Universidad Catolica de Ávila

Facultad de Ciencias y Artes

TRABAJO FIN DE GRADO

ESTUDIO DE MODELOS DE REDES NEURONALES EN EL MERCADO DE DIVISAS

CURSO DE COMPLEMENTOS DE FORMACION DE ADAPTACION A GRADO EN INGENIERIA DE SISTEMAS DE INFORMACIÓN

Alumno: **DAVID ROSA PEINADO**

Director: **MARTA GÓMEZ PÉREZ**

Convocatoria: SEPTIEMBRE **Año: 2019**

RESUMEN

El presente Trabajo Fin de Grado versa sobre la aplicación de algoritmos de inteligencia artificial a los mercados financieros, concretamente al mercado de divisas. El objetivo de este estudio es comprobar si es posible la predicción de forma rentable del movimiento del precio en el mercado de monedas, concretamente el del par EUR/USD. Para lograr este objetivo se han realizado una serie de experimentos, dónde se han entrenado distintas arquitecturas neuronales clásicas. En estos experimentos se han generado modelos separados para las operaciones de compra (BUY) y venta (SELL). Para evaluar la rentabilidad del sistema, se ha considerado como métrica el porcentaje aciertos positivos. Esta métrica nos indicará que el sistema es rentable si se alcanza un porcentaje superior al 50%. Asimismo, estos modelos se han combinado para comprobar si el resultado conjunto fuera rentable. Todos los modelos se han entrenado en tres periodos de tiempo distintos: bloque 1 (2009-2015), bloque 2 (2009-2017) y bloque 3 (2009-2019).

Los resultados obtenidos muestran un éxito moderado en la predicción de los movimientos. SI bien se han conseguido porcentajes superiores al 50%, en la mayoría de los casos, hay un periodo de tiempo, el llamado bloque 3, dónde no se consiguen resultados positivos en el conjunto de desarrollo /test. Además, no se han tenido en cuenta el coste de transacciones financieras, el cual podría obligar a tener un porcentaje sensiblemente más alto del inicialmente esperado para hacer el sistema rentable.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO I: INTRODUCCIÓN

1.1. CONTEXTUALIZACIÓN

El presente trabajo trata de usar técnicas de aprendizaje supervisado, concretamente, aprendizaje de redes neuronales, para predecir tendencias en el mercado de divisas.

Efectivamente, la industria financiera ha usado a lo largo del tiempo herramientas cuantitativas y de análisis técnico para predecir tendencias en los precios de los activos que cotizan en los distintos mercados financieros.

El desarrollo y aplicación de técnicas de aprendizaje profundo es de enorme interés en la industria financiera como se demuestra con el incremento exponencial de tesis e investigaciones publicadas en los últimos años.

1.2. MOTIVACIÓN

La motivación para la realización de este trabajo reside en la necesidad de demostrar de una que las técnicas de aprendizaje de redes neuronales son capaces de generar predicciones confiables sobre el movimiento del precio de un activo de divisas. Y de esta forma proveer a los miembros del mercado financiero un modelo robusto y reproducible capaz de generar beneficios.

La tarea de superar al mercado y ofrecer unos retornos superiores, no es una tarea fácil y desde el punto de vista más académico se ha defendido la llamada Hipótesis del Mercado Eficiente, EMH (acrónimo en inglés) en adelante. Según esta hipótesis, los precios de los activos ya reflejan toda la información disponible en ese momento y, por lo tanto, resulta imposible superar al mercado de forma constante. En cualquier caso, es un entorno desafiante para académicos y no académicos. El aprendizaje profundo unido al incremento de la capacidad de cómputo podría proporcionar la prueba de que la EMH no es cierta, y los mercados pueden ser predecibles hasta cierto punto, usando características conocidas previamente de datos pasados.

1.3. OBJETIVOS

El propósito general del proyecto será lograr un modelo de aprendizaje con todas sus peculiaridades que consiga un sistema de trading con un porcentaje de aciertos superior al 50%.

Este modelo deberá ser capaz de generar predicciones suficientemente buenas acerca del movimiento del tipo de cambio de un par de divisas, concretamente del euro-dólar.

Para conseguir este objetivo general se deberán alcanzar los siguientes subobjetivos:

- El modelo desarrollado se ajustará a una arquitectura de red neural con todos sus nodos conectados

- El proceso de generado del modelo deberá ser correctamente descrito para favorecer su reproducibilidad.
- El activo subyacente sobre el que se aplicará el modelo será el tipo de cambio euro-dólar, en adelante EURUSD.
- Para que el objetivo general sea correctamente alcanzado, el porcentaje de aciertos debe ser superior al 50% en todos los periodos de entrenamiento-test.
- Se deberán usar datos de un proveedor reconocido y confiable.

Como objetivo secundario quedaría el poder probar el algoritmo de aprendizaje sobre una tarjeta gráfica con múltiples núcleos para comprobar la notable mejora en tiempo de cómputo que hoy en día ofrece el modelo GPU de computación.

1.4. ALCANCE DEL PROYECTO

El proyecto persigue demostrar que el aprendizaje de redes neuronales mejora la predicción de tendencias en los mercados de divisas. Para ello se ofrece una descripción clara y precisa de los pasos que han llevado al desarrollo del modelo:

- Elección de algoritmo de aprendizaje
- Elección de características de los datos de entrenamiento
- Elección de los valores de los distintos parámetros de aprendizaje de la red neural.

Queda excluido del alcance del proyecto el desarrollo de aplicaciones visuales que pudieran ofrecer una simulación interactiva del funcionamiento del modelo. Esto podría ser planteado como una expansión de presente TFG, tomando la forma de una tesis máster.

1.5. PLANIFICACIÓN

En la realización de trabajo fin de grado se han empleado un total de 6 meses, desde marzo a agosto del año 2019. A continuación, se exponen las principales tareas realizadas y el cronograma asociado a las mismas (ver Tablas 1.5.1):

- Recopilación de documentación, búsqueda bibliográfica: libros, documentos de investigación y otros trabajos realizados por terceros (tesis, tfgs..)
- Elaboración del marco teórico
- Obtención de datos de entrenamiento
- Desarrollo e implementación del modelo
- Análisis de resultados

- Elaboración de las conclusiones

Tabla 1.1. Cronograma de actuación para el trabajo fin de grado reflejado en horas de trabajo.

Actividad	Mes 1	Mes 2	Mes 3	Mes 4	Mes 5	Mes 6
Recopilación de información	20					
Estudio mercado de divisas y aprendizaje profundo	10	20				
Estudio de las librerías software necesarias		5	5			
Obtención de los datos necesarios			5			
Diseño de la solución			20			
Desarrollo e implementación del sofware				5	5	20
Análisis de los resultados						10
Redacción de la memoria	5	5	5	5	30	40

Fuente: Elaboración propia.

CAPÍTULO II: MARCO TEÓRICO Y ESTADO DEL ARTE

2.1. MERCADO DE DIVISAS

El mercado de divisas, conocido como Forex o FX, es un mercado descentralizado dónde los activos son divisas. Se trata del mercado más grande del mundo con un volumen de negociación que supera con creces cualquier mercado centralizado o no centralizado de todo el globo.

Las divisas se negocian en cruces, en tipos de cambio entre dos monedas distinta. El cruce de divisas más negociado es el del EUR/USD seguido de lejos por el cruce del dólar/yen o USD/JPY.

Al ser un mercado no centralizado, no existe ningún *exchange*, estilo bolsa de Madrid o bolsa de Frankfurt que refleje un precio único de cruce de divisas. Estos precios van a depender de los miembros o agentes que participan en el mismo mercado.

Otra característica importante de estos mercados, es que son mercados de 24h, esto es, no existe un cierre estricto de mercado. En la práctica se considera el cierre de la bolsa de New York como referencia para la toma de cruces de cambio diario.

Además de New York, se consideran como importantes núcleos de negociación de divisas a la bolsa de Londres y Tokio. Las aperturas y cierras de estas bolsas son tenidas muy en cuenta por los operadores bursátiles ya que es cuando se toman las referencias de cambio que se aplican al día siguiente en las transacciones bancarias.

2.1.1. ACCESO AL MERCADO FOREX

Los operadores bursátiles comúnmente conocidos como traders acceden al mercado Forex mediante intermediarios llamados brókers. Estos brókers actúan entre los traders y los proveedores de liquidez.

A menudo los proveedores de liquidez son las propias instituciones financieras. Los brókers son necesarios para que, de una forma transparente, los traders puedan realizar transacciones con ellas, ya que estas no aceptarían volúmenes de negociación extremadamente bajos. Por lo tanto, es el propio bróker es el encargado de enrutar las órdenes de una forma acumulada hacia el proveedor de liquidez.

A veces los propios brókers hacen de contrapartida directamente de las órdenes de los traders. Esto supone un conflicto de intereses, ya que la pérdida de los traders es un beneficio para el propio bróker. Esto ha y sigue causando muchos problemas, hasta el punto de que los reguladores han tenido que intervenir, imponiendo fuertes sanciones a brókers, por comportamientos no adecuados con sus clientes: ofrecimiento de peores precios de los reales, retraso en la ejecución de órdenes, no aceptación selectiva de transacciones…

En cualquier caso, para los operadores no institucionales con volúmenes de negociación medios-bajo, el acceso al mercado a través de un bróker sigue siendo necesario.

2.1.2. PLATAFORMAS DE TRADING

Para operar, los brókers ofrecen distintas plataformas para acceder a la liquidez del mercado. Estas plataformas ofrecen una interfaz más o menos sencilla que permite a los traders realizar operaciones.

Las plataformas actuales más comunes son las siguientes:

- Metatrader 4 (http://www.metatrader4.com)
- Metatrader 5 (http://www.metatrader5.com)
- JForex3 (https://www.dukascopy.com/land/trading/bank/platforms/jforex3/)
- cTrader (https://ctrader.com/)

Todas estas plataformas ofrecen un interfaz visual (Figura 1) dónde se puede observar la evolución del precio por unidad de tiempo.

Figura 2.1. Vista de velas en metatrader 4

Fuente: Elaboración propia

En las plataformas se encuentran disponibles distintas opciones de visualización, permitiendo así modificar los intervalos de tiempo del gráfico (Figura 2). Los intervalos de tiempo disponibles en las plataformas suelen ser:

- Mensual
- Semanal
- Diario
- 4 Horas
- 1 Hora
- 30 minutos
- 15 minutos
- 5 minutos
- 1 minuto

Figura 2.2. Opciones de intervalos de tiempo

Fuente: Elaboración propia

La forma de representar la evolución del precio suele ser en forma de 'vela' (Figura 3). Esta vela contiene, de forma esquemática, los valores de apertura, valor más alto, valor más bajo y cierre del intervalo de tiempo que ocupa. Comúnmente para referirnos a datos en forma de vela nos referimos a datos OHLC (open, high, low, y close en inglés).

Figura 2.3. Detalle de una vela.

Fuente: Elaboración propia

Las operaciones bursátiles (Figura 4) que permiten estas plataformas son:

- Órdenes ejecutadas a **mercado**:
  - Compra a mercado de X lotes.
  - Venta a mercado de X lotes
- Órdenes **pendientes** (ejecutadas cuando el precio alcance un valor dado):
  - Compra límite de X lotes. Sólo se puede colocar por debajo del nivel actual de precio.
  - Venta límite de X lotes. Sólo se puede colocar por encima del nivel actual de precio.
  - Compra stop de X lotes. Sólo se puede colocar por encima del nivel actual de precio.
  - Venta stop de Y lotes. Sólo se puede colocar por debajo del nivel actual de precio.

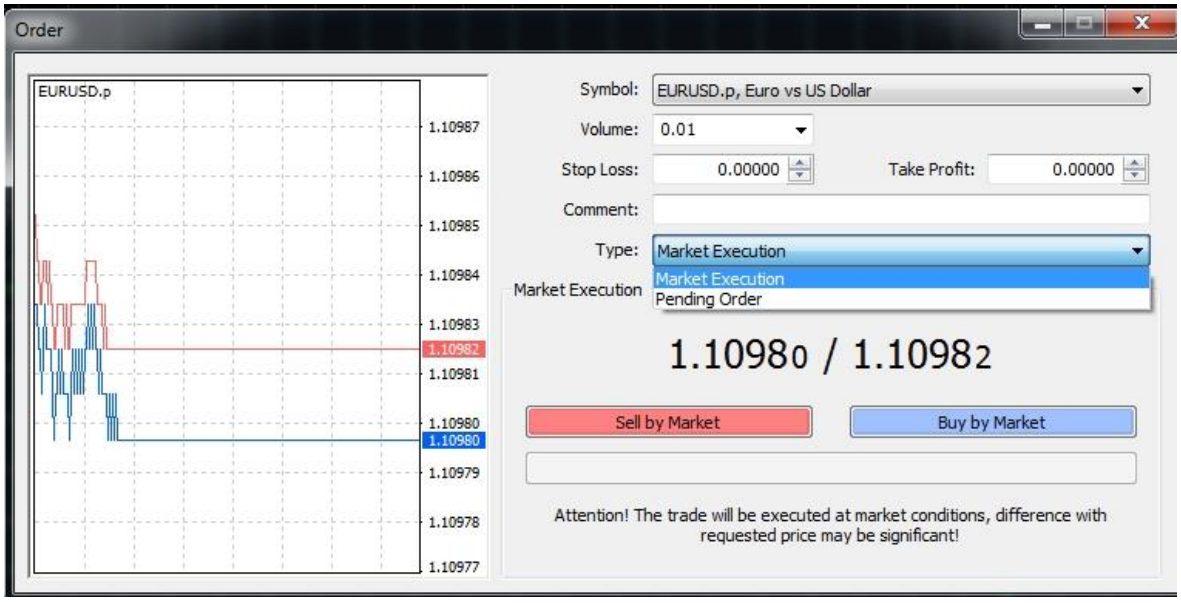

Figura 2.4. Detalle de interfaz de órdenes de ejecución a mercado.

