INSTITUTO TECNOLÓGICO DE CIUDAD MADERO

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

MAESTRÍA EN CIENCIAS DE LA INGENIERÍA

TESIS

“Método SVR-ESAR Discreto para Covid-19”

Que para obtener el Grado de:

Maestría En Ciencias De La Ingeniería

Presenta:

Ing. Gerardo de Jesús Martinez Neri

G21073003

1107152

Director(a):

Dr. Juan Javier Gonzalez Barbosa

202134

Co-Director(a):

Dr. Luis Fortino Cisneros Sinencio

Cd. Madero, Tamaulipas Mayo 2023

Ciudad Madero, Tamaulipas, 22/mayo/2023

Oficio No.: U.057/2023
Asunto: Autorización de impresión de tesis

C. GERARDO DE JESÚS MARTINEZ NERI
No. DE CONTROL G21073003
P R E S E N T E

Me es grato comunicarle que después de la revisión realizada por el Jurado designado para su Examen de Grado de Maestría en Ciencias de la Ingeniería, se acordó autorizar la impresión de su tesis titulada:

"MÉTODO SVR-ESAR DISCRETO PARA COVID-19"

El Jurado está integrado por los siguientes catedráticos:

PRESIDENTE:	DR.	JUAN JAVIER GONZÁLEZ BARBOSA
SECRETARIO:	DR.	JUAN FRAUSTO SOLÍS
VOCAL:	DR.	LUIS FORTINO CISNEROS SINENCIO
SUPLENTE:	DRA.	GUADALUPE CASTILLA VALDEZ
DIRECTOR DE TESIS:	DR.	JUAN JAVIER GONZÁLEZ BARBOSA
CO-DIRECTOR:	DR.	LUIS FORTINO CISNEROS SINENCIO

Es muy satisfactorio para la División de Estudios de Posgrado e Investigación compartir con usted el logro de esta meta. Espero que continúe con éxito su desarrollo profesional y dedique su experiencia e inteligencia en beneficio de México.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
"Por mi patria y por mi bien"®

EDUCACIÓN | TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE CIUDAD MADERO
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

MARCO ANTONIO CORONEL GARCÍA
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

ccp. Archivo
MACG 'NRV'

DECLARACIONES DE ORIGINALIDAD, PROPIEDAD INTELECTUAL, CESIÓN DE DERECHOS Y/O CONFIDENCIALIDAD

Yo, Gerardo de Jesús Martinez Neri estudiante del Instituto Tecnológico de Ciudad Madero con el número de control G21073003; declaro en pleno uso de mis facultades mentales que el proyecto que realicé denominado *"***Método SVR-ESAR Discreto para Covid-19"** fue desarrollado respetando los derechos intelectuales de las personas que crearon conceptos mediante las citas en las cuales indico su autoría, y cuyos datos se proporcionan en la bibliografía de la presente; así también la confidencialidad de los métodos, resultados o conocimiento que se hayan generado en el trabajo que se realizó.

Por tal motivo me responsabilizo del contenido, autenticidad y alcance del presente proyecto.

Altamira, Tamaulipas a Mayo del 2023.

DEDICATORIA

Dedico esta tesis a mis padres Eladia Neri y Gerardo Martínez; también en especial a mi abuelo Rafael Corona [+] , quienes han sido mi mayor apoyo y motivación en todo momento.

A mi hermana Liliana por ser un apoyo constante y no dejarme solo en ningún momento.

A mi madre por enseñarme lo que es el amor.

Gracias por su amor incondicional, su paciencia y por creer en mí siempre.

A mi director de tesis, por su guía experta, paciencia y dedicación para ayudarme a alcanzar mis objetivos.

A mis amigos y compañeros de investigación, por su colaboración, apoyo y por compartir conmigo esta experiencia inolvidable. A todas las personas que de alguna manera han contribuido a la realización de esta tesis, mi más sincero agradecimiento.

AGRADECIMIENTOS

Al Consejo Nacional de Ciencia y Tecnología (CONACyT), A la División de Estudios de Posgrado e Investigación (DEPI) del Instituto Tecnológico de Ciudad Madero, por el apoyo económico y académico proporcionado para la realización de la maestría y mejorar mi nivel como estudiante.

A mi director, el Dr. Juan Javier González Barbosa, y al profesor Dr. Juan Frauto Solís, por su gran apoyo académico, su enorme paciencia y conocimiento para la realización de esta Tesis.

A mis compañeros de la maestría, por estar siempre presentes y por compartir sus experiencias y aprendizajes.

Por su puesto: A Dios, quien me dio la vida y me dio la oportunidad de llegar hasta donde estoy.

Método SVR-ESAR Discreto para Covid-19

Gerardo de Jesús Martinez Neri

Resumen

Desde 2019, muchas personas en México han enfermado por coronavirus (COVID -19), que es causada por el virus del Síndrome Respiratorio Agudo Severo 2 (SARS-Cov2). Por ello, los protocolos sanitarios establecidos en todo el país para combatir la propagación del virus deben adaptarse en función de los constantes cambios de contagio. El objetivo fundamental de una previsión es reducir el rango de incertidumbre en el que se toman las decisiones que afectan al futuro y, por tanto, a todas las partes implicadas. La ayuda de estas previsiones en la determinación de nuevas medidas en la lucha contra el COVID se manifiesta. En este trabajo se presenta un modelo para el pronóstico a 10 días de los casos de infección de COVID -19 en México denominado Perceptrón Multicapa - Regresión de Vectores de Soporte con Suavizado Exponencial (MLP-SVRES). La regresión de vectores de apoyo y el perceptrón multicapa se utilizan como métodos de predicción. El suavizado exponencial se utiliza como método de mejora. El periodo de entrenamiento se define entre el 1 de agosto y el 31 de octubre de 2021. Los resultados muestran que la aplicación del método de suavizado exponencial reduce drásticamente el error que medimos con la métrica MAPE.

Discrete SVR-ESAR Method for Covid-19

Gerardo de Jesús Martinez Neri

Abstract

Since 2019, many people in Mexico have become ill with coronavirus disease (COVID -19), which is caused by the Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-Cov2) virus. For this reason, the health protocols established throughout the country to combat the spread of the virus must be adapted according to the constant changes in contagion. The fundamental objective of a forecast is to reduce the range of uncertainty within which decisions are made that affect the future and, therefore, with it all parties involved. The help of such forecasts in determining new measures in the fight against COVID is expressed. In this work, we present a model for a 10-day forecast of the infection cases of COVID -19 in Mexico called Multi-layer Perceptron - Support Vector Regression with Exponential Smoothing (MLP-SVRES). Support Vector Regression and Multi-layer Perceptron are used as forecast methods. Exponential Smoothing is used as an enhancement method. The training period is defined from August 1 to October 31, 2021. The results show that applying the exponential smoothing method drastically reduces the error which we measure with the MAPE metric.

Índice General

Índice Tablas

Índice de Figuras

Nomenclatura

SVR : Suppor Vector Regression
MLP : Multilayer Perceptron (Perceptrón Multicapa)

Capítulo 1

1 Introducción

La humanidad ha sufrido más de 20 grandes epidemias y pandemias de las que se tiene evidencia, entre ellas destacan cuatro por su velocidad de contagio y mortalidad: La viruela, la peste negra, la gripe española y VIH. Por lo anterior, la necesidad de crear modelos que permita entender las fluctuaciones de epidemia es de suma importancia.

Estos modelos pueden ayudar a los científicos y responsables de salud pública a predecir la propagación de una enfermedad y tomar medidas para controlarla. Además, los modelos también pueden ayudar a comprender mejor los factores que influyen en la propagación de una enfermedad, como la movilidad de las personas, el contacto cercano, la densidad poblacional y otros factores socioeconómicos.

En los últimos años, con la llegada de nuevas tecnologías y herramientas de análisis de datos, se han desarrollado modelos cada vez más sofisticados para predecir y controlar las epidemias. Algunos de estos modelos incluyen técnicas de aprendizaje automático y redes neuronales, que pueden analizar grandes conjuntos de datos y detectar patrones y tendencias.

Además, el Modelo SIR permite evaluar la efectividad de las medidas de control y prevención implementadas. Por ejemplo, se pueden simular diferentes escenarios y comparar los resultados para determinar cuál es la estrategia más efectiva para reducir la tasa de infección y mortalidad. Este modelo también es útil para predecir el momento en que la epidemia alcanzará su pico y comenzará a disminuir, lo que permite planificar y prepararse adecuadamente.

Capítulo 1

Otro aspecto importante es que el Modelo SIR es adaptable y se puede mejorar con más datos e información. Por ejemplo, se pueden incorporar factores como la edad, la condición médica subyacente, la densidad poblacional, entre otros, para hacer una predicción más precisa y personalizada.

Modelos epidemiologicos como el Modelo SIR son valiosos al predecir la evolución de la epidemia y evaluar la eficacia de las medidas de control y prevención. Es importante actualizar y mejorar estos modelos con los últimos datos disponibles para responder eficazmente a futuras crisis de salud pública.

1.1 Planteamiento del problema

La implementación de métodos inteligentes en la predicción de nuevos casos epidémicos puede mejorar significativamente la precisión de los resultados y permitir la toma de decisiones más efectivas. Al utilizar técnicas de aprendizaje automático y análisis de datos avanzados, se pueden obtener predicciones más precisas y detalladas sobre la propagación de enfermedades infecciosas.

La combinación de las ecuaciones del modelo SIR y los métodos inteligentes nos permite obtener una visión más completa de la situación y, por ende, desarrollar planes de acción más efectivos para prevenir y controlar epidemias. Al contar con una mejor comprensión de los factores que influyen en la propagación de enfermedades, podemos tomar medidas

Capítulo 1

preventivas más eficaces, como la cuarentena temprana, el seguimiento de contactos cercanos y la distribución eficiente de recursos de salud.

1.2 Objetivos

- Desarrollar un sistema para el pronostico de los casos confirmados de COVID con la ayuda de la resolución de las ecuaciones diferenciales del modelo SIR.

- Presentar un modelo para la predicción de 10 días de los casos confirmados de COVID - 19 en México utilizando un Perceptrón Multicapa (MLP), Suppor Vector Regression (SVR) con Suavizado Exponencial (MLP-SVRES).

1.3 Justificación del estudio

- Aunque observar fenómenos naturales y experimentos sociales nos permite aprender sobre las epidemias del pasado y sus patrones de dispersión, precisamos nuevo conocimiento para prevenir crisis futuras.
- Se pueden emplear las técnicas de Modelado y Simulación para anticipar medidas de respuesta venideras. La simulación se sustenta en la construcción de un modelo matemático que explica el comportamiento del fenómeno en cuestión. Dicho modelo integra una serie de ecuaciones y variables que describen las distintas características

que impactan en el fenómeno, tales como la velocidad de propagación y la tasa de recuperación.

- Al simular múltiples casos, se puede evaluar el impacto de diferentes medidas de prevención y control de enfermedades infecciosas y determinar la efectividad de cada una de ellas. La simulación por computadora también puede ser utilizada para predecir la propagación de una enfermedad y para desarrollar planes de acción para controlar y prevenir su expansión.

- El uso de modelos y simulación, en conjunto con datos obtenidos de casos previos de epidemias y enfermedades contagiosas puede proveer mecanismos efectivos de predicción

Capítulo 1

Capítulo 2

2 Antecedentes/Marco Teórico

A continuación se presenta el marco teórico que sustenta el presente trabajo en el cual se presentará una revisión del estado del arte en cuanto a la aplicación de técnicas de aprendizaje automático en la predicción de la evolución de la pandemia de COVID-19, con un enfoque específico en las máquinas de vectores de soporte y las redes neuronales multicapa. Se abordarán también los antecedentes teóricos de estas técnicas y su aplicación en otras áreas de estudio. Con ello, se espera contribuir al conocimiento actual sobre la aplicación de técnicas de aprendizaje automático en la predicción de eventos relacionados con la pandemia de COVID-19.

2.1 Pronósticos

Los pronósticos son estimaciones futuras basadas en patrones y tendencias identificados en datos históricos. Estas estimaciones se utilizan en una variedad de industrias, desde finanzas y economía hasta la gestión de inventarios y la toma de decisiones empresariales. (Makridakis, 1998)

Una de las mayores ventajas de los pronósticos es que permiten a las empresas planificar y tomar decisiones informadas. Por ejemplo, una empresa puede usar un pronóstico de ventas futuras para determinar cuánto inventario deben mantener en sus almacenes o cuántos empleados deben contratar. También pueden ayudar a las empresas a identificar tendencias en el mercado y ajustar sus estrategias en consecuencia. Además, los pronósticos pueden ser útiles para la gestión de riesgos.

