UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

Redes neuronales y series de tiempo: Una aplicación en valores de activos financieros

Proyecto de Investigación

Kevin Ricardo Rojas Satián

Licenciatura en Matemáticas

Trabajo de titulación presentado como requisito para la obtención del título de Licenciado en Matemáticas

Quito, 21 de julio de 2018

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

HOJA DE CALIFICACIÓN DE TRABAJO DE TITULACIÓN

Redes neuronales y series de tiempo: Una aplicación en valores de activos financieros

Kevin Ricardo Rojas Satián

Calificación:

Nombre del profesor, Título académico Ulrich Schlickewei, Ph.D.

Firma del profesor: .....

Quito, 21 de julio de 2018

Derechos de Autor

Por medio del presente documento certifico que he leído todas las Políticas y Manuales de la Universidad San Francisco de Quito USFQ, incluyendo la Política de Propiedad Intelectual USFQ, y estoy de acuerdo con su contenido, por lo que los derechos de propiedad intelectual del presente trabajo quedan sujetos a lo dispuesto en esas Políticas.

Asimismo, autorizo a la USFQ para que realice la digitalización y publicación de este trabajo en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Firma del estudiante:	.....
Nombres y apellidos:	Kevin Ricardo Rojas Satián
Código:	00118750
Cédula de identidad:	1722658604
Lugar y fecha:	Quito, 21 de julio de 2018

AGRADECIMIENTOS

Es mi deseo agradecer principalmente a mi tutor Ulrich Schlickewei por todas las enseñas, tiempo, dedicación y comprensión que ha tenido conmigo durante toda la realización del presente proyecto.

De igual manera, quiero agradecer a mis padres César y Mercedes así como a mi hermano Konrad y mi prima Jacqueline por ser los pivotes de mi vida familiar y llenar cada uno de mis días con felicidad y esperanza.

Agradecer de todo corazón a David Hervas, John Skukalek, Carlos Jiménez, y Oihane Fernández por haberme formado profesionalmente.

Quisiera agradecer además a Alexis, Nicolás, Daniela, Roberto, Joel, Darío, Camila y Vanessa; por haber hecho de estos años la mejor experiencia universitaria posible. A Darío M. y Santiago B. por haber cambiado mi visión de la vida y recordarme que tengo que crecer cada día.

Finalmente, quiero agradecer a Andrea Calles por su apoyo y aliento incondicional sin quién no hubiese podido completar con éxito el presente documento.

Gracias a todos.

RESUMEN

Una red neuronal artificial es un modelo matemático que utiliza un sistema de capas internas y externas conectadas a través de unas estructuras denominadas neuronas y que en conjunto simulan la arquitectura de las conexiones entre neuronas dentro del cerebro humano. Una vez que estas redes neuronales atravesaron el proceso de aprendizaje sobre un conjunto de datos conocidos se convierten en algoritmos capaces de predecir, en un rango previamente establecido, el comportamiento que pudieran tener un conjunto del mismo tipo de datos de los cuales solo se conocen las etapas previas y no los desenlaces en el comportamiento. Este documento resume el contenido teórico, así como técnicas propias del aprendizaje supervisado a través de arquitecturas de deep-learning con el objetivo de obtener predicciones de precios de acciones considerando diferentes tipos de modelos a través del uso de herramientas computacionales especializadas para una posterior comparación entre los mismos.

Palabras Clave: Inteligencia artificial, aprendizaje profundo, redes neuronales, red completamente conectada, red neuronal recurrente, LSTM, acciones de empresa.

ABSTRACT

An artificial neural network is a mathematical model provided with a system of internal and external layers connected through a group of structures called neurons simulating the architecture of the connections between neurons within the human brain. Once these neural networks went through the learning process on a set of known data, they become algorithms capable of predicting, in a previously established range, the behavior that a same-type data set could perform by knowing only the previous stages. This document summarizes the theoretical content, as well as the techniques of supervised learning by using different deep-learning architectures with the aim of obtaining predictions of stock prices through the use of specialized computational tools for a later comparison between them.

Keywords: Artificial intelligence, deep-learning, neural networks, feedforward network, recurrent neural network, LSTM, stock prices.

TABLA DE CONTENIDOS

INDICE DE FIGURAS

TABLE **Page**

CAPITULO I: PANORAMA DE LA INTELIGENCIA ARTIFICIAL Y APLICACION EN FINANZAS

2.1 Introducción

A mediados del siglo XX, el desarrollo de las computadoras se encontraba en auge con el advenimiento de las primeras maquinas electro-mecánicas orientadas a la resolución de problemas. Los distintos aportes de estos artefactos han permitido que a día de hoy, los grupos de personas, pueden acceder a tecnologías cada vez mas útiles para el diario vivir de todos los seres humanos (Chollet & Allaire, 2018).

Muchas industrias en el último siglo han dependido de forma importante del desarrollo y avance de las computadoras, así como de sus distintas tecnologías y aplicaciones; por otro lado, otra compañías se han erguido tomando el desarrollo de las tecnologías como pivote en la oferta de un determinado servicio (Cockburn, Henderson, & Stern, 2017). La realidad es que hoy en día no puede concebirse un modelo de negocios que no tome en consideración a las computadoras y sus herramientas como elementos fundamentales para el desarrollo de productos y servicios, efecto que ha provocado una tremenda subida en el intercambio de bienes y servicios que día a día se lleva a cabo en todas las economías a nivel mundial (Górriz & Gargallo, 2004). En la figura 2.1 se evidencia como el GDP global ha crecido de forma exponencial, superando inclusive la razón de crecimiento de la población.

Durante las últimas tres décadas, una de las herramientas de computación más utilizadas desde el punto de vista académico y empresarial ha sido la **inteligencia artificial** y sus múltiples ramas por la cantidad de aplicaciones que esta puede tener, que van desde el reconocimiento de voz hasta la predicción de fenómenos naturales (Cockburn et al., 2017); en particular el aprovechamiento de las herramientas provistas por la inteligencia artificial en el estudio del comportamiento del mercado financiero es uno de los campos que cuenta con más adeptos, motivo por el cual se han abierto puertas para que muchos académicos y empresarios inviertan tiempo y recursos en la preparación de modelos cada vez más complejos que mejoren los resultados a nivel de comunidad mejorando la calidad de vida de las personas.

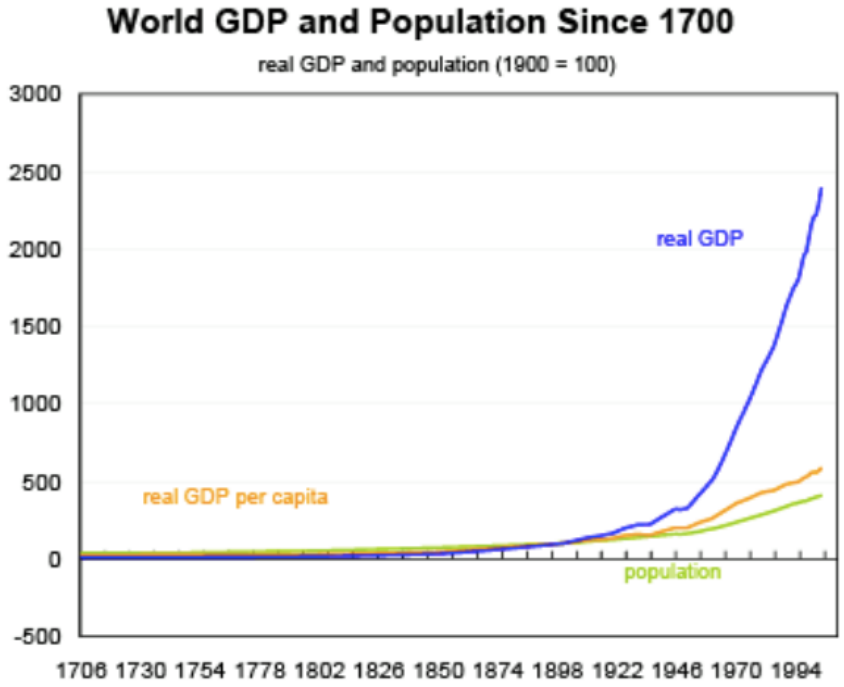

Figure 2.1: PIB y población en el tiempo

2.2 Inteligencia Artificial

Tuvo sus orígenes reales en los años 1950's a través de los trabajos publicados por Alan Turing en los que buscaba descubrir como una computadora pudiera tener la capacidad de pensar o al menos proveer conocimiento nuevo; todo esto basado en el trabajo propuesto por Ada Lovelace en los albores de la computación de mediados del siglo XIX. A pesar de no tener una definición universal, la inteligencia artificial puede entenderse como: *el esfuerzo de automatizar tareas intelectuales normalmente ejecutadas por los seres humanos* (Chollet & Allaire, 2018). A partir de este punto es que surge la opción de pensar que programas de computadora primitivos conocidos como **inteligencia artificial simbólica** hechos para resolver una partida de ajedrez o un juego de cartas pudieran entenderse como inteligencia artificial, sin embargo, esta programación depende de forma directa de un conjunto de reglas establecidas para ejecutar resultados a partir de los insumos que reciba (Chollet & Allaire, 2018). El problema que este tipo de programas enfrenta es que se vuelven limitados cuando resuelven actividades más complejas

como el reconocimiento facial o el reconocimiento de voz; por ello resulta conveniente cambiar el paradigma en el que programa procede (véase figura 2.2), procurando que sea el mismo el que define las "reglas" con las que obtiene resultados a partir de los datos de insumos; este paradigma se conoce como **aprendizaje de maquinas**, convirtiéndose en el motivante principal para una clasificación de la inteligencia artificial.

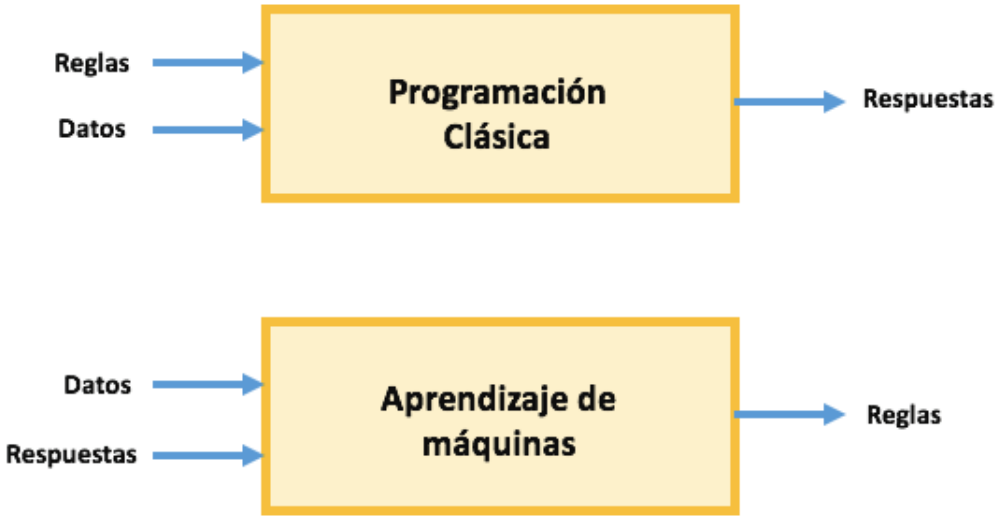

Figure 2.2: Comparación: Paradigma de progrmación para Deep-learning y Paradigma programación simbólica

2.2.1 Clasificación de la inteligencia Artificial

La problemática de la inteligencia artificial se la ha encarado desde muchas aristas en la actualidad y es por ello que su clasificación surge ante la necesidad de identificar los distintos enfoques que la misma tiene. El aprendizaje de maquinas es una de las sub-ramas de la inteligencia artificial y de las ciencias de la computación encargada del estudio de mecanismos y modelos que le permiten a la computadora aprender; esto es encargarse de generar un proceso de aprendizaje en la computadora que aproxime de la mejor forma posible un comportamiento esperado a través de un conjunto de datos de entrada que se provean para el análisis en cuestión (Michalski, Carbonell, & Mitchell, 2013).

