Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Matemáticas

Implementación de Redes Neuronales en Serie de Tiempo para generar un portafolio de inversión en Criptomonedas

Trabajo Especial de Grado presentado ante
la ilustre Universidad Central de Venezuela
por el **Br. Jorge A. Flores S.** para optar al
título de Licenciado en Matemática.

Tutor: Dr. José Benito Hernández.
Cotutor: MSc. Jesús Lares.

Caracas, Junio 2018

Nosotros, los abajo firmantes, designados por la Universidad Central de Venezuela como integrantes del Jurado Examinador del Trabajo Especial de Grado titulado " **Implementación de Redes Neuronales en Serie de Tiempo para generar un portafolio de inversión en Criptomonedas**", presentado por el **Br. Jorge A. Flores Sánchez**, titular de la Cédula de Identidad **20.328.317**, certificamos que este trabajo cumple con los requisitos exigidos por nuestra Magna Casa de Estudios para optar al título de **Licenciado en Matemática**.

Dr. José Benito Hernández
Tutor

prof. Robinson Rivas M.I.

MSc. Jesús Lares
Jurado

Dra. Mairene Colina
Jurado

Índice general

0.1. Resumen

El presente trabajo fue llevado a cabo con la finalidad de implementar dos tipos de redes neuronales las cuales se encargan de predecir el comportamiento y será aplicado en campo de las Criptomonedas, siendo este un tema de gran importancia en la actualidad debido al desarrollo tecnológico que va avanzando de manera exponencial y la economía se ve inmersa en este. Por otra parte, se llevó a cabo modelos de análisis de serie de tiempo llamados VAR, los cuales permiten estudiar el comportamiento a lo largo de los años. Adicionalmente se ejecutaron estas técnicas mediante el software de R y Python. Esto a su vez contribuye con la toma de decisiones al momento de generar un portafolio de inversión.

0.2. Palabras Claves:

Redes Neuronales, Serie de Tiempo, Criptomonedas.

El presente trabajo especial de grado está dedicado a todas aquellas personas que confiaron en mí. A su vez, a mis padres, novia, familiares y seres queridos.

Primeramente le agradezco a dios, a la virgen de Fátima, San Judas Tadeo y José Gregorio Hernández por haberme guiado durante todo el transcurso de la carrera.

A mis padres, Fabiola Sánchez y Jorge Flores los cuales son pilar fundamental en mi vida y gracias a ellos estoy donde estoy, además por todo el apoyo que me han dado. A mi hermana Fabiana por apoyarme.

Quiero agradecer enormemente a mi tutor el Dr. José Benito Hernández por su incansable labor, ayuda y correcciones en el presente trabajo. A su vez a mi cotutor el MSc. Jesús Lares ya que, él fue quien generó toda la motivación para realizar el trabajo especial de grado.

También quiero darle las gracias a mis compañeros de clases, Arturo Carreño, Miguel Porro, Juan Acosta por su ayuda incondicional durante toda la carrera. A su vez le agradezco a Alfredo Quintana, él fue de gran motivación y ayuda durante todo el proceso del presente trabajo especial de grado.

Por último quiero darle las gracias a mi novia, Isabel Oliveros. Una persona muy importante en vida, estuvo conmigo desde la mitad de la carrera y nunca dejó de apoyarme, siempre confió en mí y me ayudo enormemente en todo el proceso de este trabajo.

Capítulo 1

Introducción

1.1. Introducción

El presente trabajo especial de grado surge bajo el interés de exponer diferentes técnicas que serán implementadas para predecir el comportamiento (valor) que tendrán las Criptomonedas. En la actualidad en el mercado de las Criptomonedas exiten alrededor de 1575 modedas, las cuales se dividen en dos grupos: Minables y No Minables, caracterizándose la primera por tener una cantidad finita y en la segunda su cantidad puede aumentar al transcurrir el tiempo. Para dicho trabajo se seleccionó una muestra de cinco monedas las cuales fueron: **Bitcoin, Dash, Etherum, Litecoin y Ripple.** Los datos para realizar dicho trabajo se obtuvieron mediante web Scraping vía la extensión de Google Chrome y se extrajeron de la página web llamada Coinmarketcap. Para ver los datos puede hacer click en el enlace que se presenta a continuación (Datos).

Las técnicas implantadas están basadas en Redes Neuronales y análisis de Serie de Tiempo Multivariado, especialmente se llevó a cabo los modelos **VAR** (Vectores Autoregresivos). A su vez, se utilizaron dos Redes Neuronales, una **LSTM** (Long Short Term Memory), la cual permite observar cómo será el comportamiento de la Criptomoneda a lo largo del tiempo, esto se podrá apreciar mejor en el capítulo 4 de análisis y resultados. Por otra parte, fue ejecutada la Red Neuronal clásica **MLP** (Multilayer Perceptron) con la finalidad de predecir si el valor de la moneda aumenta o disminuye.

El estudio es realizado mediante los software de R y Python, en estos se trabaja de la siguiente forma. En el lenguaje de programación Python se implementan los dos tipos de Redes Neuronales mencionadas anterior-

mente, luego en el lenguaje de programación R se lleva a cabo el análisis de Serie de Tiempo el cual se enfoca en los modelos **VAR**.

Las técnicas mencionadas anteriormente tienen como finalidad facilitar la toma de decisiones al comprador en el momento que desee invertir en el mercado de Critptomonedas, siendo este un mercado que genera cierta desconfianza por el riesgo que percibe el individuo a la hora de invertir. Mediante este trabajo especial de grado se pretende generar confianza siempre y cuando su interés sea las monedas con que realizamos el estudio.

El trabajo de especial de grado está estructurado en cuatro capítulos, los cuales se describen a continuación:

• **Capítulo 1.** Este es el capítulo introductorio al trabajo especial de grado, también se incluyen algunos estudios previos y se plantean los objetivos.

• **Capítulo 2.** Este capítulo consta de toda la parte teórica sobre las redes neuronales MLP y LSTM, a su vez, se definen las métricas que serán utilizadas para medir el modelo.

• **Capítulo 3.** En este apartado se da una breve introducción sobre el tema serie de tiempo, además se explica el fundamento teórico de los modelos VAR y con qué finalidad es aplicado al presente trabajo especial de grado.

• **Capítulo 4.** En este capítulo se describe todo el proceso que se realizó para encontrar el mejor modelo, a su vez, el análisis y resultados de las tres técnicas que fueron implementadas con la finalidad de proporcionar un portafolofio de inversión basándonos en las monedas con las que se realizó el trabajo especial de grado.

1.2. Antecedentes y Motivación

Actualmente el sector financiero está experimentando grandes avances y a su vez transformaciones debido a las exigencias y demanda de los consumidores. De acuerdo con esto y la tendencia a la adaptación de las nuevas tecnologías entra en contexto la criptomoneda como una opción de medio financiero en donde el consumidor utiliza la moneda digital bajo el mismo sentido de la moneda física pero todo mediante internet, en donde se permiten transacciones y transferencias sin límite. (Sánchez Gil , Terán Varela, 2018).

Quien describe el término de criptomoneda como un nuevo tipo de dinero descentralizado es Wei Dei en 1998. Es importante destacar que esta tecnología está basada en la criptografía que es la ciencia que se dedica al estudio de métodos de encriptación de información. (Ordinas, 2017).

Ahora bien, en todo sistema de transacciones debe existir en físico un archivo en donde se encuentra reflejado el saldo de quienes poseen cuentas bancarias. Esto está cerrado al público por lo cual se necesita de una tercera persona como gobiernos, banco, notarios entre otros para poder aprobar las transacciones. Esto a partir del año 2008 es diferente para quienes invierten en criptomoneda, ya que, existe la tecnología blockchain que elimina las terceras partes y es definida como "libro contable público descentralizado diseñado para registrar las transacciones en un entorno protegido". Dicha tecnología es la plataforma base de la criptomoneda Bitcoin, siendo esta última una de las infinitas aplicaciones de la tecnología blockchain. (Georges, 2017).

Debido a que hoy en día se está haciendo cada vez más conocido el término criptomoneda y a su vez ha aumentado exponencialmente el interés de muchos individuos con respecto a esto, parece una idea pertinente conocer el comportamiento de estas para incentivar a quienes aún sienten que puede ser un riesgo invertir en esta nueva tecnología. De acuerdo con esto, un método para predecir el comportamiento es mediante una serie de tiempo que es " un conjunto de observaciones x_t , cada una registrada a un tiempo específico t". Dado el comportamiento de las criptomonedas, esto generará una gran cantidad de datos lo cual permite introducirse en el área de Ciencia de datos. Esta ciencia se encarga de impartir conocimientos, encontrar algún patrón o comportamientos como resultado de aplicar ciertos algoritmos y efectuar dicho análisis a los datos.

En esta oportunidad cada serie de tiempo permitirá realizar los respectivos análisis desarrollando diferentes metodologías adicionales a los modelos clásicos de serie de tiempo, esta metodología será las redes neuronales. Estas son una amplia familia de algoritmos que han formado la base para el reciente resurgimiento en el campo computacional llamado Deep Learning. Los primeros trabajos en redes neuronales en realidad comenzaron en los años 50 y 60. Recientemente, ha experimentado un resurgimiento de interés, ya que, el Deep Learning ha logrado impresionantes resultados de vanguardia en tareas específicas que van desde la clasificación de objetos en imágenes, hasta la traducción automática rápida y precisa.

De acuerdo a lo expuesto anteriormente y las investigaciones previas, para la realizar el presente trabajo especial de grado se pautó como objetivo general implementar diferentes técnicas basadas en Redes Neuronales con la finalidad de prestar una ayuda en la toma de decisiones en el momento que se desee invertir en el mercado de Criptomonedas, sin embargo, hay factores externos que no están al alcance de los algoritmos, pero aun así, la finalidad del autor es llevar a cabo dicho trabajo con la mayor precisión posible.

Ahora bien, del objetivo de la investigación derivan objetivos específicos, en donde se plantea que:

El estudio será realizado basándose en el comportamiento de las monedas mencionadas anteriormente a partir de la fecha del 1ro de Enero del 2016 hasta el 12 de Marzo del 2018. Este período es el seleccionado para el estudio debido a que en los dos últimos años es donde el valor de la Criptomoneda ha tenido un comportamiento aleatorio, ya que, en los años anteriores su comportamiento siempre tendió a crecer y en esta oportunidad no sería de gran ayuda para obtener una buena predicción de su valor.

El trabajo es llevado a cabo mediante técnicas en donde es utilizado el lenguaje de programación Python y R. Para hacer posible esto, en Python se utilizaron las librerías de Pandas, Matplotlib, Numpy, Scikit-Learn, Keras. Todas estas librerías son de gran ayuda para realizar el estudio, con ellas se logran la lectura de los datos, realizar los respectivos gráficos y además implementar los algoritmos que en esta oportunidad son Redes Neuronales. En el software de R la librería utilizada es VARS donde se ejecutarán los modelos VAR y luego hacer el respectivo análisis de Serie de Tiempo.

Mediante un estudio de investigación previo se obtuvo que la Red Neuronal que mejor se ajusta cuando se pretende trabajar con datos relacionados al tiempo, siendo estos los datos que generan una Serie de Tiempo es la Red Neuronal LSTM, esto será explicado detalladamente en el capítulo 2.

