**UNIVERSIDAD NACIONAL AGRARIA**

**LA MOLINA**

**ESCUELA DE POSGRADO**

**MAESTRÍA EN ESTADÍSTICA APLICADA**



**ESTADISTICA ACTUARIAL**

**“SERIES DE TIEMPO CON REDES NEURONALES”**

Integrantes:

Enrique Figueroa

Jaime Gómez

Roberto León Leyva

ASESOR:

Jesús Eduardo Gamboa Unsihuay

La Molina, 2019

Contenido

[I. INTRODUCCIÓN 2](#_Toc12982341)

[II. MARCO TEORICO 3](#_Toc12982342)

[Entendiendo las Redes Neuronales, NN 3](#_Toc12982343)

[Linea de Tiempo del Desarrollo de las Redes Neuronales, NN 5](#_Toc12982344)

[Esquema básico de una neurona 6](#_Toc12982345)

[Funciones de Activación 7](#_Toc12982346)

[Entendiendo el funcionamiento de una red neuronal 8](#_Toc12982347)

[Taxonomía de las Redes Neuronales. 10](#_Toc12982348)

[Redes Neuronales LSTM 14](#_Toc12982349)

[III. APLICACIÓN DE REDES NEURONALES 16](#_Toc12982350)

[Objetivo General 16](#_Toc12982351)

[Conociendo la Serie de Tiempo. 16](#_Toc12982352)

[Analizando la serie de tiempo con el modelo Holt Winters. 20](#_Toc12982353)

[Analizando la serie de tiempo con el modelo ARIMA. 24](#_Toc12982354)

[Analizando la serie de tiempo con el modelo GARCH. 26](#_Toc12982355)

[Analizando la serie de tiempo con redes neuronales LSTM 31](#_Toc12982356)

[IV. CONCLUSIONES 35](#_Toc12982357)

[V. REFERENCIAS BIBLIOGRAFICAS 36](#_Toc12982358)

[VI. ANEXOS 37](#_Toc12982359)

# INTRODUCCIÓN

Las Redes Neuronales, NN, constituyen en estos días uno de métodos más potentes y de desarrollo en las técnicas de Machine Learning, se les utiliza para un sinnúmero de tareas desde la clasificación hasta la predicción, sus diversos métodos y variantes constituyen uno de los elementos más ricos en investigación.

El esquema básico de una NN son neuronas o perceptrones interconectados que utilizan una función de activación para aportar la no linealidad del modelo y través de métodos de la técnica de Gradiente Descendente y backpropagation ajustar los pesos de las interconexiones, similares a una regresión múltiple, de modo que se minimice los valores de una función objetivo denominada loss. En general los elementos de una NN se pueden ordenar a través de capas y cuando se hace utilizan varias capas aparece la arquitectura denominada “Deep Learning”.

En el presente trabajo se utilizará un tipo especial de NN denominado LSTM, Long Short-Term Memory, una subcategoría de las recurrentes NN, RNN. Las RNN del tipo LSTM poseen la característica de utilizar información de una secuencia para explicar valores actuales, por lo que es evidente su adecuación a los modelos de Series de Tiempo.

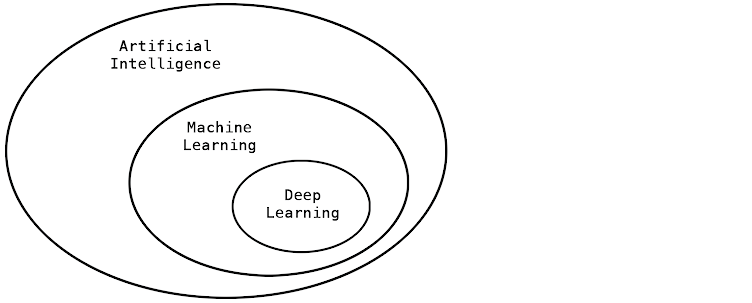
A modo de ejemplo se utilizarán las cotizaciones diarias de APPLE, una de las mayores empresas globales se aplicaran y compararan los resultados de las NN del tipo LSTM con los tradicionales técnicas de series de tiempo Holt Winters, ARIMA y GARCH.

# 

# MARCO TEORICO

Entendiendo las Redes Neuronales, NN

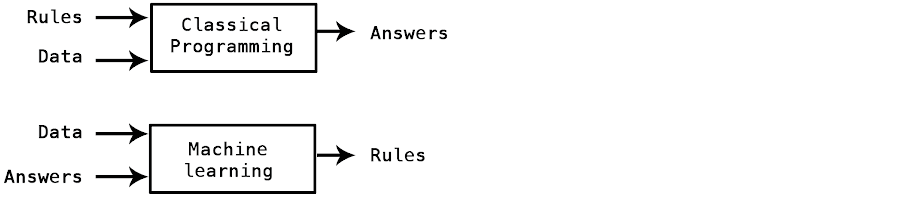
Las NN son técnicas de machine learning, y estas a su vez un subcampo de la inteligencia artificial. Esta relación se muestra en el siguiente gráfico tomado del libro de Chollet[[1]](#footnote-1) “Deep Learning with R”. El Deep Learning es una arquitectura de NN donde las neuronas o perceptrones son ordenados en capas.



Lo anterior nos obliga a conceptualizar brevemente lo que se entiende por Inteligencia Artificial, un campo que surgió en la década de 1950 con los trabajos de Alan Turing y su teoría del autómata y cuyas ramificaciones se siguen desarrollando hasta nuestros días. Una definición concisa es “el esfuerzo por automatizar tareas de naturaleza intelectuales que son realizadas por humanos”[[2]](#footnote-2).

Los primeros trabajos de Inteligencia Artificial, AI, se centraron en la programación de reglas que pudieran resolver problemas a este enfoque se le denominó symbolic AI, estas técnicas resultaron útiles para resolver problemas como jugar el ajedrez, sin embargo, no dieron resultados para resolver problemas difusos como la clasificación.

Un conjunto de nuevas técnicas se desarrollaron para resolver estos problemas y el punto de partida fue que se podían “entrenar” algoritmos para que encontraran estructuras estadísticas que les permitiera identificar reglas para automatizar tareas como la clasificación. La comparación de ambos enfoques se presenta en la siguiente figura[[3]](#footnote-3).



Una red neuronal es una técnica de Machine Learning que crea una proyección o mapeo no lineal entre un una entrada (multidimensional) y la salida(multidimensional pero menor a la entrada). En términos más coloquiales, Anderson resume este hecho en la frase[[4]](#footnote-4):

*“Las redes neuronales artificiales son una clase de estadística no lineal para amateurs”.*

Existe una confusión común al pensar de que las redes neuronales son una forma de modelar la forma de cómo aprende el cerebro humano, esto no tiene evidencia y el estudio de estos fenómenos corresponde a las ciencias cognitivas. El nombre de Red Neuronal, NN, tiene una inspiración en las neuronas y cómo se conectan estas unas con otras para formar redes.

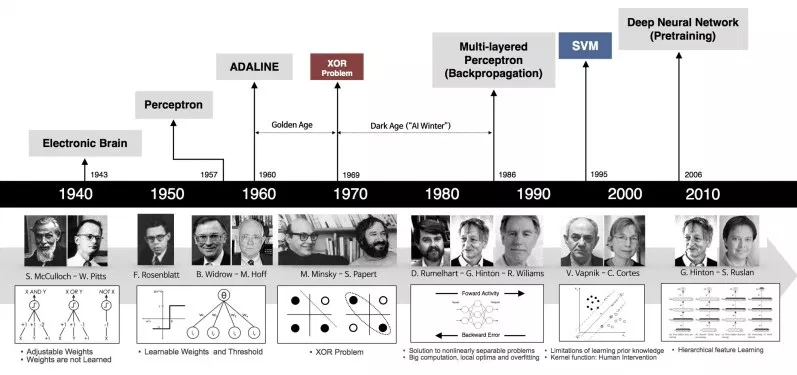
La siguiente figura es una ironía de lo que no es una red neuronal y es conveniente que se destierre esta concepción errónea que le añade al concepto misticismo y obscurantismo y lo aleja de lo que realmente es: una técnica de machine learning[[5]](#footnote-5).



Linea de Tiempo del Desarrollo de las Redes Neuronales, NN

La siguiente figura[[6]](#footnote-6) muestra los principales desarrollos que contribuyeron al desarrollo de las redes neuronales. En la década de los 60 del siglo pasado se generó una expectativa muy grande sobre los problemas que resolverían estas técnicas, sin embargo, los resultados no llegaron a satisfacer las inversiones en investigación y se produjo lo que se denomina la “era de hielo de la inteligencia artificial” que duró más de 25 años y recién a mediados de los 90 se comenzaron los nuevos desarrollos.

El despegue se retomó cuando comenzaron a implementarse las técnicas para la clasificación de imágenes y se ganaron concursos de identificación de imágenes del reto ImageNet, que contenía alrededor de 1000 tipos de imágenes con 1.4 millones de imágenes. Hasta 2011 los algoritmos ganadores lograban una precisión del 74.3%, sin embargo, en 2012 un equipo liderado por Alex Krizhevsky y asesorado por Geoffey Hinton logró el sorprendente 83.6% y para el 2015 logró el 96.4% estos resultados generaron el nuevo auge de las técnicas de redes neuronales. Desde 2012 las redes neuronales convolucionales se han convertido en el estándar para las tareas de visión por computadora.

