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

Universidad Industrial de Santander

UNIVERSIDAD INDUSTRIAL DE SANTANDER
SISTEMA DE TRABAJOS DE GRADO
ADMINISTRACIÓN DE TRABAJOS DE GRADO

Fecha Impresión:
06 junio 2018

Codigo:	19033	**Fecha Presentacion:** 15-dic-2017	
Titulo:			
Predicción de series financieras con redes neuronales recurrentes.			
Nota Proyecto:	4.5	**Fecha Registro Nota:** 06-jun-2018	
Estado:	APROBADO		
Tipo Trabajo:	INVESTIGACION		

Estudiantes

Código	Nombre	Programa Académico
2131432	RUEDA ROJAS EDWIN JAHIR	11-INGENIERIA DE SISTEMAS

Directores

Documento	Nombre	Clase	Firma
C-13929892	FABIO MARTINEZ CARRILLO	DIRECTOR	

Calificadores

Documento	Nombre	Firma
C-91175190	JORGE LUIS CHACON VELASCO	
C-13805233	FERNANDO RUIZ DIAZ	

ENTREGA DE TRABAJOS DE GRADO, TRABAJOS DE INVESTIGACION O TESIS Y AUTORIZACIÓN DE SU USO A FAVOR DE LA UIS

Yo, **Edwin Jahir Rueda Rojas**, mayor de edad, vecino de Bucaramanga, identificado con la Cédula de Ciudadanía No. 1098768953 de Bucaramanga, actuando en nombre propio, en mi calidad de autor del trabajo de grado, del trabajo de investigación, o de la tesis denominada(o):

Predicción de series financieras con redes neuronales recurrentes,

hago entrega del ejemplar respectivo y de sus anexos de ser el caso, en formato digital o electrónico (CD o DVD) y autorizo a LA UNIVERSIDAD INDUSTRIAL DE SANTANDER, para que en los términos establecidos en la Ley 23 de 1982, Ley 44 de 1993, decisión Andina 351 de 1993, Decreto 460 de 1995 y demás normas generales sobre la materia, utilice y use en todas sus formas, los derechos patrimoniales de reproducción, comunicación pública, transformación y distribución (alquiler, préstamo público e importación) que me corresponden como creador de la obra objeto del presente documento. PARÁGRAFO: La presente autorización se hace extensiva no sólo a las facultades y derechos de uso sobre la obra en formato o soporte material, sino también para formato virtual, electrónico, digital, óptico, uso en red, Internet, extranet, intranet, etc., y en general para cualquier formato conocido o por conocer.

EL AUTOR – ESTUDIANTE, manifiesta que la obra objeto de la presente autorización es original y la realizó sin violar o usurpar derechos de autor de terceros, por lo tanto la obra es de su exclusiva autoría y detenta la titularidad sobre la misma. PARÁGRAFO: En caso de presentarse cualquier reclamación o acción por parte de un tercero en cuanto a los derechos de autor sobre la obra en cuestión, EL AUTOR / ESTUDIANTE, asumirá toda la responsabilidad, y saldrá en defensa de los derechos aquí autorizados; para todos los efectos la Universidad actúa como un tercero de buena fe.

Para constancia se firma el presente documento en dos (02) ejemplares del mismo valor y tenor, en Bucaramanga, a los 9 días del mes de Junio de Dos Mil Dieciocho 2018.

EL AUTOR / ESTUDIANTE:

(Firma).....

Nombre Edwin Jahir Rueda Rojas

AGRADECIMIENTOS

El autor expresa su agradecimiento:

Al grupo de Cómputo Avanzado y a Gran Escala (CAGE) por brindar la infraestructura computacional necesaria para realizar los cálculos requeridos, y al profesor Raúl Ramos por ser el guía de este proyecto de investigación.

A mis compañeros de universidad por formar parte de mi formación como profesional y persona, más exactamente a mis colegas en el grupo CAGE y MACV.

A la escuela de Ingeniería de Sistemas e Informática y a sus docentes por brindarme la información necesaria para formarme como profesional.

A mis padres y hermanos por ser el complemento de mi vida, en especial a mi hermana por ser mi motivación.

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

GLOSARIO

Ask: Precio al que el mercado ofrece un 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.

OHLC: *Open, High, Low, Close*. Valor de apertura, pico más alto, pico más bajo y cierre, para un intervalo de tiempo dado en un par de divisas.

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:

Las series de tiempo tienen cabida en múltiples áreas, tales como las comunicaciones, la salud y las finanzas. Las RNN son un subconjunto de redes neuronales las cuales se inspiran en el funcionamiento neuronal humano, esto debido a que sus unidades LSTM tienen la capacidad de recordar características a lo largo del tiempo. En esta investigación se aborda la predicción en el mercado de divisas. Para ello hay que decir que el problema fundamental es la variación tan rápida que presentan dichos pares de divisas, siendo así difícil obtener una buena predicción, causando así, pérdidas de dinero.

Este trabajo de investigación usando datos de Quandl y TrueFX implementa un tamaño de ventana fijo y unas características adicionales las cuales tienden a aumentar la precisión de los modelos construidos, siendo el OHLC y las multiseñales unas de las características significantes, así mismo, se implementaron técnicas clásicas de *machine learning* en las cuales la mejor predicción vino dada por un modelo GaussianNB el cual arrojó una precisión del 26.12% la cual involucra una pérdida diaria en base a la estrategia de trading planteada de 0.0013 USD, equivalente a 3.70 COP, lo que nos indica una mejora respecto a plantear una aleatoriedad como estrategia de trading, lo que nos indica que los modelos tratan de aprender cosas de la señal pero no lo suficiente para producir resultados favorables, por ello se recurre a las arquitecturas de redes neuronales, las cuales arrojan un resultado más favorable, ya que con la misma estrategia de trading propuesta, estas interactúan más con el mercado y pierden menos, alrededor de 0.00005566 USD diarios, el equivalente a 0.15 COP, lo que nos permite intuir que a mayor complejidad de RNN y un mayor descriptor de la señal, se podrían producir mejores resultados.

[1] Trabajo de Grado

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

ABSTRACT

Title: Time series prediction with recurrent neural networks[1]
Author: Edwin Jahir Rueda Rojas[2]
Keywords: Machine Learning, Deep Learning, data analytics, recurrent neural networks, time series, bid, ask, spread, LSTM, forex, foreing exchange, trading, brokers.

DESCRIPTION:

Time series cover multiple areas, such as communications, health and finance. The RNNs are a subset of neural networks which are based on human neural functioning, because their LSMT units have the ability to remember characteristics over time. Being the LSTM units used for the search of passwords, the learning of physiological models for the behavior of the glucose in the blood and the prediction in the stock market, being this problem the one of work in the investigation. It must be said that the fundamental problem is the rapid variation that the stock market presents, specifically that of the currencies, making it difficult to obtain a good prediction, which generates losses of money.

This research work with data from Quandl and TrueFX implements a fixed window size and additional features to increase the probability of accuracy of the predictive model, being the OHLC and the multisignals significant characteristics, likewise, classical techniques of machine learning were implemented in which the best prediction was given by a GaussianNB model which gave a precision of 26.12% which involves a daily loss based on the strategy of trading raised of 0.00013 USD, equivalent to 3.70 COP, which indicates an improvement over if there was randomness as a trading strategy, this says that the models try to learn things from the signal but not enough to produce favorable results, for this reason, we use neural network architectures, which are more accurate, since by implementing the same proposed trading strategy, they interact more with the market and lose less, around 0.00005566 USD per day, the equivalent to 0.15 COP, which allows us to guess that the grater the complexity of the RNN and the greater the descriptor of the signal, the better results could be produced.

[1] Research Work.

[2] School of Physical-Mechanical Engineering. Department of Systems Engineering and Informatics. Advisor, Fabio Martínez Carrillo, PhD. Sub-advisor, Raúl Ramos Pollán, PhD.

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 los pares 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.

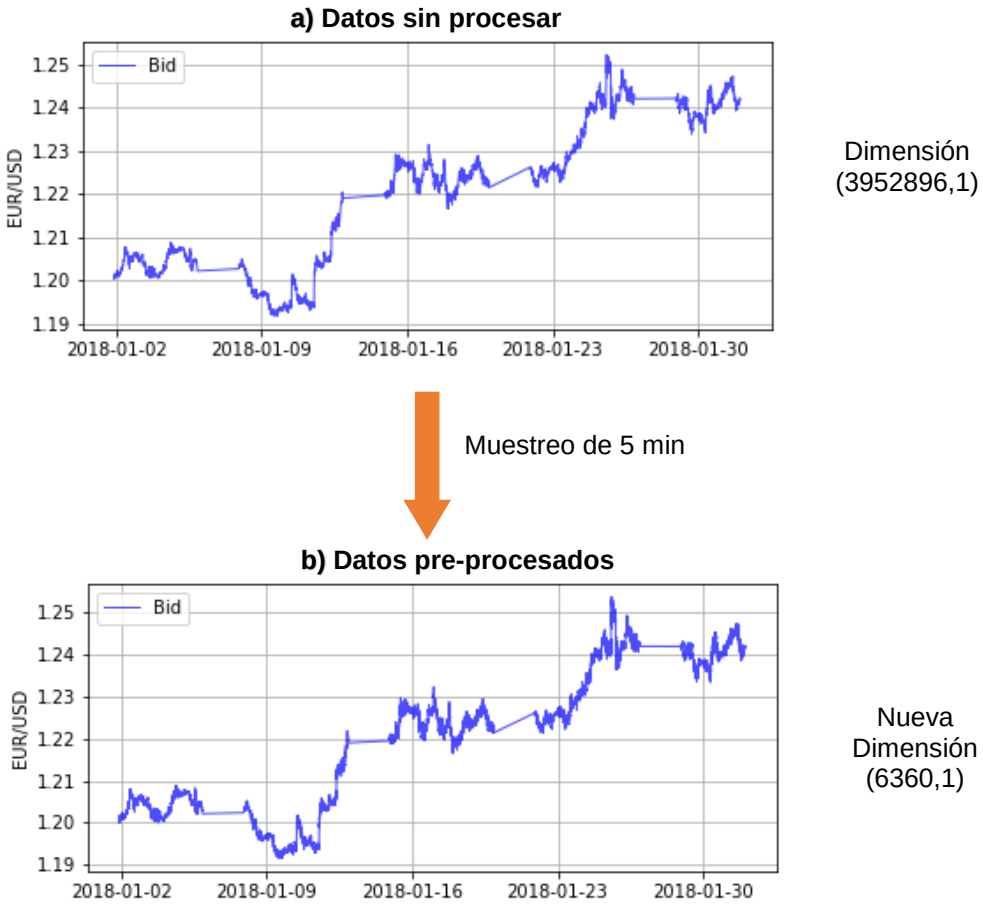
Finalmente se añadirán características específicas de cada señal, tales como el *OHLC* y la media del último intervalo de muestreo, dado esto, se tendrán en cuenta ciertos tamaños de ventana para las señales, para ver cómo estás se comportan al momento de aplicar los modelos de *machine learning* y las *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.