Por último, será generado un portafolio de inversión en el mercado de Criptomodenas, tomando en cuenta las cinco monedas con la que se realizó el trabajo, y a su vez, teniendo la finalidad de facilitar y guiar al individuo interesado en invertir en dicho mercado, brindando mayor confiabilidad mediante un porcentaje de seguridad donde la inversión genere más beneficio que riesgo.

Capítulo 2

Redes Neuronales

Para dar inicio al capítulo se describe qué es una red neuronal, siendo esta un modelo simplificado del modo en que el sistema nervioso humano procesa la información. Por otra parte, se desarrolla una aproximación teórica de las redes neuronales MLP y LSTM, estas son dos de los tres modelos a implementar para predecir el comportamiento de las Criptomonedas mencionadas en el capítulo 1.

Cabe destacar que las redes neuronales mencionadas anteriormente son herramientas útiles en aplicaciones de la predicción en la minería de texto.

2.1. Red Neuronal MLP

De acuerdo con (Picón Viana, 2011) una de las redes neuronales utilizadas en el análisis de clasificación es la MLP (Multilayer Perceptrons). Un MLP está compuesto por una capa de entrada, una de salida y una o más capas ocultas. La información siempre es transmitida desde la capa de entrada hacia la capa de salida.

Ahora bien, los autores (Goodfellow, Bengio, y Courville, 2016) mencionan que es importante explicar un tipo de red neuronal artificial llamada perceptron. Los perceptrones fueron desarrollados en los años 1950 y 1960 por el científico Frank Rosenblatt, inspirado por trabajos anteriores de Warren McCulloch y Walter Pitts siendo esta la base de las redes neuronales.

Un perceptron toma varias entradas binarias y produce una sola salida binaria. Rosenblatt propuso una

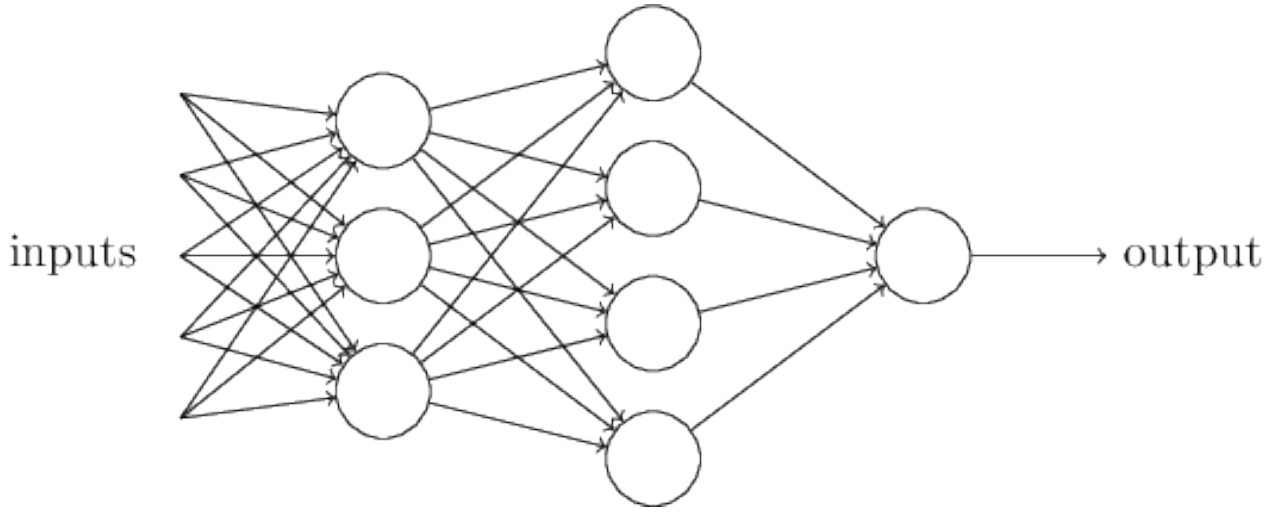
regla para computar la salida. El introdujo peso, los cuales son números reales que expresan la importancia de la respectiva entrada hacia la salida. La salida de la red neuronal es determinada por si la suma ponderada de estos pesos es mayor o menor a determinado parametro de la red neuronal. A continuación se muestra en términos algebráicos lo expuesto anteriormente:

$$salida = \begin{cases} 0 & si \quad \sum_j w_j x_j \leq A \\ 1 & si \quad \sum_j w_j x_j > A \end{cases} ;$$

donde x representa las entradas, w representa los pesos y A representa el parámetro definido en la red.

Los perceptrones pueden ser utilizados para la toma de decisiones en donde el valor que tenga más peso será tomado en cuenta para producir la salida. Al variar los pesos y el parámetro definido en la red podemos tener diferentes modelos de toma de decisión.

En la imagen que se muestra a continuación se describe como los perceptrones se conectan entre si y además como es el funcionamiento de la red. (Goodfellow, Bengio, y Courville, 2016)



En la primera columna (Layer) de la red, los perceptrones estan realizando tres decisiones simples al ponderar la evidencia de las entradas. En la segunda columna (capa oculta) de perceptrones cada uno de ellos toma una decisión al sopesar los resultados de la primera columna de toma de decisiones, de esta manera

los perceptrones de la segunda columna toman la decisión de manera más abstracta y compleja que la de la primera columna y al finalizar la decisión que tenga más peso será la salida.

A continuación se explica cómo es el proceso paso a paso que se realiza en la red MLP, primero se toman las entradas, éstas entran en los perceptrones, luego que esto se realice obtenemos una salida y esta salida va dirigida hacia la capa oculta que recibe como entrada neta lo siguiente,

$$net_j = \sum_{i=j}^{N} w_{ij} x_i + \theta_j,$$

donde θ es el umbral de la neurona que se supone como peso asociado a una neurona ficticia con un lugar de salida igual a 1.

Las neuronas en esta capa oculta aplican funciones no lineales, estas funciones se conocen como funciones de activación con el fin de transformar la señales recibidas. Por lo general una de estas funciones es la tangente hiperbólica $(\tanh)$ sin embargo, existenten otros tipos de funciones. Aplicando esta función a la señal recibida se obtiene lo siguiente,

$$b_j = f(net_j),$$

donde f es la función de activación, net_j es la entrada a la capa oculta y b_j es el valor de salida de la neurona j.

Este valor se transfiere a través de los pesos V_{kj} hacia la capa de salida:

$$net_k = \sum_{j=1}^{L} V_{kj} b_j + \theta_k.$$

En la capa de salida se aplica la misma operación que en la capa anterior, las neuronas de esta última capa proporcionan la salida, y_k de la red que viene dada por:

$$y_k = f(net_k).$$

Luego, empieza una etapa de aprendizaje o entrenamiento de la Red Neuronal, el objetivo de este es minimizar el error entre la salida obtenida por la red y la salida deseada.

La función de error que se intenta minimizar para cada patrón p, E_p, está definida por:

$$E_p = \frac{1}{2} \sum_{k=1}^{M} (d_{pk} - y_{pk})^2$$

donde d_{pk} es la salida esperada para la neurona de salida k y y_{pk} es la salida que proporciona la red ante la presentación del patrón p.

Como E_p es función de todos los pesos de la red, el gradiente de E_p es un vector igual a la derivada parcial de E_p respecto a cada uno de los pesos. El gradiente toma la dirección que determina el incremento más rápido en el error, mientras que la dirección opuesta determina el decremento más rápido en el error. Por tanto, el error puede reducirse ajustando cada peso en la dirección y se puede expresar de la siguiente forma:

$$-\sum_{p=1}^{p} \frac{\partial E_p}{\partial w_{ij}}.$$

Una vez que fue expresado el procedimiento de la arquitectura de la red, se procede a explicar el alcance que tiene la red neuronal MLP en este estudio y con qué finalidad se realizó.

En primer lugar se extrajeron los datos de la página Coinarketcap, a continuación se anexa la imagen en donde se muestra como están expresados estos:

Date	Open	High	Low	Close	Volume	Market Cap
2016-01-01	430.72	436.25	427.52	434.33	36,278,900	6,473,530,000
2016-01-02	434.62	436.06	431.87	433.44	30,096,600	6,533,630,000
2016-01-03	433.58	433.74	424.71	430.01	39,633,800	6,519,500,000
2016-01-04	430.06	434.52	429.08	433.09	38,477,500	6,468,180,000
2016-01-05	433.07	434.18	429.68	431.96	34,522,600	6,515,380,000
2016-01-06	431.86	431.86	426.34	429.11	34,042,500	6,498,830,000
2016-01-07	430.01	458.77	429.08	458.05	87,562,200	6,472,580,000
2016-01-08	457.54	462.93	447.94	453.23	56,993,000	6,888,600,000
2016-01-09	453.38	454.64	446.89	447.61	32,278,000	6,828,000,000
2016-01-10	448.24	448.31	440.35	447.99	35,995,900	6,752,210,000
2016-01-11	448.70	450.66	443.86	448.43	40,450,000	6,761,090,000
2016-01-12	448.18	448.18	435.69	435.69	115,607,000	6,755,220,000
2016-01-13	434.67	435.19	424.44	432.37	173,888,000	6,553,350,000
2016-01-14	432.29	433.32	427.85	430.31	43,945,500	6,519,110,000
2016-01-15	430.26	430.26	364.33	364.33	153,351,000	6,489,870,000
2016-01-16	365.07	390.56	354.91	387.54	120,352,000	5,507,790,000
2016-01-17	387.15	390.97	380.09	382.30	45,319,600	5,842,270,000

Figura 2.1: Una muestra del archivo de datos de la moneda Bitcoin, donde se aprecian las variables Open, High, Low, Close, Volume y Market Cap.

Como se aprecia en la Figura 2.1, aparecen las columnas Open, High y Low. Dichas variables son independientes y de estas va a depender la variable Close siendo esta la variable dependiente. Este procedimiento se transformó en un problema de aprendizaje supervisado, con la finalidad de predecir si aumenta o decrece el valor de la columna Close. Para ello se llevó a cabo el siguiente procedimiento:

En la columna Close, se realizó una operación en donde el resultado fue una clasificación binaria en la cual " cero " denota que el valor decrece y "uno"que el valor aumenta. La operación consiste en restar el valor del día siguiente con el valor del día anterior y si el resultado de esta resta es menor que cero se le asigna " cero " y si por el contrario, el resultado es mayor que cero se le asigna " uno ".

Luego de realizar este procedimiento se implementa la red neuronal MLP en Python conocida como **MLP-Classifier.** Para medir el modelo se utilizó el método conocido como matriz de confusión el cual consiste en lo siguiente:

MATRIZ DE CONFUSIÓN PARA CLASIFICACIÓN BINARIA

TN	FP
FN	TP

Donde **TN:** Verdadero Negativo, **FN:** Falso Negativo,**TP:** Verdadero Positivo y **FP:** Falso Positivo.

Ahora bien, una vez ha sido observada la estructura de la matriz, procedemos a definir las medidas que están inmersas en dicha matriz:

Accuracy: Mide la bondad de un modelo de clasificación como la proporción de resultados verdaderos en el total de las instancias. A continuación se muestra la fórmula:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}.$$

Recall: Es la precisión de todas las instancias positivas que el modelo clasifica correctamente.

$$Recall = \frac{TP}{TP + FN}.$$

Precision: Es la proporción de todos los resultados positivos.

$$Precision = \frac{TP}{TP + FP}.$$

FScore: Calculado como el promedio ponderado de Precision y Recall entre 0 y 1, donde el valor ideal es 1.

$$FScore = \frac{2TP}{2TP + FN + FP}.$$

Los resultados y el análisis se podrán apreciar en el Capítulo 4.