Fuente: Plataforma metatrader 4

Además de las operaciones mencionadas, las plataformas permiten especificar un precio máximo de pérdida, que en el caso de alcanzarlo la operación se cerraría automáticamente causando una pérdida. Por otro lado, también es posible especificar un precio de toma de beneficios, el cual una vez alcanzado lanza el proceso de cierre de operación, en este caso causando

un beneficio. Esto no deja de ser una simplificación, ya que aunque se alcancen los niveles de cierre tanto al alza como a la baja, la orden no tiene por qué ejecutarse a ese precio, ya que la orden de cierre se ejecuta con el precio actual de mercado que, debido a la latencia del propio bróker, podría diferir en unas décimas de punto, causando una desviación de precio, también llamada *slippage*. Esto es especialmente cierto cuando se opera durante períodos de máxima volatilidad como son los eventos de anuncios de los bancos centrales, cambios de política monetaria o tasas de empleo de los Estados Unidos.

2.1.3. INDICADORES TÉCNICOS

Las plataformas de inversión proporcionan una serie de herramientas, llamadas indicadores técnicos, que ayudan al operador en su toma de decisiones. Además de incluir indicadores estándar, a menudo, el propio bróker proporciona acceso a un lenguaje de programación para incorporar nuestros propios indicadores. Estos indicadores pasarán a estar disponibles en la plataforma de igual forma que los 'estándar'.

Algunos de los indicadores técnicos más usados son (cita):

- Media móvil (SMA)
- RSI
- CCI
- MACD
- Bandas de Bollinger

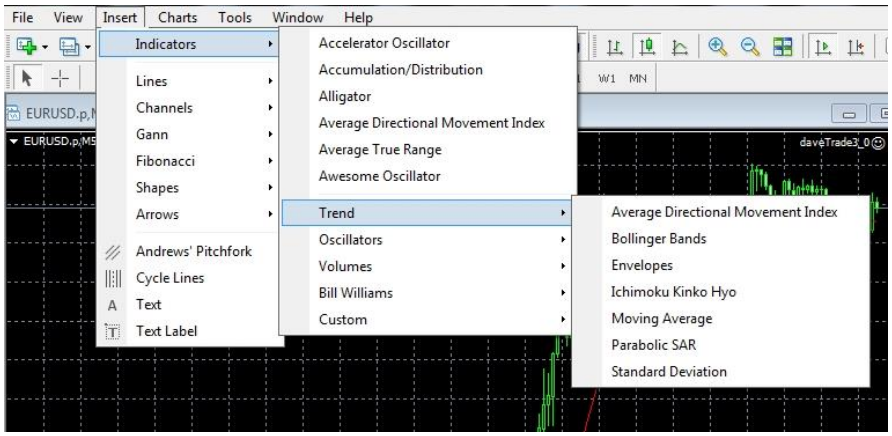

Figura 2.5. Indicadores técnicos en plataforma metatrader 4

Fuente: Plataforma Metatrader 4

Figura 2.6. Media Móvil (20) en gráfico de velas

Fuente: Elaboración propia

2.1.4. MOVIMIENTO EN EL MERCADO DE DIVISAS

En el mercado de divisas, el mínimo movimiento aceptado, es de 0.00001 puntos para los pares como el EUR/USD. Sin embargo, la unidad más importante de movimiento es el llamado *pip* que representa un movimiento de 0.00010 en el mercado del EUR/USD

2.1.5. ESTRATEGIAS DE TRADING

El conjunto de estrategias de operaciones de trading podemos dividirlas, en la mayoría de los casos, en dos categorías:

- **Operaciones de continuación de tendencia o de *breakout***. En este tipo de operaciones se trata de conseguir puntos de mercado aprovechando la posible continuación de un movimiento anterior ya iniciado con anterioridad, podemos observar el ejemplo en la figura 2.1.4.1, la rotura del precio 1.1075 hace que se entre con una orden de compra, siendo posible obtener beneficios ya que el precio sigue subiendo.

- **Operaciones de retroceso de tendencia o de *reversal***. A veces conocidas también como de reversión a la media, porque lo que buscan es conseguir obtener puntos de mercado aprovechando que el precio puede volver a un valor ya visitado en u periodo de tiempo anterior. En la figura 2.1.4.2, se produce un nuevo *retest* del precio 1.1150, entrando con una orden de venta se podría obtener un beneficio ya que el precio continúa bajando a precios ya visitados con anterioridad.

Figura 2.7. Breakout

Fuente: Elaboración propia

Figura 2.8. Operación de retroceso

Fuente: Elaboración propia

2.1.6. EVALUACIÓN DE LAS ESTRATEGIAS DE TRADING

Las estrategias de trading son evaluadas desde el punto de vista de cuál ha sido su retorno o **beneficio** durante un periodo de tiempo y cuál ha sido su mayor caída desde un beneficio alcanzado dado. Este último concepto también conocido como ***Drawdown*** en inglés.

Otras métricas importantes son las siguientes:

- **Porcentaje de aciertos.** Es la tasa de operaciones ganadores sobre el total de operaciones, medida en tanto por 100.
- **Factor de beneficio** o Profit Factor (PF) en inglés. Se obtiene al dividir el beneficio de las operaciones positivas entre las pérdidas de las operaciones negativas. Se mide en valor absoluto. Una estrategia rentable debe de tener un PF>1.0.

2.1.7. COSTES DE TRANSACCIÓN

Los costes de transacción son aquellos costes derivados de la realización de una operación bursátil. Suelen ser de dos tipos:

- **Spread** fijo o variable. Es la diferencia entre el precio de compra y el de venta actual.
- **Comisión** del bróker. Es un importe fijo que impone el bróker por cada operación completa de compra-venta de un activo.

2.2. APRENDIZAJE AUTOMÁTICO

Cuando hablamos de aprendizaje automático o aprendizaje máquina nos referimos a un tipo de estructuras que aprenden a tomar decisiones sin la necesidad de ser indicadas a priori o explícitamente.

A pesar de que muchas de las ideas y algoritmos de tipo aprendizaje máquina han estado alrededor nuestro durante muchos años, es durante los últimos años, cuando se ha empezado a aplicar en cada vez en más campos de nuestra sociedad. Esto está ocurriendo a dos importantes factores:

- **Disponibilidad de datos**. La gente pasa más tiempo realizando actividades que dejan rastro digital, y por lo tanto, datos que pueden servir para desarrollar modelos de aprendizaje máquina.
- **Aumento de la capacidad computacional.** Gracias a la capacidad CPU y GPU de las nuevas generaciones de ordenadores, se pueden entrenar modelos cada vez más complejos.

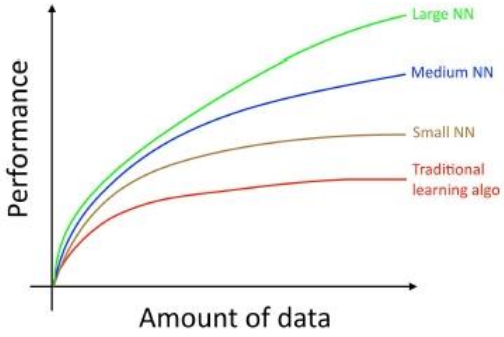

Figura 2.9. Evolución rendimiento frente a cantidad datos

Fuente: Coursera

Los campos de aplicación dónde nos encontramos el aprendizaje automático hoy en día, son múltiples (cita):

- Conducción autónoma
- Visión por ordenador
- Chatbots
- Sistemas de recomendación
- Predicción de la demanda industrial
- Predicción de tendencias en mercados financieros

2.2.1. TIPOS DE APRENDIZAJE

Existen tres tipos de aprendizaje:

- **Aprendizaje supervisado**. En este tipo de aprendizaje el modelo aprende a mapear un conjunto de datos X con su salida Y. Para ello se proporciona un conjunto de datos de entrenamiento cuya clase es conocida (ver Figura 2.10 con clase X y clase O). De esta forma el algoritmo aprende las relaciones existentes entre los datos/características de entrada y su clase asociada. Algunos de los algoritmos supervisados serían:
    - **K vecinos más próximos**

      https://es.wikipedia.org/wiki/K_vecinos_m%C3%A1s_pr%C3%B3ximos
    - **Redes neuronales**

      https://es.wikipedia.org/wiki/Red_neuronal_artificial
    - **Máquinas de vectores de soporte**

      https://es.wikipedia.org/wiki/M%C3%A1quinas_de_vectores_de_soporte

- **Clasificador Bayesiano ingenuo**

  https://es.wikipedia.org/wiki/Clasificador_bayesiano_ingenuo

- **Árboles de decisión**

  https://es.wikipedia.org/wiki/%C3%81rbol_de_decisi%C3%B3n

- **Regresión logística**

  https://es.wikipedia.org/wiki/Regresi%C3%B3n_log%C3%ADstica

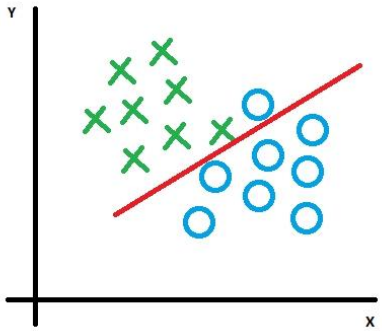

Figura 2.10 Aprendizaje supervisado

Fuente: Elaboración propia

- **Aprendizaje no supervisado**. En este caso el algoritmo aprende a identificar patrones en los datos. No se le proporcionan datos etiquetados, es decir, no se conoce la clase asociada a los datos / características de entrada del modelo. El algoritmo agrupa los datos de acuerdo a los patrones que ha descubierto en ellos (ver Figura 2.11, dónde el algoritmo ha identificado 3 clases distintas. Ejemplos de este tipo de aprendizaje serían:

  - **K-Medias**

    https://es.wikipedia.org/wiki/K-medias

  - **Mezcla de Gaussianas**

    http://catarina.udlap.mx/u_dl_a/tales/documentos/lmt/ramirez_a_e/capitulo3.pdf

  - **Agrupamiento jerárquico**

    https://es.wikipedia.org/wiki/Agrupamiento_jer%C3%A1rquico

  - **Mapas auto-organizados**

    https://es.wikipedia.org/wiki/Mapa_autoorganizado

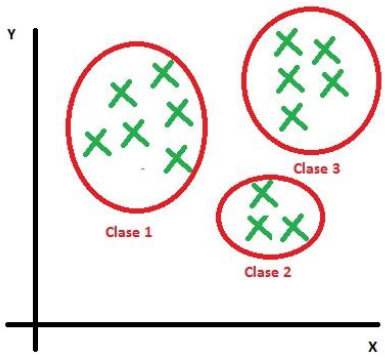

Figura 2.11. Aprendizaje no supervisado. Tres clases.

Fuente: Elaboración propia.

- **Aprendizaje por refuerzo**. En este aprendizaje, el agente debe escoger acciones que maximicen su recompensa en un ambiente dado. Algoritmos basados en este tipo de aprendizaje son:
    - **Programación dinámica**

      https://es.wikipedia.org/wiki/Programación_dinámica
    - **Q-learning**

      https://en.wikipedia.org/wiki/Q-learning
    - **SARSA**

      https://en.wikipedia.org/wiki/State–action–reward–state–action

2.2.2. NEURONA ARTIFICIAL

Las neuronas artificiales son software que procesan unas entradas y generan una salida. Éstas se agrupan en redes neuronales las cuales forman parte de la clasificación de algoritmo de aprendizaje supervisado.

Teniendo como modelo a las neuronas biológicas, las neuronas artificiales se componen de un nodo que procesa las entradas que recibe de acuerdo a unos pesos dados y a una función de activación determinada para, a priori, generar una salida. El aprendizaje del modelo de red neuronal, se basa en la sucesiva actualización del valor de los pesos asignados en cada nodo, minimizando el error de predicción en la salida del modelo.