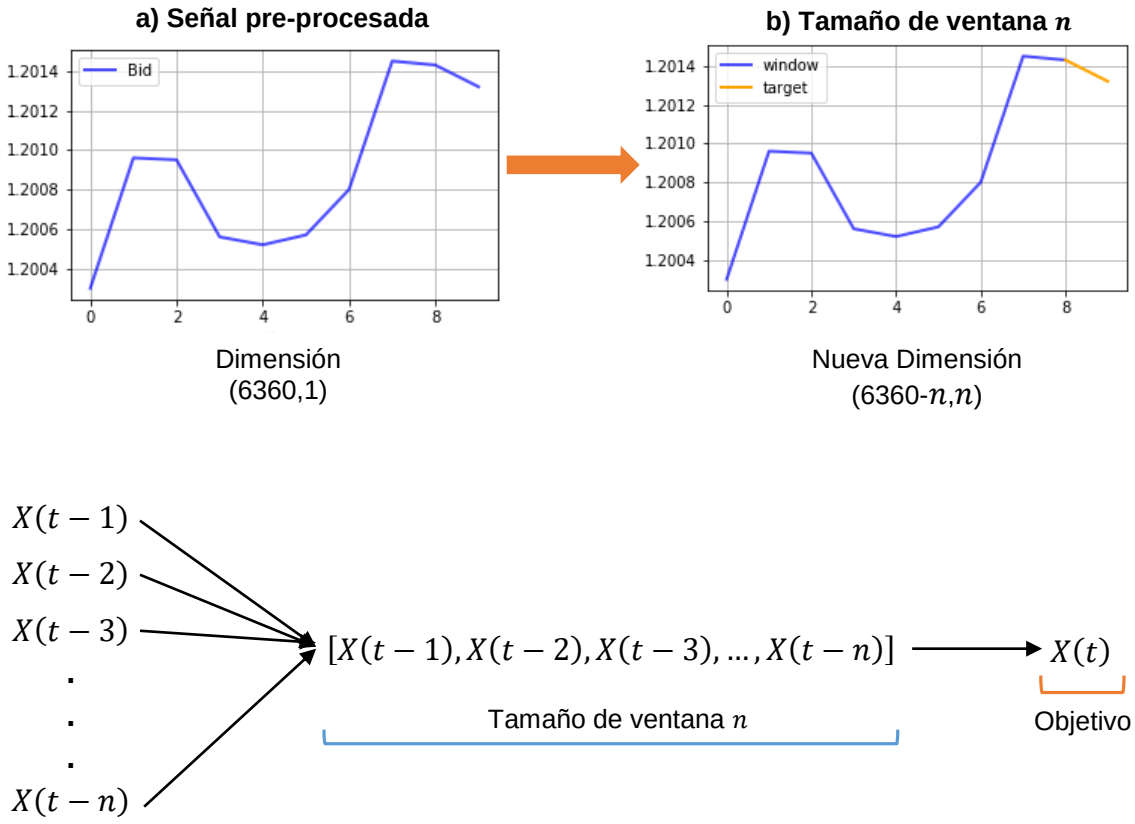
3.1.2 Tamaño de ventana. Un tamaño de ventana es un factor primordial para construir el conjunto de datos el cual constituirá la entrada al modelo predictivo, el estado del arte plantea diversas variantes a tener en cuenta para calcular dicho tamaño de ventana [10], en este trabajo de investigación se 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.

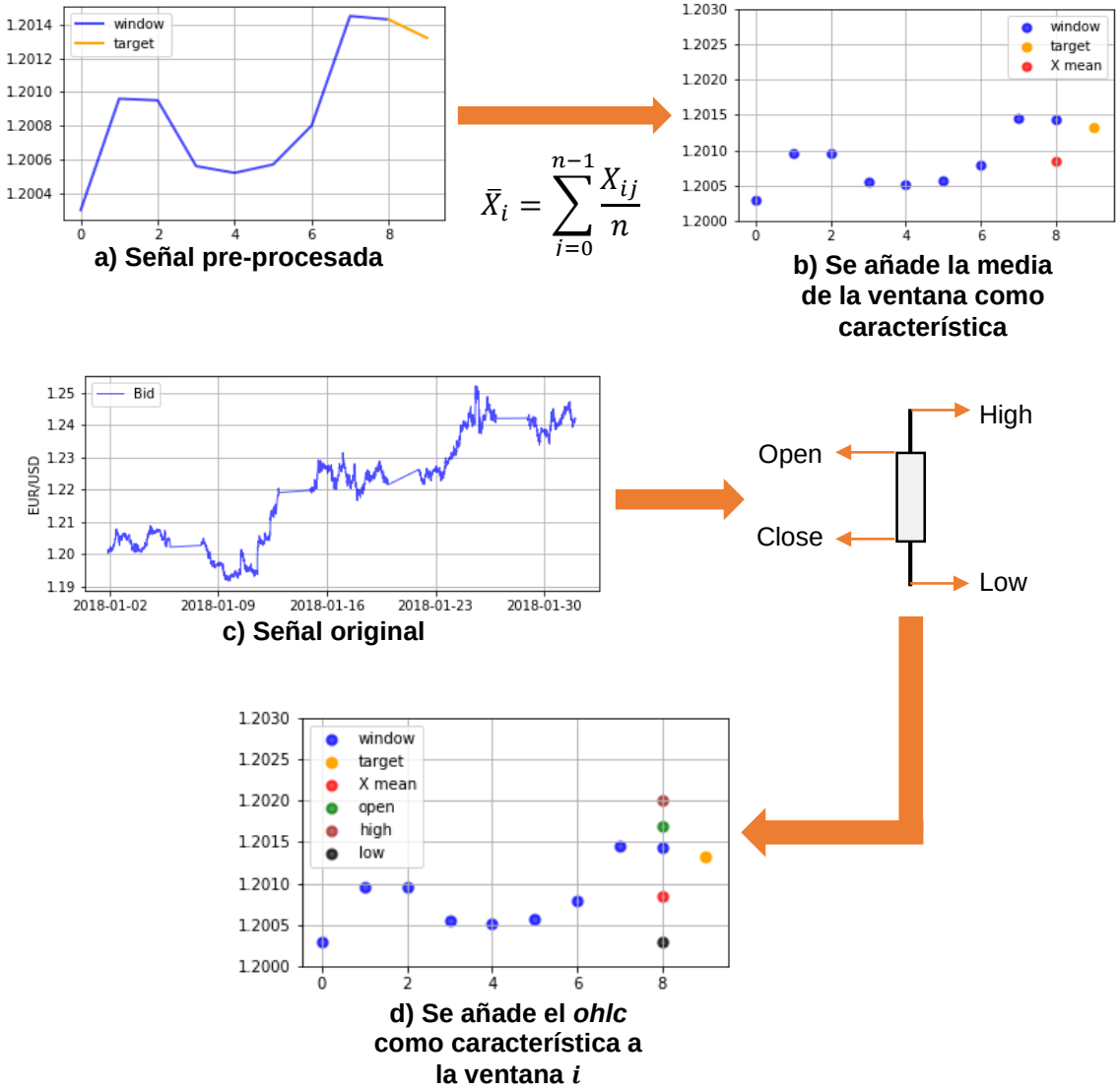


3.1.3 Extracción de características. La extracción de características es una pieza fundamental a la hora de enriquecer el contenido de nuestros datos, teniendo en cuenta que solo tenemos la señal del valor del *bid* con el tamaño de ventana n para entrenar nuestros modelos, hay que tratar de enriquecer 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.