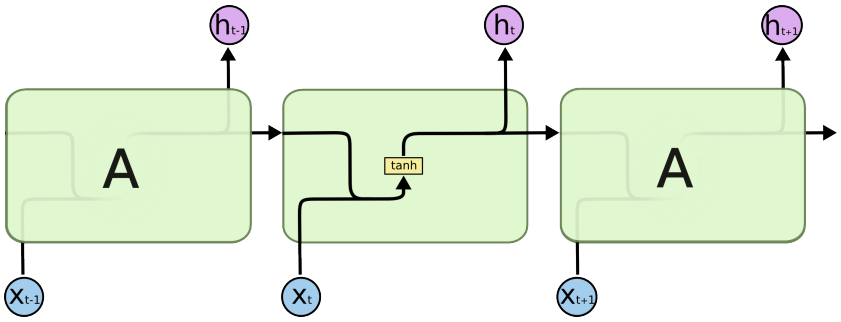
2.2. Red Neuronal LSTM

Las redes neuronales recurrentes también conocidas como RNN (por su siglas en inglés " Recurrent Neuronal Networks ") son redes con ciclos que permiten que la información persista o se conecte entre sí. Estos ciclos permiten que la información avance desde el paso anterior hasta el siguiente paso dentro de la red. Estas redes se pueden considerar como copias múltiples de la misma red, cada una pasando un mensaje a su sucesor. (Olah, 2015).

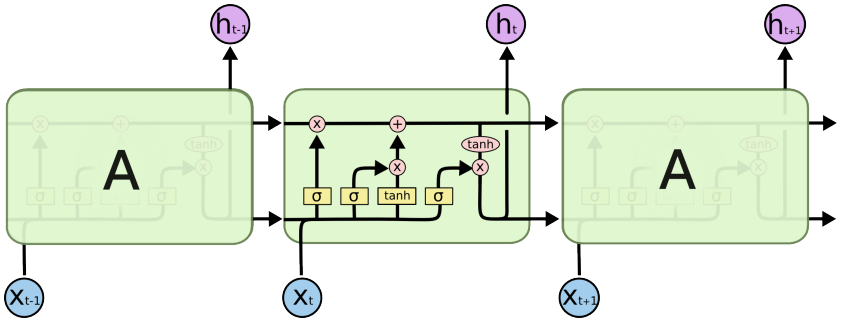
Las LSTM (redes de memoria a corto plazo), son un tipo especial de RNN, y son capaces de aprender dependencias a largo plazo. Estas redes fueron introducidas por Hochreiter y Schmidhuber (1997). Adicionalmente, muchos autores como Felix Gers, Fred Cummins, Santiago Fernandez, Justin Bayer, Daan Wierstra, Julian Togelius, Faustino Gomez, Matteo Gagliolo, and Alex Graves han contribuido a lo largo del tiempo con el estudio hasta la actualidad del LSTM. Estas funcionan de manera eficaz en una gran variedad de problemas, y ahora son ampliamente utilizadas. (Olah, 2015).

Las LSTM están diseñadas explícitamente para evitar el problema de dependencia a largo plazo. Recordar la información durante largos períodos de tiempo es prácticamente su comportamiento predeterminado. Todas las redes neuronales recurrentes tienen forma de una cadena de módulos (Olah, 2015).

De acuerdo con (Olah, 2015) todas las redes neuronales recurrentes tienen la forma de una cadena de módulos repetitivos de red neuronal. En RNN estándar, este módulo de repetición tiene una estructura muy simple, como una sola capa que contiene la función tangente hiperbólica(tanh).



A continuación se presenta la imagen de como está estructurada la red neuronal LSTM.



En la imagen anterior, los LSTM también tienen esta estructura tipo cadena, pero el módulo de repetición tiene una estructura diferente. En lugar de tener una sola capa de red neuronal, hay cuatro capas interactuando de una manera muy especial. A su vez, en el diagrama de arriba, cada línea lleva un vector completo, desde la salida de un nodo hasta las entradas de otros.

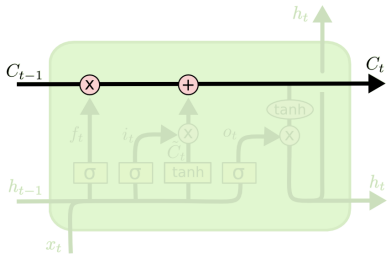
Los círculos rosados representan operaciones puntuales, como la adición de vectores, mientras que los cuadros amarillos son capas de aprendizaje de la red neuronal. Las líneas que se fusionan denotan concatenación,

mientras que una línea de horquilla denota que su contenido se copia y las copias van a diferentes ubicaciones.(Olah, 2015).

La idea central detrás de los LSTM

La clave para LSTM es el estado de la celda, la línea horizontal que se extiende por la parte superior del diagrama.

El estado de la celda es como una cinta transportadora. Corre directamente por toda la cadena, con solo algunas interacciones lineales. Debido a este comportamiento es muy fácil que la información fluya sin cambios.

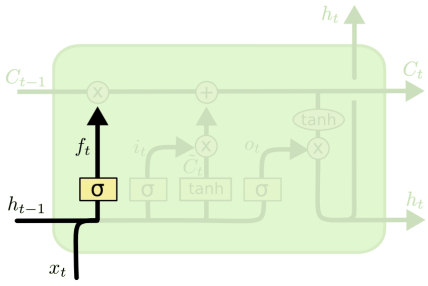


Ahora bien, es importante destacar que las LSTM tienen la capacidad para eliminar o agregar información. Esto es regulado por "puertas". Estas puertas conllevan a que exista la opción de la entrada de información. Están compuestas por funciones de activación. Estas funciones de activación muestran números entre 0 y 1, teniendo como finalidad permitir o denegar el acceso de la información. El valor cero indica que no debe entrar nada, y el valor 1 indica que la información puede entrar. (Olah, 2015).

Recorrido de la LSTM paso a paso

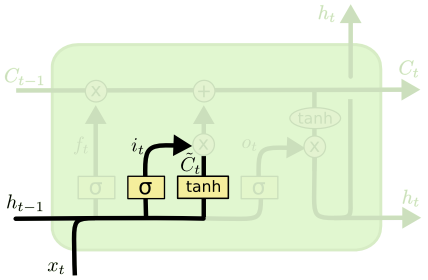
El primer paso en nuestra LSTM es decidir qué información vamos a arrojar del estado de la celda. Esta decisión es tomada por una función de activación en la capa llamada "capa de puerta de olvido". Mira h_{t-1} y x_t, y genera un número entre 0 y 1 para cada número en el estado de celda C_{t-1}. Donde 1 representa "mantener completamente esto" mientras que un 0 representa "deshacerse por completo de esto". (Olah, 2015).

A continuación se presenta la imagen del proceso que se está realizando



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

El siguiente paso es decidir qué nueva información vamos a almacenar en el estado de la celda. Esto tiene dos partes. Primero, una capa en la cual contiene una función de activación, dicha capa es llamada "capa de puerta de entrada" en la cual decide qué valores actualizaremos. A continuación, otra capa contiene una función de activación la cual es la función tangente hiperbólica $(tanh)$ con la finalidad de crear un vector de nuevos valores candidatos, C_t, eso se podría agregar al estado. En el siguiente paso, combinaremos estos dos para crear una actualización del estado. (Olah, 2015).

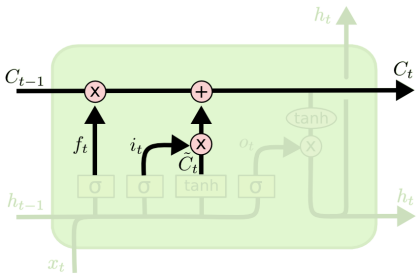


$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\overline{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

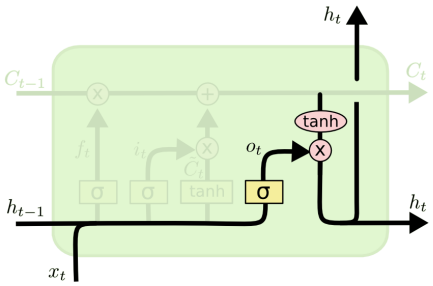
Ahora es el momento de actualizar el estado de celda anterior, C_{t-1}, en el nuevo estado de celda C_t. Los pasos anteriores ya decidieron qué hacer, solo tenemos que hacerlo.

Multiplicamos el estado anterior por f_t, olvidando las cosas que decidimos olvidar antes. Luego le agregamos $i_t * \overline{C}_t$. Estos son los nuevos valores candidatos, ajustados por cuánto decidimos actualizar cada valor de estado. (Olah, 2015).



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

Finalmente, tenemos que decidir qué vamos a generar. Este resultado se basará en nuestro estado de celda, pero será una versión filtrada. Primero, ejecutamos una capa la cual contiene una función de activación encargada de decidir qué partes del estado de celda generará la salida. Luego, colocamos el estado de la celda a través de la función tangente hiperbólica $(tanh)$ esto con la finalidad de obtener los valores que están entre -1 y 1. Luego multiplicando esto por la salida de la función de activación mencionada en primera instancia, de modo que solo generemos las partes que decidimos. (Olah, 2015).



$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

De esta forma se explica de una manera simple el funcionamiento paso a paso de la LSTM, sin embargo no todas las LSTM son iguales a las anteriores.

Capítulo 3

Modelos VAR

En este capítulo introduciremos el fundamento teórico de las Series de Tiempo, así como la explicación y el fundamento matemático del modelo Vectores Autorregresivos (VAR).

3.1. Series de Tiempo

Una serie de tiempo es el resultado de observar los valores de un proceso a lo largo del tiempo en intervalos regulares y ordenados cronológicamente (Peña, 2010).

Una serie de tiempo es un conjunto de observaciones x_t , cada una registrada a un tiempo específico t.

Las series de tiempo cuentan con tres componentes, las cuales son:

Tendencia: Se puede definir como un cambio a largo plazo que se produce en la relación al nivel medio, o el cambio a largo plazo de la media. La tendencia se identifica con un movimiento suave de la serie a largo plazo.

Estacional: Muchas series de tiempo presentan cierta periodicidad o dicho de otro modo, variación de cierto período (semestral, mensual, etc.) Estos efectos son fáciles de entender y se pueden medir explícitamente o incluso se pueden eliminar de la serie de datos, a este proceso se le llama desestacionalización de la serie.

Componente aleatoria: Esta componente no responde a ningún patrón de comportamiento, sino que es el resultado de factores fortuitos o aleatorios que inciden de forma aislada en una serie de tiempo.

De las tres componentes mencionadas anteriormente, la componente de tendencia y la componente estacional son componentes determinísticas, mientras que la última es aleatoria. De esta forma podemos escribir la serie de tiempo de la siguiente manera:

$$X_t = T_t + E_t + \epsilon_t.$$

Donde T_t indica la tendencia, E_t la componente estacional y por último ϵ_t la componente aleatoria.

El análisis de dichas series de tiempo tiene como objetivo final la predicción del valor de una variable a través del estudio y observación de lo ocurrido anteriormente.

