Anuario Jurídico y Económico Escurialense, **LVII (2024) 395-410 / ISSN: 1133-3677**

La predicción masiva como servicio en la RED

The prediction at Scale, as a service, to the WWW

Dr. Agustín ALONSO RODRIGUEZ
Real Centro Universitario
"Escorial-María Cristina"
San Lorenzo del Escorial

Resumen: Las grandes empresas tecnológicas que operan en la Red, han desarrollado, en respuesta a las peticiones de sus clientes, programas infomáticos para el análisis y la predicción de acontecimos, a escala industrial, utilizando series temporales.

En este trabajo, se utiliza el programa de software libre, *Prophet* para el análisis de la serie cronológica que recoge la evolución del tipo de interés de los Bonos del Tesoro, a 10 años, en España, desde enero de 1980 hasta agosto de 2023.

Abstract: The big operating technological companies in the WWW, have developed, in response to the request of their clients, statistical programs for the analysis and prediction of events, at Scale, with the use of time series. In this paper, it is used one of these free source software programs *Prophet* for the analysis of the time series representing the evolution of the Long Term Government Bond Yiels: 10-Year, in Spain, from January 1980 to August of 2023.

Palabras clave: La RED, la predicción como servicio a clientes, series temporales, automatismo, rapidez, programas: Prophet, Python.

Keywords: The WWW, the forecasts at Scale, as a service to clients, time series analysis, software: Prophet, Python.

Sumario:

Recibido: septiembre 2023.
Aceptado: noviembre 2023

I. INTRODUCCIÓN

Recientemente, las grandes empresas tecnológicas que operan en la WEB, han desarrrollado paquetes estadísticos y publicado trabajos en vistas a atender la demanda de sus clientes para la elaboración de predicciones. Para estos desarrolladores de software el objetivo es la obtención de predicciones *a escala industrial*, de manera rápida y fácil.

Google, Facebook, Twiter, Amazon, y otros, han generado paquetes, no todos, *open source* para atender a la demanda de análisis y prediciones factibles a partir de la enorme información disponible en la WEB. Nace así **la prediccion como servicio**.

Este es un ámbito de acuciante actualidad, al que se desea dar cabida en este trabajo.

II. EL PROGRAMA PROPHET

Este paquete, *open source*, creado por Meta (FaceBook,Taylor y Letham, 2018) para la predicción con series temporales, a *escala industrial*, toma en consideración tendencias no lineales, y múltiples periodos estacionales: anuales, mensuales, semanales, diarios. Su implementación en *Python* permite obtener predicciones con rapidez y mínimo esfuerzo, permitiendo, además, el refinamiento manual de los resultados.

Prophet puede considerarse un modelo de regresión no lineal de la forma

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

para la modelización de una serie temporal, *y(t)* como combinación lineal

- de una tendencia *g(t)*, no lineal por secciones,

- de un componente estacional *s(t)*, que incorpora las diferentes tendencias estacionales,

- y de *h(t)* que captura los efectos resultantes de las celebraciones festivas,

modelo al que se añade el correspondiente término de error: ϵ_t, *ruido blanco*.

Los puntos de cambio (*knots*) en la tendencia lineal son automáticamente establecidos, si no se han especificado previamente.

El componente estacional consiste en los elementos de la transormación de Fourier de los periodos relevantes.

Los efectos de las festividades son incorporados mediante variables *dummy*.

El modelo es estimado como un modelo bayesiano, para incorporar de manera automática los puntos de cambio y las otras características del modelo. El componente tendencial *g(t)* permite incorporar cambios no periódicos a largo plazo, *s(t)* modeliza el cambio periódico anual, mensual, semanal o diario, y $h(t)$ incorpora los efectos irregulares de más de un día de duración. Por último, el término de error ϵ_t, incorpora todo aquello que no puede ser atribuído a los componentes del modelo, y se supone que tiene distribución normal, con media cero y varianza constante.

