PREDICCIÓN DE SERIES FINANCIERAS CON REDES NEURONALES RECURRENTES

EDWIN JAHIR RUEDA ROJAS

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2018

PREDICCIÓN DE SERIES FINANCIERAS CON REDES NEURONALES RECURRENTES

EDWIN JAHIR RUEDA ROJAS

TRABAJO DE GRADO PARA OPTAR AL TÍTULO DE INGENIERO DE SISTEMAS E INFORMÁTICA

DIRECTOR
FABIO MARTÍNEZ CARRILLO, Ph.D
Profesor EISI

CODIRECTOR
RAÚL RAMOS POLLÁN, Ph.D
Profesor UdeA

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2018

CONTENIDO

LISTA DE TABLAS

(Se enuncian de forma numérica, con el título y la respectiva página)

LISTA DE FIGURAS

GLOSARIO

Ask: Precio al que el mercado ofrece el par de divisas, en un instante de tiempo t, el precio del *Ask,* siempre es mayor al precio del *Bid.*

Bid: Precio al que el mercado compra un par de divisas.

Brokers: Intermediarios entre la persona natural y el mercado *forex*, usualmente brindan programas para orientar en dicho mercado para que el usuario invierta y así ellos ganar una comisión por transacción efectuada.

Divisas: Par de monedas extranjeras, el cual una de ellas está en referencia con la otra, el par más común es EUR/USD, donde se mira el precio del euro en dólares.

Forex: Del inglés *foreign exchange,* es el mercado en el cual se opera con el cambio de divisas, ya sea comprando o vendiendo.

Series de tiempo: Conjunto de datos los cuales están definidos en un tiempo específico.

Spread: Diferencia entre el precio del *Ask* y el *Bid*. Un spread bajo y regular en el tiempo hace que el mercado sea más estable.

RESUMEN

Título: Predicción de series financieras con redes neuronales recurrentes[1]

Autor: Edwin Jahir Rueda Rojas[2]

Palabras Clave: *Machine learning*, *Deep Learning,* analítica de datos, redes neuronales recurrentes, series de tiempo, *bid*, *ask*, *spread*, LSTM, *forex,* divisas, *trading, brokers.*

DESCRIPCIÓN:

[1] Trabajo de Grado

[2] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: Fabio Martínez Carrillo, Co-Director: Raúl Ramos Pollán

ABSTRACT

Title: Time series prediction with recurrent neural networks

Author: Edwin Jahir Rueda Rojas

Keywords: Machine Learning, Deep Learning, data analytics, recurrent neural networks, time series, bid, ask, spread, LSTM, forex, foreing exchange, trading, brokers.

INTRODUCCIÓN

Las series de tiempo abarcan un sin número de áreas, tales como las comunicaciones, el transporte, la salud y las finanzas [1]. Las redes neuronales recurrentes (RNN) son un subconjunto de redes neuronales inspiradas en el funcionamiento de una neurona humana, ya que estas bajo su configuración basada en unidades LSTM (*Long Short Term Memory*) son capaces de recordar características por cierto periodo de tiempo, es por esto que se dice que son capaces de asemejar el funcionamiento neuronal. Estas unidades LSTM han sido utilizadas en diferentes ámbitos tales como la clasificación del espectro de radio solar [2], la búsqueda de contraseñas [3], el aprendizaje de modelos fisiológicos para el comportamiento de la glucosa en la sangre [4], y la predicción en el mercado de valores [5], siendo este último el de mayor importancia en los últimos años.

En el estado del arte se han propuesto varios planteamientos con métodos de *Machine Learning* y *Deep Learning,* en los cuales se destaca la utilización de máquinas de soporte vectorial(SVM) [6] y las redes neuronales recurrentes(RNN) [7], teniendo estas técnicas un aporte significativo al mercado de las divisas. Stock Neural es una compañía la cual desarrolló una aplicación basada en RNN bastante robusta, tomando los datos financieros de Quandl, entrenaron una red neuronal recurrente con datos históricos de 10 años para dar una predicción de los próximos 5 días de *trading*, utilizando unidades LSTM hicieron que esta fuera lo suficientemente robusta para poder tener un acierto del 80% en sus predicción, generando hasta un retorno anual del 33% [7]. También se propuso un estudio basado en SVM, en el cual se destaca un acierto en la predicción de hasta un 60%, arrojando como conclusión de que una arquitectura de red neuronal robusta podría arrojar un mejor resultado [6].

Yuqing Dai e Yuning Zhang [8] propusieron un modelo de predicción basado en SVM con una estrategia de *trading* la cual determinaba si comprar o vender acciones o divisas dependiendo del PNL que arrojara el modelo. En un principio los resultados obtenidos fueron del 55,2% de precisión, probando con 3 meses de datos, cerca de 1470 datos por día y con un 70% para entrenar el modelo y el otro 30% para probarlo. Una última prueba la realizaron con una ventana de 44 días y 16 características, la cual arrojo un porcentaje de acierto del 79,3%.

La principal contribución de este trabajo de grado es plantear una arquitectura de red neuronal recurrente con unidades LSTM y plantear una estrategia de *trading* para operar en el mercado *forex*. Para poder llegar a este objetivo primero se obtendrán datos de fuentes variables, tales como TrueFX y Quandl para poder hacer un pre-procesado de estos datos, teniendo en cuenta el tiempo de muestreo, el tamaño de ventana a tomar, *OHLC* (*Open, High, Low, Close)* y ciertas características que sean relevantes, así como añadir otras señales financieras, asiendo así multiseñales de entrada para así utilizar modelos clásicos de *machine learning* para establecer una línea base de predicción con respecto a una señal de referencia y a las métricas tenidas en cuenta para realizar dicha predicción, una de estas métricas y la más importante es el PNL (*Proffit and Loss)*, ya que es la métrica que determina cuánto dinero puedo ganar o perder dependiendo de la precisión de dicho modelo. Uno de los ítems a tener en cuenta es que el porcentaje de precisión del modelo nos da un margen global de la robustez de dicho modelo, pero puede que el PNL no sea el esperado debido al margen de ganancia de cada predicción, si el modelo elaborado acierta en los casos en los cuales se gana o se pierde poco, pero falla en los cuales las pérdidas son grandes, aunque estos sean un porcentaje pequeño de falsos positivos, igual podría generar una pérdida de dinero.