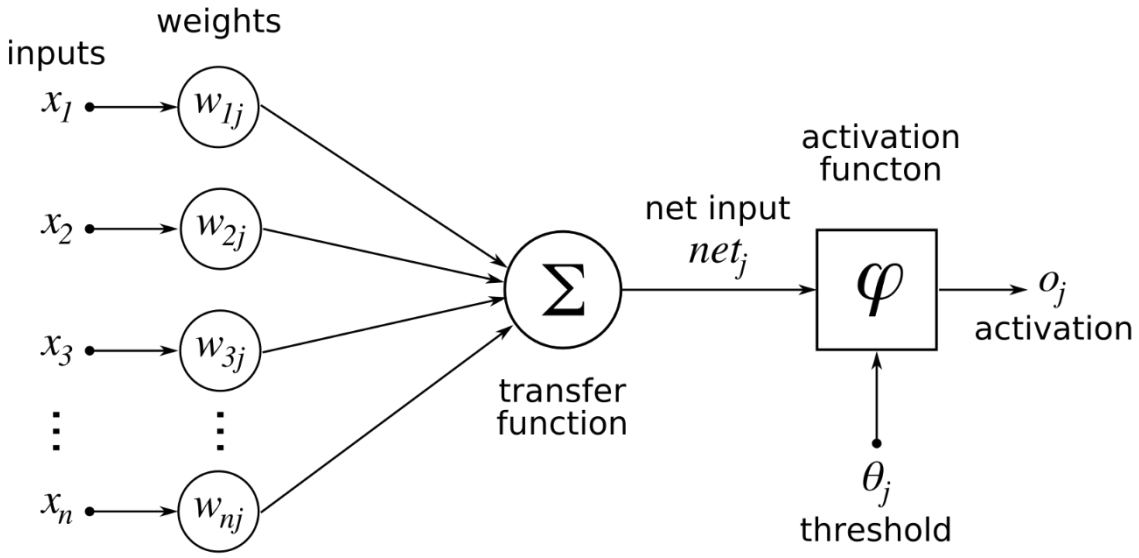

Figura 2.12. Neurona artificial.

Fuente: Wikipedia

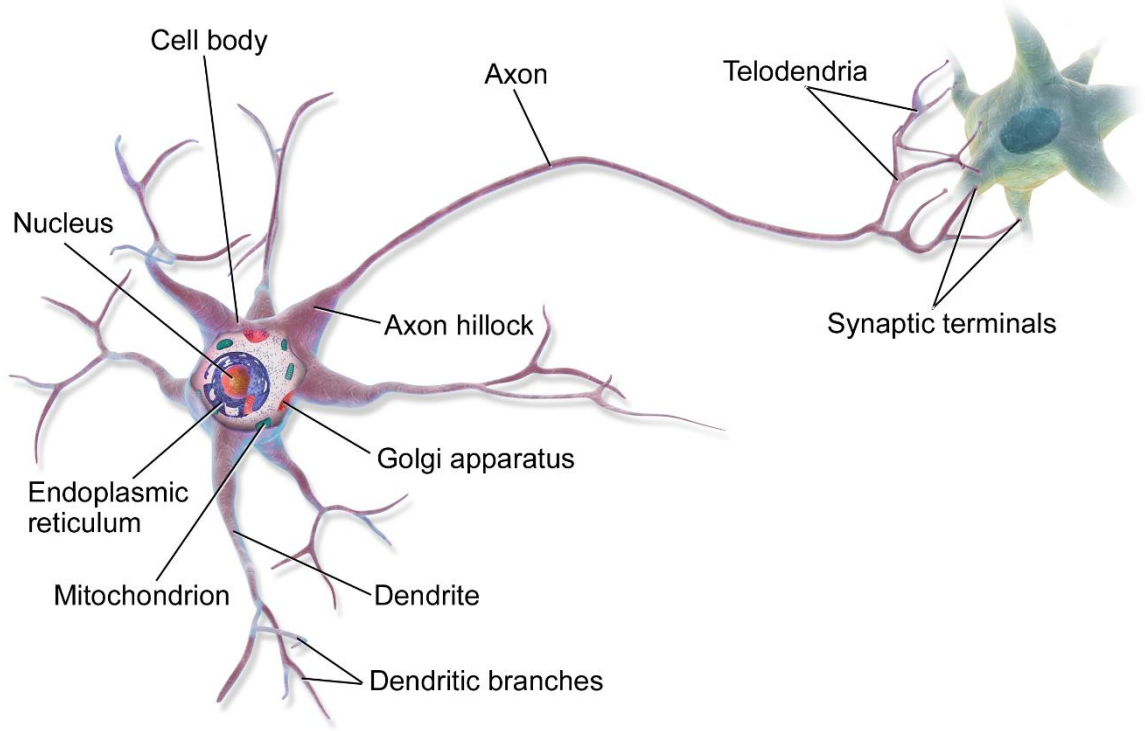

Figura 2.13. Neurona biológica.

Fuente: Wikipedia

En las figuras 2.12 y 2.13 se observan dos neuronas, una artificial y una biológica, respectivamente. En comparativa, las entradas (inputs en la figura 2.12) de una neurona artificial

se correspondería con las dendritas de la biológica. La función de transferencia y activación se correspondería con el cuerpo celular y la activación o salida sería similar a los axones, en las neuronas biológicas.

Las entradas x_i corresponderían a los datos de entrenamiento o a las salidas de otras capas neuronales en arquitecturas más complejas.

Los pesos w_{ij} se asignan a cada entrada de tal manera que sus productos son agregados en el sumatorio de la función de transferencia. Su resultado neto servirá como entrada de la función de activación. La agregación de pesos, se puede expresar de forma matemática de esta manera:

$$valorNeto = W_1 * X_1 + W_2 * X_2 + \ldots + W_n * X_n$$

La salida de la función de transferencia (valorNeto) será computada en la función de activación generando un valor que alimentará siguientes capas neuronales. Algunas de las funciones de activación más usadas actualmente son la función sigmoide, función hiperbólica o la función ReLU (Wikipedia).

2.2.3. REDES NEURONALES. APRENDIZAJE PROFUNDO

La red neural más simple es el llamado *Perceptron* (Rosenblatt, F. 1958). El perceptrón es una red neural de una sola capa, esto es, un conjunto de entrada X unido a sus pesos W y una función de activación que devuelve una salida (ver figura 2.12). Se trata de un simple modelo lineal de clasificación binaria con una relación simple entrada- salida.

El precursor del perceptrón fue el *Threshold Logic Unit (TLU)* desarrollado por McCulloh y Pitts en 1943. Este modelo podía aprender las puertas lógicas AND y OR. McCulloh y Pitts introdujeron el concepto de análisis de actividad neuronal basada en umbrales y sumas ponderadas que fueron los conceptos claves para la evolución y aparición del perceptrón.

El perceptrón apareció en 1957 en el Cornell Aeronautical Laboratory, siendo su inventor Frank Rosenblant. Las implementaciones iniciales eran muy limitadas, la primera implementación software fue para el IBM 704 y más tarde fue implementado en el Mark I Perceptron

El perceptrón simplemente sumará las entradas con sus pesos asociados junto a un valor llamado bias (b) obteniendo un valor neto, este valor neto alimentará la función de activación que simplemente será una función que devolverá 1 si el valor neto supera un umbral dado y 0 en otro caso. La definición formal de esto sería:

$$f(x) = 1\ si\ x > \theta; 0\ si\ x \leq \theta$$

Podemos observar cuáles son los parámetros que intervienen en el perceptrón en la Tabla 2.1.

Tabla 2.1. Parámetros del perceptrón.

Parámetro	Descripción
W	Vector de valores reales con los pesos de las conexiones
*W*X*	Producto escalar. Computa una suma ponderada ($\sum_1^n w_i x_i$)
n	Número de entradas del perceptrón
b	El concepto de bias (sesgo)

Fuente: Elaboración propia.

El algoritmo de aprendizaje del perceptrón cambiará los valores de los pesos W hasta que todas las entradas X sean correctamente clasificadas. Si el problema no es linealmente separable el proceso de actualización de los pesos W no terminará nunca. Un ejemplo de función no linealmente separable es la función XOR (ver Figura 2.14). El perceptrón no encuentra pesos W que mapeen las entradas X con las salidas esperadas Y de una tabla XOR. Este problema causó gran frustración en la creciente comunidad que se estaba generando entorno al aprendizaje máquina y causó el llamado 'Invierno de la Inteligencia Artificial' en la década de los 70. A pesar de que se descubrió que un perceptrón multicapa podría resolver muchos de los problemas no lineales planteados, no fue hasta mediados de la década de los 80 cuando el aprendizaje máquina volvió a resurgir con la popularización del algoritmo llamado *Backpropagation* (Linnainmaa, S. 1970). Este algoritmo consigue adaptar los pesos W propagando los errores encontrados en la clasificación desde las últimas capas a las primeras capas en sentido inverso.

x_0	x_1	y
0	0	0
0	1	1
1	0	1
1	1	0

Figura 2.14. Función XOR

Fuente: Elaboración propia

La unión de múltiples capas neuronales da lugar a las llamadas redes neuronales. Si estas redes neuronales no son circulares son llamadas como *Feed-Forward Neural Networks*. Por lo

tanto, las redes neuronales son una concatenación de sucesivas capas de neuronas, dónde las entradas de la capa n son las salidas de las funciones de activación de la capa n-1.

La estructura de una red neuronal estaría formada por:

- **Capa de entrada**. Contienen las entradas y pesos asociados. Preceden a las capas ocultas. En las redes neuronales clásicas todas las entradas están completamente conectadas con las neuronas de la siguiente capa oculta.
- **Capas ocultas.** Reciben como entradas las salidas de la capa anterior y la transforman a través de la función de activación.
- **Capa de salida**. Tras general la salida, se comprueba el error generado y se procesan los pesos de las conexiones de la red usando el algoritmo *backpropagation*.

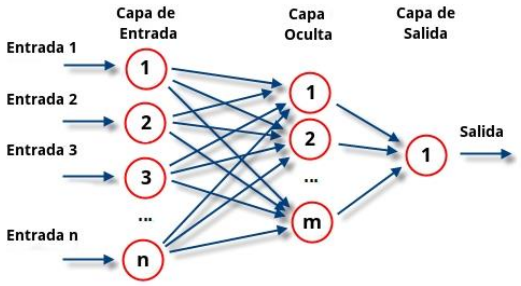

Figura 2.15. Red neuronal multicapa.

Fuente: Wikipedia

En este momento, con las redes neuronales multicapa, toman relevancia las llamadas funciones de activación. Ya que van a ser estas las responsables de interconectar y propagar los distintos valores hacia las capas más avanzadas. El tipo de función de activación es a nivel de capa, es decir, dos capas diferentes pueden tener dos funciones de activación también diferentes.

Tabla 2.2. Algunas funciones de activación

Función	Descripción	Rango
Identidad	f(x) = WX	$(-\infty, +\infty)$
Escalón	f(x) = H(x), 1 si x >= 0, 0 si x<0	(0,1)
Sigmoidea	$f(x) = 1 / (1+e^{-x})$	(0,1)
Tanh	f(x) = sinh(x) /cosh(x)	(-1,1)
ReLU	f(x) = max(0,x)	(0,x)

Fuente: Elaboración propia.

Otro concepto es el de función de error. La función de error computa la discrepancia entre el valor predicho y el valor actual real de un conjunto de datos. El modelo es más robusto mientras más pequeño sea el error, luego el objetivo a la hora de entrenar una red neuronal es minimizar este error.

Tabla 2.3. Algunas funciones de error

Función	Descripción
MSE	Error cuadrático medio
MAE	Error absoluto medio
XENT	Cross Entropía, o log error

Fuente: Elaboración propia.

2.2.3.1. DATOS DE ENTRADA. CONJUNTOS DE ENTRENAMIENTO Y DE DESARROLLO.

A la hora de alimentar una red neural de datos, estos de dividirán en dos conjuntos: el conjunto de datos de entrenamiento y el conjunto de datos de validación o de desarrollo.

El algoritmo de aprendizaje usará los datos de entrenamiento para aprender la función que mapea el conjunto de entrada X con el conjunto de salida Y. Posteriormente una vez que el entrenamiento haya finalizado usaremos el conjunto de evaluación para evaluar la validez del modelo.

A la hora de determinar qué conjunto de los datos planos formarán parte de uno u de otro, debemos considerar el tamaño total de este. Una técnica habitual ha sido la del 80-20, es decir un 80% para el conjunto de entrenamiento y un 20% para el conjunto de desarrollo. En la época actual, la época del big data, si por ejemplo se tienen 1 millón de datos, se podría considerar un 95-5 o incluso un 99% para el conjunto de entrenamiento.

