# Práctica 1
# Aplicación de RNA

Inteligencia Artificial en las Organizaciones

Grupo 83-1

*Miguel Gutierrez Pérez*
100383537@alumnos.uc3m.es

*Mario Lozano Cortés*
100383511@alumnos.uc3m.es

*Alba Reinders Sánchez*
100383444@alumnos.uc3m.es

*Alejandro Valverde Mahou*
100383383@alumnos.uc3m.es

12 de octubre de 2020

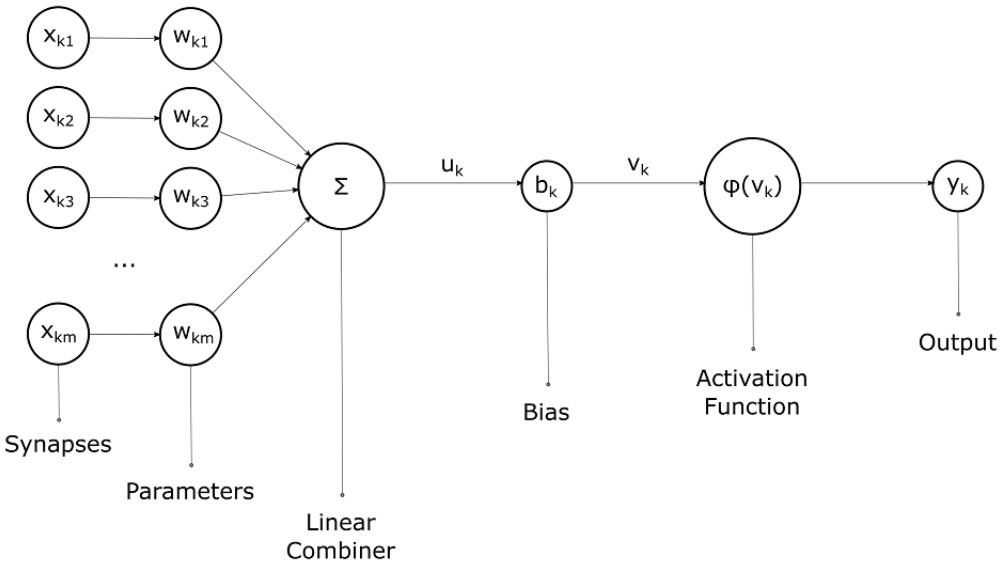
# Índice

# 1. Introducción

El objetivo de la asignatura de Inteligencia Artificial en las Organizaciones es el poner en relieve el encaje de las diversas técnicas de IA en contextos reales. Con esta motivación en mente, parece evidente que es imprescindible lograr obtener soluciones a problemas apremiantes con el conjunto de técnicas disponibles. Por su parte, la **epidemia** del **SARS-CoV-2** supone un **reto** para la humanidad y constituye una oportunidad para demostrar el potencial de las nuevas herramientas de IA de las que disponemos para hacer frente a este nuevo desafio.

Uno de los **modelos computacionales** cuya aplicación resulta atractiva es el de las **Redes de Neuronas Artificiales**. La capacidad de aprender de grandes conjuntos de datos hacen de esta una opción perfecta para comprender y predecir los contagios causados por el virus.

Para comprender cómo cumplir con este objetivo debemos tener en cuenta cómo es posible que una red de neuronas aprenda. La respuesta se haya en su estructura. A nivel básico la estructura de una RNA supone un conjunto de entradas multiplicadas por unos pesos a modo de impulso nervioso al que se le puede aplicar determinadas funciones de activación. Dicha estructura de una neurona puede ser vista a continuación.




Estructura de una neurona

Por lo tanto, hemos trasladado el problema a la **selección de los pesos** precisos para realizar predicciones lo más exactas posibles. El modelo a emplear se tratará del **Perceptrón Multicapa**. Consecuentemente el trabajo realizado en esta práctica consiste en *tratar los datos para la aplicación del modelo elegido*, *realizar los experimentos oportunos para dar con una configuración de la red apropiada* y *analizar los resultados* obtenidos haciendo una reflexión crítica del proceso seguido.

La práctica se divide en dos partes: la primera trata de resolver un problema de **regresión** utilizando el **Perceptrón Multicapa** y la segunda trata de realizar predecicciones utilizando **series temporales** con la herramienta *Weka*.

# 2. Parte 1: Regresión

## 2.1. Planteamiento y desarrollo del problema a tratar

En esta primera parte se pretende predecir la transmisión del virus *COVID-19* partiendo de un conjunto de datos que se explicarán en la siguiente sección. Se trata pues de un problema de regresión que se va a llevar a cabo mediante una RNA, a la que se van a modificar sus parámetros hasta conseguir aquellos con los que se genere un modelo con menor error a la hora de predecir.