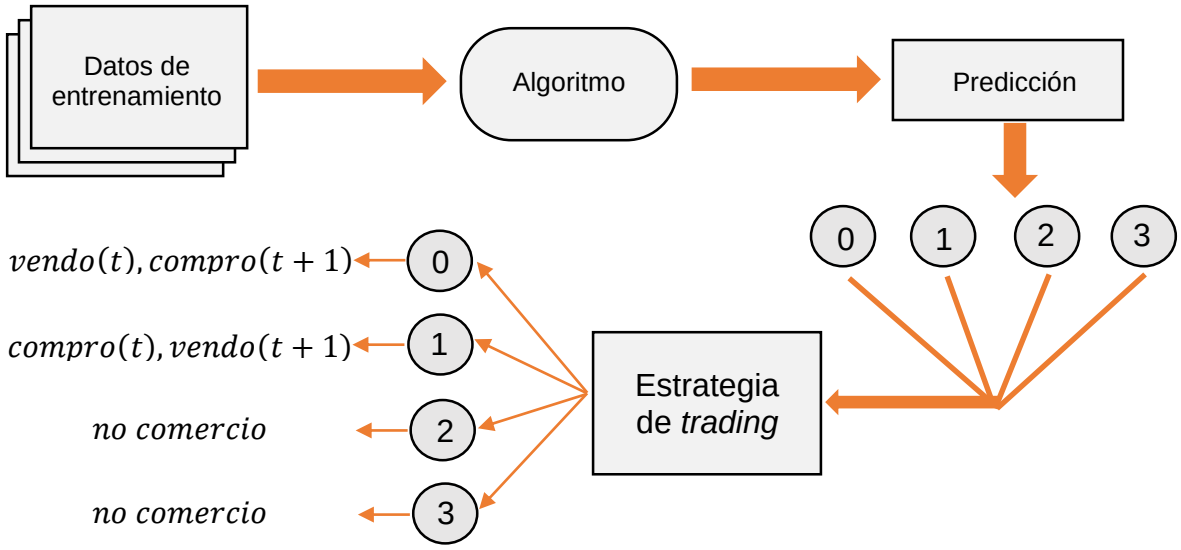


3.2 ESTRATEGIA DE TRADING

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.



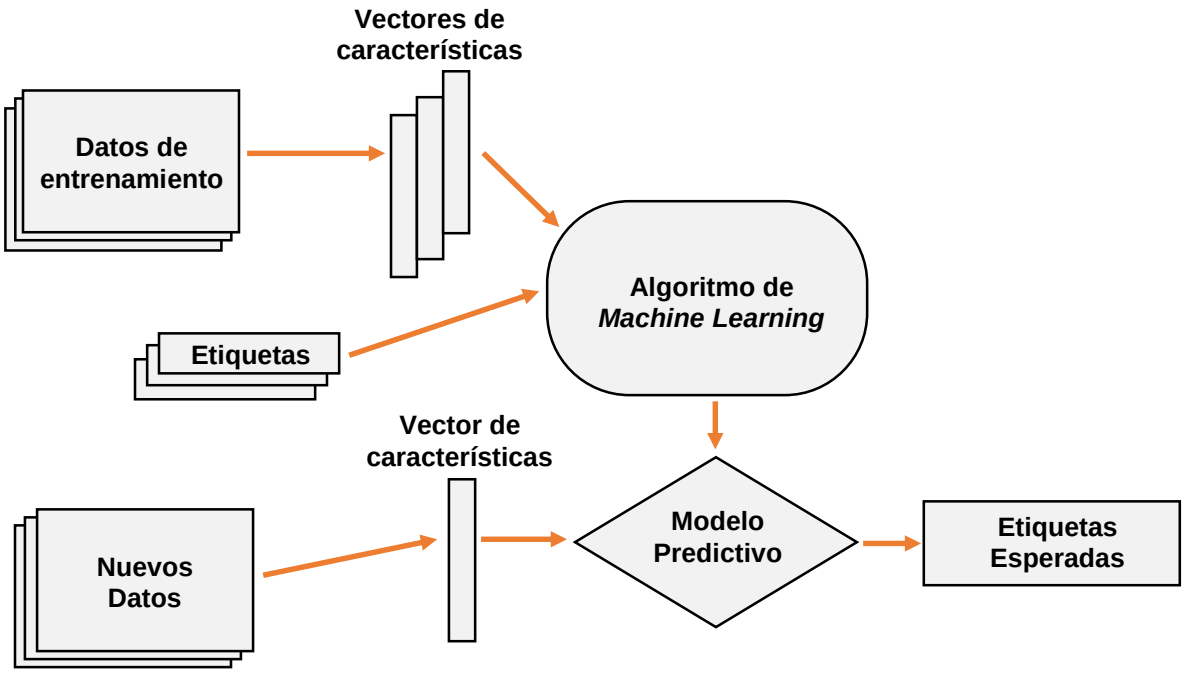
3.3 MACHINE LEARNING

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.



3.3.1 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.

$$^{argmin}_{\theta} J(\theta)$$

3.3.2 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.

3.3.2.1 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:

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

3.3.2.2 K neighbors. 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]:

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

3.3.2.3 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].

3.3.2.4 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].

3.4 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].

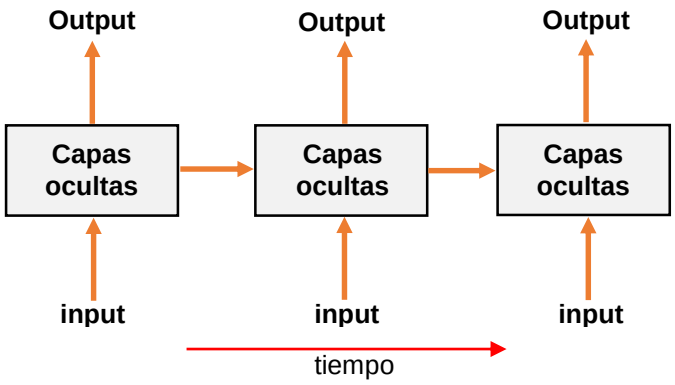
3.4.1 Redes neuronales recurrentes.

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.

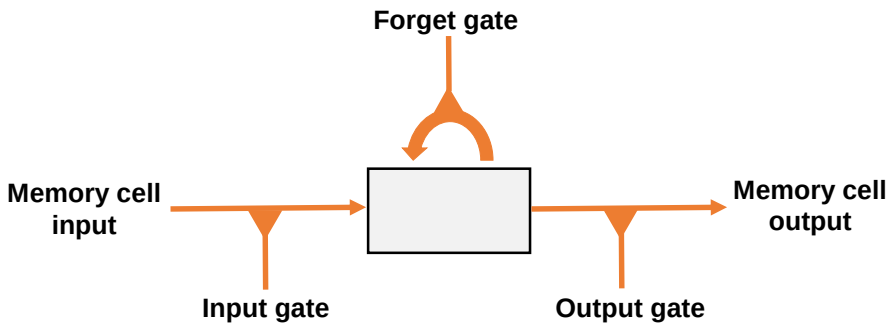
La figura 6 muestra la dinámica de una red neuronal a través del tiempo.

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



3.4.1.1 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:

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

3.5 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].

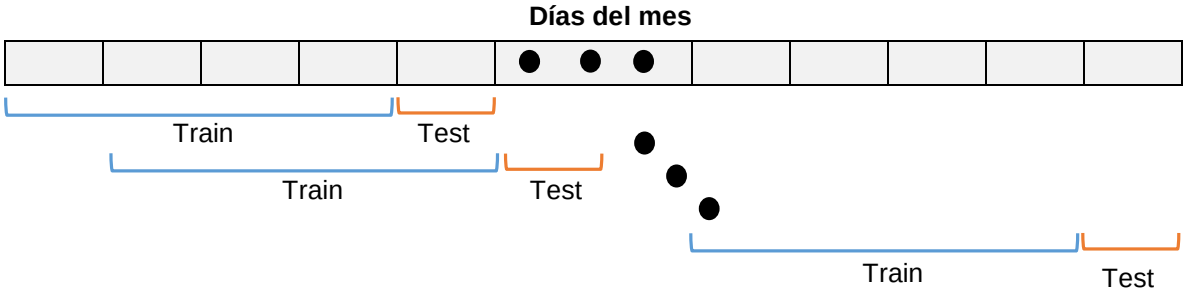


Figura 8. Representación de validación por días.

3.5.1 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.

3.5.1.1 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 y teniendo en cuenta que las celdas demarcadas con "-" se deben a que o no aplican o la estrategia de *trading* no opera:

Tabla 1. Definición de la métrica *PNL* según la estrategia de *trading*.

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

3.5.1.2 Métricas 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

El método propuesto en este trabajo de investigación fue puesto a prueba con datos públicos de fuentes como Quandl [23] y TrueFX [24]. Estos conjuntos de datos hacen referencia al par de divisas los cuales vienen dados en series de tiempo, donde por cada tiempo t se tienen tres valores, los cuales hacen referencia al *bid, ask* y *spread*. Se resalta que, para fines de evaluar el desempeño de los modelos predictivos planteados, se escoge el par de divisas EUR/USD ya que es el par con mayor cabida en el mercado *forex*, así mismo las predicciones realizadas están hechas en base al precio del *bid*. Siendo así, se emplea el pre-procesado descrito en la figura 1 para los datos del mes de enero del 2018, los cuales se muestrean en diferentes intervalos de tiempo tales como, 10 [s], 20 [s], 30 [s], 40 [s], 50 [s], 1 [min], 1[min] con 30 [s] y finalmente cada 5 [min], planteando así diferentes conjuntos de datos con diferentes tamaños, los cuales se ven con detalle en la tabla 2.

Tabla 2. Reducción de dimensión aplicando el muestreo.

Dimensión original	Muestreo	Nueva Dimensión
(3952896,1)	10 [s]	(184135,1)
(3952896,1)	20 [s]	(94850,1)
(3952896,1)	30 [s]	(63473,1)
(3952896,1)	40 [s]	(47651,1)
(3952896,1)	50 [s]	(38128,1)
(3952896,1)	1 [min]	(31777,1)
(3952896,1)	1 [min] 30[s]	(21186,1)
(3952896,1)	5 [min]	(6360,1)

Una vez tenido en cuenta el muestreo por segundos, se procede a calcular de forma experimental el tamaño óptimo de ventana n [10], siendo los valores de tamaño de ventana $n = 3$ y $n = 7$ los más apropiados para construir los modelos predictivos, esto en base a una metodología de prueba y error. Una vez definidos los tamaños de ventana a utilizar se procede a la construcción del conjunto de datos teniendo en cuenta lo explicado en la figura 2 y con un muestreo de 5 minutos, haciendo así que los datos pre-procesados cambien su tamaño de la siguiente manera:

Tabla 3: Cambio de dimensión según el tamaño de ventana n.