Como ejemplos de series de tiempo podemos encontrar el valor del Bitcoin, Ripple, Dash, Litecoin, Ethereum.

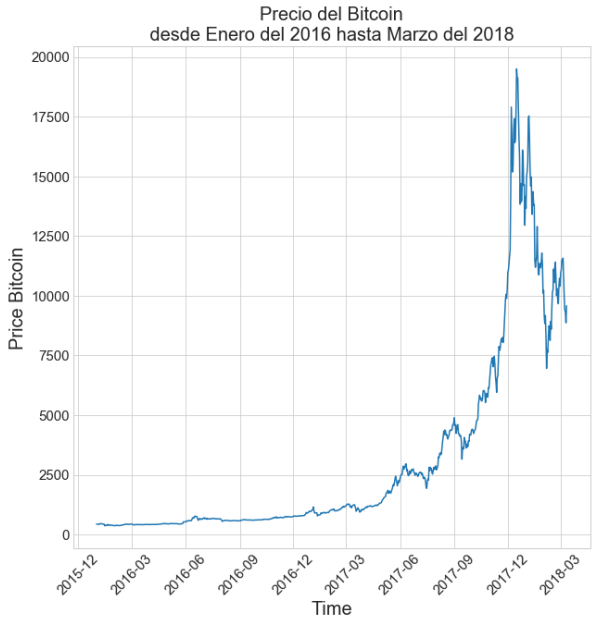


Figura 3.1: Precio del Bitcoin desde enero del 2016 hasta marzo del 2018.

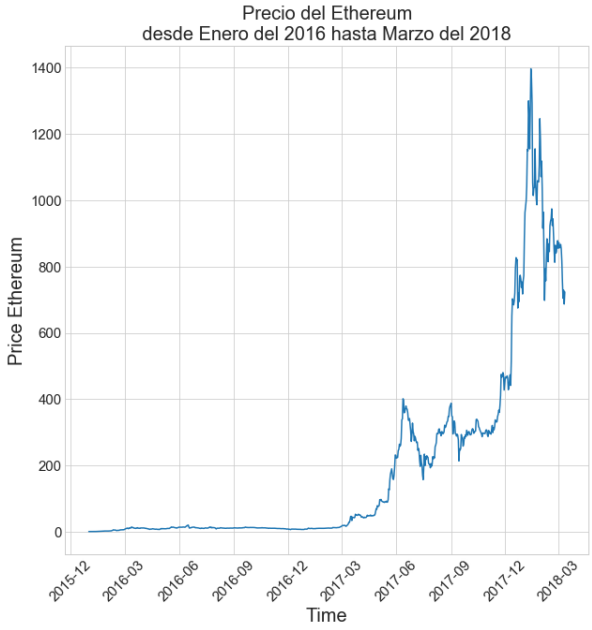


Figura 3.2: Precio del Ethereum desde enero del 2016 hasta marzo del 2018.

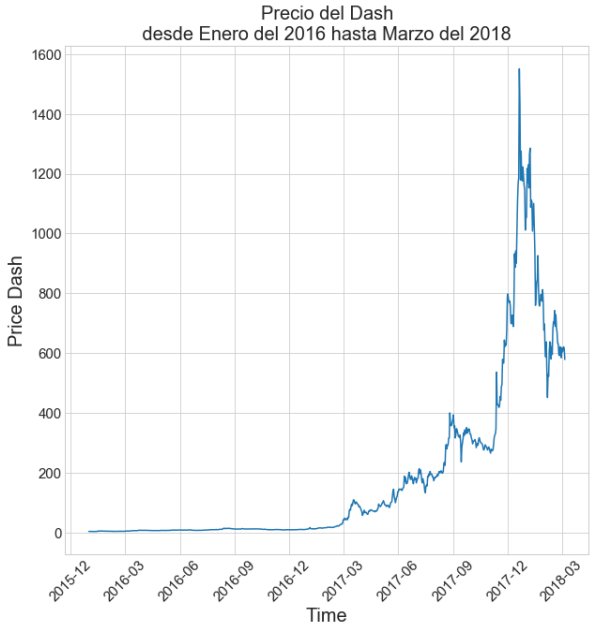


Figura 3.3: Precio del Dash desde enero del 2016 hasta marzo del 2018.

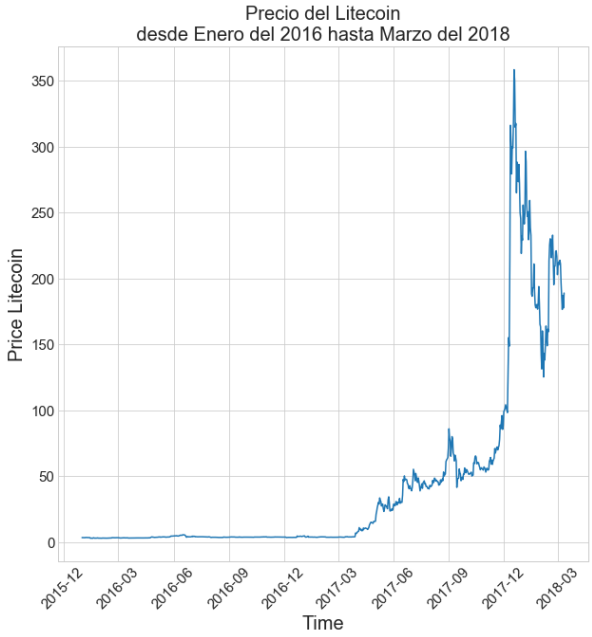


Figura 3.4: Precio del Litecoin desde enero del 2016 hasta marzo del 2018.

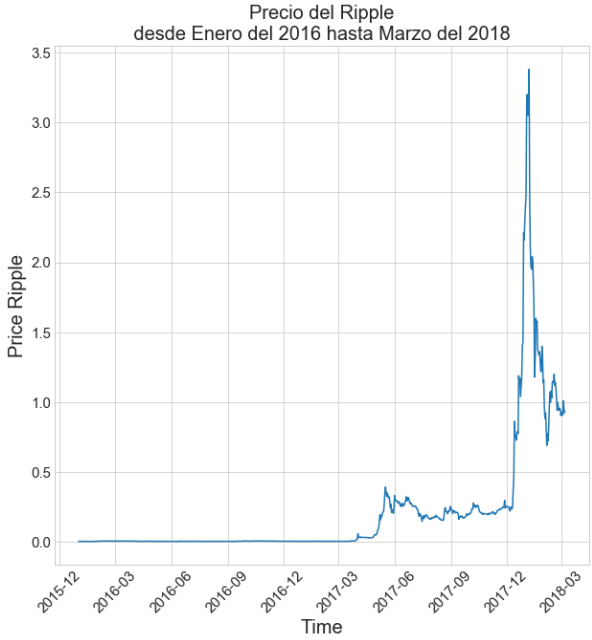


Figura 3.5: Precio del Ripple desde enero del 2016 hasta marzo del 2018.

Las anteriores cinco gráficas nos indican el valor de cada moneda respectivamente, desde el primero de enero del 2016 hasta el 12 de marzo del 2018. Observando detalladamente su comporportamiento las cinco se han comportando de una forma muy similiar. Siendo el Bitcoin la moneda más importante, en esta ocación los gráficos parecen indicar que el comportamiento del Bitcoin determina el comportamiento de las otras criptomonedas, sin embargo para hacer una afirmación más acertada haría falta realizar modelos de ajustes y pruebas de bondad de ajuste, lo cual no es el objetivo de este trabajo, sino determinar el comportamiento de cada una de ellas individualmente.

Las series de tiempo se pueden clasificar de la siguiente forma:

Series univariantes: son aquellas en las que se toma una única variable de observación como respuesta.

Series multivariantes: son aquellas, a diferencia de las univariantes, que toman más de una variable de respuesta por observación.

Series estacionarias: son aquellas cuyos valores de respuesta son estables en el tiempo alrededor de un nivel constante, sin mostrar una clara tendencia a crecer o a decrecer.

Series no estacionarias: son aquellas que pueden mostrar una tendencia, estacionalidad u otros efectos evolutivos a lo largo del tiempo.

Para el estudio se tiene una serie de tiempo multivariante en la cual utilizamos las variables Open, High y Low con la finalidad de predecir la variable Close de las respectivas monedas.

A continuación se da una breve introducción a los procesos autorregresivos.

Un proceso autorregresivo es aquel proceso estacionario cuya variable específica de salida depende de los valores que ha tomado la serie de tiempo en el pasado.

Los procesos autorregresivos tienen su base en el modelo de regresión simple, que explican la evolución de una variable y_t, como función lineal de otra variable x_t, mediante la ecuación:

$$y_t = c + bx_t + a_t,$$

donde c y b son constantes por determinar y a una variable aleatoria normal con media cero y varianza constante. Aplicando esta estructura de dependencia y sustituyendo $y_t = z_t$ y $x_t = z_{t-1}$ se obtiene el proceso autorregresivo de primer orden en el que el valor presente de la serie sólo depende de forma lineal del último valor observado (Peña, 2010).

Un modelo autoregresivo de orden p, abreviado AR(p), es de la forma

$$x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \ldots + \phi_p x_{t-p} + w_t,$$

donde x_t es estacionario, $\phi_1, \phi_2, \ldots, \phi_p$ son constantes $(\phi_p \neq 0)$. A menos que se declare lo contrario, se asume que w_t es un ruido blanco gaussiano de media cero y varianza σ_w^2. La media de x_t en la ecuación anterior es cero. Si la media μ de x_t no es cero, reemplazamos x_t por $(x_t - \mu)$, es decir

$$x_t - \mu = \phi_1(x_{t-1} - \mu) + \phi_2(x_{t-2} - \mu) + \ldots + \phi_p(x_{t-p} - \mu) + w_t,$$

o escribimos,

$$x_t = \alpha + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \ldots + \phi_p x_{t-p} + w_t,$$

donde, $\alpha = \mu(1 - \phi_1 + \phi_2 + \ldots + \phi_p)$.

3.2. Modelo VAR

De acuerdo con (Tsay, 2013) el modelo de series de tiempo multivariante más comúnmente utilizado es el modelo vector autorregresivo (VAR), particularmente en la econométrica por buenas razones. Primero el modelo es relativamente fácil de estimar. Uno puede usar el método de mínimos cuadrados (siglas en inglés LS), método de máxima verosimilitud (ML), o método bayesiano. Los tres métodos de estimación tienen soluciones de forma cerrada. Para un modelo VAR, las estimaciones de mínimos cuadrados son asintóticamente equivalente a las estimaciones de ML y para los métodos de mínimos cuadrados ordinarios (MCO) las estimaciones son las mismas que las estimaciones de mínimos cuadrados generalizados (GLS). Finalmente, Los

modelos VAR son similares a las regresiones lineales múltiples multivariantes ampliamente utilizadas en análisis estadístico multivariable. La serie de tiempo multivariante y_t sigue un modelo VAR de orden p, VAR(p), si

$$y_t = \sum_{i=1}^{p} A_i y_{t-i} + u_t,$$

donde A_i son matrices de la forma $(K \times K)$ y K es el número de ecuaciones endógenas que tiene el problema, para $i = 1, \ldots, p$ y u_t es un proceso K-dimensional con $E(u_t) = 0$ y matriz de covarianza definida positiva invariante en el tiempo $E(u_t * u_t^T) = \sum u$ (ruido blanco) (Paff, 2008).

