Islas Canarias
Del 15 al 19 de noviembre de 2021

Un análisis de modelos de pronóstico para COVID-19 en Cantabria

Lezcano Lastra, Alberto
Instituto Cántabro de Estadística
lezcano_a@cantabria.es

Llamosas García, Gonzalo
Instituto Cántabro de Estadística
llamosas_g@cantabria.es

López Cagigas, Alejandro
Instituto Cántabro de Estadística
lopez_ale@cantabria.es

Parra Rodríguez, Francisco Javier
Instituto Cántabro de Estadística
parra_fj@cantabria.es

Resumen:

La pandemia del COVID-19 ha supuesto un grave reto sanitario, económico y social en todo el mundo. Para hacerlo frente, gobiernos de todo el planeta han planteado restricciones de movilidad que han agravado considerablemente la calidad de vida de millones de personas. En este contexto, los modelos estadísticos de series temporales han vuelto a ocupar la primera línea. El uso de pronósticos de aprendizaje automático se ha mostrado eficaz para anticipar la evolución de los diferentes shocks epidemiológicos. A partir de datos del Servicio Cántabro de salud, hemos realizado proyecciones de casos diarios a 30 días. Este trabajo compara diferentes metodologías de series temporales para analizar su rendimiento en las predicciones de casos positivos a través de diferentes métricas. También se describen las tecnologías empleadas en el aplicativo web del Instituto Cántabro de Estadística, donde se muestra la experiencia diaria en la recogida, procesamiento y actualización de datos para el diseño de indicadores socio-sanitarios.

1. Introducción

El último gran proceso pandémico, que comenzó el 31 de diciembre de 2019 en la ciudad china de Wuhan, supuso la identificación de un nuevo tipo de coronavirus (2019-nCoV) asociado a casos de neumonía severa. Este virus se propagó rápidamente y se convirtió en un gran problema mundial. La enfermedad provocada por este virus recibió el nombre de COVID-19, a su vez que el virus en concreto fue denominado SARS-CoV-2 debido a sus similitudes con el SARS CoV-1 (Rismanbaf, 2020).

Los ratios de contagio y transmisión del virus que da lugar al COVID-19 son de los más elevados, en comparación con otros procesos virales existentes. Su rápido ritmo de contagios hizo posible que el virus pudiese extenderse por todo el mundo en tiempo récord. De manera similar a otros brotes epidémicos originados por enfermedades infecciones, el éxito de controlar la expansión del virus se basa en recoger información significativa sobre la evolución de la epidemia. Para ello, es necesario llevar a cabo un seguimiento correcto de los casos, así como aumentar la fiabilidad de las predicciones futuras con cada nueva fuente de datos (Chakraborty and Ghosh, 2020)

En la literatura académica existen diversos estudios centrados en pronósticos de enfermedades epidémicas. Algunas de las metodologías más empleadas en la predicción de series estadísticas para enfermedades infecciones son los modelos epidemiológicos (SIR), que asumen que la dinámica de transmisión es más rápida que la demográfica; el modelo autorregresivo integrado de media móvil (ARIMA), que ha sido usado para el pronóstico de enfermedades como la fiebre hemorrágica (Liu et al., 2011), brucelosis (Cao et al., 2020), gripe (He et al., 2018) y COVID-19 (Ceylan, 2020); y la regresión local que combina la sencillez de la regresión lineal por mínimos cuadrados con la flexibilidad de la regresión no lineal.

Otros modelos que recientemente han ganado gran importancia entre la literatura académica por su buen comportamiento predictor de casos COVID-19 (Kwekha-Rashid et al., 2021; Sujath et al. 2020) son los basados en aprendizaje automático (machine learning), que combinan una fase de aprendizaje (training), con otra fase de prueba (testing). Los modelos más utilizados por la bibliografía académica son las redes neuronales, los bosques aleatorios y el algoritmo de K-vecinos próximos.

A partir de datos diarios de casos positivos por PCR del Servicio Cántabro de Salud, este documento de trabajo compara los modelos utilizados para el pronóstico de casos activos a 30 días utilizando diferentes métricas de rendimiento.

2. Metodología

2.1 Datos

Para la realización de esta comparativa de modelos se utilizan datos de la plataforma "Información Coronavirus" del Servicio Cántabro de Salud (scsalud.es/coronavirus). Esta fuente de información se actualiza diariamente incluyendo variables como el número de casos diarios, el acumulado de casos a 7 y 14 días, el número de hospitalizados y el total de ingresos por hospitales dentro de la región. La recogida de datos tuvo lugar el día del primer caso positivo por SARS-CoV-2 en Cantabria y se extiende hasta el periodo actual.