El objetivo es predecir con la mayor exactitud posible los valores en 1, 2 y 3 días a futuro en dos países: **España** y **Brasil**. Para ello, se entrena una red con todos los datos, y se aplica la predicción sobre estos dos.

Para realizar la predicción en 2 y 3 días a futuro, se toman como entrada, además de los valores reales, los valores que la red ha predicho anteriormente.

## 2.2. Tratamiento de datos

Los datos que se han utilizado pertenecen al *Novel Coronavirus (COVID-19) Cases Data* de la página *The Humanitarian Data Exchange*[Referencia 4], se trata de una recopilación de **datos epidemiológicos del COVID-19** desde el día 22 de enero de 2020.

En concreto, los datos de interés son los que se encuentran en el fichero de casos diarios confirmados, este está compuesto de **266** provincias/estados de distintos países/regiones, de los cuales se recopila el **número de infectados totales** desde el 22 de enero hasta la fecha.

Los primeros atributos son: nombre de la provincia/estado, país/región, longitud y latitud (del país). Seguidos del número de contagiados acumulados por día.

El tratamiento de los datos que se ha llevado a cabo es el siguiente:

- En primer lugar, se ha eliminado la comilla simple del nombre de un país del fichero.
- Se han renombrado los atributos correspondientes a las fechas de forma que el día actual se convierte en *Día 0* y el resto en *Día -1*, *Día -2*,...

## 2.3. Acercamiento con WEKA

Para poder trabajar con la herramienta *WEKA* se transforma el fichero de formato *.csv* a *.arff*. Como la configuración por defecto que ofrece *WEKA* para el algoritmo del perceptrón multicapa no es adecuada para la resolución del problema, es necesario alterar sus parámetros. Los valores que admiten modificación son:

- Número de ciclos
- Tasa de aprendizje
- Número de capas ocultas

Para poder realizar varios experimentos con distintos modelos y configuraciones diferentes, se usa la herramienta *Experimenter* de *WEKA*.

Debido a la cantidad de tiempo requerido para realizar el entrenamiento y a distintos problemas como: errores al leer los datos, valores de error extremadamente altos en algunos experimentos, poca precisión de los modelos y falta de personalización de la red, se decide descartar este acercamiento por uno que permite resolver estos problemas de manera sencilla.

## 2.4. Acercamiento con Python-Tensorflow

La principal ventaja de la biblioteca *Tensorflow* es la personalización de las redes. Esto permite la modificación de valores como la función de coste, la función de optimización, o el número de neuronas por cada capa de la red. Además, junto a la biblioteca de cálculo numérico *numpy* permite que el entrenamiento de estas redes sea mucho más ágil.

Una de las facilidades que ofrece es la capacidad de crear redes neuronales de forma rápida, y poder entrenarla y realizar predicciones con apenas unas líneas de código.

Gracias a esta tecnología ha sido posible realizar distintos experimentos hasta que se ha encontrado la configuración de la red que es capaz de resolver el problema de la mejor manera.

Para realizar la lectura de los datos se ha hecho uso de la biblioteca *pandas*, concretamente, la función de *read_csv*. Después, se ha transformado en una lista de *numpy*, lo que ha facilitado su manejo dividiendo los datos e forma rápida y sencilla.

Se han creado 3 archivos de *Python* con el propósito de resolver el problema:

1. **Perceptrón con 'K Fold'**[Anexo 1]

   Este primer archivo usa una función de la bibioteca *sklearn* que implementa el algoritmo '*K Fold*'. Se ha decidido realizar la división de los datos en 10 partes.

   El archivo genera y guarda los modelos para, posteriormente, poder usarlos en la predicción.

2. **Perceptrón con 'split percentage'**[Anexo 2]

   El segundo archivo genera modelos entrenados usando la técnica de división de datos en conjunto de entrenamiento y de test. Estos modelos también se guardan para poder realizar predicciones con los mismos.

3. **Predicción usando los modelos generados**[Anexo 3]

   El archivo realiza la predicción, usando los modelos cargados, con los dos países seleccionados (España y Brasil). Las predicciones se realizan, como se ha explicado antes, sobre el primer día usando los datos reales. Despúes, los datos reales, junto a este primer día predicho, se usan para realizar la predicción del segundo día. El mismo proceso se hace para realizar la tercera predicción.

## 2.5. Configuración de experimentos con MLP

Se han planteado 4 modelos en función de los días seleccionados para entrenar a la red: modelo con 7 días, modelo con 15 días, modelo con 30 días y modelo con 60 días.

Se ha elegido esta división, que corresponden con 1 semana, 2 semanas, 1 mes y 2 meses respectivamente, ya que se quiere comprobar cuánto tiempo es necesario conocer de antemano para preever la evolución de infectados por *COVID-19* en una región.