Se define como ruido blanco a una sucesión de variables aleatorias no-correlacionadas, u_t con media 0 y varianza σ_u^2 .

Una característica importante de un proceso VAR (p) es su estabilidad. Esto significa que genera una serie de tiempo estacionaria con media invariante en el tiempo, varianzas y estructura de covarianza, dando suficientes valores iniciales. Uno puede verificar esto evaluando el polinomio característico:

$$det(I_K - A_i z - \ldots - A_p z^p) \neq 0 \;\; para \;\; |z| \leq 1.$$

Si la solución de la ecuación anterior tiene una raíz para $z = 1$, entonces algunas o todas las variables en el proceso VAR(p) están integradas de orden uno, es decir, $I(1)$. Puede ser el caso, que la cointegración entre las variables sí existe (Paff, 2008).

Se define la cointegración cuando existe una relación fuerte a largo plazo entre las variables. Es decir, que dos variables estén cointegradas implica que aunque crezcan a lo largo del tiempo, lo hacen de forma sincronizada. Mantienen dicha relación a lo largo del tiempo .

En la práctica, la estabilidad de un proceso empírico VAR (p) puede analizarse considerando el cálculo de los autovalores de la matriz de coeficientes. Un proceso VAR(p) puede escribirse como un proceso VAR(1) de la siguiente forma:

$$\xi_t = A\xi_{t-1} + v_t,$$

donde :

$$\xi_t = \begin{bmatrix} y_t \\ : \\ : \\ y_{t-p+1} \end{bmatrix}, A = \begin{bmatrix} A_1 & A_2 & \cdots & A_{p-1} & A_p \\ I & 0 & \cdots & 0 & 0 \\ 0 & I & \cdots & 0 & 0 \\ : & : & \ddots & : & : \\ 0 & 0 & \cdots & I & 0 \end{bmatrix}, v_t = \begin{bmatrix} u_t \\ : \\ : \\ 0 \end{bmatrix},$$

por lo que las dimensiones de los vectores ξ_t y v_t son $(Kp \times 1)$ y la dimensión de la matriz A es $(Kp \times Kp)$. Si el módulo de los autovalores de A son menores que uno, entonces el proceso VAR(p) es estable.

Para una muestra dada de las variables endógenas $y_1, \cdots, y_T$ y suficientes valores de pre-muestreo $y_{-p+1}, \cdots, y_0$, los coeficientes de un proceso VAR(p) se pueden estimar eficientemente mediante mínimos cuadrados aplicado por separado a cada una de las ecuaciones.

Una vez que se ha estimado un modelo VAR(p), el camino está abierta para su posterior análisis. Un investigador podría o debería estar interesado en las pruebas de diagnóstico, como las pruebas de ausencia de autocorrelación, heterocedasticidad o no normalidad en el proceso de error. Él podría estar interesado aún más en la inferencia causal, pronosticar y / o diagnosticar el comportamiento dinámico del modelo empírico, es decir, funciones de respuesta de impulso conocido como (IRF por sus siglas en inglés impulse response functions) y pronosticar la descomposición de la varianza del error (FEVD las siglas indican en inglés forecast error variance decomposition). Los dos últimos se basan en la descomposición media móvil de Wold para procesos VAR(p) estables que se define como:

$$y_t = \phi_0 u_t + \phi_1 u_{t-1} + \phi_2 u_{t-2} + \ldots,$$

donde $\phi_0 = I_K$ y ϕ_s pueden calcularse recursivamente de acuerdo con:

$$\phi_s = \sum_{j=1}^{s} \phi_{s-j} A_j \quad para \quad s = 1, 2, \ldots,$$

donde $A_j = 0$ para $j > p$.

Finalmente, las predicciones para $h \geq 1$ de un proceso empírico VAR(p) pueden generarse recursivamente de acuerdo con:

$$y_{T+h|T} = A_1 y_{T+h-1|T} + \ldots + A_p y_{T+h-p|T},$$

donde $y_{T+j|T} = y_{T+j}$ para $j \leq 0$. La matriz de covarianza de error de pronóstico se da como:

$$Cov\left(\begin{bmatrix} y_{T+1} - y_{T+1|T} \\ \vdots \\ \vdots \\ y_{T+h} - y_{T+h|T} \end{bmatrix}\right) = \begin{bmatrix} I & 0 & \cdots & 0 \\ \phi_1 & I & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ \phi_{h-1} & \phi_{h-2} & \cdots & I \end{bmatrix} (\sum u \otimes I_h) \begin{bmatrix} I & 0 & \cdots & 0 \\ \phi_1 & I & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ \phi_{h-1} & \phi_{h-2} & \cdots & I \end{bmatrix}^T$$

y las matrices ϕ_i son las matrices de coeficientes empíricos de la representación del promedio móvil de Wold de un proceso VAR(p) estable como se muestra arriba. El operador $\otimes$ es el producto Kronecker, el cual se define de la siguiente forma:

Dadas $A = (a_{ij}) \in \mathbb{K}^{m \times n}$ y $B = (b_{ij}) \in \mathbb{K}^{p \times q}$, el producto Kronecker se define como:

$$A \otimes B = (a_{ij}B)_{ij},$$

donde cada bloque $(a_{ij}B)$ esta en $\mathbb{K}^{p \times q}$ y $A \otimes B \in \mathbb{K}^{np \times mq}$. Este producto se realiza entre matrices de cualquier orden y no es conmutativo.

Capítulo 4

Análisis y Resultados

A continuación realizamos el análisis e implementación de los tres modelos explicados en los capítulos previos. El análisis se realiza por moneda, explicando cada modelo para dicha moneda. El ajuste de cada modelo se realizó de forma empírica ya que se ejecutó mediante ensayo y error con la finalidad de encontrar el mejor ajuste para cada modelo de acuerdo con las métricas mencionadas en los capítulos anteriores.

Las monedas con las que se realizó el presente estudio son las siguientes: Bitcoin, Dash, Ethereum, Litecoin y Ripple. A su vez, recordando que en el capítulo 2 se muestra una imagen de como están estructurados los datos (véase la Figura 2.1.) Además, se recapitula la finalidad con la que se implementaron los tres modelos. La red neuronal MLP, se usó con el fin de predecir si el precio de la moneda disminuye o aumenta al el día siguiente. Luego, la red neuronal LSTM se utilizó para pronosticar los valores del precio de la moneda en periodos de tiempo específicos. Por último, el modelo VAR se empleó con la finalidad de predecir valores futuros del precio de las monedas.

Se dará inicio con la primera moneda.

Bitcoin

Primer modelo, la red neuronal MLP. Esta red neuronal consta de 3 capas ocultas, la primera capa está conformada por 1000 perceptrones, la segunda con 100 y la última con 10 perceptrones. A su vez con el parámetro de regularización α con un valor de 0.01 y se utilizó como función de activación la función logistic.

De esta forma se inserta el código del ajuste de la red.

```
In [380]: from sklearn.metrics import confusion_matrix
          clf = MLPClassifier(hidden_layer_sizes = [1000,100,10], alpha = 0.01,
                              activation = 'logistic',
                              random_state = 0, solver='lbfgs').fit(X_train, y_train)
          clf_predicted = clf.predict(X_test)
          confusion = confusion_matrix(y_test, clf_predicted)
```

Con el ajuste de esta red se obtuvieron los siguientes resultados, tomando en cuenta la matriz de confusión y sus respectivas métricas.

Matriz de confusión

```
Out[382]: array([[ 80,   8],
                 [ 11, 102]], dtype=int64)
```

Resultado de la evaluación del modelo

```
Accuracy: 0.91
Precision: 0.93
Recall: 0.90
F1: 0.91
```

De esta forma se logró obtener una exactitud del 91 % lo que indica resultados positivos, ya que se tiene una alta probabilidad que el modelo clasifique de forma correcta, recordando que es una clasificación binaria la cual "0" indica que el precio al día siguiente disminuye y "1" indica que el precio de la moneda aumenta para el día siguiente.

Ahora bien, para el analísis del Recall, Precision y F1-Score se hizo enfoque en la siguiente tabla

	precision	recall	f1-score	support
0	0.88	0.91	0.89	88
1	0.93	0.90	0.91	113
avg / total	0.91	0.91	0.91	201

Esta tabla nos indica como fue medido el modelo por clases, es decir, "0" y "1". Para Precision obtenemos que nuestro modelo clasifica con 93 % de acierto la clase "1", esto quiere decir que clasifica bastante bien cuando el precio de la moneda al día siguiente va a aumentar y con un 88 % cuando el precio de la moneda va a disminuir. Sin embargo, en promedio se obtiene un 91 % de acierto para Precision. Luego para Recall se obtiene que la clase 0 tiene un 91 % de acierto y la clase 1 tiene un 90 %, a pesar de ello se obtiene un promedio para las dos clases del 91 %, lo que indica que sigue siendo un buen ajuste para el modelo. Por último F1-Score es el promedio de Precision con Recall por ende obtenemos igual un 91 % de acierto.

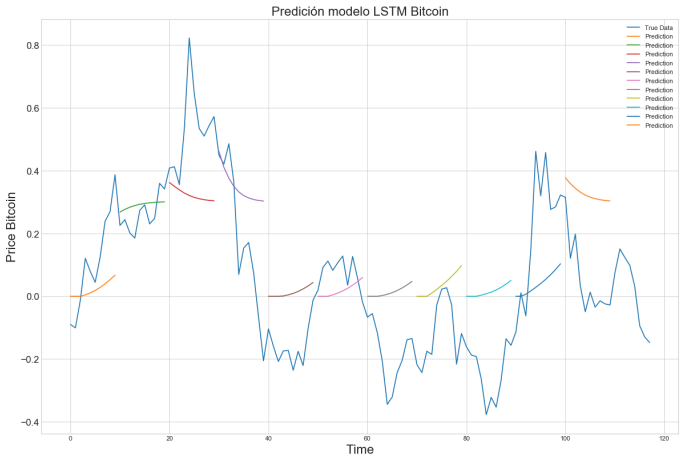
En general mediante este modelo de red neuronal MLP que se ajustó para predecir si el precio del Bitcoin aumenta o disminuye al día siguiente, se obtuvieron resultados positivos y se puede tomar en cuenta para futuras inversiones.

De acuerdo con el segundo modelo red neuronal LSTM, se realiza el análisis de los resultados obtenidos. Para realizar el ajuste de este modelo solo se tomó en cuenta la variable Close para cada una de las monedas.

Se ajustó una red neuronal con las siguientes características: La LSTM consta de cuatro capas. Una capa de entrada, luego se conecta con otra capa que consta de 10 neuronas que a su vez está conectada con otra capa de 100 neuronas y por último una capa de salida. Además se utilizó la función de activación "relu" y se entrenó con 100 Epoch, esto quiere decir que el modelo fue entrenado en 100 instancias.

Este modelo se utilizó con la finalidad de predecir la tendencia del comportamiento del Bitcoin específicamente, en un período de tiempo de cada 10 días.

A continuación se presenta la gráfica del resultado obtenido en dicho modelo



En la presente gráfica se observa el valor del precio del Bitcoin en los últimos cuatro meses, estos valores se normalizaron ya que el rango del precio de la moneda varía de gran forma, por ende se llevó el conjunto de datos a la misma escala. A su vez se tienen las predicciones que arrojó el modelo, las cuales son el precio del Bitcoin en periodos de tiempo de cada 10 días. Dichas predicciones son las rectas discontinuas de colores.