Capítulo 2

Otra ventaja de los pronósticos es que pueden mejorar la eficiencia operacional. Por ejemplo, una empresa puede usar un pronóstico de crecimiento del mercado para determinar si una inversión en un nuevo producto o línea de negocios es viable, los pronósticos son una herramienta valiosa para la toma de decisiones en una variedad de industrias que les permiten a las empresas planificar, mitigar el riesgo, mejorar la eficiencia operacional y tomar decisiones de inversión informadas. En la figura 1 vemos algunos ejemplos de tipos de pronósticos. Dependiendo del campo de estudio o aplicación, pueden existir otros tipos de pronósticos específicos

Capítulo 2

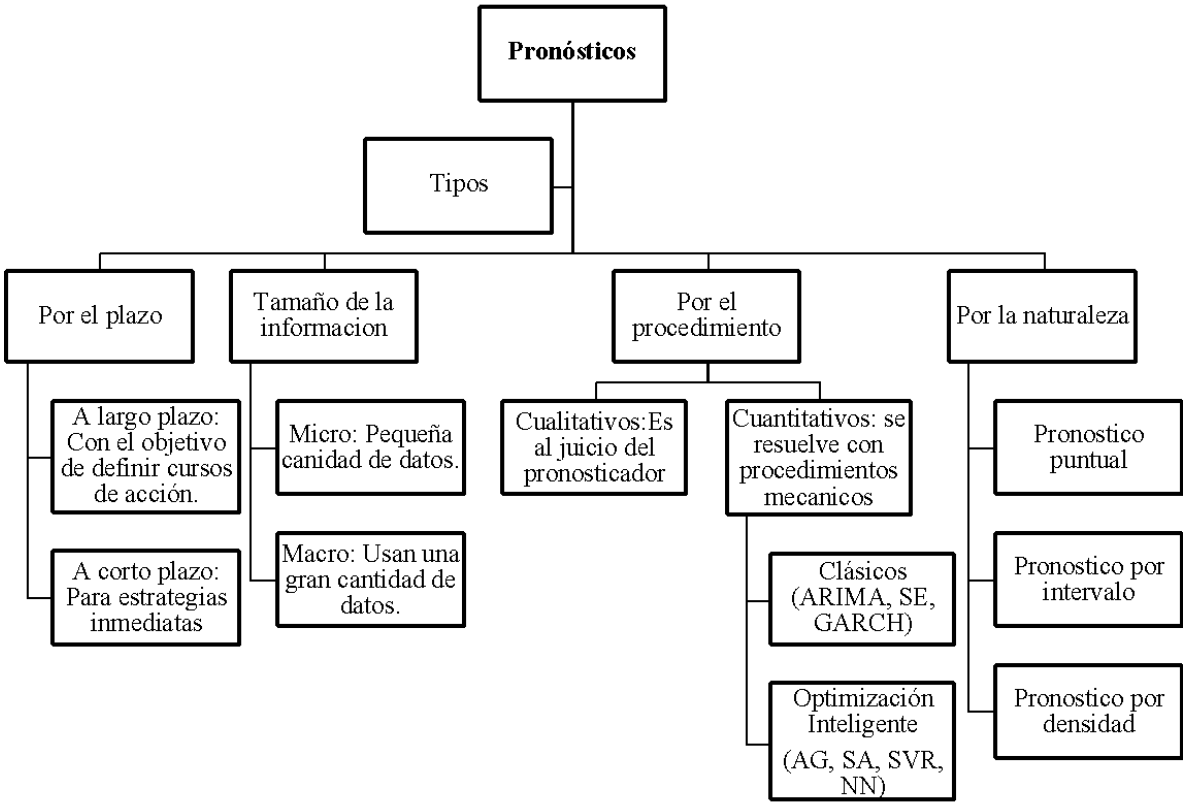

Figura 1. Tipos de pronósticos

Fuente: J. E. Hanke and D. W. Wichern

Capítulo 2

2.2 Métodos inteligentes para predicción

Los métodos inteligentes para la predicción son una clase de técnicas de aprendizaje automático que se utilizan para predecir una variable objetivo a partir de un conjunto de datos de entrada. Estos métodos se basan en la idea de que los patrones en los datos de entrada pueden ser aprendidos y utilizados para predecir la variable objetivo. (Hastie, 2009)

Hay muchos tipos diferentes de métodos inteligentes para la predicción, incluyendo regresión lineal, regresión logística, redes neuronales, árboles de decisión, k-vecinos más cercanos (k-NN), support vector machine (SVM) y random forest. Cada uno de estos métodos se aplica a diferentes tipos de problemas y tiene sus propias fortalezas y debilidades.

Además de estos métodos de predicción, también existen técnicas más avanzadas como los modelos de aprendizaje profundo, como las redes neuronales profundas (deep neural networks) y las redes de recurrentes (recurrent neural networks). Estos métodos utilizan una arquitectura más compleja y multi-capa que permite aprender patrones más complejos en los datos de entrada. (Bishop, 2006)

Existe una amplia variedad de métodos inteligentes para la predicción que se pueden aplicar a diferentes problemas. La elección del método adecuado depende de las características

Capítulo 2

específicas del problema, el tamaño y la naturaleza de los datos de entrada, y las metas de predicción. Es importante evaluar diferentes métodos y seleccionar el método que ofrezca la mejor precisión y generalización para un problema dado.

2.3 Support Vector Machine

El SVM o Support Vector Machines es un algoritmo de aprendizaje automático supervisado ampliamente utilizado para resolver problemas de clasificación y regresión. Fue introducido por el investigador Vapnik en 1995 y desde entonces ha sido ampliamente utilizado en aplicaciones en una amplia gama de campos, incluyendo la biología, la finanzas, el reconocimiento de patrones y la minería de datos.

El funcionamiento del SVM se basa en la idea de encontrar un hiperplano en un espacio de características que separe los datos en dos o más clases. Este hiperplano es conocido como vector de soporte y es el que maximiza el margen , es decir, la distancia entre el hiperplano y los vectores más cercanos de cada clase. Estos vectores más cercanos son conocidos como vectores de soporte y son los que definen la linea divisora. (Cortes, 1995)

Además, el SVM también tiene la capacidad de manejar datos no lineales a través de técnicas de kernel, que permiten transformar los datos de entrada en un espacio de características más adecuado para su separación. Esto significa que el SVM es capaz de resolver problemas complejos que no pueden ser resueltos mediante técnicas lineales.

Capítulo 2

Otro aspecto importante del SVM es su capacidad de manejar problemas de clasificación multi-clase. A diferencia de otros algoritmos de aprendizaje automático que requieren la construcción de un modelo para cada clase, el SVM utiliza un enfoque de uno contra todos, donde se construyen modelos binarios para cada clase y se realiza una selección final basada en votación. (Schölkopf, 2002)

En términos de desempeño, el SVM es conocido por su capacidad de generalización y su robustez ante el sobre entrenamiento. Además, su capacidad de manejar una gran cantidad de datos y su eficiencia en términos de tiempo de entrenamiento lo hacen una excelente opción para aplicaciones en tiempo real.

SVM es un algoritmo de aprendizaje automático versátil y efectivo que ha sido ampliamente utilizado en una amplia gama de aplicaciones. Su capacidad de manejar datos no lineales, resolver problemas de clasificación multi-clase y su buen desempeño en términos de generalización y eficiencia lo hacen una excelente opción para una amplia gama de problemas.

2.4 Support Vector Regression

Los vectores de soporte de regresión (Support Vector Regression , SVR, por sus siglas en inglés) es un algoritmo de aprendizaje automático que se utiliza en el campo de la minería de

Capítulo 2

datos para resolver problemas de regresión. A diferencia de otros métodos de regresión, como la regresión lineal, el SVR no se limita a lineas rectas y puede producir curvas complejas.

Utiliza una función de costo que es una versión adaptada de la típica función de costo utilizada en el Support Vector Machines (SVM). La función de costo en el SVR se compone de dos partes: una penalización por error y una regularización. La penalización por error se utiliza para evitar que los errores se vuelvan demasiado grandes, mientras que la regularización ayuda a prevenir el sobreajuste de los datos.

En el entrenamiento del SVR, el algoritmo trata de encontrar los vectores de soporte que se encuentran a cierta distancia de la frontera de decisión. Estos vectores de soporte se utilizan para formar una frontera que separa los datos en dos regiones. La frontera se elige de manera tal que minimiza la suma de los errores y la penalización por error. (Smola, 2004)

El método SVR se utiliza en una amplia variedad de aplicaciones, incluyendo la estimación de precios de acciones, la predicción de ingresos y la detección de fallos en sistemas. En estos casos, el SVR puede proporcionar una solución más precisa y robusta que otros métodos de regresión.

SVR es una herramienta valiosa en el campo de la minería de datos y el aprendizaje automático que permite resolver problemas de regresión con una gran precisión y robustez. Su capacidad para manejar curvas complejas y su enfoque en la penalización de errores y la regularización lo hacen ideal para una amplia variedad de aplicaciones.

Capítulo 2

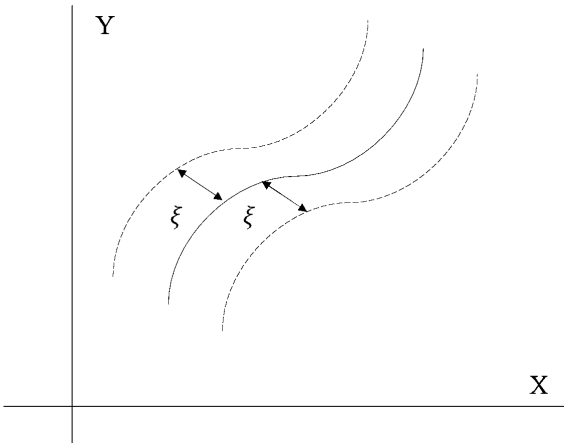

Figura 2. Representación de SVR

[V. Vapnik,(1997)]

2.5 Perceptrón Multicapa, MLP.

El Perceptrón Multicapa (MLP por sus siglas en inglés) es un tipo de algoritmo de aprendizaje automático que se utiliza en el aprendizaje profundo y el análisis de datos. Es una técnica de aprendizaje supervisado que se basa en la estructura de una red neuronal artificial, donde los nodos de una capa se conectan a los nodos de la capa siguiente a través de pesos y bias.
El propósito principal del MLP es clasificar patrones complejos y hacer predicciones a partir de una gran cantidad de entradas. A diferencia de otros algoritmos de aprendizaje, el MLP es capaz de realizar múltiples transformaciones de las entradas para llegar a una solución final,

Capítulo 2

lo que lo hace especialmente útil para problemas que requieren un procesamiento más profundo de los datos. (Rosenblatt, 1958)

El funcionamiento del MLP se basa en la retropropagación, un algoritmo que permite calcular la derivada parcial de una función objetivo con respecto a los pesos y bias de la red. La retropropagación es un proceso iterativo que se utiliza para optimizar los pesos y bias de la red, permitiendo que la red aprenda de los datos de entrenamiento y se adapte para hacer mejores predicciones en el futuro.

Una de las principales ventajas del MLP es su capacidad para manejar problemas de regresión y clasificación de manera efectiva. También es capaz de manejar una gran cantidad de variables y patrones complejos, lo que lo hace ideal para aplicaciones en áreas como la investigación médica, la finanzas, la marketing y la biología.
Sin embargo, el MLP también tiene algunos desafíos, como la posibilidad de que se produzca un sobreajuste y la dificultad para interpretar los resultados. Para resolver estos problemas, es importante trabajar con una cantidad adecuada de datos de entrenamiento y utilizar técnicas de regularización y validación cruzada.

MLP es una técnica de aprendizaje automático muy versátil y poderosa, capaz de manejar problemas complejos y hacer predicciones precisas a partir de una gran cantidad de entradas. Es una herramienta valiosa para aquellos que buscan resolver problemas complejos y mejorar la eficiencia de sus procesos a través del aprendizaje automático.

Capítulo 2

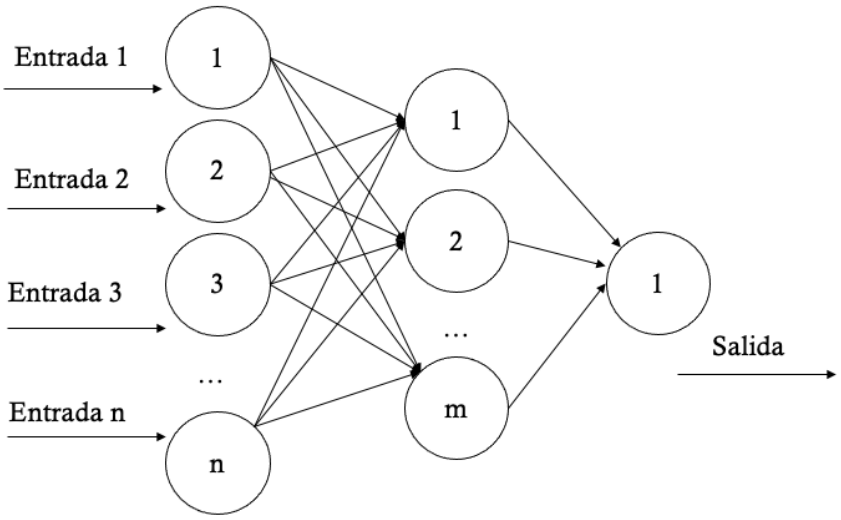

Figura 3. Representación de Perceptron.

[Rosenblatt, F. (1958)]

Capítulo 2

2.6 Modelos epidemiológicos

Los modelos epidemiológicos son una herramienta valiosa para comprender y predecir la dinámica de una enfermedad en una población. Estos modelos utilizan una combinación de matemáticas, estadísticas y epidemiología para simular la transmisión de una enfermedad entre las personas. La idea es que, al comprender cómo una enfermedad se propaga, los expertos pueden utilizar esta información para implementar medidas de prevención y control efectivas.

Los modelos epidemiológicos se basan en la interacción entre las personas afectadas por una enfermedad y las que no están afectadas. Se utilizan ecuaciones diferenciales para describir la dinámica de la transmisión de la enfermedad, y estadísticas para incorporar información sobre la incidencia y prevalencia de la enfermedad en la población. Los modelos epidemiológicos también tienen en cuenta factores como la tasa de transmisión de la enfermedad, la tasa de recuperación de las personas infectadas y la tasa de mortalidad.

Existen varios tipos de modelos epidemiológicos, incluyendo modelos SIR (Susceptible-Infectados-Recuperados), modelos SEIR (Susceptible-Expuestos-Infectados-Recuperados) los cuales son modelos compartimentales. Cada uno de estos modelos se utiliza para describir una dinámica diferente de la transmisión de la enfermedad, y cada uno se aplica a una amplia variedad de enfermedades, desde enfermedades infecciosas hasta enfermedades crónicas y degenerativas.

Capítulo 2

Son una herramienta esencial para comprender y predecir la dinámica de una enfermedad en una población. Son utilizados por epidemiólogos, expertos en salud pública y otros profesionales para tomar decisiones informadas sobre cómo prevenir y controlar la enfermedad, y para guiar la investigación y el desarrollo de nuevas intervenciones y tratamientos.

2.7 Modelo SIR

El Modelo SIR, también conocido como Modelo de la Dinámica de las Epidemias, es un modelo matemático que se utiliza para describir la dinámica de la propagación de enfermedades infecciosas. Fue desarrollado por un matemático sueco llamado Kermack y un epidemiólogo británico llamado McKendrick en 1927.