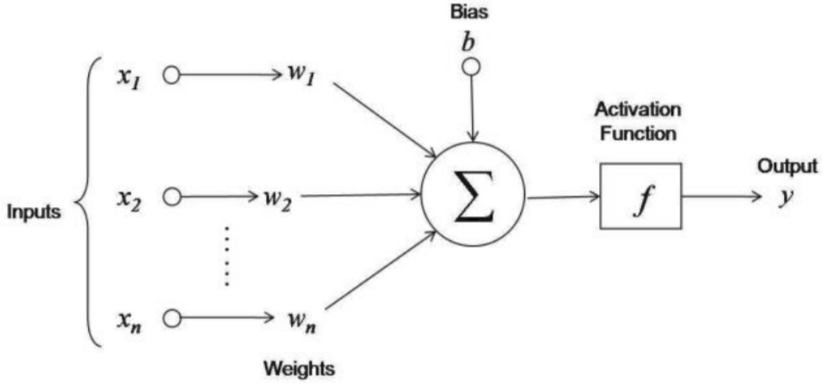
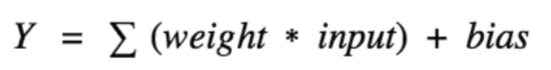


El resurgimiento de las Redes Neuronales es posible también en el contexto de:

1. Mayor potencia de Hardware y la utilización de las capacidades de las tarjetas gráficas para resolver problemas matriciales y proporcionar procesamiento paralelo a los algoritmos en especial las provenientes de la marca NVIDIA.
2. Algoritmos, la inclusión del gradiente descendente y del backpropagation junto con el Deep Learning reemplazaron prácticamente a las técnicas de SVM y Random Forest.
3. Mejores métodos de optimización como Adam, utilizado en los esquemas de Deep Learning.

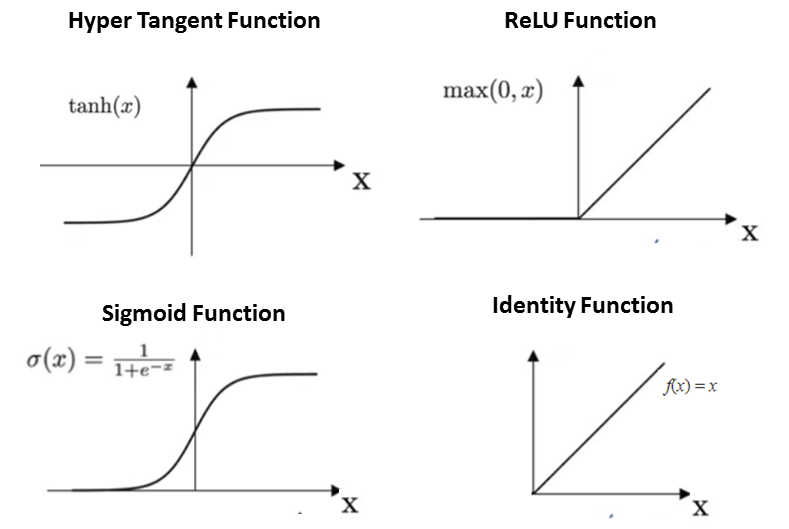
Esquema básico de una neurona

El esquema básico se presenta a continuación, en él una neurona o perceptrón funciona como un modelo de regresión múltiple donde lo que se trata de determinar son los pesos correspondientes a cada entrada, la diferencia es la utilización de la función de activación lo que proporciona la no linealidad.



Funciones de Activación

Las principales funciones de activación se presentan en la siguiente figura. En la práctica la más utilizada es la función ReLu. Esta es la responsable de añadir la no linealidad a los esquemas de las redes neuronales.

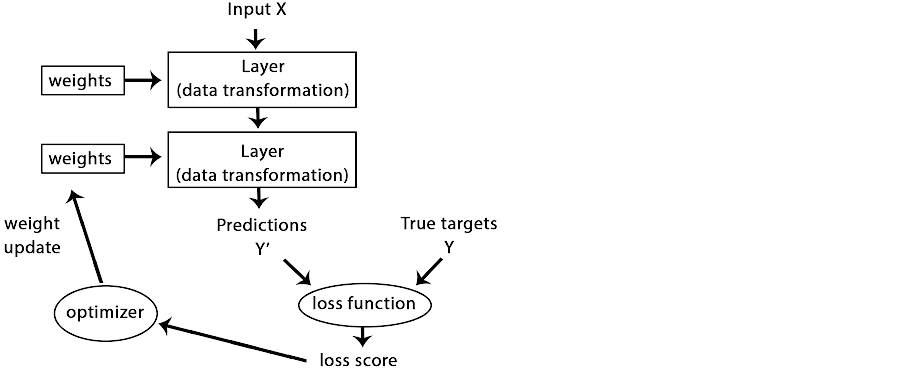


Entendiendo el funcionamiento de una red neuronal

El funcionamiento de una red neuronal se fundamenta en:

1. La determinación de los pesos *w* de las relaciones entre neuronas (interacción). En este contexto “aprendizaje” es entendido como encontrar los pesos de todas las neuronas agrupadas en capas de modo que se ajusten lo mejor posible a los *targets* (valores respuestas).
2. La función *Loss* o función objetivo que permite medir la brecha entre las estimaciones y los valores calculados.
3. El optimizador que implementa el algoritmo de *backpropation* ajustando los pesos de manera que la función *loss* se reduzca en cada iteración, utiliza internamente el gradiente descendiente para ajustar los pesos. Este elemento es el núcleo de la técnica de red neuronal y se ejecuta dependiendo de la arquitectura varias miles de veces y posibilita que los valores predichos se ajustan muy bien a los observados.

El siguiente gráfico muestra el ciclo de ejecución de los elementos descritos[[7]](#footnote-7).



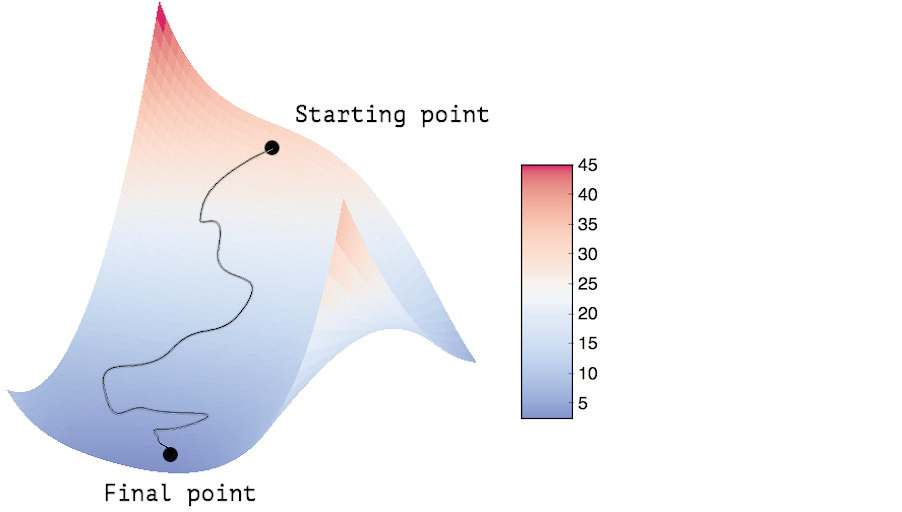
En redes neuronales se utilizan arreglos multidimensionales y se le denominan tensores o en inglés *tensor* de esta manera un vector es un 1D tensor, una matriz un 2D tensor y un arreglo de 3 dimensiones un 3D tensor y así sucesivamente.

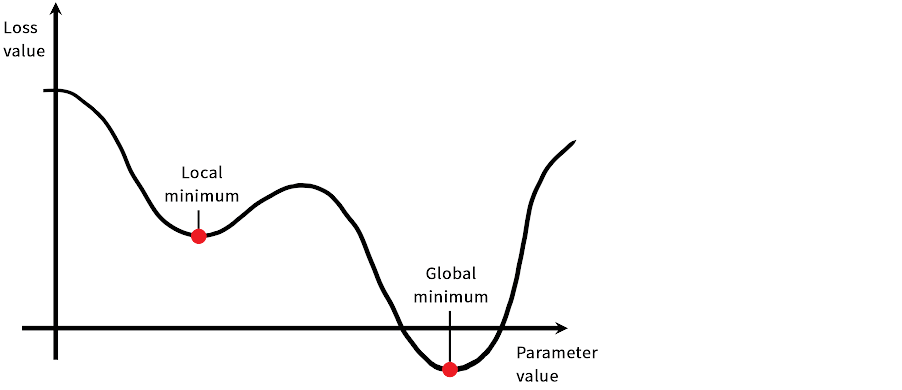
El concepto de Gradiente Descendiente o *Stochastic Gradient Descent* en inglés, es clave para la optimización de los pesos, para una red de N variables el resultado es un polinomio de grado N, esto no se puede resolver analíticamente para una red con cientos, miles o millones de neuronas o nodos. En este caso se sigue un algoritmo con los siguientes pasos generales:

1. Determinar un batch de entrenamiento de X y sus correspondiente Y.
2. Correr la red y determinar los valores predichos para Y: Y\_pred
3. Calcular la función *loss* de desajuste entre Y e Y\_pred.
4. Calcular el gradiente de la función *loss* de los parámetros de la red.
5. Ajustar levemente los parámetros en oposición del gradiente en *w – step \* (gradient)*

Un elemento importante es la determinación del *step* valores pequeños o muy grandes pueden llevarnos a que encontrar mínimos relativo o a que la búsqueda no converja.