Capítulo 1

1 PLANTEAMIENTO Y JUSTIFICACIÓN DEL PROBLEMA

La predicción del comportamiento de las series financieras como lo son las acciones en el mercado de valores y el del par de divisas (*forex)*, siempre han sido un reto complejo debido a la variabilidad de sus precios en el tiempo, siendo así que estos pueden variar entre el rango de los milisegundos, esto ha impulsado a una gran variedad de personas a tratar de resolver este problema para así generar un beneficio lucrativo.

Las técnicas de *Machine Learning* se han visto expuestas para tratar de resolver este problema, siendo las máquinas de soporte vectorial (SVM) el método más utilizado, más, sin embargo, en la actualidad se han venido aplicando redes neuronales recurrentes para resolver dicho problema, siendo estas las más acertadas en cuanto a la predicción, haciendo que ciertas agencias intermediarias o *brokers* implementen estas tecnologías para facilitar la interacción entre el cliente y el mercado financiero.

Capítulo 2

2 OBJETIVOS

2.1 OBJETIVO GENERAL

Diseñar y evaluar redes neuronales recurrentes para la predicción de señales financieras basadas en múltiples señales.

2.2 OBJETIVOS ESPECÍFICOS

- Adquirir datos financieros de diferentes fuentes.
- Desarrollar un pre-procesado de los datos multiseñal obtenidos anteriormente.
- Establecer métricas de predicción y tiempo de ejecución para evaluar el rendimiento del objetivo.
- Establecer una línea base de desempeño predictivo con métodos clásicos de *machine learning*
- Realizar una exploración de arquitecturas RNNs y configuraciones multiseñal.
- Evaluar el rendimiento de los modelos predictivos y proponer estrategias de trading.

Capítulo 3

3 MÉTODO PROPUESTO

En esta investigación se plantea un pre-procesado de las señales financieras, el cual consiste en analizar las señales que se tienen como entrada para hacerles una limpieza ya que estas vienen con cierto ruido generado bien sea por la demora causada por los datos al viajar al servidor en que son almacenados o por otras razones. Posteriormente se definen ciertos intervalos de tiempo para muestrear la señal y así obtener el intervalo de tiempo más apropiado con el cuál operar, ya que se necesita un intervalo de tiempo razonable para poder generar un incremento o decremento considerable en la señal como para generar una ganancia.

En el caso de múltiples señales, se considerará juntar señales financieras con cierto grado de similitud, teniendo en cuenta el principio de interdependencia de dichas señales [9], a su vez el pre-procesado consistirá básicamente en el mismo efectuado en las señales simples, solo que para las múltiples señales se tendrán en cuenta factores como el indexado del tiempo, ya que los datos en ambas señales deben corresponder al mismo instante de tiempo para poder juntarlos y así crear la multiseñal.

Finalmente se añadirán características específicas de cada señal, tales como el *OHLC* y la media del último intervalo de muestreo, dado esto, se tendrán en cuenta ciertos tamaños de ventana para las señales, para ver cómo estás se comportan al momento de aplicar los modelos de *machine learning* y las *RNNs*, para así tener un estimado de que tan robustos son dichos modelos en base a una estrategia de *trading* y las métricas de rendimiento definidas con anterioridad.

3.1 PRE-PROCESADO

Uno de los pasos principales de este proyecto, es el de hacer un pre-procesado de los datos el cual sea lo bastante característico para poder partir de un buen supuesto. Dado que los datos de señales referentes al mercado *forex,* o sea al par de divisas están dados en el orden de los milisegundos, esto hace que sea más complejo elaborar un modelo predictivo el cual tenga un acierto bueno para la metodología de *trading* propuesta en este proyecto.

3.1.1 MUESTREO DE LOS DATOS

Un muestro de los datos como primer paso es lo fundamental, ya que estos datos vienen en el orden de los milisegundos pero sin presentar un salto fijo de tiempo t, por ende se nos hace necesario sub muestrearlos en un orden t más apropiado para los posteriores cálculos. Para ello se plantean varios tamaños de muestreo t, siendo el muestreo de $t = 5\ [min]$ el más apropiado para ello debido a que con este intervalo t ya se hace más factible generar una ganancia con respecto a la compra y venta de divisas en el mercado *forex.* Teniendo así la señal original $S[t]$, de esta se obtiene una señal resultante $X[t]$, siendo t una muestra con periodicidad $t = 5\ [min]$ de la señal original, teniendo en cuenta que si la señal original no tiene una muestra en el tiempo $S[t]$, se toma la muestra inmediatamente anterior representada por $S[t-1]$, para que así la señal resultante no presente huecos. En la figura 1 se muestra como el muestreo en t minutos de los datos hace que la dimensionalidad de estos se reduzca significativamente permitiendo reducir un poco la volatilidad de estos y haciendo más factible el aprendizaje de los modelos. Cabe resaltar que este mercado opera de lunes a viernes.