Este modelo se basa en la idea de que la población total de una comunidad se puede dividir en tres grupos: personas susceptibles (S), personas infectadas (I) y personas recuperadas o inmunizadas (R). Se asume que las personas en el grupo S pueden ser infectadas por las personas en el grupo I, mientras que las personas en el grupo R están protegidas contra futuras infecciones. (Castillo-Chavez, 2002)

El Modelo SIR utiliza ecuaciones diferenciales para describir la dinámica de la epidemia. La tasa de cambio de la población de personas susceptibles depende de la tasa de contacto con las personas infectadas y la probabilidad de transmisión de la enfermedad, mientras que la

tasa de cambio de la población de personas infectadas depende de la tasa de infección y la tasa de recuperación.

El Modelo SIR se ha utilizado para predecir la dinámica de la propagación de enfermedades como el VIH/SIDA, la gripe, el ébola y la pandemia actual de COVID-19. Además, este modelo se utiliza para evaluar la efectividad de las medidas de control y prevención de enfermedades, como la vacunación y las medidas de distanciamiento social.

Es importante destacar que el Modelo SIR es un modelo simplificado y no tiene en cuenta todas las variables realistas que pueden afectar la dinámica de la epidemia, como la presencia de asintomáticos o la existencia de brotes secundarios. Sin embargo, es una herramienta valiosa para comprender la dinámica de la propagación de enfermedades y tomar medidas para proteger la salud pública.

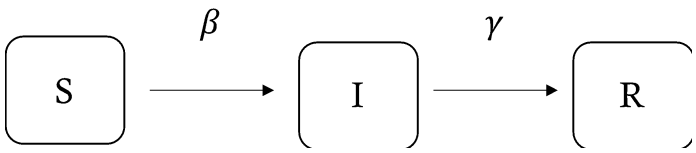

Figura 4. Flujo de Tranmisión en Modelo SIS

[Castillo-Chavez, C. & Yakubu, A. (2002)]

Capítulo 2

2.8 Número reproductivo básico (R_0).

El Número Reproductivo Básico (R_0) es un concepto importante en epidemiología y se utiliza para describir la capacidad de una enfermedad infecciosa para propagarse de una persona a otra. Este número representa la cantidad promedio de personas que una persona infectada puede infectar en un entorno en el que ninguna persona está protegida o inmunizada contra la enfermedad. (White LF, 2008)

El valor de R0 se utiliza para predecir la tasa de propagación de una enfermedad en una población y para determinar la intensidad de una epidemia. Un valor de R_0 alto indica que una enfermedad se está propagando rápidamente, mientras que un valor bajo indica una propagación más lenta.

El cálculo de R0 depende de varios factores, incluyendo la tasa de transmisión de la enfermedad, la duración de la enfermedad y la cantidad de contactos que una persona infectada tiene con otras personas. Además, el valor de R0 puede cambiar a lo largo del tiempo y en diferentes entornos, ya que la transmisión de la enfermedad puede ser influenciada por la implementación de medidas de control y prevención, la inmunidad de la población y la presencia de variantes de la enfermedad.

El conocimiento de R0 es esencial para la planificación de respuestas a epidemias y pandemias, ya que permite a los responsables de la salud pública determinar la efectividad de las medidas de control y prevención implementadas y adaptarlas en consecuencia. Por

ejemplo, si se logra reducir el valor de R0 a debajo de 1, la epidemia eventualmente disminuirá y se extinguirá.

En conclusión, el Número Reproductivo Básico es un indicador clave para entender la propagación de enfermedades infecciosas y tomar medidas efectivas para controlarlas. La comprensión de su valor y su evolución es fundamental para la respuesta a emergencias de salud pública y para proteger a la población contra la transmisión de enfermedades.

$$\mathcal{R}_0 = \frac{\beta}{\gamma}$$

Figura 6. Fórmula de R_0

[White LF, Pagano M.(2008)]

Capítulo 2

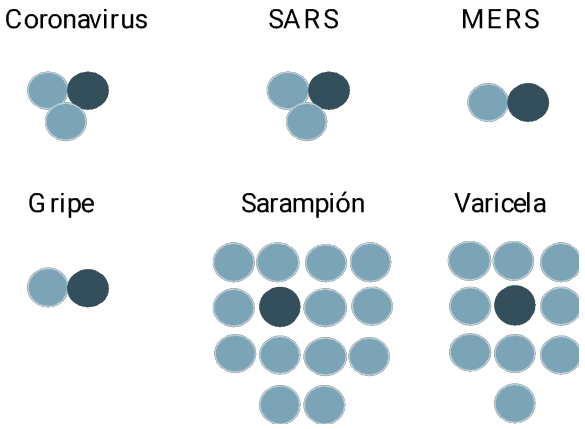

Figura 7. R_0 de diferentes virus.

2.9 Métricas de medición de error.

Las métricas de medición de error son herramientas utilizadas para evaluar la precisión y efectividad de un modelo de aprendizaje automático. Estas métricas permiten a los científicos de datos y otros profesionales en el campo de la inteligencia artificial comparar diferentes modelos y determinar cuál es el más adecuado para una tarea específica. Algunas de las

métricas de medición de error más comunes incluyen la precisión, el recall, la medida F1, la matriz de confusión y el coeficiente de correlación de Pearson.

También se utilizan para evaluar una amplia variedad de problemas, incluyendo la clasificación, la regresión y la identificación de tendencias. Al elegir la métrica adecuada, los profesionales pueden maximizar la precisión y eficacia de sus modelos, mejorar la toma de decisiones y proporcionar resultados más precisos para sus clientes o usuarios finales.

Algunas de las métricas de medición de error más comúnmente utilizadas:

1. Error absoluto medio (MAE): Es la media de los errores absolutos y se usa para medir la magnitud del error. (L. Ying, 2016)
2. Error cuadrático medio (MSE): Es la media de los errores cuadrados y se utiliza para medir la variación en los datos. (J. G. C. M. Smit, 2002)
3. Root Mean Squared Error (RMSE): Es la raíz cuadrada del error cuadrático medio y se utiliza para medir la variación en los datos. (A. R. Delen, 2009)
4. Error relativo medio (MAPE): Es la media del error relativo y se usa para medir la precisión de las predicciones. (Makridakis, Forecasting: methods and applications (2nd ed.)., 1998)
5. Coeficiente de determinación (R^2): Es una medida de la calidad de ajuste de un modelo y se define como la proporción de la variabilidad en los datos explicada por el modelo. 14.(Vandewalle, 1999)

Capítulo 2

2.10 Error porcentual absoluto medio

Mean Absolute Percentage Error (MAPE) es una métrica ampliamente utilizada para medir la precisión de un modelo de pronóstico o para evaluar la eficacia de una estrategia de pronóstico. Esta métrica representa la media del porcentaje absoluto de error en cada observación. En otras palabras, es una medida de la magnitud del error relativo a la magnitud de la variable pronosticada. (Makridakis, Forecasting: methods and applications (2nd ed.)., 1998)

La fórmula para calcular el MAPE es la siguiente:

MAPE = (1/n) * Σ|(Actual - Pronosticado)/ Actual| * 100

dónde:

- n es el número de observaciones
- Actual es el valor real o observado
- Pronosticado es el valor pronosticado por el modelo

El MAPE se utiliza comúnmente en la industria y en la investigación para evaluar la precisión de los modelos de pronóstico. Es una métrica fácil de entender y interpretar, y es útil para comparar la precisión de diferentes modelos o para evaluar la efectividad de diferentes estrategias de pronóstico.

Capítulo 2

Sin embargo, hay algunas limitaciones al utilizar el MAPE, una de las mayores limitaciones es que el MAPE puede ser altamente sensible a los outliers o valores atípicos. Además, el MAPE puede ser muy distorsionado cuando los valores pronosticados son muy pequeños en comparación con los valores reales, lo que puede resultar en valores de MAPE extremadamente grandes. Por lo tanto, es importante tener en cuenta estas limitaciones al utilizar el MAPE y considerar otras métricas de evaluación de modelos para complementar la evaluación de la precisión de los modelos de pronóstico.

2.11 Error porcentual absoluto medio simétrico

El SMAPE, también conocido como la tasa de errores absolutos porcentuales simétricos, es una métrica común utilizada en el análisis de pronósticos financieros y en la evaluación de modelos de pronóstico de series de tiempo. La principal diferencia entre el SMAPE y otras métricas de error, como el MAPE, es que el SMAPE toma en cuenta tanto el error en las estimaciones superiores como las inferiores. (Koehler, 1989)

A diferencia de otras métricas, como la Mean Absolute Percentage Error (MAPE), SMAPE es simétrico, lo que significa que tanto los errores de sobreestimación como de subestimación tienen el mismo peso en el cálculo.

La ecuación del SMAPE se calcula como la suma del error absoluto dividido entre la suma del valor real y la estimación, y luego multiplicado por 200.

La siguiente es la fórmula:

SMAPE = 200 * (Σ |real - estimado|) / (Σ (real + estimado))

El valor de SMAPE varía entre 0 y 100, con valores más bajos indicando una mayor precisión del modelo. Un modelo con un SMAPE de 0 significa que ha logrado un pronóstico perfecto, mientras que un modelo con un SMAPE de 100 indica que el error es igual al valor pronosticado.

El SMAPE es una métrica popular para evaluar modelos de pronóstico en situaciones en las que los errores en las estimaciones superiores y inferiores son igualmente importantes, y cuando se necesita tener una visión más balanceada de la precisión de un modelo.

Capítulo 2

Capítulo 3

3 Metodología

En esta sección, se describe la metodología MLP-SVRES para el pronostico de casos diarios de COVID-19. Los tipos de casos utilizados para la prueba de la metodología son los casos diarios de personas contagiadas, susceptibles, fallecidas y recuperadas a nivel nacional pertenecientes al país de México. Los métodos de pronóstico son SIR, SVR y MLP, mientras que SE es empleado como método de suavizamiento (mejorar). En la Figura 8. se observa la metodología propuesta MLP-SVRES la cual está compuesta por 3 fases generales: las series de tiempo de COVID-19, los métodos de pronóstico, el modelo de ecuaciones diferenciales, el método de mejora y las métricas de evaluación. Las fases de la metodología MLP-SVMES son descritas en las siguientes secciones.

La metodología MLP-SVRES se ha utilizado con éxito para predecir los casos diarios de COVID-19 en México. Al proporcionar una estructura sólida y eficiente para el pronóstico de casos diarios, esta metodología puede ser valiosa para ayudar a los profesionales de la salud y tomadores de decisiones a anticipar la propagación del virus y tomar medidas preventivas en consecuencia. Dado que la pandemia de COVID-19 sigue siendo un problema global, la metodología MLP-SVRES tiene el potencial de ser adaptada y aplicada en otros países afectados por la pandemia. A medida que continuamos lidiando con los efectos de la pandemia, se espera que la metodología MLP-SVRES y otras herramientas de pronóstico sigan siendo de gran valor en la lucha contra COVID-19 y otras enfermedades infecciosas en el futuro.

Capítulo 3

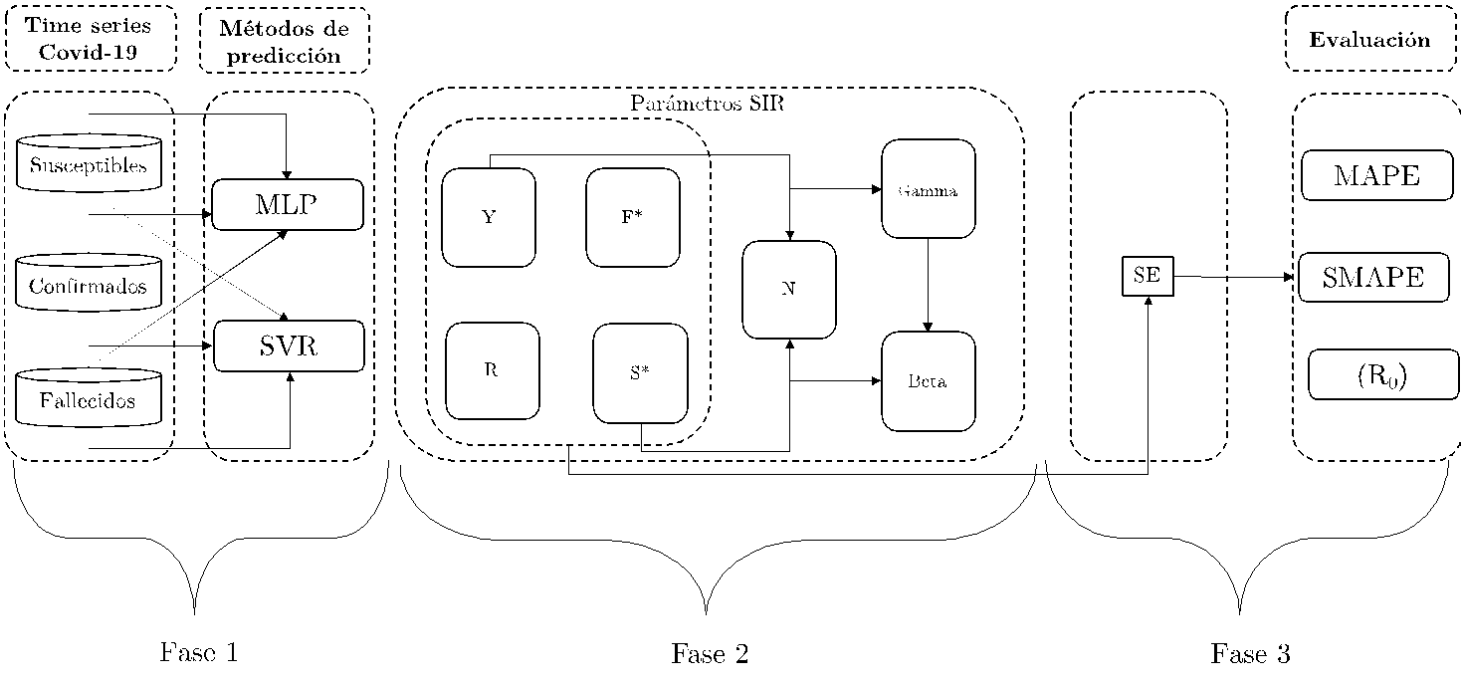

Figura 8. Metodología MLP-SVRES propuesta.