Como se aprecia en la gráfica no todas las predicciones coinciden con el comportamiento original de la serie, sin embargo, en algunos casos sí. Cabe destacar que el modelo obtuvo una función de pérdida para el entrenamiento de 0,0039 y para la validación de 0,0025. Con esto se afirma que para que un modelo esté ajustado correctamente y se puedan obtener resultados positivos, la función de pérdida para entrenamiento y validación a medida que transcurran las instancias tienen que converger a cero.

Este fue el mejor ajuste que se pudo conseguir, a pesar de ello no se obtuvieron los resultados deseados.

El último modelo en que se predijo el comportamiento del Bitcoin, es el modelo de serie de tiempo llamado VAR. En este modelo se utilizaron las variables Open, High y Low para predecir la variable Close.

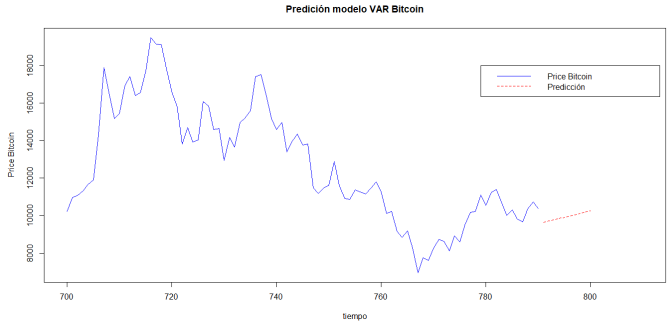
El proceso para ajustar el modelo se realizó de la siguiente forma.

Primero se aplicó la función logaritmo al conjunto de datos, de esta manera se redujo la escala de los valores para así poder encontrar exitosamente el p óptimo.

Segundo, se calculó el p óptimo para el modelo. Esto se hizo mediante la función conocida en el software estadístico R como VARselect. Esta arrojó como resultado el valor $p = 1$, lo cual indica que se tiene un modelo VAR(1), es decir, un modelo de orden uno.

Tercero, se evaluó que el p-valor del modelo sea menor que 0,05 y de esta forma se cumple, ya que el modelo arrojó un valor de $2{,}2 \times 10^{-16}$. A su vez, se obtuvo un error residual estándar de 0,0423 con 795 grados de libertad.

Por último se realizó la predicción para los últimos diez días y el resultado que se obtuvo se puede apreciar en la siguiente gráfica



En la gráfica se observa el precio del Bitcoin en los últimos tres meses y adicional a esto la predicción del modelo. En la predicción observamos que en los últimos diez días el Bitcoin tendría una tendencia a aumentar.

A continuación se presenta el análisis de los resultados de la segunda moneda.

Dash

Se inicia con la red neuronal MLP.

Esta red neuronal está compuesta por 3 capas ocultas. La primera capa está conformada por 1000 perceptrones, la segunda con 100 y la última con 10 perceptrones. En esta oportunidad las capas ocultas de la red coinciden con la red del Bitcoin, sin embargo no se ajustó la misma red debido a que no presentaba resultados positivos, por esa razón se modificó el parámetro de regularización α con un valor de 0,001 y se utilizó como función de activación la función *tanh*.

Seguidamente se presenta el código del ajuste de la red MLP para el Dash.

```
In [87]: from sklearn.metrics import confusion_matrix
         clf = MLPClassifier(hidden_layer_sizes = [1000,100,10], alpha = 0.001,
                             activation = 'tanh',
                             random_state = 0, solver='lbfgs').fit(X_train, y_train)
         clf_predicted = clf.predict(X_test)
         confusion = confusion_matrix(y_test, clf_predicted)
```

De acuerdo con los resultados obtenidos mediante la matriz de confusión y sus respectivas métricas se tiene lo siguiente.

Matriz de confusión

```
Out[89]: array([[81, 13],
                [13, 92]], dtype=int64)
```

Resultado de la evaluación del modelo

```
Accuracy: 0.87
Precision: 0.88
Recall: 0.88
F1: 0.88
```

De esta forma se logró obtener una exactitud del 87 %. Este porcentaje indica resultados positivos, con un mayor margen de error comparado con el ajuste que se le hizo al Bitcoin. De acuerdo con esto existe una alta probabilidad que el modelo clasifique de forma correcta, recordando nuevamente que es una clasificación binaria la cual "0" indica que el precio al día siguiente disminuye y "1" indica que el precio de la moneda aumenta para el día siguiente.

Por otra parte, para el analísis del Recall, Precision y F1-Score se hizo enfoque en la siguiente tabla

	precision	recall	f1-score	support
0	0.86	0.86	0.86	94
1	0.88	0.88	0.88	105
avg / total	0.87	0.87	0.87	199

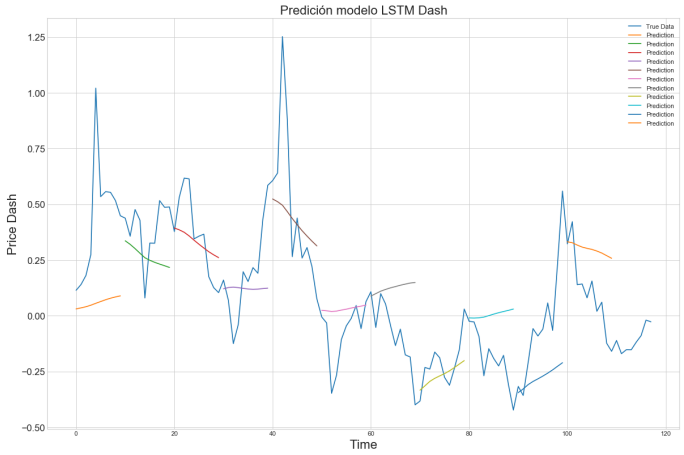
Esta tabla indica como fue medido el modelo por clases, es decir, “0“ y “1“. En cuanto a Precision el modelo clasifica la clase “1“ con un 88 % de acierto, esto determina que el modelo clasifica bien cuando el precio de la moneda al día siguiente va a aumentar y con un 86 % cuando el precio de la moneda va a disminuir. Ahora bien, se obtiene un 87 % de acierto para Precision en promedio. Luego para Recall se obtuvieron los mismos resultados que para Presicion, 88 % para la clase “1“ y 86 % para la clase “0“. En promedio general se obtiene un 87 %. Por último, F1-Score es el promedio de Precision con Recall y por ende se obtiene de igual manera un 87 % de acierto. Cabe destacar que esta red neuronal clasifica mejor cuando el precio del Dash va a aumentar el día siguiente.

En general mediante este modelo de red neuronal MLP que se ajustó para predecir si el precio del Dash aumenta o disminuye al día siguiente, se obtuvieron óptimos resultados y se puede tomar en cuenta para predecir cuando aumenta el precio de la moneda y de esta forma poder realizar futuras inversiones.

Ahora se muestra el ajuste que se realizó para la red neuronal LSTM.

La LSTM consta de cuatro capas. Una capa de entrada, luego se conecta con otra capa que consta de 10 neuronas que a su vez está conectada con otra capa de 100 neuronas y por último una capa de salida. Además se utilizó la función de activación “tanh“ y se entrenó con 150 Epoch.

A continuación se presenta la gráfica del resultado obtenido por el ajuste de dicha red.



En la presente gráfica se observa el valor del precio del Dash en los últimos cuatro meses, estos valores se normalizaron ya que el rango del precio de la moneda varía de gran forma, por ende se llevó el conjunto de datos a la misma escala. A su vez se tienen las predicciones que arrojó la LSTM, las cuales son el precio del Dash en periodos de tiempo de cada 10 días. Dichas predicciones son las rectas discontinuas de colores.

En la gráfica que se presentó anteriormente se visualiza que no coinciden todas las predicciones con el comportamiento original de la serie, sin embargo, en algunos casos esto si ocurre. El modelo obtuvo una función de pérdida para el entrenamiento de 0,0080 y para la validación de 0,0026. Con esto se afirma que para que un modelo esté ajustado correctamente y se puedan obtener resultados positivos, la función de pérdida para entrenamiento y validación a medida que transcurran las instancias tienen que converger a cero. Si bien no se alcazaron los resultados deseados este fue el mejor ajuste que se logró conseguir.

Por otra parte, se muestra el modelo que se ajustó de serie de tiempo VAR. En este modelo como se mencionó anteriormente se utilizaron las variables Open, High y Low para predecir la variable Close.

El ajuste de este modelo se realizó siguiendo los siguientes pasos:

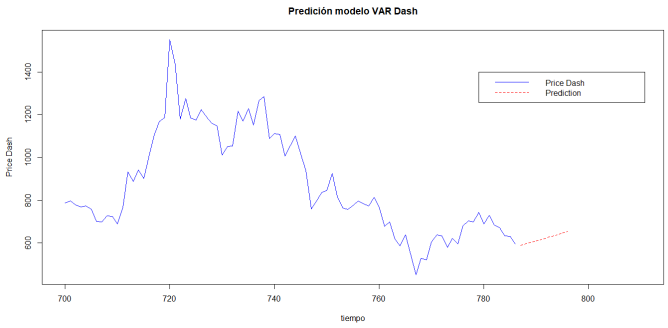
Primero se aplicó la función logaritmo a los datos, con la finalidad de reducir la escala de los valores, para así poder encontrar exitosamente el p óptimo.

Segundo, se determinó el p óptimo para el modelo. Esto se hizo mediante la función VARselect en R. Esta

arrojó como resultado el valor $p = 2$, lo que indica que se tiene un modelo VAR(2).

Tercero, se evaluó que el p-valor del modelo sea menor que 0,05 y de esta forma se cumple, ya que el modelo arrojó un valor de $2{,}2 \times 10^{-16}$. A su vez, se obtuvo un error residual estándar de 0,06426 con 785 grados de libertad.

Por último, se realizó la predicción para los últimos diez días y el resultado que se obtuvo se puede apreciar en la siguiente gráfica



En la gráfica se observa el precio del Dash en los últimos tres meses y adicional a esto la predicción del modelo. En la predicción observamos que en los últimos diez días el Dash tendría una ligera tendencia a aumentar.

Ahora bien, se procede a realizar el análisis de la tercera moneda.

Ethereum

Siguiendo el formato se inicia con la red neuronal MLP, en la cual se muestra como está estructurada la red.

Esta red neuronal está constituida de 3 capas ocultas, las cuales estan conformadas por 100 perceptrones cada una. El parámetro de regularización α con un valor de 0,1 y se utilizó como función de activación la función tanh.

De esta forma se aprecia el código del ajuste de la red.

```
In [119]: from sklearn.metrics import confusion_matrix
          clf = MLPClassifier(hidden_layer_sizes = [100,100,100], alpha = 0.1,
                              activation = 'tanh',
                              random_state = 0, solver='lbfgs').fit(X_train, y_train)
          clf_predicted = clf.predict(X_test)
          confusion = confusion_matrix(y_test, clf_predicted)
```

A continuación se presenta los resultados de la matriz de confusión para el presente modelo.

