GRADO EN ECONOMÍA

Machine Learning aplicado a las CCSS

Predicción del PIB con Modelos de ML

10 de septiembre de 2023

Autor: Gustavo Eduardo Vargas Núñez

Profesor: Alberto Muñóz Cabanes

Índice

1. Introducción

El Producto Interno Bruto (PIB) per cápita es uno de los indicadores económicos más referenciados a nivel mundial. Al representar el valor promedio de la producción de bienes y servicios que corresponde a cada habitante de un país en un periodo determinado, este indicador nos proporciona una instantánea de la salud económica de una nación y su capacidad para proporcionar prosperidad a su población. Considerar el PIB per cápita es esencial, ya que no sólo refleja la magnitud de la economía en sí, sino también hasta qué punto los beneficios de esta economía están siendo compartidos entre los ciudadanos. Su estudio nos permite comprender las disparidades económicas entre naciones y regiones, y evaluar la eficacia de las políticas económicas. Asimismo, es una herramienta invaluable para analizar tendencias a lo largo del tiempo, identificar desafíos y proyectar escenarios futuros.

En el presente estudio[1], vamos a predecir el Producto Interno Bruto (PIB) per cápita mediante la aplicación de técnicas avanzadas de Machine Learning. En específico, se han seleccionado modelos basados en árboles, incluyendo Árboles de Decisión, Random Forest, XGBoost y Light GBM, con la intención de explorar y evaluar sus capacidades predictivas en contextos económicos. Además, la predicción se hará sobre las variaciones interanuales de PIB per cápita y no sobre los valores absolutos.

2. Datos

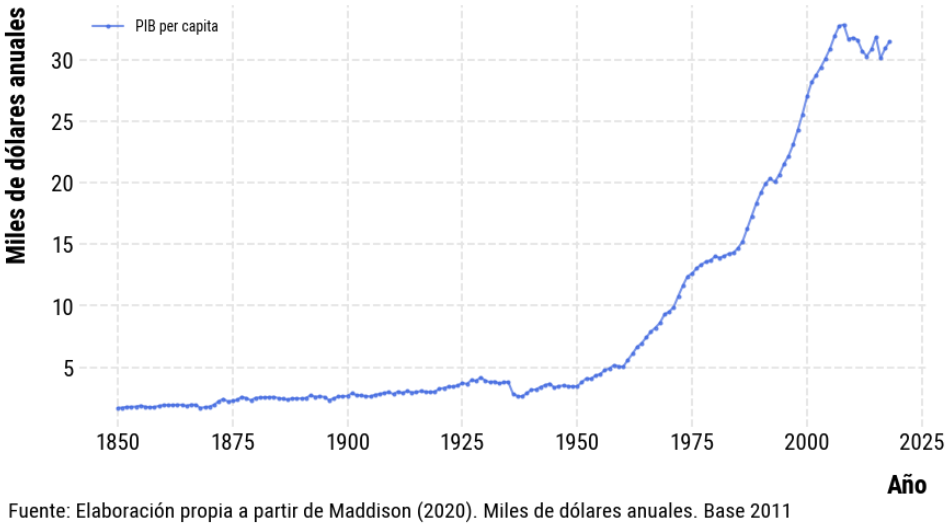
El PIB per cápita es una medida económica que se obtiene al dividir el Producto Interno Bruto (PIB) total de un país por su población total. Es utilizado frecuentemente como un indicador de la renta media de los habitantes de una nación y puede expresarse en términos nominales o reales.

En el contexto de esta investigación, se ha recurrido a la base de datos del proyecto Maddison, una fuente reconocida por su meticulosa recopilación y análisis de series temporales a largo plazo del PIB. En específico, los datos seleccionados hacen referencia al PIB real per cápita, y están ajustados a dólares constantes

[1] En este repositorio se puede ver el código, tanto de jupyter como el fichero latex, que se han usado para generar este documento

del año 2011. Esta elección garantiza una comparación homogénea y ajustada por inflación a lo largo del periodo de estudio, permitiendo así un análisis más preciso y fundamentado sobre las tendencias y comportamientos del PIB per cápita a nivel global. Usaremos los datos disponibles para España a partir de 1850 hasta 2018, como se puede ver en la Figura 1.

