

TP FINAL MLF 2024 Ignacio Cortés

El trabajo está inspirado en resolver cierto tipo de problemas con los que me enfrento en las series de tiempo de variables macroeconómicas. Para resumir las complicaciones que presentan este tipo diré que

- 1) Los datos son contruídos de forma agregada en base a relevamientos microeconómicos, por lo que presentan ruido, cuando no cierta inconsistencia entre períodos
- 2) Las series de datos son breves, típicamente 1 valor mensual, por lo que en 15 años se cuenta efectivamente con unos 180 datos
- 3) Existe correlación entre variables y más aún, cointegración, lo que indica que estas correlaciones pueden estar sostenidas a largo plazo
- 4) Las variables pueden ser autorregresivas, esto quiere decir que valores pasados de una misma variable pueden influir a tiempo futuro en la misma variable. Si además se considera ruido, también tendrá una media móvil en función de la propagación
- 5) Normalmente, el movimiento (impulso) de una variable produce un impacto (respuesta) en otra variable, transcurrida una cierta cantidad de tiempo (lags o rezagos)
- 6) Son sensibles a movimientos exógenos, como eventos naturales o políticos que exceden las consideraciones del modelo

La gran importancia de las series de tiempo y las complicaciones que traen aparejadas me impulsaron a explorar las herramientas que vimos en el curso con objetivo de aplicarlas.

- 1) En primer lugar utilizaré datos sintéticos, contruídos con cierto criterio y con las complicaciones anteriormente mencionadas para sintonizar un modelo adecuado que pueda generar predicciones sobre una variable objetivo o target.
- 2) Posteriormente probaré el modelo para datos en variaciones porcentuales en lugar de niveles agregados (el mismo set de datos que en 1 pero en lugar de nivel de precios, variación porcentual entre meses).
- 3) Luego trabajaré en una serie temporal con un proceso autorregresivo, media móvil (ARMA) y probaré el método para forecast, es decir, para estimación de valores futuros.
- 4) Finalmente, reproduciré algo similar al primer punto, pero para datos reales de la inflación y distintos predictores que consideraré.

En la medida que avanzo en el documento, mencionaré resaltado en verde qué script es el que adjunto para reproducir los resultados que informo. Los archivos que iré mencionando son:

- **feature_importance.ipynb**
- **modelo ecuación cuantitativa.ipynb**
- **autorregresivo.ipynb**
- **porcentuales.ipynb**
- **datos_log.csv**
- **passthrough.ipynb**

Primer set de datos sintéticos: LA ECUACIÓN CUANTITATIVA DEL DINERO

Para el primer dataset imaginario, me inspiré en la “Ecuación cuantitativa del dinero”. Una ecuación económica teórica que no necesariamente se cumple empíricamente, pero es reconocida en general sobre todo por las escuelas ortodoxas de economía. También hay cierto consenso acerca de que en ciertos límites es de esperar que se cumpla la relación:

$$P \cdot Q = M \cdot V$$

$$(\text{Precios } P) * (\text{Producción } Q) = (\text{Masa de dinero } M) * (\text{Velocidad de circulación del dinero } V)$$

Básicamente, ésta relación indica que el **nivel de precios P** en una economía cerrada (que produce la inflación), depende de la masa de dinero circulante (que se modula con la emisión monetaria) multiplicada por la velocidad de circulación de ese dinero y dividida por el nivel de producción.

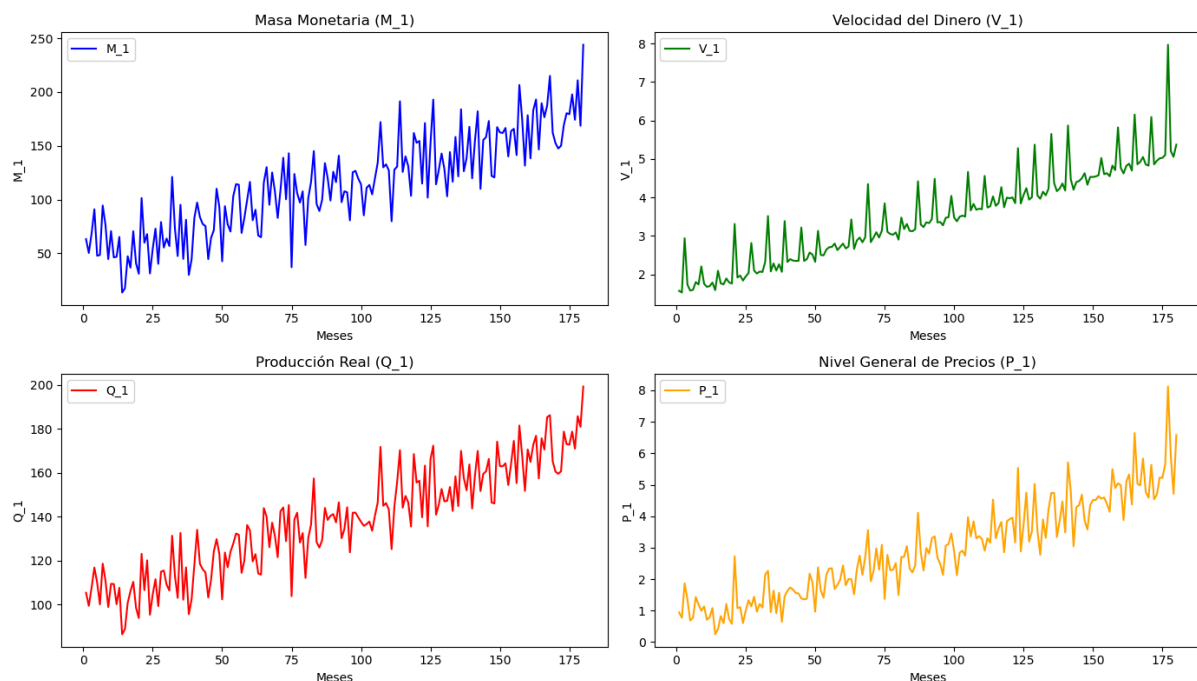


Gráfico de los datos sintéticos

Aquí, P_1 ha sido creado mediante la relación de la ecuación cuantitativa.

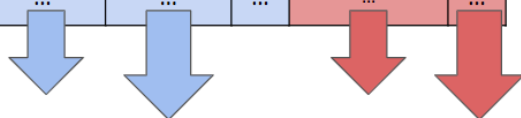
$$P = MV/Q$$

Ahora bien, el impacto que puede tener la expansión de la base monetaria (emisión o aumento de la variable M) sobre el nivel de precios P. O el aumento de la velocidad de circulación V. O incluso del nivel de producción. No necesariamente impactan en el nivel de precios en tiempo presente. Es esperable que exista un lag temporal entre el cambio en alguna de éstas variables y su respuesta en la medición (a T+0) de P. Simplemente porque en la economía los procesos tienen rezagos y los cambios no se propagan de forma instantánea.

Para simular este hecho, he construido con estos datos un dataframe que posee la variable P tal com la he calculado hasta ahora (que será la variable target), pero he introducido desfases negativos en las demás variables (V,M y Q) (que serán las variables feature).

A priori en se desconoce cuál es el desfase real entre un impulso y su respuesta, por lo que en esta primera parte, con estos datos construidos con ruido, me he propuesto “descubrir” cuáles son los lags que producen la función de respuesta esperada. Para ello, he construido un dataset que incluya a todas las features con distintos lags positivos -hacia el futuro- (como trayendo a tiempo presente su comportamiento en el pasado para distintas cantidades de tiempo -hasta 6 meses).

Meses	Nivel de precios (P)	V	M	Q	V + 1	V + 2	...	M + 1	...
1	P1	V_lag?	M_lag?	Q_lag?	Nan	Nan	...	Nan	...
2	P2	V_lag? + 1	M_lag? + 1	Q_lag? + 1	V_lag?	Nan	...	M_lag?	...
3	P3	V_lag? + 2	M_lag? + 2	Q_lag? + 2	V_lag? + 1	V_lag?	...	M_lag? + 1	...
4	P4	V_lag? + 3	M_lag? + 3	Q_lag? + 3	V_lag? + 2	V_lag? + 1	...	M_lag? + 2	...
5	P5	V_lag? + 4	M_lag? + 4	Q_lag? + 4	V_lag? + 3	V_lag? + 2	...	M_lag? + 3	...
...



Eso ha generado una cierta cantidad de nuevas features; que no son más que distintos desplazamientos las tres variables originales cuyos lags los conozco por construcción pero el modelo lo desconoce.

	Meses	Velocidad del Dinero (V)	Masa Monetaria (M)	Producción Real (Q)	Nivel de Precios (P)	V_lag_0	M_lag_0	Q_lag_0	V_lag_1	M_lag_1	...
0	8	1.673658	44.563140	110.874347	1.196066	1.673658	44.563140	110.874347	1.754256	76.357289	...
1	9	1.693427	70.564001	98.905256	0.993256	1.693427	70.564001	98.905256	1.673658	44.563140	...
2	10	1.784196	46.114558	109.425600	1.131246	1.784196	46.114558	109.425600	1.693427	70.564001	...
3	11	1.588672	46.756756	109.337244	0.705889	1.588672	46.756756	109.337244	1.784196	46.114558	...
4	12	2.092062	65.149057	100.142702	0.790663	2.092062	65.149057	100.142702	1.588672	46.756756	...
...
165	173	5.102718	180.328799	178.686507	4.688608	5.102718	180.328799	178.686507	5.027669	169.172457	...
166	174	7.967666	179.417270	173.011520	5.226180	7.967666	179.417270	173.011520	5.102718	180.328799	...
167	175	5.205353	197.805947	172.766908	5.221200	5.205353	197.805947	172.766908	7.967666	179.417270	...
168	176	5.053534	174.225047	178.671832	5.649173	5.053534	174.225047	178.671832	5.205353	197.805947	...
169	177	5.372017	210.938352	170.930019	8.121259	5.372017	210.938352	170.930019	5.053534	174.225047	...

