Universidad Nacional de Colombia

Predicción del comportamiento del par BTC/USDT usando modelos de aprendizaje de máquina

Autores:
Leonel Fernando Ardila Peña, email:
lfardilap@unal.edu.co
Juan Sebastián Flórez Jiménez, email:
jflorezj@unal.edu.co

June 6, 2018

Abstract

Se implementan dos modelos de aprendizaje máquina para predecir el cambio en el valor del Bitcoin y uno se estos es emparejado con una estrategia básica de trading para demostrar la capacidad predictiva y su gran potencial. Se predice el cambio en el valor del par Bitcoin vs Theter (BTC/USDT) utilizando una Red Neuronal Recurrente tipo LSTM (Long-Short Term Memory). Se predice la tendencia del mercado (alsista/bajista) del precio del Bitcoin usando tres algoritmos de clasificación: Random Forest, Logistic Regression, Neural Network (Multi Layer Perceptron).

1 Introducción

Las series de tiempo aparecen naturalmente en el estudio de sistemas dinámicos, como lo son el movimiento de grandes conglomerados de partículas en medios acuosos (movimiento de proteinas) y la variación del precio de una acción en el mercado de valores. Para estudiarlas se han desarrollado varias herramientas estádisticas, e.g. los modelos ARIMA (Auto Regresive Integrated Moving Average) y modelos de redes neuronales como las LSTM (Long Short Term memory) han demostrado tener una habilidad predictiva notable.

El abordaje tradicional para obtener información de un sistema que esté representado por una serie de tiempo es ajustar un modelo que permita predecir el comportamiento de la serie, en esta dirección se han hecho importantes desarrollos siendo uno de ellos la red neuronal con memoria LSTM, la cual surgió como una alternativa a los modelos clásicos de aprendizaje de máquina que no podían captar el comportamiento de las series de tiempo. La red neuronal LSTM tiene memoria, lo cual implica que se plantea como una alternativa atractiva para modelar sistemas que dependen de sus estados anteriores, una característica difícil de reproducir con redes neuronales unidireccionales.

Una red neuronal LSTM es una red neuronal recurrente (RNN) cuyos bloques son del tipo LSTM, i.e. cada bloque tiene una unidad de memoria, una compuerta de entrada, una compuerta de olvido y una compuerta de salida, donde cada compuerta es una neurona. Este modelo suele ser entrenado usando backpropagation, ya que el error se guarda en las unidades de memoria y no es disipado por el tamaño de la red neuronal recurrrente, sin

embargo, las aplicaciones comerciales de las redes neuronales LSTM utilizan CTC (connectionist temporal classification) como método de entrenamiento.

Dentro de los modelos clásicos para predecir series de tiempo se encuentra el modelo modelo ARIMA(p,d,q) el cual agrupa tres atributos de las series de tiempo,

- **A**uto **R**egressive: Se realiza una regresión de la variables con sus anteriores valores
- **I**ntegrated: Los valores a ajustar son las diferencias de la serie; primeras diferencias, segundas diferencias, etc.
- **M**oving **A**verage: el error de regresión es una combinación lineal de errores de valores pasados y el valor actual.

Se debe encontrar un valor adecuado para los parámetros (p,d,q), que están asociados con,

- **p**: número de lags del modelo autoregresivo
- **d**: órden de diferenciación de la serie de tiempo
- **q**: órden del modelo de promedio móvil (moving average)

Para ello se analizaron la autocorrelación y autocorrelación parcial de las primeras y segundas diferencias, con esta información se determinó una cota para los valores p,q y luego se encontró el valor de p,q que permiten encontrar un mejor ajuste a la serie de tiempo al comparar el Criterio de información de Aike (AIC)para cada par de valores de p,q. Este modelo se utilizó para predecir el comportamiento del valor del Bitcoin frente al Theter (BTC/USDT).

Se implementó un modelo poco tradicional para predecir el comportamiento de series de tiempo el cual consiste en utilizar clasificadores para predecir una característica de la serie en el siguiente paso, en este caso se desea predecir si el valor del Bitcoin frente al Theter va a aumentar, disminuir o se va a mantener estable (su variación es menor a una cota) durante el siguiente paso. Se utilizaron tres modelos de clasificación, Random Forest, Regresión logística y Redes Neuronales Unidireccionales y se utilizaron para predecir el comportamiento del valor del BTC frente al USDT en marcos de tiempo

de 1 hora y 3 minutos. El uso de clasificadores implica establecer atributos para que el clasificador prediga la característica deseada, la primera opción que surge es el valor de la serie de tiempo en los anteriores pasos de tiempo, sin embargo es deseable realizar un preprocesamiento de esta serie de tiempo para extraer los atributos que se consideren importantes y usarlos como variables para clasificador, esto mejora en gran medida la calidad del modelo implementado especialmente si este modelo es un clasificador lineal. Este modelo fue propuesto en el artítculo **Trading Bitcoin and Online Time Series Prediction** escrito por Muhammad J. Amjad y Devavrat Shah, ambos asociados con el MIT.

2 Datos

Los mercados de criptodivisas mas grandes del mundo (Bitfines, Bitrex, Binance) cuentan con API's que permiten obtener información histórica del mercado en cada uno de los marcos temporales ya mencionados, esta información contiene la variación del valor de pares de criptodivisas, el volumen de las transacciones realizadas en cada marco de tiempo, La cantidad de transacciones realizadas en un marco de tiempo, y el volumen de las ordenes puestas en un marco de tiempo.

Para entrenar los modelos se emplea la información historica por horas del precio del par BTC/USDT y la información histórica del volumen de transacciones suministrada por la API de Binance. Durante cada hora (3 minutos) se sunimistra el precio de apertura (Precio al que se comerciaba cuando empieza la hora), el precio de cierre (Precio al que se comerciaba cuando finaliza la hora), precio mas bajo (menor precio alcanzado durante la hora), el precio mas alto (mayor precio alcanzado durante la hora) y el volumen de las transacciones realizadas en cada marco temporal, estos datos son etiquetandose respectivamente como **Open**, **Close**, **High**, **Low**, **Volume**.