2.2 Modelos

Modelo ARIMA

El modelo autorregresivo integrado de media móvil (ARIMA) es una de las metodologías más empleadas en investigación de series temporales (Aditya et al. 2021, Roy et al., 2021). La sección AR del modelo ARIMA expresa que la variable que evoluciona se hace una regresión sobre sus propios valores anteriores. En una serie temporal estacionaria, la media del término de error es cero y la varianza se expresa como δ^2. Si Y_t muestra el valor de la serie temporal en el tiempo t, la expresión de esta serie temporal como un proceso autorregresivo de orden p es como en la ecuación (1) y se muestra como AR(p).

$$Yt = \delta + \varphi_1 Y_{t-1} + \varphi_2 Y_{t-2} + \cdots + \varphi_p Y_{t-p} + \varepsilon_t \quad (1)$$

En esta ecuación, δ es una constante y ε_t es el término de error. Las series temporales como un proceso de media móvil de "q" se puede expresar de la siguiente forma:

$$Yt = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} \quad (2)$$

La expresión de ARMA (p, q) puede obtenerse combinando las dos expresiones: AR(p) y MA(q):

$$Yt = \delta + \varphi_1 Y_{t-1} + \cdots + \varphi_p Y_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} \quad (3)$$

Si la serie temporal procesada no es estacionaria, puede hacerse estacionaria tomando el proceso de diferencia "d" veces. Una vez tomada la diferencia de la serie Y_t no estacionaria, la serie ΔY, que expresa la característica estacionaria puede ser calculada por la ecuación 4.

$$\Delta Y_t = Y_t - Y_{t-1} = Y_t - LY_t = Y'_t \quad (4)$$

El proceso ARIMA (p, d, q) se puede encontrar generalmente utilizando la ecuación 5.

$$\left(1 - \varphi_1 L - \varphi_1 L^2 - \cdots - \varphi_p L^q\right)\Delta^d Y_t = \delta + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} \quad (5)$$

La autocorrelación parcial (PACF) puede ser utilizada para encontrar el valor del parámetro AR y los gráficos del correlograma de las funciones de autocorrelación pueden ser utilizados para lograr el valor del MA. Para obtener el parámetro más adecuado en el

enfoque ARIMA, el rendimiento del modelo suele medirse mediante la expresión del criterio de información de Akaike (AIC). El AIC se puede calcular de la siguiente manera:

$$AIC = -2\log(L) + 2(p + q + k) \quad (6)$$

En esta ecuación, L es la probabilidad de los datos, p es el orden de la fracción autorregresiva, q es el orden de la fracción de media móvil y k es el intercepto del modelo ARIMA. Según este parámetro, el modelo con el criterio AIC más bajo se considera más acertado que los demás. En este estudio, los parámetros que mostraron el mayor rendimiento fueron los del modelo ARIMA (2,2,5).

Regresión local

La regresión local (local regression, LOESS) utiliza un enfoque distinto para el ajuste de curvas, que supone calcular el ajuste en cada punto X_0 usando solamente las observaciones de entrenamiento próximas.

El algoritmo se describe a continuación:

1. Obtener la fracción de datos de entrenamiento $S = \frac{k}{n}$ cuyos X_i están más próximos a X_0.

2. Asignar una función de peso $K_{i0} = K(X_{i,} X_0)$ a cada punto, de forma que el punto más alejado de X_0 posea ponderación 0, y el más cercano disponga del mayor peso. Como excepción de estos K vecinos más cercanos, el resto tienen ponderación 0.

3. Ajustar una regresión ponderada por mínimos cuadrados de Y_1 sobre X_i usando pesos citados anteriormente, obteniendo estimadores de $\beta_0\ y\ \beta_1$ que minimicen la siguiente ecuación:

$$\sum_{i=1}^{n} K_{i0}\,(y_i - \beta_0 - \beta_1 X_i)^2 \quad (8)$$

4. El valor ajustado en X_0 está dado por:

$$\hat{f}(X_0) = \widehat{\beta_0} + \widehat{\beta_1} X_0 \quad (9)$$

La regresión local puede generalizarse a un escenario con múltiples predictores, sin embargo, resulta ineficiente en casos con más de 3 o 4 predictores por la falta de observaciones cercanas a X_0.

Un factor que hay que tener en cuenta de la regresión local es la baja interpretabilidad de los datos.

Red neuronal

Una red neuronal es un algoritmo de clasificación biológica. Consiste en un número (usualmente grande) de unidades de procesamiento similares a las neuronas, organizadas por capas. Cada unidad es una capa que está conectada con todas las unidades de la capa anterior. Estas conexiones no son todas iguales: cada conexión puede tener una fuerza o peso diferente. Los pesos o ponderaciones de estas conexiones codifican el conocimiento de una red, cuyas unidades se denominan nodos.

Los datos entran por las capas de entrada y cruzan la red, capa por capa, hasta llegar a las capas de salida. Durante el funcionamiento normal, es decir, cuando actúa como clasificador, no hay retroalimentación entre las capas. Por eso este tipo de redes se denominan redes neuronales retroalimentadas (feed-forward).

Las redes neuronales retroalimentadas se utilizan principalmente para el aprendizaje supervisado en los casos que los datos que deben aprenderse no son secuenciales ni dependen del tiempo. Es decir, las redes neuronales retroalimentadas calculan una función F sobre una entrada de tamaño fijo X tal que $f(x) \approx y$ para los pares de entrenamiento (x, y). Por otro lado, las redes neuronales recurrentes aprenden datos secuenciales, calculando g sobre una entrada de longitud variable $Xk = \{x1, \dots, xk\}$, de manera que $g(X_k) \approx y_k$ para los pares de entrenamiento (X_n, Y_n) para todo $1 \leq k \leq n$.