Hay que destacar que el modelo no tiene en consideración la dependencia o asociación temporal entre las observaciones de la serie, como es el caso de los modelos *ARIMA(p,d,q)*, en el que las obsrvaciones futuras dependen de las pasadas. Se puede decir que *Prophet* ajusta una curva a la serie temporal, en lugar de buscar el *proceso subyacente* de la serie temporal. Aunque de esta manera hay una pérdida de información, sin embargo, este ajuste es muy flexible, al permitir acomodar múltiples periodos estacionales y cambios tendenciales. Este procedimiento es también robusto ante las observaciones aisladas y ante la ausencia de observaciones.

Al permitir la presencia de múltiples periodos estacionales, así como los efectos de las festividades, *Prophet* funciona mejor con series que presentan fuertes componentes estacionales.

Por último, señalar que *Prophet* está también operativo en los lenguajes de programación **R** y**Julia**.

Antes de entrar en los detalles del ejemplo que sigue, parece conveniente destacar las cuatro lineas de código que permiten la predicción con el programa *Prophet*. Cabe enunerarlas como sigue:

1.- incoar el modelo mediate: **modelo=Prophet()**

2.- estimar el modelo mediante: **modelo.fit(datos)**

3.- creación del *dataframe*, (la matriz de filas y columnnas), donde se alamcenarán las predicciones, mediante la instrucción:

Future=modelo.make_future_dataframe (periodos, frequencia)

4.- obtención de las predicciones mediante:

forecast=modelo.predict(future)

Cuatro líneas de código en conformidad con el objetivo de los creadores: facilidad y rapidez.

III. LA SERIE: TIPO DE INTERÉS DE LOS BONOS DEL TESORO

Esta serie recoge los tipos de interés de los Bonos del Tesoro, a largo plazo, diez años. Datos tomados de la Base de Datos del *Federal Reserve Bank*, con fecha: 21-09-23, que a su vez se remite a la OCDE: *Main Economic Indicators*, complete database. Son datos mensuales, desde enero 1980 hasta agosto de 2023, no ajustados estacionalmente. Un total de 524 observaciones.

Su representación gráfica viene recogida en la figura 1.

IV. ANÁLISIS Y PREDICIÓN CON PROPHET

Una vez en Python se cargan los necesarios paquetes, y se incorporan los datos

```
[1]: import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     from prophet import Prophet
```

Las primeras observaciones de la serie, junto con la fecha, aparecen en la siguiente tabla:

```
[2]: df                                                              =
     pd.read_csv("h:/articulo.anuario.2024/interes.Spain.csv")
     df.head()
```

```
[2]:          DATE  InteresBonos
0   1980.01.01   15.50
1   1980.02.01   15.40
2   1980.03.     15.72
3   1980.04.01   15.98
4   1980.05.01   15.90
```

Prophet requiere nombrar los nombres de las variables como **ds** e **y**.

```
[3]: df.columns=("ds","y")
```

hecho lo,cual, la serie aparece representada en la figura 1

```
[4]: ax = df.set_index("ds").plot(figsize=(10,6))
     ax.set_title("Interes Bonos a 10 años")
     ax.set_ylabel("Tipo de interés Bonos")
     ax.set_xlabel("Fechas")
     plt.show()
```

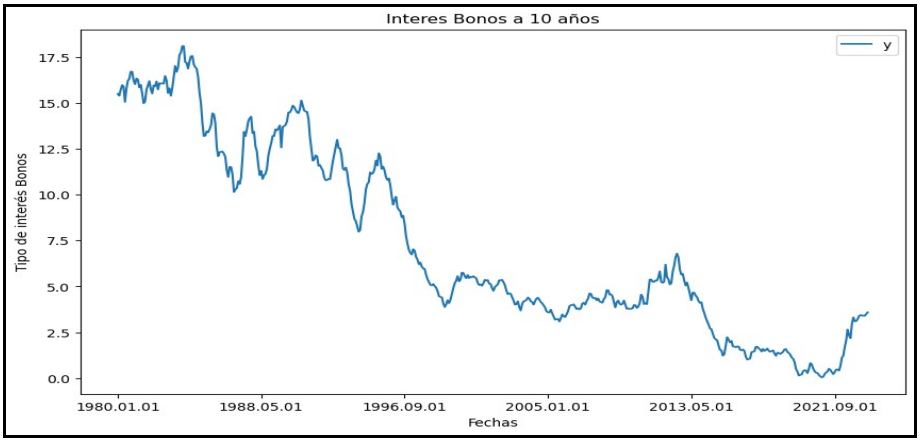

Figura 1. Tipo de interés de los Bonos del Tesoro a 10 años: 1980-1923.