Los datos del precio empleados van desde el 2017-08-13 hasta el 2018-06-02, siendo en total 34930 datos (6986 datos por serie de tiempo) para el marco de 1 hora, y 698600 datos (139720 datos por serie de tiempo) para el marco de 3 minutos. Para la red neuronal LSTM se emplea el marco de tiempo de 1 hora y se utilizan 4 series de tiempo que corresponden al precio de cierre, el precio de apretura, el máximo precio y el mínimo precio, los datos se dividen

Figure 1: Precio histórico del Bitcoin (BTC) con respecto al theter (USDT).

en un 80% empleados para entrenar los sistemas y el 20% restante para validarlo. El clasificador utiliza dos series de tiempo que corresponden al precio de cierre y el volumen de las transacciones, emplea dos marcos temporales el marco de 1 hora y el marco de 3 minutos, y se dividen los datos en tres conjuntos 80% para entrenar el clasificador, 10% para ajustar parámetros el modelo y obtener la mayor ganancia posible al realizar transacciones

3 Set-Up experimental

El lenguaje de programación empleado para construir los sistemas predictivos fue python, los paquetes empleados fueron: keras, pandas, numpy, matplotlib, sklearn, statsmodels. A partir de la librería keras se contruye cuatro redes neuronales que como entrada recibe una ventana de valores históricos de las cuatro variables, y como resultado, cada una de ellas pronosticará el valor que puede llegar a adquirir cada una (Open, Close, High, Low). Cabe resaltar que las redes neuronales tienen cada una dos capas ocultas con 168

neuronas LSTM.

La librería statsmodels de python permite ajustar un modelo ARIMA(p,d,q) a una serie de tiempo univariada, realizar predicciones con este modelo, calcular la eficiencia del modelo y la presición del ajuste realizado a la serie de tiempo empleada para entrenarlo. Se tomó la serie de tiempo de con diferencias entre los datos de 1 hora y se ajustó la serie para el precio de cierre del par BTCUSDT, luego se analizó la capacidad de éste modelo para predecir la serie de tiempo; el conjunto de datos de dividieron en 80% entrenamiento y 20% testeo.

4 Métodología

Inicialmente Se organizan los datos proveidos por la API en un data-Frame de pandas despreciando información irrelevante y sólamente condensando la información de los precios de interés (open, close, High, Low, Volume). Debido a que los valores de los precios son demasiado altos (algunos alcanzan los 19800 USDT), se deben normalizar los datos a partir de MinMaxScaler() del paquete preprossesing de sklearn para evitar problemas al entrenar la red neuronal.

Una Vez normalizados los datos estos se agruparan numpys arrays y luego se ordenan en subconjuntos mas pequeños cuyo número de items dependerá del número de valores historicos se observe atrás en el tiempo para predecir el siguiente. Finalmente los datos son divididos en 80% de ellos para entrenamiento y el 20% restante para validar el modelo.

Para predecir los valores que adoptará se diseña un sistema que toma la informaciòn unos pasos atras en el tiempo, es decir una hora antes, dos horas antes,..., n horas antes y a partir de esta información arrojara un conjunto de precios de apertura, cierre, máximo y mínimo.

4.1 Algoritmos de clasificación

Inicialmente Se organizan los datos proveidos por la API en un DataFrame de pandas del cuál se extraerá información a usarse en el entrenamiento de

los clasificadores. Se utilizan dos conjuntos de características para predecir el comportamiento de la serie de tiempo y se compará el desempeño de los modelos entrenados con cada conjunto de características.

Reproduciendo lo propuesto por Muhammad J Amjad y Devavrat Shah en el artículo **Trading Bitcoin and Online Time Series Prediction** se debe realizar un preprocesamiento de las series de tiempo para extraer los atributos que se utilizarán para clasifica la tendencia del valor del precio del BTC.

Sea p[t] el precio de cierre en el tiempo t, se define la serie de las primeras diferencias y[t]=p[t]-p[t-1] y se define x[t] como el indicador del cambio en el precio de cierre en el paso t, es decir, sea $\theta \in \mathbb{R}$, defina

$$x[t] = \begin{cases} -1, y[t] < -\theta \\ 1, y[t] > \theta \\ 0, \text{en otro caso} \end{cases} \tag{1}$$

Sea v[t] el volumen de las transacciones en el tiempo t, se define la serie de primeras diferencias z[t]=v[t]-v[t-1] y se define w[t] como el indicador del cambio en el volumen de transacciones en el paso t, es decir, sea $\phi \in \mathbb{R}$, defina

$$w[t] = \begin{cases} -1, z[t] < -\phi \\ 1, z[t] > \phi \\ 0, \text{en otro caso} \end{cases} \tag{2}$$

Usando esta información se procede a extraer los atributos que se usarán para realizar la clasificación.

- **Propuesta de atributos en el artículo reproducido**

  1. p[t-1]/Max_Price, precio normalizado del BTC en el paso t-1
  2. y[t-d]/p[t-(d+1)], cambio porcentual del precio del BTC para los anteriores d pasos
  3. $T_{\sigma,d_1}[t-1] = \sum_{s=t-d_1}^{t-1} \mathbb{1}\{x[s] = \sigma\}$: la cuenta de cuantos cambios tipo σ se presentarón en $x[t-d_1 : t-1]$ para cada d_1 cada $\sigma \in \{-1, 0-1\}$

4. $C_\sigma[t-1] = max_k(k : \sigma = x[t-1] = x[t-1] = ... = x[t-k])$ para cada $\sigma \in \{-1, 0, 1\}$