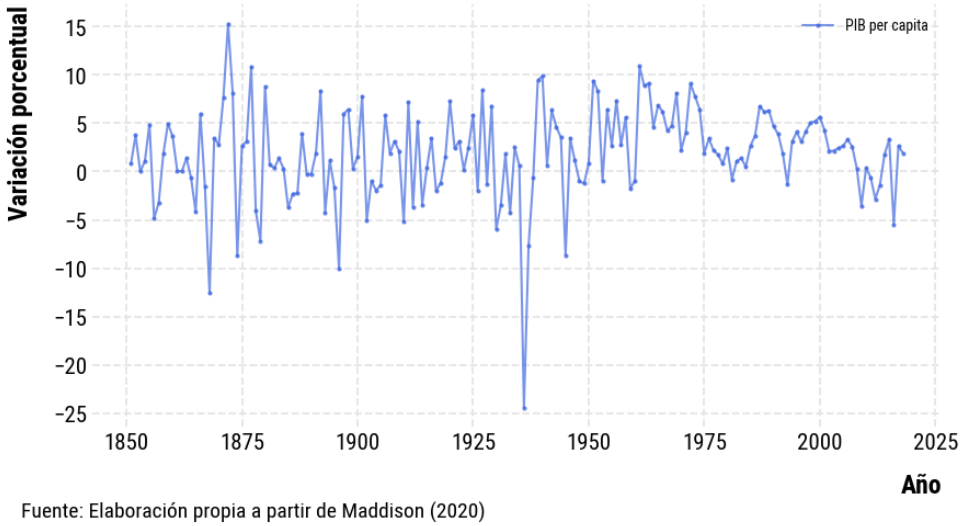
Figura 1: PIB per cápita de España



Fuente: Elaboración propia a partir de Maddison (2020). Miles de dólares anuales. Base 2011

De cara a los experimentos, utilizaremos los rendimientos interanuales correspondientes a los años de estudio. Se pueden ver en la Figura 2.

Figura 2: Variación anual del PIB per cápita de España



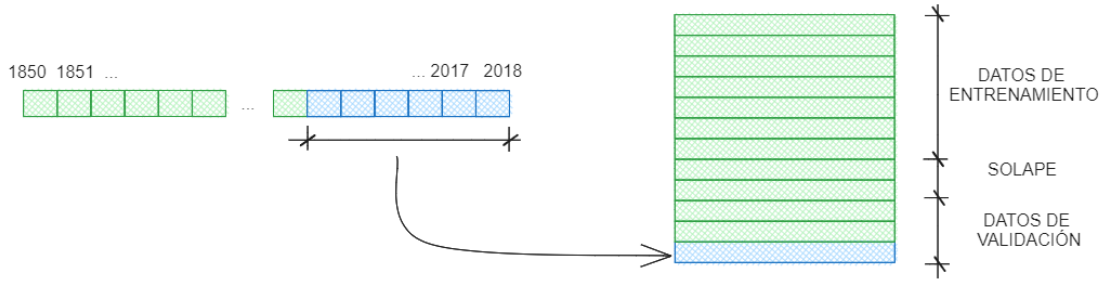
Fuente: Elaboración propia a partir de Maddison (2020)

3. Metodología

Se adopta un enfoque de procesamiento de datos orientado a la estructuración de un problema de regresión, de modo que necesitamos convertir la serie temporal en lo que se conoce como *data tabulares*. Con este propósito, se segmentarán los datos en ventanas temporales de 8 años. De cada ventana, se emplearán los datos correspondientes a los primeros 7 años para prever el valor del octavo año. Los algoritmos seleccionados para este estudio están diseñados para minimizar el error cuadrático.

Es esencial mencionar que basar la evaluación del modelo en el error calculado sobre los datos de entrenamiento puede ser inapropiado. Por ende, es necesario dividir el conjunto de datos en otros dos: *datos de entrenamiento* y *datos de test*. A fin de evitar solapamientos y garantizar la integridad de la información temporal, se eliminarán las 7 filas que se encuentran entre dichos dos subconjuntos. Este procedimiento y su lógica se ilustran en la Figura 3.

Figura 3: Esquema de tratamiento de datos



3.1. Algoritmos

Los algoritmos que vamos a utilizar son algoritmos basados en árboles. Se ha decidido hacerlo así por su mejor capacidad predictiva con pequeñas base de datos, además de una mejor interpretabilidad en el *porqué* de sus predicciones.