Matriz de confusión

```
Out[121]: array([[ 75,   8],
                 [ 13, 105]], dtype=int64)
```

Resultados de la evaluación del modelo

```
Accuracy: 0.90
Precision: 0.93
Recall: 0.89
F1: 0.91
```

Ahora bien, se logró obtener una exactitud del 90 %. Lo que indica resultados positivos, ya que se tiene una alta probabilidad de que el modelo clasifique de forma correcta, recordando al igual que en los dos casos anteriores que es una clasificación binaria.

Por otra parte, para el análisis del Recall, Precision y F1-score se hizo énfasis en la siguiente tabla.

	precision	recall	f1-score	support
0	0.85	0.90	0.88	83
1	0.93	0.89	0.91	118
avg / total	0.90	0.90	0.90	201

Para Precision obtenemos que nuestro modelo clasifica con 93 % de acierto la clase "1", esto quiere decir que predice eficazmente cuando el precio de la moneda va a aumentar al día siguiente y con un 85 % cuando el

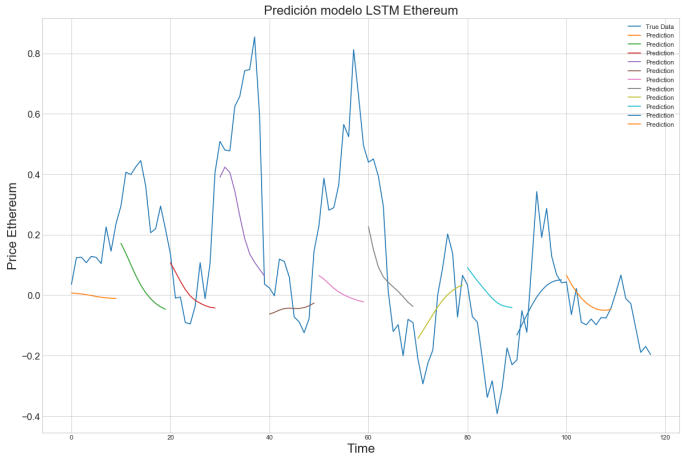
precio de la moneda va a disminuir. En promedio se obtiene un 90 % de acierto para Precision. Para Recall se obtiene que la clase 0 tiene un 90 % de acierto y la clase 1 tiene un 89 %. De acuerdo con esto existe un promedio del 90 % para ambas clases. Este porcentage indica que el modelo esta siendo ajustado correctamente. Por último, F1-Score tiene un 90 % de acierto.

De acuerdo con el objetivo de la investigación que consistió en predecir si el precio de la moneda disminuye o aumenta al día siguiente, para el Ethereum se obtuvieron resultados positivos y se puede tomar en cuenta para futuras inversiones.

A continuación se muestra el ajuste que se realizó para la red neuronal LSTM.

La LSTM consta de cuatro capas. Una capa de entrada, otra capa que consta de 10 neuronas, otra capa de 100 neuronas y por último una capa de salida. Además se utilizó la función de activación "linear" y se entrenó con 150 Epoch.

A continuacón se presenta la gráfica del resultado obtenido por el ajuste de la red.



En la presente gráfica se observa el valor del precio del Ethereum en los últimos cuatro meses, estos valores se normalizaron ya que el rango del precio de la moneda varía considerablemente, por ende se llevó el conjunto de datos a la misma escala. A su vez se tienen las predicciones que arrojó el modelo, las cuales son el precio del Ethereum en periodos de tiempo de cada 10 días. Dichas predicciones son las líneas discontinuas de colores.

Como se aprecia en la gráfica no todas las predicciones coinciden con el comportamiento original de la serie, sin embargo, para cuando el precio de la moneda disminuye si se comporta igual. Cabe destacar que el modelo obtuvo una función de pérdida para el entrenamiento de 0,0106 y para la validación de 0,0087. Recordando que se necesita que la función de pérdida para entrenamiento y validación converjan a cero para alcanzar un buen ajuste. Este fue el mejor ajuste que se logró conseguir, sin embargo, con respecto a las otras dos redes LSTM ésta ha sido la que ha tenido menor rendimiento con respecto a la función de pérdida en el entrenamiento .

Culminando el análisis para el Ethereum se muestra el modelo VAR.

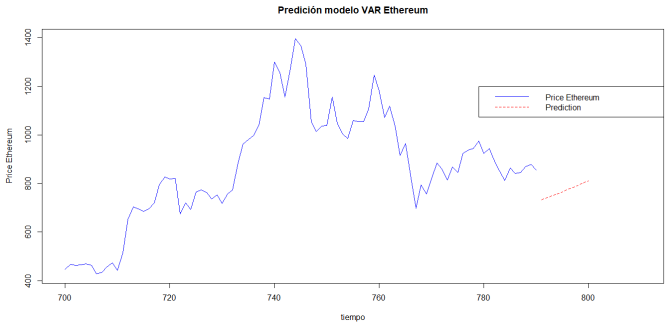
El ajuste de este modelo se realizó siguiendo el procedimiento que se muestra a continuación.

Se aplicó la función logaritmo al conjunto de datos, de esta manera se redujo la escala de los valores para así poder encontrar exitosamente el p óptimo.

Luego, se estimó el p óptimo para el modelo. Esto se hizo mediante la función conocida en el software R como VARselect. Esta arrojó como resultado el valor $p = 1$, lo cual indica que se tiene un modelo de orden uno.

Posteriormente, se evaluó que el p-valor del modelo sea menor que 0,05 y de esta forma se cumple, ya que el modelo arrojó un valor de $2{,}2 \times 10^{-16}$. A su vez, se obtuvo un error residual estándar de 0,06912 con 795 grados de libertad.

Para concluir se realizó la predicción para los últimos diez días y el resultado que se obtuvo se puede apreciar en la siguiente gráfica



En la gráfica se observa el precio del Ethereum en los últimos tres meses y adicional a esto la predicción del modelo. En la predicción observamos que en los últimos diez días el Ethereum tendría una tendencia a aumentar y el precio estaría alrededor de los $700 y $800.

Se procede con la cuarta moneda.

Litecoin

Siguiendo el esquema se comienza el análisis con la red MLP.

Esta red neuronal consta de 3 capas ocultas, la primera y segunda capa está conformada por 100 perceptrones, y la última con 10 perceptrones. A su vez, con el parámetro de regularización α con un valor de 0,001 y como función de activación se utilizó la función tanh. Además se entrenó la red con el conjunto de datos normalizados, dicha normalización se llevó a cabo mediante la función MinMaxScaler en Python.

De esta manera se muestra el código del ajuste de la red.

```
In [82]: from sklearn.metrics import confusion_matrix
         clf = MLPClassifier(hidden_layer_sizes = [100,100,10], alpha = 0.001,
                             activation = 'tanh',
                             random_state = 0, solver='lbfgs').fit(X_train_scaled, y_train)
         clf_predicted = clf.predict(X_test_scaled)
         confusion = confusion_matrix(y_test, clf_predicted)
```

Tomando en cuenta la matriz de confusión y sus respectivas métricas se observan los siguientes resultados.

Matriz de confusión

```
Out[84]: array([[ 73,  12],
                [ 12, 104]], dtype=int64)
```

Resultado de la evaluación del modelo

```
Accuracy: 0.88
Precision: 0.90
Recall: 0.90
F1: 0.90
```

Se obtuvó una exactitud del 88 % lo cual indica resultados satisfactorios, ya que garantiza que el modelo clasifique de forma correcta, teniendo en cuenta que es una clasificación binaria.

Para realizar el análisis se tomó en cuenta la siguiente tabla con las métricas Recall, Precision y F1-score.

	precision	recall	f1-score	support
0	0.86	0.86	0.86	85
1	0.90	0.90	0.90	116
avg / total	0.88	0.88	0.88	201

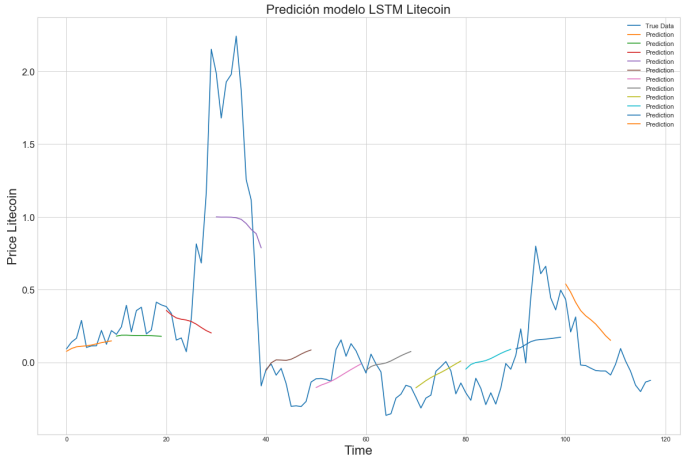
Para Precision obtenemos un 90 % de acierto para la clase “1“, este porcentage indica que el modelo clasifica bastante bien cuando el precio de la moneda al día siguiente va a aumentar y con un 86 % cuando el precio de la moneda va a disminuir. En promedio se obtiene un 88 % de acierto para Precision. Luego para Recall la clase “0 “ tiene un 86 % de acierto y la clase “1“ tiene un 90 %, en promedio se obtiene un 88 % para las dos clases, lo que indica que sigue siendo un buen ajuste para el modelo. Por último, para F1-Score se tiene un 88 % de acierto.

Para predecir si el precio del Litecoin aumenta o disminuye al día siguiente se ajustó el modelo red neuronal MLP y se obtuvieron resultados positivos, garantizando que puede ser utilizado para criterios de decisión en futuras inversiones ya que tiene un pequeño margen de error.

De acuerdo con el segundo modelo red neuronal LSTM, se realiza el análisis de los resultados obtenidos.

La LSTM está compuesta por cuatro capas. Una capa de entrada,una capa de 10 neuronas, otra capa de 100 neuronas y por último una capa de salida. Además se utilizó la función de activación "tanh" y se entrenó con 150 Epoch.

En la presente gráfica se muestran los resultados obtenidos de dicho modelo.



En la presente gráfica se observa el valor del precio del Litecoin en los últimos cuatro meses, estos valores se normalizaron ya que el rango del precio de la moneda varía de gran forma, por ende se llevó el conjunto de datos a la misma escala. A su vez se tienen las predicciones que arrojó el ajuste de la red LSTM, las cuales son el precio del Litecoin en periodos de tiempo de cada 10 días. Dichas predicciones son las rectas discontinuas de colores.

De acuerdo a lo expuesto anteriormente no todas las predicciones coinciden con el comportamiento original de la serie, en esta ocasión este fue uno de los ajustes mas complicados debido al comportamiento irregular de la moneda, sin embargo, en algunos casos sí se logra observar que la predicción y la serie coinciden. Cabe destacar que el modelo obtuvo una función de pérdida para el entrenamiento de 0,0072 y para la validación de 0,0028. Con esto se afirma que para que un modelo esté ajustado correctamente y se puedan obtener resultados satisfactorios, la función de pérdida para entrenamiento y validación a medida que transcurran las instancias tienen que converger a cero.

El último modelo que se le ajustó al Litecoin fue el modelo de serie de tiempo VAR.