Este esquema de clasificación de la inteligencia artificial, puede entenderse como una jerarquía que se conduce hacia el deep-learning que es el contexto específico del siguiente documento, esta jerarquía puede entenderse a través del esquema presentado en la figura 2.3

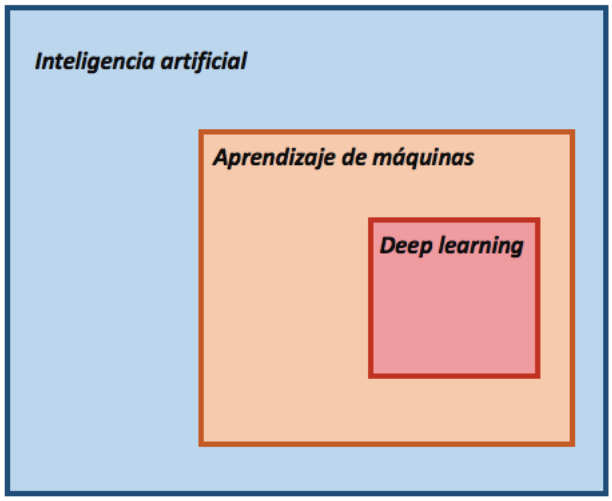

Figure 2.3: Jerarquía del Deep-Learning en IA

2.2.2 Deep Learning

Profundizando aun más el estudio de la inteligencia artificial, a través del aprendizaje de máquinas y tomando en cuenta los propósitos del presente trabajo, se procede con la explicación del Deep Learning, cuya particularidad respecto del resto de elementos encontrados dentro del aprendizaje de máquinas es la utilización de **capas**, inter-conectadas entre sí, cuyas conexiones normalmente están ocultas a la vista del usuario, pero cuyo propósito principal es el de ejecutar distintos grupos de cálculos de forma independiente con la intención de consolidar resultados que describan de mejor manera los fenómenos en cuestión (LeCun, Bengio, & Hinton, 2015).

Este modelo de representación por capas, se define en términos actuales como **redes neuronales**, que consiste en la compilación de varias capas apiladas una por encima de la otra.

2.2.3 Redes Neuronales

Una red neuronal es un modelo computacional que utiliza un sistema de **capas** internas y externas conectadas a través de unas estructuras denominadas **unidades** y que en conjunto simulan la arquitectura de las conexiones entre neuronas dentro del cerebro humano. Una vez que estas redes neuronales atravesaron el proceso de aprendizaje sobre un conjunto de datos conocidos se convierten en algoritmos capaces de predecir, en un rango previamente establecido, el comportamiento que pudieran tener un conjunto del mismo tipo de datos de los cuales solo se conocen las etapas previas y no los desenlaces en el comportamiento (Chollet & Allaire, 2018).

Un ejemplo de este estudio de capas, es el que tiene lugar en el procesamiento y reconocimiento de imágenes, como se observa en las figuras 2.4 y 2.5.

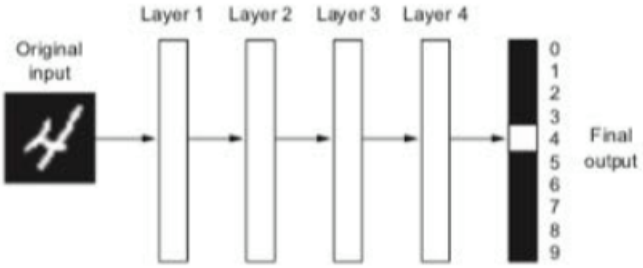

Figure 2.4: Estructura para procesamiento de imagenes

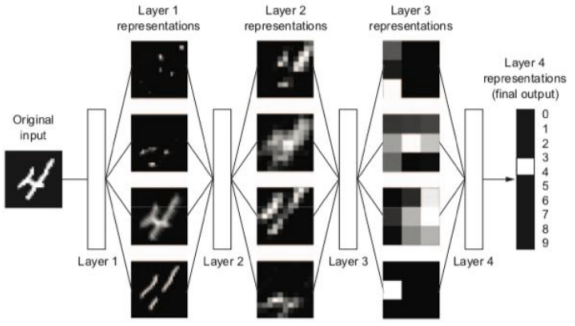

Figure 2.5: Estructura para procesamiento de imagenes (completo)

2.3 Contexto Económico

La bolsa de valores es una organización privada la cual dirige un mercado primario y un mercado secundario de acciones en donde se cierran diariamente transacciones de compra y venta de títulos empresariales de distinta índole. El mercado está destinado a cualquier inversionista particular o institucional capaz de destinar sus ahorros al mercado de valores, acudiendo primeramente al corredor de bolsa (casa de valores) que trabaja como intermediario. Las acciones son títulos del capital social de una empresa emitidos por la misma. Estas confieren a su titular accionista el derecho total en la participación dentro de la empresa que aluda a su desarrollo, al igual que en los beneficios económicos que esta presente(Griffin, Nardari, & Stulz, 2004).

La bolsa de valores trabaja a través de una rueda electrónica la cual se basa en un sistema operativo en donde está presente la interconexión de las ofertas y demandas, los calces y cierres de operaciones. Por medio de este software se permite que las casas de valores accedan para que los clientes realicen la compra y venta de papeles en respuesta a su propia búsqueda de rentabilidad de la inversión hecha. La organización cumple con tres funciones fundamentales destinadas a conferir un financiamiento a las empresas para así diversificar riesgos, proveer información de los distintos panoramas de los inversores y servir de foro para plasmar las cotizaciones de las empresas.

2.4 Motivación

Sistemas de inteligencia artificial, en particular aquellos que utilizan redes neuronales han sido de uso recurrente en varios países incluyendo el Ecuador y con aplicaciones en varias ramas de las ciencias. Se han encontrado trabajos con aplicaciones en dinámica empresarial, planificación de construcciones y hasta pruebas en algunos campos de la medicina (Escobar, 2002).

En el presente documento se propone la implementación de una red neuronal capaz de aprender sobre un conjunto de datos de activos financieros y a partir de ahí entender el funcionamiento de las arquitecturas elementales de redes neuronales con orientación a aplicaciones sobre otros campos.

2.5 Objetivos generales del presente trabajo

El acercamiento a las herramientas de inteligencia artificial para profesionales de los campos de la economía y la administración resultan de vital importancia para países en desarrollo que no han podido automatizar la mayoría de tareas financieras pues supone un desafío para las futuras generaciones desarrollar estas herramientas y proveerlas al sistema financiero tomando como propósito la mejora de la vida de las personas (Marwala, 2013).

En este trabajo se **estudia, analiza y resume el contenido teórico y algunas técnicas relacionadas con el aprendizaje de máquinas y el deep-learning** paso a paso, con el propósito final de resumir y explicar los fundamentos dentro de la materia que permitan de forma accesible describir y desarrollar un conjunto de herramientas de redes neuronales que en el capítulo final serán usadas para hacer predicciones sobre un conjunto de datos tomados como series de tiempo.

A partir del conjunto de técnicas y métodos estudiados, se **busca hacer una aplicación en predicción de precios de acciones sobre conjuntos de datos de libre acceso a través de la utilización de herramientas computacionales** relacionadas con el ámbito del Deep-learning que permitan una comparación entre los diferentes métodos desarrollados en la parte teórica.

CAPITULO II: REDES NEURONALES E INTELIGENCIA ARTIFICIAL

3.1 Neuronas

El estudio de las redes neuronales inicia con el entendimiento de la unidad fundamental: la neurona. Una **neurona** se define como una unidad de cálculo mínima que procura modelar el comportamiento de una neurona biológica y que en interacción con otras unidades similares forman un red neuronal que busca emular el funcionamiento del cerebro (LeCun et al., 2015).

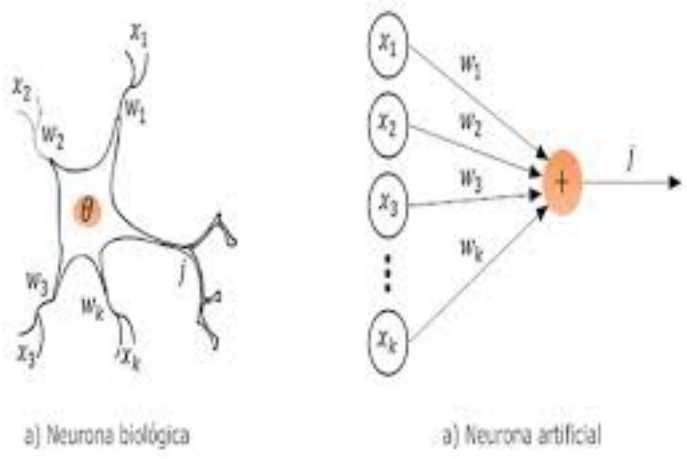

Figura 7. Neurona biológica versus artificial

Figure 3.1: Representación: Comparación neurona biológica con la neurona artificial

Dentro del entendimiento matemático, una neurona es una caja negra con un conjunto de elementos, cuya función es la de recibir un vector de insumos y devolver una respuesta. La neurona fundamental, también conocida como la neurona de McCulloch tiene como elementos fundamentales los valores de entrada y de salida, la función de activación, los pesos sinápticos y el umbral(Chollet & Allaire, 2018).

La respuesta de una neurona en función de los insumos X_i se calcula mediante la expresión 3.1:

$$y_i =: f^{W,b} = f_a\left(\sum_{i=1}^{N} W_i X_i + b\right) \tag{3.1}$$

Donde se identifica con X_i al vector de insumos, W_i los pesos sinápticos, b el umbral de la neurona y f_a la función de activación.

Los valores contenidos en el vector de insumos X_i son ponderados a través de un conjunto de pesos W_i que serán los primeros parámetros a ajustarse en cada una de las neuronas. Adicional de los insumos, cada neurona tiene una conexión adicional conocida como umbral, que aporta un valor a la expresión ponderada de los insumos que no depende de los mismos, pero que se ajusta directamente a la neurona. El valor obtenido en conjunto por los valores ponderados y el umbral atraviesan a un siguiente etapa donde son evaluadas en una función f (función de activación) que determina la forma en la que esta valor sera transmitido como valor de salida. De esta forma, la expresión para cada neurona viene dada como sigue:

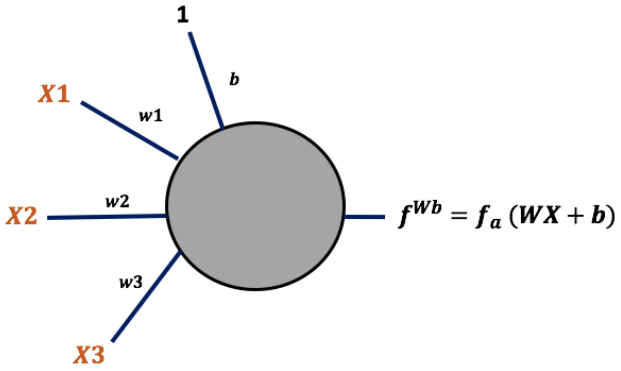

Figure 3.2: Representación: Neurona y función de activación

3.1.1 Funciones de Activación

En teoría de la computación, una función de activación es aquella que determina si el contenido dentro de cada una de las neuronas debe o no ser transmitida hacia la siguiente capa. Dentro del contexto de redes neuronales, cada una de las neuronas que componen la red posee una función de activación que regula la actividad o comunicación que cada una de las neuronas tiene con su siguiente conexión, en la mayoría de los casos permite controlar factores como el signo de la salida desde la neurona, generando una activación sobre un dominio binario. Las funciones de activación más primitivas, permitían o

negaban el paso definitivo de un elemento de salida desde una neurona de forma tajante. Esta función primitiva se puede escribir generalmente como:

$$f_a(x) = \begin{cases} 1 & W \cdot X - b \geq 0 \\ 0 & \text{otro caso} \end{cases}$$

Actualmente sin embargo, las funciones de activación buscan permitir el paso parcial de la salida de una neurona a través de un control sobre un rango de salidas más extensos y cuyas expresiones matemáticas sean diferenciables. La función de activación mas común es la sigmoideal, la cual se define a continuación:

$$f_a(x) = \frac{1}{1 + e^{-x}} \tag{3.2}$$

A pesar de no ser la única función de activación disponible, es una de las más usadas puesto que tiene un comportamiento mas suave y puede ser calculada su derivada a lo largo de todo su dominio.

Una de las mas importantes particularidades de esta función, es que permite realizar cálculos sencillos en lo referente al gradiente descendiente que es el método de optimización en las redes neuronales y que se podrá deducir en la siguiente sección, esto ocurre en razón de que a partir de la ecuación 3.2, y tomando en cuenta su derivada:

$$f'_a(x) = -\frac{1}{(1 + e^{-x})^2(-e^{-x})}$$

Además del hecho de que ambas funciones son continuas y definidas en todos los valores reales, se puede obtener que:

$$f_a(x) = \frac{1}{1 + e^{-x}}$$

$$\longrightarrow 1 + e^{-x} = \frac{1}{f_a} \longrightarrow e^{-x} = \frac{1}{f_a} - 1$$

Con lo que finalmente:

$$f'(x) = f^2[\frac{1}{f} - 1] = f - f^2$$

$$\Longrightarrow f'_a(x) = f_a - f_a^2 \tag{3.3}$$

3.2 Redes Neuronales

El concepto de neurona como agente individual resulta limitado hasta cuando se complementa con otras neuronas, puesto que de esta forma se puede emular la sinapsis dentro del cerebro y como tal completar el concepto.