- **Propuesta de atributos nuestra**

1. p[t-1]/Max_Price, precio normalizado del BTC en el paso t-1
2. y[t-d]/p[t-(d+1)], cambio porcentual del precio del BTC para los anteriores d pasos
3. $T_{\sigma,d_1}[t-1] = \sum_{s=t-d_1}^{t-1} \mathbb{1}\{x[s] = \sigma\}$: la cuenta de cuantos cambios tipo σ se presentarón en $x[t-d_1 : t-1]$ para cada d_1 cada $\sigma \in \{-1, 0-1\}$
4. $C_\sigma[t-1] = max_k(k : \sigma = x[t-1] = x[t-1] = ... = x[t-k])$ para cada $\sigma \in \{-1, 0, 1\}$
5. v[t-1]/Max_Volume, volumen de transacciones normalizado en el paso t-1
6. z[t-d]/v[t-(d+1)], cambio porcentual del precio del BTC para los anteriores d pasos
7. $Tvol_{\sigma,d_1}[t-1] = \sum_{s=t-d_1}^{t-1} \mathbb{1}\{x[s] = \sigma\}$: la cuenta de cuantos cambios tipo σ se presentarón en $x[t-d_1 : t-1]$ para cada d_1 cada $\sigma \in \{-1, 0-1\}$
8. $C_\sigma[t-1] = max_k(k : \sigma = x[t-1] = x[t-1] = ... = x[t-k])$ para cada $\sigma \in \{-1, 0, 1\}$

Al momento de poner la información sobre la cantidad de cantidad de cambios en $x[t-d_1 : t-1]$ se obvia $d_1 = 1$ y se define un intervalo de recolección de información INT_INF, tal que se registrarán datos de la cantidad de cambios en $x[t-d_1 : t-1]$ si y solo si $d_1 \% INT\_INF = 0$, esto permite eliminar información redudante de los atributos, adicionalmente se define una cota superior para d_1 llamada $VENTANA$ de observación. No es necesario que $d = VENTANA$ o $d = d_1$, se escoge el valor de d como uno la distancia temporal donde la autocorrelación de la serie de las primeras diferencias deja de ser predominantemente negativa y empieza a oscilar al rededor del valor nulo.

5 Evaluación

5.1 ARIMA(p,d,q)

Se observa el comportamiento de la serie predicha para el conjunto de testeo y se compará con la serie real, tras realizar cada predicción se vuelve a entrenar el modelo ARIMA(p,d,q) incluyendo en los datos de entrenamiento el valor real de la serie en el tiempo que se acabó de predecir.

5.2 LSTM

Para evaluar la fiabilidad de la red neuronal del modelo se comparan los resultados predichos con los datos reales, al considerar el histograma de las diferencia entre los datos real y predicho se determina el sesgo de la predicción del precio. Además para evitar inconvenientes de sobreajuste y bajo ajuste se calcula el error sobre los datos de entrenamiento y testeo como función del numero de epoch. Finalmente se grafica el precio predicho y real por cada una de las variables.

5.3 Clasificador

El clasificador se junta con una estratégia básica de trading y la eficiencia del modelo se determina en función del porcentaje de ganancia del agente que ejecuta el trading sobre el conjunto de testo que corresponde al último 10% del conjunto de datos, cabe resltar que el conjunto de testeo corresponde a una tendencia bajista, lo cual hace mucho más díficil para el agente obtener ganacias, es decir, no perder dinero en ese conjunto de datos es considerado un buen resultado.
El clasificador realiza predicciones con una determinada probabilidad, lo ideal es que las desiciones se tomen si se tiene una buena certeza sobre el comportamiento del mercado por lo tanto, se utiliza el conjunto de validación para encontrar el valor de certeza ideal para obtener el mayor retorno. El valor para la certeza γ se escoge entre 0.5 y 1.0.
Se define,

$$\hat{x}[t] = \begin{cases} \sigma, P(x[t] = \sigma)\gamma \\ 0, \text{en otro caso} \end{cases} \tag{3}$$

El agente inicia con 1BTC y puede hacer dos tipos de movimientos,

1. Vender la cantidad de BTC al precio de cierre
2. Comprar todo lo que se tiene en BTC al precio de cierre

Se supone que la orden se cierra en el intervalo de tiempo que puede ser de 1 hora o 3 minutos, es decir, no se modela la posible aversión del mercado a comprar o vender; suponer que la posición se cierra dentro del marco de tiempo es una buena aproximación en primera instancia ya que se está vendiendo o comprando al precio del mercado.

6 Resultados

Se grafica la función de autocorrelación y autocorrelación parcial de la serie original del valor del BTC en USDT, esto permite ver que esta serie no sigue un comportamiento normal y los datos se encuentran fuertemente relacionados.

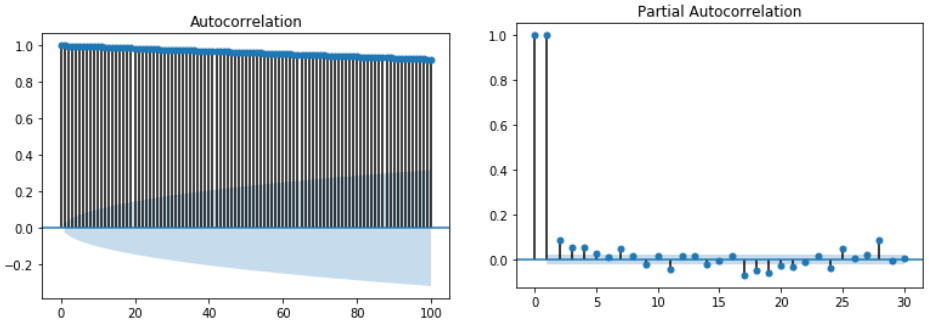

Figure 2: Función de autocorrelación y autocorrelación parcial para la serie de tiempo del valor del BTC en USDT

6.1 ARIMA(p,d,q)

Se genera el histograma de las primeras y segundas diferencias de la serie de tiempo y se realiza la gráfica QQ de cada uno de estos histogramas revelando que el comportamiento de ambas diferencias es aproximadamente normal con solas pesadas.

