Comparación de rendimiento entre Random Forest y Random Forest for Time Series en predicciones del valor del índice S&P 500

Esteban Nicolás Larena

Asignatura Bases de Datos Masivas (11088).

Resumen

Random Forest ha presentado algunas dificultades en los problemas que requieren predecir una serie temporal. La razón es que el algoritmo requiere realizar un muestreo aleatorio (bootstrap) que causa la pérdida de las dependencias temporales.

Una variante que apunta a resolver esa limitación, es el Random Forest for Time Series (RFTS). Plantea varias alternativas de bootstrap que permitirían conservar las dependencias temporales. En este trabajo, se implementaron y probaron las alternativas de RFTS. Para ello, se trabajó sobre el historial del precio del S&P 500. La hipótesis de este trabajo es que se pueden lograr mejores predicciones con RFTS que con Random Forest al intentar predecir el precio del S&P 500.

Los experimentos de este trabajo, inicialmente obtuvieron un mejor rendimiento con RFTS que con RF. Sin embargo, al interpretar detenidamente los primeros resultados y realizar más experimentos, se han llegado a resultados ambiguos. Donde no queda tan claro qué variante funciona mejor para predecir la serie del valor del S&P 500.

Keywords: Random Forest, Random Forest for Time Series, Serie Temporal, S&P 500, Machine learning

1. Introducción

Según Abril[1], una serie temporal es el registro sobre cómo algo cambia en el tiempo. Una de sus características es que siempre tienen una variación aleatoria y esa variación es uno de los objetos de estudio al analizar las series temporales. Se diferencian dos ejes, el análisis descriptivo (búsqueda de patrones) y el modelado. El modelado de series temporales plantea en que las variaciones pueden ser explicadas, lo cual dio lugar a enfoques como las diferentes variantes de modelos autorregresivos (AR[2, 9], MA[2], ARMA[2], ARIMA[2] entre otros), que permiten extrapolar valores futuros.

El aprendizaje automático permite identificar patrones y hacer predicciones. Se puede diferenciar a algoritmos supervisados de los no supervisados. Los supervisados aprenden a partir de datos etiquetados, donde se conocen tanto las variables independientes como la variable dependiente, permitiendo predecir resultados a partir de nuevos datos similares.

El algoritmo de aprendizaje automático Random Forest (RF) permite hacer predicciones a partir de la combinación de múltiples árboles de decisión combinados como un ensamble. La construcción de los árboles se genera de manera aleatoria con un método de boostraping y se sortean tanto las muestras como las variables que serán evaluadas para formar parte de esos árboles aleatorios[3]. El proceso de boostraping también realiza un muestreo aleatorio en el conjunto de entrenamiento, quedándose con un subconjunto. Inicialmente el algoritmo RF no resulta eficiente cuando se trabaja con datos que tienen dependencia en el tiempo, es que, al realizar un muestreo aleatorio, la dependencia se rompe porque se separan los datos[6]. Goehry[6] propuso alternativas al muestreo aleatorio que no rompan esa dependencia y además propuso Random Forest for Time Series (RFTS) como una variante de RF, que implementa esas alternativas. O sea que, la diferencia de RFTS con RF es el método de bootstrap.

1.1. Trabajos relacionados

Doumèche et al.[5] intentaron predecir la demanda eléctrica cuando hay eventos disruptivos que cambian los patrones de consumo. Como sucedió en la crisis eléctrica europea de los años 2022-2023. En este trabajo, se pudo mejorar el rendimiento de varios modelos (incluyendo RFTS) introduciendo datos de movilidad (mobility data).

Doumèche et al.[4] propusieron un benchmark y un algoritmo optimizado (WeaKL) que unifica varias técnicas de restricciones lineales para la predicción de series temporales (forecasting time series). Dentro de las técnicas que utiliza se encuentra el bootstrapping de RFTS.

La relación principal entre ambos trabajos y el presentado aquí, es que se estudia series temporales. Aunque en los trabajos de Doumèche, se trabaja con más algoritmos. También se trabaja con datos sobre gasto eléctrico[5], que podría tener un comportamiento diferente a los de acciones de la bosla. De modo que los resultados pueden ser diferentes. Aquí tampoco se propone un brenchmark, aunque si se deja disponible implementaciones de RFTS. Las implementaciones están sujetas a interpretaciones de las variantes de RFTS[6], y se explica en detalle en el apéndice B.

1.2. Objetivos del trabajo

El objetivo principal de éste trabajo es verificar si los algoritmos de RFTS tienen mejor rendimiento que RF para modelar series temporales del precio del S&P 500 en un rango de fechas entre el comienzo de 2018 y el final de 2019.

Objetivos secundarios:

- Generar un conjunto de datos de S&P 500 a través de datos públicos y con técnicas de web scraping. Además, de la limpieza y transformación de los datos resultantes.
- Implementar RFTS en Python extendiendo la clase RandomForestRegression de Scikit-Learn.
- Buscar parámetros adecuados para una comparación adecuada de RFTS con RF.
- Evaluar el rendimiento del modelado de series temporales con ambos algoritmos.

2. Materiales y métodos

2.1. Datasets

Se generó un dataset construido ad-hoc para este proyecto, a partir de técnicas de web scraping orientadas a la recolección automática de datos del SP500. Luego, se aplicó un proceso de transformación y limpieza de datos para asegurar su calidad y adecuación al análisis.

2.1.1. Web scraping en sitio Yahoo finanzas

Se elaboró un dataset con una técnica de web scraping. Se deja disponible un script[1] en Python que descarga los datos disponibles en el sitio web Yahoo Finanzas[2]. También se adjunta un snapshoot[3]. Se recuperaron los valores de la acción "^GSPC" también llamada SP500.

Finalmente el esquema de datos obtenido contiene las siguientes columnas:

- Date, fecha de la muestra.
- Ticker, símbolo con el cual cotiza una compañía dada.
- Adj Close, cierre ajustado. Similar al cierre sólo que tiene en cuenta los dividendos.
- Close, último precio de una acción al cerrar el día.
- High, precio más alto que alcanza una acción en cualquier operación, dentro del día.
- Low, precio más bajo.
- Open, precio de apertura de una acción.
- Volume, cantidad de operaciones en el día.