Una **red neuronal** se define como un conjunto de neuronas concatenadas entre si a través de un proceso secuencial en el que la salida de cada una de las neuronas se vuelve en uno de los insumos de entrada de la siguiente neurona involucrada en el proceso. El vector de insumos inicial ingresa en las primeras neuronas en dependencia de la arquitectura de la misma y después de completar el proceso de concatenación la última neurona ofrece como salida el resultado final de toda la red. Es por ello que desde un punto de vista matemático, una red neuronal no es otra cosa que un mapeo F, puesto que toma un conjunto de valores de entrada dentro de las condiciones establecidas en la entrada y ofrece una salida que se ajuste a las condiciones de lo que se busca.

3.2.1 Estructura de una unión de neuronas: Capas

Una neurona dentro de una red neuronal puede concatenarse con otra, sin embargo, no todas estarán conectadas entre ellas puesto que el proceso secuencial sigue una estructura determinada por una serie de capas. En redes neuronales, una **capa** es un conjunto de neuronas que no interactúan entre si por encontrarse al mismo nivel, pero que en conjunto resultan en el vector de insumos para una siguiente capa de neuronas. El conjunto de capas de neuronas forman el mapeo del cual se obtiene el resultado de la red.

Este mapeo característico F esta formado por tres grupos de capas: La **capa de entrada**, la **capa de salida** y el conjunto de **capas intermedias** (o capas ocultas), cuyo número va a variar dependiendo del modelo que se construye. Cada una de estas capas actúa de forma independiente convirtiendo características de la información recibida en factores que se comunican hacia el siguiente nivel en la arquitectura.

La expresión matemática para la l+1-ésima, después de obtener los insumos desde la l-ésima capa viene dada por la expresión como sigue:

$$\mathbf{Z}^{l+1}(W_l, b_l) =: \mathbf{f}_l^{W_l,bl} = \mathbf{f}_l(Z^l) = \begin{pmatrix} f_{l,1}(W_{l,1}, b_{l,1}, Z^l) \\ \vdots \\ f_{l,N_l}(W_{l,N_l}, b_{l,N_l}, Z^l) \end{pmatrix}. \tag{3.4}$$

Cuya función principal es la de generar una respuesta en base a los valores aprendidos provenientes de las neuronas anteriores así como de sus propios niveles de activación. Esta respuesta es la que se emite hacia adelante para ser un dato de entrada en las neuronas de la siguiente capa.

En la resolución de problemas, se puede utilizar diferentes tipos de arquitecturas, las cuales dependerán de la cantidad de neuronas que contenga cada una de las capas y la cantidad correspondiente de las mismas. Es importante señalar que capas con diferente número de unidades pueden pertenecer a una misma arquitectura. Estas capas al juntarse forman el perceptrón por medio de una composición de funciones que puede escribirse como sigue:

$$\widehat{\mathbf{Y}} := F(\mathbf{X}) := \left(\mathbf{f}_1^{W_1 b_1} \circ \cdots \mathbf{f}_l^{W_l b_l}\right)(\mathbf{X}) \tag{3.5}$$

Considerando que cada neurona para el ejemplo emite una misma información, entonces cada capa puede escribirse como:

3.2.2 Ejemplos de redes neuronales

Considerando la idea fundamental de la concatenación, son muchas las formas en las que se pueden conectar las neuronas con la intención de obtener un resultado.

3.2.2.1 Red Neuronal Elemental

El ejemplo fundamental es el de una red que contiene 3 capas: la capa de entrada, la capa de salida y una sola capa intermedia formada por dos neuronas. La respresentación de la misma se puede ver a continuación:

Donde se puede identificas el elemento de entre unitario en la primera capa y notado con a_1. Los pesos sinápticos en las conexiones hacia la segunda capa, donde se disponen dos neuronas notadas como a_1 y a_2, además de los pesos sinápticos correspondientes al paso hacia la tercera capa que en este caso resulta en la capa de salida y que se nota con y_1.

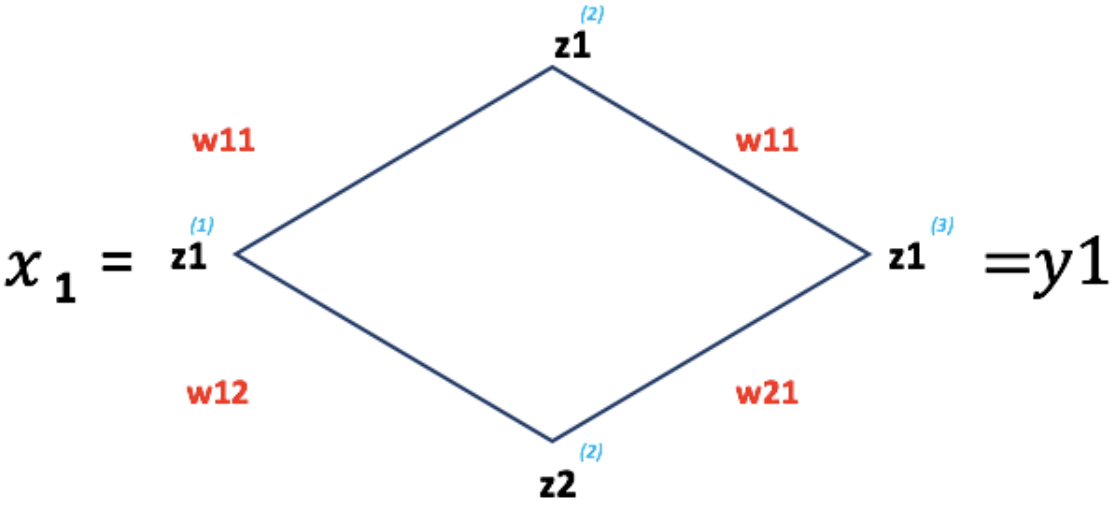

Figure 3.3: Representación: Red Neuronal elemental con notación

3.2.2.2 Aplicación de red neuronal: Función XOR

Una aplicación de red neuronal se puede hacer con arquitectura simple orientada al aprendizaje de la función *y* inclusiva, fundamental en la lógica que recibe de entrada dos elementos binarios conteniendo 1 y 0, y devolviendo 1 cuando exclusivamente uno de los elementos es uno y cero en otros casos. De esta forma, la ecuación puede escribirse como:

$$f_{XOR} : \mathbb{R}^2 \rightarrow \mathbb{R} \tag{3.6}$$

$$([0,0];[0,1];[1,0];[1,1]) \rightsquigarrow (0;1;1;0)$$

Para este problema, se hace uso de una arquitectura de tres capas: El vector de entrada con dos elementos x_1, x_1, una capa intermedia con dos neuronas y finalmente una salida con la predicción y (Goodfellow, Bengio, & Courville, 2016). El esquema se puede visualizar en la figura

Tomando en cuenta la ecuación 3.4, obtenemos expresiones para la primera capa con los elementos de entrada (los pares x_0, x_1), así como la capa oculta, misma que recibirá como elementos de entrada, los elementos de salida de la primera capa, de esta forma las ecuaciones para las capas vienen dadas como sigue:

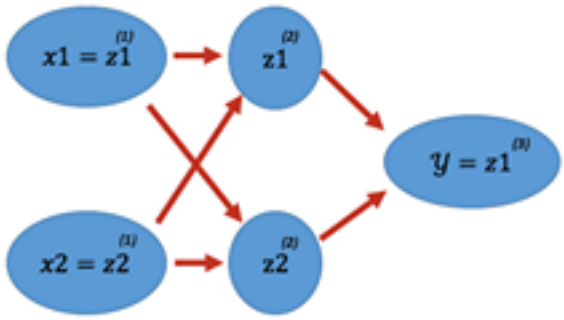

Figure 3.4: Red Neuronal para aprendizaje de función XOR

$$Z^1 = f_{a1}(W_1^T X_0 + b_1)$$

$$Z^2 = f_{a2}(W_2^T Z^1 + b_2)$$

En estas ecuaciones puede identificarse la misma estructura que la obtenida en la ecuación 3.4, con lo que W notará los pesos sinápticos, X, Z los insumos de entrada y de paso a la segunda capa y b los umbrales para cada capa. Las función de activación en la primera capa es una reLu (recttified linear unit), que recibe como entrada un valor y devuelve el mismo valor en caso de ser positivo o 0 en caso de ser negativo, matemáticamente siendo el valor de entrada a entonces la función se leerá como $f_{reLu}(a) = \text{máx}(0; a)$. Por otro lado, la función de activación para la capa intermedia es la función identidad. De esta forma, como se mencionó el mapeo F puede obtenerse a través de la composición de funciones con lo que el modelo de redes neuronales para la función XOR quedaría como se muestra en la ecuación 3.7

$$f_{XOR}(X_0, W_1, b_1, W_2, b_2) = W_1^T \max[0; W_1^T X_0 + b_1] + b_2 \tag{3.7}$$

La solución para esta arquitectura viene dada con los siguientes valores:

$$X_0 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}; \quad W_0 = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}; \quad b_0 = \begin{pmatrix} 0 \\ -1 \end{pmatrix}$$

$$X_1 = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{pmatrix}; \quad W_1 = \begin{pmatrix} -1 \\ 2 \end{pmatrix}; \quad b_1 = 0$$

3.3 Redes Neuronales: Relación con el aprendizaje supervisado

Un conjunto de datos de entrenamiento es un grupo de parejas de la forma (x_i, y_i), en el cual se distingue entre el vector x_i de k variables que funciona como insumo y un valor y_i que hace las veces de respuesta o resultado.

El **aprendizaje supervisado** es la técnica de búsqueda de una función

$$f : \mathbb{R}^k \to \mathbb{R}^l$$

a partir de un conjunto de datos de entrenamiento y aplicados sobre un conjunto de datos de prueba con la intención de obtener un vector de salidas $f(x_i)$ tales que la diferencia con el conjunto de resultados y_i sea la menor posible en términos del criterio que se este tomando en cuenta.

3.3.1 Aprendizaje supervisado en redes neuronales

Las redes neuronales pueden usarse para aprendizaje supervisado puesta que considera un vector de insumo como valores de entrada y vector de valores de resultados; establece esta función por medio de una arquitectura especifica y obtiene predicciones que serán comparados con el conjunto de resultados por medio de un criterio de cercanía que mejor describa la naturaleza del problema. No obstante, se tiene que tomar en cuenta que las funciones que pueden obtenerse por medio de este mecanismo están limitadas a las arquitecturas que puedan construirse tomando en cuenta el comportamiento de las neuronas, de las redes neuronales, forma de los datos de entrada y el vector de resultados que se quiere obtener.

3.3.2 Loss Function

El propósito principal de las redes neuronales entonces es la construcción de una función capaz de entregar una predicción lo más semejante posible a los resultados a partir de un vector de insumos;

por esta razón es que al momento de definir un criterio de proximidad para comparar los resultados se requiere de una función adicional, conocida como **loss function**.

La loss function se define como la herramienta de un modelo de redes neuronales encargada de medir o cuantificar la diferencia existente entre la predicción obtenida por uno de sus modelos y los resultados reales.