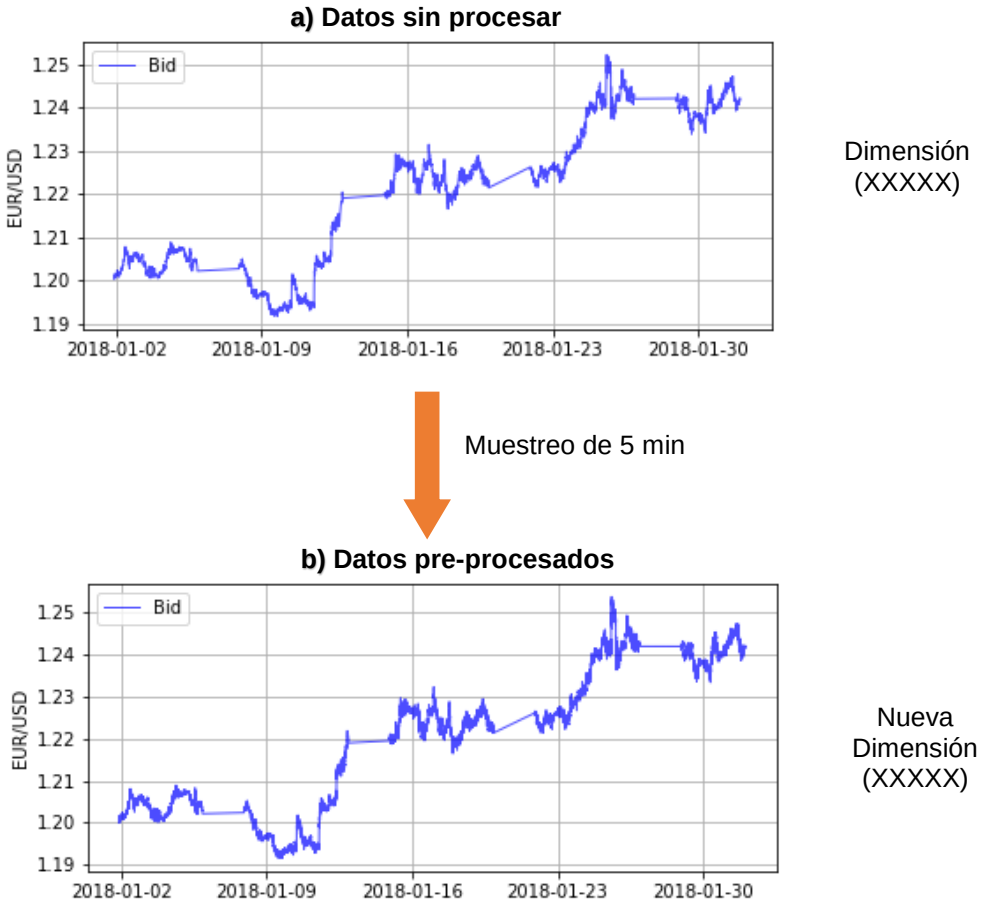

Figura 1. Representación de la reducción de los datos financieros.

3.1.2 TAMAÑO DE VENTANA

Un tamaño de ventana es un factor primordial para construir el conjunto de datos el cual constituirá la entrada al modelo predictivo, el estado del arte plantea diversas variantes a tener en cuenta para calcular dicho tamaño de ventana [10], en este trabajo de investigación se calcula el tamaño de la ventana de forma experimental, siendo el tamaño de ventana resultante el que mejor se comporta con base a los resultados obtenidos.

Así mismo para la señal pre-procesada $X[t]$, se tiene que para un tamaño de ventana n, la ventana resultante i está definida así:

$$X_i[t] = X_{t-1}, X_{t-2}, X_{t-3} \dots X_{t-n}$$

En la figura 2 se expresa con mayor claridad cómo funciona el tamaño de ventana para obtener el conjunto de datos y que dimensión toma este conjunto.

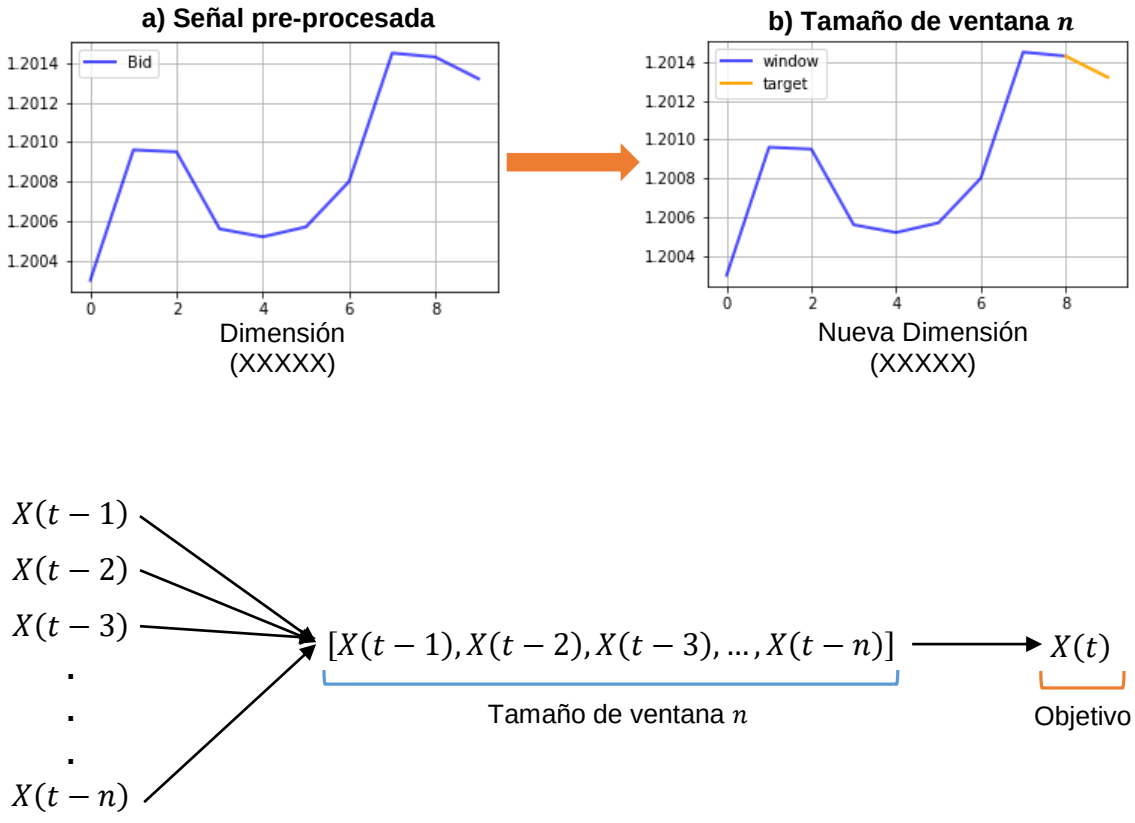

Figura 2. Funcionamiento del tamaño de ventana, si el tamaño de la señal es m, el tamaño de la señal resultante será $m - n$, donde n es el tamaño de ventana.