[1] https://github.com/NicLarUniversidad/BDM/blob/main/web_scraping/yahoo_scraping.py
[2] https://finance.yahoo.com/
[3] https://www.kaggle.com/datasets/nicolslarena/sp500-2018-2020

2.1.2. Preprocesamiento. Transformaciones de datos

Todos los datasets fueron transformados removiendo las filas que contengan al menos un nulo. Además, sólo se trabajaron con los campos correspondientes a la fecha, el símbolo de la empresa y los precios más altos y bajos. A partir de esos campos se calcularon los siguientes campos:

- DateOrdinal: Puesto que se trabaja con regresión se transformó la fecha a ordinal.
- Precio promedio: Promedio entre los precios (no se trabajó con los dos originales).
- Día de la semana (DayOfWeek).
- Semana del año.
- Día del año.
- Es comienzo de mes (Booleano, 0/1).
- Es final de mes (Booleano, 0/1).
- Cuarto. Correspondiente a que cuarto (quarter) del año corresponde la fecha.
- Número de serie. Fecha máxima del dataset – fecha + 1.
- Diferencia diaria. Precio de ayer - precio de hoy.
- ¿Subió? (Booleano).
- ¿Bajó? (Booleano).
- Cantidad de veces que subió en los últimos 30 días.
- Cantidad de veces que bajó en los últimos 30 días.
- Cantidad de veces que subió en los últimos 7 días.
- Cantidad de veces que bajó en los últimos 7 días.
- Cantidad de veces que subió en los últimos 14 días.
- Cantidad de veces que bajó en los últimos 14 días.
- Cantidad de días seguidos que estuvo al alza.
- Cantidad de días seguidos que estuvo a la baja.
- Máxima cantidad de días subidos que estuvo al alza en los últimos 30 días.
- Máxima cantidad de días subidos que estuvo a la baja en los últimos 30 días.
- Máxima cantidad de días subidos que estuvo a la baja en los últimos 30 días.
- Media móvil (SMA) de los últimos 30 días.
- Índice de disparidad de los últimos 30 días.
- Media móvil exponencial (EMA) de los últimos 30 días.
- Tendencia de los últimos 30 días.
- Volatilidad de los últimos 30 días.

Se tomaron los datos entre 01/01/2018 y 31/12/2020, inclusive. Se guarda el dataset armado listo para usar[4].

Se removieron las filas que contenían nulos, además se tomaron sólamente los datos entre el 1/1/2018 al 31/12/2019. Trabajando con los datos de dos años. Se entrenaron modelos con los datos del 2018 y se probaron con los del 2019.

Inicialmente, se tomaron también los datos del 2020, por motivos de escalabilidad en las pruebas. Sin embargo, luego no se llegaron a utilizar esos datos.

2.2. Algoritmos de bootstrap en RFTS

Goehry[6, 7] propuso 3 alternativas a las que llamó:

- Non-overlaping block bootstrap.
- Moving block bootstrap.
- Circular block bootstrap.

Las alternativas tienen en común que definen un subconjunto de muestras que es continuo en el tiempo llamado bloque. Cada conjunto de entrenamiento se puede formar por varios bloques. La forma de seleccionar la cantidad de muestras y el criterio parece estar sujeto a interpretación. Sin embargo, en RFTS es necesario que las muestras estén ordenadas por tiempo o que al implementar el algoritmo se realice el ordenamiento.

[4] https://github.com/NicLarUniversidad/BDM/blob/main/test/sp500_enriquecido.csv

2.2.1. Non-overlaping block bootstrap

Dado un conjunto de entrenamiento de tamaño N. Toma bloques de tamaño B, siendo B < N. Cada bloque es creado seleccionando aleatoriamente cualquier muestra y tomando las B – 1 siguientes. Se agrega un ejemplo en la figura 1.

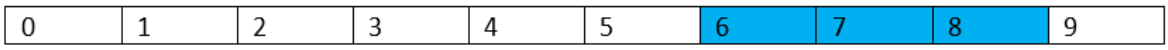


Figura 1: Selección de muestras con el método de Non-overlaping bootstrap. En este ejemplo, se tiene un conjunto de entrenamiento de 10 muestras, se define un tamaño de bloque de 3 y se selecciona a partir de la muestra del índice 6. El bloque sería el conjunto [6 – 8].

En este trabajo, se pudo encontrar dos metodologías al seleccionar el conjunto de entrenamiento al utilizar bloques. Se puede tratar como un conjunto, donde cada muestra es única. O se puede utilizar un sistema de pesos donde cada muestra puede ser entrenada con un peso de 0 para indicar que no va a alterar el modelo. O un peso mayor a 0 que va a impactar en lo que va a impactar en el entrenamiento del modelo. Interpreto que este método intenta simular que una muestra se pueda repetir en un conjunto.

En este trabajo, se utilizó el método del conjunto para Non-overlaping bootstrap y el método de los pesos para los demás algoritmos. La razón es que al usar conjuntos en Python se puede tener un mejor rendimiento en los tiempos de procesamientos. Y con este algoritmo tuve bajo rendimiento utilizando pesos, el cual utiliza listas.

2.2.2. Moving block bootstrap

Dado un conjunto de tamaño N y un tamaño de bloque B. Se definen bloques fijos a partir de la muestra 0. Se muestra un ejemplo en la figura 2.

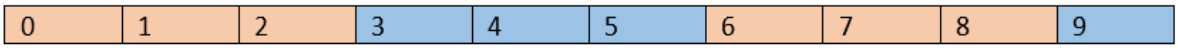


Figura 2: Selección de muestras con el método de Moving Block boostrap. En este ejemplo, se tiene 10 muestras y un tamaño de bloque = 3. Se predefinen los bloques 1:[0, 1, 2], 2:[3, 4, 5], 3:[6, 7, 8], 4:[9]. Si se selecciona aleatoriamente los bloques 2 y 4, se usaría el conjunto [3, 4, 5, 9].

Los bloques tienen un índice entre 0 y (N / B) – 1. Por lo que, la forma de seleccionar el bloque es generando un número aleatorio entre 0 y (N / B) – 1.

2.2.3. Circular block bootstrap

Como las muestras de los extremos tienen menos probabilidad de ser seleccionadas, esta alternativa intenta solventar eso uniendo los extremos. Por lo que los bloques predefinidos seguirían incluso en un índice mayor a (N / B) – 1.

Figura 3: Selección de muestras con el método de Circular Block boostrap. En este ejemplo, se tiene 10 muestras y un tamaño de bloque = 3. Se predefinen los bloques 1:[0, 1, 2], 2:[3, 4, 5], 3:[6, 7, 8], 4:[9, 0, 1], 5:[2, 3, 4]... Notar que ahora se puede elegir cualquier número de bloque, mayor a 0. Si se selecciona aleatoriamente los bloques 2 y 4, se usaría el conjunto [0, 1, 3, 4, 5, 9].