Existen varios tipos de funciones objetivo, cada una de ellas sigue una descripción matemática diferente que se ajusta a las necesidad específicas del contexto de cada problema. Se considera que las loss functions se evalúan sobre todo el conjunto de entrenamiento de tamaño M. Tomando y_i como el valor del resultado y para la presente sección notando $\hat{y}_i$ al valor obtenido como predicción en la red neuronal se puede tomar en cuentas las siguientes loss functions como las más utilizadas:

3.3.2.1 Mean-squared Error

Es uno de los métodos mas antiguos y más usados, especialmente cuando se tiene que enfrentar problemas relacionados con regresiones lineales puesto que puede entenderse directamente como distancias dentro del plano euclideano. En su forma estándar, esta función se escribe como:

$$\mathcal{L} = \frac{1}{M}\sum_{i=1}^{M}(y_i - \hat{y}_i)^2 \tag{3.8}$$

3.3.2.2 Mean-squared Logarithmical Error

Semejante al descrito anteriormente, emplea logaritmos que permiten reducir las expresiones obtenidas cuando los valores en la predicción y conocidos resultan muy grandes. Resulta muy útil en modelos que presentan valores muy grandes en terminos relativos y que no permiten normalización puesto que esta función de error conserva los mencionados valores evitando el sobre-entrenamiento. Una última ventaja es la consideración de una menor penalidad sobre las ventajas que resultan muy grandes, concepto que puede aplicar a modelos donde se busque mejores resultados en variables grandes. Su expresión matemática resulta en:

$$\mathcal{L} = \frac{1}{M}\sum_{i=1}^{M}(\log(y_i + 1) - \log(\hat{y}_i + 1))^2 \tag{3.9}$$

3.3.2.3 Mean-absolute error

Esta función considera errores asumiendo directamente una métrica euclideana. Es sumamente útil en el pronóstico de resultados de predicción, puesto que se puede obtener de manera directa la diferencia existente en términos de módulo entre las predicciones y los valores reales. El mayor problema que presenta esta función de error es la dificultad para el calculo del gradiente, sin embargo, este puede solucionarse a través de distintos métodos computacionales obteniendo además resultados mas robustos. Su expresión matemática se muestra a continuación:

$$\mathcal{L} = \frac{1}{M} \sum_{i=1}^{M} \mid y_i - \hat{y}_i \mid \tag{3.10}$$

3.3.2.4 Mean-absolute percentage error

Función muy parecida a la descrita en la sección anterior, con la diferencia de que provee como resultado el cambio porcentual existente en reemplazo del cambio en términos de métrica euclideana. Su expresión matemática resulta en:

$$\mathcal{L} = \frac{1}{M} \sum_{i=1}^{M} \mid \frac{y_i - \hat{y}_i}{y_i} \mid \tag{3.11}$$

3.3.3 Panorama del problema

El **aprendizaje supervisado** con **redes neuronales** es entonces un problema de optimización matemático cuyo objetivo principal es el de definir y establecer mediante una arquitectura de redes neuronales un conjunto de parámetros que van a ser ajustados por medio de la mayor minimización posible de una **loss function** con el objetivo de obtener un mapeo que convierta un vector o matriz de insumos específicos en un predicción o salida que se acerque de la mejor forma posible al resultado obtenido en el evento o fenómeno especifico de estudio.

3.4 Resolución del problema de optimización: Cómo reducir la loss function

Una vez que se ha dejado claro el tipo de problema que se está resolviendo, se busca explicar cual es la forma correcta en la que se puede proceder para resolver el problema; esto es encontrando una forma de

reducir la loss function a través de un algoritmo eficiente que modifique gradualmente los parámetros para la obtención de unas predicciones mas cercanas a los resultados

3.4.1 Gradiente descendiente

Considerando que en los modelos de redes neuronales, las predicciones dependen de los parámetros específicos de cada neurona y de sus funciones de activación, entonces puede pensarse que las loss function también dependerán de estos mismos parámetros. Estas loss function en la mayoría de los casos cumplen con las condiciones suficientes de diferenciabilidad que garantizan la existencia del vector gradiente, elemento que permite la optimización y que se define como:

$$\nabla \mathcal{L} = \begin{pmatrix} \frac{\partial \mathcal{L}}{\partial w_1} \\ \frac{\partial \mathcal{L}}{\partial w_2} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial w_n} \end{pmatrix}. \tag{3.12}$$

Considerando que en el presente documento se hace uso de la *Mean Absolute Error* como loss function y tomando en cuenta W, b como el cojunto de parámetros a optimizar, entonces la loss function definida en términos de sus parámetros puede escribirse como en la ecuación 3.13

$$\mathcal{L} = \mathcal{L}(W, b) = \frac{1}{M} \sum_{i=1}^{M} \mid F^{W,b}(X_1) - y_i \mid \tag{3.13}$$

A partir de la ecuación 3.13, se obtienen las entradas de cada uno de los elementos del gradiente por medio de una derivación parcial, obteniendo:

$$\frac{\mathcal{L}}{\partial W_j} = \frac{1}{M} \sum_{i=1}^{M} (-1)^m \frac{F^{W,b}(X_1)}{\partial W_j} \tag{3.14}$$

A partir de la ecuación 3.14 se puede obtener una expresión para cada uno de los parámetros, elementos constituyentes del vector gradiente.

En una red neuronal básica, el método del gradiente descendiente es el método de optimización mas utilizado, este solamente hace uso del gradiente, por lo que se considera un método de primer orden. El fundamento de este método es seguir la dirección contraria a la indicada por el gradiente con la intención de obtener el siguiente conjunto de valores.

De esta forma, sea W_i el vector que contiene a todos parámetros obtenidos después de i modificaciones, entonces se tiene que:

$$W_{i+1} = W_i - v_i \nabla \mathcal{L}$$

Donde $\nabla \mathcal{L}$ denota el vector gradiente y v_i son los valores asignados como tasas de aprendizaje; mismos que se obtienen a priori o mediante un proceso de optimización externo sobre cada uno de los pasos.

3.4.2 Back-Propagation

Es el algoritmo que emplea Gradiente Descendiente para la resolución de un problema de redes neuronales. Para mostrar el funcionamiento de una forma sencilla, se considera la red neuronal elemental considerada en la figura 3.3 además de la medición por medio de una loss function *Mean Absolute Error* y una función de activación sigmoideal en cada una de las neuronas, a partir de lo cual considero la expresión de la ecuación 3.14, suponiendo que busco derivar mi función respecto del peso sináptico w_{11}, mismo que puedo escribir como:

$$Z_1^{(3)} = f(b_1^3 + w_{11}^{(2)} Z_1^{(2)} + w_{21}^{(2)} Z_2^{(2)})$$

Sin embargo, las expresiones para las neuronas dentro de esta ecuación también pueden escribirse en función de los parámetros a partir de la misma ecuación:

$$Z_1^{(2)} = f(u_1^2 + w_{11}^{(1)} Z_1^{(1)})$$

$$Z_2^{(2)} = f(u_2^2 + w_{12}^{(1)} Z_1^{(1)})$$

Con lo que finalmente, la expresión en función de los parámetros de la ren neuronal elemental se puede escribir como:

$$Z_1^{(3)} = f(u_1^3 + w_{11}^{(2)} f(u_1^2 + w_{11}^{(1)} Z_1^{(1)}) + w_{21}^{(2)} f(u_2^2 + w_{12}^{(1)} Z_1^{(1)}))$$

Esta última expresión entonces se puede concatenar con la ecuación 3.14, obteniendo que el cambio en la loss en dependencia de los parámetros se puede escribir como:

$$\frac{\partial \mathcal{L}}{\partial w_{11}} = \sum_{i=1}^{M} (-1)^m f(u_1^3 + w_{11}^{(2)} f(u_1^2 + w_{11}^{(1)} Z_1^{(1)}) + w_{21}^{(2)} f(u_2^2 + w_{12}^{(1)} Z_1^{(1)}))$$

Que finalmente, por ser una función sigmoideal se reduce a:

$$\frac{\partial \mathcal{L}}{\partial w_{11}} = \sum_{i=1}^{M} (-1)^m Z_1^{(3)} (1 - Z_1^{(3)}) w_{11}^{(2)} Z_1^{(2)} (1 - Z_1^{(2)}) a_1^{(1)} \tag{3.15}$$

3.4.3 Batch Gradient Descendent

La utilización del gradiente descendiente como modelo de optimización en los modelos de redes neuronales funciona de manera eficiente en la mayoría de los casos, sin embargo, los costos computacionales de mantener este algoritmo son sumamente altos puesto como puede verse en la descripción anterior, el mismo hace paso por todo el conjunto de datos lo que puede complicar el tiempo de ejecución. Es por ello, que en redes neuronales sobre las cuales el conjunto de insumos es más amplio es conveniente utilizar el modelo de gradiente descendiente estocástico, que a diferencia del modelo anterior, no hace paso por todo el conjunto de datos, sino que selecciona lotes de muestras sobre los cuales ir modificando y optimizando los parámetros.

El proceso consiste en tomar en cuenta un subconjunto de datos a partir del conjunto de entrenamiento de tamaño r de forma aleatoria. Este subconjunto de datos se define como **Batch de tamaño** r y contiene esa misma cantidad de elementos tomados desde el conjunto de entrenamiento de tamaño M. El proceso de optimización ocurre de forma muy similar que en el gradiente descendiente, solo que para actualizar los pesos no se requiere recorrer todo el conjunto de entrenamiento, puesto que el mismo se realiza solamente en el batch de entrenamiento, de esta forma si se tiene un batch B de tamaño r, entonces la expresión en 3.14 se convierte en

$$\frac{\mathcal{L}}{\partial W_j} = \frac{1}{M} \sum_{i \in B} (-1)^m \frac{F^{W,b}(X_1)}{\partial W_j} \tag{3.16}$$

De esta forma se puede calcular un gradiente sobre el batch, el mismo que se nota como $\nabla \mathcal{L}_B$ de una forma más eficiente. Posteriormente, la actualización de los parámetros en este método es similar a la del gradiente descendiente, salvo por el cambio en el gradiente, expresión que se muestra a continuación:

$$W_{i+1} = W_i - v_i \nabla \mathcal{L}_B$$

Este procedimiento será tomado en cuenta durante los experimentos del presente documento.

3.5 Manejo de los datos

Una vez que se ha mencionado el funcionamiento básico de las redes neuronales, se tiene que tomar en cuenta también la forma en la que los datos van a ser usados con la intención de obtener los mejores resultados. En primera instancia, una idea intuitiva sería la de tomar el conjunto total de los datos disponibles y dividirlos en partes iguales de forma tal que la primera parte pueda servir para ajustar los parámetros del modelo y la segunda parte de los datos sirva para evaluar de que forma y con que precisión el modelo esta haciendo sus predicciones.

Sin embargo, en el proceso de optimización de un modelo se tiene que tomar en cuenta los **hiper-parámetros** (componentes propias de una arquitectura de redes neuronales como la cantidad de capas o de neuronas que tendrá un modelo) que resultan ser nuestro mecanismo más directo de modificación del modelo y que dividir los datos y tomar en cuenta solamente dos partes, se estaría ajustando el modelo solamente con miras a la obtención de mejores resultados sobre el conjunto de datos escogidos para evaluar las predicciones. De esta forma, en redes neuronales se ha optado tradicionalmente por dividir el grupo de datos en tres segmentos: Entrenamiento, Validación y Prueba; con lo que los dos primeros grupos de datos se ajustan a la idea intuitiva de la separación requerida, mientras que el tercer grupo de datos de forma independiente ofrece resultados de la calidad del modelo. De esta forma, se evitan problemas como el sobre entrenamiento o los vacíos de memoria.

La selección de datos es de suma importancia, por lo cual para un conjunto de datos de series de tiempo que es el tema de interés del presente documento, sobre el cual se requiere implementar un modelo de redes neuronales considera algunas condiciones mínimas para poder ser usado de forma exitosa. Este conjunto de datos debe ser balanceado puesto que una entrada o un objetivo que contenga la ausencia de un valor no puede ser ingresado dentro del modelo. Para mejorar los resultados que pudieran obtenerse a través de procesos mas elementales como mínimos cuadrados, se requiere también que los problemas gocen de una mayor riqueza en términos de cantidad de datos y de variables que garanticen mejores predicciones.

Una vez que se ha seleccionado de forma correcta el conjunto de datos, se divide el mismo en tres partes

para garantizar que cada una de las etapas tenga acceso a un conjunto de datos diferente sobre el cual pueda ejecutar los algoritmos correspondientes. Sobre cada una de las etapas se establece una actividad específica, tanto de las funciones de optimización como del uso específico de los datos, con lo que se obtienen las etapas a continuación:

- Entrenamiento.- En esta etapa se modifican originalmente los parámetros a partir de los valores por defecto (automáticos o escogidos para el experimento). El mecanismo de optimización es la loss function que se escoja para el modelo, la cual tiende a reducirse en cada una de las épocas de aprendizaje.

- Validación.- Durante esta etapa, un conjunto de datos diferentes a los utilizados en la etapa de entrenamiento son puestos a prueba con el modelo que se va ajustando en cada época con los datos de entrenamiento y se despliega el valor obtenido para la loss function en cada caso, el cual servirá como una bandera guía para el modelo indicando que tipo de elementos tienen que tomarse en cuenta al momento de hacer una modificación.

- Prueba.- Durante ésta etapa, el conjunto de datos es totalmente nuevo, no existe ninguna modificación de los parámetros y lo que se hace es determinar a través de los parámetros ya establecidos que tan preciso resulta ser un modelo. En esta etapa se puede realizar comparaciones entre modelos.