Se han realizado distintos experimentos con cada modelo, dividiendo entre el algoritmo de '*K Fold*' y '*percentage split*' para encontrar aquel que genere los mejores resultados.

Los hiperparámetros comunes a todos los modelos son:

- **Número de capas ocultas**: son aquellas capas de una red neuronal que no son ni de entrada ni de salida. No se ha utilizado **ninguna** capa oculta, porque los resultados

experimentales proporcionan mayor precisión si no se usan. Además, tanto la herramienta de *WEKA* como los modelos generados con *Tensorflow* coincidían en esta arquitectura.

- **Tasa de aprendizaje**: es el factor por el que se multiplica el gradiente durante el entrenamiento de la red, generando el paso de gradiente, utilizado para actualizar los pesos. Se ha usado el valor que ofrece la biblioteca *Tensorflow* por defecto en el optimizador, **0.001**.
- **Número de ciclos**: es el número de veces que la red realiza el bucle de entrenamiento sobre todos los datos. Se han utilizado **500** ciclos porque a partir de este valor las mejoras de los modelos no eran significativas.
- **Optimizador**: es la función encargada de encontrar el mínimo local de la función de coste y actualizar los pesos de la red en consecuencia. En este caso el elegido es **Adam** ya que es una versión del descenso del gradiente que genera mejores resultados.
- **Función de coste**: es la función que indica el error de la red. En este caso se ha usado el **error medio absoluto**, dado que se trata de un problema de regresión. Otra posible opción habría sido el error cuadrático medio.
- **Función de activación**: es la función de cada neurona que toma como entrada la suma de todas las salidas de la capa anterior y genera una salida no lineal para la siguiente capa. En los modelos se usa la función **ReLU** en las capas ocultas, y la función **linear** en las capas de salida, ya que se trata de un problema de regresión.

Los parámetros que se modifican entre los modelos son el algoritmo usado para generar el modelo, el número de entradas de la red, y el número de neuronas en la capa de entrada.

| Algoritmo | Entradas red | Neuronas capa entrada | Error mejor experimento |
|---|---|---|---|
| K Fold | 7 | 64 | 4076.555 |
| Split | 7 | 64 | 1378.325 |
| K Fold | 15 | 64 | 2138.145 |
| Split | 15 | 64 | 2141.331 |
| K Fold | 30 | 128 | 2942.597 |
| Split | 30 | 128 | 10260.810 |
| K Fold | 60 | 128 | 3031.660 |
| Split | 60 | 128 | 8037.908 |

A pesar de que unos modelos puedan mostrar menor error, no tiene por qué ser ese modelo el mejor. Hasta que no se pruebe con casos reales no se podrá asegurar que uno es mejor que otro.

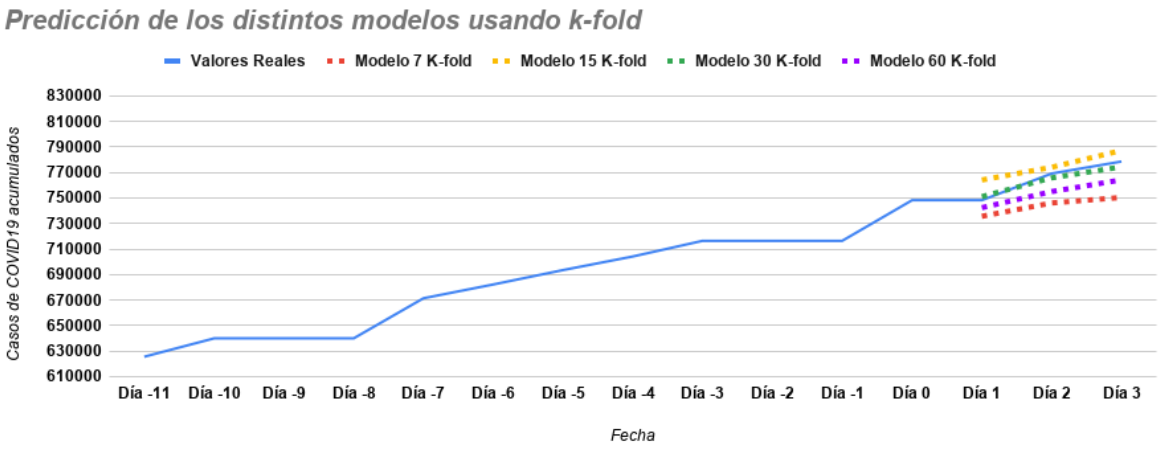
Aún así, según refleja la tabla, el error cometido con el algoritmo *K Fold* es menor en todos los casos excepto en el que sólo hay 7 entradas en la red.