3. Métricas

La raíz del error cuadrático medio (RMSE), es una medida de precisión con una escala que depende de la escala de los datos que mide. O sea, que si mide valores grandes va a tender a ser más grande. Tiene utilidad y es ampliamente utilizada en la comparación de diferentes métodos aplicados sobre el mismo conjunto de datos. Sin embargo, no debería ser utilizada sobre datos que tengan diferentes escalas entre ellos[8]. Para la comparación del

rendimiento de los diferentes modelos de RF y RFTS, es adecuado utilizar RMSE. Siempre y cuando se utilicen los mismos datos.

4. Configuración del experimento

Se realizaron 3 experimentos con los que se intenta determinar en primer lugar cuáles son los mejores parámetros para configurar RFTS. Esto además incluye un rango de pruebas apropiado replicando -dentro del alcance de este trabajo- lo realizado por Goehry[7, 6], incluso realizando gráficos similares. Ya que en ese trabajo se obtuvieron resultados favorables para RFTS frente a RF.

Durante el segundo experimento, se configuraron los modelos con los parámetros buscados en el primero y se compararon las predicciones con los valores reales. De esta forma, se buscó encontrar patrones y entender mejor los resultados del primer experimento.

Mientras que en el tercer experimento, se introducen las ventanas deslizantes. Las cuales se utilizan para volver a entrenar los modelos cada determinado período de tiempo (en la serie). De esta forma, se pueden hacer predicciones con datos actualizados. Pero sin entrenar el modelo constantemente, lo cual sería costoso.

Se procesaron los datos con el lenguaje Python y notebooks júpiter[5], las librerías Pandas[6], Numpy[7]. Mientras que se elaboraron los gráficos con las librerías Matplotlib[8] y Seaborn[9].

5. Resultados del experimento

5.1. Primera prueba. Calibración del tamaño adecuado de bloques

La primera prueba consistió en la búsqueda del tamaño apropiado de los bloques. Para ello, se optó por la estrategia de ajustar 500 modelos RF y en el caso de RFTS, se ajustaron 500 por cada tamaño de bloque y por algoritmo. También se probaron con conjuntos de pruebas diferentes.

Tabla 1: Cantidad de modelos que se van a ajustar en total. Donde 500 son las corridas de cada una de las variantes algorítmicas y para los 11 tamaños de bloques

Algoritmo	Primer experimento	Total
RF	500	500
RFTS - NO	11 x 500	5500
RFTS - MB	11 x 500	5500
RFTS - CB	11 x 500	5500
Cantidad de modelos		17000
Cantidad de mediciones (3 conjuntos de pruebas)		51000

Se probaron los tamaños de bloques 5, 6, 7, 8, 9, 10, 20, 30, 50, 100 y 200 y con conjuntos de pruebas que se conforman con los datos de 90, 180 y 365 días posteriores al más nuevo del conjunto de prueba. Además, se usaron todos los datos del 2018 para calibrar los modelos. La tabla 1 ilustra la cantidad de modelos que se entrenaron y las mediciones que se tomaron. Fueron 17000 modelos (500 x 11 x 3 + 500). Midiendo su desempeño 3 veces a cada uno (una vez por cada uno de los 3 conjuntos de pruebas). Por lo que se obtuvo una muestra de 51000 mediciones, con la métrica RMSE.

Se las ordenó con gráficos de cajas. Separándolas por algoritmo, por tamaño de bloque y conjunto de pruebas. Las mediciones de RF se dejaron a la izquierda, ya que son los valores de referencia.

[5] https://jupyter.org/
[6] https://pandas.pydata.org/
[7] https://numpy.org/
[8] https://matplotlib.org/
[9] https://seaborn.pydata.org/

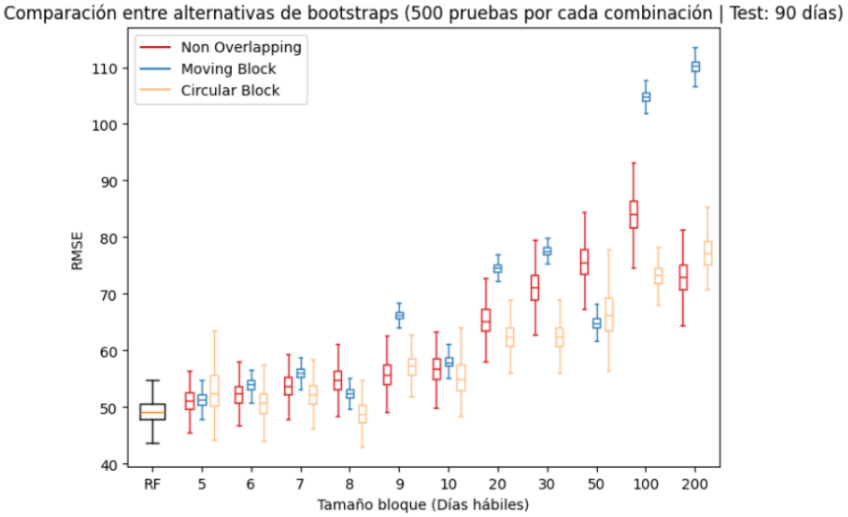


Gráfico 1: Cajas que agrupan el RMSE de la primera prueba, separadas por tamaño de bloque, algoritmo. Con el conjunto de pruebas de 90 días.

Los resultados mostrados en el gráfico 1 corresponden al medir el RMSE contra los siguientes 90 días al último día del conjunto de entrenamiento. Sólo hay una caja que parece estar más abajo que la de RF. Que es la caja de RFTS Circular Block con un tamaño de bloque 8.

Los demás resultados de RTFS no parecieron tener mejores resultados que RF al predecir 90 días hacia adelante. Aunque la variante Moving Block, mostró una menor varianza en sus resultados con respecto RF. Lo cual puede significar que a corto plazo, RF tiene mejores resultados que RFTS.