Tamaño de ventana	**Dimensión original**	**Nueva dimensión**
3	(6360,1)	(6357,3)
7	(6360,1)	(6353,7)

Con el conjunto de datos ya obtenido, se plantean unos modelos de predicción usando un método de clasificación binario, el cual se plantea mediante una regla simple que consiste en que si el $bid(t) < bid(t+1)$ la etiqueta objetivo es 1, y si el $bid(t) > bid(t+1)$ la etiqueta objetivo es 0. Se dice que es una regla simple debido a que el modelo maneja una estrategia de *trading* en la cual siempre hace una transacción en el mercado, ya sea de $compra(t)/venta(t+1)$, si la predicción es 1, o de $venta(t)/compra(t+1)$ si la predicción es 0, esto hace que el *PNL* tienda a ser negativo debido a que en ciertos intervalos de tiempo puede que el precio del $bid(t+1)$ suba respecto al precio del $bid(t)$, pero no logre subir lo suficiente para que sobrepase el precio del $ask(t)$, haciendo que el modelo haga una transacción de $compra(t)/venta(t+1)$ la cual genera pérdidas.

Dada la tarea de clasificación binaria, se procede a ser evaluada con los modelos clásicos de *machine learning* explicados anteriormente en este libro, teniendo en cuenta el método de validación representado en la figura 8, el cual genera un modelo predictivo en base a los datos de cuatro días y se pone a prueba con un

quinto día, esto así, con un paso de movimiento diario. Los resultados obtenidos para los datos del mes de enero de 2018, según el tamaño de ventana y el tiempo de muestreo se presentan a continuación:

Tabla 4. Rendimiento de los modelos para una ventana de tamaño 3 con una señal muestreada cada 10 [s]. (PNL dado en promedio diario).

Clasificador	**Muestreo**	**Tamaño de ventana**	**PNL**	**Precisión**
GaussianNB	10 [s]	3	-0.27	50.43%
Kneighbors	10 [s]	3	-0.2724	50.23%
DecisionTree	10 [s]	3	-0.2710	50.04%
RandomForest	10 [s]	3	-0.2709	50.07%

Tabla 5. Rendimiento de los modelos para una ventana de tamaño 3 con una señal muestreada cada 20 [s]. (PNL dado en promedio diario).

Clasificador	**Muestreo**	**Tamaño de ventana**	**PNL**	**Precisión**
GaussianNB	20 [s]	3	-0.1396	50.61%
Kneighbors	20 [s]	3	-0.1415	50.59%
DecisionTree	20 [s]	3	-0.1443	50.10%
RandomForest	20 [s]	3	-0.1401	50.42%

Tabla 6. Rendimiento de los modelos para una ventana de tamaño 3 con una señal muestreada cada 40 [s]. (PNL dado en promedio diario).

Clasificador	**Muestreo**	**Tamaño de ventana**	**PNL**	**Precisión**
GaussianNB	40 [s]	3	-0.0719	50.77%
Kneighbors	40 [s]	3	-0.0772	50.15%
DecisionTree	40 [s]	3	-0.0775	49.81%
RandomForest	40 [s]	3	-0.0761	50.30%

Tabla 7. Rendimiento de los modelos para una ventana de tamaño 3 con una señal muestreada cada 1 [min]. (PNL dado en promedio diario).

Clasificador	Muestreo	Tamaño de ventana	PNL	Precisión
GaussianNB	1 [min]	3	-0.0501	50.19%
Kneighbors	1 [min]	3	-0.0547	49.56%
DecisionTree	1 [min]	3	-0.0492	50.48%
RandomForest	1 [min]	3	-0.0513	50.32%

Tabla 8. Rendimiento de los modelos para una ventana de tamaño 3 con una señal muestreada cada 5 [min]. (PNL dado en promedio diario).

Clasificador	Muestreo	Tamaño de ventana	PNL	Precisión
GaussianNB	5 [min]	3	-0.0054	59.05%
Kneighbors	5 [min]	3	-0.0106	53.90%
DecisionTree	5 [min]	3	-0.0104	50.02%
RandomForest	5 [min]	3	-0.0083	51.18%

Dados estos resultados de clasificación binaria, se puede ver que a medida que el rango de muestreo t aumenta, el PNL disminuye, pero la precisión de los modelos no varía significativamente, esto se debe a que al aumentar el tiempo de muestreo, aumenta la probabilidad de generar ganancias por encima de la probabilidad de perdidas, por lo tanto, esto hace que el modelo teniendo aun una precisión casi igual para los diferentes muestreos, el PNL disminuya considerablemente.

Dado que un tamaño de ventana $n = 3$ es considerablemente pequeño, se hace necesario probar con un tamaño de ventana el cual brinde más datos acerca de cómo se viene comportando la señal, para ello se utiliza un tamaño de ventana de $n = 7$, el cual mediante la experimentación, arroja mejores resultados respecto al tamaño de ventana de $n = 3$. Para dar solución al problema anteriormente planteado acerca de la regla de *trading* simple, se plantea un problema de

clasificación multiclase, siendo así una clasificación con cuatro etiquetas de la siguiente forma:

- **Etiqueta 0 (cero):** Para los casos en los cuales se hace una acción de $venta(t)/compra(t+1)$ la cual genera ganancia.
- **Etiqueta 1 (uno):** Para los casos en los cuales se hace una acción de $compra(t)/venta(t+1)$ la cual genera ganancia.
- **Etiqueta 2 (dos):** Para los casos en los cuales se hace una acción de $venta(t)/compra(t+1)$ la cual genera pérdida.
- **Etiqueta 3 (tres):** Para los casos en los cuales se hace una acción de $compra(t)/venta(t+1)$ la cual genera pérdida.

Partiendo del planteamiento multiclase con el tamaño de ventana $n = 7$ y con la estrategia de *trading* para ello, la cual consiste en operar en el mercado solo para las predicciones de 1 (uno) ó 0 (cero), se tienen los siguientes resultados:

Tabla 9. Rendimiento de los modelos para una ventana de tamaño 7 con una señal muestreada cada 20 [s]. (PNL dado en promedio diario).

Clasificador	Muestreo	Tamaño de ventana	PNL	Precisión
GaussianNB	20 [s]	7	-0.1173	28.65%
Kneighbors	20 [s]	7	-0.1063	29.50%
DecisionTree	20 [s]	7	-0.0941	28.16%
RandomForest	20 [s]	7	-0.1092	29.29%

Tabla 10. Rendimiento de los modelos para una ventana de tamaño 7 con una señal muestreada cada 40 [s]. (PNL dado en promedio diario).

Clasificador	Muestreo	Tamaño de ventana	PNL	Precisión
GaussianNB	40 [s]	7	-0.0602	31.67%
Kneighbors	40 [s]	7	-0.0656	32.50%
DecisionTree	40 [s]	7	-0.0509	29.41%
RandomForest	40 [s]	7	-0.0619	32.26%

Tabla 11. Rendimiento de los modelos para una ventana de tamaño 7 con una señal muestreada cada 1 [min]. (PNL dado en promedio diario).

Clasificador	Muestreo	Tamaño de ventana	PNL	Precisión
GaussianNB	1 [min]	7	-0.0438	33.84%
Kneighbors	1 [min]	7	-0.0485	34.44%
DecisionTree	1 [min]	7	-0.0408	32.11%
RandomForest	1 [min]	7	-0.0412	32.85%

Tabla 12. Rendimiento de los modelos para una ventana de tamaño 7 con una señal muestreada cada 5 [min]. (PNL dado en promedio diario).

Clasificador	Muestreo	Tamaño de ventana	PNL	Precisión
GaussianNB	5 [min]	7	-0.0042	33.83%
Kneighbors	5 [min]	7	-0.0092	37.79%
DecisionTree	5 [min]	7	-0.0084	34.26%
RandomForest	5 [min]	7	-0.0069	40.48%

Como se observa, la precisión de los modelos aumenta debido al aumento de las características en la ventana, y el PNL tiende a disminuir en su pérdida debido a la estrategia de *trading* propuesta en la cual solo se opera en dos de sus cuatro clases.

Este mismo método de clasificación a cuatro clases se aplica a dos multiseñales, las cuales funcionan como lo muestra la figura 9. Para ello se toman las señales pre-procesadas de los pares EUR/USD, GBP/USD y CAD/USD para así predecir las clases del par EUR/USD (ver tabla 13).

Figura 9. Multiseñal con dos señales de entrada, EUR/USD y CAD/USD para predecir la señal EUR/USD.

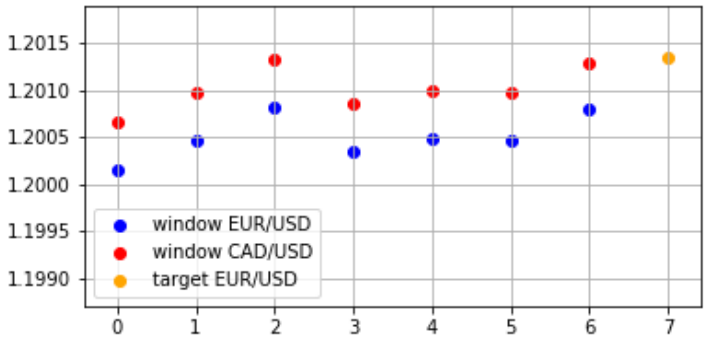


Tabla 13. Rendimiento de los modelos multiseñal para una ventana de tamaño 7 con una multiseñal muestreada cada 5 [min]. (PNL dado en promedio diario).