Para este estudio, se ha optado por emplear algoritmos basados en árboles. Esta elección responde a la eficacia demostrada de dichos algoritmos con conjuntos de datos de dimensiones reducidas, así como a la interpretabilidad con la que permiten entender las razones de sus predicciones mediante el método de *feature*

importance. Ésto contrasta con los algoritmos de *aprendizaje profundo*, donde la interpretabilidad es, a priori, nula.

3.1.1. Árboles de Decisión

Un Árbol de Decisión para regresión es una herramienta de aprendizaje supervisado que predice valores continuos en lugar de etiquetas de clase. La estructura de un árbol de decisión consta de nodos, ramas y hojas. Cada nodo representa una característica o atributo del conjunto de datos, cada rama representa una decisión basada en un valor de ese atributo y cada hoja es el resultado, que en el caso de la regresión es un valor numérico[2].

Algoritmo 1 Árbol de Decisión para Regresión

```
Procedimiento ÁRBOL REGRESIÓN(Datos, Criterio)
    si todos los ejemplos en Datos tienen el mismo valor entonces
        devolver valor promedio
    si Lista de Atributos está vacía o se cumple otro criterio de parada entonces
        devolver valor promedio de los valores en Datos
    MejorAtributo ← seleccionarMejorAtributo(Datos, Criterio)
    Árbol ← nuevo nodo con MejorAtributo
    para cada valor v posible de MejorAtributo hacer
        Datos_v ← ejemplos en Datos donde MejorAtributo = v
        si Datos_v está vacío entonces
            añadir una rama a Árbol con etiqueta v
            nodo hoja con valor promedio en Datos
        sino
            añadir rama a Árbol con etiqueta v
            ÁRBOL REGRESIÓN(Datos_v, Criterio)
    devolver Árbol
```

El proceso de construcción de un árbol de regresión implica seleccionar el atributo que proporciona la mejor división en función de un criterio determinado, generalmente la reducción de varianza. Una vez que se selecciona un atributo para un nodo, el conjunto de datos se divide en subconjuntos según los valores de ese atributo. Este proceso se repite de manera recursiva para cada subconjunto hasta

[2]Para más detalles del algoritmo, se puede ver [2]

que se alcanza un criterio de parada, como una profundidad máxima del árbol o un número mínimo de puntos de datos en un nodo.

3.1.2. Random Forest

Random Forest es un algoritmo que combina múltiples árboles de decisión para construir un modelo más robusto y preciso. Se basa en dos técnicas principales: bagging y feature randomness.

- Bagging: Para cada árbol del bosque, se toma una muestra con reemplazo (conocida como muestra bootstrap) del conjunto de datos original. Esto significa que algunos ejemplos pueden ser seleccionados más de una vez y otros no ser seleccionados en absoluto para un árbol en particular.

- Feature randomness: Al construir cada árbol, en lugar de buscar la mejor característica para dividir en cada nodo, se selecciona un subconjunto aleatorio de características y luego se busca la mejor característica para dividir solo dentro de ese subconjunto.

Una vez entrenado el modelo, para realizar una predicción, se ejecuta el ejemplo a través de todos los árboles en el bosque y se toma el promedio de sus predicciones para obtener la predicción final del modelo[3].

Algoritmo 2 Random Forest para Regresión

```
Procedimiento RandomForestRegresión(Datos, n_árboles, n_características)
    Inicializar lista vacía Árboles
    para i = 1 hasta n_árboles hacer
        Muestra_bootstrap ← generarMuestraBootstrap(Datos)
        Árbol ← construirÁrbolDecisión(Muestra_bootstrap, n_características)
        Añadir Árbol a Árboles
    Procedimiento predecir(ejemplo)
        predicciones ← [Árbol.predecir(ejemplo) para Árbol en Árboles]
        devolver promedio(predicciones)
```

[3]Para más detalles del algoritmo, se puede ver [3]

3.1.3. XGBoost

XGBoost, que proviene de 'Extreme Gradient Boosting', es un algoritmo de aprendizaje supervisado que utiliza un enfoque de boosting para construir un conjunto de árboles de decisión. El objetivo principal de XGBoost es optimizar un conjunto de modelos débiles, generalmente árboles de decisión, de manera aditiva y secuencial[4]. Se caracteriza por:

Algoritmo 3 XGBoost para Regresión

Procedimiento XGBoostRegresión(Datos, n_árboles, learning_rate)
  Inicializar modelo con predicciones constantes (p. ej., promedio de los objetivos)
  para $i = 1$ hasta n_árboles **hacer**
    Calcular gradientes y hessianos de la función de pérdida
    Construir árbol usando gradientes y hessianos
    Actualizar modelo sumando learning_rate $\times$ predicciones del nuevo árbol
  Procedimiento predecir(ejemplo)
    Sumar predicciones de todos los árboles para el ejemplo
    devolver Predicción final

- Boosting: Construye árboles de manera secuencial, donde cada árbol corrige los errores del anterior. A diferencia de otros algoritmos de boosting, XGBoost también tiene en cuenta un término de regularización en su función objetivo, lo que ayuda a evitar el sobreajuste.

- Descenso de Gradiente: A medida que construye cada árbol, XGBoost utiliza el descenso de gradiente para minimizar el error entre las predicciones actuales y los valores reales. Específicamente, calcula el gradiente (primera derivada) y el hessiano (segunda derivada) de la función de pérdida con respecto a las predicciones del modelo anterior para guiar la construcción del siguiente árbol.

- Pruning: A diferencia de la mayoría de los algoritmos de árboles que utilizan una estrategia de 'pruning' basada en la profundidad (limitan la profundidad máxima de un árbol), XGBoost utiliza 'depth-first pruning'. Es decir, permite que un árbol crezca en profundidad hasta que cumpla con ciertos criterios,

[4]Para más detalles del algoritmo, se puede ver [4]

y luego poda el árbol retrocediendo y eliminando ramas que aportan poco valor.

3.1.4. LightGBM

LightGBM, acrónimo de 'Light Gradient Boosting Machine', es un framework de boosting optimizado para conjuntos de datos grandes y altamente dimensional[5]. Tiene las siguientes características:

Algoritmo 4 LightGBM para Regresión

```
1: Procedimiento LightGBMRegresión(Datos, n_árboles, learning_rate)
2:     Inicializar modelo con predicciones constantes
3:     para i = 1 hasta n_árboles hacer
4:         Calcular gradientes de la función de pérdida
5:         Construir árbol utilizando estrategia de crecimiento basada en hojas
6:         Actualizar modelo sumando learning_rate × predicciones del nuevo árbol
7:     Procedimiento predecir(ejemplo)
8:         Sumar predicciones de todos los árboles para el ejemplo
9:         devolver Predicción final
```

- Crecimiento basado en hojas: Mientras que los enfoques tradicionales crecen el árbol de manera uniforme, LightGBM divide la hoja que aporta la máxima reducción en la función de pérdida, permitiendo un aprendizaje más dirigido.
- Optimización para Conjuntos de Datos Grandes: LightGBM utiliza histogramas para agrupar características continuas en intervalos discretos. Estos histogramas reducen significativamente el número de divisiones a considerar y, por lo tanto, aceleran el proceso de formación del árbol.
- Uso Eficiente de la Memoria: Las optimizaciones de LightGBM, como la creación de histogramas, permiten que el algoritmo utilice menos memoria en comparación con otros métodos de boosting.
- Paralelización y Escalabilidad: Utiliza técnicas como la construcción de árboles basada en características para paralelizar el entrenamiento, lo que mejora la eficiencia en conjuntos de datos con un alto número de características.

[5]Para más detalles del algoritmo, se puede ver [5]

- Soporte para Valores Nulos: Durante el proceso de construcción del árbol, determina la dirección óptima para las observaciones con valores nulos, ya sea a la división izquierda o derecha.

3.2. Experimentación

3.2.1. Cross Validation

Cross Validation, o validación cruzada, es una técnica de evaluación de modelos en el aprendizaje automático que busca estimar el rendimiento de un modelo en datos no vistos. La idea central es dividir el conjunto de datos original en "k"subconjuntos (o "folds"). En cada iteración, se utiliza un subconjunto como datos de validación y los "k-1"subconjuntos restantes como datos de entrenamiento.