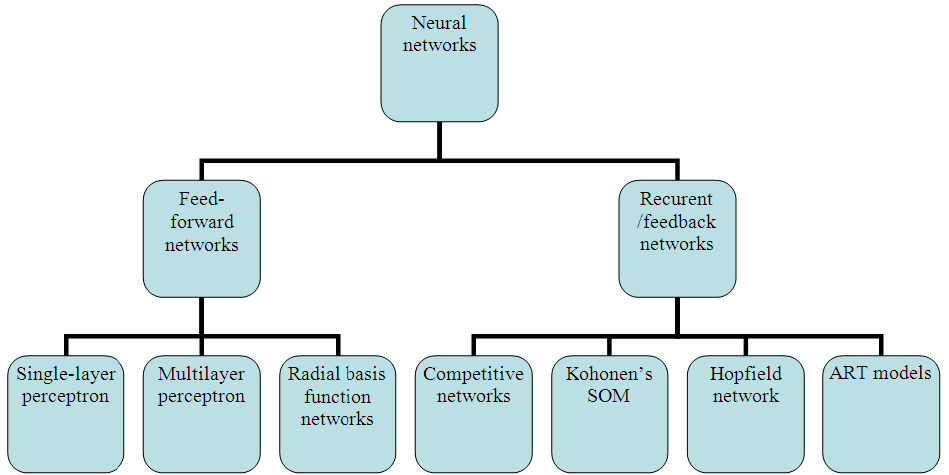
Gráficamente una función de dos variables se puede graficar como una superficie, tal como se presenta en el gráfico siguiente, y el método descrito servirá para ir de un punto de mayor nivel a otro de menor nivel. Analizar una sola variable nos puede dar una idea de lo que se refiere a mínimo local y global[[8]](#footnote-8).

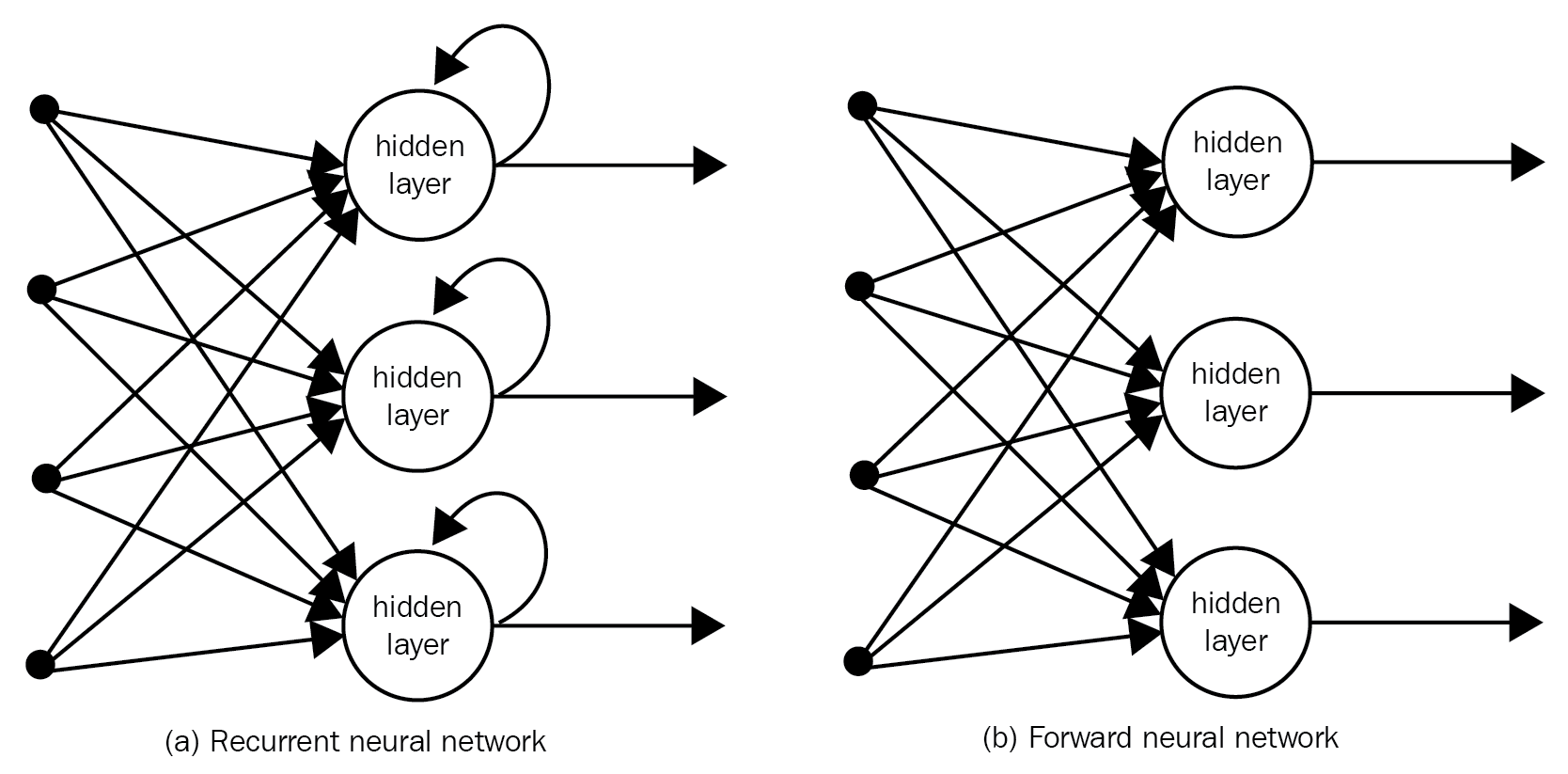




Taxonomía de las Redes Neuronales.

1. Con base al feedback: direccional y recurrente, el siguiente gráfico muestra una clasificación realizada por Gardner y Dorling en 1998.

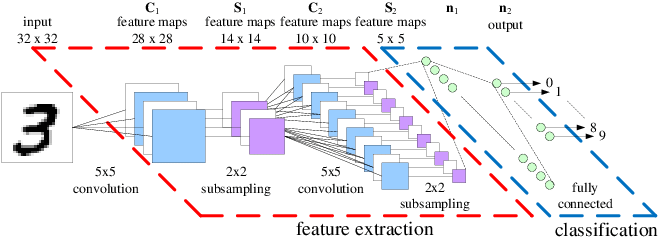




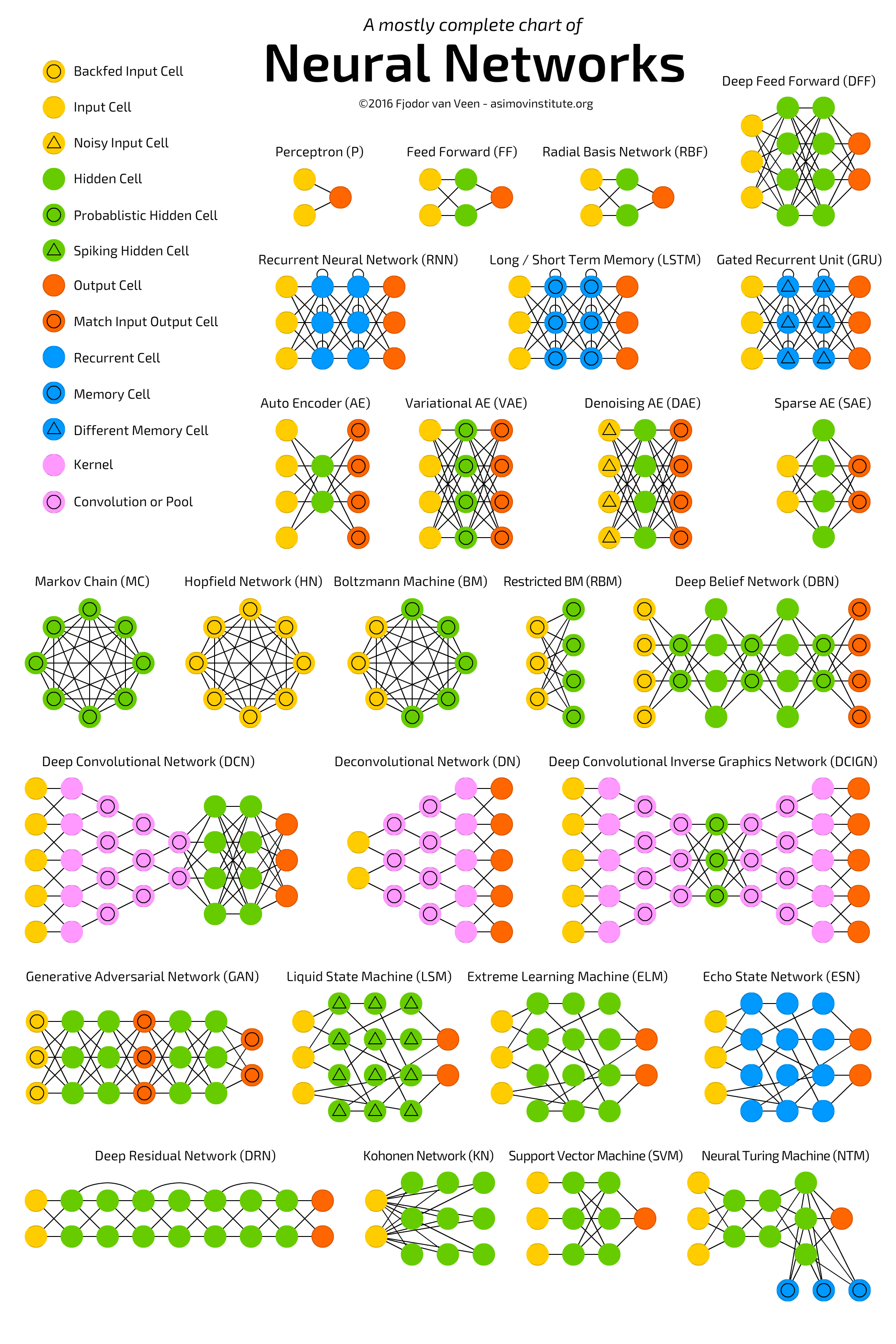
Respecto a las FNN se han desarrollado los modelos: Autoencoder, Probabilistic, Time Delay y Convolutional.