El siguiente paso consiste en particionar la serie en dos subseries: *train* para estimar el modelo, y *test* para comprobar el ajuste de la predicción a los datos reales. Por tratarse de observaciones mensuales, se reserva un año, 12 observaciones, para la subserie *test*.

```
[5]: train=df[:-12]
     test=df[-12:]
```

A continuación, se invoca el programa Prophet creando el modelo **m**

```
[6]: m=Prophet()
     m.fit(df)
```

```
12:56:24 - cmdstanpy - INFO - Chain [1] start processing
12:56:24 - cmdstanpy - INFO - Chain [1] done processing
```

```
[6]: <prophet.forecaster.Prophet at 0x29f9bb8d720>
```

Para obtener las predicciones, es necesario generar un *dataframe*, la estructura matricial de filas y columnas, en donde se insertarán las predicciones. Sea su nombre **future**

```
[7]: future=m.make_future_dataframe(periods=12,freq="MS")
```

Y con la instrucción *forecast* se obtienen las predicciones: *yhat*. En la siguiente tabla aparecen las cinco últimas predicciones, para el año 2024, junto con los límites inferior y superior del 80% de confianza.

```
[8]: forecast = m.predict(future)
     forecast[["ds","yhat","yhat_lower","yhat_upper"]].tail()
```

```
[8]:        ds          yhat    yhat_lower    yhat_upper
531 2024-04-01    0.541956     -1.134817      2.270401
532 2024-05-01    0.578809     -1.009360      2.298981
533 2024-06-01    0.544641     -1.143312      2.294047
534 2024-07-01    0.539553     -1.166394      2.142955
535 2024-08-01    0.560029     -1.126411      2.243028
```

En la figura 2, se representa el resultado de todo lo hecho

```
[9]: m.plot(forecast)
     plt.show()
```

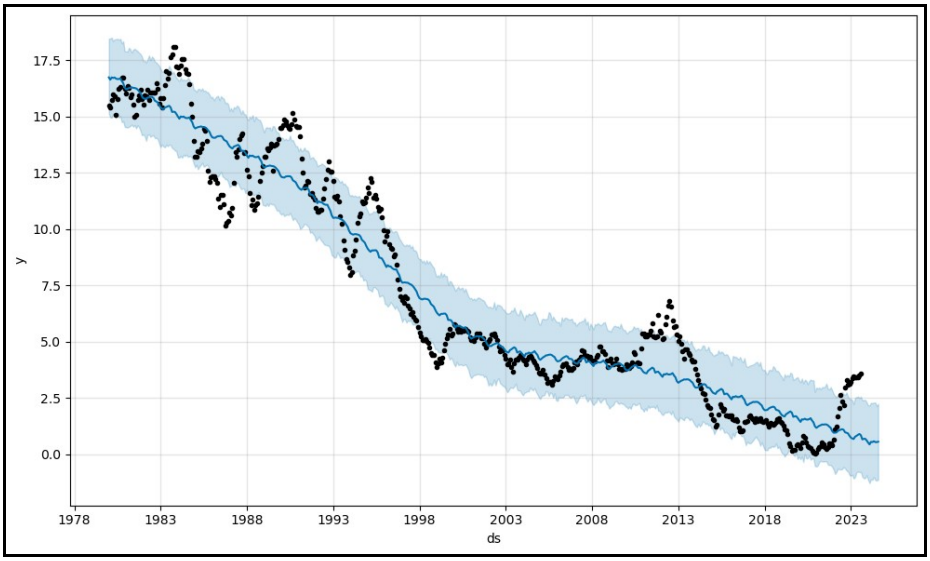

Figura 2. Serie *real* y serie *ajustada*.