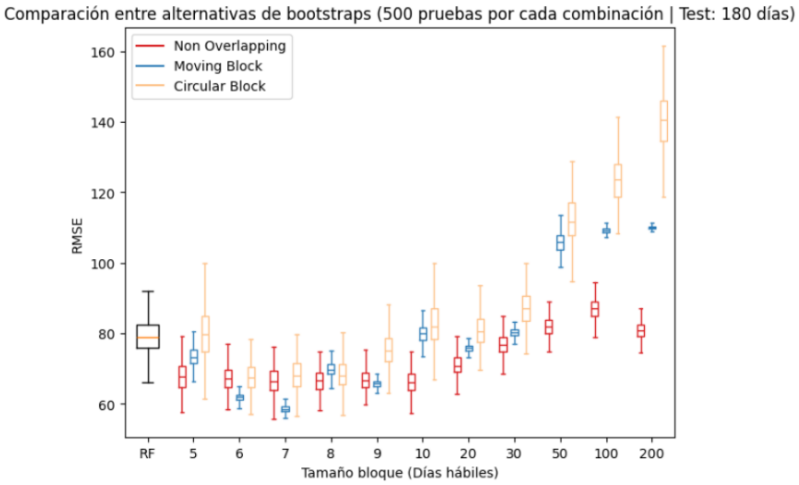


Gráfico 2: Cajas que agrupan el RMSE de la primera prueba, separadas por tamaño de bloque, algoritmo. Con el conjunto de pruebas de 180 días.

Al predecir los 180 días siguientes al valor más nuevo del conjunto de entrenamiento (Gráfico 2), se puede notar fácilmente una mejora de las variantes de RFTS con respecto a RF. Las medianas de las cajas, parecen acercarse a una hipérbola en la que en un primer tramo se nota una mejora constante hasta un mínimo de RMSE. Luego, hay un segundo en el que, a medida que aumenta el tamaño de bloque también aumenta el RMSE. Aunque el incremento del RMSE es de una forma un poco más atenuada al llegar a los tamaños de bloques más altos.

La variante Moving Block, sigue mostrando una menor varianza que RF. Mientras que Non Overlapping muestra una varianza similar a RF y Circular Block muestra una varianza mayor a RF. Se obtuvieron resultados notablemente mejores o parecidos a RF hasta una tamaño de bloque de 30, a partir del siguiente tamaño (50) se comenzaron a tener peores resultados. Sobre todo con Moving Bock y Circular block. Aunque Non Overlapping con un tamaño de bloque 200, tuvo una leve mejora frente a RF.

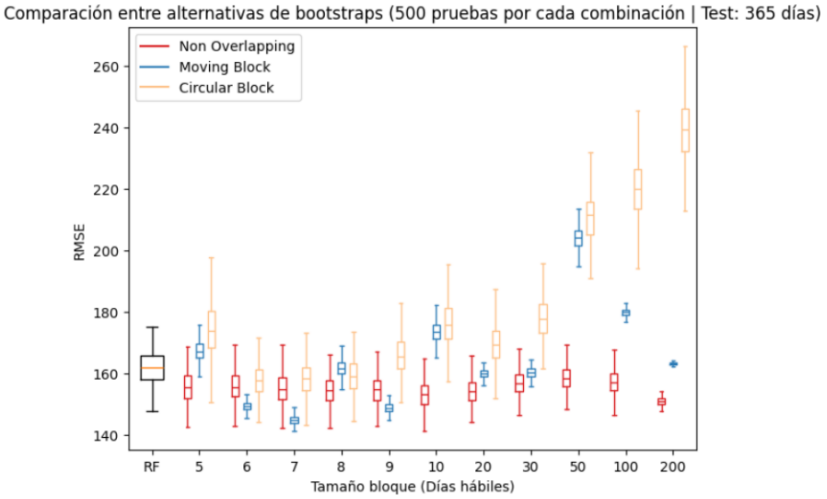


Gráfico 3: Cajas que agrupan el RMSE de la primera prueba, separadas por tamaño de bloque, algoritmo. Con el conjunto de pruebas de 365 días.

Comparando con las predicciones de 365 días luego (Gráfico 3), se puede ver una mejora con todos los tamaños de bloque de la variante Non Overlapping. Moving block sigue destacándose por tener una variación menor a las demás. Sin embargo, se repite que al llegar a los tamaños de bloques más grandes Non Overlapping sigue manteniendo resultados parecidos a los tamaños anteriores. Mientras que las demás variantes sufren un deterioro notable en su rendimiento. Lo cual podría significar que Non Overlapping necesita menor análisis al elegir su tamaño de bloque.

A rasgos generales, Circular Block no tuvo buenos resultados en esta prueba. Mientras que Moving Block se caracterizó por tener una menor variación que los demás algoritmos. Y Non Overlapping mostró una mayor tolerancia de tamaños de bloques. Por lo que en casos similares al que se estudia en este trabajo, se podría plantear utilizar Non Overlapping o Moving Block según se requiera realizar pruebas con varios tamaños de bloques o si se desea tener una menor varianza en los resultados.

De acuerdo a los mejores resultados, se decidió continuar con el tamaño de bloque 7 para Moving Block y 6 para Circular Block. Mientras con Non Overlapping se utilizó tamaño 10. También se define un conjunto de pruebas de 180 días. Ya que presenta un mejor desempeño general de las variantes de RTFS en comparación de RF. No son el grupo con menor error. Sin embargo, de los que menos error tienen, son los que tienen menor varianza en las 3 alternativas. Se puede observar en el gráfico 2 que la mejora de algunas configuraciones, tienen una mayor proporción que en el gráfico 3. Por lo que en los próximos experimentos se utilizó el conjunto de entrenamiento de 180 días (gráfico 2).

Tabla 2: Valores de RMSE promedio y desvíación estándar según los diferentes tamaños de bloques con relación a RF tradicional.

Tamaño de bloque	RFTS Non-Overlapping		RFTS Moving-Block		RFTS Circular-Block		Random Forest	
	Media	Variación	Media	Variación	Media	Variación	Media	Variación
RF	-	-	-	-	-	-	79.33	±17.14
5	67.79	±24.29	73.42	±15.24	80.22	±48.32	-	-
6	67.48	±26.83	61.88	±8.14	67.76	±26.17	-	-
7	66.71	±22.35	58.62	±6.52	68.26	±29.96	-	-
8	66.55	±22.95	69.91	±12.9	68.31	±29.24	-	-
9	66.88	±24.33	65.76	±6.01	75.44	±31.10	-	-
10	66.25	±20.17	79.99	±16.71	82.62	±47.24	-	-
20	71.10	±19.40	75.80	±6.12	80.97	±26.32	-	-
30	76.79	±17.92	80.24	±7.58	87.15	±29.24	-	-
50	81.79	±18.35	105.82	±17.25	112.14	±48.62	-	-
100	86.92	±17.46	109.29	±5.47	123.75	±40.14	-	-
200	80.88	±14.72	110.00	±3.19	140.58	±47.35	-	-

5.2. Segunda prueba. Comparación de predicciones contra resultados esperados.

La segunda prueba consistió simplemente en entrenar un modelo de cada variante y graficar sus resultados contra los valores reales o esperados. Como se dijo anteriormente, se va a utilizar un tamaño de bloque 6 para Moving Block y Circular Block. Mientras con Non Overlapping se utilizó tamaño 10.