Respecto a las RNN se han desarrollado más modelos entre los que se encuentran: SOM (Self Organization Maps), BAM (Bidirectional Associate Memory) y LSTM (Long Short Term Memory) que es el modelo que se utilizará para modelar la red neuronal.

1. Por el número de capas: Shallow (ligero) y Deep (profundo). El siguiente gráfico muestra un ejemplo de Deep Learning aplicado al reconocimiento de dígitos en forma manuscrita.



1. Otras arquitecturas, el siguiente esquema muestra otras posibilidades en las topologías de las redes neuronales[[9]](#footnote-9).

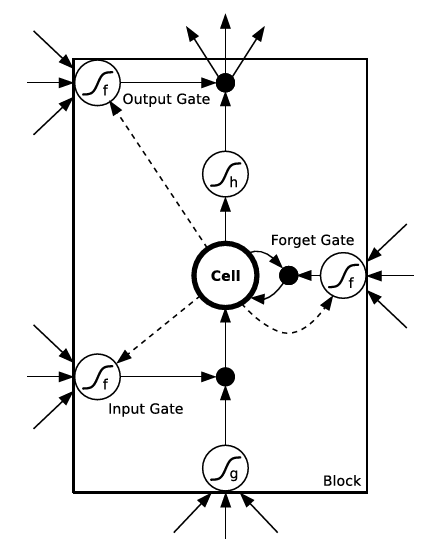


Redes Neuronales LSTM

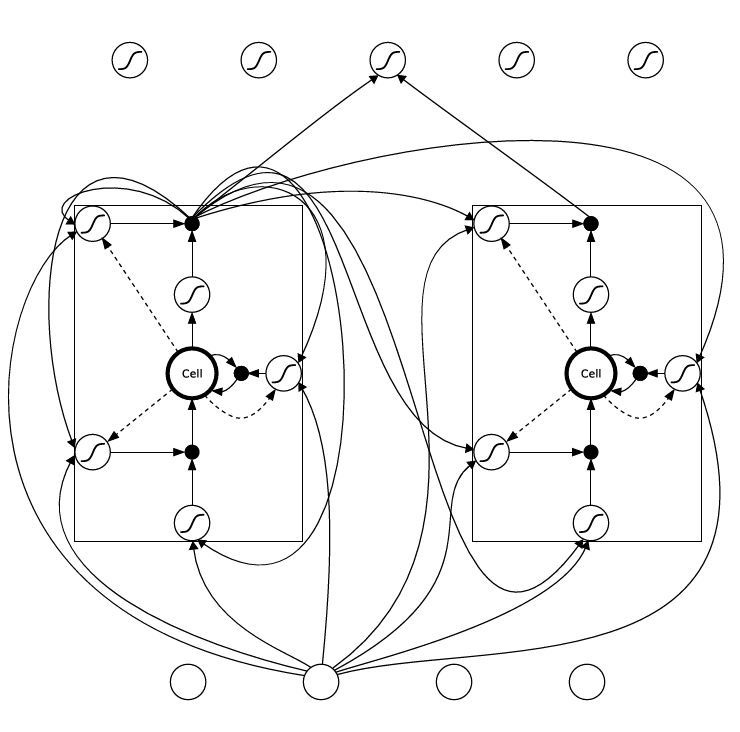
Es un tipo de RNN y el que se utilizará para realizar el estudio, el gráfico que se muestra a continuación muestra la arquitectura de una bloque de procesamiento[[10]](#footnote-10).

Cada bloque de procesamiento contiene tres compuertas de no linealidad que colectan información y controlan la activación vía multiplicación (círculos negros). Las compuertas *input* y *output* multiplican el *input* y *output* de la célula mientras que la compuerta *forget* multiplica el estado previo de la célula No se aplican funciones de activación dentro de la célula. La función de activación de la compuerta ‘f’ es generalmente la función sigmoide logística de modo que la salida de la compuerta de activación está entre 0 (compuerta cerrada) a 1 (compuerta abierta). Las funciones de activación de ingreso y salida de la célula (‘g’ y ‘h’) son generalmente la tahh o sigmoide logística, aunque en algunos casos es la función identidad los pesos de las conexiones entre la célula y las compuertas se muestran en líneas entrecortadas. Las únicas salidas desde el bloque al resto de la red emanan desde la multiplicación de la compuerta *output*.

La red LSTM consiste que se muestra en la figura siguiente consiste de 4 unidades de input, una capa oculta compuesta de 2 bloques LSTM y 5 unidades de salida. Se puede apreciar que cada bloque contiene 4 inputs y solamente una salida.



LSTM: Funcionamiento de un bloque de procesamiento



LSTM: Ejemplo de esquema de funcionamiento entre bloques

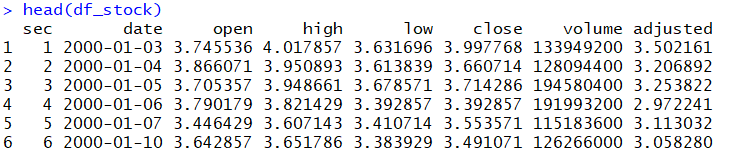
# APLICACIÓN DE REDES NEURONALES

Objetivo General

Aplicar la redes neuronales LSTM para modelar y predecir las cotizaciones las cotizaciones diarias de Apple, uno de los gigantes de la tecnología y líder indiscutible del sector.

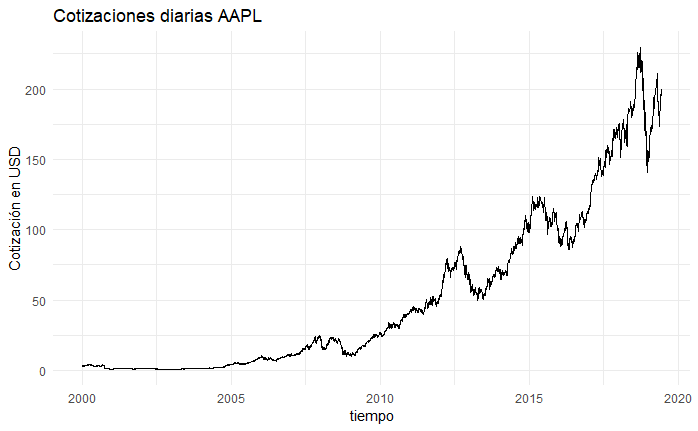
Conociendo la Serie de Tiempo.

La serie de tiempo a analizar corresponde al periodo enero 2000 a junio 2019, correspondiendo a 4901 observaciones. La data generada tiene como fuente a Yahoo Finance, la estructura de la información descargada se muestra en el siguiente cuadro.

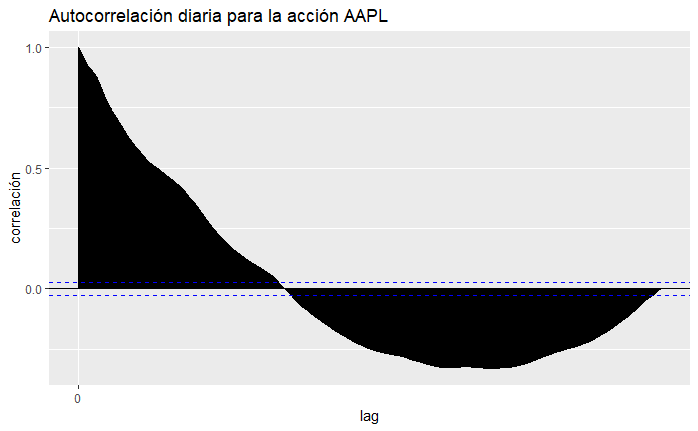


Dado que en tan largo periodo de tiempo se han producido entrega de dividendos, splits y otras acciones que modifican el precio de cierre de la cotización se utilizará la columna “adjusted” que corresponde al precio ajustado de la acción y que incorpora los movimientos descritos previamente.

El gráfico de la serie de tiempo se presenta a continuación y se aprecia la historia de los últimos 18 años y medio en términos generales se podría indicar que existe hasta el 2006 un crecimiento lento luego comienza a aumentar la cotización hasta el 2008 donde es afectado por crisis financiera hasta el 2009 y luego se recupera para seguir una tendencia creciente pero con fuertes fluctuaciones. En términos generales se podría concluir que existe una tendencia creciente.