La figura muestra la serie completa y el ajuste obtenido, junto con su banda de confianza.

Es posible también comparar las observaciones de la serie *test*, con los valores de la serie *yhat*. Para ello, se aislan las predicciones en la variable **pred**

```
[10]: pred = forecast["yhat"]
```

En la figura 3, se muestran representadas: la serie *train*, la serie *test* y las predicciones.

```
[11]: fig, ax=plt.subplots()
      ax.plot(train["y"], ls="-", label="Serie actual")
      ax.plot(test["y"], ls=":", label="Serie test")
      ax.plot(forecast["yhat"],                    ls="--",
      label="Prediccion")
      ax.set_xlabel("Tiempo")
      ax.legend(loc="best")
      plt.show()
```

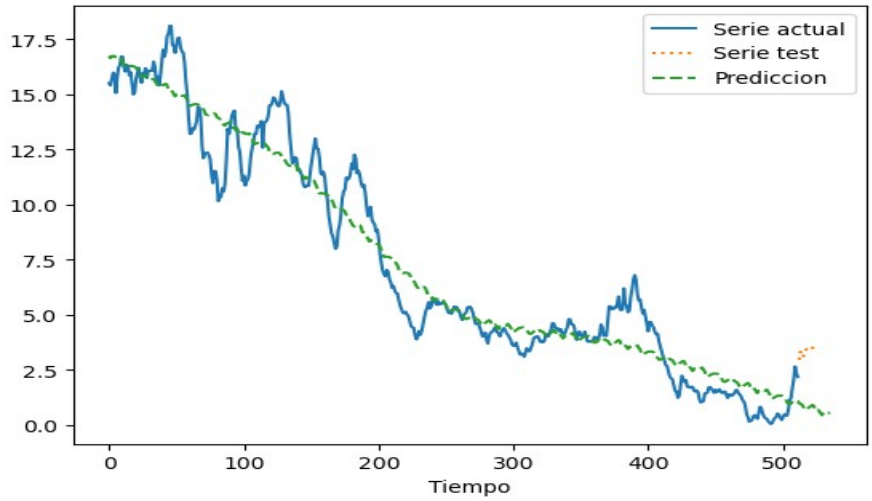

Figura 3. Representación de la serie *train*, de la serie *test* y de las predicciones *yhat*.

Se puede cuantificar la diferencia entre las observaciones de la serie *test* y las predicciones obtenidas, acudiendo al error cuadrático medio: *mse*, y al error absoluto medio: *mae*, mediante

```
[12]: from sklearn.metrics import mean_squared_error,
      mean_absolute_error
```

```
[13]: prophet_mse=mean_squared_error(test["y"], pred[-12:])
      print(prophet_mse)
```

```
      7.575803130896678
```

```
[14]: prophet_mae = mean_absolute_error(test["y"],
      pred[-12:])
      print(prophet_mae)
```

```
      2.743014171025363
```

No parece un ajuste perfecto.

Lo presentado hasta este momento constituye lo básico del programa *Prophet* en cuanto al tema concreto de la predicción: pocas instrucciones y muchos resultados. Pero *Prophet* posee otras habilidades, algunas de las cuales, se pueden agrupar como sigue:

1.- Representación gráfica de los componentes de un modelo.

2.- Representación gráfica de los puntos de cambio en la tendencia: *knots*.

3.- Funciones para la diagnósis del modelo:

a) cross-validation

b) estadísticos numéricos.

Se muestran algunos ejemplos.