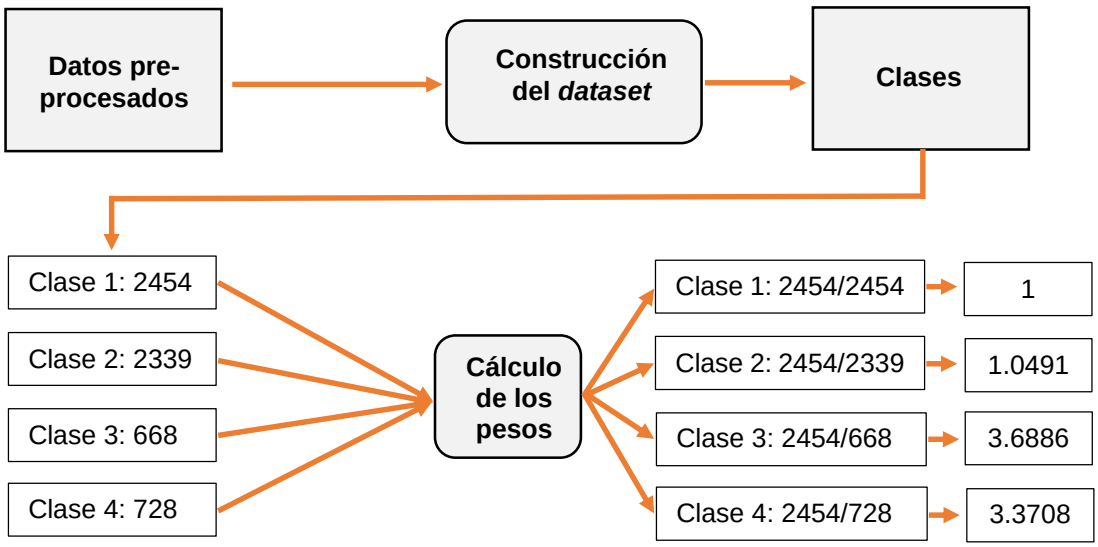
Clasificador	Muestreo	Tamaño de ventana	PNL	Precisión
GaussianNB	5 [min]	7	-0.0034	28.31%
Kneighbors	5 [min]	7	-0.0085	39.46%
DecisionTree	5 [min]	7	-0.0055	31.89%
RandomForest	5 [min]	7	-0.0075	38.69%

Se observa que el PNL mejora en cuanto a que la perdida diaria disminuye para modelos entrenados con una multiseñal, esto corrobora la hipótesis de que las señales financieras son interdependientes.

Una anotación importante es que al plantear la clasificación a cuatro clases el conjunto de datos se desequilibra a favor de las etiquetas 1 y 0, haciendo que los

modelos predictivos generen una predicción desbalanceada, lo que conlleva a una predicción errónea. Para ello se utiliza un vector de pesos el cual se le pasa al modelo a entrenar para que este tenga en cuenta las clases que tienen menos datos. La figura 10 representa cómo funciona el planteamiento de pesos.

Figura 10. Cálculo de los pesos para balancear las clases.



Planteado el vector de pesos se procede a probar como funciona con los modelos, tanto para señal simple EUR/USD, como para la multiseñal probada anteriormente.

Tabla 14: Rendimiento de los modelos para una ventana de tamaño 7 con una señal muestreada cada 5 [min] y un vector de pesos. (PNL dado en promedio diario).

Clasificador	Muestreo	Tamaño de ventana	PNL	Precisión
GaussianNB	5 [min]	7	-0.0023	26.15%
Kneighbors	5 [min]	7	-0.0087	37.51%
DecisionTree	5 [min]	7	-0.0079	39.25%
RandomForest	5 [min]	7	-0.0068	32.66%

Tabla 15: Rendimiento de los modelos multiseñal para una ventana de tamaño 7 con una multiseñal muestreada cada 5 [min] y un vector de pesos. (PNL dado en promedio diario).

Clasificador	Muestreo	Tamaño de ventana	PNL	Precisión
GaussianNB	5 [min]	7	-0.0029	24.52%
Kneighbors	5 [min]	7	-0.0079	39.13%
DecisionTree	5 [min]	7	-0.0072	39.88%
RandomForest	5 [min]	7	-0.0072	32.56%

Es notorio que el rendimiento promedio de los modelos tiende a mejorar cuando se implementa una multiseñal, esto debido a la magnitud de características que se generan para cada ventada. A su vez se implementa el *OHLC* para añadirlo a cada ventana (ver figura 3), obteniendo así la mejor predicción basándonos en el PNL para los métodos de *machine learning* propuestos.

Tabla 16: Rendimiento de los modelos para una ventana de tamaño 7 con una señal muestreada cada 5 [min], con un vector de pesos y su OHLC. (PNL dado en promedio diario).

Clasificador	Muestreo	Tamaño de ventana	PNL	Precisión
GaussianNB	5 [min]	7	-0.0013	26.12%
Kneighbors	5 [min]	7	-0.0105	34.47%
DecisionTree	5 [min]	7	-0.0072	34.60%
RandomForest	5 [min]	7	-0.0073	33.65%

Para cuestiones de claridad se grafican las estadísticas de los mejores resultados obtenidos con métodos de *machine learning*, los cuales hacen referencia a la tabla 16.

Figura 11. Mejor rendimiento para el modelo GausianNB (ver tabla 16)

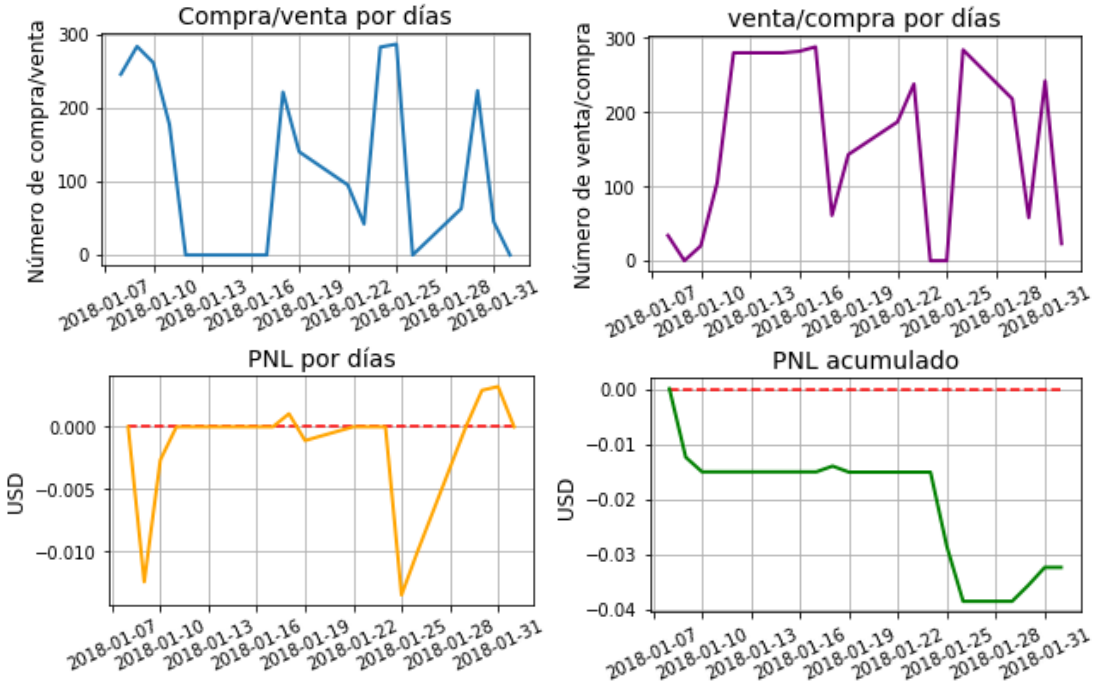


Figura 12. Mejor rendimiento para el modelo Kneighbors (ver tabla 16)

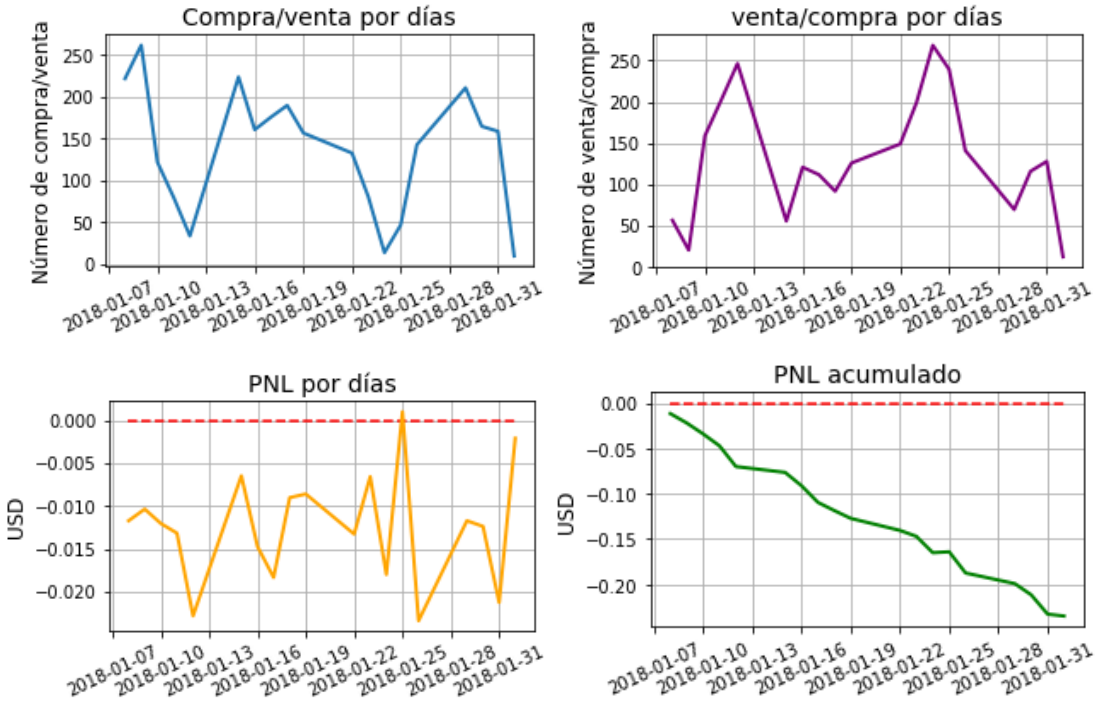


Figura 13. Mejor rendimiento para el modelo DecisionTree (ver tabla 16)