Revisando la autocorrelación



Se observa que existe una fuerte autocorrelación a lo largo de toda la serie de tiempo y también se puede sospechar de la presencia de tendencia.

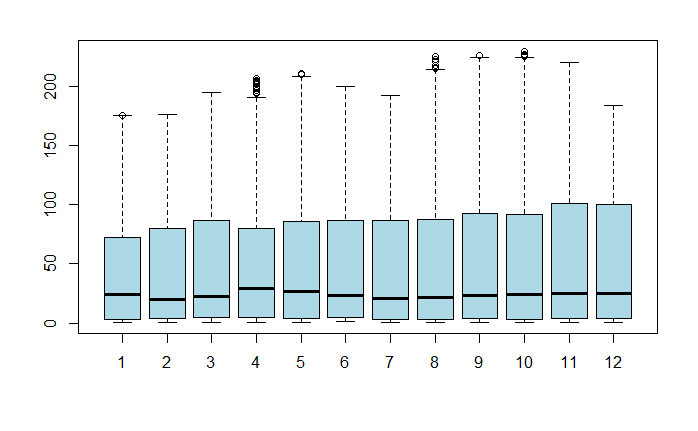
Revisando la tendencia

La prueba de Cox-Stuart muestra que existe tendencia creciente a lo largo de toda la serie.

La prueba de Rho de Spearman también muestra también una correlación positiva alta con el valor de 0.975.

Revisando la estacionalidad

No se aprecia la presencia de estacionalidad desde el punto de vista gráfico de autocorrelaciones, aplicaremos el diagrama de cajas para ampliar la exploración.



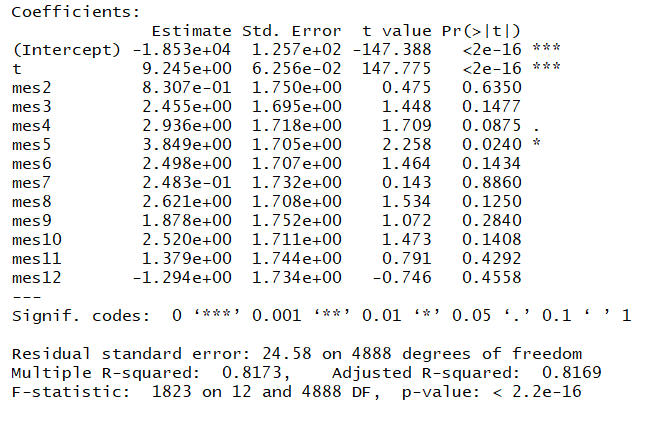
El gráfico de cajas tampoco muestra la presencia de estacionalidad.

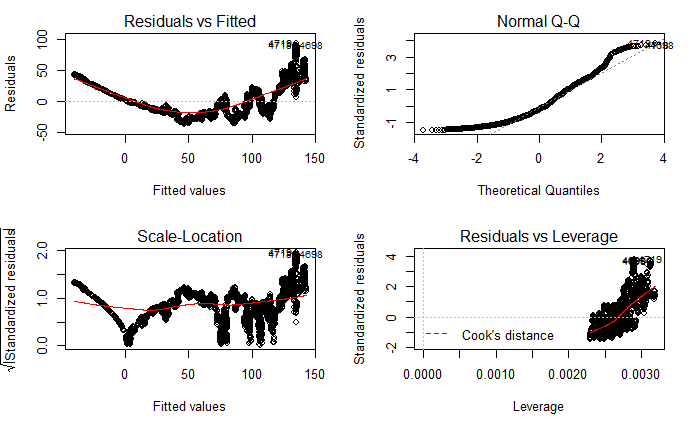
La aplicación de la prueba de Kruskal Wallis corrobora la sospecha de inexistencia de estacionalidad, con p-valor = 1.

Revisión de ajuste de modelo de regresión lineal simple

Aplicaremos el modelo: Y ~ alpha + Beta1 \* t + mes

Los resultados muestran que la variable mes es no significativa y depende básicamente de la variable tiempo, a continuación, se presenta el modelo.





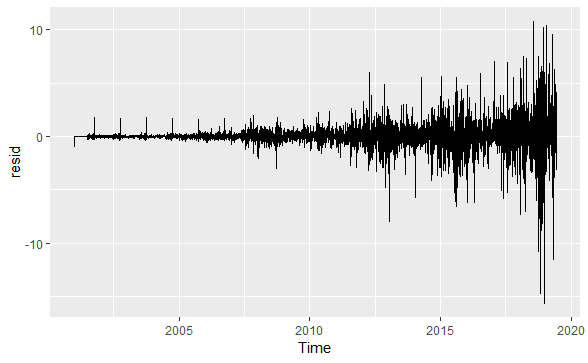
Las principales conclusiones son:

* El modelo de regresión general es muy significativo R2 = 0.817.
* Según el modelo el mes 5 resulta significativo.
* Se sospecha que no se cumple la normalidad en el gráfico de cuantiles, especialmente en los extremos.
* El valor de la predicción puntual no es muy bueno, indicó el valor de 143.55 para el 27 de junio cuando el valor real fue de 199.74.

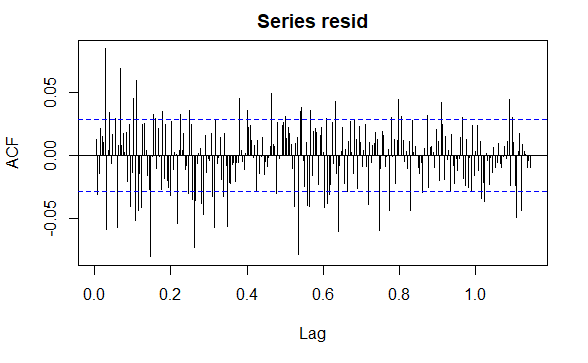
Analizando la serie de tiempo con el modelo Holt Winters.

Se ajustó la serie el modelo Holt Winter con todos los parámetros y estacionalidad aditiva. Los resultados son los siguientes:

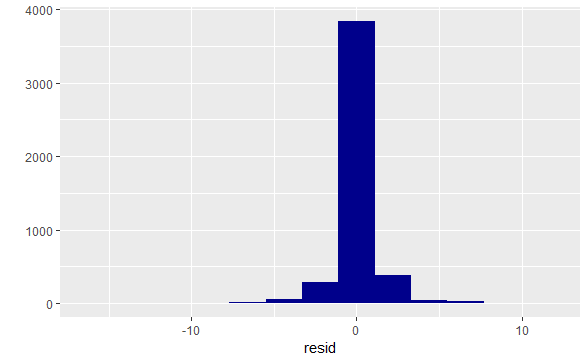
* alpha = 1.000000, beta = 0.000987, gamma = 0.090077
* El nivel estimado del modelo depende del último valor.
* La pendiente del modelo depende mayormente de los valores históricos, se mantiene constante.
* La estacionalidad del modelo depende mayormente de los valores históricos, se mantiene constante.
* Los residu ales no muestran un ruido blanco tal como se aprecia en la figura siguiente.



* La autocorrelación de los residuales muestran varios valores significativos, sin embargo, no hay evidencias de algún patrón.



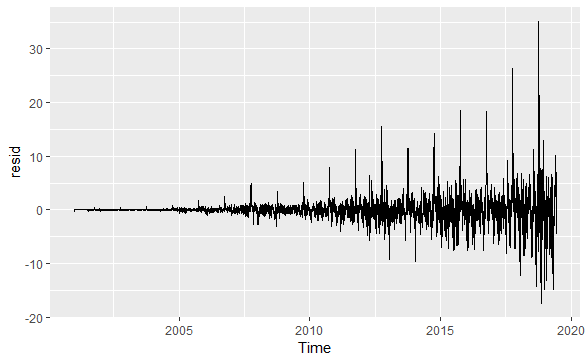
* La prueba t muestra que la media se puede considerar 0.
* El histograma de residuales muestra que existe una distribución simétrica pero concentrada alrededor de 0.



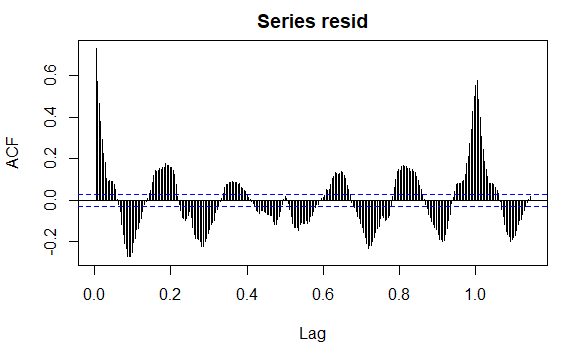
* La prueba de Shapiro indica que no existe normalidad.

Se probó **un modelo multiplicativo de Holt Winter**, las conclusiones fueron las siguientes:

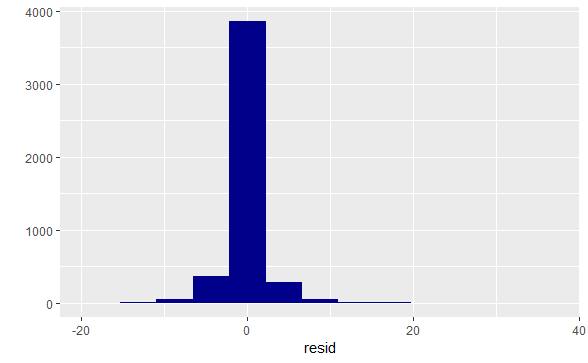
* alpha = 0.782295 beta = 0.000000 gamma = 1.000000
* El nivel estimado del modelo depende mayormente de los últimos valores, es cambiante.
* La pendiente del modelo solo depende de los valores históricos.
* La estacionalidad del modelo depende del último valor.
* Los residuales muestran un patrón que no corresponde a ruido blanco, se aprecia cierta estacionalidad al final de cada año y que los valores aumentan con el tiempo.