3.6 Desarrollo del presento documento: Arquitecturas y datos específicos

Este documento ofrece como propósito fundamental el desarrollo de arquitecturas de redes neuronales que puedan ser usadas y aplicadas con éxito en el ámbito del mercado financiero, recordando que una de las metas es la de predecir de la mejor forma posible el precio de una acción. Es por esto, que como conjunto de datos se utilizarán las variables relacionadas con el precio de una acción para la empresa de datos Google, la misma que se describe con detalle en el capítulo siguiente y que tomará como insumos de entrada los datos para las variables en algunos períodos de tiempo y como salida el precio ajustado de la acción.

La filosofía del proceso de un modelo de redes neuronales es la de recibir un conjunto de datos,

sean estos vectores, tensores o secuencias y conseguir generar predicciones a través de las mismas que mejor se adapten a los valores reales, entregados en forma de valor escalar, vectorial o tensorial.

En el problema planteado en el presente documento, se busca ingresar por medio de un generador de secuencias un bloque de entradas cuya dimensión depende de los parámetros externos antes ya especificados para obtener una salida de valor escalar, asociado con la predicción del modelo para una de las variables dentro del conjunto de datos.

Concretamente, si se considera r como el rezago del problema, p como el número de pasos y v como el número de variables, entonces el modelo puede entenderse como una función:

$$F(\mathbf{X}) = \mathbf{Y}_{\text{pred}} \tag{3.17}$$

Donde $\mathbf{X} \in \mathbb{R}^{\frac{vr}{p}}$, mientras que $\mathbf{Y}_{\text{pred}} \in \mathbb{R}$

A lo largo de las diferentes arquitecturas implementadas, se modificaran los valores para definir adecuadamente el problema que se esta planteando, puesto que cada arquitectura provee sus mejores resultados en dependencia de la modificación de los hiper-parámetros.

3.6.1 Parámetros externos

Son todos los parámetros que tomamos en cuenta como contexto específico del problema. Esto quiere decir que no se relacionan directamente con la arquitectura o con las herramientas que provee el software, pero que si determinan la optimalidad del modelo a través de una correcta elección de los mismos.

El problema de análisis principal en el presente documento, es el de generar predicciones sobre el valor de cierre que toma la acción de una empresa a partir de un conjunto de variables auxiliares y de un conjunto de datos del pasado. Los datos de entrada en este modelo serán secuencias de datos comprendidas por varias entradas cada una de ellas con sus variables específicas.

Para describir un proceso de predicción se tiene que tomar en cuenta que mucho tiene que ver la cantidad de datos que se toma del pasado durante el análisis, la cantidad de períodos de tiempo hacia el futuro, así como la cantidad de iteraciones que se hace durante un periodo.

En este problema específicamente, la formulación quedaría como sigue:

" Teniendo en cuenta la información financiera de los últimos REZAGO, muestreados cada período de PASO días, se puede predecir el precio de cierre de la acción para un día después. "

Donde claramente se entiende que los parámetros describen:

1. REZAGO.- Observaciones hacia el pasado.
2. PASO.- Cada cuantos días se toma el muestreo.

3.6.2 Feedforward Network

La primera representación posible que se analiza es la de red neuronal hacia adelante (Deep Forward). Idealizar la forma y el funcionamiento del perceptrón multicapa es recurrir al pensamiento de una maquina sencilla con modulos ajustables. Una representación mental se muestra a continuación:

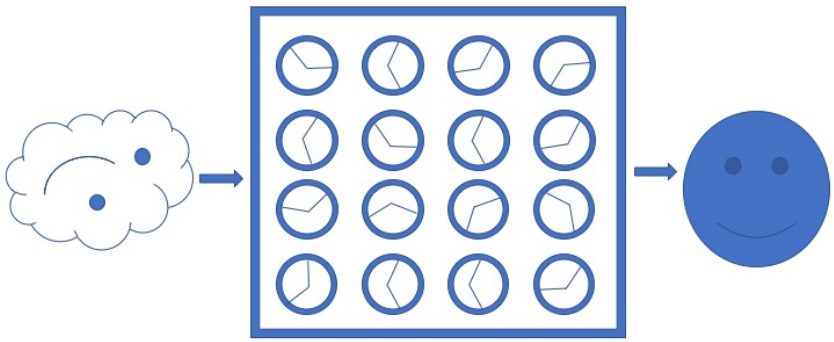

Figure 3.5: Representacion mental del proceso de predicción.

En donde se tiene una maquina capaz de obtener cierta información y en base al aprendizaje previamente realizado puede predecir información importante. El entrenamiento es la etapa en la que se definen la posición de las rodelas para que la maquina de predicción pueda cumplir con su trabajo de la mejor forma, pues se asume que esta viene sin un aprendizaje y que debe ajustarse al modelo sobre el cual estamos trabajando.

La Feedforward Network es en efecto equivalente al perceptrón multicapa, en este caso los hiper-parámetros serán los elementos que definen la arquitectura especifica, los cuales son:

- Tamaño del vector de insumos
- Número de capas
- Número de neuronas en cada capa
- Tamaño del vector de predicciones

Para el experimento se especifica la arquitectura en el siguiente capítulo, pero con el propósito de identificar los elementos por medio de un ejemplo se muestra a continuación el esquema de una red Feedforward Network con un vector de cinco insumos, un total de tres capas internas con tres, dos y tres neuronas respectivamente y un vector de cuatro entradas como resultado; que puede verse en la fig 3.6

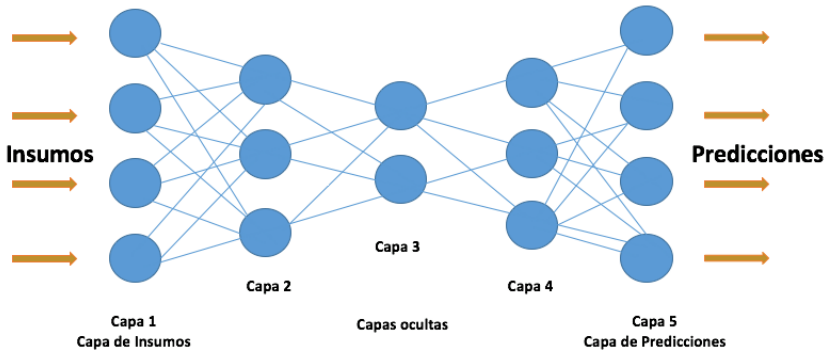

Figure 3.6: Ejemplo de FeedForward Network

3.6.3 Red Neuronal Recurrente(RNN)

Cuando se procesa información, se puede requerir tomar en cuenta la información de lo ocurrido un periodo atrás en el tiempo, puesto que esto permite explicar de forma directa ciertos resultados y se adapta de forma mas natural a la forma de raciocinio humano. En particular en este proyecto, los

insumos de entrada serán datos de series de tiempo y por tanto una arquitectura de redes neuronales que considere una recurrencia puede resultar importante para el análisis correspondiente.

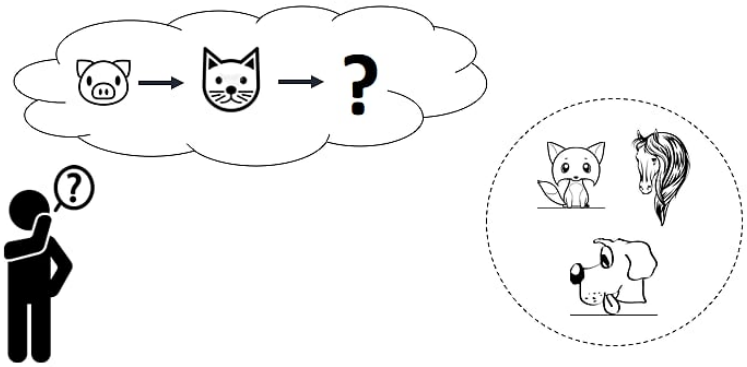

Figure 3.7: Proceso mental de recurrencia humana

La **red neuronal recurrente** es una arquitectura de redes neuronales que consiste en conectar una unidad de procesamiento consigo misma a través del tiempo, esto quiere decir que cada nueva capa dentro de la arquitectura es la misma unidad de procesamiento pero en un período de tiempo diferente. Esta unidad de procesamiento posee como característica propia un vector de estado conocido como **estado oculto** que se actualiza en cada período de tiempo a partir de sus carácteriticas en el pasado y de los nuevos insumos que recibe (estos elementos son consideradas entradas para la capa) y cuya información se utiliza para generar una salida desde cada una de las capas. De esta forma, cada capa en la red es un período de tiempo de la unidad de procesamiento (entendido como una neurona) que entrega una respuesta en función del estado oculto para ese período de tiempo.

Este proceso traducido a notación matemática consiste en obtener para el tiempo presente el vector de estado oculto h que capturen de forma eficiente información del período anterior por medio de la ecuación:

$$h_t = \tanh(W_h h_{t-1} + W x_t) \tag{3.18}$$

Donde la expresión de tangente hiperbólica representa la función de activación de la actualización del estado oculto, que es la más común en este caso. W_h esta formado por una matriz de recurrencia (con nuevos parámetros incorporados, pero cuyo compotamiento es similar al de los pesos W en la neurona

simple) y W son los pesos (parámetros) propios de la neurona anterior.

Una vez obtenido el vector de valores ocultos h, se obtiene la expresión de salida de cada una de las neuronas dentro de la red recurrente por medio de la expresión:

$$y_t = f(W_y h_t + b) \tag{3.19}$$

Donde W_y es la matriz de pesos de costumbre, que en consideración de la forma en la que se calcula el vector del estado oculto permite obtener salidas desde la neurona actual con conisderación del pasado. De esta forma, el esquema de una recurrencia simple se puede tomar en la figura 3.8

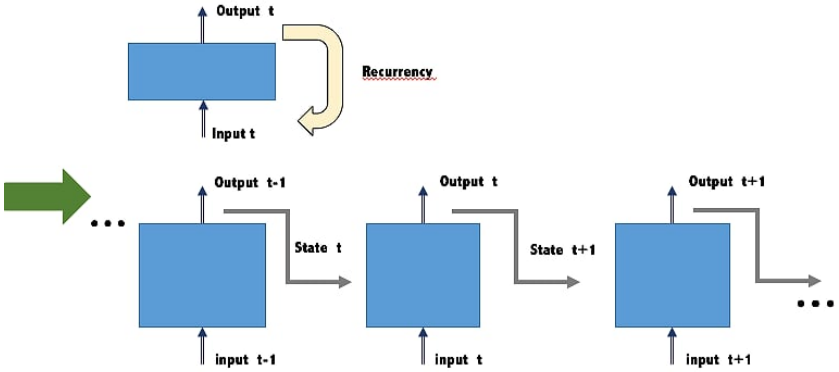

Figure 3.8: Esquema elemental de una red neuronal recurrente

3.6.4 LSTM

Las redes neuronales recurrentes simples no ofrecen una predicción tan completa, puesto que no son capaces de tomar en cuenta lo ocurrido en el pasado de algunos periodos antes del actual. Para el inicio de los años 90's sin embargo, se desarollo la arquitectura LSTM, que toma en cuenta una celda de memoria completa, más completa que el estado oculto de la RNN que carga información desde el pasado con el propósito de mejorar la red.

Esta celda de información, muy parecida a la unidad de procesamiento de las RNN pero con más elementos que durante cada período se actualizan y se modifican, a través de un conjunto de funciones

conocidas como **puertas** que no son otra cosa que capas neuronales con un propósito específico dentro de la arquitectura y que utilizan como función de activación mas común entre los textos académicos la sigmoideal, razón por la que se le representa usualmente con σ, cuya respuesta es el paso o no de los elementos dentro de la celda.

3.6.4.1 Elementos que componen la LSTM

Este tipo de redes, para el cual cada capa se entiende como un período de tiempo de la celda de memoria, poseen mucho elementos que permiten una mejora respecto de la red anterior, puesto que permite almacenar otros espacios en memoria. Estos elementos se pueden describir a través de funciones:

1. **Forget Gate.-** Es la primera de estas funciones puerta, toma en cuenta el vector de estado en $t-1$, así como el insumo para el período t y su rol fundamental es el de decidir que información contenida en la celda de información es la que se mantiene y cual se desecha.

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

2. **Input Gate.-** Toma los mismos elementos de entrada que la función descrita previamente, sin embargo, su propósito es la decidir cuales de los nuevos valores pudieran añadirse a la celda de la información.