3.1.3 EXTRACCIÓN DE CARACTERISTICAS

La extracción de características es una pieza fundamental a la hora de enriquecer el contenido de nuestros datos, teniendo en cuenta que solo tenemos la señal del valor del *bid* con el tamaño de ventana n para entrenar nuestros modelos, hay que tratar de enriquecer la señal lo que más se pueda, para ello se recurre a la estadística descriptiva para calcular la media de las ventanas, siendo $X_{ij}[t]$ la señal pre-procesada en un muestreo de frecuencia t, de filas i, donde cada fila representa una ventana y cada j una muestra de cada ventana, la media de cada ventana i viene dada por:

$$\bar{X}_i = \sum_{j=0}^{n-1} \frac{X_{ij}}{n}$$

El *OHLC* es otra de las principales características la cual se puede extraer de la señal original $S[t]$ para ser añadida a nuestra señal pre-procesada $X[t]$, este consiste en calcular los valores *open, high, low and close* para cierto intervalo t, que en este caso es igual a 5 minutos, siendo así que para los último 5 minutos de la señal original $S[t]$, se extrae el valor más alto, el más bajo, el primero que aparece, y el último. La figura 3 nos ilustra un poco más como se extraen estas características:

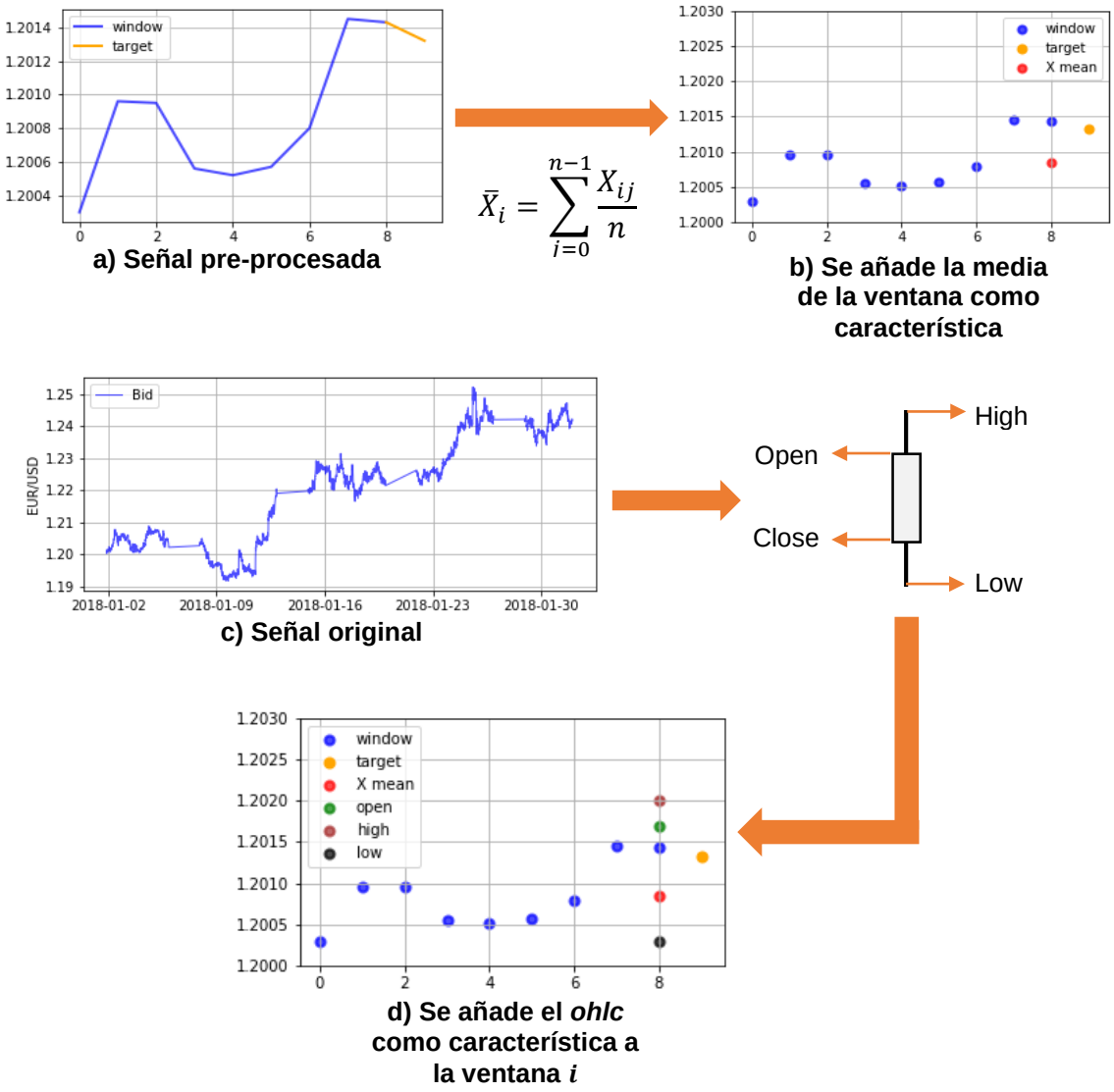

Figura 3. Extracción de características para enriquecer la ventana i.

3.2 ESTRATEGIA DE TRADING

Proponer una estrategia de trading es lo más importante después de pre-procesar los datos, ya que en base a dicha estrategia se podrá calcular el rendimiento de los modelos de *machine learning* y *deep learning* planteados. Se propone una estrategia segura, en la cual operaremos en el mercado solo si la predicción del

modelo así lo indica, estableciendo así, valores enteros para cada actuar. La figura 4 nos ilustra con más claridad la estrategia propuesta.