* El autocorrelograma confirma estacionalidad.



* La prueba t muestra que los valores se pueden considerar como 0.
* Los residuales presenta un forma simétrica pero concentrada en 0.

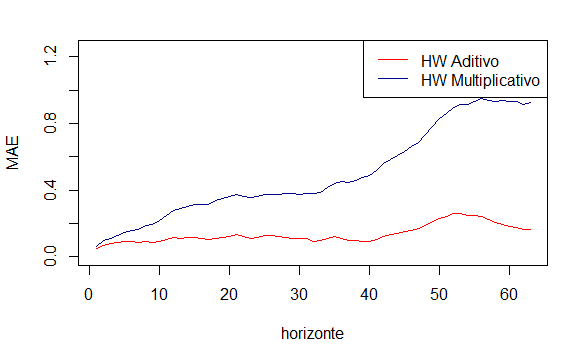


* La prueba de Shapiro muestra que no existe normalidad.

Verificación con cross validation

Se hicieron las pruebas con tamaño de entrenamiento de 36 meses y prueba de 3 meses.Las conclusiones son las siguientes:

* La evaluación del modelo falla en las primeras iteraciones esto es debido a la gran variabilidad que empieza el 2008.
* Utilizando el MAE se concluye que el mejor modelo es el aditivo.



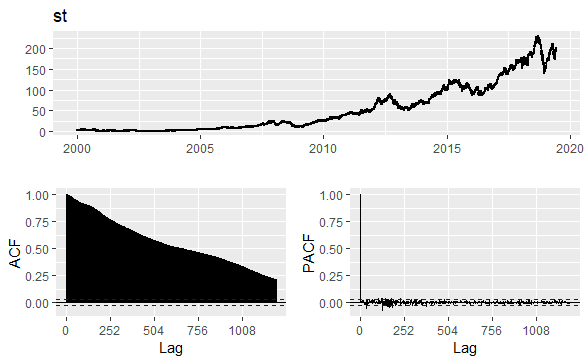
La predicción para el valor del día siguiente resultó muy exacto con 199.84 cuando la cotización real fue de 199.74.

Como conclusión podemos optar por el modelo Holt Winter Aditivo.

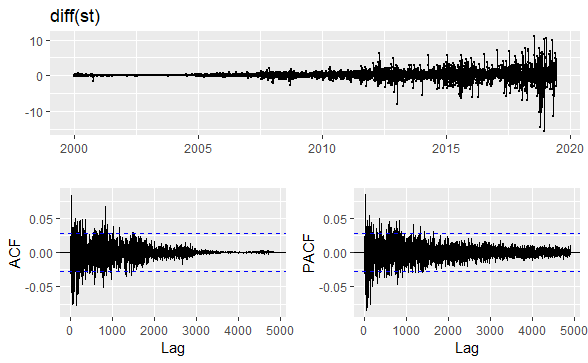
Analizando la serie de tiempo con el modelo ARIMA.

Revisando la estacionariedad

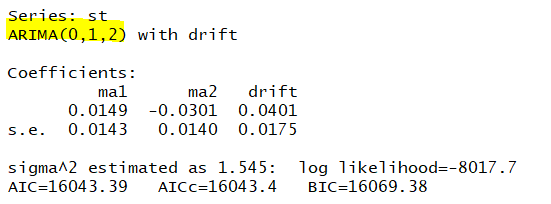
* El gráfico autocorrelación muestra una fuerte correlación, sin embargo, el PACF muestra valores poco significativos.



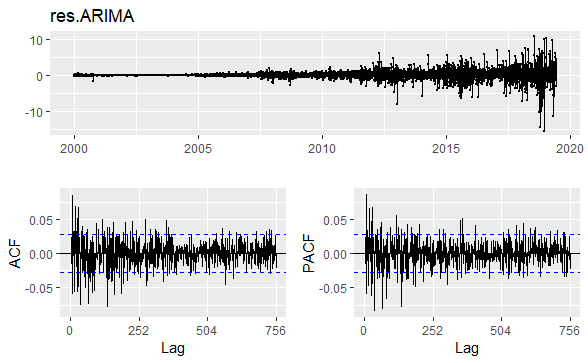
* Las pruebas de estacionariedad muestran que esta no se encuentra presente.
* Diferenciaremos para volver a realizar el análisis y en este caso no se rechaza la estacionariedad.
* Los diagramas de ACF y PACF muestran un patron de extinción.



* El autorima nos muestra que tenemos d=1 y q=2



* El gráfico de residuales no presenta esquema de ruido blanco.



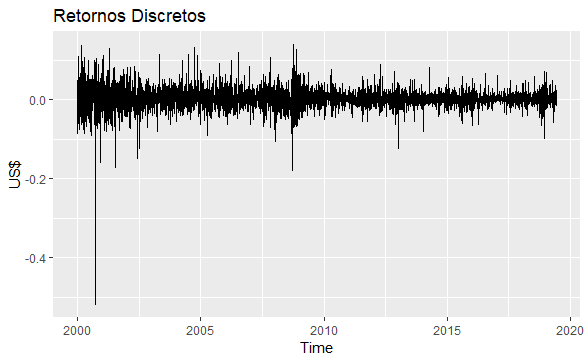
* La prueba de Shapiro indica que no existe normalidad en los residuales.
* La proyección del modelo para el 27 de junio es de 199.96 valor muy cercano al real.

Evaluar las alternativas de software abierto y sus librerías de funciones para determinar la plataforma que mejor se ajuste al procesamiento de datos y

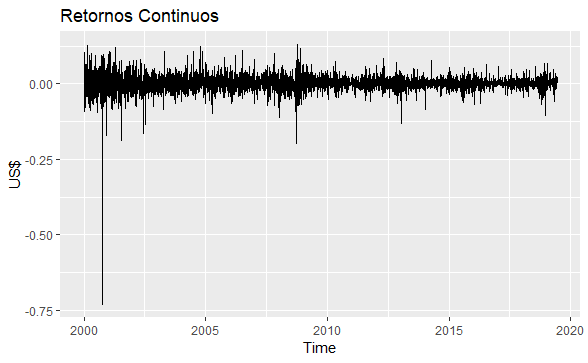
Analizando la serie de tiempo con el modelo GARCH.

Evaluaremos los modelos con dos aproximaciones retornos discretos (R1) y retornos continuos (R2). Las conclusiones son las siguientes:

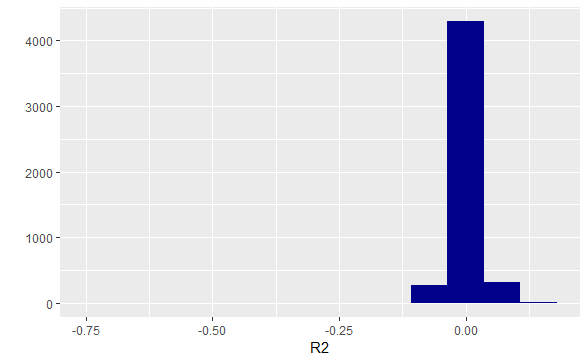
* Los retornos discretos muestran un patrón con mucha volatilidad



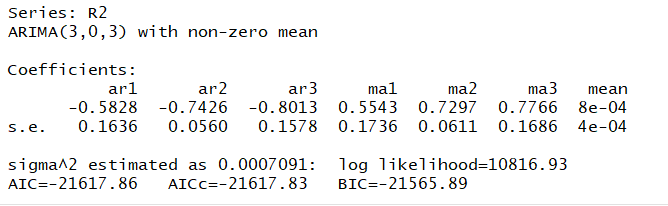
* Los retornos continuos muestran el mismo patrón que los discretos.



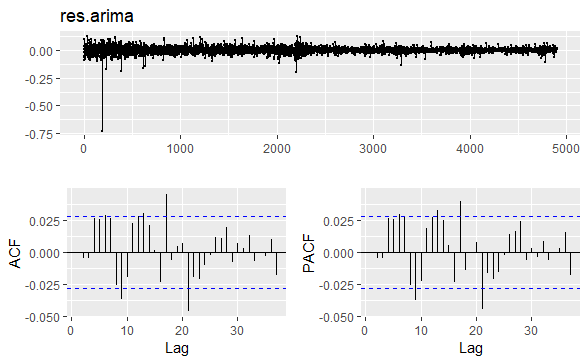
* Los rendimientos son simétricos sin embargo, concentrados alrededor de 0. La prueba de curtosis muestra que la forma de la distribución es leptocúrtica.