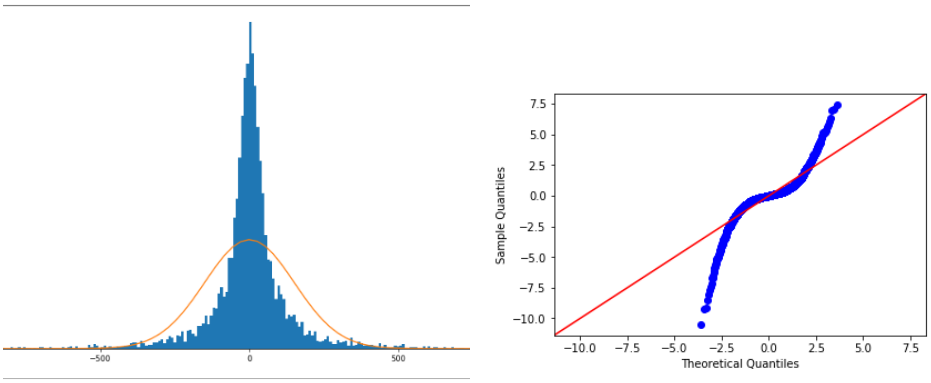

Figure 3: Histograma para la serie de las primeras diferencias y la gráfica QQ de éste histograma

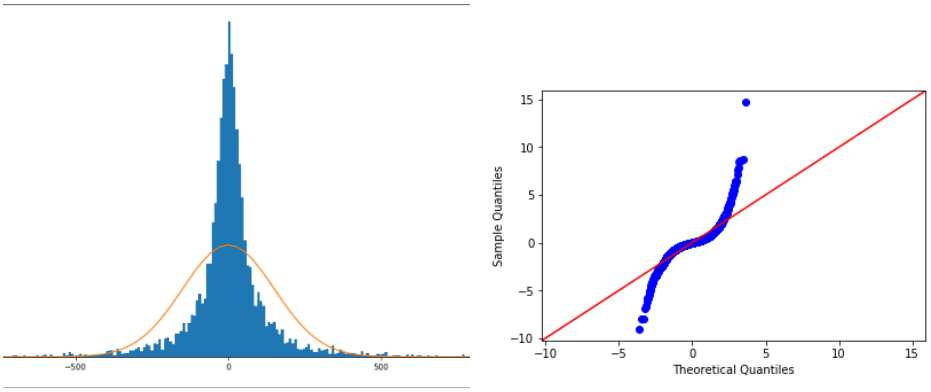

Figure 4: Histograma para la serie de las segundas diferencias y la gráfica QQ de éste histograma

Se grafíca la función de correlación de las primeras y segundas diferencias la cual revela que el orden de diferenciación ideal para ajustar un modelo ARIMA(p,d,q) es d=1, y se debe tener que $p, q \leq 8$.

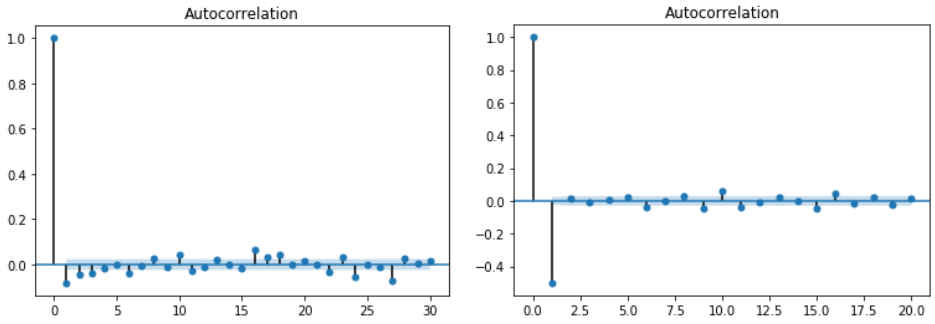

Figure 5: Izquierda: función de autocorrelación para la serie de las primeras diferencias. Derecha: función de autocorrelación para la serie de las segundas diferencias.

Se calcula el índice AIC para cada modelo ARIMA(p,1,q) con $p, q \in \{0, ..., 8\}$ encontando que la mejor combinación de parámetros para ajustar la serie de tiempo del valor del BTC en USDT es p=5,d=1,q=4. Se procede a ajustar el conjunto de entrenamiento con un modelo ARIMA(5,1,4) produciendo la siguiente gráfica,

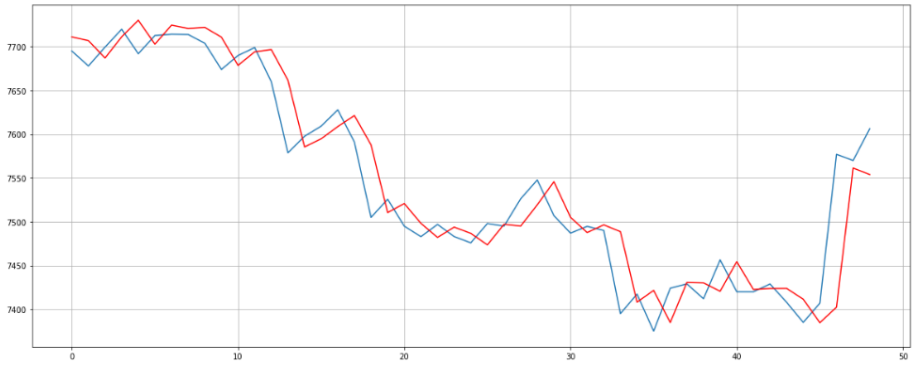

Figure 6: Predicción realizada por el modelo ARIMA(5,1,4) para los primeros 50 pasos. Azul:Serie del precio de cierre, Rojo: Predicción realizada por el modelo ARIMA(5,1,4)

En este caso el modelo ARIMA(5,1,4) está dando información mínima sobre el comportamiento de la serie de tiempo, ya que está copiando en su mayoría el estado de la serie en el paso anterior lo cual es inutil al momento de predecir

una serie de tiempo, se concluye que no es posible realizar predicción de la serie del valor del BTC utilizando un modelo ARIMA.