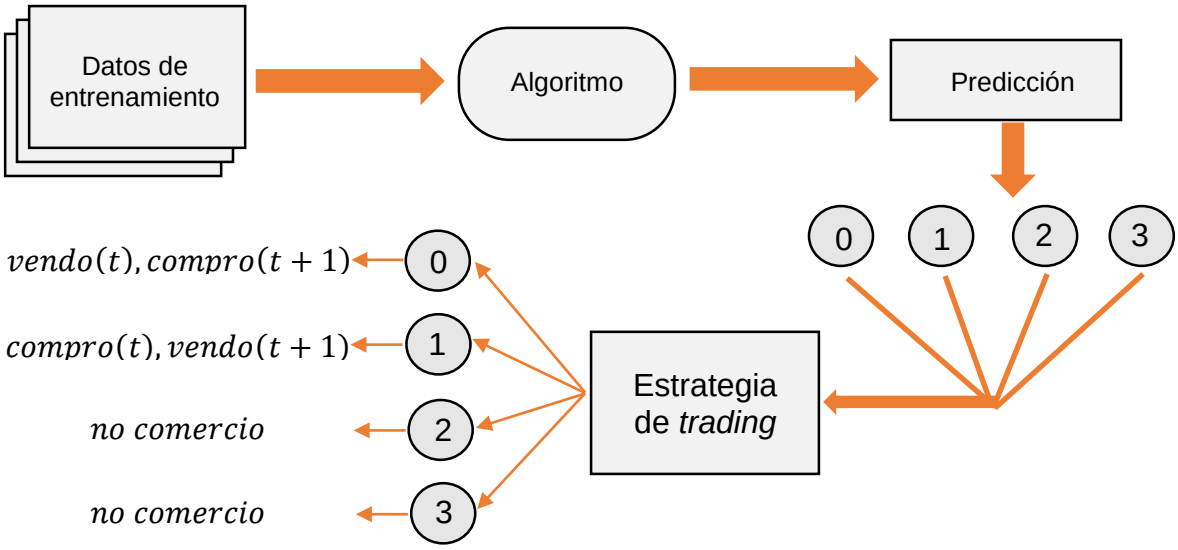

Figura 4. Planteamiento de la estrategia de *trading* según la predicción del modelo.

3.3 MACHINE LEARNING

Para definir el significado de *machine learning* o Máquinas de Aprendizaje, haré referencia a la definición del libro de Tom M. Mitchell el cual desde su aporte a este campo expresa que el *machine learning* se refiere a la cuestión de cómo construir programas computacionales los cuales mejoran con la experiencia. Se dice que un programa de computadora aprende de una experiencia **E** con respecto a unas clases de tareas **T** y con medida de desempeño **P**, si su desempeño en tareas **T**, medido por **P**, mejora con la experiencia **E** [11].

Cabe resaltar que el Aprendizaje de máquina cuenta con dos ramas, una de ellas es el aprendizaje supervisado el cual tiene cabida en este trabajo de investigación y consiste en proveer a la máquina con un conjunto de datos de entrada, los cuales se encuentran etiquetados, es decir que cada conjunto de características es

distinguible mediante una etiqueta que los clasifica. Por otro lado, el aprendizaje no supervisado es en el cual cada conjunto de características es indistinguible ya que no cuenta con etiquetas, así su función involucra también clasificar el conjunto de características suministradas. La figura 5 representa el esquema que realizan las técnicas de aprendizaje supervisado.

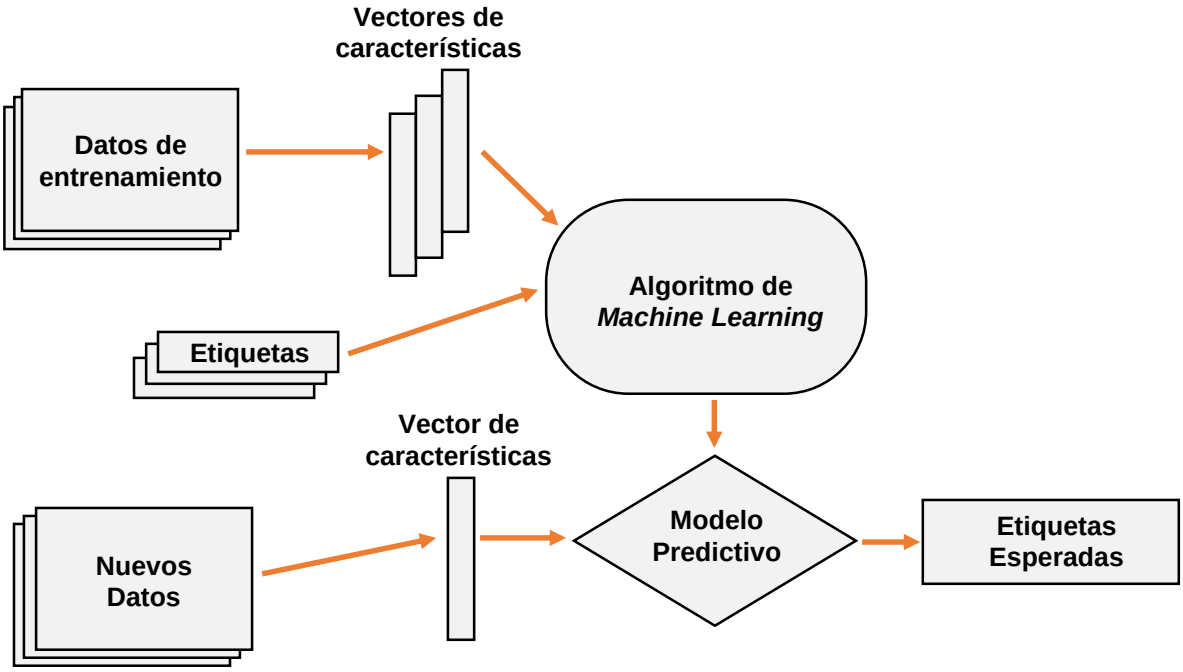

Figura 5. Esquema del proceso de clasificación en aprendizaje supervisado.