1. Representación gráfica de los componentes de un modelo

En la figura 4 se tiene la representación gráfica de los componentes de nuestro modelo

```
[15]: fig3 = m.plot_components(forecast)
      plt.plot()
```

```
[15]:  []
```

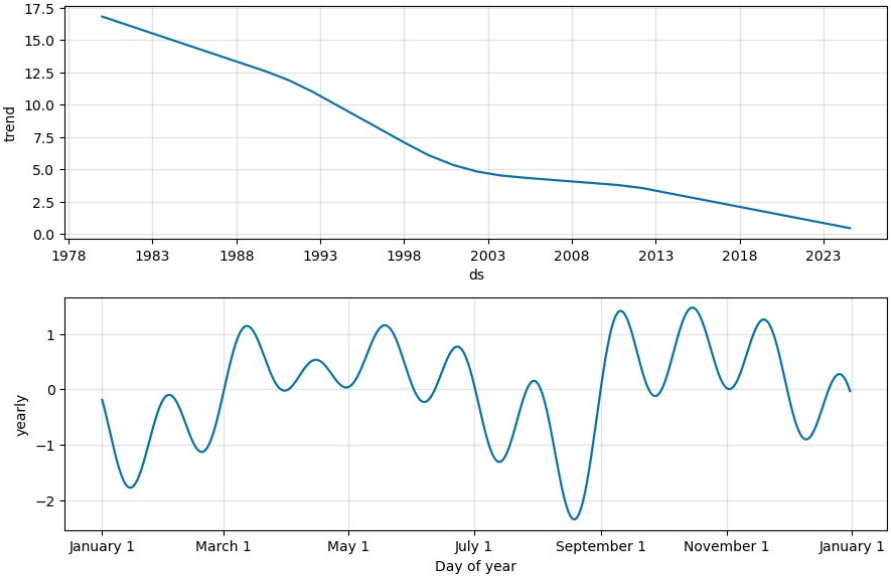

Figura 4. Componentes del modelo.

2. Representación de los puntos de cambio (knots) en la tendencia

En la figura 5, aparecen los puntos de cambio en la tendencia, registrados por *Prophet*, resultado de las instrucciones:

```
[16]: From prophet.plot import add_changepoints_to_plot
      fig4=m.plot(forecast)
      a=add_changepoints_to_plot(fig4.gca(), m,forecast)
```

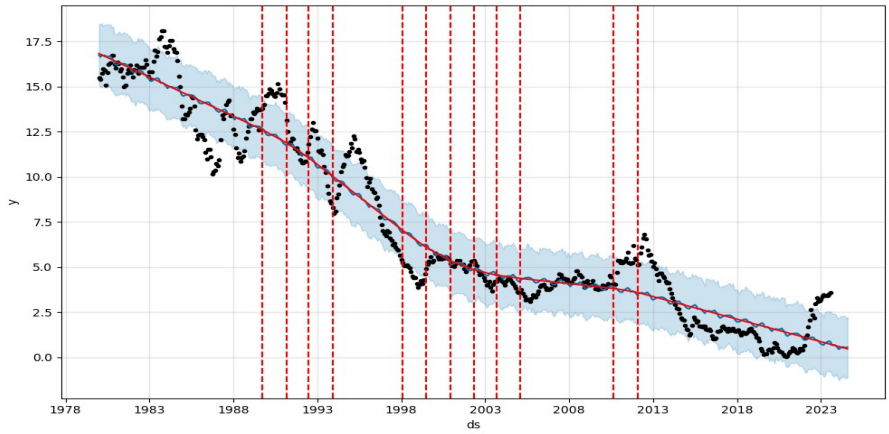

Figura 5. Puntos de cambio en la tendencia.

3. Funciones para la diagnósis de un modelo.

Prophet cuenta con una rica colección de procedimientos para efectuar la diagnosis de un modelo. Cabe mencionar, en primer lugar, la función *cross-validation*.