Otro aspecto a considerar, es el formato de los datos de entrada. Los datos de entrada forman un vector X que contienen números reales o naturales. En los problemas de clasificación binaria, dónde la salida va a tomar o el valor 0 o el valor 1, podría ser conveniente formar vectores de entrada con valores binarios 0, 1. De tal formar que un individuo x_i de entrada con 7 características podría ser (0,1,1,1,1,1,0). En cambio, si nos encontramos en un problema de regresión, dónde la salida Y, es el precio del par de divisas, podríamos utilizar valores reales, como por ejemplo los valores del activo en tiempos anteriores:

$$X_i = (1.2345, 1.230, 1.220, 1.2250)$$

2.2.3.2. ENTRENAMIENTO DE LA RED NEURAL

El objetivo de toda red neuronal es tener unos parámetros (pesos W) que amplifiquen la señal y mapeen correctamente el vector de entrada X a la salida Y. Aquellos pesos más grandes tendrán mayor peso en la red neural que aquellos que son menores.

La forma de entrenar la red, es procesando los pesos y los bias, ajustándolos de tal manera que unos crezcan y otros bajen en valor, y así de esta manera unas características tendrán más importancia que otras. Para ello se utiliza un algoritmo de optimización llamado **gradiente descendiente.** El gradiente descendiente es un algoritmo iterativo de optimización que trata de buscar los mínimos de funciones convexas y diferenciables. En este caso se trata de minimizar el error entre la predicción realizada por la red y los datos de salida esperados para un conjunto de entrada dado.

El algoritmo del gradiente descendente forma parte del proceso de actualización del modelo neural. La forma de actualizarlo sigue la del algoritmo conocido como *backpropagation*, que actualiza los pesos de las capas anteriores de acuerdo al error encontrado y repartiéndolo a través de los distintos pesos.

2.2.3.3. REDES PROFUNDAS

Cuando hablamos de redes profundas, las diferenciamos respecto a las redes clásicas en:

- Tienen más neuronas
- Tienen formas más complejas de conectar las distintas capas
- Ha habido un incremento exponencial en la potencia de entrenamiento
- Existen herramientas de extracción de características

Las capas no tienen por qué estar completamente interconectadas, existen arquitecturas que pueden conectar los nodos de distinta manera como pasa en las redes convolucionales (CNNS) y en las redes recurrentes.

Al ser redes más complejas tienen más parámetros a optimizar era necesario un aumento de la capacidad de computación, que a su vez facilitó el desarrollo de redes más inteligentes capaces de extraer características de los datos de una forma más automática.

El campo del aprendizaje profundo se encuentra en continua evolución en campos como el reconocimiento de imágenes, reconocimiento de voz, desarrollo de inteligencia superior a humana en videojuegos, conducción autónoma, traducción de textos...

Este aumento de la complejidad de las redes neuronales ha traído consigo la aparición y necesidad de calibrar conceptos como hiperparámetros, regularización, paralelización, uso de GPUs..Esto hace necesario la elección de un framework / librería de desarrollo adecuado que incluya todos estos conceptos para una correcta implementación de cualquier solución.

2.2.3.4. PARÁMETROS E HIPERPARÁMETROS

Los parámetros del modelo de red neural son W y b, que son los pesos y bias /sesgos respectivamente. Los hiperparámetros son parámetros que controlan la actualización y modificación de W y b. Es decir, los hiperparámetros intervienen en que forma o manera toman y cambian de valor los pesos y los sesgos.

En la tabla 4 podemos encontrar algunos de estos hiperparámetros.

Tabla 2.4. Parámetros red neural

Parámetro	Descripción
W	Pesos de las distintas conexiones
b	Bias o sesgo

Fuente: Elaboración propia.

Tabla 2.5. Algunos hiperparámetros

Nombre	Descripción
Ratio de aprendizaje	Controla cómo evolucionan los parámetros
Número de épocas	Número de veces que se van a actualizar cada uno de los datos de entrenamiento
Número de capas ocultas	1..10..30..
Número de unidades por capa	5,10,20…
Función de activación	Tanh(x),sigmoid…
Tamaño del batch	Número de vectores de entrenamiento que se evaluarán antes de que se actualice el modelo., es decir, la cantidad de 'samples' que se procesarán antes de que se termine la época actual (epoch en inglés)

Fuente: Elaboración propia.

2.2.3.5. MEDIDAS DE RENDIMIENTO

A la hora de medir la idoneidad de un modelo de red neural es necesario usar alguna métrica que exprese la calidad de la solución para el problema planteado. Para esto se formula la matriz de confusión (ver Tabla 2.6) y sus métricas asociadas.

Las métricas son las siguientes:

- **Precisión**. Es la proporción entre las predicciones correctas y el total de predicciones. Precisión = (tp+tn) / (tp+tn+fp+fn)
- **Exactitud**. Se refiere por la proporción de los aciertos reales del modelo y el total de los casos positivos. Exactitud = (tp) / (tp+fn)
- **Sensibilidad**. También conocida como tasa de verdaderos positivos. Se define como la proporción de casos positivos identificados correctamente. Sensibilidad = (tp) / (tp+fp)
- **Especificidad**. También conocida como tasa de verdaderos negativos. Especificidad = (tn)/ (tn+fn)

Tabla 2.6. Matriz de confusión

Real	Predicción	
	Positivo	*Negativo*
Positivo	tp	fn
Negativo	fp	tn

Fuente: Elaboración propia.

2.2.3.6. PROBLEMA DEL OVERFITTING. REGULARIZACIÓN

El concepto de *overfitting* se refiere al ajuste elevado, a veces perfecto, a los datos de entrada. Encontrando de esta manera patrones y relaciones entre los datos que no son correctas, a menudo casuales y simple ruido. En las figuras 2.16, 2.17 y 2.18 podemos observar este concepto gráficamente. En la figura 2.16, el ajuste no es demasiado bueno y se dice que se encuentra *underffited* o con sesgo alto. En la figura 2.17, nos encontramos justo el caso contrario, el ajuste a los datos de entrada es perfecto, luego se encuentra en estado de *overfitting*. En la figura 2.18 encontramos el caso deseado, un estado balanceado entre lo que sería tener un alto sesgo o una alta varianza.

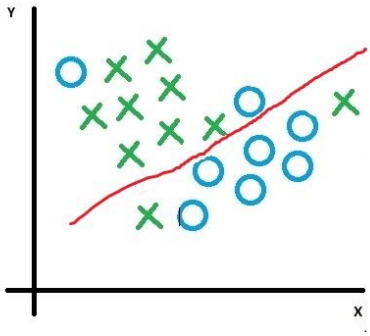

Figura 2.16. Underfitting. Sesgo alto

Fuente: Elaboración propia

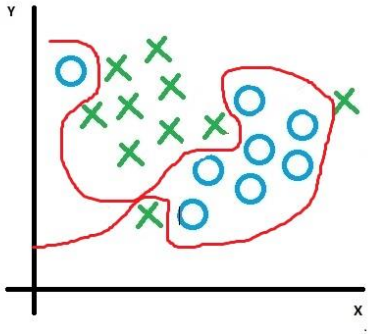

Figura 2.17. Curvefitting. Varianza elevada

Fuente: Elaboración propia

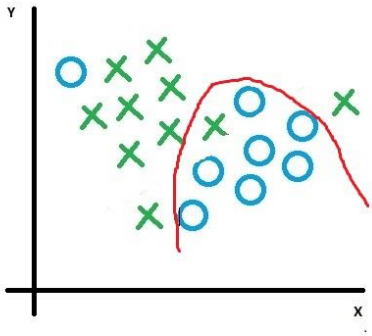

Figura 2.18. Balanceado. Bias y varianza ajustadas.

Fuente: Elaboración propia

Para determinar si un modelo en encuentra con un alto o sesgo o alta varianza, debemos de comparar los errores que nos encontramos en los conjuntos de entrenamiento y validación / desarrollo. Si estos errores difieren mucho, podríamos encontrarnos en uno de los dos casos no deseados. Para determinar si el sesgo es alto, debemos comparar, en la medida de lo posible, con el porcentaje de error a nivel humano. En la tabla 2.7 podemos encontrar algunos de los casos posibles y valores que nos dan idea del concepto.

Para considerar una alta varianza comparamos los errores con el conjunto de desarrollo frente al conjunto de entrenamiento, si la diferencia es claramente elevada, nos enfrentamos a un caso de varianza alta. En el último caso podemos observar que tiene tanto un sesgo y varianza pequeñas. El sesgo es pequeño porque para este caso hemos comparado con un error a nivel humano del 0%, y el error de entrenamiento no se encuentra lejos de ese valor (0.5%).

Tabla 2.7. Ejemplo sesgo-varianza.

Error entrenamiento	Error desarrollo	Sesgo / varianza
1%	*11%*	Varianza elevada
15%	*16%*	Sesgo elevado
15%	*30%*	Sesgo elevado y varianza elevada
0.5%	*1%*	Sesgo bajo y varianza baja

Fuente: Elaboración propia.

Las soluciones a estos dos problemas (sesgo y varianza) que se han ido proponiendo a lo largo del tiempo son las que se pueden observar en la tabla 2.8. Al principio del desarrollo del aprendizaje profundo no se encontraban herramientas para disminuir tanto sesgo como varianza, siempre surgía una especie compromiso, de la forma que por ejemplo si disminuía el sesgo, la varianza se veía afectada y viceversa. Actualmente con técnicas como las que encontramos en los algoritmos de regularización, que ayudan a disminuir la varianza y que no afectan en demasía al sesgo siempre que tengamos una red suficientemente grande.

Algunas de las técnicas de regularización son (CITA):

- Regularización L1
- Regularización L2
- *Dropout*
- Parada temprana

L1 y L2 penalizan los grandes pesos W y los decrementan. La regularización **dropout** desactiva unidades de activación basándose en una probabilidad P. Siendo P a menudo 0.5, lo que quiere decir que, en una iteración dada, una unidad de activación concreta tiene un 50% de desactivarse, que es esencialmente poner su peso a 0.

La parada temprana, dejará de entrenar la red cuando se alcancen algunas de estas situaciones:

- Máximo número de épocas
- Tiempo máximo excedido
- No encuentra mejora en un número determinado de épocas

Tabla 2.8. Soluciones a problemas de sesgo y varianza

Problema	Solución
Sesgo elevado	Red neural más grande Entrenamiento más largo Cambio de arquitectura de red neural
Varianza elevada	Conseguir más datos Regularización Cambio de arquitectura de red neural
15%	Sesgo elevado y varianza elevada
0.5%	Sesgo bajo y varianza baja

Fuente: Elaboración propia.

2.2.5. FRAMEWORK DL4J

Deeplearning4j, DL4J, es la primera librería de nivel comercial, abierta y distribuida para los lenguajes Java y Scala. Se encuentra integrada con Hadoop y Apache Spark. Permite el despliegue en entornos distribuidos tanto de CPUs como de GPUs.

DL4J es de código abierto con licencia Apache 2.0. Trabaja conjuntamente con otras librerías de desarrollo de aprendizaje máquina como son Tensorflow y Keras.

DL4J incluye implementaciones de:

- Máquina de Boltzmann
- Deep Belief Net
- Deep Autoencoder
- Stacked Autoencoded
- Recursive Neural Tensor Network

- Word2vec
- Doc2vec
- GloVe

Todos estos algoritmos incluyen versiones distribuidas paralelas que se integran en Apache Hadoop (cita) y Spark

El framework incluye librerías para la monitorización del modelo, ver figuras 2.2.5.1 y 2.2.5.2