Mediante el uso de árboles de regresión, he buscado el “feature importance”. Podría ser una combinación de éstas tres variables en algún lag oportuno para cada una. Pero lo curioso, para esta

primera instancia, es que no he necesitado exigir al modelo una respuesta óptima para cada combinación. Vale decir que en este caso no he construido las variables con autorregresión, es decir, el valor pasado de una cualquiera de las variables no es una regresión de uno anterior necesariamente (pero en economía bien podría suceder diferente).

El feature importance ha revelado una feature importante por cada variable. He probado esto para distinto número de lags razonables (0 a 3 meses) y el regresor ha descubierto con éxito los lags deseados con éste método.

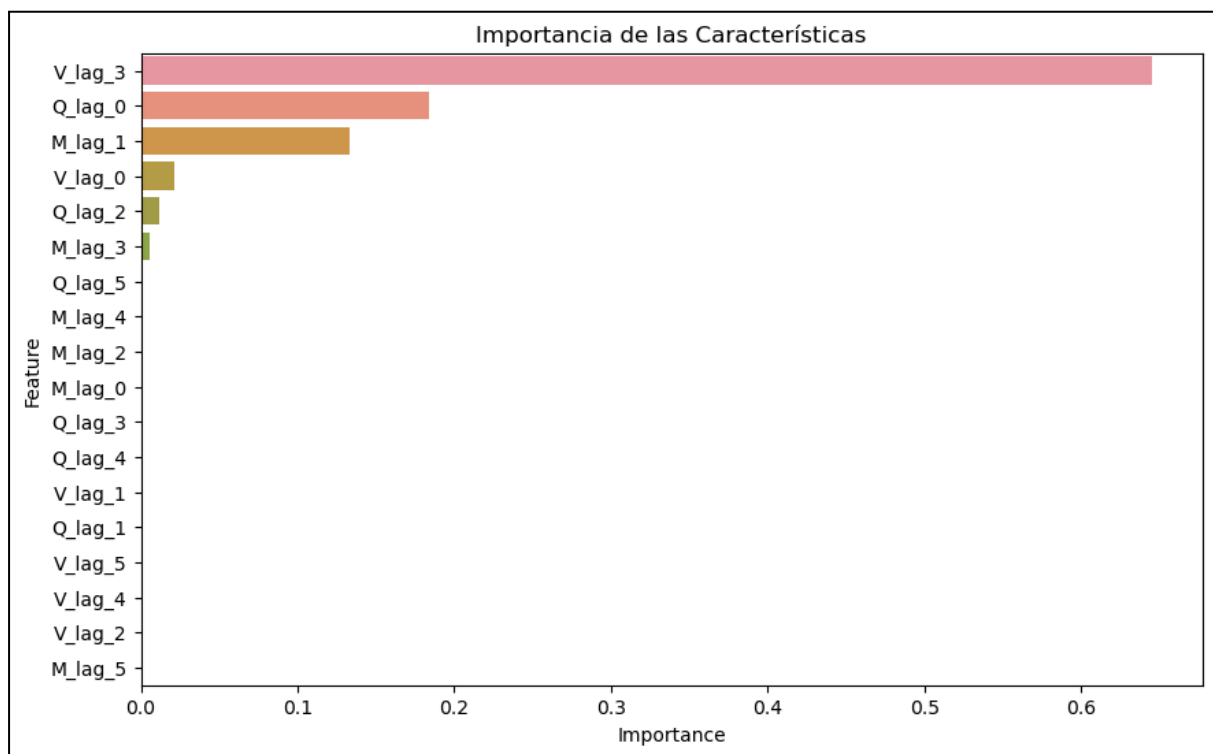
Por ejemplo, para la variable

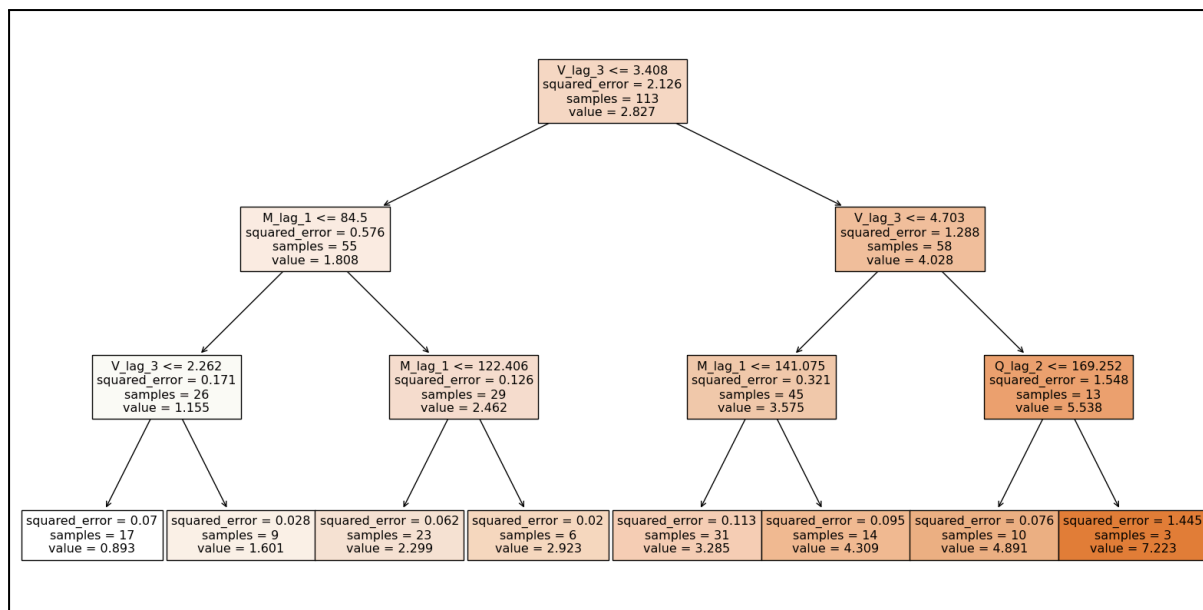
M con 1 lag,

La variable V con 3 lags y

Q con 0 lag

obtengo:





Detalle del árbol:

```

|--- V_lag_3 <= 3.41
|   |--- M_lag_1 <= 84.50
|   |   |--- V_lag_3 <= 2.26
|   |   |   |--- value: [0.89]
|   |   |   |--- V_lag_3 > 2.26
|   |   |   |--- value: [1.60]
|   |   |--- M_lag_1 > 84.50
|   |   |   |--- M_lag_1 <= 122.41
|   |   |   |   |--- value: [2.30]
|   |   |   |   |--- M_lag_1 > 122.41
|   |   |   |   |--- value: [2.92]
|   |--- V_lag_3 > 3.41
|   |   |--- V_lag_3 <= 4.70
|   |   |   |--- M_lag_1 <= 141.07
|   |   |   |   |--- value: [3.28]
|   |   |   |   |--- M_lag_1 > 141.07
|   |   |   |   |--- value: [4.31]
|   |   |--- V_lag_3 > 4.70
|   |   |   |--- Q_lag_2 <= 169.25
|   |   |   |   |--- value: [4.89]
|   |   |   |   |--- Q_lag_2 > 169.25
|   |   |   |   |--- value: [7.22]
  
```

Observo que el 95% o más del target se explica con la importancia de efectivamente, las variables desplazadas con los lags de construcción del dataset. Lo cual es una buena noticia. El método sirve para el objetivo que buscaba.

Este árbol que se ha detallado ha encontrado que efectivamente, con los lags introducidos en el momento de la construcción de los datos, se puede predecir con buena bondad de ajuste los valores del target. He probado distintos parámetros en la configuración del árbol y en virtud de no overfitear

me he quedado con esta versión reducida que satisface el ajuste que estaba buscando. De todas formas, la eficiencia no es necesariamente requerida en esta etapa del método que simplemente trata de encontrar el feature importance.

Un algoritmo PCA podría ser adecuado para crear un set de datos de entrenamiento y luego aplicar el método de bootstrapping. El problema es que un algoritmo de PCA realiza combinaciones de las features y las colapsa en la dirección de máxima explicación del target (capturar la máxima varianza) Puede ser potente para realizar un modelo predictivo, pero se pierde interpretación del fenómeno económico de impulsos respuestas.

Notar que en la construcción de los datos, he introducido ruido gaussiano de duración mayor a un período de tiempo. Y el feature importance ha relevado como semi-importante algunos lags alrededor de los principales. Lo que me sugiere que es un método bueno para descubrir el orden de rezagos.

Esta parte del ejercicio se puede encontrar en el archivo **feature_importance.ipynb**

Para la siguiente parte del ejercicio, seguimos el archivo **modelo ecuacion cuantitativa.ipynb**

(se repite al principio del script la construcción del DF de feature importance, pero el objetivo está en las partes del código que siguen, en las que intento explorar las capacidades de ajuste de un modelo de random forest para los datos sintéticos).

El objetivo de esta parte del ejercicio es encontrar los rezagos que explican a la variable target para así crear un NUEVO dataset que posea al mismo nivel temporal el índice del target junto con el impulso en la feature correspondiente. Es decir, olvidaré el resto de las features que no me interesan. Si por ejemplo, en el mes de junio el nivel de precios tiene un valor influenciado por la masa monetaria del mes anterior, entonces el data frame para esta nueva etapa tendrá la variable masa monetaria adelantada un mes para así estar el impulso y la respuesta en una misma fila del dataframe.

¿Por qué realizo esto?

justamente porque uno de los problemas de las series de tiempo en econometría son la poca cantidad de datos disponibles. Construir un dataset de esta manera, me puede permitir realizar bootstrapping y entrenar múltiples árboles y con un dataset relativamente pequeño, estimar parámetros del modelo con cierta robustez. Además, este método, permitiría transformar de alguna manera los datos que deberían ajustarse en un modelo paramétrico con uno no paramétrico.