3.1 Serie de Tiempo

La colección de series de tiempo está compuesta por los casos de COVID-19 del tipo confirmado, sospechoso y fallecidos. Las series de tiempo fueron obtenidas por los datos abiertos publicados por la Dirección General de Epidemiología en México (Mexico., 2021).

Capítulo 3

La selección del periodo acontece a los periodos con mayor fluctuación, el cual abarco del 1 de agosto del 2021 al 31 de octubre del 2021.

Debido a que las series no presentaron valores a típicos, no se requirió aplicar técnicas de limpieza de datos. Para poder evaluar el rendimiento de la metodología MLP-SVRES, las series de tiempo fueron segmentadas en tres conjuntos, siendo un 79% para entrenamiento, 11% para validación y 11% para prueba.

3.2 Métodos de predicción

El propósito de esta fase es entrenar modelos para pronosticar los casos de COVID-19, los cuales son SVR y MLP métodos asociados a ML y DL. Los métodos de ML y DL se caracterizan por tener un aprendizaje supervisado, por lo que los datos sometidos a estos son del tipo problema supervisado. Un problema supervisado está definido por una matriz de observaciones X_{ij} y sus respectivas clases y_i para resolver la expresión $y = wx + b$. El conjunto de entrenamiento esta dado por $T = \{(x_i, y_i), i = 1, \dots, m\}$, donde cada $x_i \in R^n$ es la i-ésima muestra de entrada conteniendo n características y $y_i \in R$ es la muestra de salida. Sin embargo, los datos de COVID-19 obtenidos para entrenar la metodología MLP-SVRES están en un formato de serie de tiempo $y_1, \dots, y_T$, donde y_t denota el tiempo de número de casos de COVID-19 del tipo susceptibles, confirmados y fallecidos en el tiempo t, y T es el tiempo total.

Capítulo 3

Por lo anterior es necesario transformar las series de tiempo de COVID-19 en un problema supervisado. En la Figura 9, se observa un ejemplo de la transformación de una serie de tiempo a un problema supervisado. La serie de tiempo presenta un tiempo $T = 76$, mientras que la transformación tiene un total de 94 instancias X con su clase y. Los datos transformados son utilizados para la sintonización y entrenamiento de los métodos de pronóstico SVR y MLP.

Determinado el formato de los datos de entrada, se procede a la modelación de SVR y MLP en función a estos. La regresión de vector de soporte (SVR) se basa en encontrar la mejor línea o curva que modele la tendencia de los datos de entrenamiento y sea capaz de generalizar el conocimiento para predecir los valores en el futuro. SVR, tomando los

Capítulo 3

principios de SVM, tiene como objetivo determinar una línea o curva que ayude a predecir los valores futuros.

Figura 9. Transformación de una serie de tiempo: a) serie de tiempo; b) problema supervisado

Time series

	(t)
1	548
2	643
3	920
4	1406
5	2075
6	2877
7	5509
8	6087
9	8141
10	9802
⋮	⋮
74	82543
75	82602
76	82665

(a)

Supervised problem with *lag (t-3)*

	X			y
	(t-3)	(t-2)	(t-1)	(t)
0	NA	NA	NA	548
1	NA	NA	548	643
2	NA	548	643	920
3	548	643	920	1406
4	643	920	1406	2075
5	920.0	1406.0	2075.0	2877.0
6	1406.0	2075.0	2877.0	5509.0
7	2075.0	2877.0	5509.0	6087.0
8	2877.0	5509.0	6087.0	8141.0
9	5509.0	6087.0	8141.0	9802.0
..	...	...	...	...
90	82198	82279	82361	82432
91	82279	82361	82432	82511
92	82361	82432	82511	82543
93	82432	82511	82543	82602
94	82511	82543	82602	82665

(b)

Capítulo 3

3.3 Modelo de ecuación diferencial

El modelo SIR consiste en la discretización de las ecuaciones dependientes del modelo SIR para ajustar el pronóstico obtenido de la fase 1. En esta sección se presenta la formulación del modelo matemático del modelo SIR para predecir casos de personas recuperadas de COVID-19 con propósitos de obtener el número reproductivo básico R_0, es decir la velocidad con la que se propaga la enfermedad. Suponemos que la población en el país de México esta dividida en tres clases: susceptibles (S), infectados (I) y recuperados (R); donde las clases están relacionadas a través de la tasa de infección β y el periodo infeccioso promedio γ, como se muestra en la Figura 11.

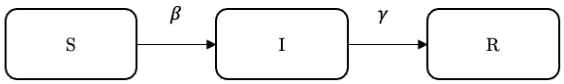

Figura 10. Modelo SIR

SIR es un modelo que compartimenta una población N en S, I y R para describir el ujo entre ellas. Sin embargo, la evaluación del modelo SIR puede ser muy complejo ya que es un modelo continuo por lo tanto es necesario trasformar o discretizar el modelo. Discretizar el modelo SIR nos permitirá analizar de manera mas sencilla al modelo matemático y encontrar una solución. Por lo anterior, se presenta y describe la discretización del modelo SIR para casos de COVID-19 para el país de México.

Capítulo 3

Se asume que la población total N se mantiene constante en el tiempo t o sea que es la suma de los compartimentos o casos susceptibles, infectados y recuperados como se muestra en la Ecuación 3.

$$N = S(t) + I(t) + R(t), \tag{3}$$

La tasa de contagio $c(t)$ esta dada por la Ecuación 4

$$c(t) = \beta I(t) N, \tag{4}$$

Donde

N es la población total, $I(t)$ es el número de infectados en el tiempo t, β la tasa de infeccion. Dados los planteamientos anteriores se procede a normalizar las variables en función del total de la población, como se muestra en la Ecuación 5.

$$S(t) = \frac{S(t)}{N}, I(t) = \frac{I(t)}{N}, \ R(t) = \frac{R(t)}{N} \tag{5}$$

Discretizando las ecuaciones del modelo SIR con respecto a t en ambos miembros de las expresiones, se obtiene las siguientes Ecuaciones.

Capítulo 3

(6)

(7)

(8)

- $\frac{d\,S}{d(t)} = -\beta\ *S\ *I$
- $\frac{d\,I}{d\,(t)} = \beta * S\ *I\ -\ \gamma\ *I$
- $\frac{d\,R}{d\,(t)} = \gamma\ *I$

Para las ecuaciones 9,10 y 11 se considera que nuestra d(t) y nuestras variables conocidas (S, I,R) pasan a estar en función del tiempo f (t).

Capítulo 3

- $\frac{d\,S}{d(t)} = -\beta(t) * S(t) * I(t)$ (9)

- $\frac{d\,I}{d\,(t)} = \beta(t) * S(t) * I(t) - \gamma(t) * I(t)$ (10)

- $\frac{d\,R}{d\,(t)} = \gamma(t) * I(t)$ (11)

- Para las ecuaciones 12,13 y 14 se considera un nuevo elemento de iteración (K).
- K = ∅ que representa un instante de tiempo.

- $\Delta S\,(K) = -\beta(K) * S(K) * I(K)$ $S_{k+1} - S_k = \beta_K * S_k + I_k$

- $\Delta I\,(K) = \beta(k) * S(K) * I(K) - \gamma(k) * I(k)$ $I_{k+1} - I_k = \beta_K * S_k * I_k - \gamma_k$

- $\Delta R\,(K) = \gamma(k) * I(K)$ $R_{k+1} - R_k = \gamma_k * I_k$

Condiciones Iniciales.

Capítulo 3

Se consideran las siguintes ecucaciones como las condiciones iniciales obtenidas después de la discretización.

$T = k = 0 ; S_k = S_0 = N$

- $I_k = I_0 = 0$

$$S_{k+1} - S_k = \beta_K * S_k + I_k$$

$$I_{k+1} - I_k = \beta_K * S_k * I_k - \gamma_k$$

$$R_{k+1} - R_k = \gamma_k * I_k$$

Aplicación de Condición inicial

- $S_1 - S_0 = \beta_0 * S_0 + I_0$ (a)
- $I_1 - I_0 = \beta 0 * S_0 * I_0 - \gamma 0$ (b)
- $R_1 - R_0 = \gamma 0 * I 0$ (c)
- Sustitución
- $S 1 = \beta 0 * S_0 + I_0 + S_0$
- $I_{1=} - \gamma 0$
- $R_1 - R_{0=} \gamma 0 * I_0$

Aplicamos las condiciones iniciales al dia 1. El cuál se considera como indeterminado debido a que matematicamente no es posible dividir entre cero.

Solución del dia 1

Capítulo 3

Ecuaciones

- $S_{1} = \beta 0 * S_{0} + I_{0} + S_{0}$
- $I_{I} = -\gamma 0$
- $R_{1} - R_{0} = \gamma 0 * I_{0}$

Valores Conocidos :

- $S_{0} = N ; I_{0} = 0 ; R_{0} = 0 ; R_{1}$
- $S_{1} = \beta 0 * N \therefore \beta 0 = \frac{S1}{N}$
- $I_{1} = -\gamma 0$
- $\gamma 0 = \frac{R_{1} - R0}{I0}$ indeterminado

Solución del dia 2.

Ecuaciones

$S_{k+1} - S_{k} = \beta_{K} * S_{k} + I_{k}$

$I_{k+1} - I_{k} = \beta_{K} * S_{k} * I_{k} - \gamma_{k}$

$R_{k+1} - R_{k} = \gamma_{k} * I_{k}$

K + 1 = 2

$S_{2} - S_{1} = \beta_{1} * S_{I} + I_{1}$

$I_{2} - I_{1} = \beta_{1} * S_{1} * I_{1} - \gamma_{1}$

$R_{2} - R_{1} = \gamma_{1} * I1 \therefore \gamma_{1} = \frac{R2 - R1}{I1}$

$\beta_{1} * S_{1} * I_{1} = \frac{I2 - I1}{\gamma_{1}} \therefore \beta_{1} = -\frac{I2 - I1}{\gamma 1 * S1 * I1}$

Fase 3

Capítulo 3

3.4 Método de Mejora

El método de mejora consiste en utilizar el método de ES como método de mejora, con el propósito de perfeccionar el pronóstico obtenido por el ajuste en la fase 2.

Las métricas de evaluación empleadas para medir el desempeño de la metodología propuesta son error porcentual absoluto medio (MAPE) y Error porcentual de la media absoluta simétrica (SMAPE), que son las métricas mas utilizadas en base al estado del arte. El MAPE, también conocido como desviación promedio absoluta porcentual (MAPE), mide la exactitud de un método para la construcción ajustada de valores de serie de tiempo en estadísticas.

En este trabajo, las métricas MAPE y SMAPE son utilizadas para medir el desempeño del modelo. Las métricas están definidas en la Ecuación y .

$$\text{MAPE} = \frac{100}{N} \times + \sum_{i=1}^{N} \quad \left|\frac{x_i - \hat{x_i}}{x_i}\right|,$$

$$\text{SMAPE} = \frac{200}{N} \times + \sum_{i=1}^{N} \quad \frac{|x_i - \hat{x_i}|}{[x_i] + [\hat{x_i}]},$$

Capítulo 3

En relación a las series de tiempo empleadas en este estudio, es importante destacar que se ha llevado a cabo un proceso de ajuste en las series originales, obtenidas a partir de la página oficial de datos de COVID-19 en México, que incluyen datos sobre Confirmados, Sospechosos, Negativos y Defunciones. El objetivo de este ajuste es facilitar la comprensión del lector y mejorar la precisión en el uso de los datos.

En este sentido, se ha realizado un cambio en el nombre de la serie Defunciones, que ahora se presenta bajo la denominación de Fallecidos. Por otro lado, la serie Confirmados se ha mantenido intacta en cuanto a su nombre y contenido.

En lo que respecta a las series Sospechosos y Negativos, se ha aplicado una resta entre ambas, tomando en consideración los datos obtenidos diariamente. Esta operación permite obtener una nueva serie de datos denominada Susceptibles.

Capítulo 3

En resumen, se ha llevado a cabo un ajuste en las series de tiempo con el fin de facilitar la interpretación y análisis de los datos por parte del lector, y mejorar la precisión en la utilización de los mismos en este estudio.

Además, es importante señalar que este proceso de ajuste ha sido realizado con rigurosidad y siguiendo los criterios de las autoridades sanitarias competentes. De esta forma, se garantiza que los datos obtenidos sean precisos y fiables, lo que a su vez, permitirá realizar un análisis más completo y riguroso de la situación de la pandemia en México.

Cabe destacar que la serie de datos Fallecidos se ha ajustado debido a que el término "defunciones" puede resultar poco comprensible para algunos lectores, mientras que el término "fallecidos" es más claro y conciso.

De esta manera, se busca evitar confusiones y mejorar la comprensión de la información presentada. En el caso de las series Sospechosos y Negativos, se ha realizado una resta para obtener la serie Susceptibles, ya que esta última es una variable de gran importancia para el análisis de la propagación del virus y la evolución de la pandemia.
En definitiva, el proceso de ajuste en las series de tiempo presentadas en este trabajo permite mejorar la calidad de los datos y su interpretación, lo que resulta fundamental para la toma de decisiones y la implementación de políticas públicas efectivas en la lucha contra la pandemia de COVID-19 en México.

Capítulo 3

Capítulo 4

4 Análisis y Resultados

Se desarrollaron dos experimentos con resultados diferentes. En el primer experimento se muestra el modelo MLP-SVRES donde se utilizan dos clasificadores como método primario de predicción de casos confirmados de COVID-19 y que consta de tres fases. Los clasificadores utilizados en este trabajo son un Perceptron Multicapa (MLP) y el método de Support Vector Regression (SVR).