Dado que en la subserie *test* se incluyeron 12 observaciones, o sea un año, para la ejecución de la función *cross-validation* es necesario establecer un horizonte, aquí, 730 días, es decir dos años. Hay que señalar que el programa requiere mencionar **days** como unidad precisa de referencia.

```
[17]: from prophet.diagnostics import cross_validation
```

función que se ejecuta mediante:

```
[18]: df_cv = cross_validation(m,horizon = "730days")
```

tras unos segundos, se obtiene como resultado, el *dataframe* encabezado:

```
[19]: df_cv.head()
```

```
[19]:          ds       yhat  yhat_lower  yhat_uppe      y      cutoff
      0 1986-09-01  11.148138   10.406997   11.83785 11.10  1986-08-10
      1 1986-10-01  10.970327   10.268105   11.64725 10.15  1986-08-10
      2 1986-11-01  10.617641    9.931227   11.30543 10.29  1986-08-10
      3 1986-12-01   9.840255    9.102170   10.55015 10.36  1986-08-10
      4 1987-01-01   9.693486    9.009616   10.38258 10.74  1986-08-10
                                                    1
```

y finalizado como sigue:

```
[20]: df_cv.tail()
```

```
[20]:            ds      yhat  yhat_lower  yhat_upper       y      cutoff
      859 2023-04-01 -0.812002   -2.331095    0.626483 3.408167 2021-08-01
      860 2023-05-01 -0.781590   -2.389561    0.811556 3.413727 2021-08-01
      861 2023-06-01 -0.764208   -2.288564    0.692115 3.400000 2021-08-01
      862 2023-07-01 -0.747751   -2.418176    0.733902 3.500000 2021-08-01
      863 2023-08-01 -0.850077   -2.379917    0.742414 3.590000 2021-08-01
```

El resultado obtenido, es susceptible de ser representado, como se muestra en la figura 6.

```
[21]: fig, ax = plt.subplots() ax.plot(df_cv["yhat"],
      label ="yhat")
      ax.plot(df_cv["y"], "--", label="Serie actual")
      ax.set_ylabel("Interés Bonos")ax.set_xlabel("Tiempo")
      ax.legend(loc="best")
      plt.show()
```

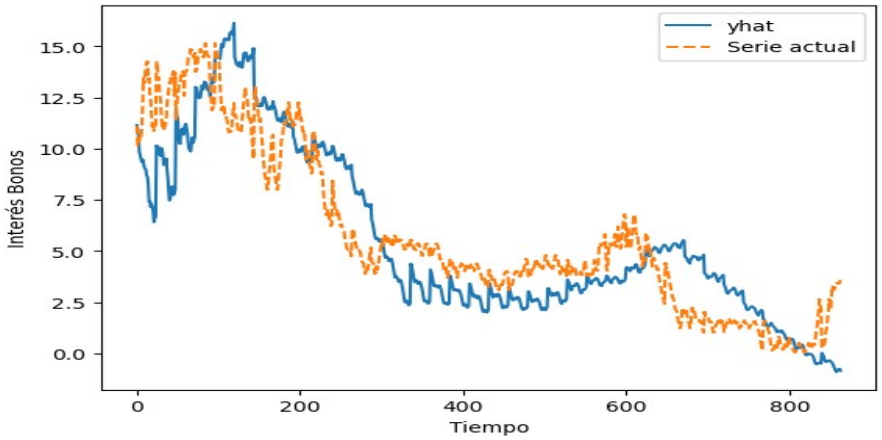

Figura 6. Resultado de la predicción tras aplicar *cross-validation*.

4. Estadísticos numéricos.

Prophet brinda también una gran variedad de estadísticos para la diagnósis del modelo. Todos ellos se presentan en la siguiente tabla:

```
[22]: From prophet.diagnostics import performance_metrics

[23]: df_perf = performance_metrics(df_cv)

[24]: df_perf.head()
```

```
[24]:  horizon   mse       rmse      mae       mape      mdape     smape    \
      0 86 days  2.392768  1.546858  1.179679  0.395701  0.199745  0.294309
      1 87 days  2.215225  1.488363  1.159671  0.401443  0.202448  0.301240
      2 88 days  2.140553  1.463063  1.147939  0.406445  0.203355  0.307837
      3 89 days  2.071673  1.439331  1.130477  0.405793  0.209211  0.309643
      4 90 days  2.119579  1.455877  1.143710  0.427694  0.211046  0.320700

                 coverage
              0  0.744186
              1  0.761628
              2  0.773256
              3  0.779070
              4  0.767442
```

resultados que es posible representar, en su evolución temporal. Por ejemplo, para el **mse**, tenemos representada su evolución en la figura 7.

```
[25]: from prophet.plot import plot_cross_validation_metric
```

```
[26]: fig=plot_cross_validation_metric(df_cv,metric="mse")
      plt.plot()
```

```
[26]: []
```