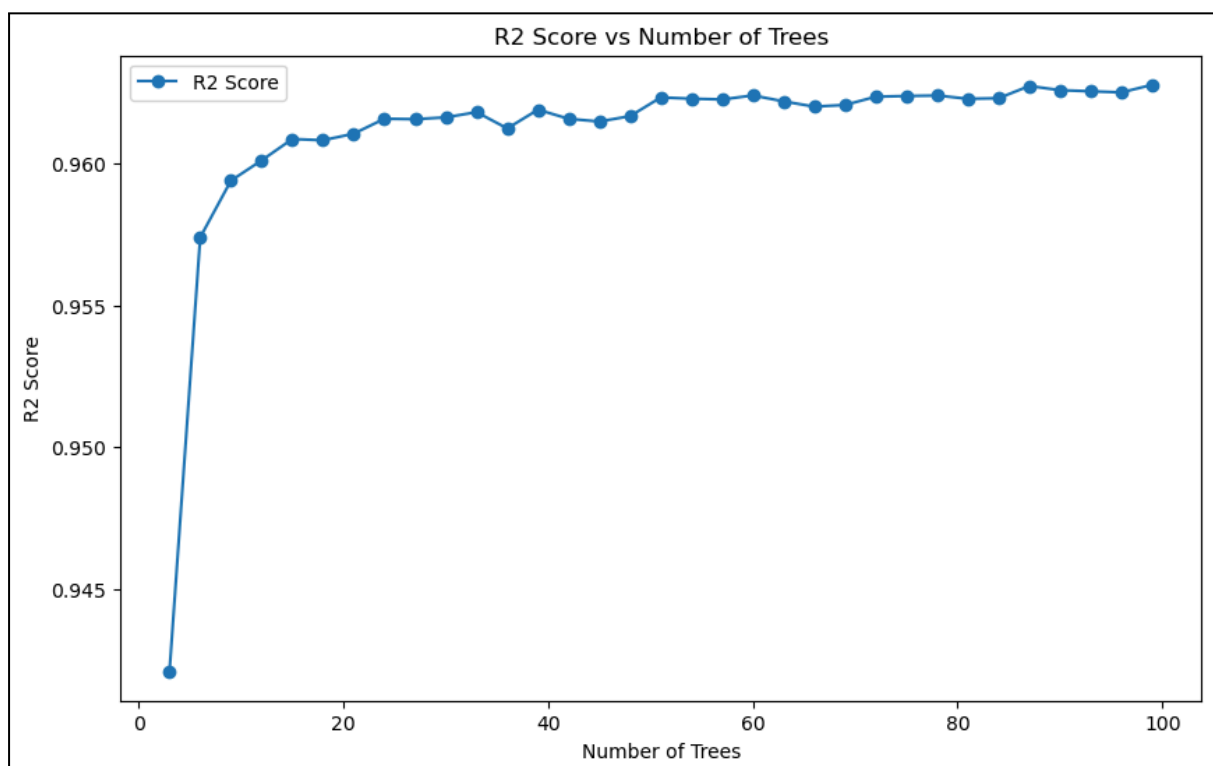
Es normal que las series de tiempo de macroeconomía sean breves. Por distintas razones. Por lo general, cosas como niveles de precios se calculan en base a una canasta de bienes que con el tiempo puede ser cambiada según las prácticas de consumo de la población, o que haya influencia en precios debido a cambios tecnológicos/productivos. La velocidad de circulación misma del dinero puede ser variable en función de los cambios de métodos de pago, hoy en día el pago mediante transferencia es un facilitador de la circulación con respecto de tiempos donde se utilizaban cheques o billetes. y un largo etcétera.

Existen intentos de parte de los departamentos de estadística que se dedican a construir los datos para crear coeficientes de empalme en índices de series históricas, pero, incluso la confiabilidad en los datos que poseen, puede ser cuestionable.

Es común en series de tiempo escindir datos consecutivos de la cola para luego comparar la predicción con los datos. O incluso en bootstrapping, y considerando la importancia de los lags, he leído que un método utilizado es realiza block bootstrapping, para así incorporar en el modelo esta característica de los impactos con rezagos. Con este método que estoy proponiendo, el block bootstrapping no sería en principio, necesario. Y el modelo no pierde interpretabilidad.

Cabe preguntarse entonces para el conjunto de entrenamiento, qué cantidad de árboles deberé utilizar para tener un ajuste aceptable. Esto se puede observar en en el siguiente gráfico, para los parámetros

```
8 # Espacio de parámetros
9 params = {
10     'sample_size_ratio': 1/3, # Cantidad de datos de muestreo (1/3 del total)
11     'validation_size_ratio': 0.2, # Cantidad de datos de validación (20% del total de datos)
12     'n_estimators': 100, # Número de árboles en el random forest
13     'tree_type': 'regressor', # Tipo de árbol: 'regressor' o 'classifier'
14     'random_state': 42, # Semilla para la reproducibilidad
15     'max_depth': None, # Profundidad máxima de los árboles
16     'min_samples_leaf': 1, # Número mínimo de muestras por hoja
17     'n_splits': 5 # Número de splits para bootstrapping
18 }
19
```

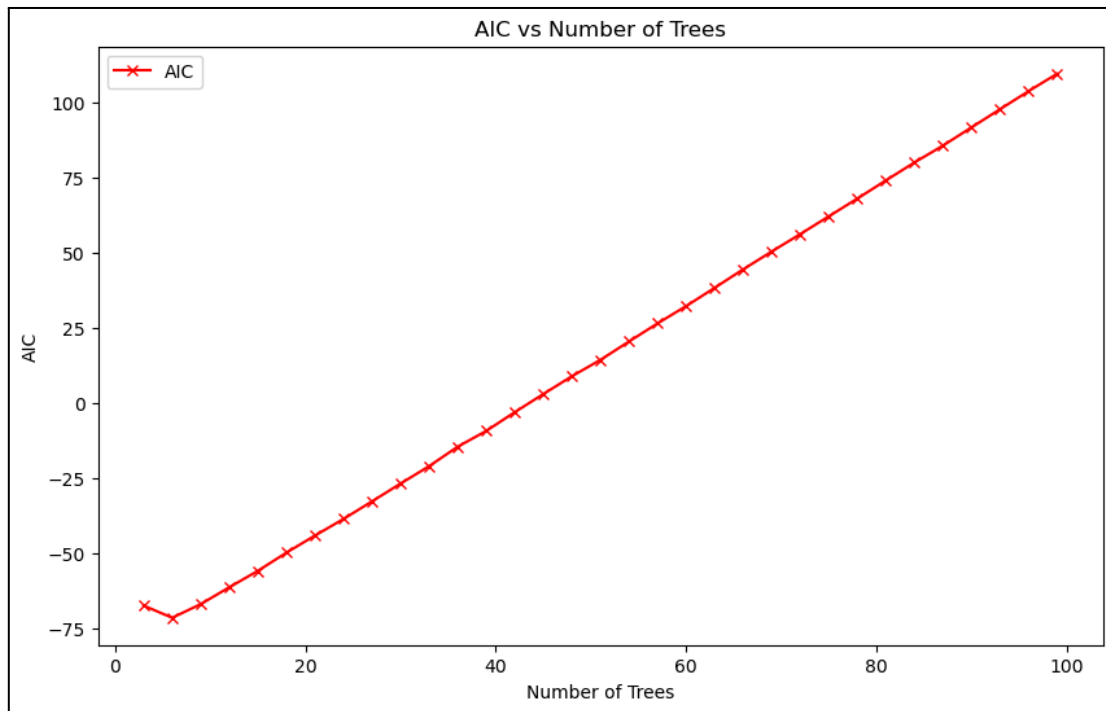


Parece ser, que a partir de los 5 árboles (splits) ya podríamos garantizar que el R cuadrado se encuentra por encima de 0.95. Pero ese no es un criterio suficiente para elegir un buen compromiso entre la complejidad del modelo y el ajuste adecuado. Para elegir esto me apoyé en el criterio de

información de akaike (AIC) que propone un mínimo óptimo para el logaritmo de la verosimilitud penalizada por la cantidad de parámetros de un modelo

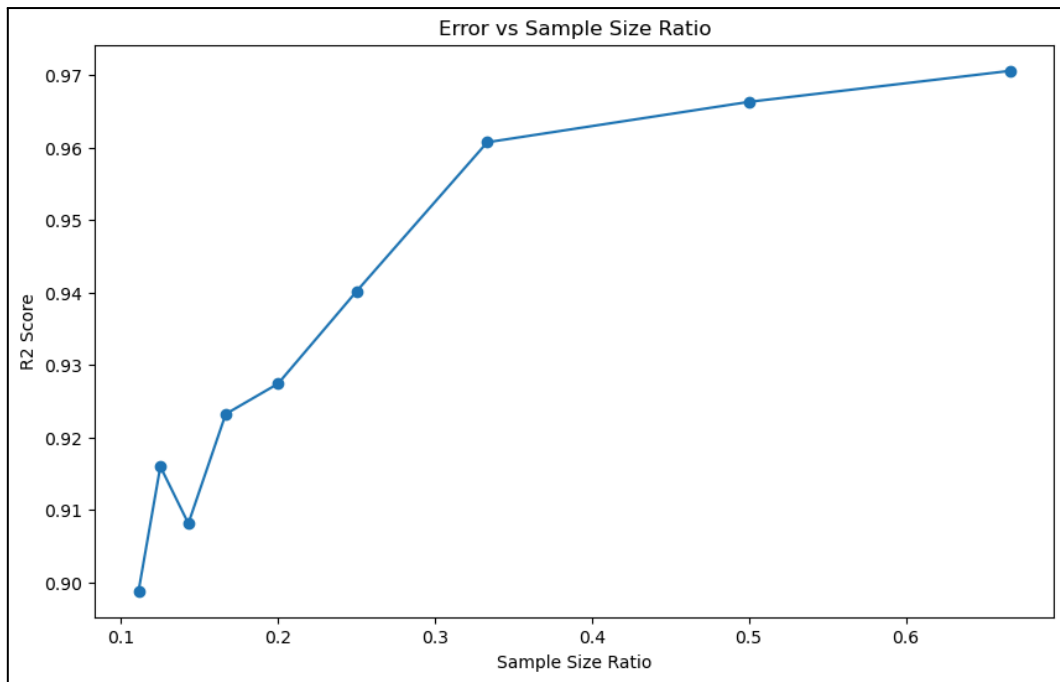
https://en.wikipedia.org/wiki/Akaike_information_criterion