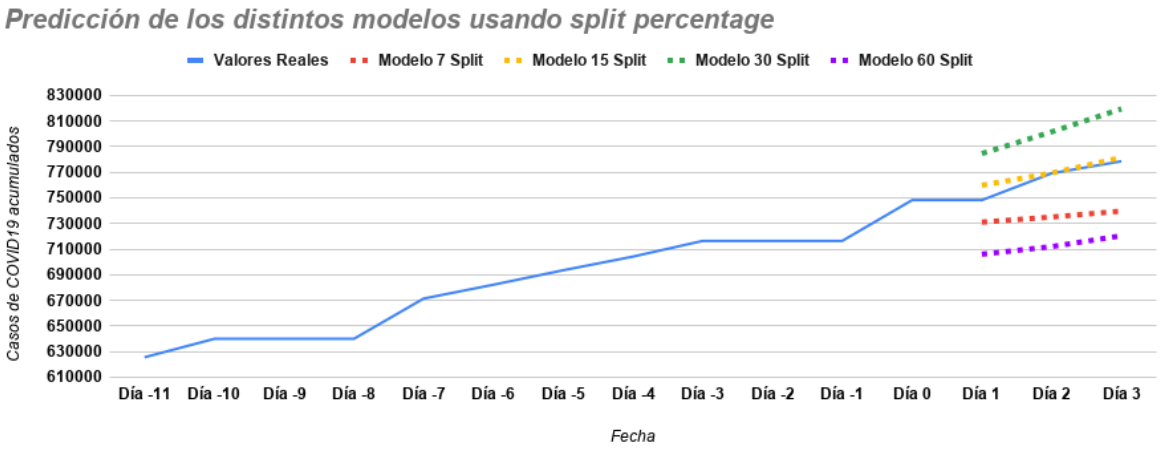
## 2.6. Análisis de resultados

En primer lugar se van a analizar los resultado obtenidos acerca de la predicción de los contagiados en **España**, usando gráficas que permitan comparar los distintos modelos generados, en función a sus resultados y sus errores, y a continuación, se realizará el mismo proceso con los datos de **Brasil**.

### 2.6.1. Resultados de España

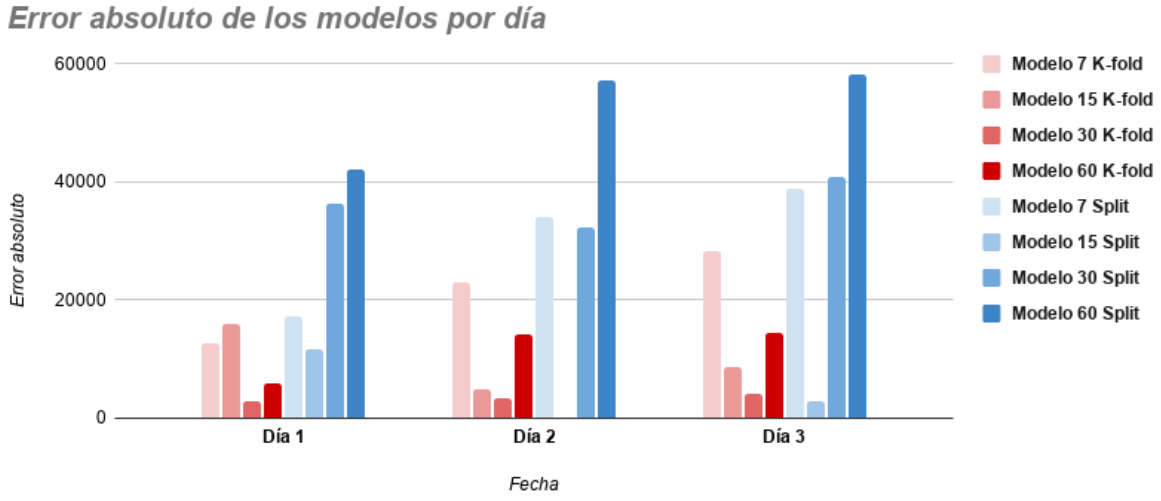
Las siguientes dos gráficas muestran las predicciones de los modelos respecto a los datos de España. Se ha decidido mostrar hasta 11 días atrás para poder observar la evolución de los datos con perspectiva.




Se puede observar gráficamente que los mejores modelos para este problema son el **Modelo 30 K Fold** y el **Modelo 15 Split**. A pesar de esto, los modelos de *K fold* se acercan mucho más a los valores reales, y se puede atribuir la precisión del **Modelo 15 Split** al azar.

La siguiente gráfica muestra la evolución del error absoluto en función de los días predichos.