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

3. **Candidata.-** Los valores que pudieran o no ser ingresados a la celda de la información dependerán del conjunto de insumos, así como de los valores contenidos en el vector de estado, la expresión para la generación de estos candidatos entonces viene dada por:

$$\overline{C_t} = \tanh(W_c[h_{t-1}, x_t] + b_c)$$

Estado Actual de Celda.- A partir de los elementos anteriores se modifica la celda mediante la expresión:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \overline{C_t}$$

4. **Output Gate.-** Actúa como regulador de la celda respecto de su proyección hacia el output. Toma en cuenta los valores de insumo del tiempo presente y los valores del vector de estado, teniendo como función controlar cuales de los elementos de la celda ya actualizada son los que se ofrecen hacia el output.

$$O_t = \sigma(W_O[h_{t-1}, x_t] + b_O)$$

Con lo que la salida desde esta neurona viene dada por la expresión a continuación:

$$y_t = O_t \cdot \tanh C_t$$

3.6.4.2 Explicación del proceso de la LSTM

La arquitectura de la red LSTM es considerada una red neuronal recurrente puesto que en cada una de sus capas recibe como entradas un vector de insumos así como un vector de estado, sin embargo, incorpora la idea de una celda de información, la cual contiene valores que solo se van modificando gradualmente con el paso de los períodos de tiempo pero que se mantiene como una misma estructura, por lo que no depende solo del vector de estado del período de tiempo anterior.

A partir de los elementos descritos entonces, se puede entender que la salida obtenida en esta neurona mucho va a depender de esta celda de estado, por lo que su actualización hacia el tiempo t es fundamental. En primera instancia, la Forget Gate se calcula para determinar por medio de un producto que información de la celda es la que continúa y cual se olvida, proceso que corresponde al primer término de la actualización de la celda. Por otro lado, los nuevos valores a incorporarse se obtienen desde la expresión candidata y se ponderan a través de la Input Gate como manera de control de la actualización de la celda. Finalmente, la salida desde el tiempo t depende de una tercera puerta que es encargada de decidir la salida pero que no modifica el estado actualizado de la celda que también es transmitido hacia la siguiente neurona.

Se puede entonces evidenciar que la celda de información es el nuevo insumo que recibirá cada una de las capas de la arquitectura y cuyo propósito fundamental es el de mantener información de los cambios según como los períodos vayan transcurriendo.

El proceso puede resumirse en la figura 3.9, donde las puertas estarán notadas por σ y las operaciones con los signos que corresponden. Se evidencia que la celda de estado se transmite en la linea horizontal superior y es continua durante las etapas.

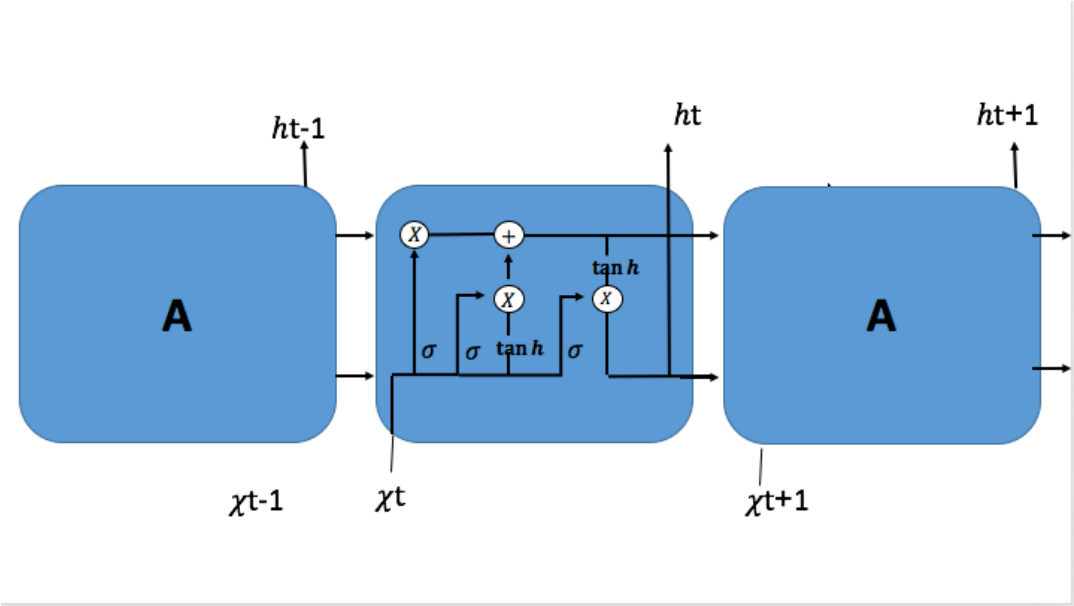

Figure 3.9: Estructura de la arquitectura LSTM

3.7 Software computacional y cálculos en el ordenador

El estudio de los algoritmos de inteligencia artificial se ha puesto en auge con el contexto tecnológico en el se ha vivido estos últimos años a través de la mejora de hardware y software así como por la consolidación de distintas empresas de software que han fomentada una cultura de sana en competencia entre muchas de ellas por desarrollar cada vez mejores mecanismo de aplicación de sus sistemas.

3.7.1 Keras y R

R es un software ambiente de desarrollo libre cuyo propósito principal es el análisis estadístico y el de la creación de gráficos académicos profesionales. Sus orígenes se remontan a un lenguaje mas primitivo desarrollado por Bell conocido como S y cuya filosofía era la de convertir en software ideas de forma rápida y apegada al pensamiento humano.

R provee un conjunto de herramientas estadísticas y gráficas que permiten el análisis de diferentes modelos estadísticos así como una adecuada representación gráfica de cada uno de los elementos que se analizan con la intención de volverlos académicos y de alto nivel. Entre otras ventajas que ofrece el sistema de desarrollo de R se encuentran:

- Efectividad en el manejo y almacenaje de datos
- Operadores adecuadamente programadas para trabajar con arreglos, matrices y tensores
- Acceso sencillo a funciones elementales de la programación como lazos, condicionales entre otros.

Por todas estas ventajas, R resulta ser un software de gran ayuda y apoyo para el análisis matemático de series de tiempo entre otros sistemas de predicciones, además la facilidad de almacenaje de datos y de manipulación tensorial permite acceder a grandes bloques de datos y distintos métodos con varios parámetros que se ajustan a las necesidades requeridas por el contenido teórico del Deep Learning.

3.7.1.1 Keras y Tensorflow

Tensorflow es una biblioteca de software de libre acceso para los usuarios cuya principal potencialidad es la del calculo numérico de alto rendimiento. Originalmente desarrollado por los miembros del equipo de Google Brain dentro de la organización AI de Google tiene un sólido respaldo en materia de Machine y Deep Learning.

Por su parte, Keras es una interfaz de aplicación de alto nivel cuya filosofía fundamental es la de permitir una pronta experimentación al usuario sobre los distintos software. Keras también permite aprovechar las bibliotecas de Tensorflow, teniendo entre muchas ventajas:

- Homogenizar el código para CPU's y GPU's
- Utilizar interfaces amigables con el usuario para generar fácilmente prototipos de modelos para Deep Learning.
- Soportar cualquier arquitectura. Lo que permite construir prácticamente cualquier modelo de Deep Learning, desde una red de memoria hasta una maquina de Turing neural.

En conjunto estos dos elementos permiten acceder a bibliotecas y herramientas de Deep Learning que se ajustan a las necesidad del contenido teórico del presente documento.

3.7.2 Funciones Generadoras

Dada la naturaleza del procesamiento de datos del paquete R, la implementación de un modelo de series de tiempo requiere procesar lotes de series de tiempo que actúen como los datos de entrada y que contengan las objetivos esperados. Estos datos se tienen que procesar de forma independiente a través de lo que se conoce como función generadora, la cual construye a partir de un grupo de parámetros establecidos los lotes de datos que son procesables dentro de los modelos del Keras. Estos parámetros deben contener los indices sobre los cuales se tomaran las referencias en la tabla, una condición de aleatoriedad, el tamaño del lote de datos y las especificaciones de los adelantos, retrasos y unidades consideradas correspondientes para la generación de los lotes.

CAPITULO III: EXPERIMENTOS Y RESULTADOS

4.1 Descripción de las bases de datos

Los datos a utilizarse para los experimentos son valores históricos relacionados con el mercado accionario de la Empresa Google entre las que incluyen el valor accionario de apertura, cierre, ajuste y volumen diarios comprendidos entre las fechas del 3 Enero del año 2007 y el 29 de Junio de 2018. Esta base de datos fue obtenida a partir de la biblioteca "quantmod", disponible directamente en R. Dentro de la base de datos obtenida, la variable de interés principal es la del precio de ajuste de la acción puesto que la misma valora de forma adecuada el comportamiento del precio durante el día y a través de procesos directos de la bolsa de valores contiene con mayor precisión el funcionamiento de la empresa. El comportamiento de esta variable de cierre se muestra en la figura 4.1.

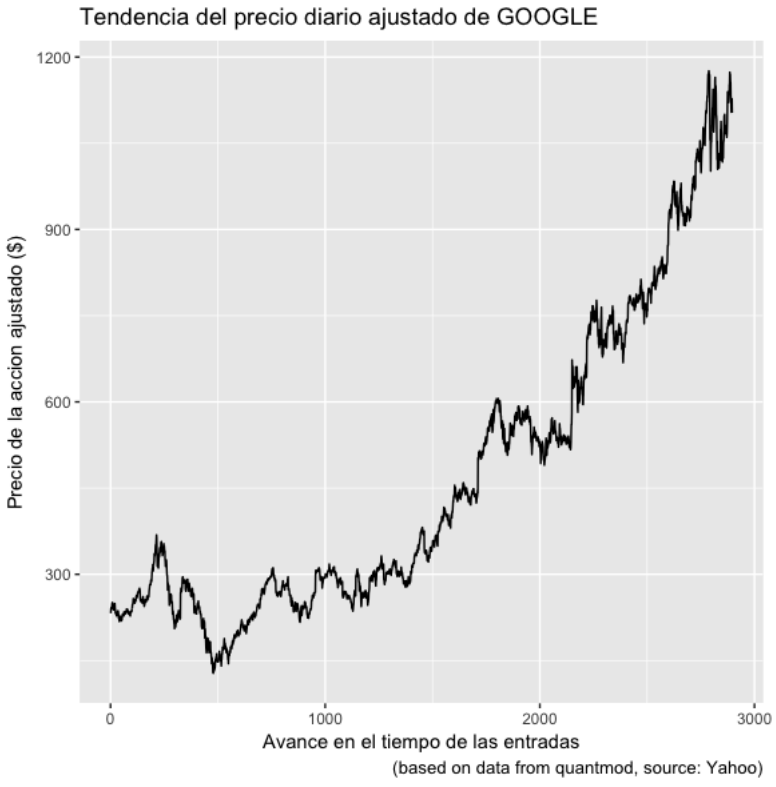

Figure 4.1: Precio diario ajustado de acciones de Google

El resto de variables se tomaran como parte de los datos de entrada en los modelos, las mismas son:

- Precio de Apertura
- Precio más alto
- Precio más bajo
- Precio de cierre
- Volumen de acciones en el mercado
- Rezago de un período del precio ajustado
- Variable binaria de tendencia

Se considera que la tendencia en los datos Google, por ser una empresa considerada de alto precio de cotización en la bolsa tendrá una mayor volatilidad y que por tanto resultará en precios más variables (Stankevicien & Akelaitis, 2014). Es por ello que para este análisis del funcionamiento de las arquitecturas de redes neuronales como modelos de optimización se tomará en cuenta la base de datos segmentada de forma tal que pueda capturarse las regiones mas apropiadas para el funcionamiento de las arquitecturas planteadas.

4.1.1 Loss function tomada en cuenta dentro de los experimentos

Como se ha explicado en la sección teórica del presente documento, en el desarrollo de los experimentos a continuación se busca reducir la distancia existente entre el resultado real y la predicción obtenida a través de una loss función que mejor se adapta a las necesidades de este problema en específico; a partir de *Henrique*, se entiende que la mayoría de autores dedicados a la predicción de valores financieros utilizan (entre otras) como loss function la Mean Absolute Error y la Mean Absolute Percentage Error puesto que tratando de predecir valores reales como lo son los precios y volúmenes, puede directamente medirse el error en la métrica usual (Henrique, Sobreiro, & Kimura, 2018) . En este documento, todos los experimentos se llevarán a cabo con las dos métricas antes mencionadas, ponderando los resultados obtenidos a lo largo de las épocas de entrenamiento para obtener un promedio del error obtenido.