En lugar de calcular la verosimilitud para este modelo de random forest, he optado por el logaritmo de MSE (Mean squared error) penalizado por la cantidad de árboles que se agrega al modelo y efectivamente, el criterio sugiere elegir una cantidad de árboles de alrededor de 10 o menor. Tal como sugiere el siguiente gráfico (observar como en la medida que se van agregando splits el criterio empieza a penalizar por complejidad del modelo):

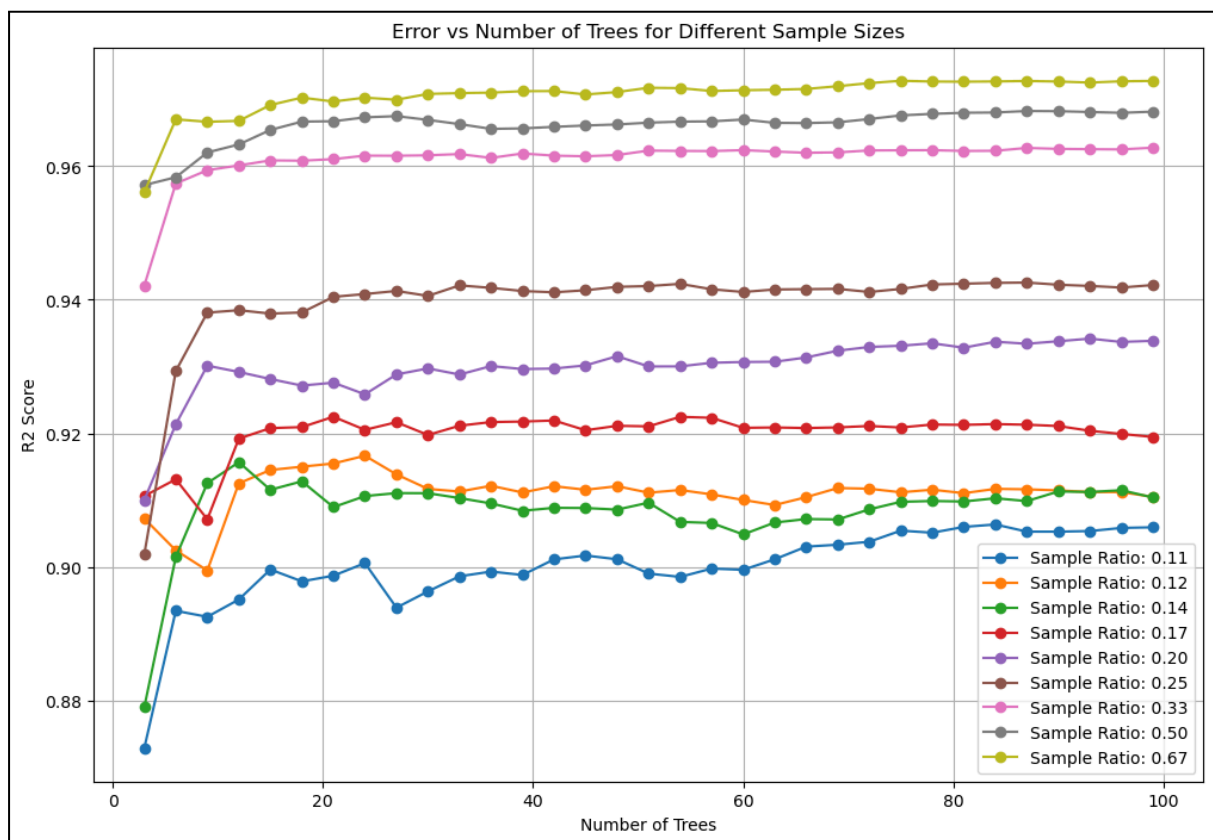


De ésta manera se puede elegir un modelo parsimonioso.

No es la única perilla que podemos explorar en el espacio de parámetros del modelo, observemos que consistentemente con lo esperado el r^2 mejora en la medida que los conjuntos de bagging son mayores (muestreos aleatorios)



Si combinamos esta información, en conjunto con el número de árboles, podemos identificar qué modelo puede ser más conveniente, resumo la información en el siguiente gráfico:



Ahora bien, cuando se realiza bagging aleatorio, existe la chance de que algunos datos queden por fuera del conjunto de entrenamiento. En este caso particular, yo he definido manualmente un 20%

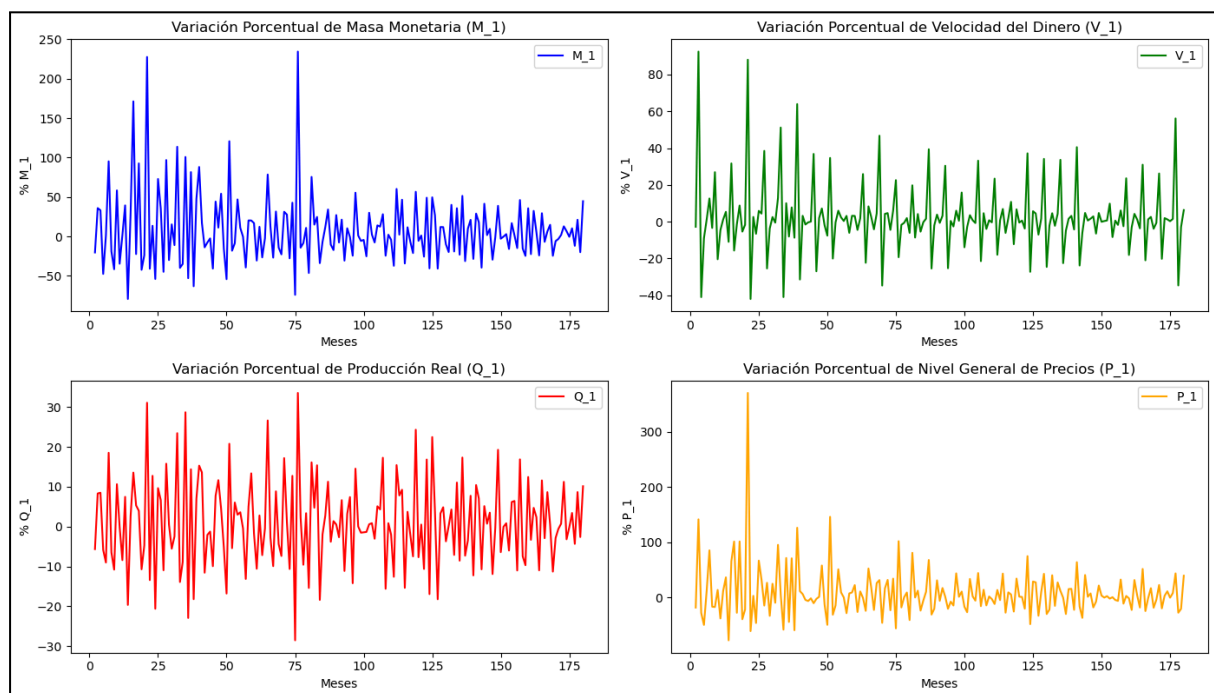
de los datos para el conjunto de validación, pero del otro 80% que va al muestreo aleatorio, existe la posibilidad de que no se aprovechen todos los datos para el training. Esto debe ser considerado en el momento de realizar un ajuste.

Datos (ecuación cuantitativa) en variaciones porcentuales.

Es posible también, encontrarse con series de tiempo que poseen datos en variaciones porcentuales en lugar de niveles, tales como indagué en la primer parte del ejercicio. Esto puede tener complicaciones sobre todo porque la volatilidad de los datos se hace más presente y si se remueve el trend es posible que las variaciones porcentuales se parezcan mucho al ruido blanco.

No obstante, he decidido implementar el modelo de la ecuación cuantitativa en variaciones porcentuales para todas sus variables para evaluar la robustez del método.

El script puede encontrarse en el archivo **porcentuales.ipynb**



En este caso el descubrimiento de “feature importance” también puede estar afectado por la construcción de los datos, puesto que una variación porcentual implica calcular la diferencia entre períodos.