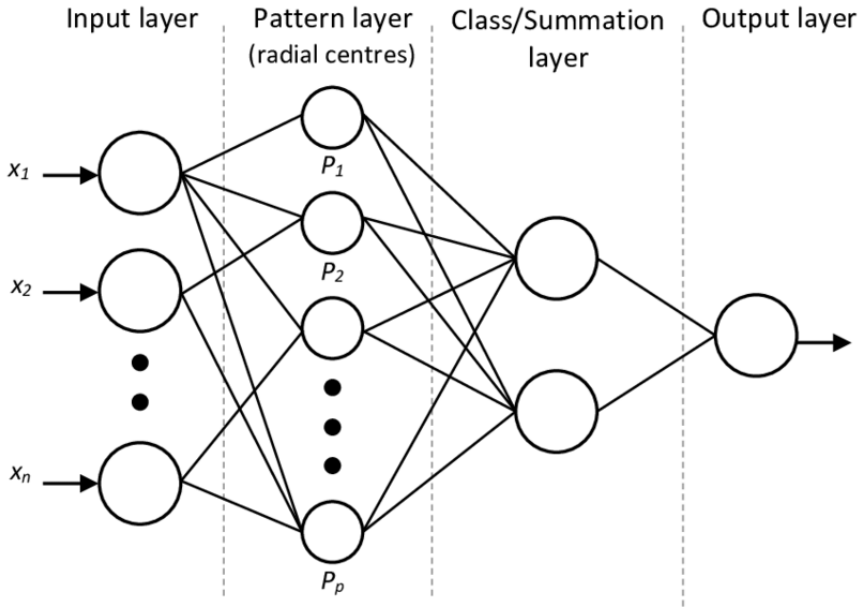


Figura 1. Ilustración gráfica de una red neuronal artificial

Para ejecutar la red neuronal a través de R Studio, se emplea una librería de alto nivel de Python denominada Keras, así como una implementación de Google, conocida como Tensorflow, que actúa como motor de la red neuronal implementada en el comparativo de modelos.

Bosque aleatorio

El bosque aleatorio es un algoritmo de aprendizaje automático supervisado. Es uno de los algoritmos más utilizados debido a su precisión, simplicidad y flexibilidad. El hecho de que pueda utilizarse para tareas de clasificación y regresión, combinado con su naturaleza no lineal, lo hace muy adaptable a una serie de datos y situaciones.

El término "bosque de decisión aleatorio" fue propuesto por primera vez en 1995 por Tin Kam Ho (1995), quien desarrolló una fórmula para utilizar datos aleatorios para crear predicciones.

Se denomina bosque porque se organiza sobre un conjunto de árboles de decisión. Los datos de estos árboles se fusionan para garantizar las predicciones más precisas. Mientras que un árbol de decisión en solitario tiene un resultado y un rango estrecho de grupos, el bosque asegura un resultado más precios con un mayor número de grupos y decisiones. Tiene la ventaja añadida de incorporar aleatoriedad al modelo al encontrar la mejor característica entre un subconjunto aleatorio de datos.

El algoritmo de entrenamiento para los bosques aleatorios aplica la técnica genera de agregación de bootstrap (bagging). Dado un conjunto de entrenamiento $X = x_1, \dots, x_n$ con respuestas $Y = y_{1,} \dots, y_n$, la técnica de bagging selecciona repetidamente una muestra aleatoria con reemplazo (B veces) del conjunto de entrenamiento y ajusta los árboles a estas muestras:

Para b = 1, …, B:

1. Muestra, con reemplazo, n ejemplos de entrenamiento de X, Y; llamados Xb, Yb.
2. Entrenar un árbol de clasificación o regresión fb on Xb, Yb.

Después de la fase de entrenamiento, las predicciones para las muestras no vistas en x' pueden hacerse promediando las predicciones de todos los árboles de regresión individuales sobre x'.

$$\hat{f} = \frac{1}{B} \sum_{b=1}^{B} f_b\,(x') \qquad (10)$$

O tomando el voto mayoritario en el caso de los árboles de clasificación.

Este procedimiento de bootstrapping conduce a un mejor rendimiento del modelo, ya que disminuye su varianza sin aumentar el sesgo. Esto significa que mientras las predicciones de un solo árbol son muy sensibles al ruido en su conjunto de entrenamiento, la media de muchos árboles no lo es, siempre que los árboles no estén correlacionados. El simple hecho de entrenar muchos árboles en un único conjunto de entrenamiento daría lugar a árboles muy correlacionados (o incluso al mismo árbol muchas veces, si el algoritmo de entrenamiento es determinista); el muestreo bootstrap es una forma de descorrelacionar los árboles mostrándoles diferentes conjuntos de entrenamiento.