4.1.2 Interpretación de los resultados

Como se ha mencionado, lo que se busca es predecir el precio ajustado de una acción, sin embargo, dado que variable como el volumen, tendencia y precios tienen unidades completamente diferentes se requiere hacer una normalización de la base de datos antes de proceder con los cálculos. Esta normalización se hace para cada una de las variables, incluyendo la del precio ajustado de la acción, lo que provoca que el resultado en términos de loss function no pueda traducirse de forma exacta a lenguaje convencional por lo que sería necesaria una conversión sobre los resultados obtenidos.

Esta conversión consiste en multiplicar el valor obtenido como promedio en la loss function por la desviación estándar del conjunto de datos, a partir de lo cual los resultados pueden entenderse en el lenguaje convencional como la cantidad de dólares con la cual estamos fallando en nuestras predicciones:

$$\text{Dólares de falla} = Loss * std(\text{P. Ajustado})$$

4.1.3 Regiones de similar crecimiento

Una primera estrategia de optimización, consiste en tomar las regiones de la base de datos cuyos comportamientos sean similares y separar estos datos como los conjuntos de entrenamiento, validación y prueba respectivamente. Por la tendencia creciente que tiene la base de datos, se disponen de los siguientes períodos con las especificaciones del conjunto de datos seleccionados respectivamente:

1. Conjunto de Entrenamiento.- Los datos comprendidos entre el 6 de Noviembre de 2015 y el 1 de Mayo de 2017.

2. Conjunto de Validación.- Los datos comprendidos entre el 11 de Mayo de 2012 y el 12 de Diciembre de 2013.

3. Conjunto de Prueba.- Los datos comprendidos entre el 25 de Noviembre de 2008 y el 22 de Diciembre del 2009.

Cuyo comportamiento se puede visualizar en la grafica 4.2

4.1.4 Región Estable

Por la complejidad de la predicción del modelo, una selección mas adecuada del conjunto de datos necesarios para el entrenamiento y validación, resulta de tomar la región más estable dentro del total de

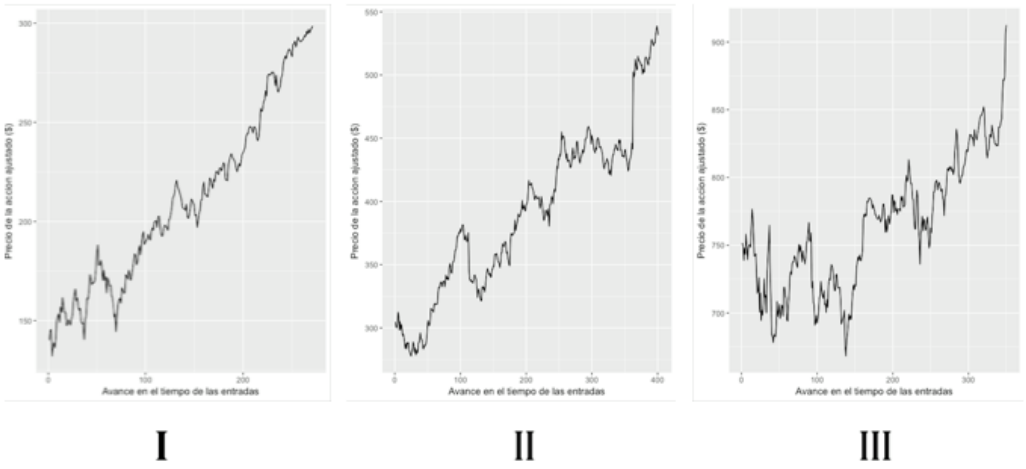

Figure 4.2: Donde el primer grupo de datos (I) representa el conjunto de prueba; el segundo grupo de datos (II) representa el coNjunto de validación y finalmente el tercer grupo de datos (III) representa el conjunto de entrenamiento

la base de datos. Por esta razón se toman los conjuntos de datos a partir de un subconjunto de la base original, considerando la región más estable, comprendida entre el 22 de Diciembre del año 2009 y el 15 de Diciembre del año 2012. Este subconjunto de datos puede visualizarse en la figura 4.3

4.1.5 Tomando una línea base: Método Naive

Previo al análisis de los resultados con las arquitecturas de redes neuronales, debe pensarse que un modelo tiene que resultar con mejores características de predicción que aquello que consideremos trivial; es por ello que incorporar un método de predicción trivial como aquel que vaticina al día siguiente el mismo resultado del valor ajustado de la acción para el día anterior puede ser un buen inicio al momento de medir y comparar el modelo.

Es por ello que considerando el conjunto de datos de la región estable, se genera una función capaz de medir el error de un método de predicción cuya salida sea exactamente igual al valor obtenido el día anterior; este método será incorporado como Naive en la tabla de resultados y permitirá tener una idea más clara de lo que esta ocurriendo.

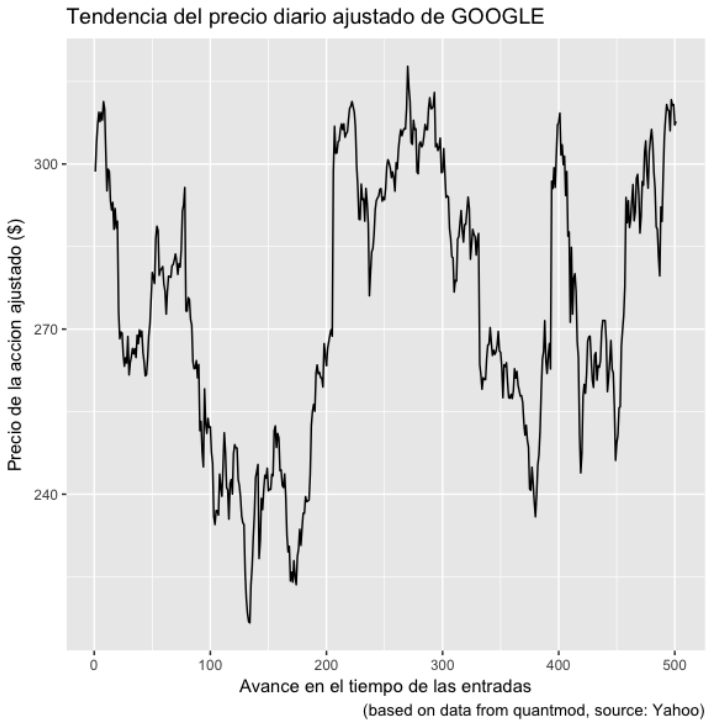

Figure 4.3: Precio diario ajustado de acciones de Google en la región estable

Es menester señalar que los precios de las acciones no fluctúan tanto entre un día y otro, sino que sus cambios se evidencian con el paso del tiempo (que es lo que se considera para el entrenamiento) por lo que un modelo que solo repite la predicción del día anterior puede ser mejor en términos de loss function puede ser mejor, pero no resulta práctico para una aplicación en la práctica.

4.2 Feed Forward Network

Considerando la primera arquitectura descrita en el presente documento, el primer modelo de predicción es una red completamente conectada (FeedForwardNetwork). En este caso se escoge una capa, con un total de 28 neuronas además de la capa de salida con una sola neurona. Esta arquitectura se escoge considerando otros modelos de algunas áreas en predicción de tiempo y a su vez optimizando la cantidad de neuronas en la capa. Por su naturaleza, en esta arquitectura todos los datos de entrada resultarán como insumos para la arquitectura, es por ello que la dimensión de entrada depende directamente de la cantidad de variables y el rezago en el tiempo.

A través de una repetición constante del modelo, modificando los parámetros externos y propios del modelo, se encuentra que el modelo con los mejores resultados para la arquitectura aquí descrita son los siguiente:

- Rezago = 5
- Paso = 2

4.2.1 Resultados obtenidos en las regiones de similar crecimiento

Con los parámetros escogidos, se obtiene el siguiente gráfico 4.4 de las valores de loss function, a lo largo de las diferentes épocas:

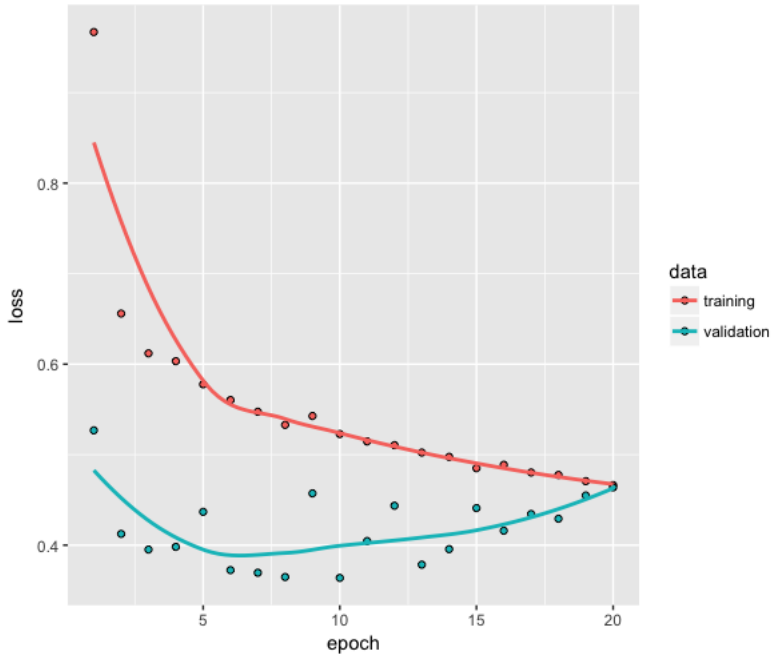

Figure 4.4: Loss functions de la primera arquitectura en regiones de similar comportamiento

De este modelo se obtienen valores para la loss de 0.5507, validation loss de 0.4179 y en la evaluación con los datos de prueba un valor de 1.46.

4.2.2 Resultados obtenidos en la región estable

Sobre este conjunto de datos, dado que los valores son continuos se toma como conjunto de entrenamiento y de validación las primeras 400 entradas, en dos conjuntos parejos, cada uno con un total de 200 entradas. De este entrenamiento y de su correspondiente validación se obtienen los resultados que se muestran a continuación:

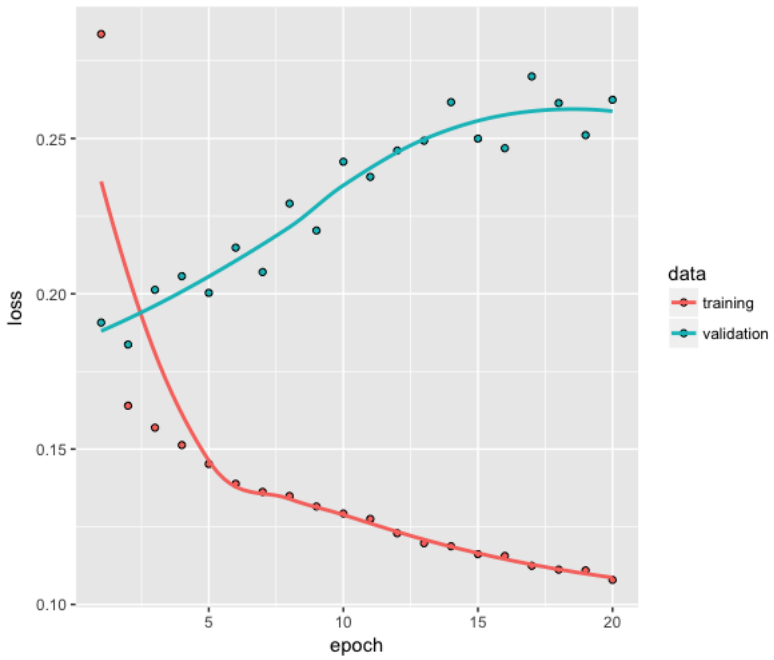

Figure 4.5: Loss functions de la primera arquitectura en region estable

Se encuentra entonces que los valores de loss function a lo largo de las etapas se han reducido de forma significativa. Esto se debe a que se encuentra un patrón mas repetitivo en el flujo de los datos dentro de esta región estable. No obstante, los valores tienen una tendencia creciente, lo cual es causado por problemas de sobre-entrenamiento en el modelo.

4.3 LSTM network

El segundo modelo que se toma en cuenta en el presente documento es una red LSTM como se describió en el capítulo anterior. Se recuerda que este modelo puede ofrecer mejores resultados pues toma en cuenta información del pasado como remanente que se introduce en la predicción del presente. Esta arquitectura contiene un sola capa LSTM interna con un total de 32 neuronas así como la capa exterior

de salida que arroja un out-put de un solo valor.