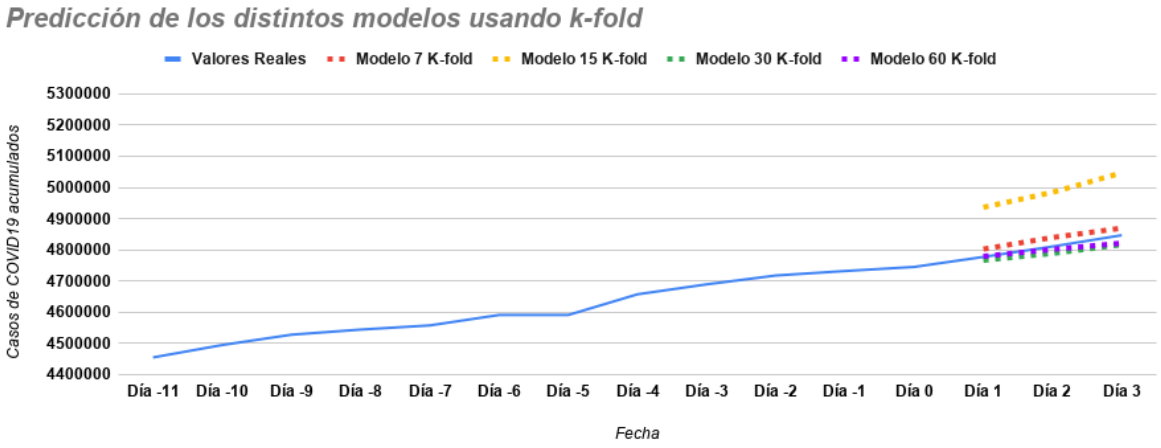

Esta gráfica demuestra que la previsión acertada del **Modelo 15 Split** es fruto del azar, porque en el primer día muestra un gran error, que corrige en el segundo día, y se vuelve a incrementar

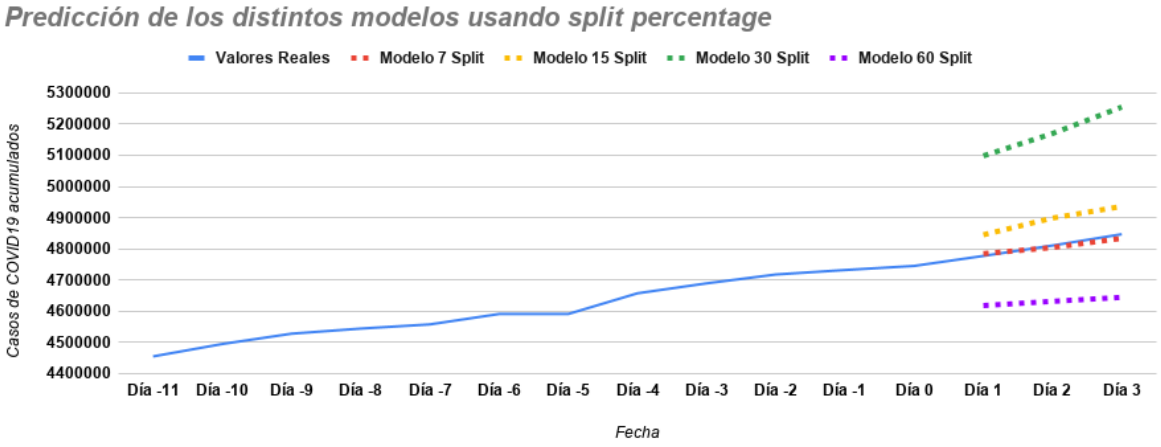
en el tercero. Si realmente fuera capaz de realizar previsiones acertadas, el error debería ser incrementeal a lo largo del tiempo.

Viendo los resultados de las dos gráficas, junto a la gráfica de error, se considera por tanto que los modelos que usan **K Fold** son mejores para este problema, concretamente, el mejor para el caso de España es el **K Fold con datos de 30 días**.

### 2.6.2. Resultados de Brasil

En el caso de Brasil se muestran, al igual que en el caso de España, hasta 11 días atrás ya que se observa mejor la evolución de los datos.