6.2 LSTM

6.2.1 Precio de apertura:

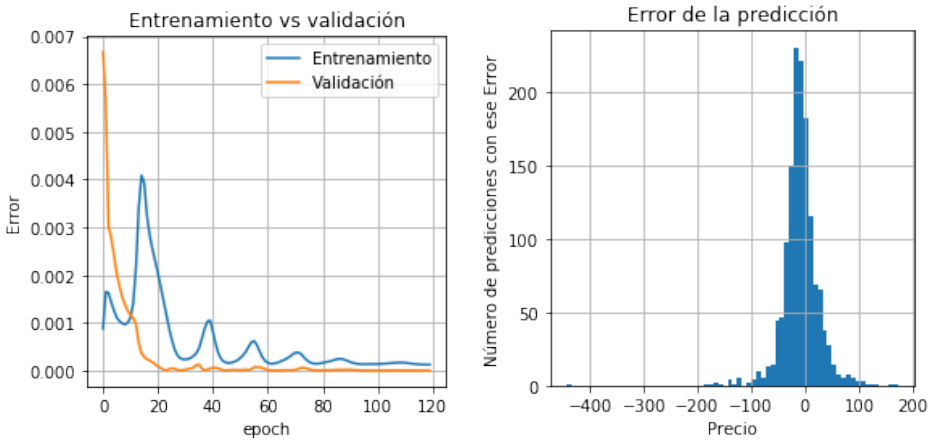

Figure 7: Error durante el entrenamiento; histograma del error de predicción para el precio de apertura por hora.

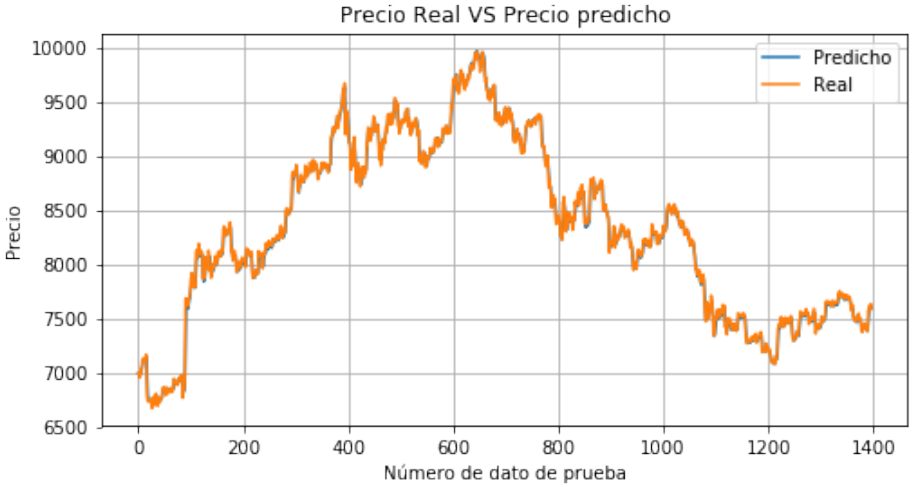

Figure 8: Precio pronosticado por la red neuronal lSTM y precio real para precio de apertura por hora.

6.2.2 Precio de cierre:

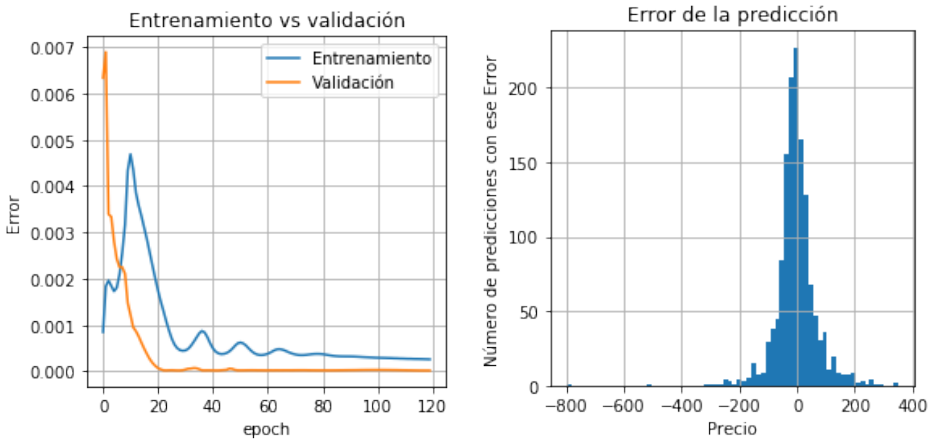

Figure 9: Error durante el entrenamiento; histograma del error de predicción para el precio de cierre por hora.

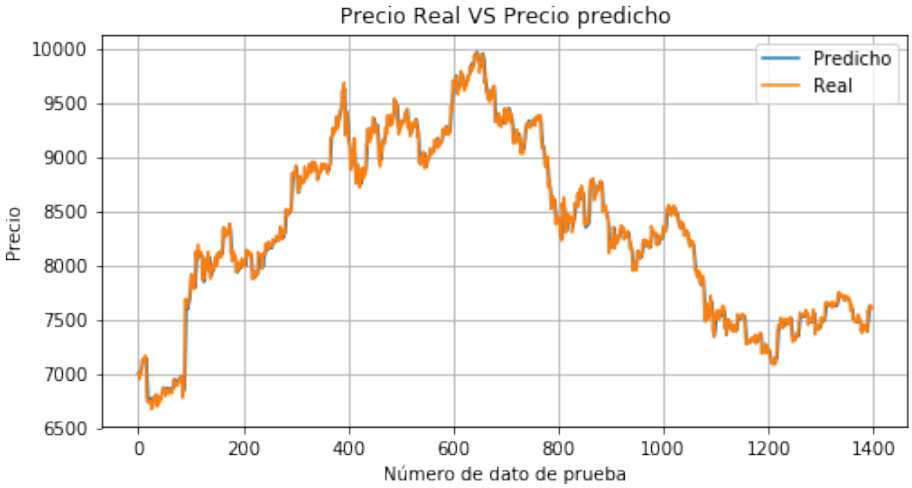

Figure 10: Precio pronosticado por la red neuronal lSTM y precio real para el máximo precio de cierre por hora.