3.2.1 CLASIFICACIÓN EN APRENDIZAJE SUPERVISADO

La clasificación es una de las partes del aprendizaje supervisado, como también lo es la regresión, la cual no tiene cabida en este proyecto de investigación. Se habla de clasificación cuando tenemos un conjunto de datos etiquetados con números enteros, o caracteres, por ejemplo, el “día” o la “noche”. Estos problemas de clasificación se abordan mediante algoritmos de aprendizaje supervisado, los cuales reciben el nombre de clasificadores, siendo estos los que generan un modelo predictivo.

Para resolver un problema de clasificación, este se tiene que ver como un problema de optimización matemática en el cual el modelo consta de un conjunto de parámetros

$$\theta = \theta_0 + \theta_1 + \theta_2 + \cdots + \theta_n$$

Los cuales interactúan con el conjunto de datos de entrada X y predicen la probabilidad de que este pertenezca a una clase.

$$h_\theta(x) = etiqueta, probabilidad$$

Dado que como plantea M. Mitchell el objetivo del modelo es mejorar cada vez más la predicción, se define una función de coste la cual se optimiza para obtener los θ_i que generan menor costo, o sea los cuales me generan una mejor predicción.

$$\underset{\theta}{argmin} J(\theta)$$

3.2.2 ALGORITMOS DE CLASIFICACIÓN

Con el fin de plantear una línea base para este proyecto de investigación, se seleccionan ciertos algoritmos de *machine learning* para las tareas de clasificación, los cuales se explicarán a continuación mostrando su lógica de funcionamiento la cual se probará en la sección de evaluación y resultados.

3.2.2.1 NAIVE BAYES

Naive Bayes es un método de clasificación en aprendizaje supervisado el cual se basa en el Teorema de Bayes, con una suposición "ingenua" (*naive)* sobre la independencia de los datos. En este trabajo de investigación se hace uso del algoritmo *Gaussian Naive Bayes* el cual asume que la probabilidad de las características es Gaussiana [12], la cual es obtenida así:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

3.2.2.2 KENIGHBORS

La clasificación basada en vecinos (neighbors), es un tipo de aprendizaje basado en instancias, o no generalizado, el cual no intenta construir un modelo interno general, sino que almacena instancias de los datos de entrenamiento [13]. La clasificación se calcula a partir de un voto de mayoría simple de los *K* vecinos más cercanos del punto, siendo la distancia Euclidiana la más comúnmente utilizada [14]:

$$d(x, x') = \sqrt{(x_1 - x_1')^2 + (x_2 - x_2')^2 + \cdots + (x_n - x_n')^2}$$

3.2.2.3 ÁRBOLES DE DECISIÓN

Un árbol de decisión es un grafo dirigido y sin bucles, cuyo objetivo es predecir el valor esperado por una serie de reglas de decisión simples [15]. A continuación se presenta el planteamiento matemático dado vectores de entrenamiento $x_i \in R^n, i = 1, \ldots, l$ y un vector de etiquetas $y \in R^l$, el árbol de decisión particiona recursivamente el espacio de modo que las muestras con las mismas etiquetas se agrupen juntas.

Los datos de un nodo m son representados por Q. Para cada división $\theta = (j, tm)$ que consiste de una característica j y un umbral tm, los subconjuntos de datos respectivos a los nodos se identifican con:

$$Q_{left}(\theta) = (x, y)|x_j \leq tm$$
$$Q_{right}(\theta) = Q \backslash Q_{left}(\theta)$$

Donde la impureza de m es calculada usando la función de impureza $H()$, cuya elección depende de la tarea que se esté resolviendo, en este caso, de clasificación.

$$G(Q,\theta) = \frac{n_{left}}{N_m} H\left(Q_{left}(\theta)\right) + \frac{n_{rigth}}{N_m} H\left(Q_{rigth}(\theta)\right)$$

Luego se trata de minimizar la impureza:

$$\theta^* = argmin_\theta G(Q,\theta)$$

Este proceso se aplica para cada subconjunto $Q_{left}(\theta^*)$ y $Q_{right}(\theta^*)$ hasta que se alcance la profundidad máxima para el árbol, siendo $N_m < min_{samples}$ ó $N_m = 1$ [16].

3.2.2.4 BOSQUE ALEATORIO

Para hablar de un bosque aleatorio (*Random Forest)*, tenemos que hablar de los métodos conjuntos (*Ensemble methods)*, donde el objetivo es combinar predicciones de varios estimadores base construidos, y con un algoritmo de aprendizaje mejorar la robustez sobre un único estimador [17]. En el *Random Forest* cada árbol en el conjunto es construido desde una muestra extraída con reemplazo desde el conjunto de datos de entrenamiento. En adición, cuando se divide un nodo durante la construcción del árbol, la división que se elige no es la mejor división entre todas las características. En cambio, la división que se selecciona es la mejor división entre un subconjunto aleatorio de las características. Como resultado de esta aleatoriedad, el sesgo del bosque generalmente aumenta ligeramente (con respecto al sesgo de un solo árbol no aleatorio) pero, debido al promedio, su varianza también disminuye, generalmente más que compensando el aumento en el sesgo, lo que arroja un modelo global mejor [18].

3.3 DEEP LEARNING

El *deep learning* es un sub campo del *machine learning* el cual se basa en algoritmos los cuales son inspirados en la estructura y función del cerebro, por ende, se les conoce como redes neuronales artificiales. Estos modelos imitan estas características arquitecturales del sistema nervioso, permitiendo que dentro del sistema global haya redes de unidades de proceso que se especialicen en la detección de determinadas características, de ahí que se presenten dos grandes arquitecturas de redes neuronales, como lo son las convolucionales, las cuales abarcan el procesamiento de imágenes y las redes neuronales recurrentes las cuales tienen cabida en este trabajo de investigación ya que permiten recordar ciertos parámetros en un tiempo t para ser utilizados en un tiempo t+1 [19].