Mientras que las pruebas se realizan con un conjunto de test de 180 días.

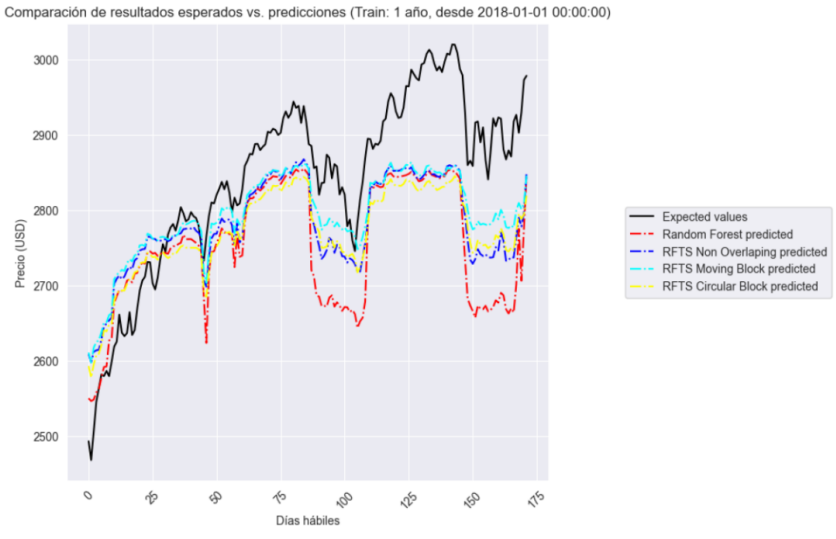


Gráfico 4: Valores esperados contra predicciones de los modelos de RF y las 3 variantes de RFTS. Utilizando tamaños de bloques para Non-Overlapping = 10, Moving Block= 7 y Circular Block= 6 y un conjunto de pruebas de 180 días.

Al observar el gráfico 4, se puede ver que los modelos intentan seguir los valores esperados y que comienzan a perder el ritmo alrededor del día 50. La diferencia más notable entre los algoritmos es que RF parece que tiene más variación en sus resultados, haciendo que su línea se acerque y aleje más de los resultados reales.

Se puede observar que en algunos intervalos, Random Forest se ajusta mejor a los valores reales. Sin embargo, hay que considerar que con estos mismos tamaños de bloque y mismo conjunto de prueba, RFTS obtuvo mejor RMSE al promediar las 500 pruebas de cada variante que utilizaron esa configuración. Por que en teoría, RFTS tiene mejor desempeño en el gráfico 4. Por lo que probablemente, eso se haya dado por los dos picos hacia abajo de los días 80 a 105 y de los días 140 a 165. Los cuales han sumado una buena cantidad al RMSE, sin importar cuán bien se haya ajustado el modelo. Además, en estos intervalos, los modelos ya habían perdido su ritmo. Por lo que considerar ese error quizás no sea tan relevante. Sin embargo, en el gráfico 1 (prueba con 90 días) no se obtuvieron buenos resultados.

5.3. Tercera prueba. Utilización de ventanas deslizantes.

Durante una tercera prueba, se volvieron a entrenar modelos utilizando ventanas deslizantes. Como se muestra en el gráfico 5, se utilizó un tamaño de ventana de 30 días hábiles. Con un rango de pruebas de 180 días.

De forma que se comenzó con un modelo de cada uno de RF y las variantes RTFS. Se calculó el RMSE en los próximos 30 días. Luego, se volvieron a entrenar otros modelos, ahora usando un año atrás a partir del último día usado en las predicciones. De forma que los primeros modelos utilizaron los datos de los días 1 al 365, los modelos de la segunda ventana utilizarán los datos de los días 31 al 395. Se agregaron líneas punteadas oscuras para separar cada ventana.

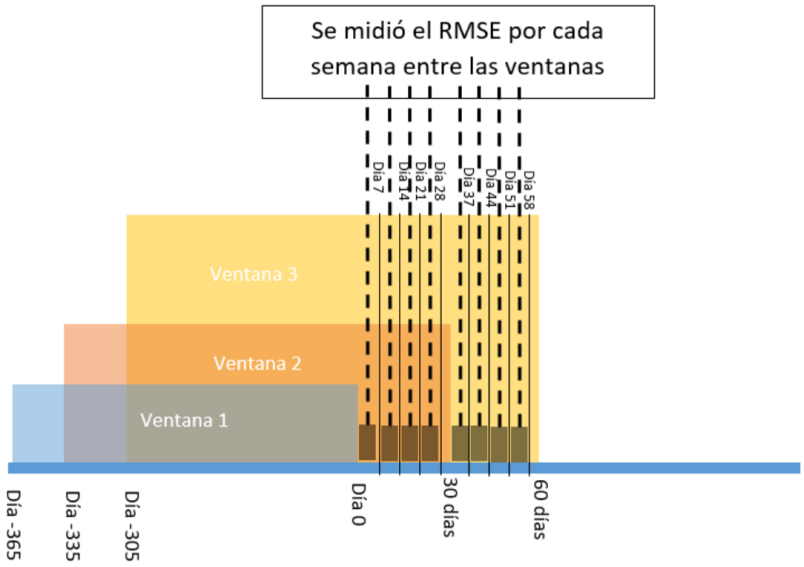


Gráfico 5: Funcionamiento del uso de las ventanas deslizantes. Al desplazarse una ventana 30 días hacia adelante, se mueve el conjunto de entrenamiento y el de pruebas 30 días cada uno. Mientras que el RMSE es medido por semana. De forma que en cada ventana se toman 4 mediciones.

Pareció oportuno agregar los valores predichos contra los reales. Sin embargo, el cálculo de RMSE no se realizó día trás día, sino que se tomaron grupos de una semana. Por ejemplo, para la primera ventana se tomaron el RMSE cada 7 días. Por eso se agregaron líneas punteadas más claras. Que representan cuándo se tomaron las medidas del RMSE. Por otro lado, los valores esperados contra las predicciones, son valores diarios. También se calculó el porcentaje de error de las predicciones. (RMSE / Valor esperado).