Adicionalmente, se puede hacer una estimación de la incertidumbre de la predicción como la desviación estándar de las predicciones de todos los árboles de regresión individuales en x'.

$$\sigma = \sqrt{\frac{\sum_{b=1}^{B}(f_b(x') - \hat{f})^2}{B-1}} \qquad (11)$$

El número de muestras/árboles, B, es un parámetro libre. Normalmente, se utilizan de unos cientos a varios miles de árboles, dependiendo del tamaño y la naturaleza del conjunto de entrenamiento. Se puede encontrar un número óptimo de arboles B utilizando la validación cruzada, o bien observando el error fuera de bolsa (out-of-bag error): el error medio de predicción en cada muestra de entrenamiento Xi en su muestra bootstrap. El error de entrenamiento y de prueba tiende a nivelarse después de que se haya ajustado un cierto número de árboles.

K vecinos más próximos

El algoritmo de K vecinos más próximos (K-NN, por sus siglas en inglés), propuesto por Evelyn Fix y Joseph Hodges (1951) y posteriormente ampliado por Thomas Cover (1967), consiste en un enfoque de clasificación no paramétrico que se utiliza tanto para la clasificación como para la regresión. En ambos casos, los inputs consisten en una serie de entrenamientos empleando la metodología de K-NN en un conjunto de datos. Los outputs dependen de si K-NN se emplea para la clasificación o para la regresión.

En la clasificación de K-NN, el output es una pertenencia a una clase. Un objeto se clasifica mediante un voto plural de sus vecinos, asignando el objeto a la clase más común entre sus K vecinos más cercanos (donde K es un número entero positivo, normalmente de tamaño pequeño). Si K = 1, el objeto se asigna a la categoría del único vecino más cercano.

En la regresión mediante K-NN, el output es el valor de la propiedad del objeto. Este valor es la media de los valores de los K vecinos más próximos.

K-NN es por tanto un clasificador en el que la función sólo se aproxima localmente y todo el cálculo se aplaza hasta la evaluación. Dado que este algoritmo se basa en la distancia para la clasificación, si las características representan diferentes unidades físicas o vienen en escalas muy diferentes, entonces la normalización de los datos de entrenamiento puede mejorar su precisión de forma considerable (Trevor, 2001).

Tanto para la clasificación como para la regresión, una técnica útil puede ser asignar pesos a las contribuciones de los vecinos, de forma que los vecinos más cercanos contribuyan más a la media que los más lejanos. Por ejemplo, un esquema de ponderación común podría consistir en otorgar a cada vecino un peso de $1/d$ donde el parámetro d es la distancia al vecino más próximo.

Los vecinos se toman de un conjunto de objetos de los que se conoce la clase (para la clasificación de K-NN) o el valor de la propiedad del objeto (para la regresión K-NN). Esto puede considerarse como el conjunto de entrenamiento para el algoritmo, aunque no se requiere un paso de entrenamiento implícito. Una peculiaridad del algoritmo K-NN es que es sensible a la estructura de los datos.

Se podría definir el marco estadístico de la metodología K-NN de la siguiente forma:

Supongamos que tenemos los siguientes pares (X_1, Y_1), (X_2, Y_2), …, (X_n, Y_n), tomando valores en R^d x {1,2}, donde Y es la etiqueta de clase X, de modo que $X/Y = \sim P_r$ para $r = 1,2$ (distribuciones de probabilidad P_r). Dada alguna norma $\|.\|$ en R^d y $X \in R^d$,

dejemos que $(X_{(1)}, Y_{(2)}), \ldots, (X_{(n),}\, Y_{(n)})$ sea una ordenación de los datos de entrenamiento tal que $\| X_{(1)} - x \| \leq \cdots \leq \| X_{(n)} - x \|$.

2.3 Tecnologías empleadas en la plataforma web del ICANE

En estos momentos, son tres los marcos de trabajo que aglutinan la mayor parte del desarrollo de frontend web: Angular (Angular, 2021), React (React, 2021) y Vue (Evan You, 2021). Tras un análisis de las características de los tres marcos de trabajo, se puede comprobar que técnicamente no hay grandes diferencias en cuanto a funcionalidad provista o patrones empleados, pero sí en cuanto a ciertos aspectos del diseño y facilidad de uso. Angular es pesado y complejo, React es el más demandado laboralmente (Sala, 2021) pero su curva de aprendizaje es pronunciada y Vue es el más ligero y sencillo de usar y aprender, siendo recomendado para proyectos con equipos pequeños o unipersonales.

Tras el estudio de la situación actual de los marcos de trabajo de frontend JavaScript, se utiliza Vue para frontend dada su gran proyección y su curva de aprendizaje suave.

Github

Es un repositorio en la nube para almacenar proyectos utilizando el sistema de control de versiones Git. Posee diferentes herramientas para trabajo colaborativo entre programadores y funcionalidades web útiles para cada proyecto. Su principal ventaja es la promoción de la colaboración y la productividad gracias a su gestor de proyectos estilo Kanban.

Vuetify

Para la construcción de interfaces de usuario utilizamos la librería Vuetify. Esta librería nos proporciona un listado de componentes reutilizables muy extenso para poder crear aplicaciones con un diseño muy atractivo y visual, siguiendo los patrones de Material Design (Material Design, 2021) especificados por Google.