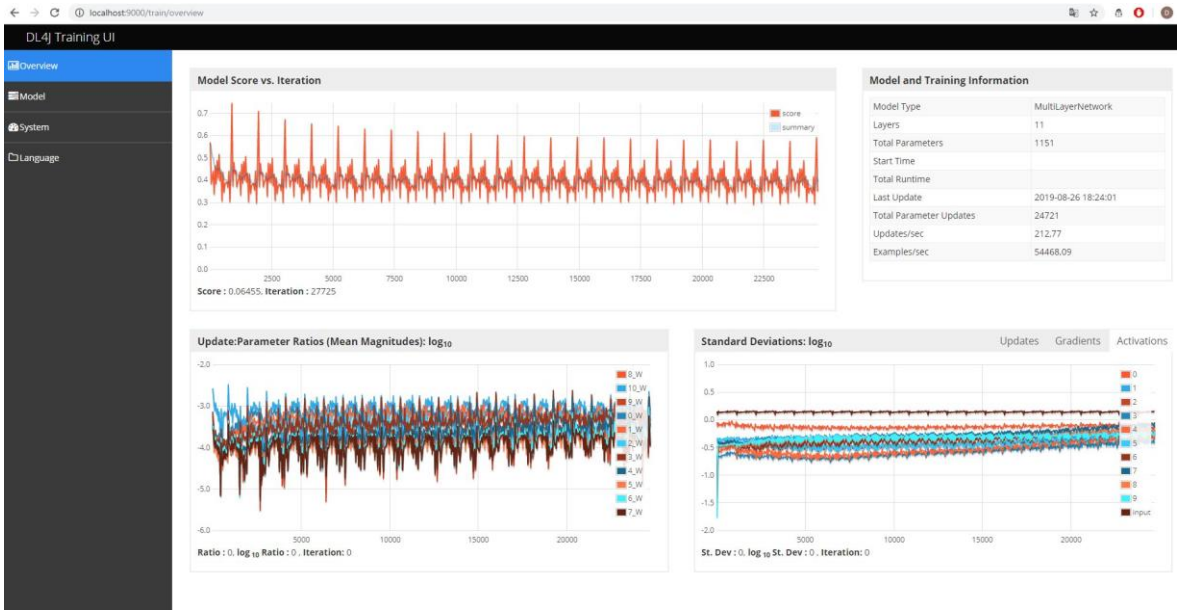

Figura 2.19. DL4J Training UI. Vista de Overview

Fuente: Elaboración propia

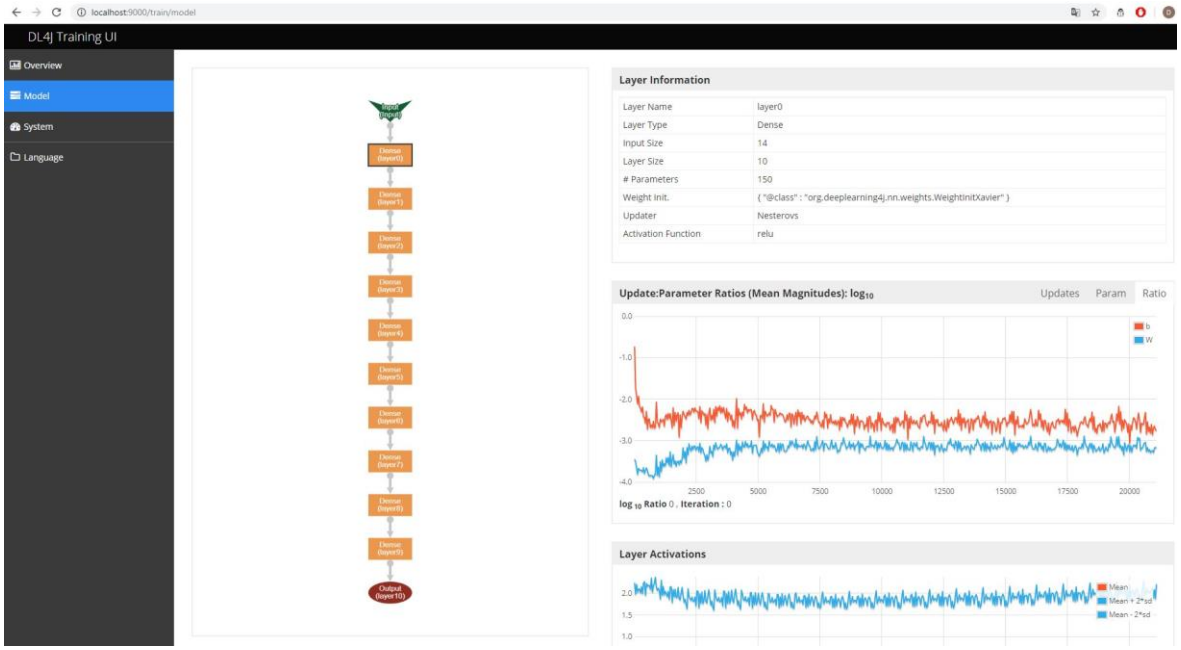

Figura 2.20. DL4J Training UI. Vista de Model

Fuente: Elaboración propia

2.2.6. ESTADO DEL ARTE DEL DEEPLEARNING A JUNIO 2019

En el informe elaborado por stateof.ai (https://www.stateof.ai/), hace constancia de algunos alcances obtenidos durante el último año en referencia al campo de la inteligencia artificial (ver figura 16).

Se hace referencia a como se ha conseguido supera al humano en muchos videojuegos y cómo los más complejos empiezan a ser abordados (Wikipedia Montezuma). Uno de los grandes logros ha sido generar un algoritmo capaz de derrotar a uno de los mejores jugadores de Starcraft II, un juego que integra duros desafíos para sistemas de aprendizaje automático como son: operar con información imperfecta, controlar un gran espacio de acción en tiempo real y hacer decisiones estratégicas sobre un largo horizonte de tiempo.

Otro de los logros en el campo, se ha producido en el químico. El programa AlphaFold (https://deepmind.com/blog/article/alphafold) predice la estructura 3D de las proteínas.

En el campo del procesamiento del lenguaje natural (NLP) se ha empezado a aplicar la técnica de transferencia de aprendizaje con éxito. Varios investigadores (Google AI's,OpenAI's Transformer..) han demostrado que modelos de lenguaje preentrenados pueden mejorar el rendimiento en una variedad de tareas NLP.

Avances en medicina:

- Diagnóstico de enfermedades oculares

- Detección y clasificación de arritmias cardiacas usando ECGs
- Reconstrucción de discurso desde la actividad neuronal del cortex

AutoML, el framework de Google que automatiza el proceso de creación de algoritmos de aprendizaje, añade una evolución en su proceso de búsqueda de hiperparámetros, aplicando algoritmos evolutivos para optimizar tanto los hiperparámetros como la arquitectura, obteniendo así redes efectivas más pequeñas.

FHI (https://www.fhi.ox.ac.uk/govai/) prevé que existe un 54% de probabilidades de que se alcance un alto nivel de inteligencia máquina para 2028. Como alto nivel de inteligencia máquina se entiendo que las máquinas podrán realizar todas las tareas económicamente relevantes hoy en día.

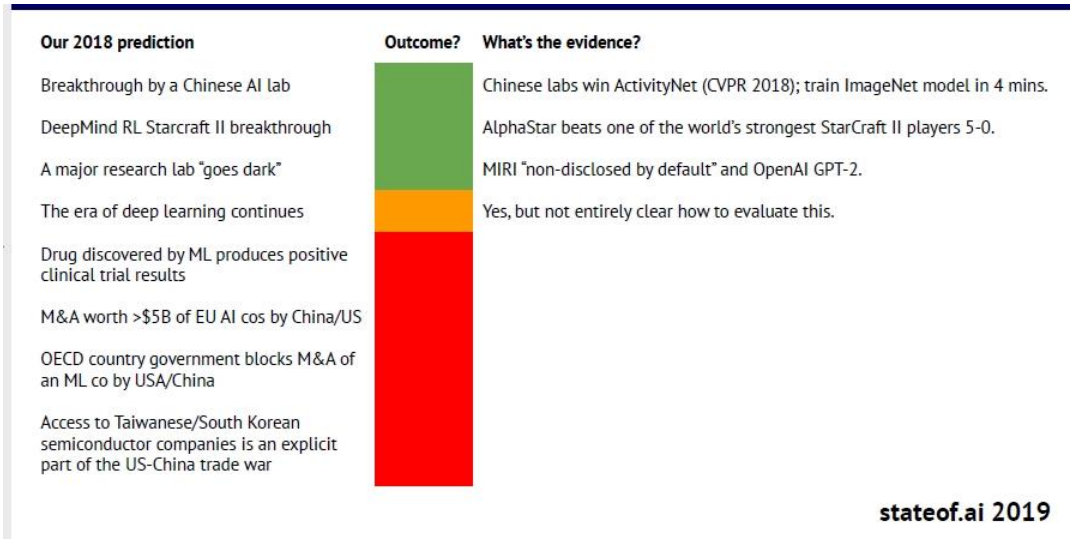

Figura 2.21. Predicciones realizadas en 2018 y resultado.

Fuente: Elaboración propia

2.2.7. APRENDIZAJE AUTOMÁTICO EN EL MERCADO DE DIVISAS. TRABAJO RELACIONADO

La literatura científica acerca del aprendizaje máquina en la aplicación sobre los mercados financieros, ha ido en aumento en los últimos años. A menudo han utilizado estrategias híbridas con otras tecnologías de optimización para la búsqueda de soluciones.

Algunos autores como Abreu,G. (2018), proponen sistemas híbridos de aprendizaje híbrido con algoritmos genéticos. En su estudio logra un retorno sobre inversión del 10.29%.

Kuroda,K. (2017) usa redes neuronales y estrategias ensambladas con algoritmos genéticos en el par USD/JPY, logra un retorno sobre inversión del 15.79%. Petropoulus,A. (2017), propone el uso

de algoritmos genéticos con múltiples algoritmos de aprendizaje máquina, para analizar las correlaciones entre pares de divisas, generando señales intradiarias.

Sidehabi, et al.(2016) propone un sistema de métodos estadísticos junto con aprendizaje máquina para predecir el precio EUR/USD.

Otros autores usan otro tipo de arquitecturas, fuera de las tradicionales *full-connected* redes neuronales. Así Yung-Cheng T. et al (2018) usa redes neuronales convolucionales para generar un modelo de trading. Sin embargo, no consigue obtener un rendimiento aceptable.

Di Persio et al.() compraran distintos tipos de arquitecturas para predecir los índices bursátiles. En esta comparación usan 3 tipos de redes: perceptrón, convolucionales y LSTM (cita). Encuentran que una de las combinaciones usadas entre wavelets y convolucionales alcanzan una precisión del 83%.

Olden M. (2016) usa algoritmos de clasificación binaria para predecir el mercado de acciones de Oslo. Trata de predecir el movimiento de 22 acciones bursátiles con hasta 37 técnicas de aprendizaje máquina. Sus resultados muestran que algunas de las técnicas pueden efectivamente, predecir las cotizaciones de esas acciones.

CAPÍTULO III: ANÁLISIS Y RESULTADOS

3.1. SOLUCIÓN PROPUESTA

Para la realización de los experimentos se usarán arquitecturas de redes neurales con varias capas ocultas. Se tratarán de redes neuronales en el sentido clásico con todos los nodos conectados.

La entrada de datos serán vectores formados por características predictivas extraídas de los datos planos del mercado de divisas del EUR/USD. La salida deberá indicar si se alcanza un precio objetivo antes que el precio de máxima perdida prevista. La codificación de la salida será de 1 si se alcanza el objetivo y de 0 si no se alcanza.

Se han creado dos tipos de modelos, uno para **operaciones de compra (BUY)** y otro para **operaciones de venta (SELL).** Posteriormente, se han combinado ambos para comprobar si se mejora a los modelos aislados.

Tanto los datos de desarrollo como los de entrenamiento / validación se etiquetarán según el modelo al que vayan ser aplicados (ver figuras 3.1 y 3.1):

- Si se va a entrenar un modelo para compras (BUY), las etiquetas de los datos serán 1 si el precio sube a un determinado nivel previsto y 0 si el precio baja a un nivel por debajo del precio máxima de pérdida.
- Si se va a entrenar un modelo para ventas (SELL), las etiquetas de los datos serán 0 si el precio baja a un determinado nivel previsto y 1 si el precio sube por encima del precio máximo de pérdida.

Figura 3.1. Etiquetado en un modelo de venta.

Fuente: Elaboración propia

Figura 3.2. Etiquetado en un modelo de compra.

Fuente: Elaboración propia

La salida de los modelos de redes neurales vendrá dada por la función sigmoide y será una salida única. Esta función se encuentra acotada entre los valores 0 y 1, tomando cualquier valor de tipo real entre estos límites. Para que un modelo de señal de entrada, ya sea de compra o de venta dependiendo del tipo, necesita que el valor de salida sigmoide sea superior a un umbral dado.

Ejemplo: con un parámetro de entrada de 0.55 en el modelo de compra:

- Si la predicción de la red neural es 0.60, el modelo indicará señal de compra.
- Si la predicción de la red neural es 0.45, el modelo no da señal.

Se puede deducir, que un modelo dado sólo puede realizar una operación, de compra o de venta si el modelo es para BUY o para SELL respectivamente. Si la señal no se activa (por encima del umbral dado), no se realiza ninguna operación.

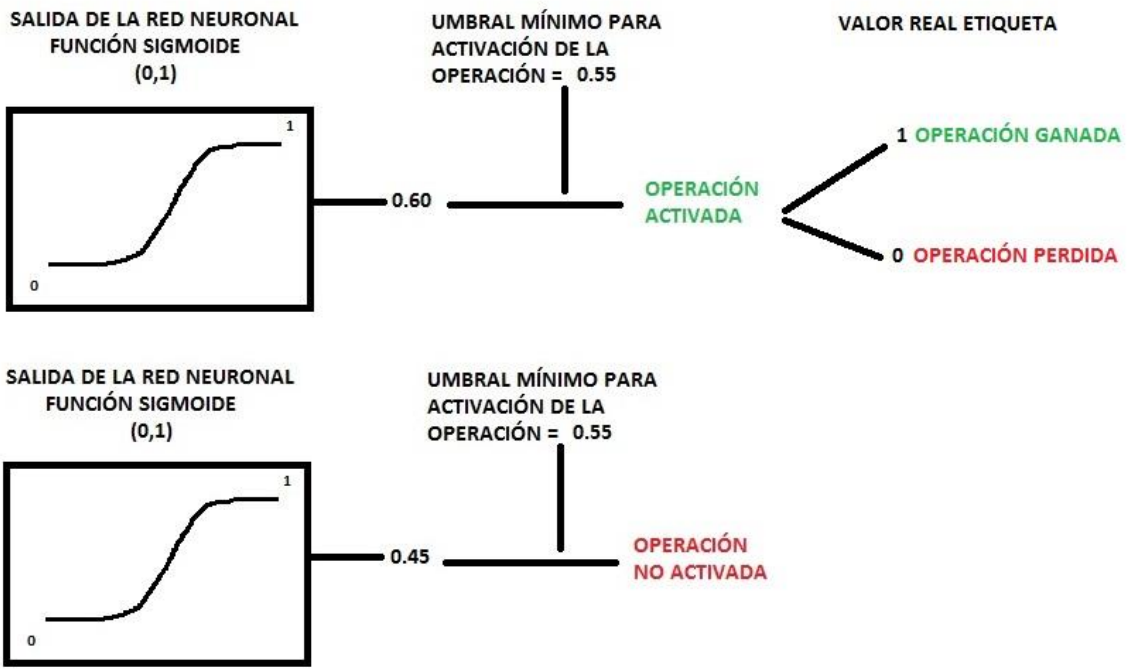

Figura 3.3. Activación de operaciones.