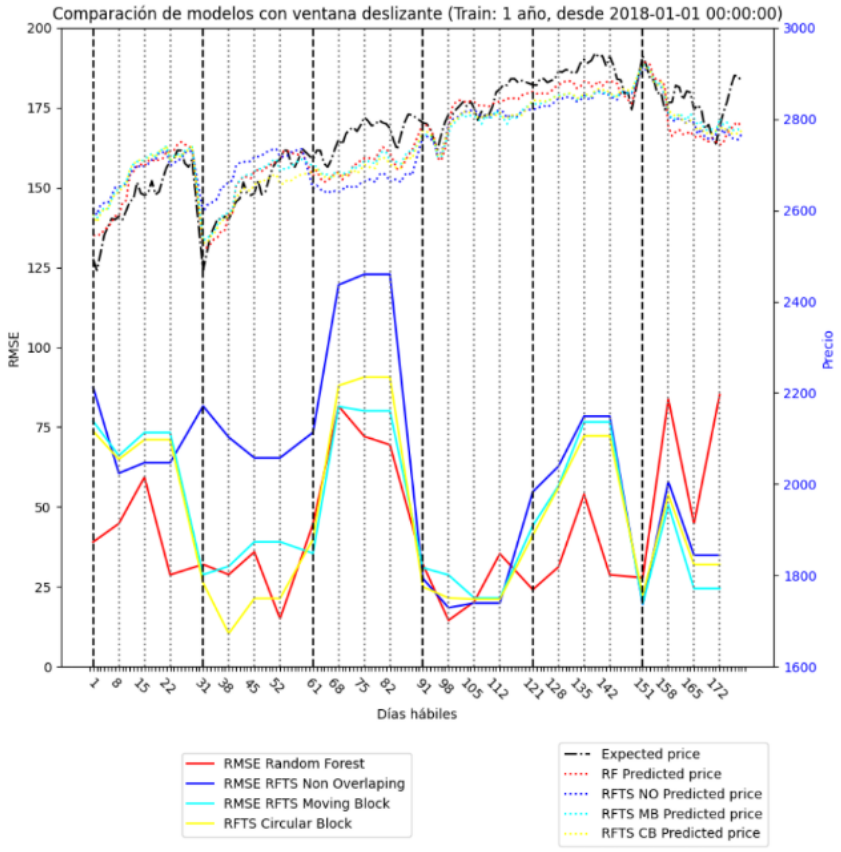


Gráfico 6: RMSE con ventanas deslizantes. Se grafica el error de los modelos de RF y las variantes de RFTS. Las ventanas deslizantes son de 30 días. También se presentan los valores esperados contra las predicciones. Las líneas punteadas oscuras encierran cada ventana. Se continúa utilizando tamaños de bloques para Non-Overlapping = 10, Moving Block= 7 y Circular Block= 6 y un conjunto de pruebas de 180 días.

Se utilizaron ventanas de 30 días, el gráfico 6 muestra dos factores. Con las líneas sólidas se muestra el RMSE que tuvieron los modelos. Para calcular el RMSE se tomaron las predicciones de una semana entera, y se midió el error semanal. Por eso hay líneas punteadas, las cuales coinciden con las mediciones del RMSE. Mientras que las líneas punteadas representan el precio predecido y los precios reales.

La primera ventana tiene una particularidad. Que es que hay una caída del precio justo en el día que se volvió a entrenar el modelo. La variante de Non Overlapping que era una de las que había tenido mejor rendimiento en el gráfico 1, no pudo ajustar lo suficiente y en la ventana dos y tres fue la que obtuvo mayor RMSE. Por lo que podemos observar que al introducir ventanas deslizantes, se obtendría un resultado diferente en cuanto resultados.

Si bien hay pequeños alejamientos de las predicciones y los valores esperados, al introducir ventanas deslizantes se obtiene un mejor ajuste. Aunque ya no hay una notable ventaja de RFTS sobre RF, de hecho parece que RF tiene mejores resultados. Considerando que en dos ventanas (1-31 y 121-151), RF fue el que logró mejor rendimiento. Y en tres ventanas (31-61, 61-91 y 91-121), obtuvo un rendimiento mejor que algunas variables. Tampoco hay una variable que se destaque. Por eso, se podría decir que RF en realidad tuvo mejor rendimiento en la mayoría de casos, salvo la última ventana.