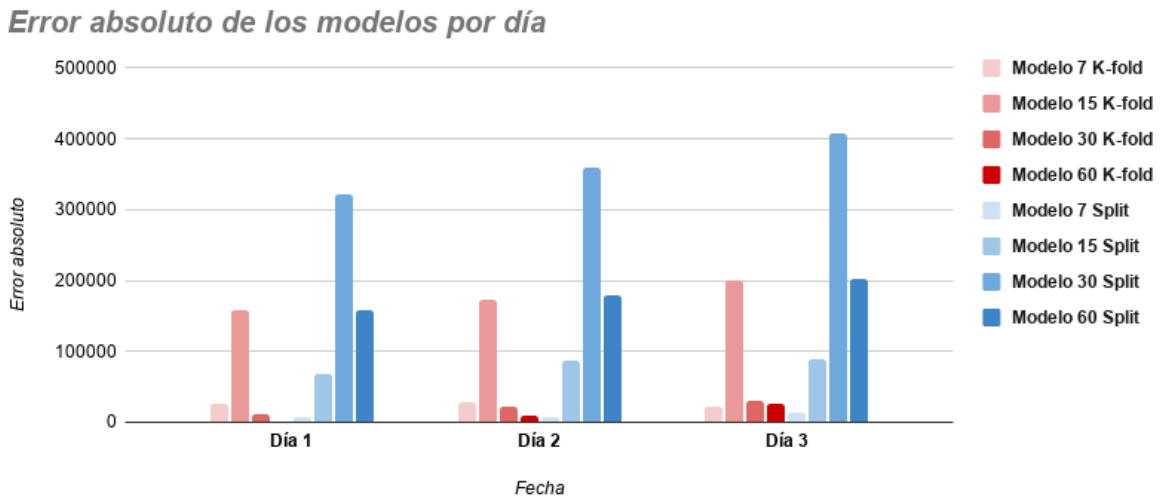



Gráficamente, se observa que los mejores modelos para este problema son el **Modelo 60 K Fold**, el **Modelo 30 K Fold** y el **Modelo 7 Split**. Al igual que en el caso de España, los modelos de *K fold* se acercan mucho más a los valores reales, y se puede atribuir la precisión del **Modelo 7 Split** al azar.

Esto se puede demostrar a su vez porque el **Modelo 7 Split** en el caso de España generaba resultados malos y el **Modelo 15 Split** que daba buenos resultados, en este caso no los da.

La siguiente gráfica muestra la evolución del error absoluto en función de los días predichos.




A pesar de que en esta gráfica el error de los modelos que usan *split percentage* es proporcional según se aumenta el día a predecir, debido a los motivos discutidos en el apartado anterior (Resultados de España) y los motivos explicados en las gráficas anteriores, se ha decidido que para el caso de Brasil los mejores modelos sean el **Modelo 60 K Fold** y el **Modelo 30 K Fold**.

Dado que el **Modelo 30 K Fold** es el que mejores resultados produce como regla general, se considera que el mejor modelo para resolver el problema de predecir los contagiados de un país cualquiera es **K Fold con datos de 30 días**.

*Para ver información más detallada de las gráficas y sus valores, consultar [Anexo 4]*

# 3. Parte 2: Series temporales

## 3.1. Descripción de las series temporales utilizadas

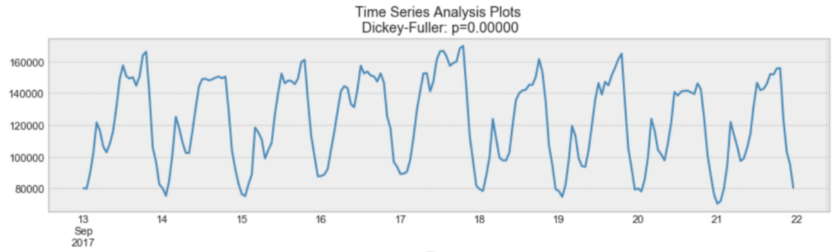
Podemos definir las series temporales como un una **sucesión de datos o muestras medidos en intervalos de tiempo regulares** de los cuales resulta especialmente interesante el factor de predicción de los valores en instantes de tiempo futuros. Dichas predicciones son posibles al detectar patrones o tendencias en los datos a lo largo del tiempo.

La pregunta que nos debemos hacer es si se corresponden los datos de contagios acumulados y muertes por territorios del *SARS-CoV-2* con una serie temporal de la que podamos realizar predicciones fiables. Para contestar a esta pregunta vamos a fijarnos en las variables de las que disponemos y la relación entre ambas. Como variable independiente nos encontramos con tiempo, mientras que el número de contagios (o muertes, según el caso) se identifica con la variable dependiente puesto que esta cifra depende en gran medida del la cifra medida en el instante de tiempo anterior. Por lo tanto **se trata de una serie temporal**.