El proceso se repite "k"veces, rotando el subconjunto de validación en cada iteración. Al final de las "kïteraciones, se obtiene una serie de métricas de rendimiento, como el error, para cada fold. Estas métricas se promedian para obtener una estimación más robusta y menos sesgada del rendimiento del modelo en comparación con usar una única división de entrenamiento y validación.

3.2.2. Grid Search

Grid Search es una técnica utilizada en el aprendizaje automático para optimizar hiperparámetros. Funciona definiendo un conjunto de valores posibles para cada hiperparámetro y luego evaluando todas las posibles combinaciones de estos valores. Para cada combinación, se entrena un modelo sobre un conjunto de datos de entrenamiento y se valida su rendimiento en un conjunto separado.

Después de examinar todas las combinaciones, se selecciona la que haya producido el mejor rendimiento en el conjunto de validación. Si bien Grid Search es exhaustivo y puede identificar una combinación óptima, tiene un alto coste computacional, especialmente cuando el espacio de búsqueda de hiperparámetros es grande o el modelo es computacionalmente intensivo.

4. Resultados

Podemos ver los resultados de los experimentos en el Cuadro 1. Los modelos de Machine Learning intentan aproximarse a la distribución de los datos con la que han sido entrenados, de ahí que sea necesario tener datos de test para poder tener una métrica significa del rendimiento de nuestro modelo.

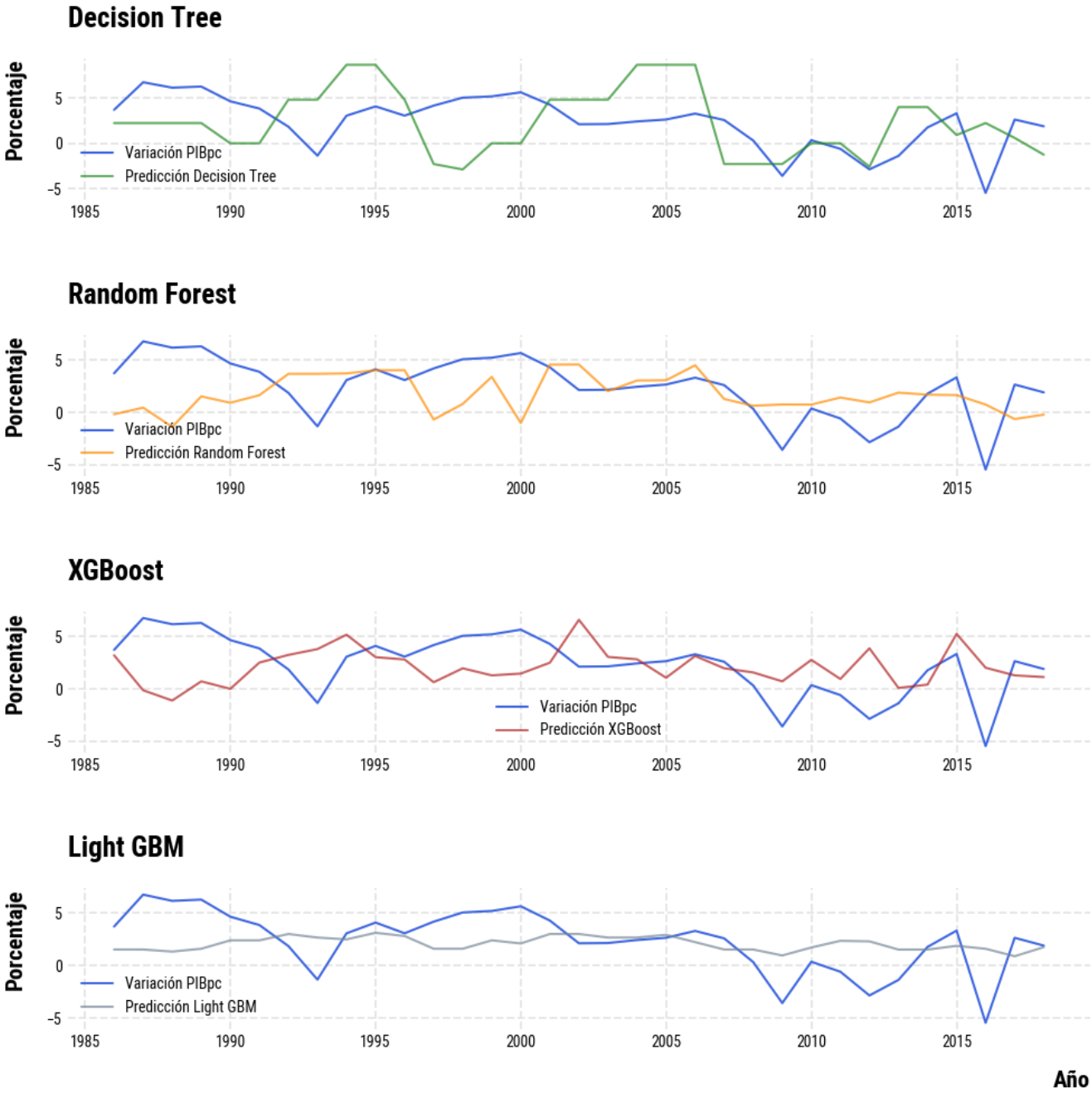
Como vemos, tanto el modelo de Árboles de Decisión como XGBoost tienen un error en entrenamiento menor que en test, lo que es normal. Lo significativo pasa en los otros dos modelos, Random Forest y Light GBM, donde el error en test es menor que en entrenamiento, especialmente en el caso del Light GBM.