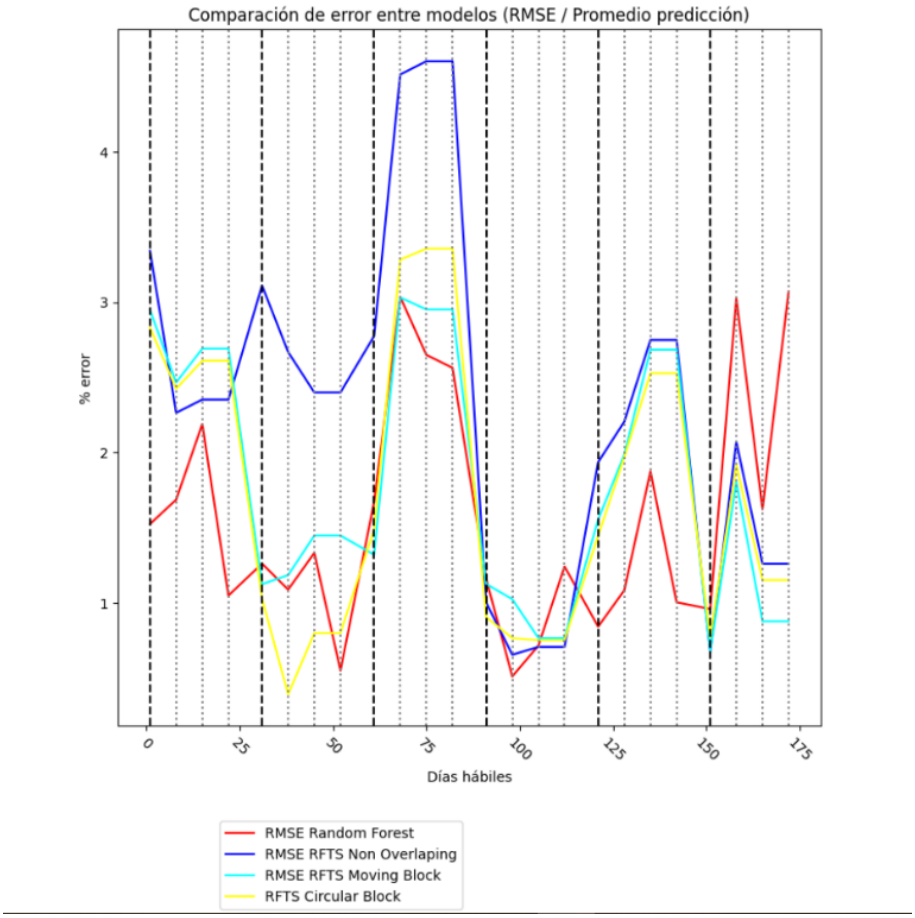


Gráfico 7: Proporción de error de los modelos de la tercera prueba. Ventanas deslizantes de 30 días y se tomaron el RMSE por semana. Se grafica la operación RMSE / Promedio del valor esperado. Se continúa utilizando tamaños de bloques para Non-Overlapping = 10, Moving Block= 7 y Circular Block= 6 y un conjunto de pruebas de 180 días.

Mientras que en el gráfico 7, se muestra el porcentaje de error. Calculado por el RMSE dividido el promedio de los valores esperados. Se obtiene una forma muy similar al RMSE del gráfico 6.

Se puede perder de vista la ventaja que parecían mostrar los algoritmos RFTS. En la segunda prueba, el mejor desempeño de RFTS frente a RF dependía de que sus predicciones se mantuvieran "más abajo" que las de RF cuando los modelos requerían de re-entrenamiento. Pero como se vió en el gráfico 1, al hacer pruebas con un conjunto de entrenamiento más acotado y actual, RF tiene mejores resultados.

Es importante considerar que los valores del conjunto de datos presentan una tendencia positiva, es decir, son crecientes. Esto genera un sesgo en las predicciones: cuando un modelo pierde el ritmo de la serie, tiende a estimar valores inferiores a los reales. Por esta razón, el modelo que predice valores "más altos" puede mostrar un mejor rendimiento aparente, aunque no necesariamente refleje una mayor precisión en la captura de la tendencia real.

Por eso, al utilizar las ventanas deslizantes, se reduce ese fenómeno que se ve en el gráfico 5. Que a mi parecer no significa que RFTS funcione mejor, sino que hay casos que simplemente se equivoca por menos al perder el ritmo.

Aquí, se plantea la hipótesis de que, en esta serie, los algoritmos de RF estudiados, quedan desactualizados porque la serie tiene una tendencia positiva. A partir de cierto punto, las predicciones quedarán siempre por debajo de los valores reales. Por eso, si se mide con un conjunto de pruebas muy amplio en el tiempo, no es fiable calcular el RMSE de todo el conjunto de pruebas (como se hizo en los gráficos 1, 2 y 3). De esa forma, sólo se podrá ver cuál modelo se aleja menos se los valores reales.

Las predicciones tienen utilidad, porque predicen los momentos de algunas caidas y subidas. Pero todos los modelos fallan a la hora de hacer predicciones precisas. Al fallar todos, no se puede decir que uno sea mejor que otro. Además, al utilizar ventanas deslizantes, RFTS pierde su ventaja contra RF. De hecho, como se analizó, RF tiene mejor rendimiento que los demás, en dos ventanas. Mientras que en tres ventanas tiene un rendimiento reñido con los demás. Y sólo se logra superar a RF en la última ventana.

6. Conclusiones

Se ha replicado el paper de Geohry[6], usando programación hecha en este trabajo (Apéndice B.1) y con un dataset de los valores del S&P 500. Inicialmente se obtuvieron resultados de acuerdo a lo esperado. De forma que RFTS, tuvo una mejora sobre RF de acuerdo al experimento de calibración.

Sin embargo, al realizar un análisis más detenido, se ha encontrado que el enfoque con el que se discriminaron los modelos, en el experimento de calibración, quizás no fue el más adecuado en el trabajo actual. Ya que una cantidad importante del RMSE podría ser causado en intervalos donde se generaron valores que podrían ser considerados extremos (o sea, los más lejanos a los valores reales) y en momentos en que los modelos ya no explican el comportamiento de la serie. Luego, al introducir el uso de ventana deslizante, la ventaja inicial de RFTS se perdió de vista. Dejando menos claro cuál variante funciona mejor, ya que los resultados se ven ceñidos. De forma que se llegaron a dos conclusiones importantes:

1. La mejora del rendimiento de las variantes de RFTS varían según el momento en el que se entrene el modelo. De forma, que al volver a entrenar los modelos, RFTS podría dejar de funcionar mejor que RF.

2. Ambas variantes de RF pierden el "ritmo" en poco tiempo. Causando que no sea útil buscar configuraciones con conjuntos de entrenamiento grandes.

Quizás estos factores tengan que ver con que la serie a predecir, es constantemente creciente. Como posibles mejoras y/o trabajos futuros se podría considerar eliminar la tendencia positiva. También, se podrían combinar el primer y tercer experimento. Buscando valores óptimos al usar ventanas deslizantes. Aunque quizás, lo más interesante sería replicar el experimento con otros datasets para ver si se obtiene el mismo comportamiento errático y tratar de explicar el motivo.

Referencias

[1] Juan Carlos Abril. "Análisis de la evolución de las técnicas de series de tiempo: Un enfoque unificado". spa. En: (dic. de 2011). Accepted: 2019-03-11T22:01:38Z Publisher: Instituto Interamericano de Estadística. ISSN: 0014-1135. URL: https://ri.conicet.gov.ar/handle/11336/71449 (visitado 28-08-2025).

[2] George Box y GM Jenkins. "Analysis: Forecasting and Control". En: *San francisco* (1976).

[3] Leo Breiman. "Random Forests". En: *Mach. Learn.* 45.1 (oct. de 2001), págs. 5-32. ISSN: 0885-6125. DOI: 10.1023/A:1010933404324. URL: https://doi.org/10.1023/A:1010933404324.

[4] Nathan Doumèche y col. *Forecasting time series with constraints*. arXiv:2502.10485 [stat]. Feb. de 2025. DOI: 10.48550/arXiv.2502.10485. URL: http://arxiv.org/abs/2502.10485 (visitado 08-08-2025).