6.2.3 Precio máximo

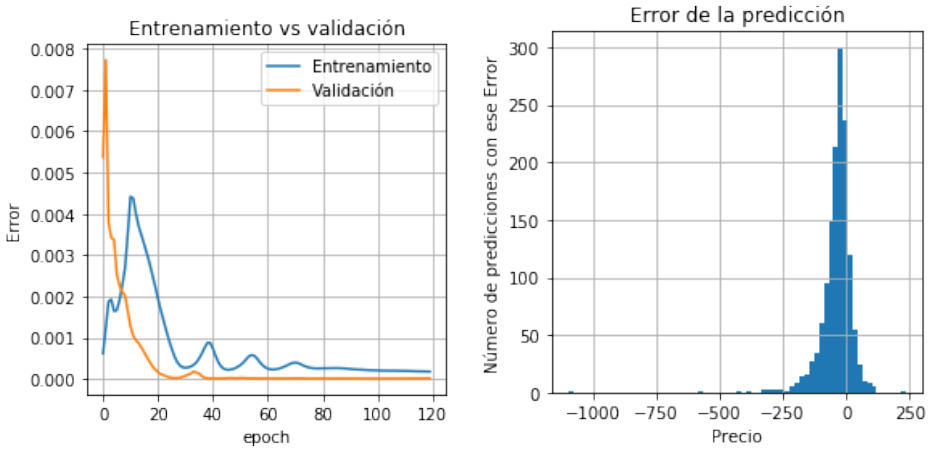

Figure 11: Error durante el entrenamiento; histograma del error de predicción para el máximo precio por hora.

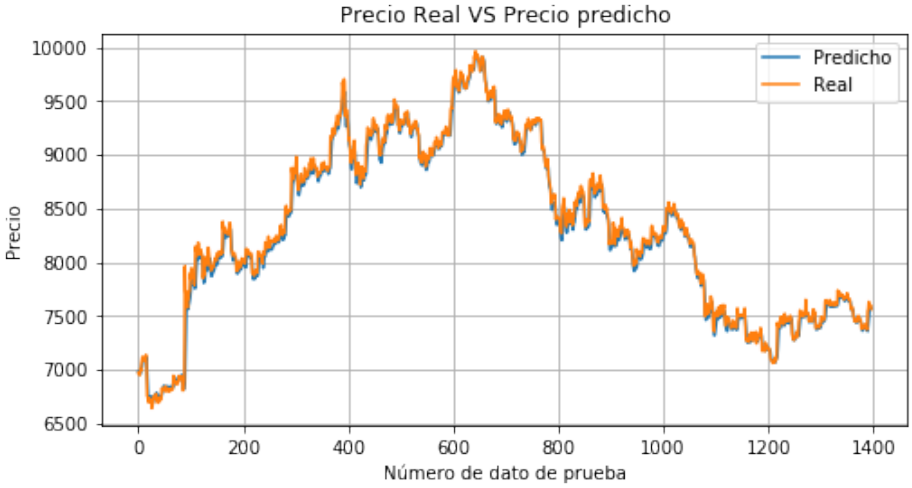

Figure 12: Precio pronosticado por la red neuronal lSTM y precio real para el máximo precio por hora.

6.2.4 Precio mínimo:

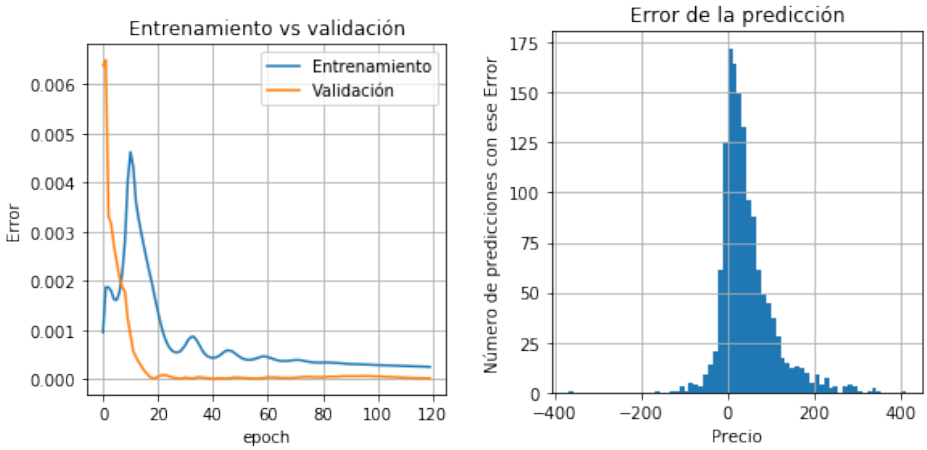

Figure 13: Error durante el entrenamiento; histograma del error de predicción para el precio mínimo por hora.

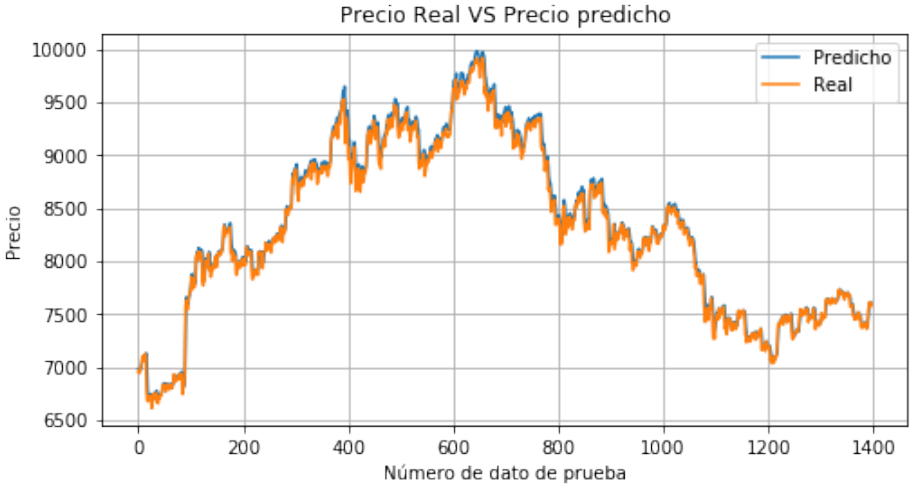

Figure 14: Precio pronosticado por la red neuronal lSTM y precio real para precio mínimo por hora.