3.3.1 REDES NEURONALES RECURRENTES

Las RNNs son un tipo de redes neuronales las cuales aumentan el rendimiento del algoritmo neuronal ya que combinan dos grandes propiedades [20]:

- Capas ocultas las cuales permiten almacenar gran información acerca del pasado de forma eficiente.
- Dinámicas no lineales que permiten actualizar sus capas ocultas en métodos complicados.

La figura 6 muestra la dinámica de una red neuronal a través del tiempo.

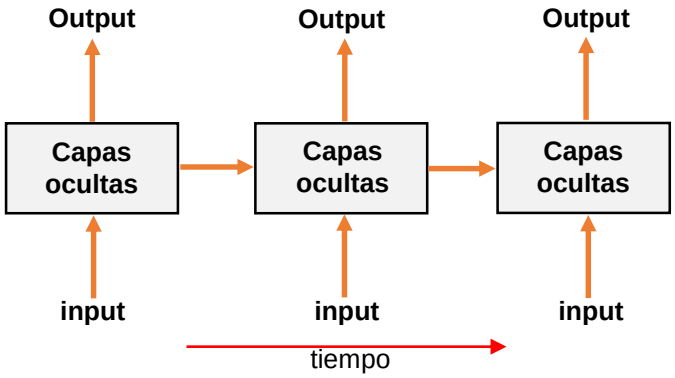

Figura 6. Funcionamiento de una red neuronal con respecto al tiempo.

3.3.1.1 LONG SHORT TERM MEMORY (LSTM)

Las unidades LSTM desarrolladas por Hochreiter y Schmindhuber (1997) introducen una nueva estructura llamada *celda de memoria* (figura 7). esta celda memoria posee cuatro componentes elementales, un *input gate* o puerta de entrada, una neurona con una conexión recurrente así mismo, una *forget gate* o puerta de olvido y un *output gate* o puerta de salida. La conexión recurrente tiene un peso 1 y asegura que el estado de dicha celda permanece constante de un paso a otro a menos de que interfiera algo externo. El *input gate* o puerta de entrada puede permitir que la señal entrante altere el estado de la celda de memoria o la bloquee. Por otro lado, el *output gate* o puerta de salida puede permitir que el estado de la celda de memoria tenga un efecto sobre otras neuronas o la evite. Finalmente, la *forget gate o* puerta de olvido puede modular la conexión auto-recurrente de la celda de memoria permitiendo que la célula recuerde u olvide su estado anterior, según sea necesario [21].

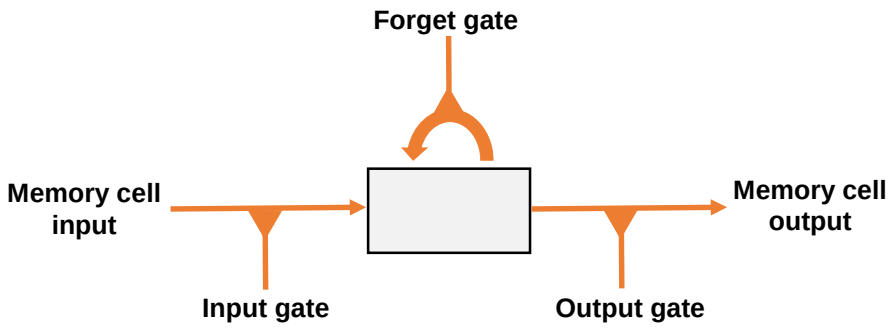

Figura 7. Unidad LSTM desarrollada por Hochreiter y Schmindhuber (1997).

Las siguientes ecuaciones describen como una capa de celda de memorias se actualiza en cada paso de tiempo t, en estas ecuaciones:

- x_t es la entrada a la capa de celda de memoria en cada tiempo t.
- $W_i, W_f, W_c, W_o, U_i, U_f, U_c, U_o \; y \; V_o$ son matrices de peso.
- $b_i, b_f, b_c \; y \; b_o$ son vectores bias.

Primero, se computan los valores para i_t, la puerta de entrada, y $\tilde{C}_t$ el cuál es el valor candidato para el estado de la celda de memoria en el tiempo t:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$
$$\tilde{C}_t = tanh\left(W_f x_t + U_c h_{t-1} + b_f\right)$$

Segundo, computamos el valor de f_t, el cual es la activación de las *forget gate* o puertas de olvido en el instante t:

$$f_t = \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right)$$

Dado el valor de activación de la puerta de entrada i_t, la activación de la puerta de olvido f_t y el valor candidato $\tilde{C}_t$, el tercer paso es computar C_t, el cual es el nuevo estado de la celda de memoria en el instante t:

$$C_t = i_t * \tilde{C}_t + f_t * C_{t-1}$$

Con el nuevo estado de la celda de memoria, el cuarto paso es calcular el valor de la puerta de salida y seguidamente, el valor de su salida:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o C_t + b_o)$$
$$h_t = o_t * tanh(C_t)$$

3.4 VALIDACIÓN DE LOS MODELOS

Para poder medir el desempeño de los modelos planteados en este trabajo de investigación primero hay que ajustar los criterios de validación. La validación del modelo consiste en cómo se definen los datos de entrenamiento y de prueba de dicho modelo, para los modelos de *machine learning* de diseña una validación por días como se aprecia en la figura 8, siendo así que el modelo se entrena durante cuatro días y este opera durante el día siguiente a su entreno, o sea en su quinto día. De los días en los cuales el modelo opera se sacarán métricas de clasificación las cuales se dirán más adelante en este libro. En el caso de las arquitecturas planteadas con *deep learning* se tomarán una cierta cantidad de datos para el entrenamiento de una red neuronal recurrente para posteriormente dar una predicción del mes siguiente a los datos de entrenamiento ya que al construir una arquitectura de red neuronal y entrenarla, esta permite establecer dos parámetros claves los cuales son:

- ***Batch*:** conjunto de N muestras las cuales se procesan en paralelo, al entrenar la red, cada *batch* o lote representa una actualización del modelo.
- ***Epoch*:** Es un corte arbitrario el cual define un paso por el conjunto de datos, usado para separar el entrenamiento en distintas fases, es útil para ver la evaluación del modelo cada vez que se actualiza el gradiente [22].