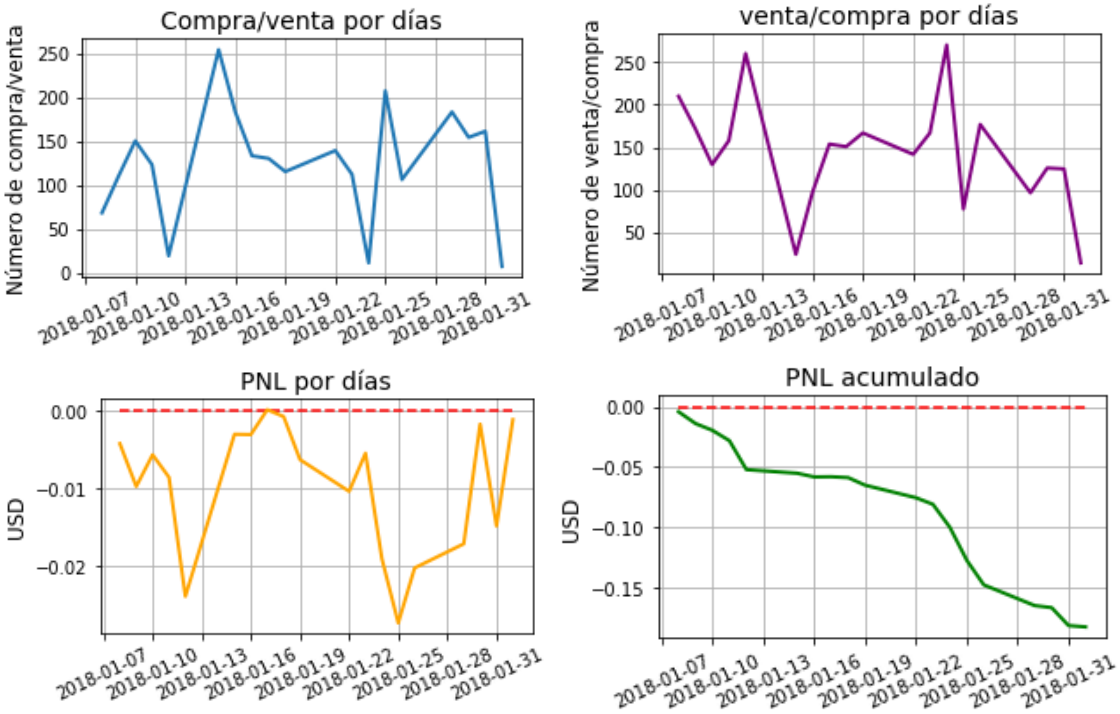
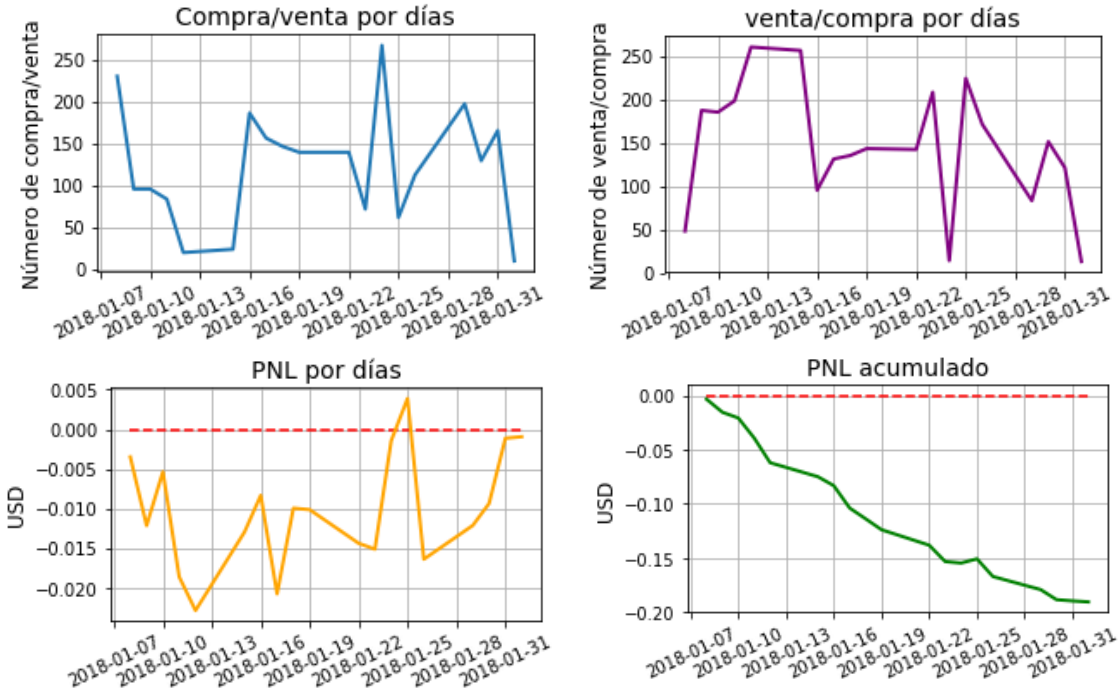
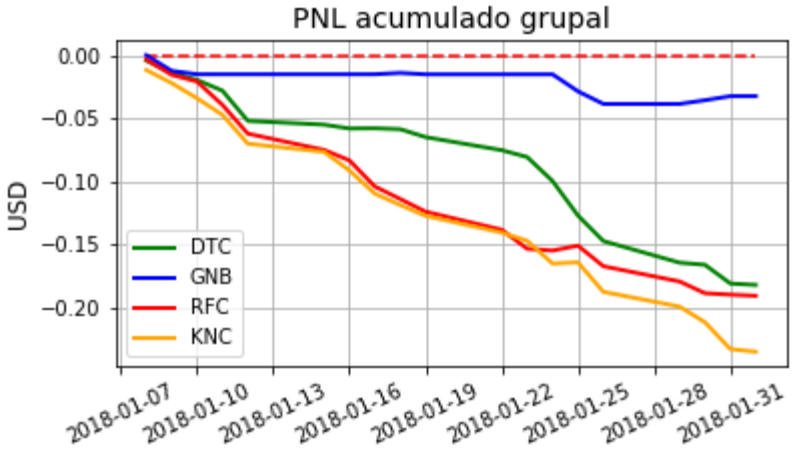


Figura 14. Mejor rendimiento para el modelo RandomForest (ver tabla 16)



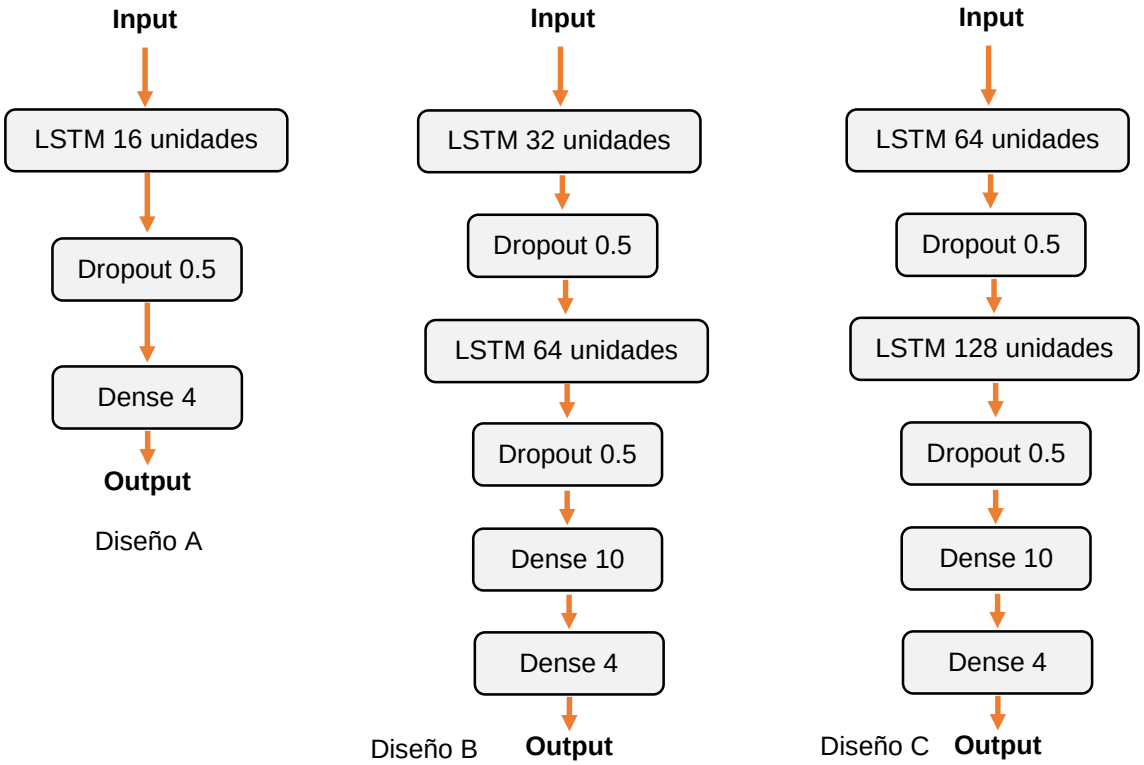
Para tener una idea de cuál modelo es mejor solamente viendo las gráficas, habría que recurrir a plasmar la acumulación o perdida de dinero de cada modelo en una sola gráfica, como se muestra a continuación, esto nos permite inferir que el GaussianNB es el más útil para esta tarea.