Precio mínimo:

En los histogramas que se presentan en las imágenes se puede evidenciar

6.3 Clasificador

Se ralizan varias pruebas exploratorias que permiten establecer valores tentativos para los parámetros θ, ϕ para cada marco de tiempo utilizado, si se utiliza la serie de tiempo con intervalos de 1 hora se utiliza $\theta = 2.5\ UDST$ y $\phi = 10\ BTC$, por el contrario si se utiliza la serie de tiempo con intervalos de 3 minutos se utiliza $\theta = 0.5\ UDST$ y $\phi = 2\ BTC$. Se escoge $d = 6$, $WINDOW = 9$, $INTERVAL = 3$ ya que presentan la mayor estabilidad en términos de generar modelos entrenados similares al iniciar con parámetros diferentes.

Se obtien modelos disimiles al utilizar diferentes parámetros inciales para entrenar el modelo de clasificación con la serie de tiempo de intervalos de 1 hora, se cree esto se debe al pequeño tamaño del conjunto de datos disponible para entrenar y testear el modelo, lo cual hace que no se cuente con suficiente información para generar un modelo que abstraiga las características principales del conjunto de entrenamiento. Los parámetros θ, ϕ, d, $WINDOW$, $INTERVAL$ pueden se usados luego como variables para generar un conjunto de clasificadores que comunmente funcionen como un clasificador grande y más estable.

Se considera que no es bueno entrenar los modelos con tan pocos datos ya que se suelen sobreentrenar y no permiten realizar buenas predicciones. Para el clasificador Random Forest se generaron 1000 árboles con una profundidad máxima de 60, el clasificador de Regresión Logística compara las tres clases entre sí (multiclass), el clasificador tipo Red Neuronal Unidireccional utiliza 4 capas escondidas de dimensiones (100,250,150,80) y una función de activación logística para cada neurona.
Todos los modelos reportan ganancias si se raliza trading con los datos del conjunto de validación debido a que la serie tiene un tendencia alsista en ese intervalo. Se presentan las gráficas del compotamiento de las inversiones utilizando cada modelo y cada conjunto de atributos, se inicia con 1BTC que al principio de la serie de tiempo de testeo se encontraba en 9405 USDT. La línea azul representa el total del patrimonio expresado en USDT, la línea amarilla muestra el cambio en el valor del precio del BTC.

6.3.1 1 Hora

- **Random Forest**

Figure 15: Rendimiento de la estrategia de trading usando un clasificador tipo Random Forest entrenado con los atributos sugeridos en el artículo reproducido

Figure 16: Rendimiento de la estrategia de trading usando un clasificador tipo Random Forest entrenado con los atributos propuestos

- **Logistic Regression**

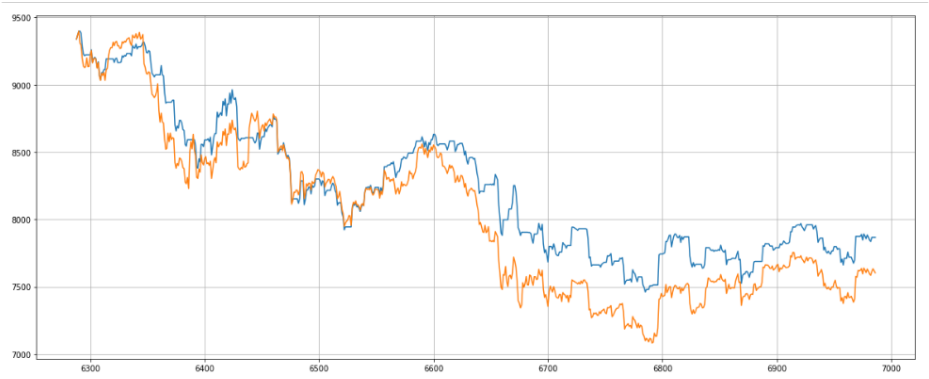

Figure 17: Rendimiento de la estrategia de trading usando un clasificador tipo Logistic Regression entrenado con los atributos sugeridos en el artículo reproducido

Figure 18: Rendimiento de la estrategia de trading usando un clasificador tipo Logistic Regression entrenado con los atributos propuestos

- **Neural Network (Multi Layer Perceptron)**

Figure 19: Rendimiento de la estrategia de trading usando un clasificador tipo Neural Network (Multi Layer Perceptron) entrenado con los atributos sugeridos en el artículo reproducido

Figure 20: Rendimiento de la estrategia de trading usando un clasificador tipo Neural Network (Multi Layer Perceptron) entrenado con los atributos propuestos