Fuente: Elaboración propia.

El porcentaje de éxito en la predicción se medirá como el total de predicciones activadas (por encima del umbral) y que estuvieran etiquetadas a 1.

En el modelo conjunto (BUY + SELL), se usarán los datos de test de ambos modelos BUY/SELL y dependiendo de las señales dadas por ambas, se escogerá un tipo de operación. Cada modelo tendrá en cuenta si su salida de la red neural da un valor por encima de un umbral dado y las decisiones operativas dependerán de si se han activado una o ambas señales.

Las **reglas para la toma de decisiones** son las siguientes:

- Si las señales son contradictorias, es decir, el modelo de compra da BUY y el modelo de venta da SELL, el modelo conjunto no opera.
- Si el modelo de compra da BUY y el modelo de venta no da SELL, el modelo conjunto opera con un BUY.
- Si el modelo de venta da SELL y el modelo de compra no da SELL, el modelo conjunto opera con un SELL.

En los distintos experimentos se considerarán distintas combinaciones de los valores de los siguientes hiperparámetros:

- **Número de capas ocultas**
- **Número de unidades por capa oculta**
- **Ratio de aprendizaje**

- **Funciones de activación.** RELU para los nodos ocultos y SIGMOID para la capa de salida.
- **Tamaño del batch.**
- **Número de épocas**
- **Porcentaje de operaciones realizadas**. Se considera un hiperparámetro porque afecta a la elección de una arquitectura u otra. La existencia de este hiperparámetro se debe a que no es suficiente con entrenar una red, es necesario que esa red active un mínimo de operaciones para que el modelo sea viable.

Para el entrenamiento y la evaluación del modelo se considerará un **reparto del conjunto de datos 80-20**, significando esto que un 80% de los datos se corresponderán al entrenamiento y un 20% se usarán para el conjunto de validación o desarrollo. En este caso, no se usarán datos para el conjunto de test.

Para el procesamiento de los datos planos se usarán clases java desarrolladas para este propósito. Como librería de apoyo de aprendizaje automático, se usará el framework **DL4J** que, en conjunción con las clases mencionadas anteriormente, permitirán realizar todos los pasos necesarios para la experimentación:

- Carga de datos planos
- Procesado y extracción de características desde los datos planos
- Construcción del modelo de red profunda
- Proceso de entrenamiento
- Proceso de evaluación

Se realizarán un total de dos experimentos, el primero de ellos evaluará los modelos BUY y/o SELL por separado y en el segundo experimento se evaluará el modelo conjunto. El formato de presentación de los datos, serán tablas de doble entrada, dónde se especificará el porcentaje de acierto del modelo. En algunos casos, dónde el proceso no haya conseguido un modelo que supere el requisito de operar en al menos un 10% de las veces, aparecerá en blanco.

3.2. OBTENCIÓN DE DATOS

Los datos se han obtenido de la plataforma JForex (cita) del bróker suizo Dukascopy (ver Figura 3.2.1). Este bróker ofrece una plataforma de descarga sencilla que proporciona datos suficientemente estables y sin errores.

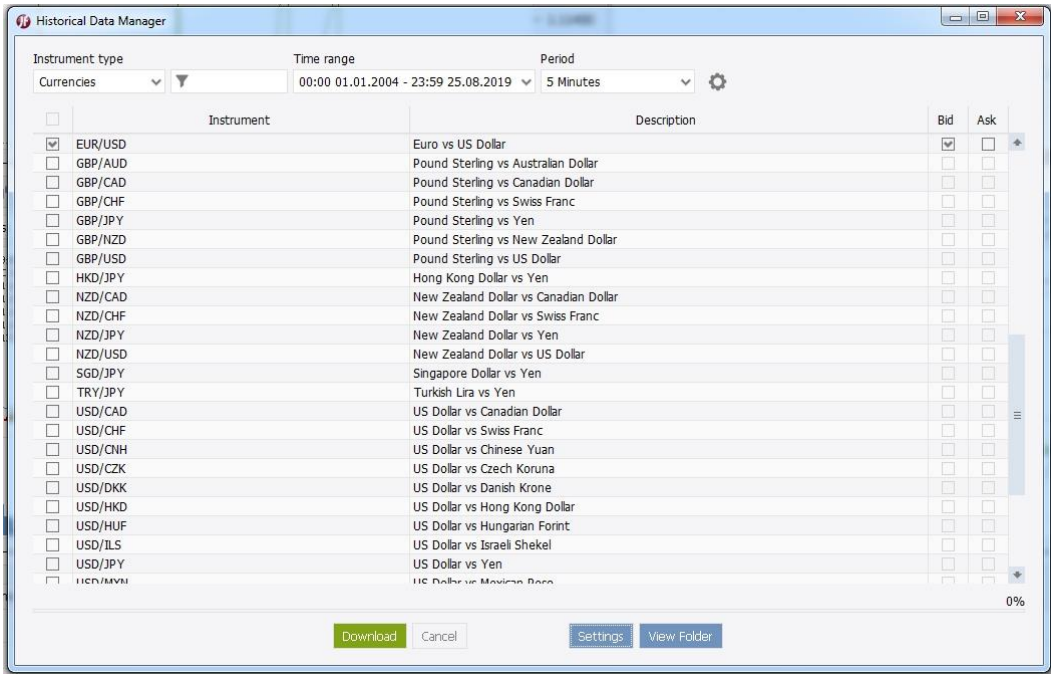

Figura 3.4. Plataforma de descarga de datos de Dukascopy.

Fuente: Plataforma Dukascopy

Se han obtenido datos planos para el par EUR/USD desde el 1 de Enero de 2004 al 25 de Agosto del 2019. Se han considerado los datos para los intervalos de tiempo de 15 minutos.

Se han considerado 3 bloques da datos para los experimentos, cada uno de ellos en sus respectivos conjuntos de entrenamiento y de validación:

- Bloque 1:
    - Entrenamiento: desde el 1 de Enero del 2009 hasta el 31 de Diciembre del 2014, haciendo un total 149856 entradas.
    - Validación: desde el 1 de Enero del 2015 hasta el 31 de Diciembre del 2015, haciendo un total de 24956 entradas.
- Bloque 2:
    - Entrenamiento: desde el 1 de Enero del 2009 hasta el 31 de Diciembre del 2015, haciendo un total 199722 entradas.

- Validación: desde el 1 de Enero del 2016 hasta el 31 de Diciembre del 2016, haciendo un total de 24923 entradas.

- Bloque 3:
  - Entrenamiento: desde el 1 de Enero del 2009 hasta el 31 de Diciembre del 2018, haciendo un total 249616 entradas.
  - Validación: desde el 1 de Enero del 2019 hasta el 25 de Agosto del 2019, haciendo un total de 16128 entradas.

3.3. EXTRACCIÓN DE CARACTERÍSTICAS

Para formar el conjunto de entradas de la red neural, es necesario obtener las características predictivas de los datos planos. Se van a utilizar vectores de entrada de hasta 30 elementos cada uno (dependiendo del experimento). Estos elementos son características extraídas de los datos planos, mediante algoritmos desarrollados para tal propósito. Las características podemos verlas reflejadas en las tablas 3.1 y 3.2.

Tabla 3.1. Características de cada entrada al modelo

Nombre	Número de elementos	Definición
Hora del día	*5*	*Hora del día de la vela actual de entrada. Su valor numérico va del 0 al 23 y se codifica en binario, dando como resultado 5 entradas formas por 0s y 1s. Ejemplo. 14 = 0 1 1 1 0*
Umbral de entrada	*1*	*Parámetro que indica si el valor ALTO de la vela justo anterior es mayor o menor que las 'n' anteriores velas ALTAS o BAJAS respectivamente.*
Umbral de entrada(binario)	*1*	*Parámetro que indica si el valor ALTO de la vela justo anterior es mayor o menor que las 'n' anteriores velas ALTAS o BAJAS respectivamente. Indicará con un 1 si la vela anterior es superior a 'n' y 0 en caso de que la vela anterior sea inferior a n' anteriores*
hval5	*1*	*Indica si el precio de apertura de la vela actual es superior a 5 pips del máximo del día anterior*
lval5	*1*	*Indica si el precio de apertura de la vela actual es inferior a 5 pips del mínimo del día anterior*
hval10	*1*	*Indica si el precio de apertura de la vela actual es superior a 10 pips del máximo del día anterior*

Fuente: Elaboración propia.

Tabla 3.2. Características de cada entrada del modelo (continuación)

Nombre	Número de elementos	Definición
diff30	*1*	*Indica la diferencia en pips entre el precio actual y la media móvil de 30 periodos*
diff40	*1*	*Indica la diferencia en pips entre el precio actual y la media móvil de 40 periodos*
diff50	*1*	*Indica la diferencia en pips entre el precio actual y la media móvil de 50 periodos*
inputsDiff(n)	*n*	*Array con las diferencias del precio actual y los n anteriores precios. Por ejemplo, si n=5, el array estaría formado por las diferencias entre el precio actual Py los 5 anteriores:*
lval10	*1*	*Indica si el precio de apertura de la vela actual es inferior 10 pips del mínimo del día anterior*
range20	*1*	*Indica si el rango actual del día es superior a 20 pips*
range30	*1*	*Indica si el rango actual del día es superior a 30 pips*
range40	*1*	*Indica si el rango actual del día es superior a 40 pips*
range50	*1*	*Indica si el rango actual del día es superior a 50 pips*

Fuente: Elaboración propia.

Las características que se van a usar son conocidas dentro de las plataformas de operaciones bursátiles. La excepción a esto, es el indicador de Umbral de Entrada.

El llamado Umbral de Entrada se trata de un indicador de posible resistencia o soporte, en el sentido de que el precio es superior /inferior a las 'n' aneriores velas. Cómo se puede observar en la Figura 3.5, en el caso del recuadro azul, la vela más a la derecha tiene un valor ALTO, que es superior a las 4 velas anteriores, con lo cual el valor del indicador sería 4. El recuadro rojo indica

otro ejemplo pero pa el caso de un valor BAJO. Podrá tener formato tanto binario como de número natural, si es binario, indicará que el valor supera un nivel dado y se indicará como 1, 0 en caso contrario

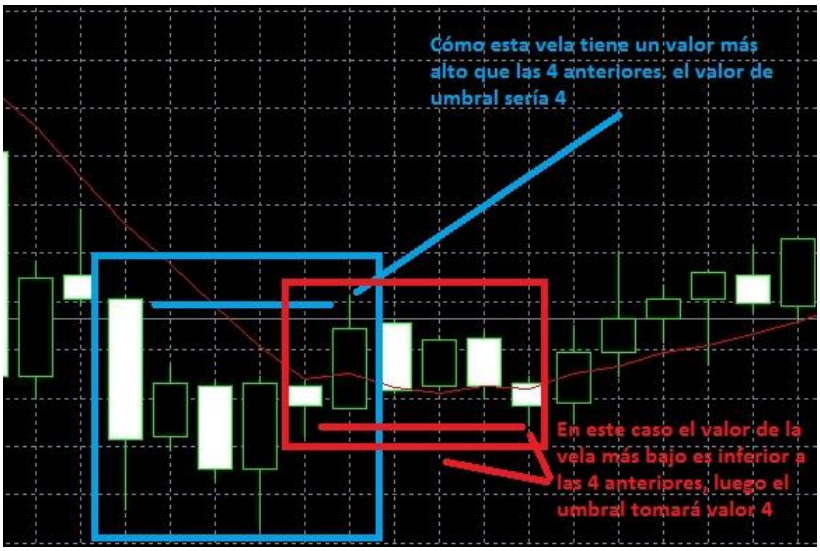

Figura 3.5. Explicación del indicador Umbral.

Fuente: Elaboración propia

3.4. PREPARACIÓN DE DATOS

Los datos de entrada para la red neural se han preprocesado de tal forma que no existan números de gran magnitud entre ellos. Los indicadores a los que se ha aplicado esta técnica aparecen en la tabla 3.3.

