqueda y filtrado de procesos, las etiquetas de las flechas indican el número de repositorios discriminados tras cada paso.

3.a. Preguntas de investigación

Se formaliza la meta del articulo. La idea es **análizar** las paquetereias 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 paquetereias ?

3.b. Criterio de inclusión

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

3.c. Buscando repositorios de dominio público en GitHub

El articulo nos menciona que filtraron los repositories 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 reelevantes, 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 cientifica

La busqueda de paqueterias solo en un repositorios podira no ser suficiente para cubrir todas las paqueterías existentes. Como por ejemplo fue el caso de la paquetería tsfresh. Los autores usaron basesd de datos como IEEE Xplore, ACM Digital Library, Web od Science, Scopus y Journal of Opne 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: tsfreash, neurodsp, EoN, nolds, y pastas.

3.e. Snowballing

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

3.f. Genérico vs dominio específico

Como ya se hablo antes, se seleccionaron los paqutes que no se enfucan en una disciplina en partícular que no sea anális de series temporales.

4. Resultados

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

Para contestar la primer pregunta de investigación, primeramente se revizarion 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 temportal una clase predefinida.
4. **Segmentación:** Crea una aproximación precisa de una serie temporal por reducción de su dimensión mientras retiene carácteristicas especiales.
5. **Pronóstico:** Dada una serie de tiempo hasta un tiempo t_n, pronósticar los siguientes valores.
6. **Detección de anomalias:** Encuentra subsecuencias de la serie de tiempo, también llamadas discords.
7. **Motif Discovery:** Encuentra cada suvsecuencia (llamada Motif) que parece governar la serie temporal recurrentemente.
8. **Rules Discovery:** Encuentra las reglas que podrían governar la asociación entre conjuntos de subsecuencias de series temporales.

Tambien definen los compnentes para la Implementación

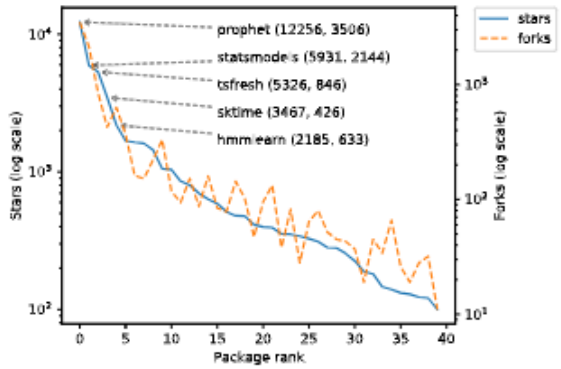
1. **Preprocesamiento:** Filtración de ruido, remover valores atípicos, imputación de valores faltantes.
2. **Representación:** Reducción de dimensionalidad, encontrar formas carácteristicas fundamentales.
3. **Medidas de semejanza:**
4. **Esquemas de identación:**

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 anomalias, y 4 prooven métodos de segmentación; 4 que prooven reconocimiento de patrones; y abarcando la identación con motif Discovery en una categoria, hay cinco paquetes que prooven dichos cometidos.

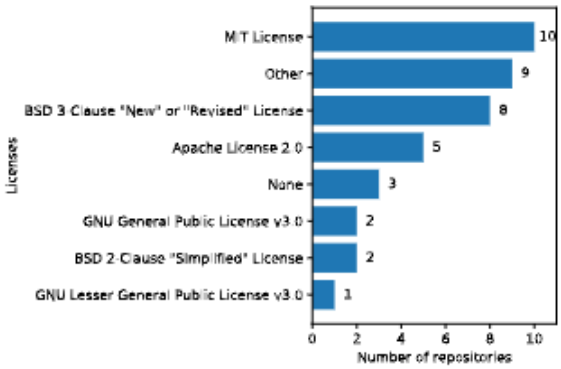
Luego, para las implementacionesd de componentes, se encontro que 4 paquetes ecplicitamente prooven reducción de dimensionalidad, 17 paquetes prooven imputación de valores faltantes,24 paquetes prooven métodos de transformación y generación de carácteristicas. y 7 paquetes prooven vétodos para computar medidades 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 optener 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 paqutes depende en numpy. Las paquetes con mayor dependencia son numpy, scipy, pandas, scikit-learn, y mathplotlib.