Para un shift de:

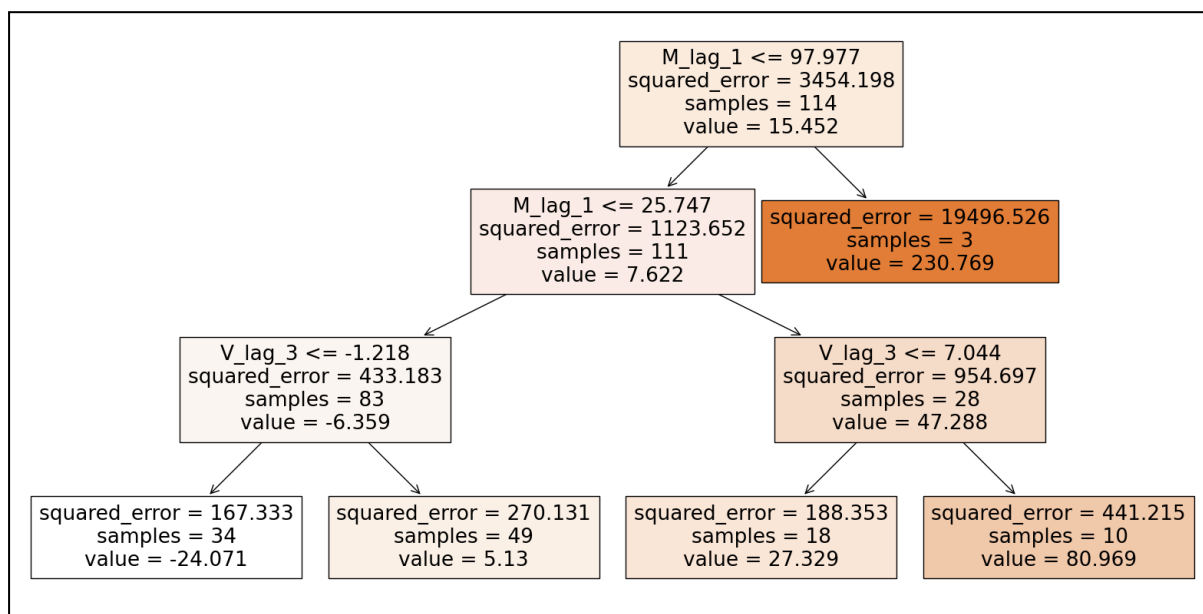
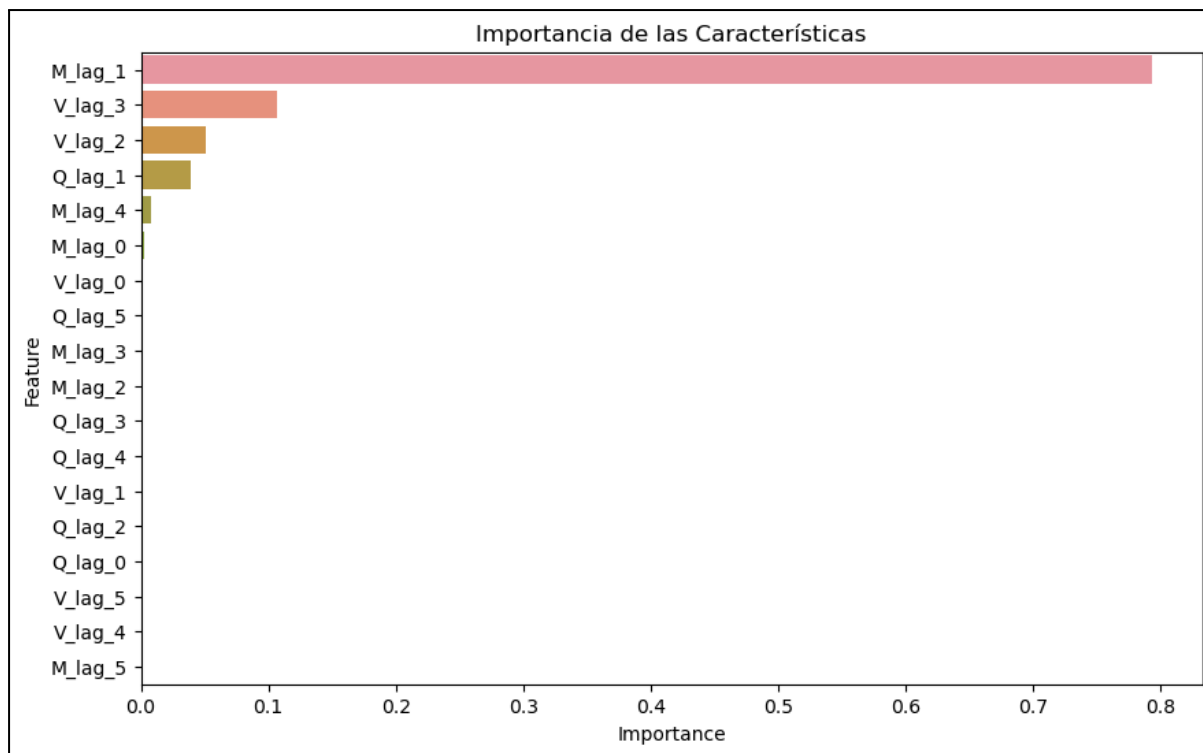
V = 3 lags

M = 1 lag

Q = 0 lag

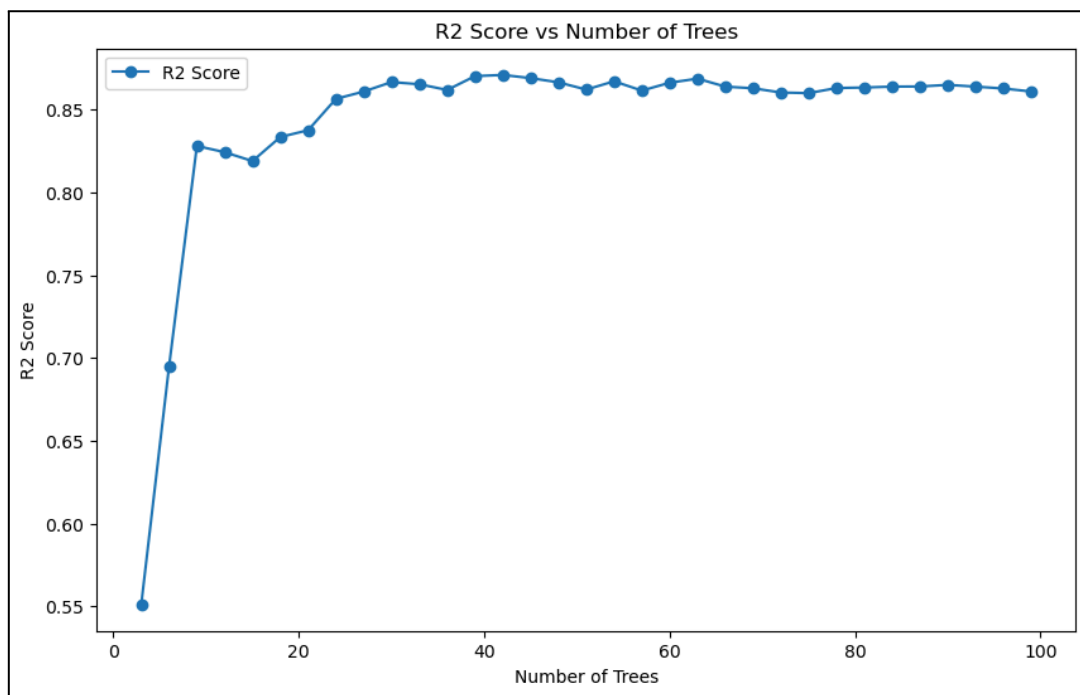
```
df['Velocidad del Dinero (V)'] = df['Velocidad del Dinero (V)'].shift(-3)
df['Producción Real (Q)'] = df['Producción Real (Q)'].shift(0)
df['Masa Monetaria (M)'] = df['Masa Monetaria (M)'].shift(-1)
```

Obtengo:



A priori me parece que tiene sentido, cuando construí el ruido en los datos propuse que para la variable M el ruido con picos gaussianos tenga un impulso de 1 mes, mientras que para V y Q propuse que dure 2 meses los picos gaussianos. Por lo tanto estaba esperando más “confusión” en el algoritmo para identificar el lag más explicativo. Eso no pareció importar tanto en la serie en niveles pero sí en la serie en porcentajes.

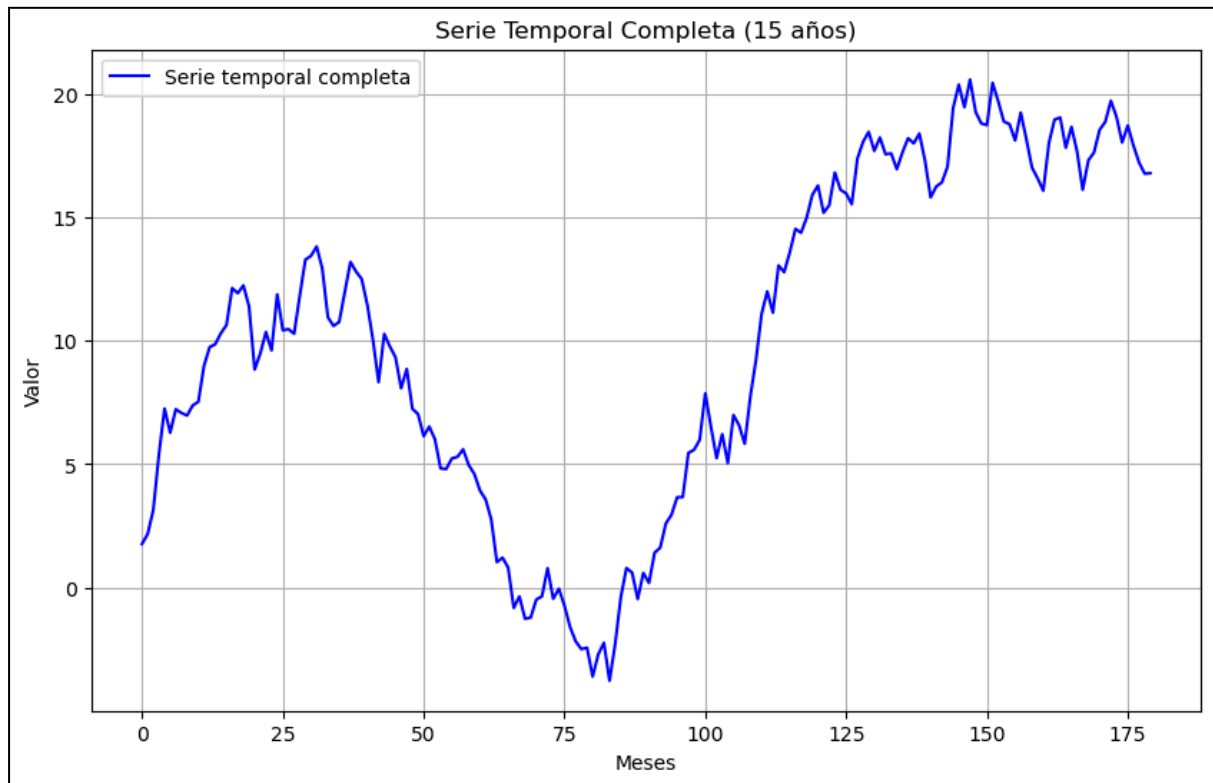
He propuesto el mismo ensayo de ajuste que en el caso de la serie en niveles. Y mientras que en la serie en niveles obtenemos R_2 superiores a 0.95. El desempeño baja a valores apenas superiores a 0.85 en este caso