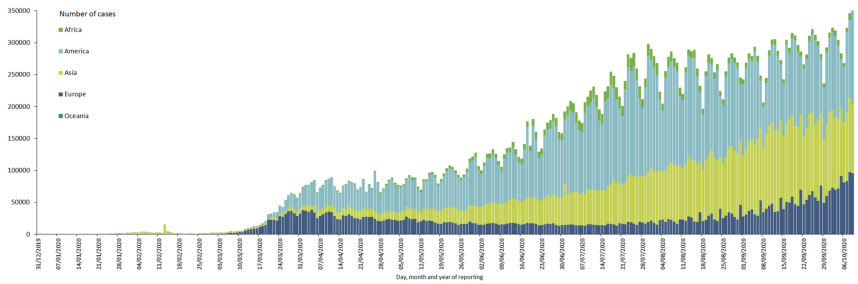
Sin embargo, existen más características en las que nos podemos fijar. [Referencia 2]

- **Estacionalidad**: Se refiere a fluctuaciones periódicas. Un análisis en profundidad de los datos de la epidemia debería considerar este factor, no obstante, dado que la pandemia no lleva entre nosotros un periodo de tiempo lo suficientemente grande como para detectar un comportamiento estacional a lo largo de las distintas épocas del año no nos fijaremos en este parámetro.
- **Estacionaridad**: No debe ser confundido con la estacional dado que hace referencia al mantenimiento de los valores estáticos tales como la media y varianza. El proceso de crecimiento descontrolado de una pandemia no puede ser considerado estacionario debido a la tendencia alcista y volátil de los datos.

A continuación se muestra un ejemplo de una serie temporal estacional y estacionaria para comprobar cómo difiere de la serie temporal de los datos del *COVID* mundiales.




Serie temporal estacional y estacionaria




Casos mundiales de *COVID*. [Referencia 3]

# 4. Proceso de entrenamiento

La red de neuronas generada seguirá el algoritmo de *Multilayer Perceptron* para la predicción de series temporales en el apartado *timeseriesForecasting* de *Weka*. Por lo tanto, el siguiente paso consiste en la selección de los parámetros de la red que permitan encontrar la mejor predicción posible.

## 4.1. Arquitecturas probadas

En el caso de *Multilayer Perceptron* existen multitud de parámetros de los cuales debemos considerar su variación y combinación para obtener una buena arquitectura de red. Algunos de ellos (*hidden layers, learning rate y training time*) han sido descritos en la la sección Parte 1 de esta memoria y por tanto se obvia la explicación. Sin embargo, la aproximación con series temporales introduce un nuevo parámetro a explicar. La definición del mismo se da a continuación.

- **Lag Lenght**: Las variables *lag* son el mecanismo por el cual se determina la **relación entre las variables pasadas y las tratadas actualmente**. Una forma de pensar en este parámetro consiste en equipararlo a la longitud de la ventana de tiempo en la que se tiene en cuenta el valor de las variables pasadas. Por consiguiente, **un valor apropiado de este parámetro en el caso considerado es 7 días** dado que los estados suelen tener problemas en las notificaciones del fin de semana, siendo apropiado estudiar el efecto de la pandemia en porciones de una semana. Por lo tanto, se usarán 7 y 14 días para las pruebas.

Se concluye que es necesario realizar la siguiente aproximación a la arquitectura de la red mostrada en la tabla contigua, combinando adecuadamente valores lógicos de los parámetros propuestos.

| ID experimento | Lag Lenght | Hidden Layers | Learning Rate | Training Time |
|---|---|---|---|---|
| 1 | 7 | a | 0,1 | 500 |
| 2 | 7 | a | 0,1 | 5000 |
| 3 | 7 | a | 0,3 | 500 |
| 4 | 7 | a | 0,3 | 5000 |
| 5 | 7 | t | 0,3 | 500 |
| 6 | 7 | t | 0,3 | 5000 |
| 7 | 7 | t | 0,1 | 500 |
| 8 | 7 | t | 0,1 | 5000 |
| 9 | 14 | a | 0,1 | 500 |
| 10 | 14 | a | 0,1 | 5000 |
| 11 | 14 | a | 0,3 | 500 |
| 12 | 14 | a | 0,3 | 5000 |
| 13 | 14 | t | 0,3 | 500 |
| 14 | 14 | t | 0,3 | 5000 |
| 15 | 14 | t | 0,1 | 500 |
| 16 | 14 | t | 0,1 | 5000 |

Tabla 1: Configuración de experimentos

Finalmente, es interesante destacar que cada uno de los experimentos debe ser dividido a su vez en *4 sub-experimentos* debido a que la herramienta de *forecasting* permite tomas los datos

de contagios y muertes de manera aislada o de manera cruzada. Es decir, para cada una de las configuraciones de los experimentos debemos realizar uno con los datos de confirmados acumulados aislados, uno con los datos de los muertos aislados y otros dos con las series cruzadas. Los resultados de los experimentos se tratan en el siguiente apartado.

## 4.2. Arquitectura seleccionada

# 5. Contexto de la práctica

La *COVID-19* está asolando al mundo entero. Desde su surgimiento a finales de 2019 se han intentado encontrar diferentes técnicas que sean capaces de frenar su avance. Una de las técnicas con más futuro, y que mejores resultados ha logrado en otros campos similares, es la inteligencia artificial, y, más concretamente, el **aprendizaje automático**. Tal como comenta *Markus Schmitt* en su artículo del 7 de Abril(nótese su fecha de publicación)[Referencia 5], se pueden usar las diferentes técnicas del aprendizaje automático para resolver los distintos problemas que plantea esta pandemia global, como puede ser **identificar la población de riesgo**, **predecir la propagación y origen de la enfermedad** o incluso **predecir pandemias futuras**, entre otras aplicaciones.