Chart.js

El desarrollo de las gráficas se realiza utilizando la librería Chart.js. Se trata de una solución open source muy popular que ha reunido alrededor de 55 mil estrellas en GitHub. Combina la sencillez con una adecuada cantidad de funcionalidades. Actualmente tiene una variedad de 8 tipos de gráficos: líneas, barras, radar, dónut, tarta, polar, burbujas y puntos.

Python

Python (Python, 2021) es un lenguaje de programación dinámico, interpretado y multiplataforma de propósito general creado a finales de los años 80 y pensado para trabajar de manera rápida y efectiva gracias a su simplicidad y flexibilidad. Está soportado por la Python Software Foundation, una organización sin ánimo de lucro cuya misión es promover, proteger e impulsar el lenguaje de programación Python, así como facilitar el crecimiento de una comunidad internacional de desarrolladores en dicha tecnología. Python es, junto con R, la tecnología más utilizada en el ámbito del tratamiento de datos

y hoy en día es el lenguaje de programación más utilizado del mundo por delante de otros como Java o C.

La flexibilidad de Python permite desarrollar módulos y paquetes de tratamiento de datos, integrarlos con otros sistemas de información a través de microservicios o publicarlos en portales de difusión de datos. La madurez de sus marcos de trabajo web ofrece, a su vez, la posibilidad de crear sencillas aplicaciones de soporte a cualquier tarea automatizable. En resumen, sus tres aplicaciones principales son: desarrollo web, ciencia de datos y scripting para automatización de tareas.

Firebase

Firebase (Firebase products, 2021) es una plataforma en la nube para el desarrollo de aplicaciones web y móvil principalmente conocida por su servicio de base de datos en la nube (Firebase Realtime Database). Este producto nos proporciona una base de datos NoSQL alojada en la nube que permite almacenar y sincronizar datos entre usuarios en tiempo real, en formato JSON.

R Studio Cloud

El análisis estadístico ha avanzado mucho en los últimos tiempos y las herramientas de software que lo implementan han evolucionado considerablemente con el potenciamiento del cálculo computacional y la programación. R Studio Cloud es una solución en la nube que presenta grandes ventajas a la hora de trabajar online en un ambiente multiplataforma en tiempo real. En la administración pública, por ejemplo, salva todos los inconvenientes existentes relacionados con la incompatibilidad de versiones y paquetes, al ser un producto unificado de la nube de R Studio. Por otro lado, un inconveniente es la posibilidad de instalar extensiones de Python como Keras, de gran tamaño, dado que el espacio disponible en la nube es limitado incluso con el paquete Premium.

3. Resultados

A diferencia de otros estudios similares sobre COVID-19, este trabajo muestra las estimaciones de casos PCR activos para cinco diferentes metodologías de estimación y clasificación. El plan de validación consiste en una primera fase de entrenamiento (training) a partir de la cual los diferentes modelos de aprendizaje automático recogen información sobre el comportamiento de la serie estadística en el pasado, y una segunda fase de testeo (training) que sirve para evaluar el rendimiento del modelo en fase de pronóstico.

Figura 2: Plan de validación de modelos a 30 días con fase de entrenamiento y testeo

A partir del plan de validación, se diseña la estrategia de evaluación de modelos utilizando un enfoque homogéneo. Se comparan los pronósticos de los modelos ARIMA, regresión local (LM), bosque aleatorio (RANDOMFOREST), red neuronal (KERAS) y K-NN (KKNN) junto con el valor observado (ACTUAL) de la serie de casos diarios PCR para Cantabria. Se puede ver en la figura 3 que mientras las metodologías ARIMA, red neuronal y KNN se ajustan mejor a los datos reales, la regresión local y el bosque aleatorio muestran un peor rendimiento y por tanto una peor aproximación a la serie original.