PARTE 3 Forecast en procesos ARMA

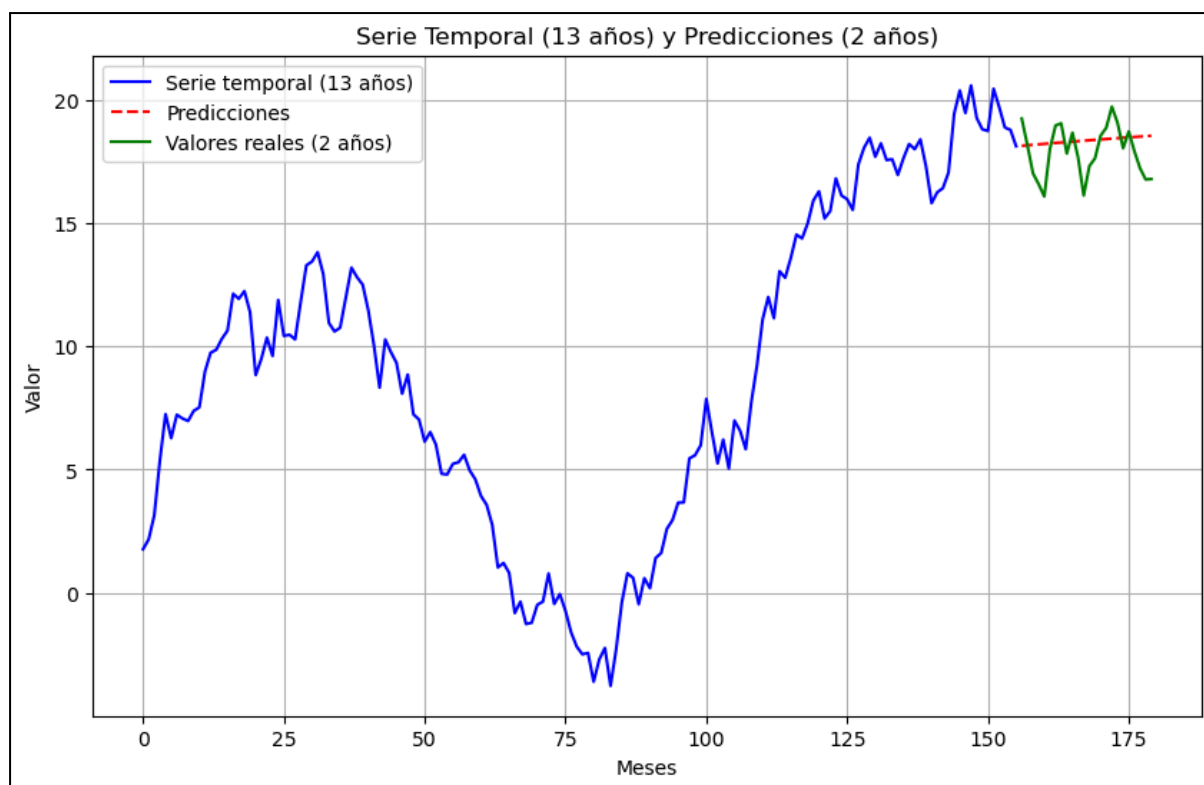
En los datos de series de tiempo, también se presentan aquellos modelos tipo autorregresivos, es decir que valores que una misma variable tiene en el pasado, afecta sus propios estados futuros. Si además se considera que estas variables poseen un ruido aleatorio, su media también será móvil. debido a esto, se los ajusta con modelos tipo ARMA.

Yo he creado un conjunto de datos sintéticos tipo ARMA también

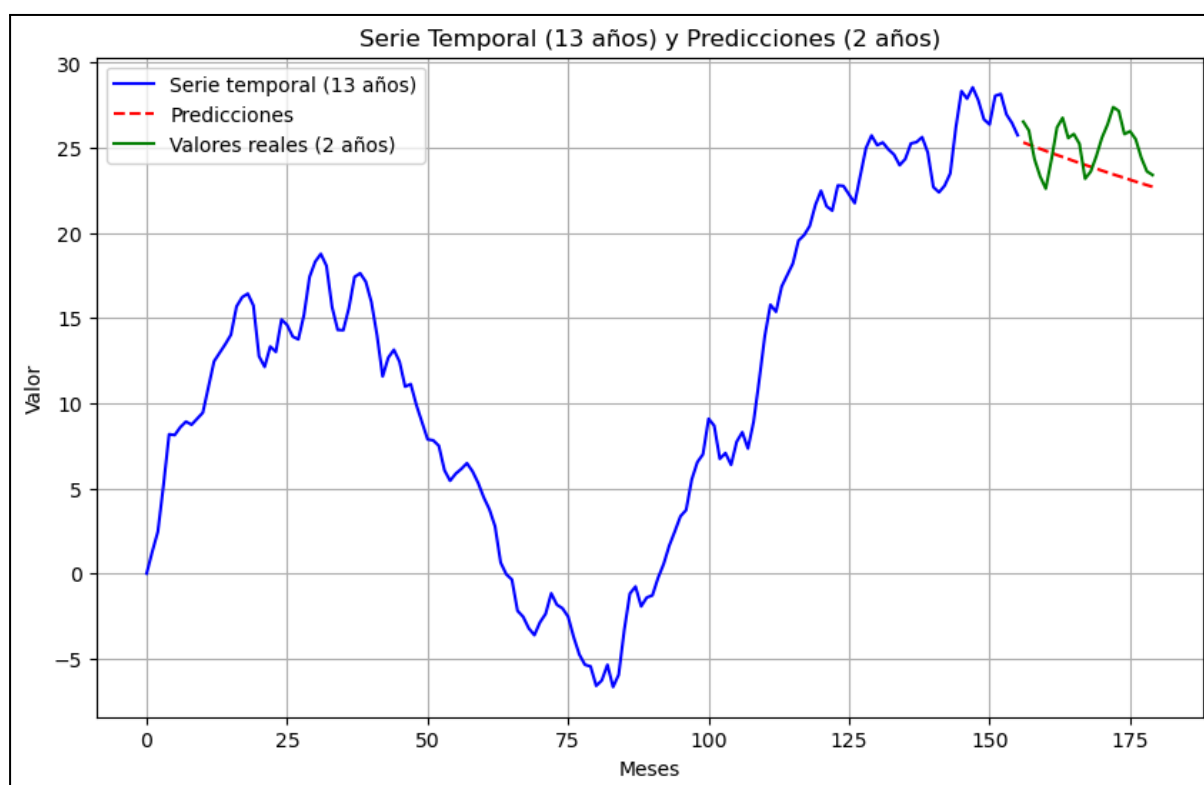


Aquí el parámetro autorregresivo es de 1 rezago.

Para un ajuste exclusivamente autorregresivo, el paquete de Statsmodels realiza una predicción que he graficado en una línea punteada en rojo para la cola de la serie



Si además consideramos la media móvil, obtenemos una predicción con el paquete de Statsmodels de ARMA tal como sigue.



Ahora bien, implementaré nuestro método ya aprendido para los datos de la ecuación cuantitativa del dinero(de la primera parte)
¿Cómo?

Bueno, en principio puedo construir un dataset de la variable autorregresiva y acompañarlo de la misma variable pero con distintos desplazamientos, es decir, reproduciendo la misma variable suponiendo la existencia de posibles lags

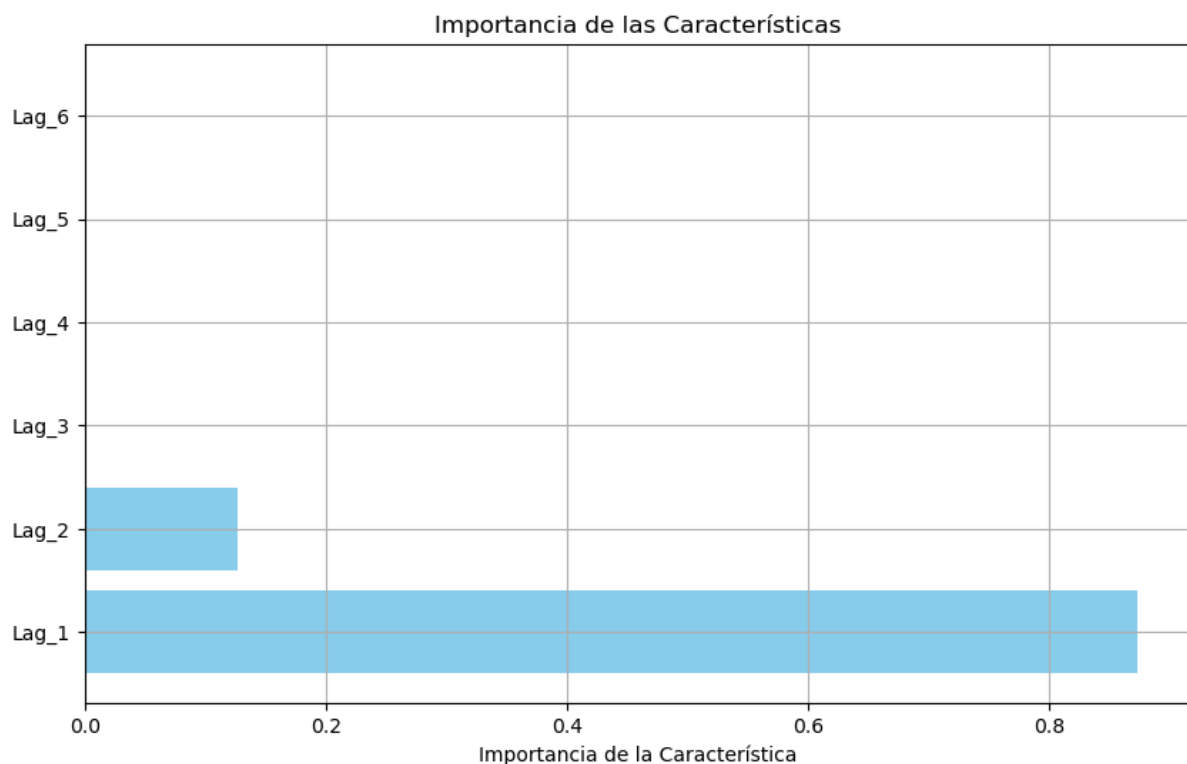
	Month	Target	Lag_1	Lag_2	Lag_3	Lag_4	Lag_5	Lag_6
0	7	8.597217	8.135768	8.179267	5.191262	2.461000	1.282183	0.000000
1	8	8.920904	8.597217	8.135768	8.179267	5.191262	2.461000	1.282183
2	9	8.742007	8.920904	8.597217	8.135768	8.179267	5.191262	2.461000
3	10	9.100996	8.742007	8.920904	8.597217	8.135768	8.179267	5.191262
4	11	9.450339	9.100996	8.742007	8.920904	8.597217	8.135768	8.179267