- Fase 1: Empezamos transformando la serie de tiempo para obtener los datos de confirmados diarios por dia en todo el país (Y) para después utilizar los clasificadores y obtener un primer pronóstico.
- Fase 2: Con el pronóstico de Y podemos utilizar las ecuaciones diferenciales discretizadas para obtener los valores de R, N, Lambda y Beta que son dependientes de Y.
- Fase 3: Después de conocer los resultados de nuestros parámetros se utiliza el método de suavizamiento exponencial para ajustar los resultados obtenidos. Se evalua mediante la métrica de MAPE

Capítulo 4

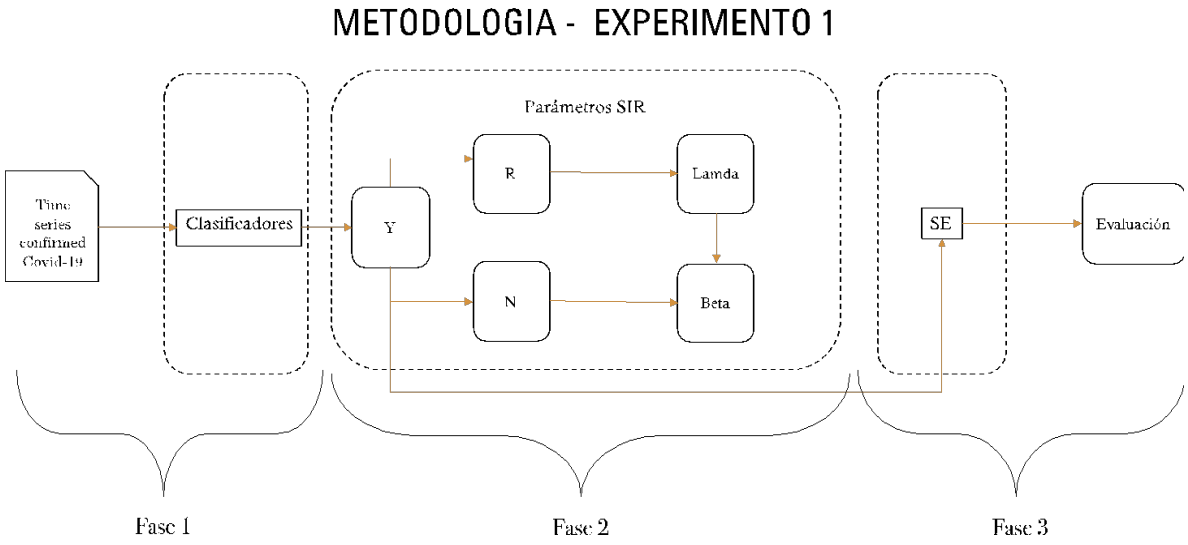

Figura 11. Modelo MLP- SVRES

Capítulo 4

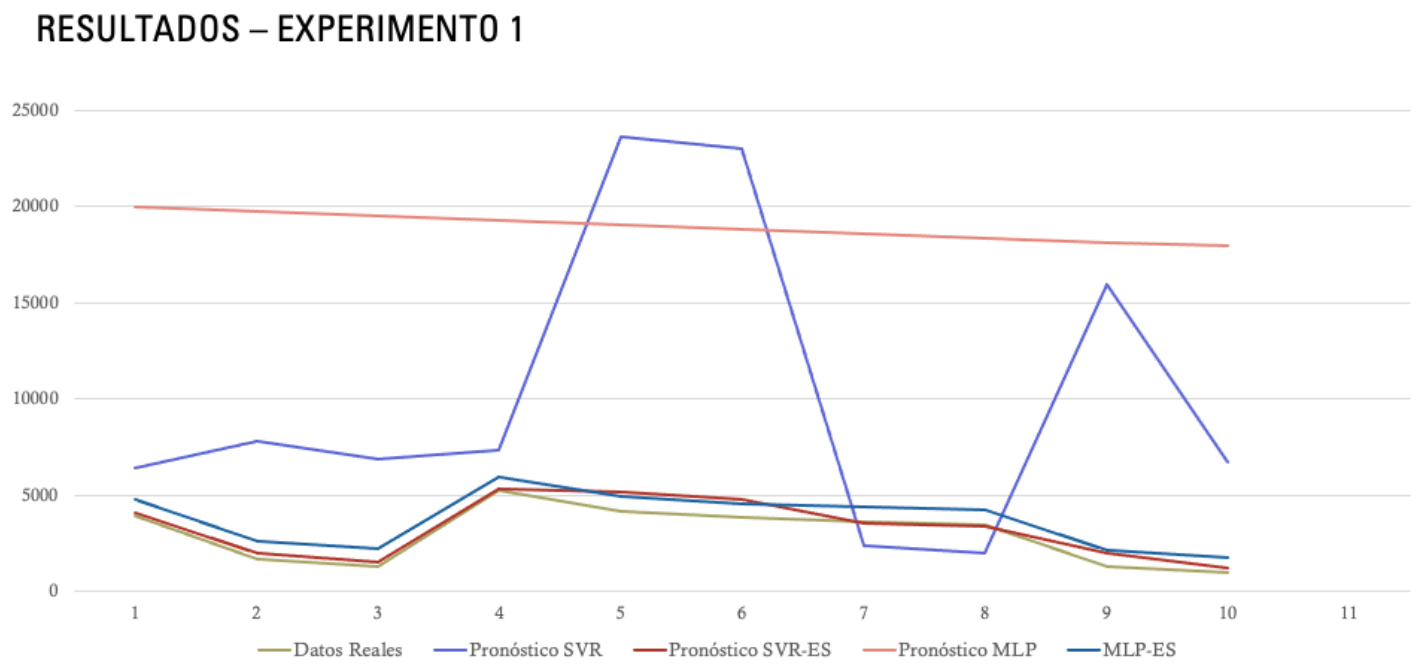

Figura 13. Grafica de resultados, experimento 1.

RESULTADOS- EXPERIMENTO 1

Datos Reales Prueba		Dia Pronosticado	Pronóstico SVR	Pronóstico SVR-ES	Pronóstico MLP	MLP-ES
83	3980	1	6437	4102.85	19961.16661	4779.05833
84	1705	2	7839	2011.7	19737.31703	2606.61585
85	1296	3	6928	1577.6	19513.46744	2206.87337
86	5252	4	7337	5356.25	19289.61786	5953.88089
87	4206	5	23617	5176.55	19065.76828	4948.98841
88	3847	6	22985	4803.9	18841.9187	4596.74594
89	3654	7	2419	3592.25	18618.06912	4402.20346
90	3485	8	2032	3412.35	18394.21954	4230.46098
91	1286	9	15931	2018.25	18170.36996	2130.2185
92	966	10	6759	1255.65	17946.52038	1815.02602

	SVR	SVR-ES	MLP	MLP-ES
MAPE	34.46388334	11.22036326	63.86658873	12.690851
R0	0.052035885	-2.970652067	-19.91552432	-3.9691263
Diferencia	3.723216885	0.700528933	-16.24434332	-0.2979453

Figura 14. Tabla de resultados, experimento 1.

Metodologia Experimento 2

- Fase 1: Empezamos transformando la serie de tiempo para obtener los datos de confirmados diarios por dia en todo el país (Y) para después utilizar los clasificadores y obtener un primer pronóstico.
- Fase 2: Con el pronóstico de Y, F*, S* y R podemos utilizar las ecuaciones diferenciales discretizadas para obtener los valores de N, Lambda y Beta.
- Fase 3: Después de conocer los resultados de nuestros parámetros se utiliza el método de suavizamiento exponencial para ajustar los resultados obtenidos. Se evalua mediante la métrica de MAPE

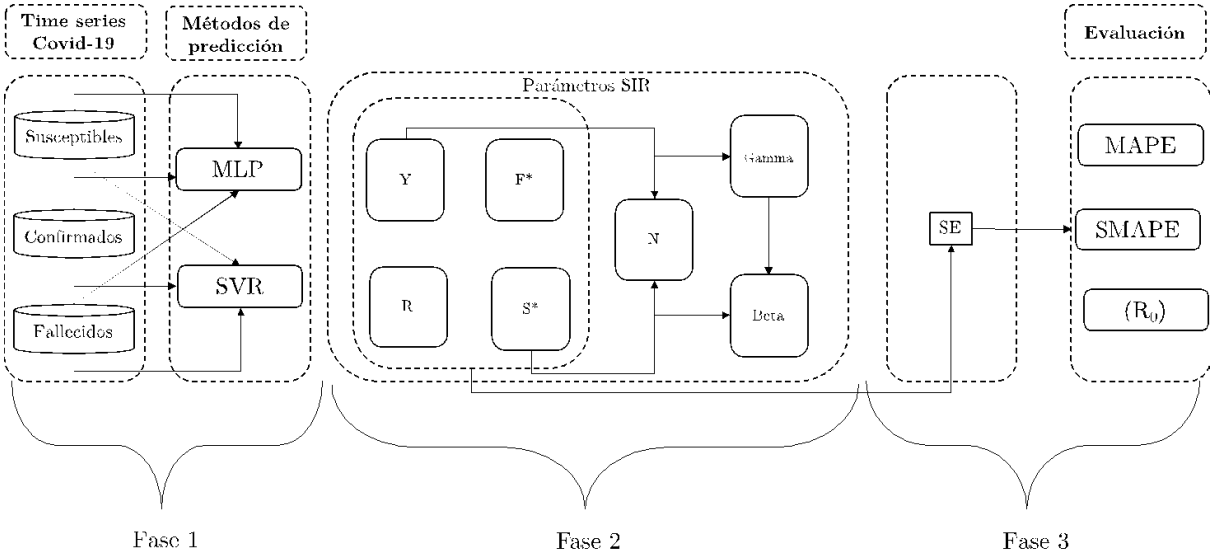

Figura 12. Modelo MLP-SVRESD

Se presentan los resultados de los dos experimentos antes mencionados. Se evalua mediante MAPE y SMAPE. En la gráfica se indica los datos de prueba utilizados en comparación de los datos predecidos. Se observa también el método utilizados y si se aplica el suavizamiento exponencial se coloca [SE].

RESULTADOS - EXPERIMENTO 2

Dia	Casos Confirmados Diarios		Dia Pronosticado	Pronóstico SVR	Pronóstico SVR-ES	Pronóstico MLP	MLP-ES
83	11238	*	1	3540.2918	10853.11459	4864	10919.3
84	10801	*	2	1991.2918	10360.51459	4972	10509.6
85	4022	*	3	7362.2918	4189.01459	5129	4077.35
86	2522	*	4	11763.3782	2984.06891	7830	2787.4
87	11803	*	5	12873.3989	11856.51995	7992	11612.5
88	12295	*	6	11982.2867	12279.36434	8218	12091.2
89	12097	*	7	8679.2863	11926.11432	8346	11909.5
90	11504	*	8	7945.7081	11326.08541	8562	11356.9
91	9734	*	9	3195.7081	9407.085405	5621	9528.35
92	3620	*	10	2108.7081	3544.435405	4182	3648.1

	SVR	SVR-ES	MLP	MLP-ES
MAPE	77.925582	3.89628	52.848545	2.48212
SMAPE	33.23	1.88	24.6	1.3

Figura 15. Grafica de resultados para el pronóstico de Sanados, experimento 2.

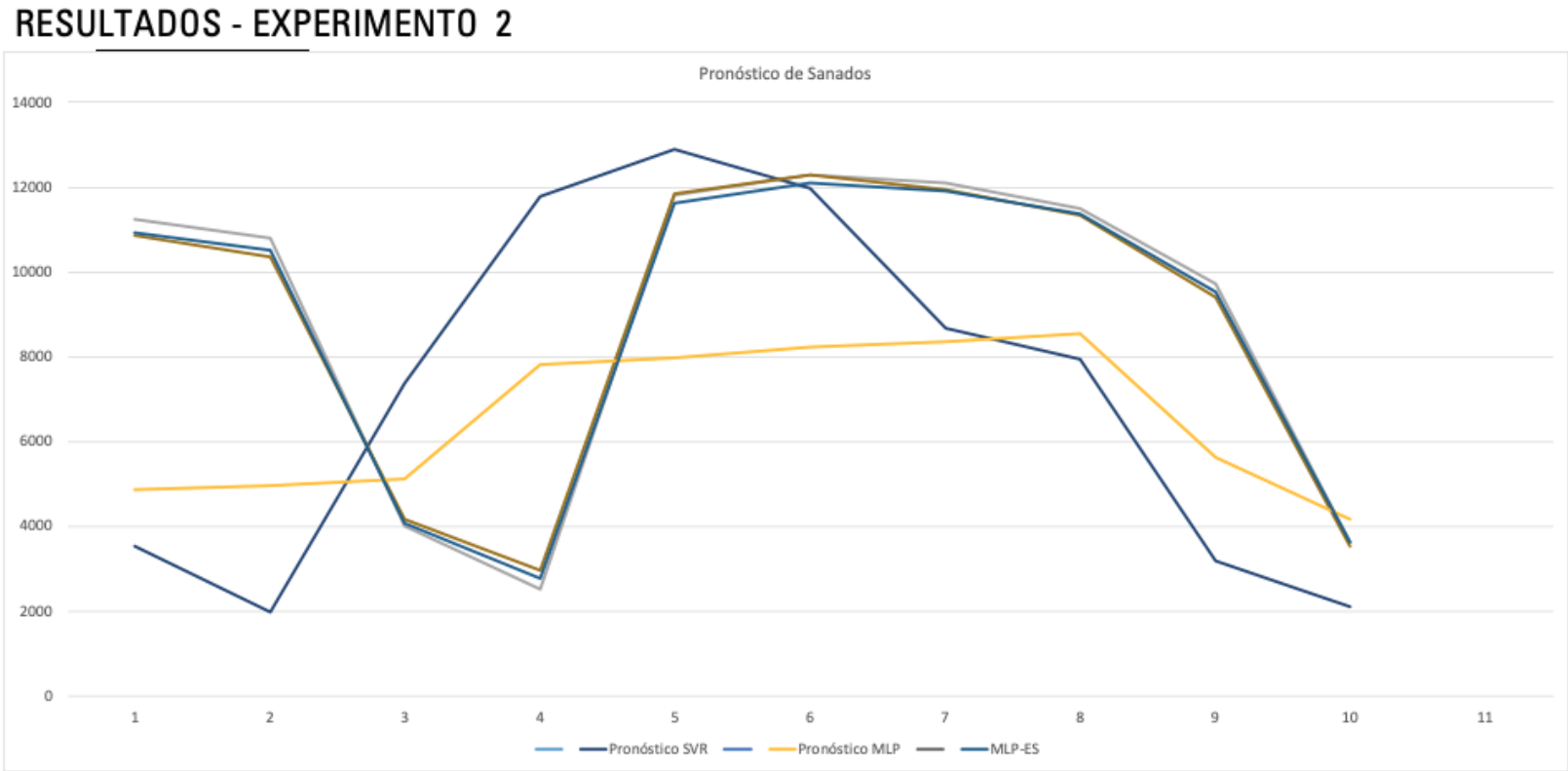

Figura 16. Tabla de resultados para el pronóstico de Sanados, experimento 2.