Cuadro 1: Resultados de los Experimentos

	Error Cuadrático Medio	
	Train	**Test**
Decision Tree	0.00122	0.00189
Random Forest	0.00153	0.00118
XGBoost	0.00023	0.00124
Light GBM	0.00216	0.00082

La Figura 4 nos permite explicar el porqué de ello. Ahí vemos la variación del PIB per cápita en azul en test y la comparamos con las distintas predicciones. En el caso de Light GBM, la predicción no varía mucho, dando una curva relativamente plana. Ésto, sin embargo, es lo que le permite tener un error relativamente pequeño, ya que en el resto de casos las predicciones se separan mucho más de los valores reales.

Figura 4: Comparación de Predicciones



5. Conclusiones

Predecir una serie temporal nunca es tarea fácil. En nuestro caso, el problema se ha visto agravado por la escasez de datos históricos que tenemos para la variable que nos interesa, el PIB per cápita. Sin embargo, hemos visto que estas predicciones son posibles. Hemos probado cuatro modelos, siendo el Light GBM con el que hemos obtenido las predicciones con menos error.

Posibles ampliaciones del trabajo tendrían en cuenta las siguientes líneas:

- El entrenamiento podría hacerse con las series de todos los países y ver si eso ayuda a mejorar la predicción. Se podría intentar, al menos, buscar un subconjunto de países con comportamiento histórico similar y ceñir el entrenamiento a ellos.

- Se debería hacer una comparación con modelos autorregresivos, especialmente los tipo GARCH, para poder comparar el grado de error.

- Parsear los datos de la forma en la que lo hemos hecho convierte el problema en uno de regresión sobre datos tabulares. Esto permite añadir nuevas *features*, con la única limitación de que dichas features deberían estar disponibles para el período estudiado.

Referencias

[1] Bolt, Jutta and J.L. Van Zanden, (2020). *Maddison style estimates of the evolution of the world economy. A new 2020 update.* Maddison Project Database.

[2] M.A. Hambali, Y.K. Saheed, T.O. Oladele, M. D. Gbolagade. *ADABOOST Ensemble Algorithms for Breast Cancer Classification.* Journal of Advances in Computer Research. Vol. 10. No. 2, May 2019. Pages: 31-52

[3] Hongquan Guo, Hoang Nguyen, Diep-Anh Vu, Xuan-Nam Bui. *Forecasting mining capital cost for open-pit mining project based on artificial neural network approach.* Resources Policy. Vol. 74, December 2021, 101474.

[4] Chao Qin, Yunfeng Zhang, Fangxun Bao, Caiming Zhang, Peide Liu, Peipei Liu. *XGBoost Optimized by Adaptative Particle Swarm Optimization for Credit Scoring.* Mathematical Problems in Engineering. Vol. 2021. Article ID 6655510.

[5] N. Dunbray, R. Rane, S. Nimje, J. Katade and S. Mavale. *A Novel Prediction Model for Diabetes Detection Using Gridsearch and A Voting Classifier between Lightgbm and KNN.* IEEE. 2021 2nd Global Conference for Advancement in Technology (GCAT), Bangalore, India, 2021, pp. 1-7