Por ejemplo, hasta 6 lags. Si bien algunos procesos estacionales como una cosecha puede tener un rezago anual (12 meses) considero que contemplar hasta 6 lags es suficiente.

Observar que cada “lag” es la misma variable target desplazada en +1 respecto del anterior.

He eliminado filas Nans.

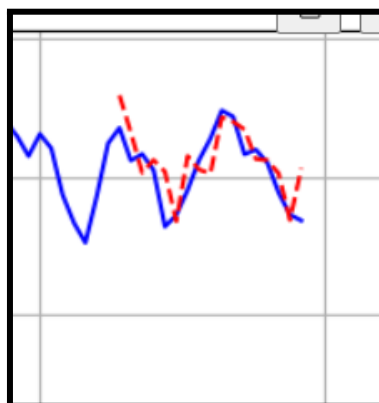
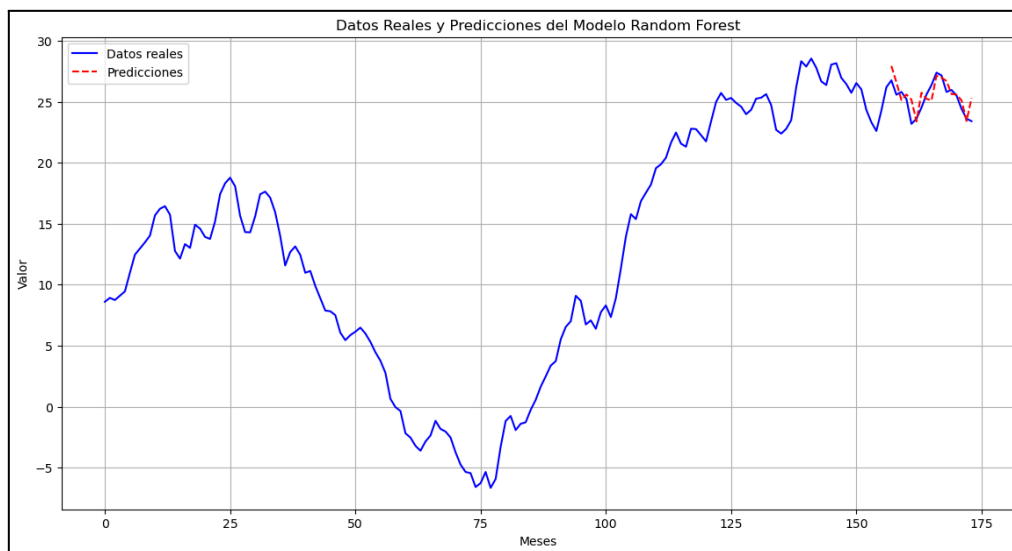
Para el dataset construido con 1 rezago y ruido, el feature importance de un regressor tree indica, tal como es deseable:



Por lo tanto, construyo para realizar bootstrapping y random forest, un nuevo dataset con la variable target original y los dos primeros lags.

	Target	Lag_1	Lag_2
0	8.597217	8.135768	8.179267
1	8.920904	8.597217	8.135768
2	8.742007	8.920904	8.597217
3	9.100996	8.742007	8.920904
4	9.450339	9.100996	8.742007

Cuando aplico el mismo procedimiento, con la experiencia ya aprendida acerca de cómo setear los parámetros de ajuste, obtengo una buena performance en la predicción o “forecast” para la variable, como podemos observar en la línea punteada roja. El ajuste es en apariencia muy aceptable.



Todos estos resultados se pueden reproducir en el archivo **autorregresivo.ipynb**

Para complementar esta parte, conviene extender el análisis hacia los intervalos de confianza del forecast, pero lo dejaré como ejercicio futuro.

Parte 4 predicción de inflación con DATOS REALES

Tengo un set de datos en el que he trabajado previamente (su análisis inspiró este ejercicio) y me interesó realizar predicciones sobre datos reales con el algoritmo trabajado hasta ahora.

El archivo donde realicé este ajuste es **passthrough.ipynb**

Además viene acompañado de los datos: **datos_log.csv**

El problema consiste en medir el passthrough. En economías abiertas, la inflación local puede estar relacionada como vimos en el ejercicio de datos sintéticos a la emisión, circulación y producción. Pero si factores como los productivos dependen de la importación o exportación de materias primas, el valor de las materias primas en el mercado internacional tiene su impacto, también lo tendrá algún a medida de emisión monetaria como M2, el tipo de cambio, que puede ser el oficial o el “blue”, y también he considerado un índice que refleja la actividad económica a través de un estimador que publica el INDEC (EMAE)

Los datos para estimar la inflación con los que trabajaré serán pues (en logaritmos)

```
columns=['loginflacion', 'logblue', 'logoficial', 'logbadlar',  
'logemae', 'logm2', 'logmat_prim'])
```

Ahora bien, he repetido esencialmente el procedimiento de la primera parte donde vimos la ecuación cuantitativa del dinero.

Paso 1, construir un subset de datos que incluyan las features rezagadas. A diferencia del caso 1, consideré que la inflación podría ser inercial y por lo tanto, he incluido en las features algunos rezagos de la misma variable target.

Luego de eso, he procedido a seleccionar mediante “feature_importance” aquellas principales features que sumadas sus relevancias, explican el 95% del target “loginflación”.

Aquí observaremos varios rezagos incluso de una misma variable, porque estoy otorgando a la posibilidad de que el modelo tenga múltiples procesos autorregresivos y de media móvil. Esto puede ser un inconveniente sobre todo para cuando se desea hacer forecast porque la propagación de errores en medias móviles disminuyen los intervalos de confianza en el modelo. Es una consideración que tendré que tener en cuenta luego. Por lo pronto utilizaremos el modelo exclusivamente para realizar predicciones sobre el conjunto de features existentes.

Así, considerando estas principales características, el dataset original compuesto por 1 target y 6 predictores, se ha transformado en un dataset de 1 target y 23 predictores (o features).

Ahora podemos empezar con el proceso de bagging.