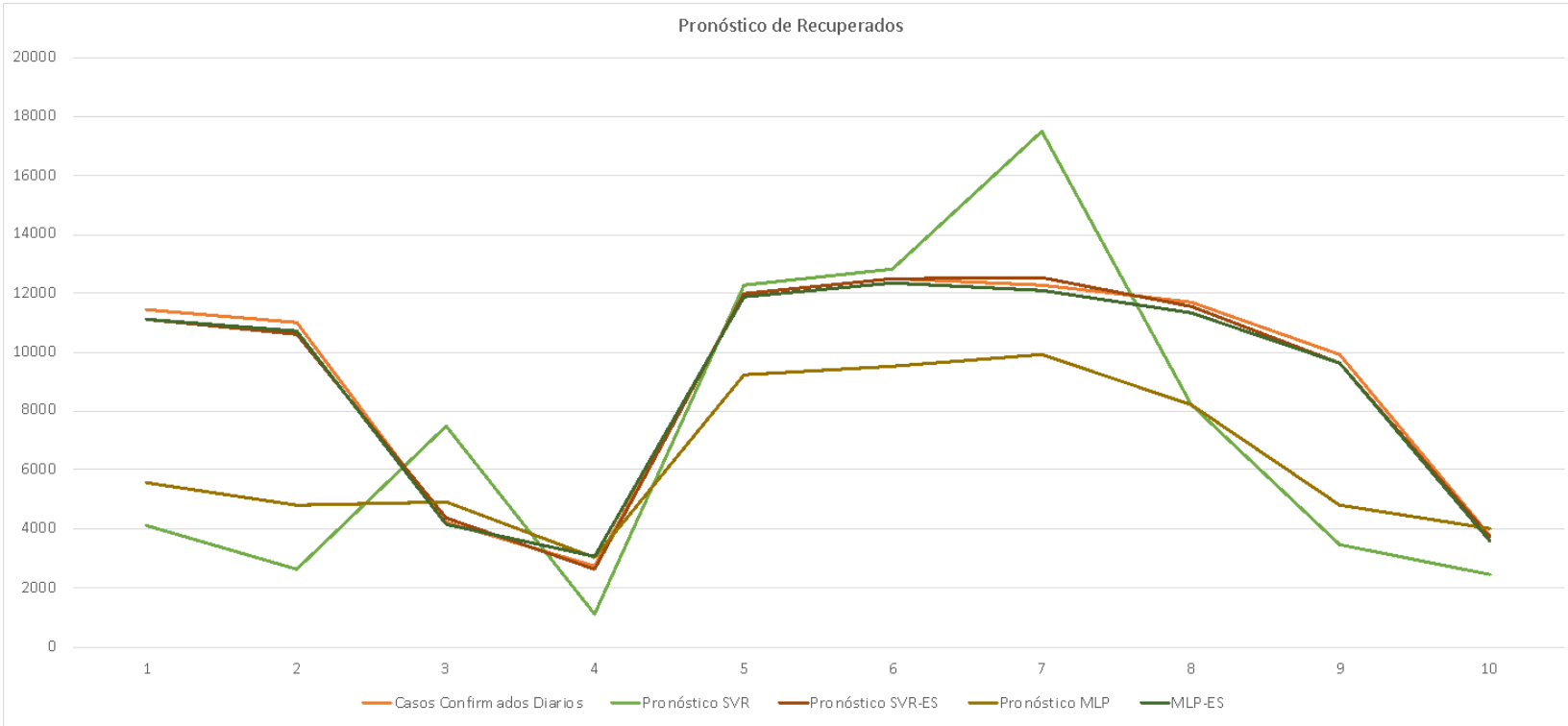

Figura 17. Grafica de resultados para el pronóstico de Recuperados, experimento 2.

Dia	Casos Confirmados Diarios		Dia Pronosticado	Pronóstico SVR	Pronóstico SVR-ES	Pronóstico MLP	MLP-ES
83	11450	*	1	4110.2918	11083.01459	5573	11117.1
84	11017	*	2	2640.6811	10598.18406	4792	10711.2
85	4218	*	3	7496.8326	4381.94163	4901	4158.05
86	2727	*	4	1097.1876	2645.50938	3019	3050.75
87	11987	*	5	12289.8719	12002.1436	9202	11864
88	12478	*	6	12809.2791	12494.56396	9527	12350.6
89	12285	*	7	17503.9173	12545.94587	9930	12081.35
90	11707	*	8	8235.9131	11533.44566	8212	11363.1
91	9908	*	9	3482.8267	9586.741335	4829	9613.15
92	3767		10	2423.3421	3699.817105	4011	3578.65

	SVR	SVR-ES	MLP	MLP-ES
MAPE	45.546397	2.516317778	24.025463	3.488946
SMAPE	28.63	1.15	17.88	1.66

Figura 18. Tabla de resultados para el pronóstico de Fallecidos, experimento 2.

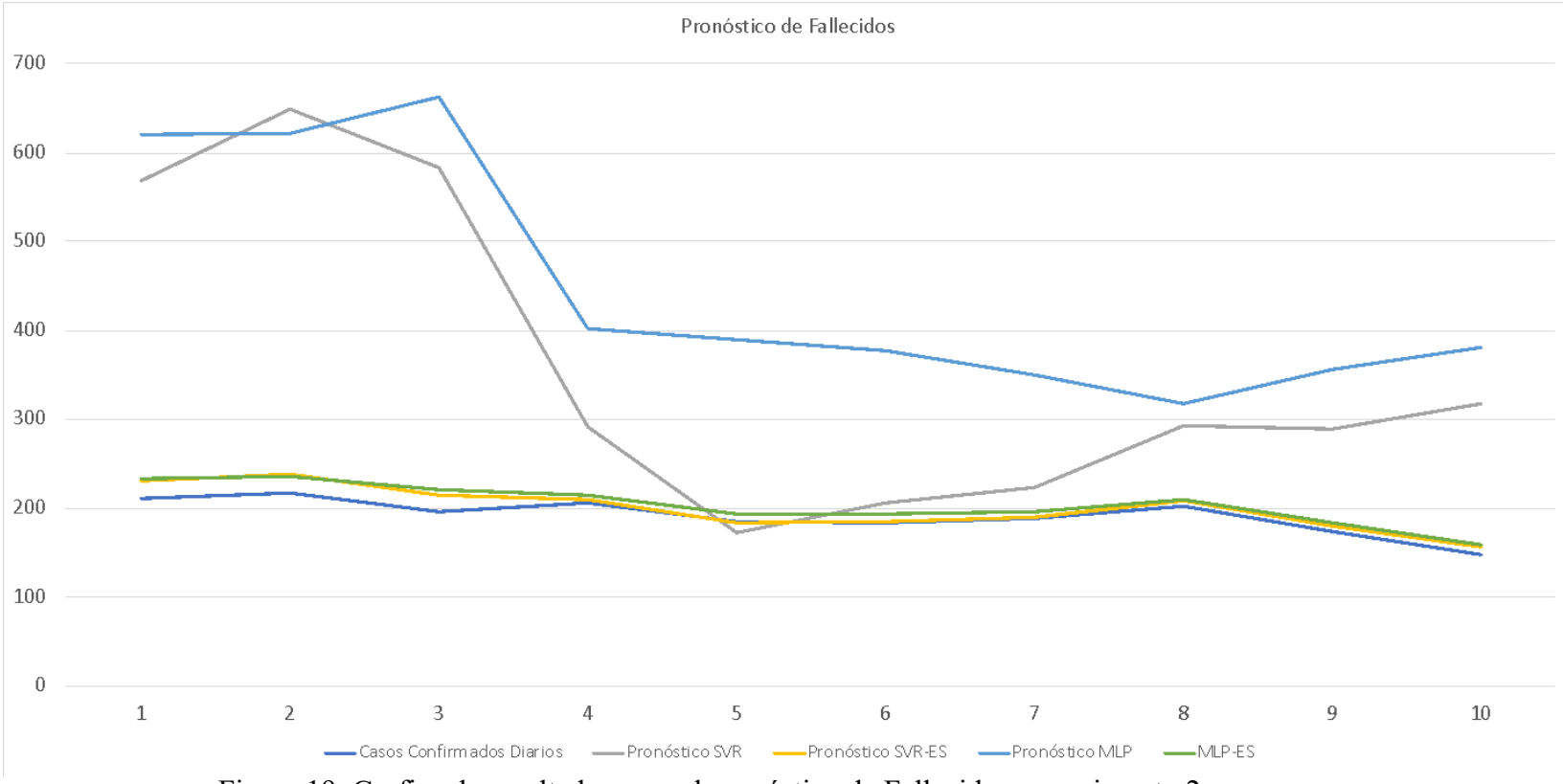

Figura 19. Grafica de resultados para el pronóstico de Fallecidos, experimento 2.

Dia	Casos Confirmados Diarios	Dia Pronosticado	Pronóstico SVR	Pronóstico SVR-ES	Pronóstico MLP	MLP-ES
83	212	1	568.2462	229.8123	619	232.35
84	216	2	647.2918	237.5645	622	236.3
85	196	3	582.0873	215.3043	662	219.3
86	205	4	289.9129	209.2456	402	214.85
87	184	5	173.0832	183.4541	389	194.25
88	183	6	205.3197	184.1159	378	192.75
89	188	7	222.9428	189.7471	349	196.05
90	203	8	291.7081	207.4354	318	208.75
91	174	9	288.1381	179.7069	357	183.15
92	147	10	316.918	155.4959	381	158.7

	SVR	SVR-ES	MLP	MLP-ES
MAPE	86.77189	4.338579	133.84118	6.692059
SMAPE	25.9	2.09	38.56	3.22

Figura 20. Tabla de resultados para el pronóstico de Fallecidos, experimento 2.

Capítulo 5

5 Conclusiones y Recomendaciones

En este trabajo se utilizaron diferentes métodos de predicción, como el SVR (Support Vector Regression) y el MLP (Multilayer Perceptron), para analizar y predecir datos relacionados con un tema específico. Durante el estudio, se encontró que el SVR proporcionó los mejores resultados en términos de precisión y rendimiento.

El SVR, basado en el aprendizaje de máquina, se destacó por su capacidad para capturar relaciones no lineales en los datos y ajustarse a patrones complejos. Esto permitió obtener predicciones más precisas en comparación con el MLP, que es otro algoritmo ampliamente utilizado en la predicción.

Además, es importante mencionar que se utilizaron datos específicos de México para llevar a cabo el análisis. Al utilizar datos locales, se pudo tener en cuenta las particularidades y características propias de este país, lo que contribuyó a mejorar la calidad de las predicciones.

En resumen, este trabajo escolar demostró que el método SVR fue el más eficaz y preciso para predecir los datos analizados, superando al MLP. Asimismo, el uso de datos de México permitió obtener resultados más contextualizados y relevantes para este país. Estos hallazgos resaltan la importancia de utilizar enfoques de aprendizaje automático y considerar la relevancia local al realizar análisis y predicciones en diferentes contextos.

6 Bibliografía

Makridakis, S. W. (1998). Forecasting: methods and applications (Vol. 2). *John Wiley & Sons.*

L. Ying, D. T. (2016). “A comparative study of prediction models for PM2.5 based on extreme learning machine and decision tree”. *Journal of Cleaner Production, vol. 145*, pp. 95–103,.

J. G. C. M. Smit, K. J. (2002). “An evaluation of performance measures for regression models". *Journal of Applied Statistics, vol. 29*, 111-124.

A. R. Delen, C. G. (2009). “A critical evaluation of the use of mean squared error and mean absolute deviation in measuring forecast accuracy". *International Journal of Forecasting. Vol.25*, 217–228,.

Makridakis, S. W. (1998). Forecasting: methods and applications (2nd ed.). . *New York: Wiley.*

Koehler, J. a. (1989). "Forecast Verification". *International Journal of Forecasting,* , 559-581.

Hastie, T. T. (2009). The elements of statistical learning: data mining, inference, and prediction. *Springer*.

Bishop, C. M. (2006). Pattern recognition and machine learning (Vol. 1). *Springer*.

Cortes, C. &. (1995). Support-vector networks. Machine learning. 273-297.

Schölkopf, B. &. (2002). Learning with kernels: support vector machines, regularization, optimization, and beyond. . *MIT press.*

Smola, A. J. (2004). A tutorial on support vector regression. *Statistics and computing, 14(3).*, 199-222.

Castillo-Chavez, C. &. (2002). Intraspecific competition, dispersal and disease dynamics in discrete-time patchy environments. . *En Mathematical Approaches for Emerging and Reemerging Infectious Diseases: An Introduction to Models, Methods and Theory.*

Anexo B

Hethcote, H. (1989). Three Basic Epidemiological Models. In: Levin, S.A., Hallam, T.G., Gross, L.J. (eds) Applied Mathematical Ecology. Biomathematics, vol 18. . *Springer, Berlin, Heidelberg.* .

White LF, P. M. (2008). Transmissibility of the influenza virus in the 1918 pandemic. PLoS One. .