[5] Nathan Doumèche y col. *Human spatial dynamics for electricity demand forecasting: the case of France during the 2022 energy crisis*. arXiv:2309.16238 [stat]. Sep. de 2023. DOI: 10.48550/arXiv.2309.16238. URL: http://arxiv.org/abs/2309.16238 (visitado 08-08-2025).

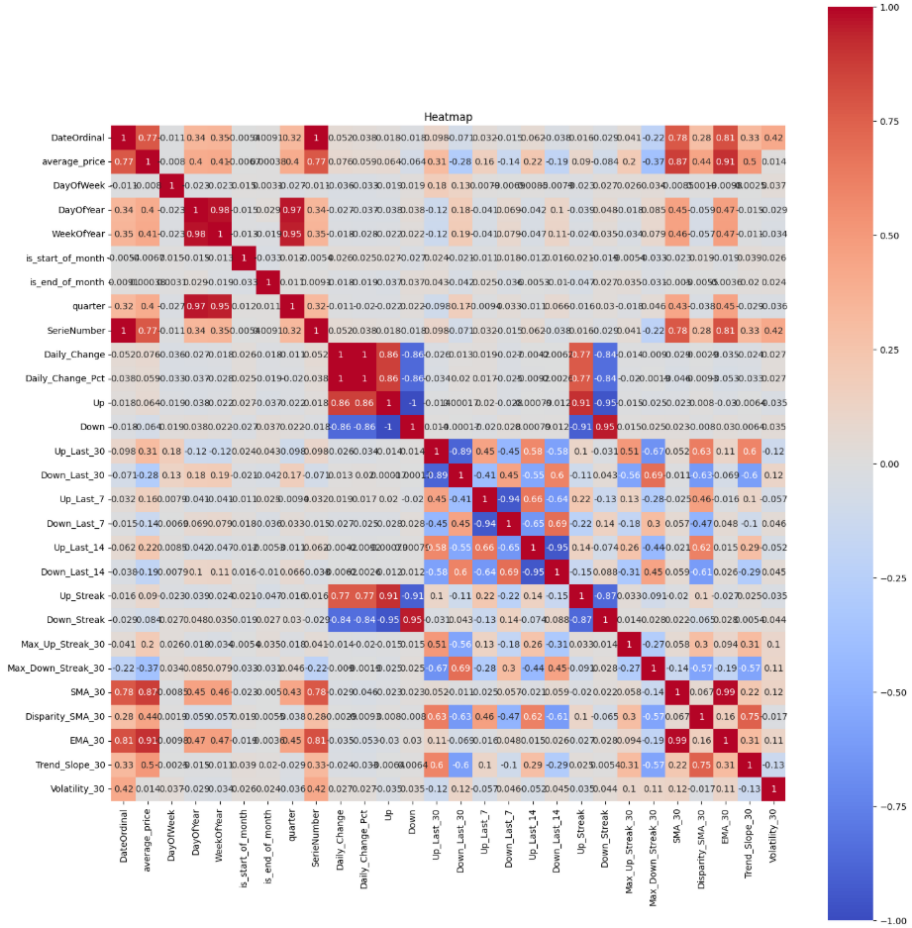
[6] Benjamin Goehry. "Random forests for time-dependent processes". En: *ESAIM: Probability and Statistics* 24 (abr. de 2020). DOI: 10.1051/ps/2020015.

[7] Benjamin Goehry y col. "Random forests for time series". En: *HAL Open Science* (2021). URL: https://hal.science/hal-03129751v1/file/Block_bootstrap_for_random_forests.pdf.

[8] Rob J Hyndman y Anne B Koehler. "Another look at measures of forecast accuracy". En: *International journal of forecasting* 22.4 (2006), págs. 679-688.

[9] GU YULE. "On a Method of Investigating Periodicities in Disturbed Series, with Special Reference to Wolfer's Sunspot Numbers (1927)". En: *The Foundations of Econometric Analysis* (1995), pág. 159.

A. Correlaciones



B. Material complementario

B.1. Implementaciones de RFTS que aporta este trabajo

Se aportan 3 implementaciones de RFTS[10]. Todas extienden la clase RandomForestRegressor de Sklearn. Y se alteraron sólo las líneas donde se realiza el sub-sampling.

Non-overlaping. Se utilizan conjuntos para seleccionar las sub-muestras. Por lo que, a diferencia de las otras dos implementaciones, cada muestra se puede utilizar una sola vez en cada sub-árbol. Inspirado en la forma de muestreo de Sklearn, partimos de un conjunto de entrenamiento de tamaño N. Agregamos un parámetro para el tamaño de bloque que aquí llamaremos B. Generamos N / B números aleatorios entre 0 y N. Dado un número aleatorio A generamos un conjunto con todos los valores entre A y (A + B), [A – A+B). Juntamos los conjuntos en uno solo, y ese conjunto es usado como peso de acuerdo a la implementación orginal de RF de Sklearn.

Moving block. Se generan N / B números aleatorios entre 0 y (B / N) y se generan listas con los números de cada bloque correspondiente a los números aleatorios. Se juntan las listas y se utiliza el sistema de pesos que viene por defecto en la implementación de Sklearn. Con mi implementación no es posible utilizar las muestras del final del conjunto de entrenamiento si el último bloque es de un tamaño menor a B.

Circular block. Se generan N / B números aleatorios entre 0 y ((B / N) * 2). Cuando se intenta elegir un bloque predefinido a cada número aleatorio se utiliza un offset. Cuando el bloque se sale del límite del conjunto, se aplica la operación de módulo contra N. Formando el círculo.

B.2. Implementaciones de terceros

B.2.1. Sklearn

La forma de selección de muestras de la implementación de RF de Sklearn[11] es la siguiente. Dado un conjunto de entrenamiento de tamaño N, se generan N números aleatorios entre 0 y N – 1. Los números aleatorios se usan como pesos de cada muestra, asociándolas con un índice que va de 0 a N – 1. Cada árbol es entrenado siguiendo ese método en el boostrap.

B.2.2. PYTS

Johann Faouzi publicó en Github una implementación de RFTS que utiliza el Non-overlaping block bootstrap[12] La forma en que selecciona las muestras es generando dos números aleatorios entre 0 y N – 1, y todas las muestras en ese rango se usan para entrenar un árbol dado. La principal diferencia con las implementaciones de este trabajo es que implementa un modelo de clasificación y yo voy a implementar un modelo de regresión.

[10] https://github.com/NicLarUniversidad/BDM
[11] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
[12] https://github.com/johannfaouzi/pyts/blob/main/pyts/classification/time_series_forest.py