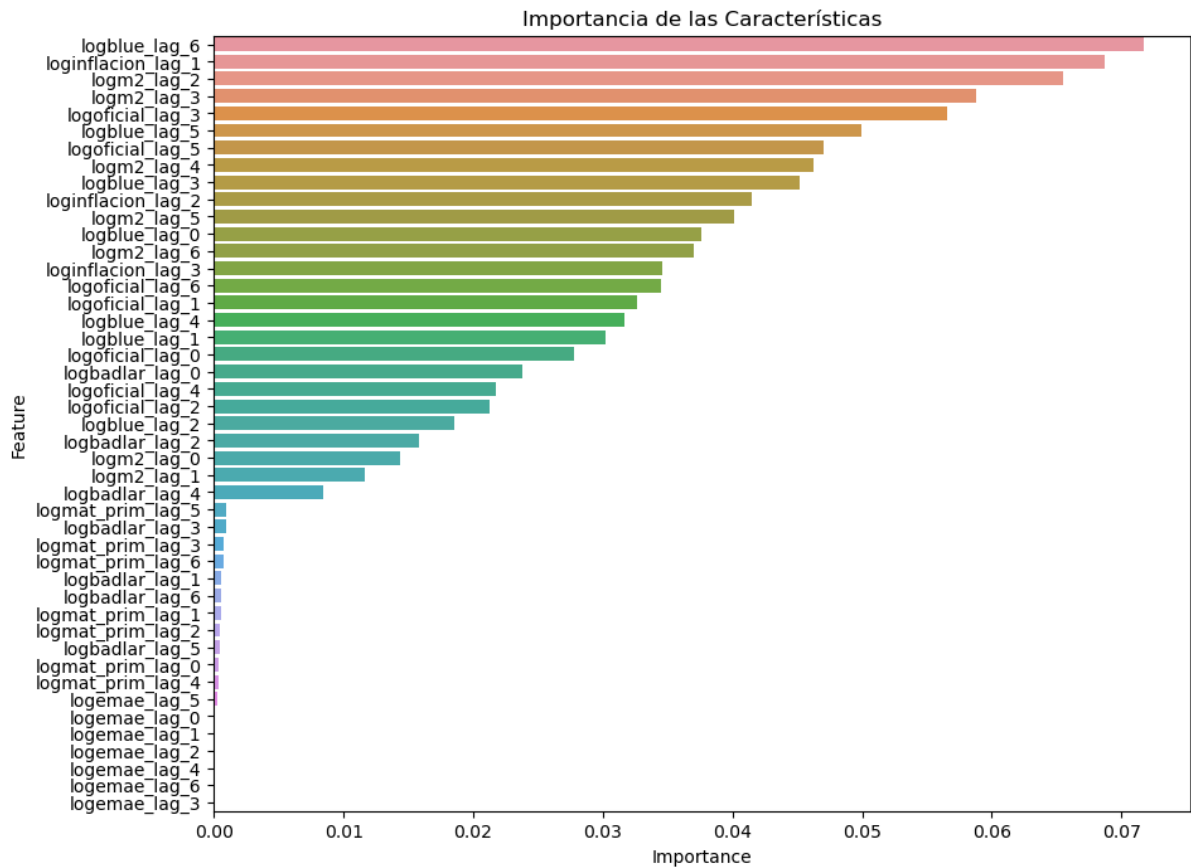
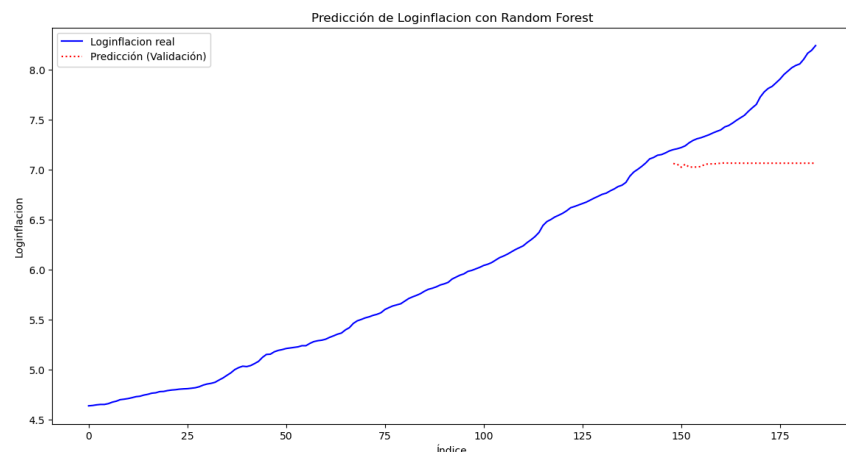


Gráfico de relevancia de las features con sus lags

Por algún motivo, este modelo no estaba captando un trend lineal de la inflación logaritimizada (nivel de precios de crecimiento exponencial). Y si bien el ajuste parecía satisfactorio, las predicciones no funcionaron como se esperaba. Muestro un conjunto de predicciones en línea punteada roja

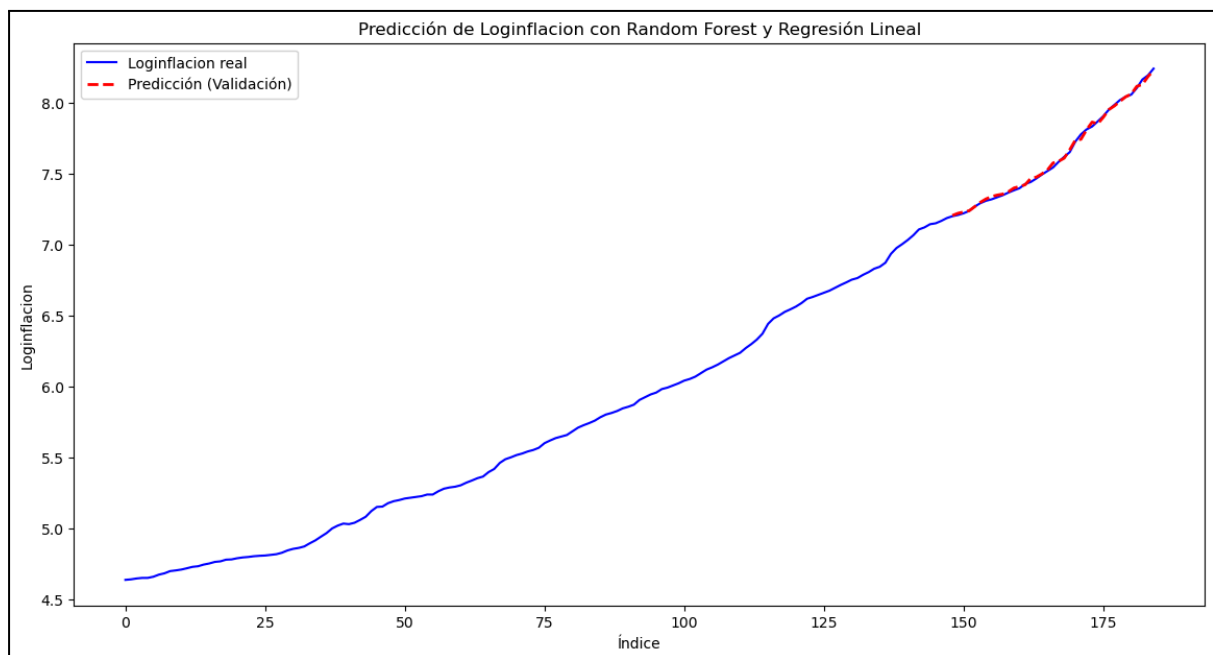


no he conseguido realizar predicciones adecuadas sin sustraer primero la tendencia lineal prevalente en los datos de la inflación. Sospecho que es posible que haya magnitudes de diferencia entre los datos no normalizados que no permiten un ajuste efectivo. Deberé indagar en los procesos de normalización de los datos para entrenar estos modelos de tipo “impulso-respuesta” de forma adecuada.

Una vez que el trend lineal ha sido sustraído de los datos de la inflación he repetido el ajuste sobre el 20% de los datos finales que han sido expresamente escindidos del conjunto de entrenamiento. Y el resultado del ajuste es bueno. Es de esperar valores de R2 altos debido a la cointegración de datos, pero es sorprendente la estabilidad del entrenamiento en una ventana tan grande para la predicción como el 20% de los datos. Para un dataset de 185 meses en total, quiere decir que el entrenamiento con el 80% de esos datos es útil para una predicción estable en los 37 meses posteriores. Sin haber considerado parámetros externos.

Train MSE: 3.861359430748384e-05
Validation MSE: 0.00023345311695874033
Train R^2: 0.9999336828149561
Validation R^2: 0.9977195472061845

Detalle del 100% de la serie de tiempo:



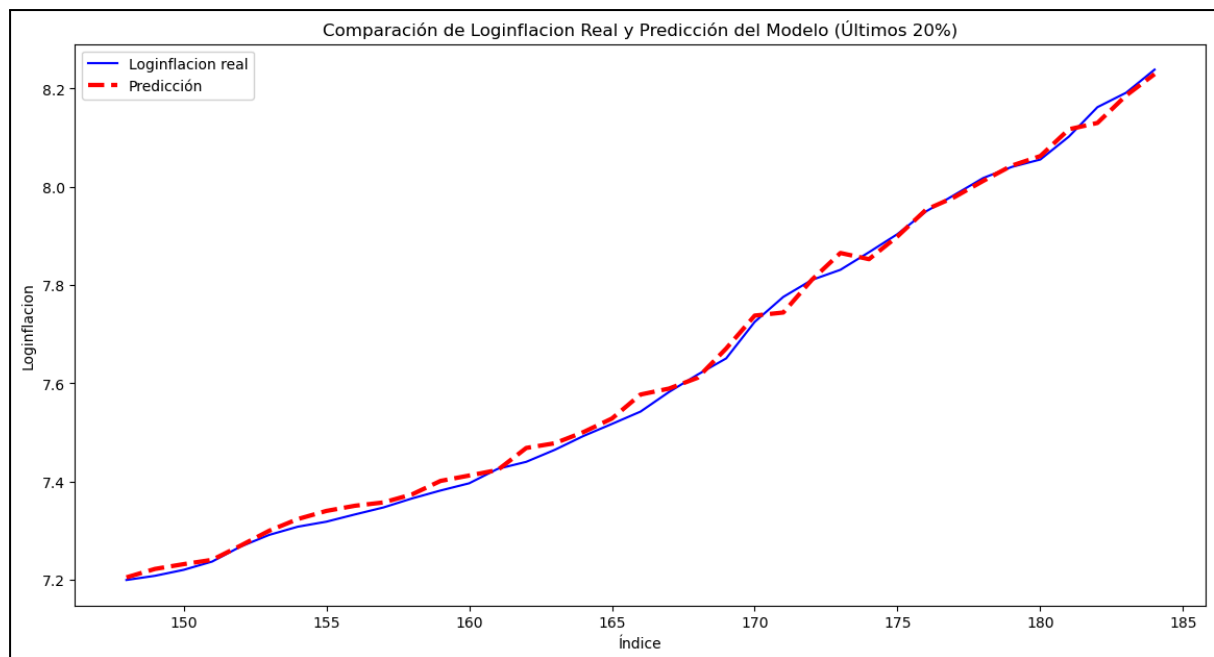


Gráfico con detalle del último 20% (intervalo de predicción)

considero que la técnica implementada de “traer a tiempo presente” distintos rezagos de las features, para luego tratar a la serie de tiempo como un junto de datos de corte transversal, lo que permite hacer bootstrapping, ha demostrado tener potencial. Los ajustes se ven bien y el modelo parece robusto tanto para predicciones como forecast, sorteando los problemas de multicolinealidad (que se pueden traducir en intervalos de confianza menores para los procesos de media móvil) y también robusto frente a intervenciones exógenas que los datos reales sin duda poseen.

Este ejercicio ha despertado ciertas inquietudes que deben ser resueltas, como por ejemplo la estimación de los intervalos de confianza en este modelo. Pero considero que el método en general puede ser una buena herramienta para tratar series de tiempo con las características mencionadas al principio del documento.