Figura 15. PNL acumulado de todos los modelos planteados.



Una vez obtenidos los resultados de los modelos de *machine learning,* se plantean algunas arquitecturas de redes neuronales recurrentes con unidades LSTM, las cuales se pueden observar en la figura 16. Aplicando la validación expuesta en la figura 8.

Figura 16. Múltiples Arquitecturas de Red Neuronal Recurrente.



Al entrenar y poner a prueba estas RNN, se puede comprobar la hipótesis del estado del arte, la cual propone que dichas redes se comportan mejor que los modelos de clásicos de *machine learning*. A continuación, se presentan los resultados obtenidos para los 3 diseños de arquitectura neuronal, teniendo en cuenta que estos diseños se evaluaron con el mes de junio de 2017.

Tabla 17: Rendimiento de los distintos diseños de RNN para una ventana de tamaño 7 con una señal muestreada cada 5 [min], con un vector de pesos y su OHLC. (PNL dado en promedio diario).

Diseño	Muestreo	Tamaño de ventana	PNL	Precisión
A	5 [min]	7	-0.0011	29.41%
B	5 [min]	7	-0.000055	28.75%
C	5 [min]	7	-0.00075	27.63

Figura 17. Rendimiento del Diseño A de la RNN (ver figura 16).

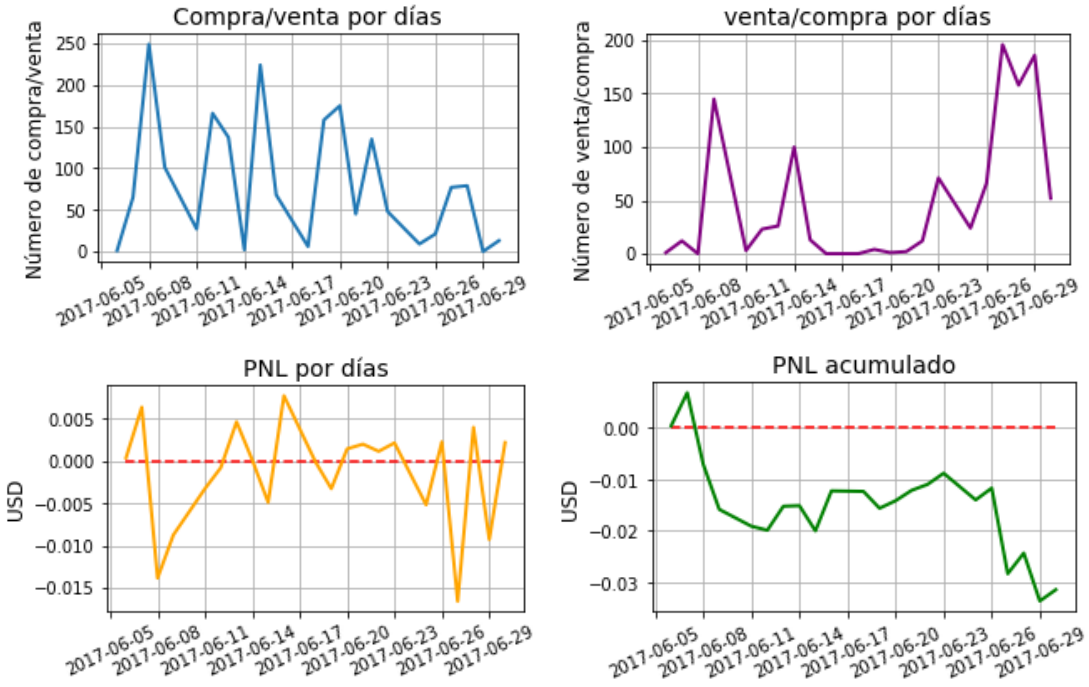


Figura 18. Rendimiento del Diseño B de la RNN (ver figura 16).

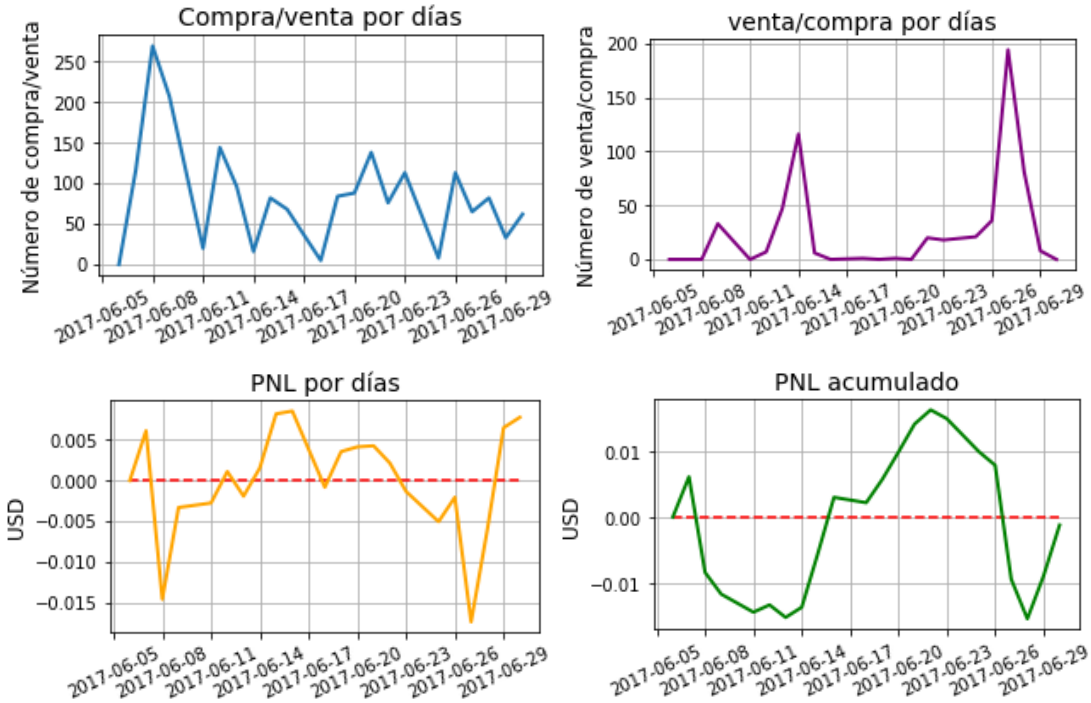
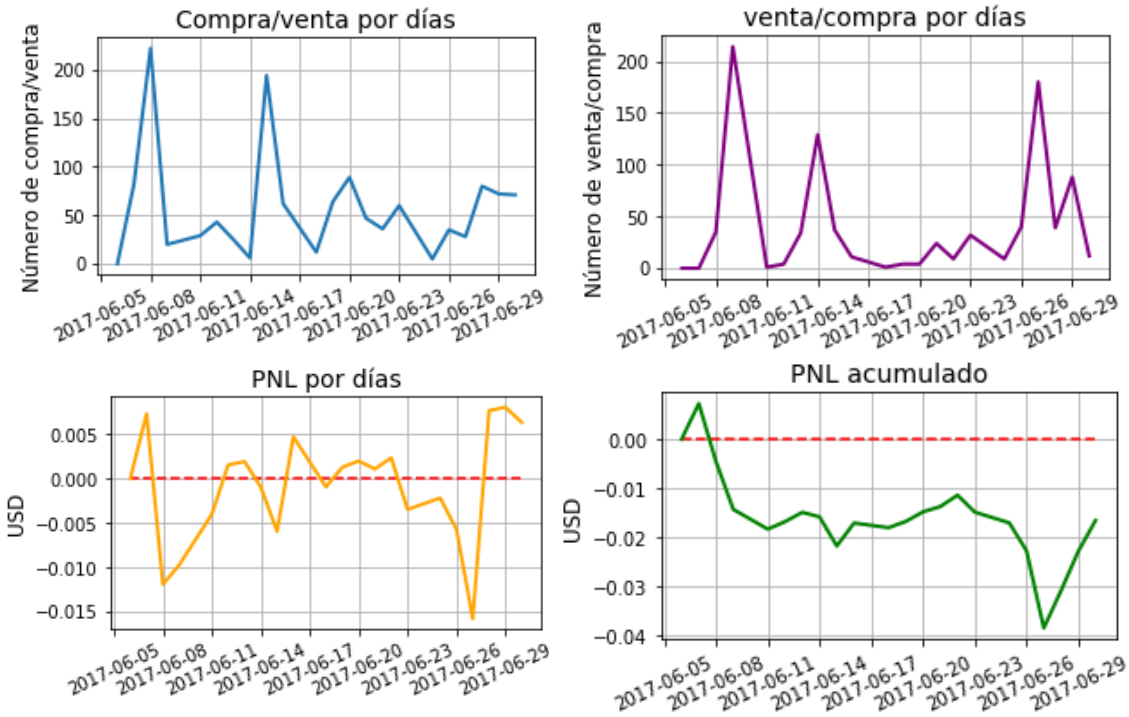
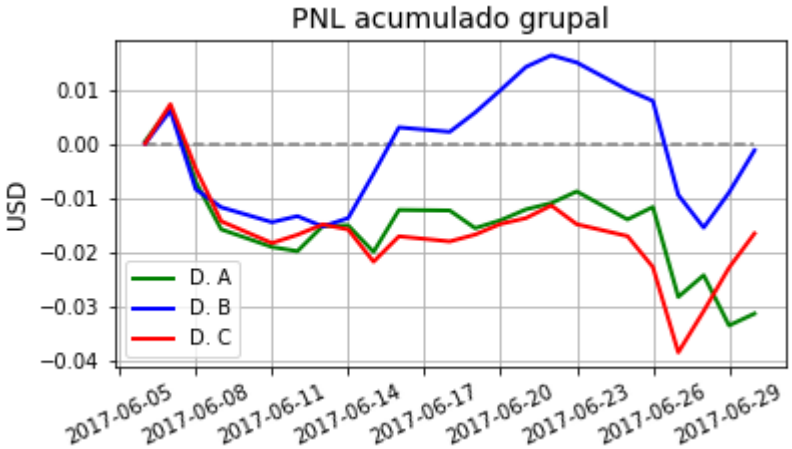