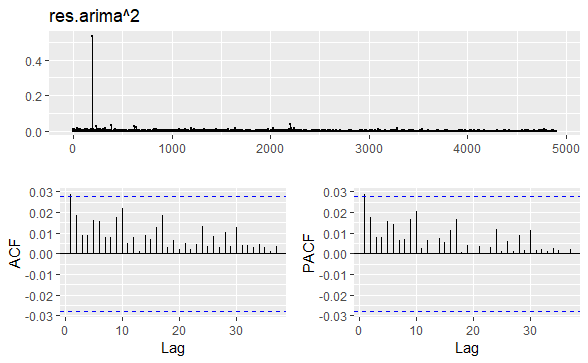
* La prueba de Shapiro indica que no hay evidencias de normalidad.
* La prueba de estacionariedad muestra que existe evidencia de varianza condicional.
* El modelo de autoarima muestra que el mejor modelo para los rendimientos continuos R2 es ARIMA(3, 0, 3) con las siguientes estimaciones.



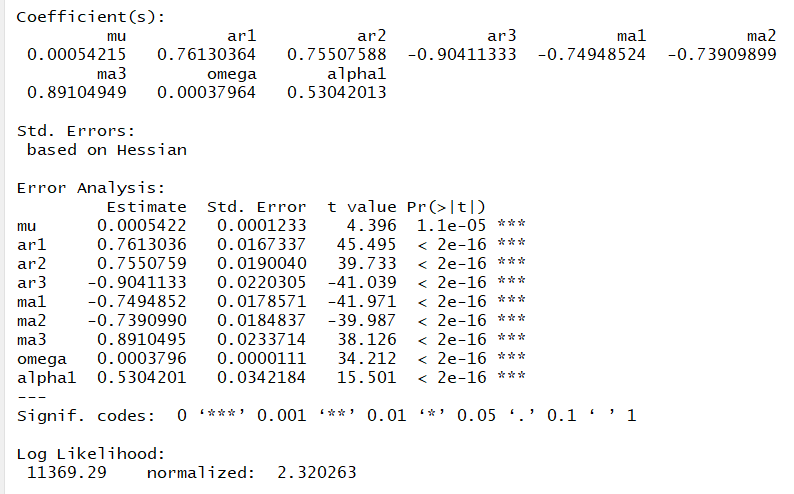
* La prueba t indica que hay evidencias de que la media de los residuales es 0.
* La prueba de Shapiro indica que no existe normalidad en los residuales. Los diagramas ACF y PACF muestran que no existen valores significativos.



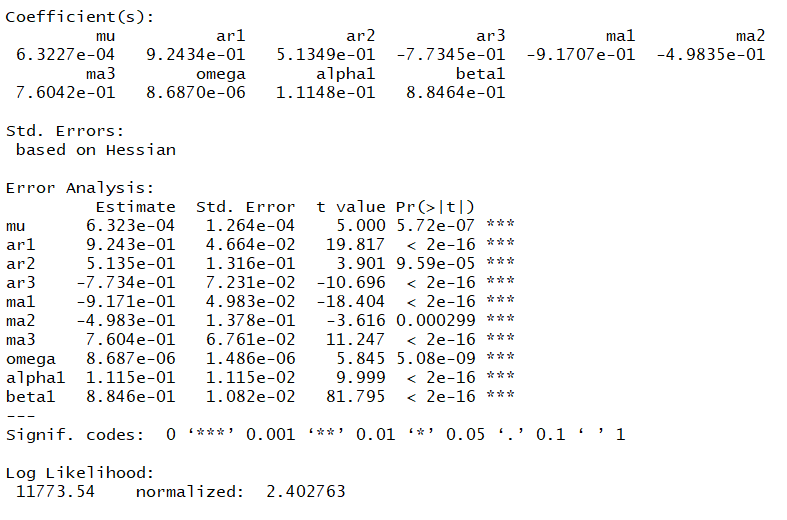
* La conclusión de los residuales es que no presentan el patrón de ruido blanco.
* Los diagramas de los residuales al cuadrado muestran una ligera autocorrelación con un desfase.



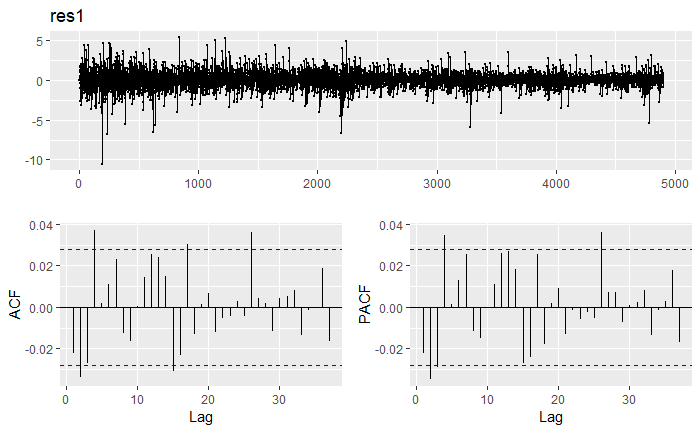
* Probando el modelo GARCH = ARMA(3,3) + GARCH(1,0), todos los coeficientes son altamente significativos.



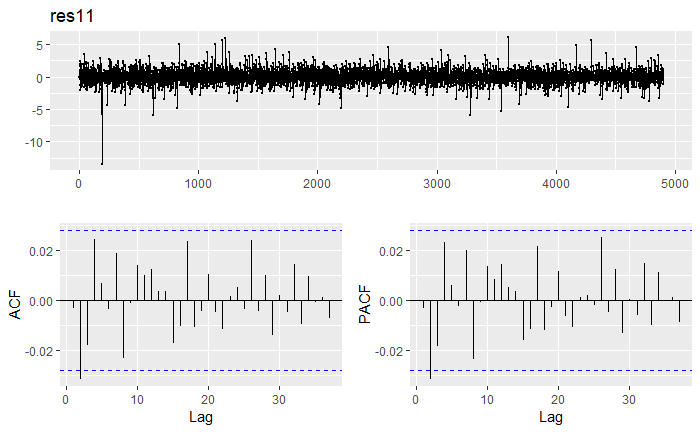
• Probando el modelo GARCH = ARMA(3,3) + GARCH(1,1), todos los coeficientes son altamente significativos



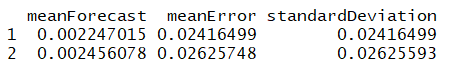
* Respecto a la prueba t de los residuales el primer modelo presenta residuales con media 0 en tanto que el segundo no.
* Respecto a la prueba de Shapiro no hay evidencias de normalidad.
* Graficando los residuales se aprecia pocos valores signficativos.



* El diagrama de los residuales de ACF y PACF para el segundo modelo no presenta valores significativos para las autocorrelaciones.



Como conclusión, de los criterios revisados elegimos el más simple que corresponde al mod.arch1y el que se utilizará para hacer la proyección de los 2 valores siguientes.



Analizando la serie de tiempo con redes neuronales LSTM

Los pasos generales que se utilizan en la configuración de un modelo de Red Neuronal son los siguientes:

1. Especificar la arquitectura
2. Compilar el modelo
3. Ajustar el modelo
4. Predecir

El paso más difícil es el ajuste por la gran cantidad de parámetros que tiene el modelo y muchos elementos tienen tras de si elementos matemáticamente complejos.

Para la implementación de las redes neuronales en R se tienen muchas opciones, se selecciona una de las más estables que corresponde al paquete R. La tabla siguiente presenta las opciones disponibles[[11]](#footnote-11).

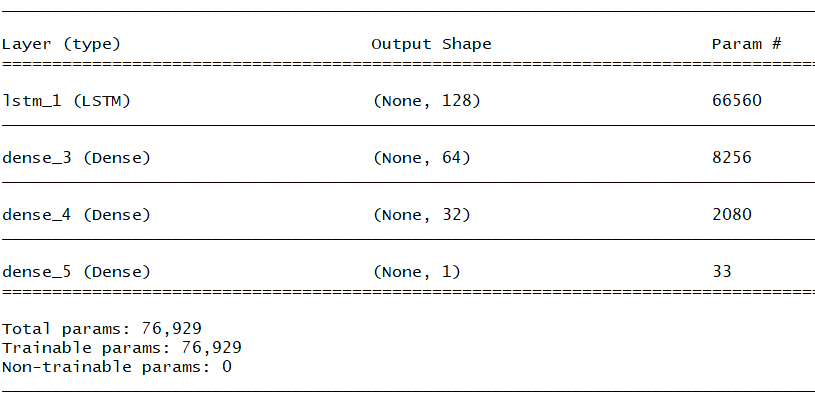
| Package | Per  centile | Description |
| --- | --- | --- |
| nnet | 96th | Software for feed-forward neural networks with a single hidden layer, and for multinomial log-linear models. |
| neuralnet | 96th | Training of neural networks using backpropagation |
| h2o | 95th | R scripting functionality for H2O |
| RSNNS | 88th | Interface to the Stuttgart Neural Network Simulator (SNNS) |
| tensorflow | 88th | Interface to TensorFlow |
| deepnet | 84th | Deep learning toolkit in R |
| darch | 79th | Package for Deep Architectures and Restricted Boltzmann Machines |
| rnn | 73rd | Package to implement Recurrent Neural Networks (RRNs) |
| FCNN4R | 52nd | Interface to the FCNN library that allows user-extensible ANNs |
| rcppDL | 7th | Implementation of basic machine learning methods with many layers (deep learning), including dA (Denoising Autoencoder), SdA (Stacked Denoising Autoencoder), RBM (Restricted Boltzmann machine) and DBN (Deep Belief Nets) |
| deepr | ??\* | Package to streamline the training, fine-tuning and predicting processes for deep learning based on darchand deepnet |
| MXNetR | ??\* | Package that brings flexible and efficient GPU computing and state-of-art deep learning to R |

Para el análisis se ejcutaron varias corridas para obtener los mejores indicadores para el criterio de MAE, la tabla que se presenta a continuación muestra los resultados de las corridas. El de mejor resultados fue el que se presenta resaltado en amarillo que corresponde a un lag=1 y las funciones de activación RELU y linear para la capa oculta.