Tabla 3.3. Preprocesado aplicado a cada característica

Característica	Preprocesado
diffN	*Se divide entre 600. Un rango anual habitual en el tipo de cambio EUR/USD es alrededor de 600, esto dejará el valor de esta característica con un valor, bajo, a menudo entre 0 y 1*
Umbral de entrada	*1) Limita el máximo de su valor, si el valor límite es 500 y el valor de la característica es 788, se actualizará a 500* *2) Una vez el valor limitado, se dividirá por ese valor. Siguiendo el ejemplo actual, se dividiría por 500 y daría un valor de 1.0. En definitiva, el valor de entrada se convierte al rango [0,1]*
range	*Se divide entre 600. El rango diario suele estar alrededor de los 600. El valor final preprocesado de este indicador estará a menudo entre 0 y 1*

Fuente: Elaboración propia

3.5. ALGORITMO. PROCESO.

El proceso general para un **único** modelo, seguirá los pasos de extracción de características, selección de hiperparámetros, fase de entrenamiento de la red neural y evaluación del mejor modelo en los datos de test.

El proceso del algoritmo en el caso del experimento con **varios** modelos (ver Figura 3.6):

1) Extracción de características para cada modelo
2) Selección de los hiperparámetros no modificables
3) Entrenamiento de cada modelo, quedando el mejor de ellos
4) Evaluación conjunta, con toma de decisiones de acuerdo a las reglas de varios modelos

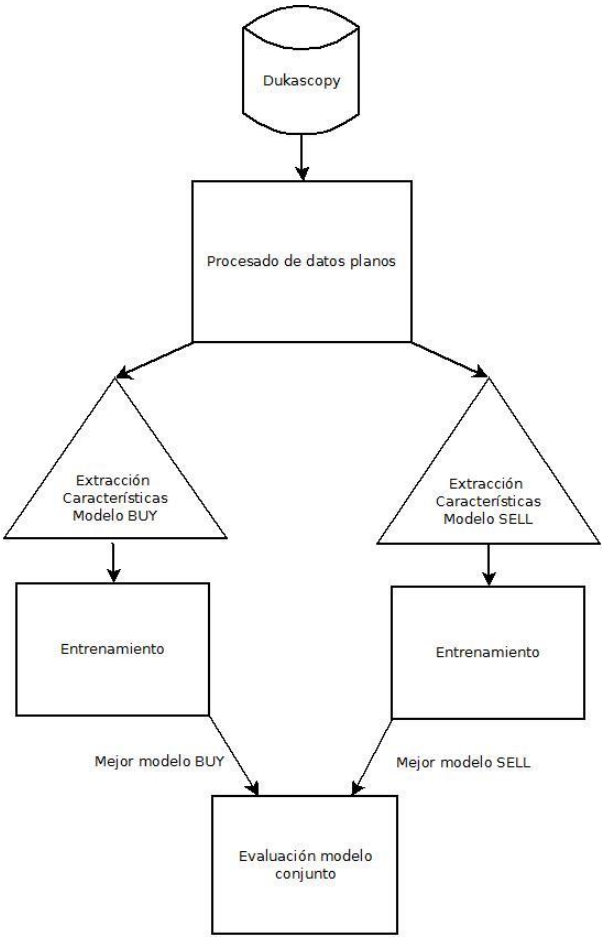

Figura 3.6. Proceso de entrenamiento y evaluación

Fuente: Elaboración propia.

Todo el código y documentación adjunta podrá ser consultada en:

https://github.com/davidautentico/Tfg_FxPredictor/tree/master/FxPredictor

3.6. RESULTADOS

3.6.1. EXPERIMENTO 1.

En este experimento se han creados modelos distintos para compras y para ventas. Los modelos se han entrenado y validado en los 3 bloques de los conjuntos de entrenamiento.

El número de nodos en las capas ocultas es de 15 y el número de capas ocultas es de 1. La función de activación de la capa de salida es la SIGMOID y la función de activación de capas intermedias es la RELU.

Se ha añadido un mecanismo de parada temprana:

- Máximo tiempo de entrenamiento: 60 segundos
- Máximo número de iteraciones: 50 épocas
- Parada si no se consigue mejora en: 1 épocas

Se ha entrenado la red neural en los 3 bloques de entrenamiento considerados y se ha medido la exactitud o porcentaje de aciertos en el conjunto de datos de desarrollo.

3.6.1.1. USANDO FUNCIÓN DE ERROR XENT

En este experimento se definirá como función de error la función XENT.

En las tablas 3.4 y 3.5 se muestran los resultados para el modelo de BUY y el SELL respectivamente. Podemos observar que el modelo BUY consigue para los bloques 1 y 2, alcanzar porcentajes de acierto por encima del 50%, sólo tiene problemas en el bloque 2 con el conjunto de desarrollo.

El modelo SELL sólo se aplica en el bloque 1, ya que el prerrequisito o hiperparámetro *Porcentaje de operaciones realizadas* se encuentra por debajo del 10% para el caso de la arquitectura usada en este experimento. Los resultados obtenidos por este modelo en el bloque 1 se encuentran por encima del 50%, lo que indica que podría ser una estrategia razonable para obtener un retorno positivo.

Tabla 3.4. Modelo para operación de compras con red neural. Red con 1 capa oculta y 15 unidades ocultas. Bloques de datos 1,2 y 3.

Umbral	Tipo	Entren. 1(%)	Desarr. 1 (%)	Entren. 2 (%)	Desarr. 2 (%)	Entren. 3 (%)	Desarr. 3 (%)
0.40	*BUY*	51.11	50.26	50.03	51.71	50.03	45.57
0.45	*BUY*	53.48	51.26	50.37	51.85	50.05	45.65
0.50	*BUY*	59.05	54.62	51.71	52.50	50.55	45.95
0.55	*BUY*	64.61	61.29	55.88	55.77	51.12	40.30
0.60	*BUY*	65.27	64.00	62.20	61.67	50.52	
0.65	*BUY*	67.20	67.83	63.65	63.62	55.26	

Fuente: Elaboración propia.

Tabla 3.5. Modelo para operación de compras con red neural. Red con 1 capa oculta y 15 unidades ocultas. Bloques de datos 1,2 y 3.

Umbral	Tipo	Entren. 1(%)	Desarr. 1 (%)	Entren. 2 (%)	Desarr. 2 (%)	Entren. 3 (%)	Desarr. 3 (%)
0.40	*SELL*	50.84	50.84				
0.45	*SELL*	51.19	51.16				
0.50	*SELL*	53.84	52.47				
0.55	*SELL*	57.95	55.39				
0.60	*SELL*	62.93	59.01				
0.65	*SELL*	65.09	61.93				

Fuente: Elaboración propia.

Las tablas 3.6 y 3.7 muestran los resultados de distintos tipos de arquitecturas aplicadas al modelo BUY. En este cao como se puede observar no se encuentran grandes beneficios al aumentar la complejidad de la red, especialmente en el caso del número de capaz. Lo que hace pensar que una capa y suficiente número de nodos puede ser una arquitectura válida para capturar la complejidad.

Tabla 3.6. Modelo para operación de compras con red neural. Variación de nodos

Nodos	Tipo	Entren. 1(%)	Desarr. 1 (%)	Entren. 2 (%)	Desarr. 2 (%)	Entren. 3 (%)	Desarr. 3 (%)
5	*BUY*			49.84	51.66	50.00	45.56
10	*BUY*			49.87	51.66	50.00	45.56
15	*BUY*			49.96	51.70	50.12	45.58
20	*BUY*			49.84	51.66	50.01	45.56
25	*BUY*			49.84	51.66	50.03	45.57
30	*BUY*			49.89	51.66	50.03	45.56
50	*BUY*			49.96	51.68	50.03	45.56
100	*BUY*			49.85	51.66	50.06	45.56

Fuente: Elaboración propia.

Tabla 3.7. Modelo para operación de compras con red neural. Variación de capas con número de nodos igual a 10.

Capas	Tipo	Entren. 1(%)	Desarr. 1 (%)	Entren. 2 (%)	Desarr. 2 (%)	Entren. 3 (%)	Desarr. 3 (%)
1	*BUY*			49.96	51.70	50.12	45.58
2	*BUY*			49.84	51.66	50.04	45.56
3	*BUY*			49.90	51.69	50.03	45.56
4	*BUY*			49.90	51.69	50.03	45.56
5	*BUY*			49.84	51.66	50.00	45.56

Fuente: Elaboración propia.

La tabla 3.8 muestra los resultados para el modelo BUY en el caso de que se varíen los precios objetivo y Stop. Estas variaciones llevan consigo cambios en las etiquetas de cada entrada de datos, y es necesario realizar un nuevo entrenamiento para cada caso.

Para el conjunto de entrenamiento 1, la red neural no ha conseguido ninguna configuración suficientemente buena, para los bloques 2 y se han conseguido configuraciones que arrojan resultados, pero a priori, no se observa resultados concluyentes sobre la idoneidad de una configuración de objetivos u otra.

Tabla 3.8. Modelo para operación de compras con red neural. Red con 1 capa oculta y 15 unidades ocultas. Bloques de datos 1,2 y 3. Variación de objetivo y máxima pérdida

Objetivo (pips)	Pérdida (pips)	Tipo	Entren. 1(%)	Desarr. 1 (%)	Entren. 2 (%)	Desarr. 2 (%)	Entren. 3 (%)	Desarr. 3 (%)
10	*10*	*BUY*			50.07	51.01	50.95	48.37
20	*20*	*BUY*			49.96	51.69	50.12	45.57
30	*30*	*BUY*			49.89	53.90		
40	*40*	*BUY*						
50	*50*	*BUY*						

Fuente: Elaboración propia

3.6.1.2. USANDO LA FUNCIÓN DE ERROR MAE

Las tablas de este apartado representan distintas combinaciones de parámetros para un modelo de compra (BUY) usando como función de pérdida, la función error medio absoluto (MAE).

Las tablas 3.9 y 3.10 muestran los porcentajes de aciertos del modelo en los conjuntos de entrenamiento y de desarrollo para los casos de variación del número de nodos y de capas, respectivamente.

Tabla 3.9. Modelo para operación de compras con red neural. Variación de nodos con 1 capa oculta.

Nodos	Tipo	Entren. 1(%)	Desarr. 1 (%)	Entren. 2 (%)	Desarr. 2 (%)	Entren. 3 (%)	Desarr. 3 (%)
5	*BUY*	54.12	52.88	53.33	52.30	51.48	46.26
10	*BUY*	50.87	50.91	52.63	52.53	51.72	47.96
15	*BUY*	55.12	54.09	51.89	51.58	51.19	46.23
20	*BUY*	52.46	53.21	52.33	52.60	51.34	46.07
25	*BUY*	52.07	51.19	51.69	51.45	53.64	48.06
30	*BUY*	51.71	51.36	51.40	51.89	51.01	45.85
50	*BUY*	51.04	50.83	51.18	51.36	52.63	46.56
100	*BUY*	53.40	51.50	51.97	52.12	54.24	46.76

Fuente: Elaboración propia.

Tabla 3.10. Modelo para operación de compras con red neural. Variación de capas con número de nodos igual a 10.

Capas	Tipo	Entren. 1(%)	Desarr. 1 (%)	Entren. 2 (%)	Desarr. 2 (%)	Entren. 3 (%)	Desarr. 3 (%)
1	*BUY*	50.87	50.91	52.63	52.53	51.72	47.96
2	*BUY*			51.08	51.59	53.07	46.30
3	*BUY*	52.53	53.04				
4	*BUY*	52.56	53.03				
5	*BUY*			50.30	51.53		

Fuente: Elaboración propia.

La tabla 3.11 muestra el nivel de aciertos cuando se mantiene una arquitectura de una capa y 15 nodos, pero se varía en el precio objetivo y de máxima pérdida.

Tabla 3.11. Modelo para operación de compras con red neural. Red con 1 capa oculta y 15 unidades ocultas. Bloques de datos 1, 2 y 3. Variación de objetivo y máxima pérdida.

Objetivo (pips)	Pérdida (pips)	Tipo	Entren. 1(%)	Desarr. 1 (%)	Entren. 2 (%)	Desarr. 2 (%)	Entren. 3 (%)	Desarr. 3 (%)
10	*10*	*BUY*	58.81	58.86	55.64	55.15	56.86	51.14
20	*20*	*BUY*	55.12	54.09	51.89	51.58	51.19	46.23
30	*30*	*BUY*			51.88	52.83		
40	*40*	*BUY*						
50	*50*	*BUY*	49.50	46.76				

Fuente: Elaboración propia.

3.6.2. EXPERIMENTO 2

En este experimento se han combinado los dos modelos creados para BUY y para SELL, de tal forma que la toma de decisiones es colegiala de acuerdo a las reglas explicadas en el apartado 3.1.

Para la evaluación del modelo sobre el bloque 1 se han utilizado la misma arquitectura que el experimento 1, esto es, 1 capa y 15 nodos ocultos. Para los bloques 2 y 3 se han tenido que buscar mejores arquitecturas para conseguir que el modelo SELL diera suficientes operaciones como para poder combinarse con el modelo BUY.