Uno de los principales problemas que se comentan, es la falta de datos necesarios para crear modelos realmente eficaces, ya que su comentario se remonta a los inicios de la pandemia. Pero ahora, 6 meses después, esa cantidad de datos es muchísimo mayor, lo que permite la generación de modelos de aprendizaje automático mucho más robustos y precisos, tal y como se ha mostrado en esta práctica.

Otra de las numerosas maneras en las que la tecnología puede ayudar es la simulación del pliegue de proteínas, esencial para comprender el virus y por tanto la búsqueda de la cura y la vacuna. Uno de los proyectos más interesantes que han surgido al respecto es *folding@home* [Referencia 6], un proyecto de computación distribuida en que están implicados universidades como *Stanford* o la *Universidad de Washington* y empresas como *Google* o *NVIDIA* que buscan simular las dinámicas de las proteínas, incluyendo su pliegue, lo cual explica su nombre.

De manera breve, lo que ellos hacen es usar el poder de cálculo de todos los ordenadores que lo tengan instalado cuando no estén en uso para encontrar la cura de diversas enfermedades como el Coronavirus, pero también el Alzheimer o el Párkinson. Lo único que el usuario debe hacer es seleccionar la cura de la enfermedad a la que desea contribuir y el programa de manera automática aprovechará el tiempo IDLE del ordenador para realizar los cálculos que los científicos necesitan. A continuación se ofrece un link con una pequeña guía de instalación en formato vídeo del software necesario, realizado por uno de los autores de este documento.[Referencia 7]

# 6. Conclusiones

Considerando todos los resultados obtenidos a lo largo de este documento, se puede afirmar que las Redes de Neuronas constituyen una buena aproximación a problemas del mundo real como en el caso de la pandemia actual. Los modelos que se generan usando estas técnicas permiten adelantarnos a los sucesos y realizar predicciones con mayor o menor exactitud. Esto es una herramienta muy potente, siempre que se posea de una cantidad de datos lo suficientemente grande.

Sin embargo, se requiere de mucho tiempo de computo para entrenar modelos eficaces y, además, actuan como *cajas negras* y por tanto no tienen la capacidad de explicar sus soluciones. Esto resulta en procesos de prueba y error para encontrar mejores resultados.

En relación al enfoque de esta práctica, los resultados obtenidos pueden representar la tendencia a seguir de la evolución de la pandemia aunque hay que tener en cuenta que los modelos no son plenamente fiables, entre otras cosas, debido a que los datos tampoco lo son, ya que este tipo de algoritmos se rigen por el principio **GIGO** (*Garbage In Garbage Out*).

Algunas posibles mejoras para este tipo de modelos sería realizando un estudio previo para los datos, para evitar errores, y conseguir información tomada de forma similar de todas las regiones. De esta forma se evitarían gran cantidad de problemas de los datos. Otra posibilidad para las redes neuronales sería utilizar técnicas de aumento de datos, de forma que se puedan usar más ejemplos distintos durante el entrenamiento.

# 7. Referencias


1. Introduction to Neurons in Neural Networks. Medium. Consultado en Octubre 2020. Url: https://medium.com/artificial-neural-networks
2. The complete guide to time series. Consultado en Octobre 2020. Url: https://towardsdatascience.com/the-complete-guide-to-time-series
3. COVID Data European Union. Consultado en Octubre 2020. Url: https://www.ecdc.europa.eu
4. The Humanitarian Data Exchange. Consultado en Octubre 2020. Url: https://data.humdata.org/dataset/novel-coronavirus
5. How to fight COVID-19 with machine learning. Consultado en Octubre 2020. Url: https://towardsdatascience.com/fight-covid-19-with-machine-learning
6. Folding@home. Consultado en Octubre 2020. Url: https://foldingathome.org
7. Descubre cómo puedes ayudar a encontrar una CURA para el CORONAVIRUS desde tu salón. Consultado en Octubre 2020. Url: https://www.youtube.com/watch?v=xqvAHnac79U

# 8. Anexos

1. Perceptron Multicapa usando '*K Fold*'
   ***perceptron_kfold.py***
2. Perceptron Multicapa usando '*split percentage*'
   ***perceptron_split.py***
3. Programa para realizar la predicción de los modelos
   ***predict.py***
4. Tabla de resultados de los experimentos
   ***valores_reales_vs_predicciones_&_errores_absolutos.xlsx***