Mexico., 1. S. (1 de Noviembre de 2021). *Datos Abiertos Dirección General de Epidemiología.* Obtenido de Datos Abiertos Dirección General de Epidemiología.: https://www.gob.mx/salud/documentos/datos-

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review, 65*, 386–408 .

Vandewalle, 1. J. (1999). “Least squares support vector machine classifiers". *Neural Processing Letters, vol. 9*, 293-300.

7 ANEXOS

Códigos Utilizados

a. Anexo 1. Código SVR - GA

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
import pandas as pd
from pandas import datetime
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from sklearn.metrics import mean_absolute_error
from random import randint
#import collections
import math
from sklearn.preprocessing import StandardScaler


def sumar(lista):
    suma=0
    for elemento in lista:
        suma += elemento
    return suma

def evaluar_MAPE(test, predictions):
    Mape=0.0
    errores=[]
    for i in range(0, len(test)):
        if(test[i]!=0):
            errores.append((math.fabs(test[i]-predictions[i]))/math.fabs(test[i]))
    if(len(errores)!=0):
        Mape=100*(sumar(errores)/len(errores))

    return Mape
    #print('Test MAPE: %.3f' % Mape)
```

```python
#Construir una poblacion(generar soluciones aleatorias)

class Solutions:
    #Individual
    def __init__(self, n): #constructor de solucion ## guines bajos dobles para ocultar, no mover esa parte
        #crear binario aleatroio de tamaño n
        kernel_random = kernel[np.random.randint(0, len(kernel))]
        C_svr_random = C[np.random.randint(0, len(C))]
        E_svr_random = epsilon[np.random.randint(0, len(epsilon))]
        self.value = [kernel_random, C_svr_random , E_svr_random]
        self.mape = 0.0
```

Figura A.1. Librerías, función del MAPE y estructura del GEN. Parte (1/7)

```python
#Evaluar la poblacion dependiendo el problema (un metodo)
def evaluarmape(ind):
    mape = 0.0
    K=ind.value[0]
    c=ind.value[1]
    E=ind.value[2]
    size1 = int(len(X_svr)-dias_pronosicados)
    size2 = int(len(y_svr)-dias_pronosicados)
    X_train, X_test, y_train, y_test = X_svr[0:size1], X_svr[0:size1], y_svr[0:size2], y_svr[0:size2]
    if (K == 'poly'):
        svr = SVR(kernel = K, C = c, epsilon = E, gamma='auto', degree=2, coef0 = 1.0)#np.random.randint(2, 5)
        svr.fit(X_train, y_train)
        Y_pred = svr.predict(X_test)
        ind.mape = evaluar_MAPE(sc_y.inverse_transform(Y_pred), sc_y.inverse_transform(y_test))
    else:
        if (K == 'linear'):
            svr = SVR(kernel = K, C = c, epsilon = E, gamma='auto')
            svr.fit(X_train, y_train)
            Y_pred = svr.predict(X_test)
            ind.mape = evaluar_MAPE(sc_y.inverse_transform(Y_pred), sc_y.inverse_transform(y_test))

        else:
            if (K == 'rbf'):
                svr = SVR(kernel = K, C = c, epsilon = E, gamma='auto', max_iter = -1)
                svr.fit(X_train, y_train)
                Y_pred = svr.predict(X_test)
                ind.mape = evaluar_MAPE(sc_y.inverse_transform(Y_pred), sc_y.inverse_transform(y_test))
            else:
                svr = SVR(kernel = K, C = c, epsilon = E, gamma='scale')
                svr.fit(X_train, y_train)
                Y_pred = svr.predict(X_test)
                ind.mape = evaluar_MAPE(sc_y.inverse_transform(Y_pred), sc_y.inverse_transform(y_test))

    return ind.mape
```

```python
def selection(populat, n_pairs):
#    print(population)
#    print(n_pairs)
    padres1=[]
    padres2=[]
    parents_pairs=[]
    for i in range(n_pairs):
        parents= np.random.choice(populat, 4,replace=False) #selecciono 4 padres de la lista de soluciones
        if(parents[0].mape < parents[1].mape):
            p1=parents[0]
        else:
            p1=parents[1]
        print(p1)
        if(parents[2].mape < parents[3].mape):
            p2=parents[2]
        else:
            p2=parents[3]
        parents_pairs.append((p1,p2))
        padres1.append(p1)
        padres2.append(p2)
#    print (parents_pairs)
    return parents_pairs, padres1, padres2
```

Figura A.2. Función Fitness y selección por ruleta. Parte (2/7)

```python
def crossover(parents_pairs):
    childs=[]
    for p1,p2 in parents_pairs:
        mask=np.random.randint(0,2,n)
        c1=Solutions(n)
        c2=Solutions(n)
        for i in range(len(mask)):
            if (mask[i]==0):
                c1.value[i]=p1.value[i]
                c2.value[i]=p2.value[i]
            else:
                c1.value[i]=p2.value[i]
                c2.value[i]=p1.value[i]
        if(c1.value==p1.value):
            c1.mape=p1.mape
        else:
            if(c1.value==p2.value):
                c1.mape=p2.mape
            else:
                evaluarmape(c1)
        if(c2.value==p1.value):
            c2.mape=p1.mape
        else:
            if(c2.value==p2.value):
                c2.mape=p2.mape
            else:
                evaluarmape(c2)
        childs.append(c1)
        childs.append(c2)

    return childs
```

```python
def mutation(populat):
#    kernel=['rbf', 'linear']
#    C=[0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4]
#    epsilon=[0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.11, 0.12, 0.13, 0.14]
    for sol in np.random.choice(populat, mutations, replace=True):
    #for sol in np.random.choice(population, percent_mutations, replace=True):
        kernel_random = kernel[np.random.randint(0, len(kernel))]
        C_svr_random = C[np.random.randint(0, len(C))]
        E_svr_random = epsilon[np.random.randint(0, len(epsilon))]
        sol.value = [kernel_random, C_svr_random , E_svr_random]
        evaluarmape(sol)
```

Figura A.3. Función de cruza y mutación. Parte (3/7)

```python
def sintonizacion_SVR_AG(pob_evaluada):
    suma_fit_gen = []
    fitpromedio_gen = []
    mejor_sol_global = []
    mejor_fit_global = 1e+21
    con_gen_sinmejora = 0
    sujeto_campeon = []
    MAPE=0
    MAPE1=0
    n_generacions = 1
    itr = 1
    while (itr <= n_generations and con_gen_sinmejora <= 5): #or con_gen_sinmejora < 3
        #seleccion de padres
        parents, padres1, padres2=selection(pob_evaluada, individuos//2)

        #crear hijos con los padres
        childs=crossover(parents)

        pob_evaluada.extend(childs)



        #mutar hijos (probabilidad)
        mutation(childs)


        #evaluar hijos
        #unir hijos con padres (nueva poblacion)
        pob_evaluada.extend(childs) #concatenamos las listas

        #ordenar poblacion
        pob_evaluada.sort(key=lambda x: x.mape, reverse=False)
```

Figura A.4. Función principal de sintonización de parámetros. Parte (4/7)

```
    #acotar poblacion
    pob_evaluada= pob_evaluada[:individuos*3]

    suma_fit= 0
    fitpromedio = 0
    for sol in population: #evaluo para calcular su fitnes en cada generacion
        suma_fit += sol.mape
        fitpromedio = suma_fit/len(population)
    suma_fit_gen.append(suma_fit)
    fitpromedio_gen.append(fitpromedio) #fitness promedio de la generacion
    sujeto_campeon= population[0]
    if (sujeto_campeon.mape >= mejor_fit_global):
        con_gen_sinmejora = con_gen_sinmejora+1
    else:
        con_gen_sinmejora = 0
        mejor_sol_global = sujeto_campeon.value
        mejor_fit_global = sujeto_campeon.mape

    print('Generacion: ', n_generacions, len(population))
    for sol in population: #imprime cada generacion
        print(sol.value,sol.mape)

    print("Parametros y MAPE del sujeto campeon: ", sujeto_campeon.value, sujeto_campeon.mape)
    itr += 1

    n_generacions +=1
print("Mejores parametros y MAPE global: ", mejor_sol_global, mejor_fit_global)
print('numero de generaciones sin mejora: ', con_gen_sinmejora)


size1 = int(len(X_svr)-dias_pronosicados)
size2 = int(len(y_svr)-dias_pronosicados)
X_train, X_test, y_train, y_test = X_svr[0:size1], X_svr[0:size1], y_svr[0:size2], y_svr[0:size2]
```

```
if(mejor_sol_global[0] == 'poly'):
    svr = SVR(kernel = mejor_sol_global[0], C = mejor_sol_global[1], epsilon = mejor_sol_global[2], gamma='auto', degree=2, coef0 = 1.0)
    svr.fit(X_train, y_train)
    Y_pred = svr.predict(X_svr[len(X_svr)-dias_pronosicados:len(X_svr)])
    MAPE = evaluar_MAPE(sc_y.inverse_transform(Y_pred), sc_y.inverse_transform(y_svr[len(y_svr)-dias_pronosicados:len(y_svr)]))


else:
    if(mejor_sol_global[0] == 'linear'):
        svr = SVR(kernel = mejor_sol_global[0], C = mejor_sol_global[1], epsilon = mejor_sol_global[2], gamma='auto')
        svr.fit(X_train, y_train)
        Y_pred = svr.predict(X_svr[len(X_svr)-dias_pronosicados:len(X_svr)])
        MAPE = evaluar_MAPE(sc_y.inverse_transform(Y_pred), sc_y.inverse_transform(y_svr[len(y_svr)-dias_pronosicados:len(y_svr)]))

    else:
        if (mejor_sol_global[0] == 'rbf'):
            svr = SVR(kernel = mejor_sol_global[0], C = mejor_sol_global[1], epsilon = mejor_sol_global[2], gamma='auto', max_iter = -1)
            svr.fit(X_train, y_train)
            Y_pred = svr.predict(X_svr[len(X_svr)-dias_pronosicados:len(X_svr)])
            MAPE = evaluar_MAPE(sc_y.inverse_transform(Y_pred), sc_y.inverse_transform(y_svr[len(y_svr)-dias_pronosicados:len(y_svr)]))



        else:
            svr = SVR(kernel = mejor_sol_global[0], C = mejor_sol_global[1], epsilon = mejor_sol_global[2], gamma='scale')
            svr.fit(X_train, y_train)
            Y_pred = svr.predict(X_svr[len(X_svr)-dias_pronosicados:len(X_svr)])
            MAPE = evaluar_MAPE(sc_y.inverse_transform(Y_pred), sc_y.inverse_transform(y_svr[len(y_svr)-dias_pronosicados:len(y_svr)]))




size1 = int(len(X_svr)-dias_pronosicados)
size2 = int(len(y_svr)-dias_pronosicados)
X_train, X_test, y_train, y_test = X_svr[0:size1], X_svr[int((size1)*0.90):size1], y_svr[0:size2], y_svr[int((size2)*0.90):size2]#[int((s
```

Figura A.5. Segunda parte de la función principal. Parte (5/7)

```
    if(mejor_sol_global[0] == 'poly'):
        svr = SVR(kernel = mejor_sol_global[0], C = mejor_sol_global[1], epsilon = mejor_sol_global[2], gamma='auto', degree=2, coef0 = 1.0)
        svr.fit(X_train, y_train)
        Y_entren_pred = svr.predict(X_test)
        MAPE1 = evaluar_MAPE(sc_y.inverse_transform(Y_entren_pred), sc_y.inverse_transform(y_test))

    else:
        if(mejor_sol_global[0] == 'linear'):
            svr = SVR(kernel = mejor_sol_global[0], C = mejor_sol_global[1], epsilon = mejor_sol_global[2], gamma='auto')
            svr.fit(X_train, y_train)
            Y_entren_pred = svr.predict(X_test)
            MAPE1 = evaluar_MAPE(sc_y.inverse_transform(Y_entren_pred), sc_y.inverse_transform(y_test))

        else:
            if (mejor_sol_global[0] == 'rbf'):
                svr = SVR(kernel = mejor_sol_global[0], C = mejor_sol_global[1], epsilon = mejor_sol_global[2], gamma='auto', max_iter = -1)
                svr.fit(X_train, y_train)
                Y_entren_pred = svr.predict(X_test)
                MAPE1 = evaluar_MAPE(sc_y.inverse_transform(Y_entren_pred), sc_y.inverse_transform(y_test))


            else:
                svr = SVR(kernel = mejor_sol_global[0], C = mejor_sol_global[1], epsilon = mejor_sol_global[2], gamma='scale')
                svr.fit(X_train, y_train)
                Y_entren_pred = svr.predict(X_test)
                MAPE1 = evaluar_MAPE(sc_y.inverse_transform(Y_entren_pred), sc_y.inverse_transform(y_test))


    print()
    print('DATOS DEL MEJOR MODELO VECTORES DE SOPORTE REGRESIÓN')
    print()
    print('Precisión del mejor modelo:')
    print('score del mejor modelo: ', svr.score(X_train, y_train))
    print('MAE del mejor modelo: ', mean_absolute_error(sc_y.inverse_transform(Y_pred), sc_y.inverse_transform(y_svr[len(y_svr)-dias_pronosicados:len(y_svr)])))
    print('MAPE del mejor modelo: ', MAPE)

    print("Mejores parametros y MAPE global: ", mejor_sol_global, mejor_fit_global)
    print('Valores reales', sc_y.inverse_transform(y_test))
    print('Valores pronosticados', sc_y.inverse_transform(Y_pred))
    print('Valores pronosticados', sc_y.inverse_transform(Y_entren_pred))
```

```
if __name__== "__main__":
    ############Parametros#############################################
    documento = 'time_series_covid_19_confirmed4.csv'
    n=3
    individuos = 100
    mutations = int(individuos*100/100)
    n_generations = 10
    kernel=['rbf', 'linear', 'sigmoid', 'poly'] #'rbf', 'linear', 'sigmoid', 'poly' 'precomputed']
    C=[0.0001, 0.001, 0.005, 0.006, 0.007, 0.008, 0.009, 0.01, 0.015, 0.02, 0.025, 0.026, 0.027, 0.028, 0.029, 0.030, 0.031, 0.032, 0.033, 0.034, 0.035, 0.04, 0.045,
    epsilon=[0.0001, 0.001, 0.005, 0.006, 0.007, 0.008, 0.009, 0.01, 0.015, 0.02, 0.025, 0.026, 0.027, 0.028, 0.029, 0.030, 0.031, 0.032, 0.033, 0.034, 0.035, 0.04,