El resto de condiciones son las mismas que en el experimento 1.

3.6.2.1 USANDO LA FUNCIÓN DE ERROR XENT

En este experimento se definirá como función de error la función XENT (cita).

Tabla 3.12. Modelo conjunto BUY + SELL.

Umbral	Tipo	Entren. 1(%)	Desarr. 1 (%)	Entren. 2 (%)	Desarr. 2 (%)	Entren. 3 (%)	Desarr. 3 (%)
0.40	*CONJUNTO*	50.62	50.70	49.91	51.67	50.02	45.57
0.45	*CONJUNTO*	50.70	50.74	49.94	51.68	50.06	45.58
0.50	*CONJUNTO*	50.90	50.82	49.96	51.70	50.12	45.58
0.55	*CONJUNTO*	52.54	51.71	50.00	51.79	50.22	45.65
0.60	*CONJUNTO*	62.02	60.42	50.01	52.02	50.61	45.81
0.65	*CONJUNTO*	64.46	64.68	50.23	52.69	59.18	52.92

Fuente: Elaboración propia.

3.6.2.2 USANDO LA FUNCIÓN DE ERROR MAE

En este experimento se definirá como función de error, la función de error absoluto medio (MAE). La tabla 3.13 muestra los resultados para el modelo conjunto usando la función de error MAE.

Tabla 3.13. Modelo conjunto BUY + SELL.

Umbral	Tipo	Entren. 1(%)	Desarr. 1 (%)	Entren. 2 (%)	Desarr. 2 (%)	Entren. 3 (%)	Desarr. 3 (%)
0.20	*CONJUNTO*	52.89	52.44	53.11	50.31	51.79	48.59
0.25	*CONJUNTO*	52.92	52.49	52.78	51.10	51.79	47.84
0.30	*CONJUNTO*	52.96	52.54	52.57	50.89	51.73	47.34
0.35	*CONJUNTO*	53.01	52.71	52.40	50.67	51.70	46.94
0.40	*CONJUNTO*	53.07	52.77	52.42	50.51	51.73	46.80
0.45	*CONJUNTO*	53.10	52.86	52.40	50.44	51.70	46.70
0.50	*CONJUNTO*	53.16	52.83	52.42	50.40	51.70	46.77
0.55	*CONJUNTO*	53.20	52.89	52.49	50.57	51.69	46.67
0.60	*CONJUNTO*	53.24	52.94	54.48	50.63	51.68	46.76
0.65	*CONJUNTO*	53.25	52.96	52.56	50.82	51.66	46.70
0.70	*CONJUNTO*	53.29	53.06	52.64	51.00	51.65	46.69
0.75	*CONJUNTO*	53.32	53.07	52.74	51.13	51.66	46.64
0.80	*CONJUNTO*	53.31	53.04	52.89	51.38	51.64	46.58
0.85	*CONJUNTO*	53.31	53.13	53.00	51.88	61.64	46.47
0.90	*CONJUNTO*	53.41	53.26	53.29	52.48	51.80	46.60
0.95	*CONJUNTO*	53.53	53.04	53.83	52.65	51.96	46.82

Fuente: Elaboración propia.

3.6.3. DISCUSIÓN DE LOS RESULTADOS

Los resultados arrojan, con claridad, que alcanzan precisiones superiores al 50% en muchos de los modelos y arquitecturas entrenadas. El modelo BUY tomado de forma individual supera el 50% de aciertos tanto para la función de clasificación binaria, XENT, cómo para la función de error MAE.

Por su parte, el modelo conjunto BUY + SELL, también supera en la mayoría de los casos el 50% de aciertos, como se pueden observar en las tablas del experimento 2.

La variación de nodos por capa, no arroja en ninguno de los casos una configuración superior a la otra. Del mismo modo, el aumento del número de capas oculta no mejora en la calidad de la solución. No parece que el aumento de complejidad de la red neuronal ayude a predecir mejor.

La Tabla 3.14 no muestra una superioridad clara de la función de pérdida de MAE frente a XENT. A priori, la función MAE es más adecuada para este problema, ya que al estar filtrando por umbrales (salida función sigmoide), interesa minimizar el valor diferencia entre ese valor predicho (número real) y el valor real. Se perdería precisión si simplemente se redondeara a 1 si la salida es mayor que 0.5. Es posible, que dentro de los datos XENT, haya pocas operaciones activadas, y como las medias no están ponderadas, los datos aparezcan tergiversados.

Tabla 3.14. Resultados acumulados para XENT y MAE.

Función de error	Entren. 1(%)	Desarr. 1 (%)	Entren. 2 (%)	Desarr. 2 (%)	Entren. 3 (%)	Desarr. 3 (%)
XENT	57.43	56.06	51.15	52.75	50.71	45.79
MAE	53.04	52.63	52.53	51.60	52.41	47.01

Fuente: Elaboración propia.

Otro de los aspectos que se observan, es como, dentro de los distintos bloques de entrenamiento y test, el bloque número 3 es el que peor se comporta con diferencia (ver Tabla 3.15). En este bloque, en muy pocas configuraciones se alcanza el 50%, lo que lo hace especialmente difícil de predecir.

Tabla 3.15. Resultados acumulados para todos los modelos.

Función de error	Entren. 1(%)	Desarr. 1 (%)	Entren. 2 (%)	Desarr. 2 (%)	Entren. 3 (%)	Desarr. 3 (%)
Total modelos	54.94	54.25	51.87	52.22	51.76	46.44

Fuente: Elaboración propia.

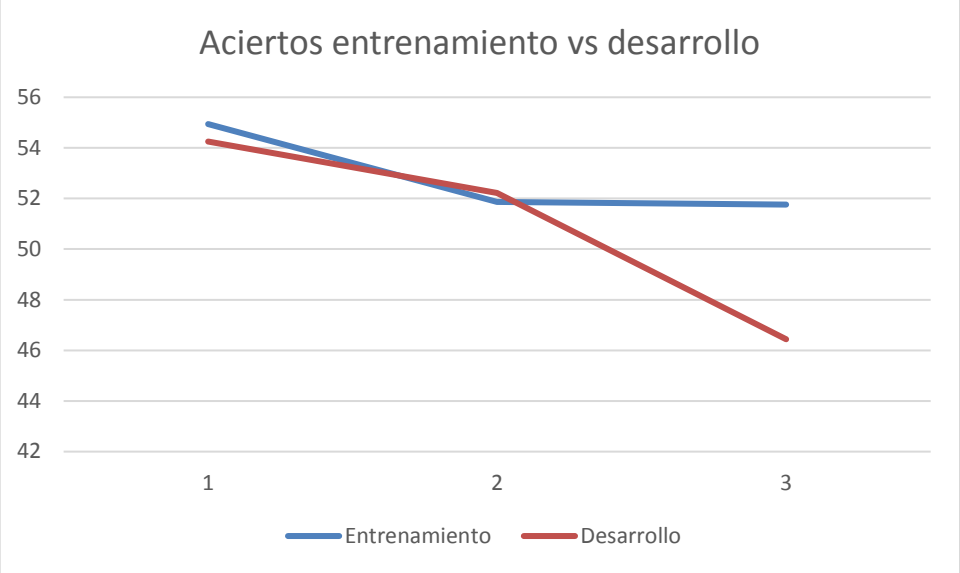

Figura 3.7. Aciertos entrenamiento vs desarrollo.

Fuente: Elaboración propia.

CAPÍTULO IV: CONCLUSIONES Y LÍNEAS FUTURAS

En este trabajo se han desarrollado una serie de modelos de redes neuronales para predecir la tendencia en el movimiento del par de divisas del EUR/USD. La predicción del movimiento de activos financieros y en concreto, del mercado de divisas es muy difícil. De hecho, algunos académicos (Fama E., 1995) insisten en que los mercados están gobernados por *random walk*, el camino aleatorio, siendo imposible predecir de forma rentable sus movimientos. También existe la corriente contraria, la de que no los mercados no siguen un camino aleatorio (Lo Andrew y MacKinlay A.C. ,1988).

En el conjunto de soluciones propuestas, se ha tratado como un problema de regresión, a pesar de que las etiquetas de los vectores de entrada son 1 o 0 en el caso de los experimentos 1 y 2. Esto ha sido así, porque el sistema de trading aplicado a la red neuronal, toma las salidas de la función sigmoide como valores de activación de las operaciones. De esta forma, si el valor de salida superaba un umbral dado, la operación (BUY o SELL) se activaba.

Los modelos desarrollados pueden funcionar aislados o conjuntamente como un comité, de tal forma que si hay discrepancia (BUY y SELL a la vez), la operación conjunta no queda activada.

Para la medición de los resultados, se ha tenido en cuenta el porcentaje de operaciones activadas correctas. Si este porcentaje es superior al 50% el sistema es rentable.

Los resultados arrojados por las simulaciones arrojan, en su mayoría, resultados positivos, especialmente en los periodos de los bloques 1 y 2. Aunque se ha indicado que un sistema es rentable si se encuentra por encima del 50%, esto podría no ser así una vez tenidos en cuenta los costes de transacción. En el caso de este trabajo, no se han tenido en cuenta estos costes añadidos.

El aumento de complejidad de las redes neuronales no ha ayudado, en general, a aumentar la capacidad de predicción del sistema. Es probable que las relaciones entre las entradas y la etiqueta de salida sean difusas, incluso inexistentes y esto dificulta la predicción. Por otra parte, no se ha empleado demasiado tiempo en entrenarlas, lo cual podría ser una mejora.

Como aspectos a trabajar para mejorar la solución:

- Entrenamientos más largos para utilizar redes más complejas
- Estudio de cada característica usada para el vector de entrada y comprobar su poder de predicción
- Añadir más modelos al comité. En este momento sólo se ha probado con un modelo BUY y otro SELL, se podrían agrupar varios modelos BUY y varios modelos SELL y tomar las decisiones de operaciones por voto mayoritario.
- Incluir los costes de transacciones para comprobar la rentabilidad real.
- Utilizar otro tipo de arquitecturas de redes neuronales como las recurrentes.
- Uso de tarjetas gráficas para el uso del procesamiento GPU. De esta manera se podrán hacer más iteraciones en los modelos en el mismo tiempo.

- Añadir nuevas características como entradas de datos. Por ejemplo, algunos autores como son Wei Bao, et al., 2017, añaden un nuevo sistema de aprendizaje usando transformaciones wavelets apiladas.
- Modificación de los datos de tal forma que dejen de estar espaciados por tiempo, y que puedan estar espaciados por volúmenes (López de Prado, M. ,2018).

BIBLIOGRAFÍA

Referencias bibliográficas:

Burkov, A. (2.019) *The Hundred-Page Machine Learning Book.* Kindle Direct Publishing.

López de Prado, M. (2.018) *Advances in Financial Machine Learning*. Ed. Wiley.

Patterson, J. y Gibson, A. (2.017) *Deep Learning: A Practitioner's Approach*. Ed. O' Reilly Media.

W. Bao, J.Yue y Y.Rao (2017*) A deep learning framework for financial time series using stacked autoencoders and longshort term memory*. PLoS ONE, vol. 12, no. 7,2017.

Andrew W Lo y A Craig MacKinlay (1988) *Stock Market prices do not follow random walks: Evidence form a simple specification test*. En: Review of financial studies 1.1, pp 41-66.

Eugene F, Fama (1995) *Random walks in stock market prices*. En: Financial analysts journal 51.1, pp. 75-80.

Olden, M. (2016) Predicting Stocks with Machine Learning (Trabajo fin de master). Recuperado de: https://www.duo.uio.no/handle/10852/51275

Sidehabi, S. y Sofyan T. (2016) *Statistical and Machine Learning approach in forex prediction based on empirical data*. Computational Intelligence and Cybernetics (CYBERNETICSCOM), 2016 International Conference on IEEE,2016.

Petropoulus, A. et al. (2017) *A stacked generalization system for automated FOREX portfolio trading*. En: Expert Systems with Applications 90, pp 290-302.

Di Persio L. y Honchar O. (2016) *Artificial neural network approach to the forecast of stock market price movements*. International Journal of Economics and Managements Systems, 1, pp 158-162.

Abreu, G. et al. (2018) *Currency Exchange prediction using machine learning, genetics algorithms and technical analysis*. https://arxiv.org/abs/1805.11232

Direcciones y páginas web consultadas:

https://es.wikipedia.org/wiki/Mercado_de_divisas

https://es.wikipedia.org/wiki/Aprendizaje_autom%C3%A1tico

https://en.wikipedia.org/wiki/Feedforward_neural_network

https://medium.com/soldai/tipos-de-aprendizaje-autom%C3%A1tico-6413e3c615e2

https://www.stateof.ai/

https://www.mql5.com/en/docs/indicators

https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a

https://en.wikipedia.org/wiki/Gradient_descent

https://en.wikipedia.org/wiki/Activation_function

https://deeplearning4j.org/

https://keras.io/