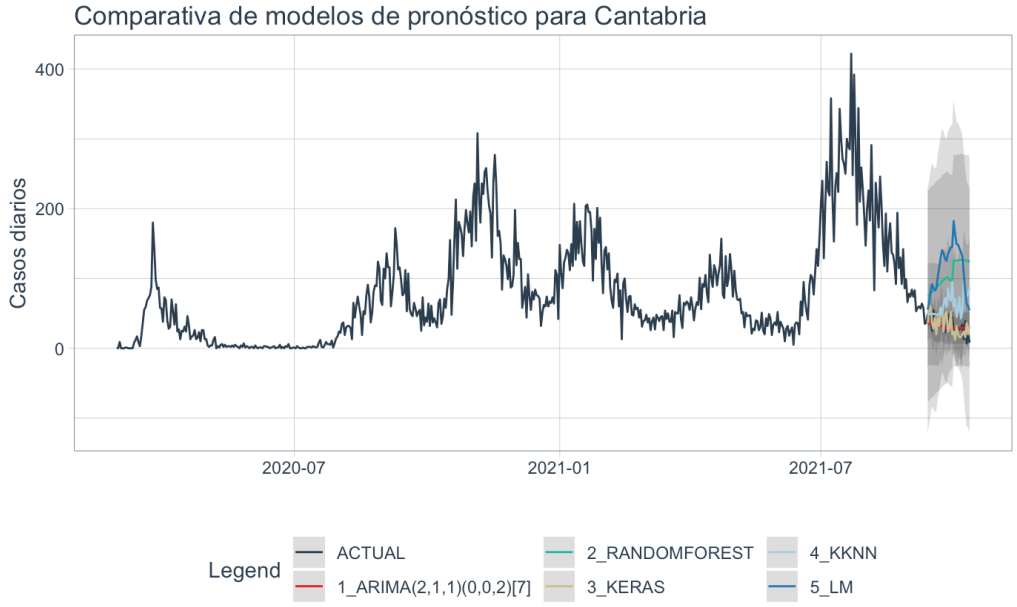


Figura 3: Pronósticos de modelos para casos PCR activos en Cantabria

En el anexo del trabajo se incluyen los pronósticos de cada uno de los modelos de predicción por separado. De igual manera, comparando cada una de las figuras con los datos reales observados es posible comprobar qué modelos tienen un mejor comportamiento en el pronóstico de datos diarios COVID-19. Por ejemplo, revisando las figuras adjuntas con la predicción individual, aquellos modelos que tienen un mayor porcentaje de éxito en la predicción son aparentemente ARIMA y red neuronal. Estos resultados también pueden observarse si echamos un vistazo los parámetros de rendimiento de modelos que se incluyen en la tabla 1.

Tabla 1. Parámetros de rendimiento de los diferentes modelos comparados

	Factor de rendimiento				
Modelo	MAE	MAPE	MASE	SMAPE	RMSE
ARIMA	9.53	43.48	0.88	31.62	11.49
Bosque Aleatorio	71.32	325.14	6.59	102.43	76.58
Red Neuronal	14.33	61.46	1.32	47.71	15.95
K-NN	29.77	158.46	2.75	61.71	36.99
Regresión local	80.51	304.25	7.44	107.30	88.33

Los parámetros de rendimiento muestran que aquellos modelos con un mejor ajuste son aquellos que tienen un menor valor asociado. Por ejemplo, MAPE muestra que ARIMA es el mejor modelo con un valor de 43.48 en comparación con la regresión local que sería el peor modelo en rendimiento con 304.25. De igual manera, el resto de parámetros muestra que por orden los mejores modelos para pronósticos de casos diarios COVID-19 en Cantabria son ARIMA (1º), Red Neuronal (2º), K-NN (3º), Bosque aleatorio (4º) y Regresión local (5º).

4. Conclusiones

Este trabajo lleva a cabo una comparativa de modelos de series temporales empleados para evaluar el comportamiento de brotes epidemiológicos durante la pandemia de COVID-19. Se plantea un análisis de métricas de rendimiento a través de la cual se evalúa un conjunto de modelos de inteligencia artificial junto con otros más clásicos como ARIMA. Los parámetros de rendimiento empleados, así como los pronósticos a 30 días realizados en comparación con datos observados, muestran que tanto la metodología de ARIMA como la de red neuronal tienen un comportamiento más que adecuado en el pronóstico de datos diarios PCR para Cantabria. Estos resultados confirman la

experiencia del personal del ICANE en la evaluación de modelos durante la pandemia del COVID-19. Tanto ARIMA como las redes neuronales, especialmente estas últimas, se ajustaron adecuadamente a las variaciones en casos diarios recogidos por el Servicio Cántabro de Salud y analizados por el Instituto Cántabro de Estadística. En ese proceso los pronósticos fueron a 14 días y mostraron que en ese espacio más corto las metodologías de inteligencia artificial tuvieron cierta ventaja.

Por otro lado, a nivel informático, la experiencia en la recogida y tratamiento de datos diarios ha mostrado el gran potencial de herramientas Cloud-computing disponible en la red. La existencia de repositorios como Github, entornos de desarrollo en la nube como Firebase, lenguajes de programación intuitivos como Python y aplicaciones de programación estadística como R Studio Cloud, han facilitado en gran medida la labor de procesamiento de información COVID-19 disponible. A pesar de la limitada infraestructura de datos COVID-19 durante las primeras etapas de la pandemia, el proceso de recolección de información fue intenso desde el principio, programando en tiempo record una interfaz COVID-19 que fuera capaz de proporcionar información útil sobre la evolución de la pandemia para el ciudadano.

En definitiva, este trabajo muestra una parte importante del trabajo diario del Instituto Cántabro de Estadística durante la pandemia del COVID-19. El aprendizaje ha sido mayúsculo durante este proceso, en el que se ha conseguido recolectar, gestionar, modelizar y analizar datos en tiempo real gracias en gran medida a las posibilidades que ofrecen las soluciones Cloud Computing en todo el proceso.

Bibliografía

Aditya Satrio, CB., Darmawan, W., Unrica Nadia, B., Hanafiah, N. (2021). Time series analysis and forecasting of coronavirus disease in Indonesia using ARIMA model and PROPHET. Procedia Computer Science, Volume 179, p. 524-532. DOI: https://doi.org/10.1016/j.procs.2021.01.036.

Angular: One framework: mobile and desktop: https://angular.io/, 2021.