    ###################################################
    df = pd.read_csv(documento)
    dias_pronosicados=10
    mejores_parametros=[]
    MAPE_del_test=[]
    valores_reales=[]
    valores_pronosticados=[]
    conjunt_entrenamiento_pronost=[]
    Country_Region=[]
    indice=[]
    MAPE_entrenamientos = []
    mape_del_pronostico = []
    X = df.iloc[0, 5:df.shape[1]].values
    Y = df.iloc[83, 5:df.shape[1]].values
    sc_X = StandardScaler()
    sc_y = StandardScaler()
    X_svr = sc_X.fit_transform(X.reshape(-1, 1))
    Y_svr = sc_y.fit_transform(Y.reshape(-1, 1))
    y_svr= Y_svr

    Country_Region.append(df.iloc[83, 2])
    indice.append(df.iloc[83, 0])
```

Figura A.6. Tercera parte de la función principal y el “Main”. Parte (6/7)

```
    """Generar poblacion inicial"""
    population_size=individuos
    population = []
    for i in range(population_size):
        population.append(Solutions(n))

    print("Poblacion sin evaluar: ")
    for sol in population:
        print(sol.value, sol.mape)

        """Evaluar poblacion inicial"""

    print("Primera poblacion: Generacion 0 ")
    for sol in population:
        evaluarmape(sol)
        print(sol.value, sol.mape)

    fit_prom_poblacion_inicial=0

    for sol in population:
#           evaluarmape(sol)
        fit_prom_poblacion_inicial+=sol.mape/len(population)
#           print(sol.value, sol.mape)
    print("Ultimos parametros y fit promedio")
    print(sol.value, fit_prom_poblacion_inicial)



    parametros, MAPE, valoresreales, valorespronosticados, entrenamiento_pronost, MAPE_DEL_PRONOSTICO, MAPE_entrenamiento = sintonizacion_SVR_AG(population)


    MAPE_entrenamientos.append(MAPE_entrenamiento)

    mejores_parametros.append(parametros)
    MAPE_del_test.append(MAPE)
    valores_reales.append(valoresreales)
    valores_pronosticados.append(valorespronosticados)
    conjunt_entrenamiento_pronost.append(entrenamiento_pronost)
```

```
    mape_del_pronostico.append(MAPE_DEL_PRONOSTICO)
#
    """Soluciones Varias de las generaciones"""


    Indice=pd.DataFrame(indice)
    Indice.columns=['INDICE']

    Country=pd.DataFrame(Country_Region)
    Country.columns=['Country/Region']

    Parametros_del_modelo=pd.DataFrame(mejores_parametros) #CONVIERTO LISTA NUMPY A DATAFRAME(PANDAS)
    Parametros_del_modelo.columns=['Kernel', 'C', 'Epsilon'] #ASIGNO NOMBRE DE A LA COLUMNA

    MAPES=pd.DataFrame(MAPE_del_test)
    MAPES.columns=['MAPE']

    MAPE_ENTRENAMIENTO=pd.DataFrame(MAPE_entrenamientos)
    MAPE_ENTRENAMIENTO.columns=['MAPE ENTRENAMIENTO']

    MAPE_PRONOSTICO=pd.DataFrame(mape_del_pronostico)
    MAPE_PRONOSTICO.columns=['MAPE DEL PRONOSTICO']

#    Valoresreales=pd.DataFrame(valores_reales)
    entrenamientopronosticado=pd.DataFrame(conjunt_entrenamiento_pronost)

    Pronostico=pd.DataFrame(valores_pronosticados)


    Soluciones = pd.concat([Indice, Country, Parametros_del_modelo, MAPES, MAPE_ENTRENAMIENTO, MAPE_PRONOSTICO, entrenamientopronosticado, Pronostico], axis=1, sort=
    Soluciones.to_csv("C:\\Users\\usuario\\Desktop\\Maestria\\Covid\\PRONOSTICOS_COVID_19\\PRONOSTICO_DE_TODOS_LOS_PAISES_SVR\\SVR_CONFIRMED\\CHINA\\CHINACORREGIDO\\
```

Figura A.7. Segunda parte del “Main”. Parte (7/7)

Anexo B

b. Anexo 2. Código SVR

Este código es un ejemplo de un modelo de aprendizaje automático de regresión utilizando la técnica de Support Vector Regression (SVR) en Python.

Primero, se importan las bibliotecas necesarias y se carga el dataset "mexy.csv" en un dataframe de pandas. Luego, se extraen las características (X) y la variable objetivo (y) del dataframe.

A continuación, se aplica la escalabilidad estándar a ambas características y la variable objetivo. Esto se hace para asegurarse de que los datos estén en una misma escala, lo que es

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('mexy.csv')
X = dataset.iloc[:, 0:1].values
y = dataset.iloc[:, 1].values

# Ajustes de escalas
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
sc_y = StandardScaler()
X = sc_X.fit_transform(X)
y = sc_y.fit_transform([y])


from sklearn.svm import SVR
regressor = SVR(kernel = 'rbf')
regressor.fit(X,y.reshape(-1,1))

SVR(C=0.1, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='auto', kernel='rbf', max_iter=-1,


# prediccion de un nuevo valor
x_trans = sc_X.transform([[10]])
y_pred = regressor.predict(x_trans)
y_pred = sc_y.inverse_transform(y_pred.reshape(-1, 1))
print("Predicción: ",y_pred)
```

importante para algoritmos de aprendizaje automático.

Figura A.2.1

Luego, se crea un objeto SVR y se entrena el modelo con los datos escalados de características y variable objetivo.

Por último, se hace una predicción para un nuevo valor (10) y se desescalan los resultados. La función print("Predicción: ",y_pred) imprimirá la predicción desescalada para el valor 10.

c. Anexo 3. Código MLP

Este código utiliza un modelo de regresión de perceptrón multicapa (MLP) para predecir una variable de respuesta y basada en una variable predictora x, usando la clase MLPRegressor del módulo sklearn.neural_network. El código utiliza la función train_test_split del módulo sklearn.model_selection para dividir los datos en conjuntos de entrenamiento y de prueba, luego entrena el modelo en el conjunto de entrenamiento utilizando el método fit. A continuación, el código entra en un bucle infinito, en el que entrena continuamente el modelo

```
import pandas as pd
import numpy as np
from sklearn.neural_network import MLPRegressor

datos= pd.read_csv("mexy.csv")
x=datos["dia"]
y=datos["y"]
X=x[:, np.newaxis]

while True:
    from sklearn.model_selection import train_test_split
    X_train, x_test, y_train, y_test =train_test_split(X,y)
    mlr=MLPRegressor(solver='lbfgs',alpha= 1e-5,hidden_layer_sizes=(3,3),random_state=1)
    mlr.fit(X_train,y_train)
    print(mlr.score(X_train,y_train))
    if mlr.score(X_train,y_train)>0.01:
      break
for i in range(10):

                    print("predicción:", mlr.predict(np.array([i]).reshape(-1,1)))
```

Figura A.3.1

hasta que la puntuación del modelo en el conjunto de entrenamiento es superior a 0,01, momento en el que sale del bucle y utiliza el modelo entrenado para hacer predicciones.

El código es un script de Python que utiliza la biblioteca scikit-learn para entrenar un regresor perceptrón multicapa (MLP) en un conjunto de datos y hacer predicciones sobre nuevos

datos. El regresor MLP se entrena utilizando el solucionador lbfgs y los hiperparámetros alpha y hidden_layer_sizes se establecen en los valores predeterminados.

El parámetro solver especifica el algoritmo utilizado para entrenar el modelo. En este caso, se utiliza el solver lbfgs, que significa "Limited-memory Broyden-Fletcher-Goldfarb-Shanno". Se trata de un algoritmo cuasi-Newton que puede utilizarse para entrenar redes perceptrón multicapa.

El parámetro alfa es un parámetro de regularización que ayuda a evitar el sobreajuste. Es un hiperparámetro, lo que significa que debe ser establecido por el usuario, en lugar de ser aprendido por el modelo durante el entrenamiento.

El parámetro hidden_layer_sizes especifica el tamaño de las capas ocultas de la red neuronal. En este caso, se establece en (3, 3), lo que significa que la red tiene dos capas ocultas, cada una con 3 neuronas. El número de neuronas en las capas ocultas y el número de capas ocultas son hiperparámetros que debe elegir el usuario.

El script comienza importando los módulos y clases necesarios de las librerías pandas, numpy y scikit-learn. La biblioteca pandas se utiliza para cargar los datos desde un archivo CSV, la biblioteca numpy se utiliza para manipular los datos, y la biblioteca scikit-learn se utiliza para entrenar el regresor MLP y hacer predicciones.

A continuación, los datos se cargan desde un archivo CSV llamado "mexy.csv" utilizando la función pandas read_csv y se almacenan en un pandas DataFrame. Las variables predictoras y de respuesta se extraen del DataFrame y se convierten en matrices numpy. Se supone que la variable de predicción es la columna "dia" y que la variable de respuesta es la columna "y".

Despues el código entra en un bucle infinito y continúa entrenando el regresor MLP y evaluando su rendimiento hasta que alcanza una puntuación superior a 0,01. Los conjuntos de entrenamiento y prueba se crean utilizando la función train_test_split y el regresor MLP se entrena utilizando el método fit. A continuación, se evalúa el rendimiento del modelo en el conjunto de entrenamiento mediante el método de puntuación. Si la puntuación es superior a 0,01, se rompe el bucle y el código continúa.

Finalmente, el código utiliza el regresor MLP entrenado para hacer predicciones sobre nuevos datos. Para ello, repasa los números del 0 al 9, realiza una predicción para cada número utilizando el método predict e imprime las predicciones en la consola. Esto permite al usuario ver el rendimiento del modelo en datos no vistos.

d. Anexo 4. Código MAPE

El código realizado en Python para calcular el MAPE (Mean Absolute Percentage Error) entre dos listas de datos.
Aquí está una explicación detallada de cada parte del código:

1. Definición de la función **mape**:

```python
def mape(real, prediction):
```

Figura A.4.1

Esta función toma dos listas como entrada: **real** y **prediction**. Estas listas representan los valores reales y las predicciones, respectivamente.

2. Inicialización de las variables **n** y **mape**:

```python
    n = len(real)
    mape = 0
```

Figura A.4.2

La variable **n** es igual al número de elementos en la lista **real**. La variable **mape** se inicializa en cero y se usará para almacenar la suma de los porcentajes absolutos de error.

3. Cálculo del MAPE:

```python
    for i in range(n):
        mape += abs(real[i] - prediction[i]) / real[i]
    mape = mape / n
```

Figura A.4.3

Este ciclo **for** itera sobre los elementos en la lista **real**. Para cada elemento, se calcula el porcentaje absoluto de error entre el valor real y la predicción correspondiente. Luego, se suma este valor a la variable **mape**.

Después de completar el ciclo **for**, se divide la variable **mape** por **n** para obtener el promedio de los porcentajes absolutos de error.

4. Devolución del resultado:

```python
    return mape
```

Figura A.4.4

Finalmente, la función devuelve el resultado del cálculo del MAPE.

5. Definición de las listas **real** y **prediction**:

```python
real = [1, 2, 3, 4, 5]
prediction = [1.1, 2.1, 2.9, 4.2, 5.1]
```

Figura A.4.5

Estas son las listas de datos que se usarán como entrada para la función **mape**.

6. Cálculo del MAPE:

```python
mape_result = mape(real, prediction)
```

Figura A.4.6

Aquí, se llama a la función **mape** y se pasa las listas **real** y **prediction** como argumentos. El resultado del cálculo se almacena en la variable **mape_result**.

7. Impresión del resultado:

```python
print("MAPE: {:.2f}%".format(mape_result * 100))
```

Figura A.4.7

Finalmente, se imprime el resultado del cálculo del MAPE en formato de porcentaje con dos decimales. La cadena de formato **"MAPE: {:.2f}%"** significa que se mostrará el texto "MAPE:" seguido del resultado del cálculo del MAPE multiplicado por 100 y formateado con dos decimales. La multiplicación por 100 se hace para obtener el porcentaje.

e. Anexo 5. Código SMAPE

El código SMAPE (Symmetric Mean Absolute Percentage Error) es una métrica que se utiliza para medir la precisión de las predicciones de un modelo de aprendizaje automático.

Es una forma de medir el error absoluto en términos de porcentaje, y es simétrico en el sentido de que tanto los valores subestimados como los sobreestimados se penalizan de la misma manera.

```
# Calcular el SMAPE
def smape(real, prediccion):
  n = len(real)
  smape = 0
  for i in range(n):
    smape += abs(real[i] - prediccion[i]) / (abs(real[i]) + abs(prediccion[i]))
  smape = smape / n
  return smape

# Valores reales
real = [11238,10801,4022,2522,11803,12295,12097,11504,9734,3620]

# Valores predicciones
prediccion = [10919.3,10509.55,4077.35,2787.4,11612.45,12091.15,11909.45,11356.9,9528.35,3648.1]
# Calcular el SMAPE
smape_result = smape(real, prediccion)

# Mostrar resultado
print("El SMAPE es del {:.2f}%".format(smape_result*100))
```

```
El SMAPE es del 1.30%
```

Figura A.6.1

En este código, **y_true** representa los valores reales y **y_pred** representa los valores predichos. La función calcula el error absoluto como la diferencia entre los valores reales y

predichos, y lo divide por la suma de los valores absolutos de los valores reales y predichos, dividido por 2.

Se tiene en cuenta que la división por cero se controla estableciendo la diferencia en 0 en caso de que el denominador sea 0. Finalmente, se calcula el promedio de los errores para obtener una métrica única.