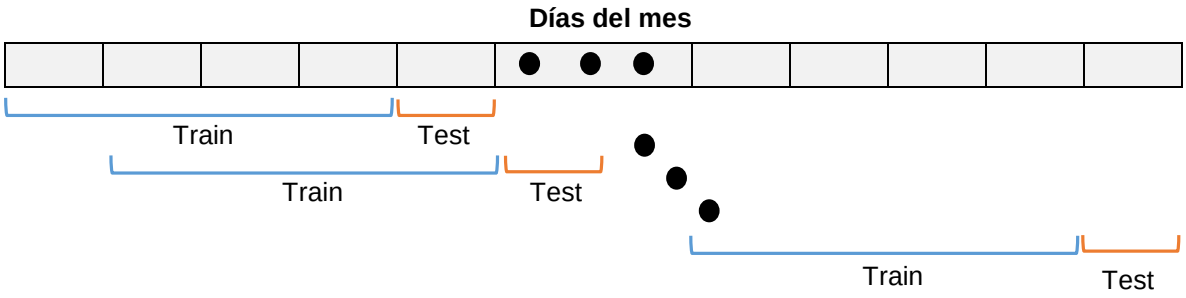

Figura 8. Representación de validación por días.

3.4.1 MÉTRICAS DE RENDIMIENTO

Las métricas de rendimiento nos dicen que tan robusto es nuestro modelo predictivo, para este trabajo de investigación se tendrán en cuenta varias métricas de predicción las cuales se mencionan a continuación.

3.4.1.1 PROFIT AND LOSS (PNL)

La ganancia o pérdida es una de las métricas más importantes del modelo, ya que esta nos dice cuánta ganancia o pérdida nos genera el modelo. El PNL está definido dependiendo de la predicción de un instante t+1 del modelo, en base a datos t y anteriores, este está definido según la predicción de la siguiente forma:

Predicción	Operación	PNL 0	PNL 1
0	venta/compra	$venta(t) - compra(t+1)$	-
1	compra/venta	-	$venta(t+1) - compra(t)$
2	-	-	-
3	-	-	-

3.4.1.2 METRICAS SECUNDARIAS

Como métricas de desempeño secundarias se tienen el promedio de compras y ventas que se realizan dadas las predicciones aplicadas a la estrategia de *trading* propuesta. También se obtiene el tamaño de la cadena de compras y ventas más larga, ya que esta nos indica cuánto dinero debemos que tener disponible para comerciar. Por último, se obtiene la precisión del modelo calculando los aciertos obtenidos de la siguiente manera:

$$precisión = \sum_{i=0}^{n-1} \frac{y_{predict} = y_{target}}{n}$$

Siendo $y_{predict}$ la predicción obtenida por el modelo acerca del vector de características entrante, y_{target} el objetivo y n el número de predicciones hechas.

Capítulo 4

4 EVALUACIÓN Y RESULTADOS

5 CONCLUSIONES

6 RECOMENDACIONES

REFERENCIAS BIBLIOGRÁFICAS

[1] Sharat C Prasad, Member, IEEE, and Piyush Prasad, *Deep Recurrent Neural Networks for Time Series Prediction.*

[2] Xuexin Yu, Long Xu, Lin Ma, Zhuo Chen, Yihnua Yan, *Solar Radio Spectrum Classification with LSTM*.

[3] Lingzhi Xu, Can Ge, Weidong Qiu, Zheng Huang, Jie Guo, Huijuan Lian, *Password Guessing Based on LSTM Recurrent Neural Networks.*

[4] Ben Athiwaratkun and Jack W. Stokes, *Malware Classification with LSTM and Gru Language Models and a Character-Level CNN.*

[5] David M. Q. Nelson, Adriano C. M. Pereira, Renato A. de Oliveira, *Stock Market's Price Movement Prediction With LSTM Neural Networks.*

[6] Lai, Ping-fu (Brian) – Wong Chung Hang, *Performance of Stock Market Prediction.*

[7] Stocksneural.net, *Provide forecasts of stocks prices using Deep Learning methods.*

[8] Yuqing Dai, Yuning Zhang, *Machine Learning in Stock Price Trend Forecasting.*

[9] Pami Dua, Divya Tuteja, *Interdependence of International Financial Market.*

[10] Rhee M. Kil, Seon Hee Park, and Seunghwan Kim, *Optimum window size for time series prediction.*

[11] Tom M. Mitchell, *Machine Learning.*

[12] Sklearn Documentation, *Gaussian Naive Bayes.*

[13] Sklearn Documentation, *Nearest Neighbors.*

[14] Kevin Zakka's Blog, *A Complete Guide to K-Nearest-Neighbors with Applications in Python and R.*

[15] Mayur Kulkarni, *Decision Trees for Classification: A Machine Learning Algorithm.*

[16] Sklearn Documentation, *Decision Trees, Mathematical Formulation.*

[17] William Koehrsen, *Random Forest Simple Explanation*

[18] Sklearn Documentation, *Ensamble Methods, Random Forest Classifier.*

[19] Raúl Arrabales, *Deep Learning: la aproximación a la percepción humana.*

[20] Geoffrey Hinton, *University of Toronto: Neural Networks for Machine Learning.*

[21] DeepLearning Documentation, *LSTM Networks for Sentiment Analysis.*

[22] Keras Documentation, *Frequently asked keras Questions.*