Cao, L., Liu, H., Li, J., Yin, X., Duan, Y., and Wang, J. (2020). Relationship of meteorological factors and human brucelosis in Hebei Province, China. Sci Total Environ.

Ceylan, Z. (2020). Estimation of COVID-19 prevalence in Italy, Spain, and France. Sci Total Environ. Doi: 10.1016/j.ijid.2018.07.003.

Chakraborty, T.; Ghosh, I. (2020). Real-time forecasts and risk assessment of novel coronavirus (COVID-19) cases: a data-driven analysis. Chaos Solitons Fractals, doi: 10.1016/j.chaos.2020.109850.

Cover, Thomas M.; Hart, Peter E. (1967). "Nearest neighbor pattern classification" (PDF). IEEE Transactions on Information Theory. 13 (1): 21–27. CiteSeerX 10.1.1.68.2616. doi:10.1109/TIT.1967.1053964.

Evan You (2021. Vue: The progressive javascript framework. https://vuejs.org/.

Firebase Products. https://firebase.google.com/products-build, 2021

Fix, Evelyn; Hodges, Joseph L. (1951). Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties (PDF). USAF Schoolf of Aviation Medicine, Randolph Field, Texas.

Hastie, Trevor. (2001). The elements of statistical learning, data mining, inference, and prediction : with 200 full-color illustrations. Tibshirani, Robert., Friedman, J. H. (Jerome H.). New York: Springer. ISBN 0-387-95284-5. OCLC 46809224.

He, Z., Tao, H. (2018). Epidemiology and ARIMA model of positive-rate of influenza viruses among children in Wuhan, China. A nine-year retrospective study. Int J Infect Dis. Doi: 10.1016/j.ijid.2018.07.003.

Kam Ho, T., (1995). Random decisión forests, “Proceeding of 3rd International Conference on Document Analysis and Recognition, p. 278-282, vol. 1, doi: 10.1109/ICDAR.

Kwekha-Rashid, A.S., Abduljabbar, H.N. & Alhayani, B. Coronavirus disease (COVID-19) cases analysis using machine-learning applications. Appl Nanosci (2021). https://doi.org/10.1007/s13204-021-01868-7

Liu, Q., Liu, X., Jiang, B., Yang, W. (2011). Forecasting incidence of hemorrhagic fever with renal síndrome in China using ARIMA model. BMC Infect Dis, 11(1). Doi: 10.1186/1471-2334-11-218.

Material Design. https://material.io/design/, 2021.

Python, a programming language that lets you work quickly and integrate systems more effectively. https://www.python.org/, 2021.

React: a javascript library for building user interfaces. https://reactjs.org/, 2021.

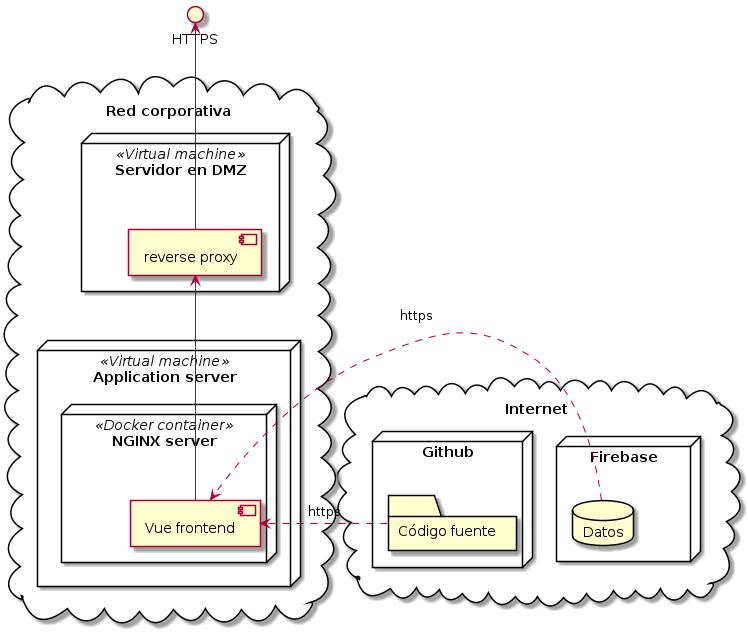
Rismanbaf A. (2020). Potential treatments for COVID-19: a narrative literature review. Arch Acad Emerg med 2020;8(1): e29.

Roy, S., Bhunia, G.S. & Shit, P.K. Spatial prediction of COVID-19 epidemic using ARIMA techniques in India. Model. Earth Syst. Environ. 7, 1385–1391 (2021). https://doi.org/10.1007/s40808-020-00890-y

Sala, X. (2021). Angular vs react vs vue demanda de empleo (2015-2018). https://www.jobfluent.com/es/blog/angular-vs-react-una-evolucion-de-la-demanda-de-empleo, 2021.

Sujath, R., Chatterjee, JM., Hassanien, AE. (2020). A machine learning forecasting model for COVID-19 pandemic in India. Stochastic Environmental Research and Risk Assessment, 34:959-972. https://doi.org/10.1007/s00477-020-01827-8.