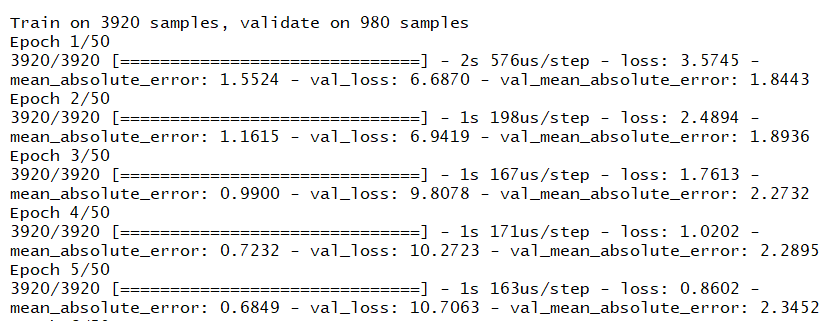


Durante la corrida del ajuste del modelo se obtienen las siguientes salidas:

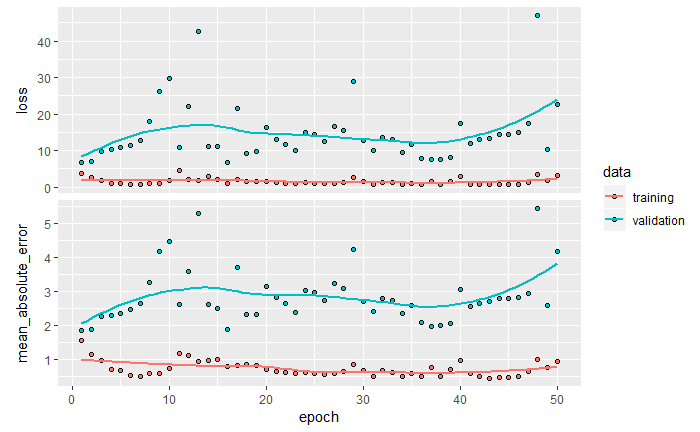
* Configuración de la red



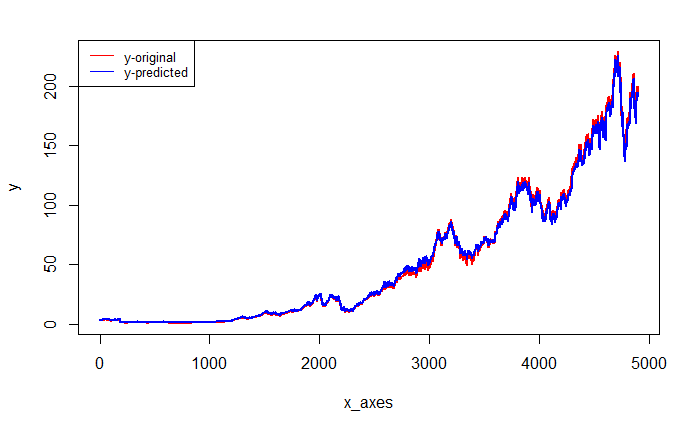
* La salida de las primeras iteraciones u épocas donde se ve como va decreciendo la función objetivo MSE.



* Revisión de los resultados de train y test.



* Revisión de varloes reales y predichos.



* Pronóstico para 5 días, se aprecia que pronostica muy preciso a corto plazo.



# CONCLUSIONES

* Las redes neuronales presentan una gran capacidad predictiva, sin embargo, el número de parámetros necesarios para la configuración, y la complejidad matemática de los optimizadores que se encuentran por detrás requieren de arduo trabajo de ajuste para que obtener buenos modelos.
* Seleccionar funciones de activación para las capas resultó tener un gran impacto en la medida de los indicadores, así por ejemplo, utilizar la función sigmoide empeoró tremendamente el indicador MAE.
* El modelo LSTM maneja por si solo los lags, cuando se amplió al modelo la ventana para regresionar con lags de 2 o 3 días los resultados fueron inferiores al lag de 1 día.
* Se requiere capacidad de cómputo para poder realizar las corridas, en algunas laptops el modelo no corrió, la configuración de la Laptop donde se ejecutó es una gamer con procesador Intel core i7 con 12 MB de RAM y con tarjeta de video Geforce de 2 MB.
* Una buena alternativa a las redes neuronales son los modelos GARCH, sin embargo, la red neuronal tiene la ventaja de ser más robusto al requerir menos supuestos para validación.

# REFERENCIAS BIBLIOGRAFICAS

Chollet F, Allaire J. (2018), Deep Learning with R, Manning Publications Co., NY

Colah, Understanding LSTM Networks, tomado de colah.github.io/posts/2015-08-Understanding-LSTMs/

Graves, Alex, Supervised Sequence Labelling with Recurrent Neural Networks

Ejemplo tomado de Datatechnote, https://www.datatechnotes.com/2019/01/regression-example-with-lstm-networks.html

Curso Deep Learning in Python, en Datacamp, dictado por Becker D, https://www.datatechnotes.com/2019/01/regression-example-with-lstm-networks.html

# ANEXOS

Código para las corridas de la red LSTM.

```{r}

library(keras)

library(astsa)

head(st)

# revisando retornos

st\_return <- diff(log(st))

plot(st\_return)

acf2(st\_return)

auto.arima(st\_return)

```

##

```{r}

# preparing data

# st\_window: number of lags to construct matrix x to

st\_window <- 1

# nrow <- length(st\_return) - st\_window

nrow <- length(st) - st\_window

data\_model <- matrix( NA , nrow = nrow, ncol = st\_window +1 )

for (i in 1:(st\_window + 1)) {

#data\_model[1:nrow, i] = st\_return[(st\_window + 2 - i ):(nrow + st\_window + 1 - i)]

data\_model[1:nrow, i] = st[(st\_window + 2 - i ):(nrow + st\_window + 1 - i)]

}

# Prepare variables: features and dependents

X <- array(data\_model[ , 2:(st\_window + 1)], dim=c(nrow, st\_window , 1) )

y <- data\_model[ , 1]

# Model

model = keras\_model\_sequential() %>%

layer\_lstm(units=128, input\_shape=c(st\_window, 1), activation="relu") %>%

layer\_dense(units=64, activation = "relu") %>%

layer\_dense(units=32) %>%

layer\_dense(units=1, activation = "linear")

model %>% compile(loss = 'mse',

optimizer = 'adam',

metrics = list("mean\_absolute\_error")

)

model %>% summary()

history <- model %>% fit(X, y, epochs=50,

batch\_size=32,

shuffle = FALSE,

verbose=T,

validation\_split = 0.20)

plot(history)

y\_pred <- model %>% predict(X)

scores <- model %>% evaluate(X, y, verbose = T)

print(scores)

x\_axes = seq(1:length(y\_pred))

plot(x\_axes, y, type="l", col="red", lwd=2)

lines(x\_axes, y\_pred, col="blue",lwd=2)

legend("topleft", legend=c("y-original", "y-predicted"),

col=c("red", "blue"), lty=1,cex=0.8)

```

### Predicting

Pronosticamos valores para 5 observaciones (1 semana)

```{r}

# h\_window: horizont to predict

h\_window <- 5

data\_predict <- numeric(h\_window + 1)

data\_predict[1] <- tail(y, 1)

for (i in 2:(h\_window+1)) {

X\_predict <- array(data\_predict[i-1], dim=c(1, st\_window , 1) )

X\_value <- model %>% predict(X\_predict)

data\_predict[i] <- X\_value[1,1]

}

print(data\_predict)

```

1. Chollet es el coautor de Keras uno de los paquetes más utilizados en el modelamiento de redes neuronales su implementación fue hecha para Python y existen paquetes en R que le hacen warpping a esta implementación para exponer las funcionalidades en R. [↑](#footnote-ref-1)
2. Tomado del libro Chollet “Deep Learning with R” [↑](#footnote-ref-2)
3. Tomado del libro de Chollet “Deep Learning with R”. [↑](#footnote-ref-3)
4. Traducción de la web Fundamental of Statistics <http://www.statistics4u.com/fundstat_eng/cc_ann_intro1.html> [↑](#footnote-ref-4)
5. Imagen tomada de <http://www.statistics4u.com/fundstat_eng/cc_ann_intro1.html> [↑](#footnote-ref-5)
6. Tomada de la web <https://sefiks.com/2017/10/14/evolution-of-neural-networks/> [↑](#footnote-ref-6)
7. Tomado del libro de Chollet “Deep Learning with R” [↑](#footnote-ref-7)
8. Las figuras han sido tomadas del libro de Chollet “Deep Learning with R” [↑](#footnote-ref-8)
9. Tomado de <https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464> [↑](#footnote-ref-9)
10. Tomado de “SupervisedSequenceLabellingwithRecurrentNeuralNetworks” de Graves. [↑](#footnote-ref-10)
11. Tutorial de Datacamp elaborado por K. Willems en la siguiente ruta <https://www.datacamp.com/community/tutorials/keras-r-deep-learning> [↑](#footnote-ref-11)