6.3.2 3 Minutos

- **Random Forest**

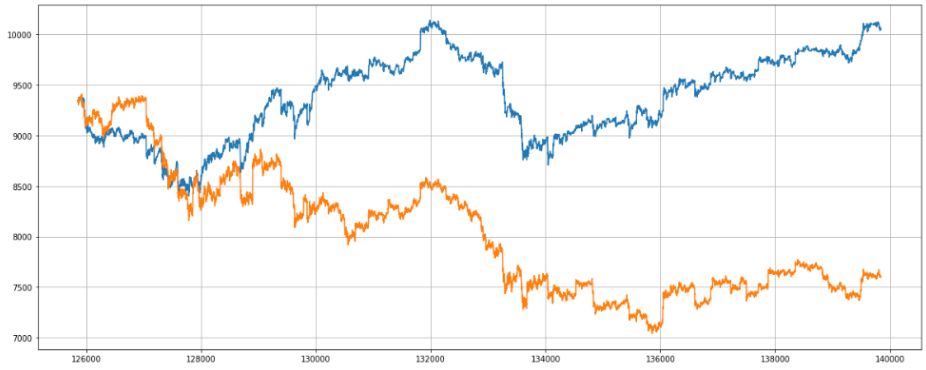

Figure 21: Rendimiento de la estrategia de trading usando un clasificador tipo Random Forest entrenado con los atributos sugeridos en el artículo reproducido

Figure 22: Rendimiento de la estrategia de trading usando un clasificador tipo Random Forest entrenado con los atributos propuestos

- **Logistic Regression**

Figure 23: Rendimiento de la estrategia de trading usando un clasificador tipo Logistic Regression entrenado con los atributos sugeridos en el artículo reproducido

Figure 24: Rendimiento de la estrategia de trading usando un clasificador tipo Logistic Regression entrenado con los atributos propuestos

- **Neural Network (Multi Layer Perceptron)**

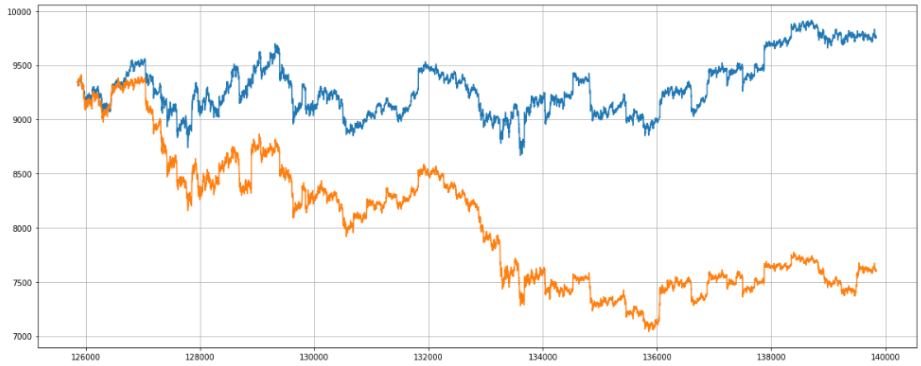

Figure 25: Rendimiento de la estrategia de trading usando un clasificador tipo Neural Network (Multi Layer Perceptron) entrenado con los atributos sugeridos en el artículo reproducido

Figure 26: Rendimiento de la estrategia de trading usando un clasificador tipo Neural Network (Multi Layer Perceptron) entrenado con los atributos propuestos

Los modelos entrenados con el marco de 1 hora no lográn obtener ganancias en el conjunto de testeo, sin embargo se observa que logran obtener un mejor retorno que el obtenido al no realizar ninguna compra o venta. Los modelos entrenados con el marco de 3 minutos logran obtener ganancias en el conjunto de testeo, y se oberva que los modelos entrenados con los atributos propuestos por nosotros lográn obtener más ganancias que los modelos entrenados con los atributos del artículo reproducido.

Es posible que la pequeña cantida de datos en el DataFrame con el marco de una hora sea el responsable de generar modelos que no lográn reproducir adecuadamente a un trader y por ende generen pérdidas, esto se observa principalmente en que los modelos entrenados con el marco de 1 hora son altamente sensibles a las condiciones iniciales y pequeños cambios en las configuraciones del modelo, como lo puede ser cambiar la semilla el generador aleaorio para realizar Stochastic Gradient Descent, o disminuir en un 10% la cantidad de árboles utilizados. Fue usual que se hiciera necesario hace pequeños cambios a los paŕametros de los modelos con tal de obtener un buen predictor, el más inestable fue la red neuronal, y así mismo fue posible obtener modelos que conseguían grandes ganancias como también era usual obtener un modelo que no hiciera ningún movimiento porque no predecía con suficiente certeza los movimientos.

Se observó que la serie con separación de 1 hora solía dar mejores resultados si se tomaban en cuenta los último 10 pasos temporales, es decir, es altamente dependiente de la historia, sin embargo, los modelos más estables se obtuvieron usando los parámetros ya mencionados.

7 Conclusiones

- Aunque parezca contradictorio, resulta mucho mas favorable usar una ventana de valores bastante pequeña del orden de 2 a 4 valores para predecir el siguiente. Cuando la ventana de precios usada es muy grande, el error de predicción del modelo aumenta significativamente.

- El conjuto de las redes neuronales LSTM resultan ser un muy destacable sistema predictivo bastante efectivo capaz de predecir el espectro de precios que el par BTC/USDT puede llegar a alcanzar durante cada hora con un error promedio de 12 USDT.

- Se implemento un modelo de clasificación que permite predecir la dinámica del valor del BTC en USDT y que al implementarse con una estratégia simple de trading permite generar un agente que al realizar transacciones cada 3 minutos genera ganacias superiores al 8 % mensual aún cuando el BTC tiene una tendencia bajista.

- Es necesario aumentar la cantidad de datos en el DataFrame de la serie de tiempo con intervalos de 1 hora de tal forma que se puedan entrenar eficientemente los modelos de clasificación evitando de esta forma sobre entrenar los datos y extraer las características generales del comportamiento del mercado.

References

[1] Felix A. Gers; Jürgen Schmidhuber; Fred Cummins. Learning to forget: Continual prediction with lstm.

[2] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. 1997.

[3] F.; Eck D.; Schmidhuber J.; Gers F. Schmidhuber, J.; Gers. Long short term memory networks for anomaly detection in time series. 2015.

[4] Muhammad J. Amjad;Devavrat Shah. Trading bitcoin and online time series prediction. 2017.

[5] I.; La Rosa M.; Dumas M. Tax, N.; Verenich. Predictive business process monitoring with lstm neural networks. 2017.