Figura 19. Rendimiento del Diseño C de la RNN (ver figura 16).



La figura 20 nos muestra con más precisión las diferencias entre las tres arquitecturas de red planteadas anteriormente, siendo el Diseño B el que presenta mejores resultados, inclusive en un momento dado con una ganancia acumulada.

Figura 20. Rendimiento de los tres diseños medidos por el PNL acumulado.



Procedemos a comparar el mejor resultado obtenido con los modelos de *machine learning* y las RNN para así ver cual tiende a comportase de la mejor manera.

Figura 21. Comparación del PNL acumulado para un mes entre el diseño de RNN B y el modelo planteado con un GaussianNB.

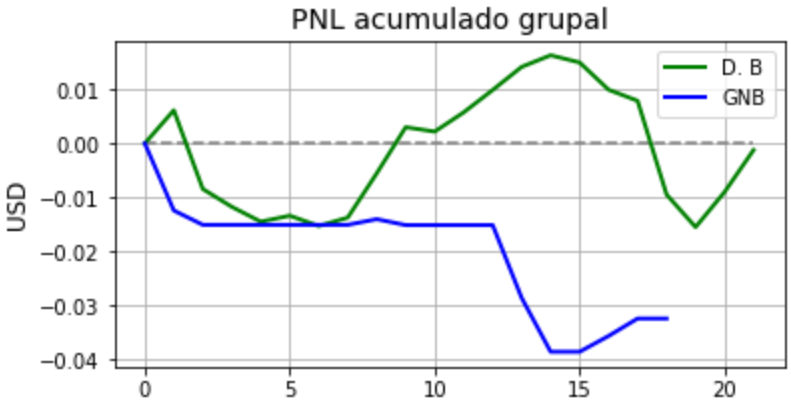


Tabla 18. Comparación entre los dos mejores modelos propuestos.

Modelo	GaussianNB	RNN B
Tamaño ventana	7	7
Muestreo	5 [min]	5 [min]
PNL	-0.0013	-0.00005566
% acierto clase 1	19.29%	38.34%
% acierto clase 0	4.87%	13.27%
% acierto clase 3	27.91%	55.39%
% acierto clase 2	53.21%	18.57%
% acierto total	15.83%	28.57%

Se puede observar que, aunque el modelo GaussianNB presenta poca pérdida de dinero, esto se debe a que poco opera en el mercado, siendo las clases 1 y 0 las cuales operan, el GaussianNB tiene un 24.16% de probabilidad para operar, mientras que la arquitectura de red neural propuesta tiene un 51.61% de operar en el mercado y aun así su rendimiento es mejor.

Finalmente se escoge el diseño de arquitectura que arrojó mejor resultado según los datos de la tabla 17 (Diseño B) y se entrena esta arquitectura con 3 meses de datos para su posterior uso. Obteniendo los siguientes resultados al entrenar el modelo con los datos del mes de enero de 2017 hasta mayo del mismo año, y probando con los datos del mes de junio.

Tabla 19. Rendimiento de la arquitectura B para datos sin validación por días.

Diseño	**Muestreo**	**Tamaño de ventana**	**PNL**	**Precisión**
B	5 [min]	7	-0.236	32.83%

5 CONCLUSIONES

- El proponer una clasificación binaria, no permite que se aproveche una estrategia de *trading* debido a la falencia presentada por la etiqueta, ya que esta me dice si el precio del *bid* sube, pero no prevé si este sube por encima del umbral necesario para generar ganancia.

- Según los resultados obtenidos con los modelos predictivos planteados, se puede establecer que basándonos en la estrategia de *trading* planteada dichos modelos funcionan mejor que una estrategia de *trading* basada en aleatoriedad, lo que nos da a entender que dichos modelos tratan de aprender características de las ventanas de entrada.

- Añadir características a la ventana n, tales como el OHLC hace que la precisión del modelo aumente, haciendo que la pérdida del modelo disminuya.

- Trabajar con multiseñales me permite aumentar las características de la ventana n, corroborando que las señales son interdependientes, haciendo que la predicción del modelo aumente.

- Debido a la variedad mínima en los datos, el tamaño de ventana n no hace que el desempeño de la predicción disminuya o aumente considerablemente, más, sin embargo, se aconseja siempre utilizar un tamaño acorde al muestreo de los datos.

- Las RNN funcionan mejor que los modelos clásicos de *machine learning*, ya que estas tienden a realizar mayor número de operaciones en el mercado y a perder menos dinero.

- La estrategia de *trading* “segura”, basada en las predicciones del modelo, permiten que se pierda muy poco dinero, aunque la precisión global del modelo sea baja.

- Los modelos de *machine learning* en general obtuvieron una precisión promedio del 32%, haciendo así, que la pérdida promedio diaria sea de 0.0013 USD, el equivalente a 3.705 COP diarios.

- Con las RNN se puede ver que estas generan ganancias ciertos días, pero su promedio de pérdida diaria es de 0.00005566 USD, el equivalente a 0.4634 COP.

- Con los resultados obtenidos con las diferentes arquitecturas de RNN probadas, se puede intuir qué si se implementara una RNN más robusta con una ventana con más características descriptivas de la señal, esto podría producir resultados más favorables.

6 RECOMENDACIONES

- Para futuras investigaciones, se deben tener en cuenta las características propias de la estadística financiera para añadirlas a cada ventana y así poder ver cómo esto afecta el nivel de precisión del modelo, que, según lo desarrollado en este trabajo de investigación, tiende a mejorar.

- Al momento de implementar una RNN, se podría construir una red más robusta en cuanto a capas y a unidades por capa, para ver cómo se comporta. la investigación me permite estimar que la precisión aumenta.

- Para la estrategia de *trading*, se podría plantear una que no dependa solo de la clase arrojada por el predictor, sino también del porcentaje de predicción con el cual el modelo hace dicha predicción.

REFERENCIAS

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

[2] Yu, Xuexin, *et al.* Solar Radio Spectrum Classification with LSTM.

[3] Xu, Lingzhi, *et al.* Password Guessing Based on LSTM Recurrent Neural Networks.

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

[5] Oliveira, Renato A., *et al.* Stock Market's Price Movement Prediction with LSTM Neural Networks.

[6] Lai, Ping-fu and Wong, Chung Hang. Performance of Stock Market Prediction.

[7] Use artificial neural networks to predict stock prices and their trends. [En línea]. (recuperado octubre de 2017). Disponible en stocksneural.net

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

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

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

[11] Mitchell, Tom M. Machine Learning.

[12] Gaussian Naive Bayes. [En línea]. Sklearn Documentation. Disponible en http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

[13] Nearest Neighbors. [En línea]. Sklearn Documentation. Disponible en http://scikit-learn.org/stable/modules/neighbors.html

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

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

[16] Decision Trees, Mathematical Formulation. [En línea]. Sklearn Documentation. Disponible en http://scikit-learn.org/stable/modules/tree.html

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

[18] Ensemble Methods, Random Forest Classifier. [En línea]. Sklearn Documentation. Disponible en http://scikit-learn.org/stable/modules/ensemble.html

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

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

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

[22] Frequently asked keras Questions. [En línea]. Keras Documentation. Disponible en https://keras.io/getting-started/faq/

[23] Core Financial Data. [En línea]. Quandl. Disponible en https://www.quandl.com

[24] TrueFX. [En línea]. Historical Tick-by-Tick Data. Disponible en https://www.truefx.com

BIBLIOGRAFÍA

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

Core Financial Data. [En línea]. Quandl. Disponible en https://www.quandl.com

Dua, Pami and Tuteja, Divya. Interdependence of International Financial Market.

Hinton, Geoffrey. University of Toronto: Neural Networks for Machine Learning.

Keras Documentation. [en línea]. The Python Deep Learning library. Disponible en https://keras.io

Mitchell, Tom M. Machine Learning.

Sklearn Documentation. [En línea]. Machine learning in Python. Disponible en http://scikit-learn.org

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

TrueFX. [En línea]. Historical Tick-by-Tick Data. Disponible en https://www.truefx.com

Xu, Lingzhi, *et al.* Password Guessing Based on LSTM Recurrent Neural Networks.

Yu, Xuexin, *et al.* Solar Radio Spectrum Classification with LSTM.