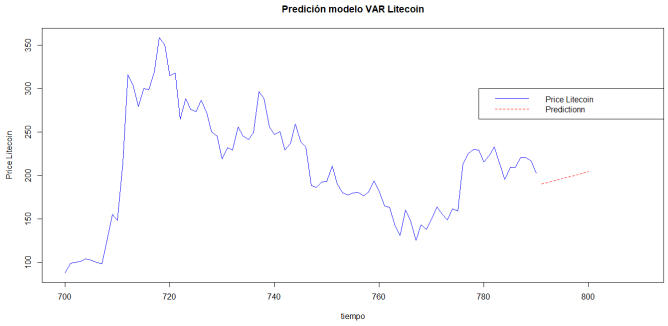
El ajuste de este modelo se realizó mediante un procedimiento que consta de cuatro pasos.

Paso uno, se aplicó la función logaritmo a los datos. Aplicando esta función se reduce la escala de los valores para así poder encontrar exitosamente el p óptimo.

Paso dos, se estimó el p óptimo para el modelo. Esto se hizo con la función VARselect en R. El resultado fue $p = 1$, lo cual indica que se tiene un modelo de orden 1.

Paso tres, se evaluó que el p-valor del modelo sea menor que 0,05 y de esta forma se cumple, ya que el modelo arrojó un valor de $2{,}2 \times 10^{-16}$. A su vez, se obtuvo un error residual estándar de 0,06224 con 795 grados de libertad.

Como paso final se realizó la predicción para los últimos diez días y el resultado que se obtuvo se puede apreciar en la siguiente gráfica



En la gráfica se observa el precio del Litecoin en los últimos tres meses y adicional a esto la predicción del modelo. En la predicción observamos que en los últimos diez días el Litecoin tendría una tendencia a aumentar y con un valor entre los $200 y $250.

Para concluir con el análisis de los resultados, se procede con última moneda.

Ripple

Para el modelo de MLP. Se ajustó una red neuronal conformada por 3 capas ocultas, la primera capa consta de 100 perceptrones, la segunda y la tercera de 10 perceptrones. A su vez, con el parámetro de regularización α con un valor de 0,0001 y como función de activación, la función tanh.

A continuación se muestra el ajuste de la red.

```
In [52]: from sklearn.metrics import confusion_matrix
         clf = MLPClassifier(hidden_layer_sizes = [100,10,10], alpha = 0.0001,
                             activation = 'tanh',
                             random_state = 0, solver='lbfgs').fit(X_train, y_train)
         clf_predicted = clf.predict(X_test)
         confusion = confusion_matrix(y_test, clf_predicted)
```

Matriz de confusión

```
Out[54]: array([[94, 13],
                [31, 61]], dtype=int64)
```

Resultado de la evaluación del modelo

```
Accuracy: 0.78
Precision: 0.82
Recall: 0.66
F1: 0.73
```

De esta forma se logró obtener una exactitud del 78 %, lo que indica resultados aceptables, sin embargo, comparando con las otras cuatro monedas este fue el ajuste más deficiente.

Ahora bien, para el analísis del Recall, Precision y F1-Score se hizo enfoque en la siguinete tabla

	precision	recall	f1-score	support
0	0.75	0.88	0.81	107
1	0.82	0.66	0.73	92
avg / total	0.79	0.78	0.78	199

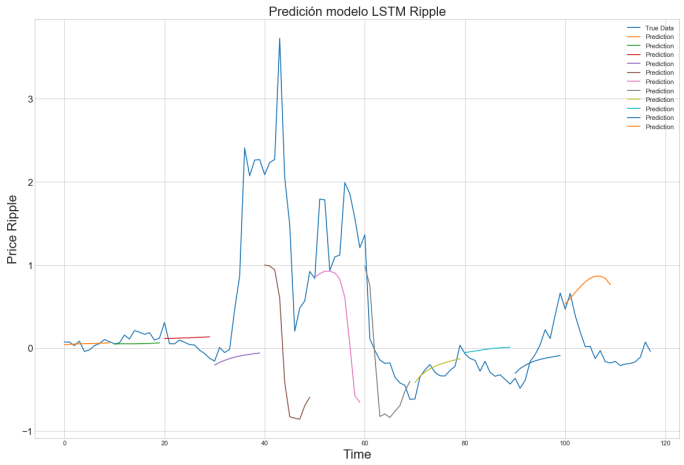
En este modelo, para Precision se obtiene un acierto del 82 % para la clase “1“ y un 75 % para la clase “0“. El promedio para Precision fue de 79 %. Para Recall se obtuvo un 88 % de acierto para la clase “0“ y un 66 % para la clase “1“. Esto indica que el modelo predice mejor cuando el precio de la moneda va a disminuir. El promedio para las dos clases fue del 78 %. Por último, F1-Score obtuvo un 78 % de acierto.

Mediante este modelo de red neuronal MLP que se ajustó para predecir si el precio del Ripple aumenta o disminuye al día siguiente, no fueron alcanzados los resultados deseados. Esto se debe a que no se cubrieron las espectativas en comparación con los modelos anteriores.

De acuerdo con el segundo modelo red neuronal LSTM, se realiza el análisis de los resultados obtenidos.

La LSTM consta de cuatro capas. Una capa de entrada, luego se conecta con otra capa que consta de 10 neuronas que a su vez está conectada con otra capa de 100 neuronas y por último una capa de salida. Además se utilizó la función de activación “tanh“ y se entrenó con 100 Epoch.

A continuación se presenta la gráfica del resultado obtenido en dicho modelo



En la presente gráfica se observa el valor del precio del Ripple en los últimos cuatro meses, estos valores se normalizaron ya que el rango del precio de la moneda varía considerablemente, por ende se llevó el conjunto de datos a la misma escala. A su vez se tienen las predicciones que arrojó el modelo, las cuales son el precio del Ripple en periodos de tiempo de cada 10 días. Dichas predicciones son las líneas discontinuas de colores.

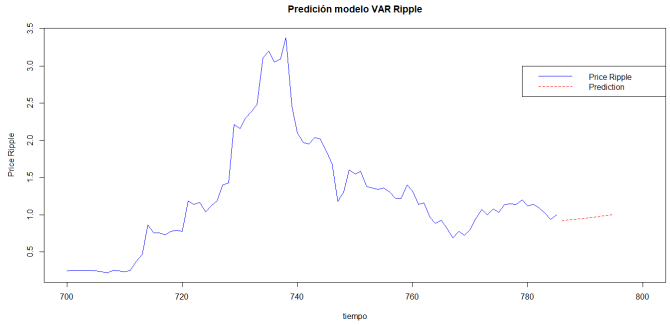
Como se aprecia en la gráfica no todas las predicciones coinciden con el comportamiento original de la

serie. Sin embargo, cuando ocurre una fuerte caida del precio el modelo predice un comportamiento similar. Cabe destacar que el modelo obtuvo una función de pérdida para el entrenamiento de 0,0877 y para la validación de 0,1412.

Por último, se ajustó el modelo de serie de tiempo VAR para el Ripple.

Como se ha mencionado en el análisis de las monedas, el primer lugar se aplicó la función logaritmo para reducir la escala de valores. Luego mediante la función VARselect se calculó el p optimo, arrojando un valor de $p = 2$. Seguido de esto, se analizó que el p-valor del modelo fuera menor que 0,05 y en este caso se cumple debido a que el valor obtenido fue de $2{,}2 \times 10^{-16}$ con un error residual estándar de 0,08171 con 785 grados de libertad.

Para finalizar, se realizó la predicción para los últimos diez días y el resultado que se obtuvo se puede apreciar en la siguiente gráfica



En la gráfica se observa el precio del Ripple en los últimos tres meses y adicional a esto la predicción del modelo. En la predicción observamos que en los últimos diez días el Ripple no tiene una tendencia marcada y el precio estaría al rededor de $1.

Capítulo 5

Conclusiones

De acuerdo con el objetivo propuesto se obtuvieron resultados positivos con la implementación de la red neuronal MLP, con la finalidad de predecir si el precio de la moneda aumentaba o disminuía al día siguiente. El modelo que tuvo mejores resultados fue el implementado para el Bitcoin, de esta forma éste será pilar fundamental para realizar futuras inversiones, debido a que esta moneda es altamente cotizada en la actualidad. Por otra parte, el segundo modelo a tomar en cuenta fue el ajuste que se realizó al Ethereum, debido a que se obtuvieron resultados positivos, a su vez, el Ethereum es la segunda moneda tomada en cuenta en el mercado de Criptomonedas. Tercero se encuentra el ajuste que se realizó al Litecoin, con este modelo se obtuvieron muy buenos resultados y con un alto nivel de confianza para invertir. Como últimas monedas encontramos el Dash y Ripple. Para el Dash se lograron resultados aceptables, sin embargo para el Ripple no, por lo que esta moneda no será tomada en cuenta para realizar futuras inversiones.

Ahora bien, para la red neuronal LSTM no se alcanzaron los resultados deseados. Uno de los factores que influyó en este proceso fue que se trabajó con una baja cantidad de datos, esto se debe a que se utilizó un periodo de tiempo igual para las cinco monedas donde sus datos no tuviesen valores faltantes.

Por último, para el modelo de serie de tiempo VAR se obtuvieron resultados positivos, sin embargo, en cuatro de las cinco monedas la predicción del modelo fue que el precio aumentaría, haciendo un contraste con los otros modelos y con el valor real esto no ocurrió de la misma forma.

Todos estos procesos dejan como enseñanza que aunque se consigan excelentes resultados mediante un

modelo, puede ocurrir algo en la cotidianidad que genere un cambio drástico en este tipo de mercados. Es importante destacar que en esta oportunidad el modelo no está diseñado para predecir ese tipo de irregularidades ni abarcar los factores externos que se puedan presentar.

Como futuras recomendaciones se propone entrenar los modelos de redes neuronales con grandes cantidades de datos para así obtener mejores resultados. Por otra parte, implementar otros modelos de series de tiempo involucrados en el área de la econometría como los modelos ARIMA o GARCH.

Bibliografía

[1] Bashir, I. (2017). *Mastering Blockchain.* Packt Publishing Ltd.

[2] Brockwell, P.J. & Davis, R.A. (1996). *Introduction to Time Series and Forecasting*., Springer.

[3] Georges, J. (2017). *La cadena de bloques (blockchain). Una tecnología disruptiva con el poder de revolucionar el sector financiero.*, EquiSoft.

[4] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* MIT Press.

[5] Gupta, M. (2017) *Blockchain for dummies.*. John Wiley and Sons, Inc.

[6] Olah, C. (2015). *Understanding LSTM Networks.*, Obtenido de http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[7] Ordinas, M. (2017). *Criptomonedas. ¿Oportunidad o Burbújas? Informe mensual de estratégias de BancaMarch.*

[8] Paff, B. (2008) *VAR, SVAR and SVEC Models: Implementation Within R Package vars.* Kronberg im Taunus: URL http://www.jstatsoft.org/v27/i04/.

[9] Peña, D. (2010). *Análisis de series temporales.* Alianza Editorial.

[10] Picón Viana, C. (2011). *Revista de economía del caribe n.8, 45-79.* Artículo de investigación Mercado Internacional.

[11] Sánchez Gil, A. & Terán Varela, O. E. (2018). *Criptomonedas como oportunidad de negocio de empresas del Sector Turístico en la Zona Sur Oriente Del estado de México* . Revista global de negocios v.6 (1) , 93-104.

[12] Shumway, R.H & Stoffer, D.S. (2006) *Time Series Analysis and Its Applications with R examples.* 2nd edition. Springer.

[13] Tsay, R. S. (2013). *Multivariate time series analysis with r and financial applications.* Chicago: John Wiley and Sons Inc.