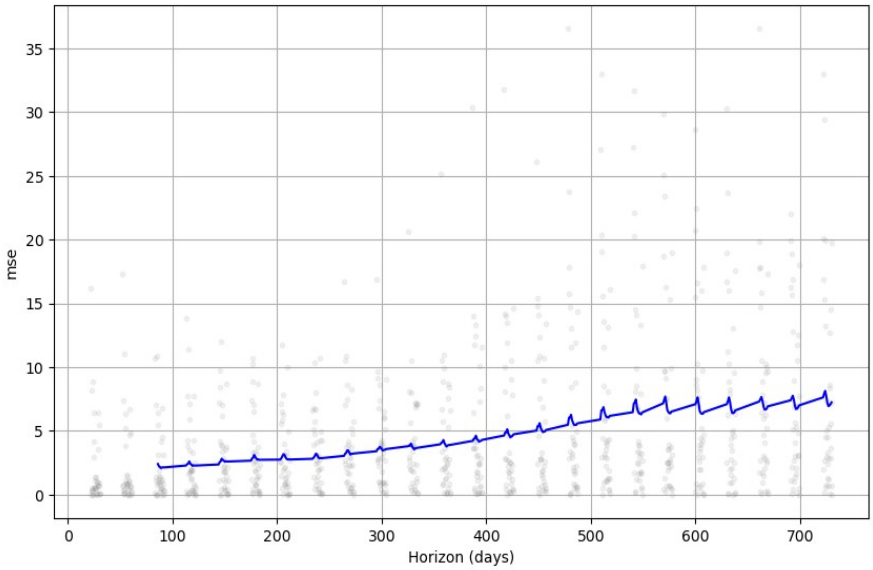

Figura 7. Evolución del error cuadrático medio: **mse**, a lo largo del horizonte establecido.

V. CONCLUSIÓN

Con el pretexto de analizar la serie histórica correspondiente al tipo de interés de los Bonos del Tesoro Español a diez años, se ha presentado un bosquejo de las posibilidades que brinda el programa *Prophet*, desarrollado por Meta (FaceBook) con el objetivo de obtener predicciones a *escala industrial* y de forma rápida y sencilla. Es éste un campo de creciente actualidad.

VI. BIBLIOGRAFÍA

- ATWAN, T. A., *Time Series Analysis with Python, Cookbook*, Packt Publishing, Birmingham- Mumbai, 2022.

- HYNDMAN, R., y ATHANASOPOULOS, G., *Forecasting, Principles and Practice*, tercera ed., O'Texts, 2021.

- JOSEPH, M., *Modern Time Series Forecasting with Python*, Packt Publishing, Birmingham- Mumbai, 2022.

- NIELSEN, A., *Practical Time Series Analysis, Prediction with Statistics and Machine Learning*, O'Reilly, Sebastopol, CA, 2019.

- PIXEIRO, M., *Time Series Forecasting in Python*, Manning Publications Co., Shelter Island, NY,2022.

- RAFFERTY, G., *Forecasting Time Series Data with Prophet*, Packt Publishing, Birmingham-Mumbai, 2023.

- SIMON, J., *Learn Amazon SageMaker*, segunda edición, Packt Publishing, Birmingham-Mumbai, 2021.

- TAYLOR Sean, J., y LETHAN, B., "Forcasting at Scale", in *The American Statistician*, 72(1) (2018) 37-45.

- VISHWAS, B.V., y PATEL, A., *Hands-on Time Series Analysis with Python, From Basics to Bleeding Edge Techniques*, Apress, Ahmedabad (India), 2020.