Se procede de la misma forma que con la arquitectura anterior, modificando uno por uno los parámetros externos, encontrando que los mejores resultado se obtienen a partir de los siguiente parámetros:

- Rezago = 4
- Paso = 2

Por conveniencia, se usa la base de datos dentro de la **región estable** como se describió en la arquitectura anterior, tomando los mismos conjuntos de datos para obtener los resultados que se muestran en la figura 4.6

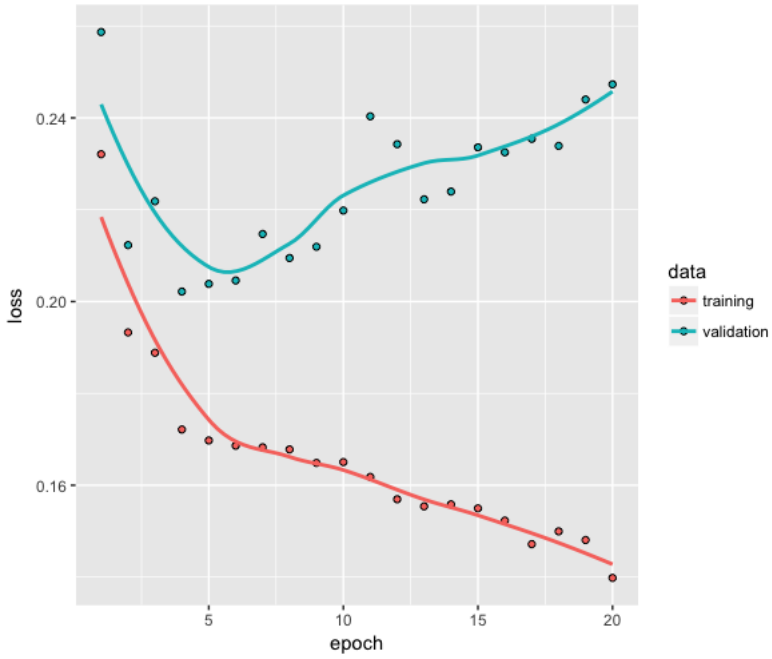

Figure 4.6: Loss functions de la segunda arquitectura en región estable

A través del gráfico se puede identificar que los valores obtenidos para las loss functions mantienen un rango de valores cercanos (aunque en promedio mas bajos) que los obtenidos por la arquitectura FFN, sin embargo, vale rescatar de ahora los valores no tuvieron un crecimiento exagerado como ocurrió con la arquitectura anterior.

4.3.1 Ligera modificación del modelo anterior

Una de las variantes actuales de la red LSTM es la red GRU, de la cual no se ha hecho detalle dentro del presente documento pero la cual funciona de forma muy similar a la LSTM. Esta red incluye ciertos componentes de activación o no activación de las memorias de los cálculos de las épocas pasadas mejorando ligeramente la red.

Los parámetros que se utilizan dentro del contexto externo del problema, son similares a los utilizados en la red LSTM en esta misma sección al igual que el conjunto de datos, tomados desde la **región estable**. Los resultados obtenidos se muestran en la figura 4.7

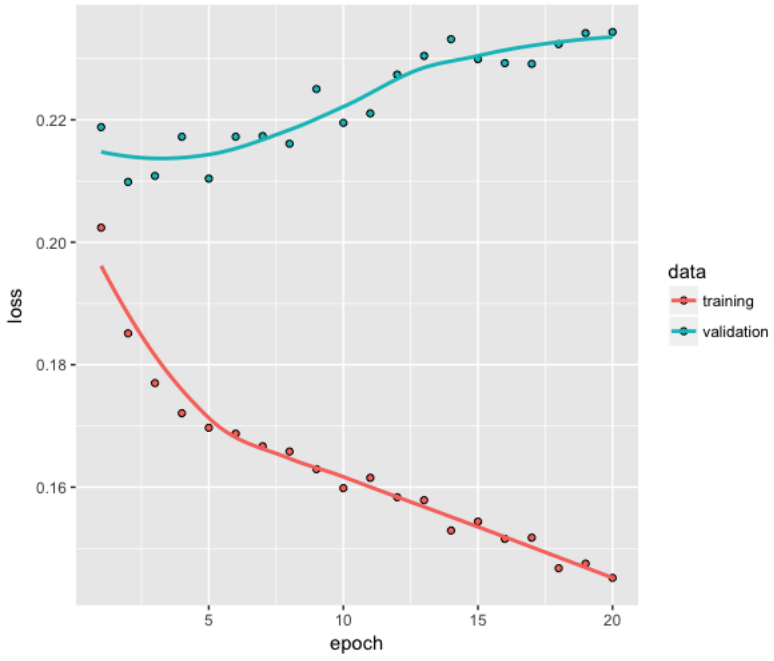

Figure 4.7: Loss functions de la segunda arquitectura (red GRU) en región estable

4.4 Doble LSTM

En esta arquitectura se introduce el recurrent dropout. Además, con el objetivo de mejorar el procesamiento de los datos se hace uso de dos capas internas del tipo LSTM: Una de ellas con 64 neuronas internas y la segunda con un total de 32 neuronas. Al final de la arquitectura se utiliza la acostumbrada capa de salida.

Una vez más, el proceso de elección de los parámetros externos se hizo probando uno por uno los resultados, con lo que se toma en para este caso las valores como sigue:

- Rezago = 4
- Paso = 2

A partir de los cuales se obtiene los resultados que se muestran en la figura 4.8

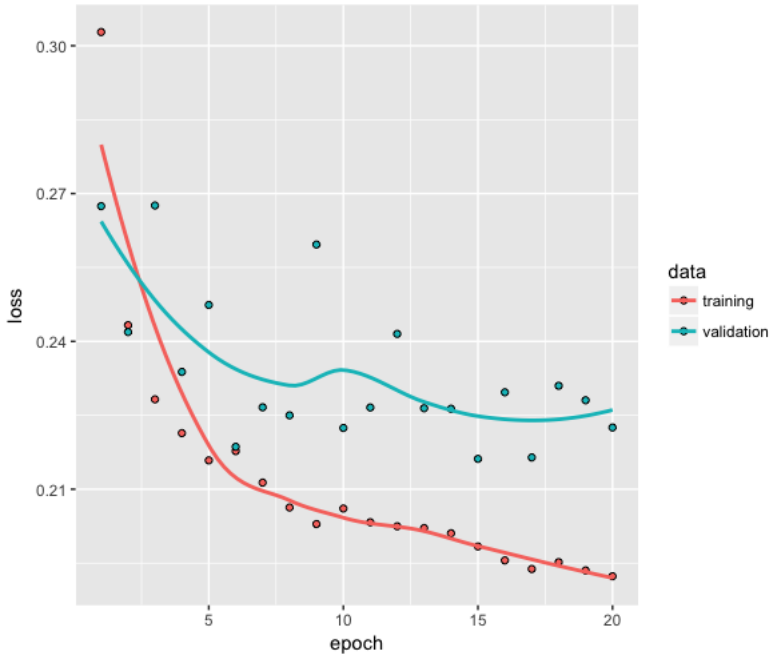

Figure 4.8: Loss functions de la tercera arquitectura (doble LSTM) en región estable

4.5 Comparación entre modelos de la región estable

A partir de los 4 modelos programados y desarrollados en el presente capítulo que toman en cuenta el conjunto de datos obtenido de la **región estable** y tomando como referencia el promedio como medida, se obtiene el cuadro 4.1 de resultados del valor de loss function (entrenamiento y validación) para cada uno de ellos:

Loss and Validation loss para cada uno de los modelos		
Arquitectura	Loss Function(MAE)	Validation Loss(MAE)
Feed Forward Network	0.1367	0.2315
LSTM	0.1656	0.2253
GRU	0.1629	0.2231
Doble LSTM	0.2116	0.2337

Table 4.1: Loss functions para los modelos

El modelo más adecuado va a ser el que provea mejores resultados en términos de mayor precisión en el vaticinio del precio ajustado de la acción. Sin embargo, se tiene que hacer uso del conjunto de datos prueba para poder medir la precisión de los modelos pues no se puede concluir a partir de la loss function de validación si un modelo es mejor que otro. Como los cuatro modelos de la región estable presentados en este documento ocupan el mismo conjunto de datos de prueba, se puede comparar que tan precisos resultaron ser los modelos, como se detalla en el cuadro 4.2

Loss obtenida a través de predict_generator		
Arquitectura	Loss Function(MAE) para conjunto de prueba	Valor en Dólares ($)
Feed Forward Network	0.2785	6,78
LSTM	0.3248	7,91
GRU	0.3203	7,80
Doble LSTM	0.3055	7,44
Naive	0.2208	5,37

Table 4.2: Resultados para todos los modelos

4.6 Conclusiones

El comportamiento humano siempre será difícil de entender o predecir (Wallace, 2009); en el ámbito de las finanzas, es éste comportamiento el que determina el precio de las acciones, el volumen de ventas y el accionar de las mismas empresas lo que hace de la predicción y aprendizaje del mismo una actividad sumamente complicada. En este documento se ha pretendido generar un acercamiento a algunos métodos matemáticos y computacionales que permitan un vaticinio y predicción de los valores de activos financieros a través de la comparación de diferentes arquitecturas utilizadas en deep-learning dando cuenta de la difícil actividad a la que se aplica su teoría.

Una vez comparadas las distintas arquitecturas del deep-learning en conjunto con el método Naive,

se encontró que los resultados obtenidos a partir de éste último resultaron mejores que aquellos obtenidos por medio de arquitecturas de redes neuronales, según como puede observarse en la tabla 4.2. El hecho de que las predicciones realizadas a través del método Naive resulten mejores obedece a dos hechos fundamentales:

1. Los precios entre las acciones no tienden a cambiar tanto entre un día y otro salvo intervenciones externas.

2. Las variaciones del precio dependerá de forma directa del comportamiento humano que obedece a estímulos emocionales, difíciles de aproximar mediante algoritmos matemáticos por la complejidad que estos representan: Decisiones gubernamentales, ventas de grandes grupos de acciones, caídas en la economía entre otros hechos propios del comportamiento humano

Por otro lado, a partir de este documento puede concluirse que entre los modelos de redes neuronales analizados, **el mejor resultó ser el de FeedForward** en razón de que se utilizó como variables de predicción los precios de apertura, cierre, alto y bajo que de alguna manera permitieron que esta arquitectura más robusta obtenga mejores resultados por sobre las de recurrencia pues la dinámica de los datos no resultó ser la más apropiada y razón por la cual se recomendaría su inclusión en futuros análisis del mismo tema.

REFERENCES

Chollet, F., & Allaire, J. (2018). *Deep learning with r*. Manning Publications Co.

Cockburn, I., Henderson, R., & Stern, S. (2017, 12). The impact of artificial intelligence on innovation.

Escobar, V. (2002). Análisis de datos con redes neuronales aplicadas al diagnóstico de la solvencia empresarial. *Cuestiones Económicas*(3), 79-132. Retrieved from https://www.bce.fin.ec/cuestiones_economicas/images/PDFS/2002/No3/Vol.18-3-2002VICTORESCOBAR.pdf

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. (http://www.deeplearningbook.org)

Griffin, J. M., Nardari, F., & Stulz, R. M. (2004, September). *Stock market trading and market conditions* (Working Paper No. 10719). National Bureau of Economic Research. Retrieved from http://www.nber.org/papers/w10719 doi: 10.3386/w10719

Górriz, C., & Gargallo, A. (2004, 01). Impacto de las tecnologías de la información en la productividad de las empresas españolas.

Henrique, B., Sobreiro, V., & Kimura, H. (2018). Stock price prediction using support vector regression on daily and up to theminute prices. *The Journal of Finance and Data Science*. Retrieved from https://www.sciencedirect.com/science/article/pii/S2405918818300060 doi: 10.1016/j.jfds.2018.04.003

LeCun, Y., Bengio, Y., & Hinton, G. (2015, 05). Deep learning. , *521*, 436-44.

Marwala, T. (2013). *Economic Modeling Using Artificial Intelligence Methods*. London: Springer, London.

Michalski, R., Carbonell, J., & Mitchell, T. (2013). *Machine learning: An artificial intelligence approach*. Springer Berlin Heidelberg. Retrieved from https://books.google.com.ec/books?id=-eqpCAAAQBAJ

Stankevicien, J., & Akelaitis, S. (2014). Impact of public announcements on stock prices: relation between values of stock prices and the price changes in lithuanian stock market. *Procedia - Social and Behavioral Sciences*(156), 538-542. Retrieved from https://www.sciencedirect.com/science/article/pii/S187704281406056X#bbib0005

Wallace, W. (2009). *Principles of scientific sociology*. Transaction Publishers. Retrieved from https://books.google.com.ec/books?id=Tr_Lo4DiI7gC