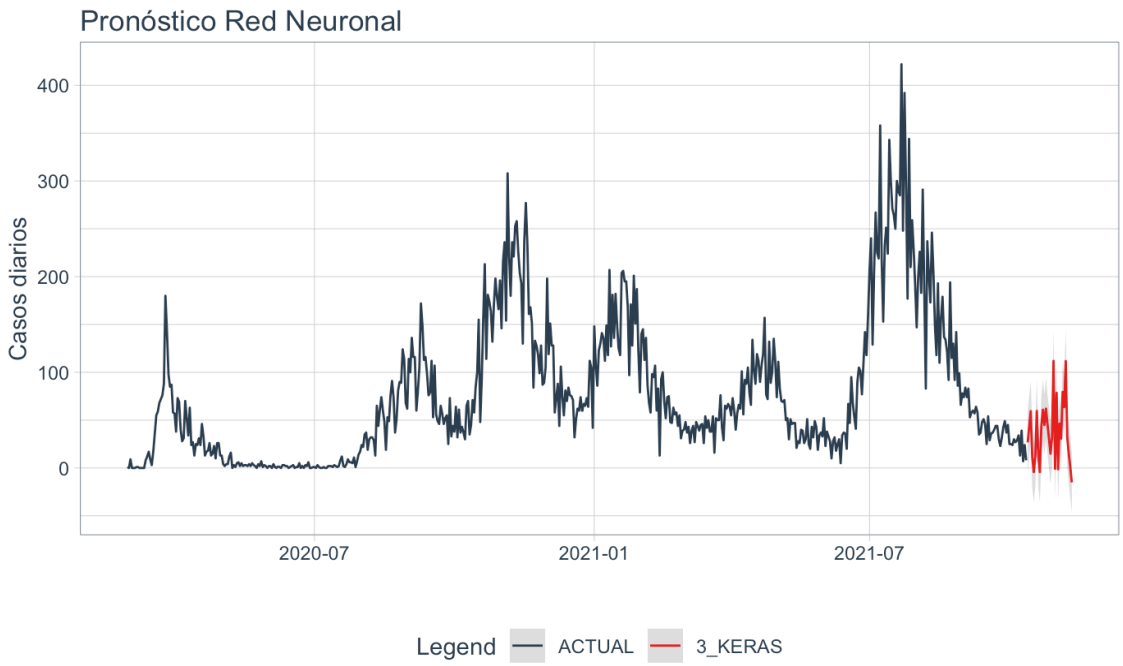
Apéndice



A1. Diagrama de despliegue

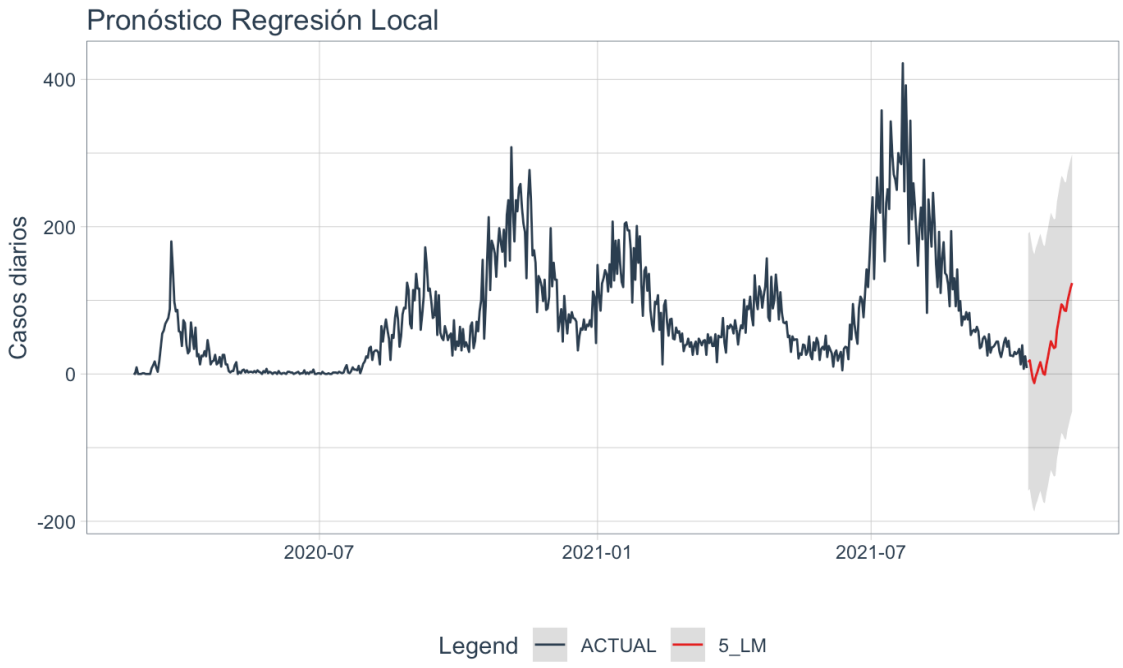
A2. Pronóstico modelo ARIMA

A3. Pronóstico Bosque aleatorio



A4. Pronóstico Red neuronal

A5. Pronóstico K-vecinos próximos (K-NN)



A6. Pronóstico Regresión local