Modelo predictivo de rentabilidad de criptomonedas para un futuro cercano

Juan David García Álvarez

Trabajo de grado presentado para optar al título de Especialista en Analítica y Ciencia de Datos

Asesor
Sebastián Rodríguez Colina, Machine Learning Engineer

Universidad de Antioquia
Facultad de Ingeniería
Especialización en Analítica y Ciencia de Datos
Medellín, Antioquia, Colombia
2022

Cita	García Álvarez [1]
Referencia Estilo IEEE (2020)	[1] J.D. García Álvarez, "Modelo predictivo de rentabilidad de criptomonedas para un futuro cercano", Trabajo de grado especialización, Especialización en Analítica y Ciencia de Datos, Universidad de Antioquia, Medellín, Antioquia, Colombia, 2022.

Especialización en Analítica y Ciencia de Datos, Cohorte III.

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

RESUMEN

Una criptomoneda es un activo digital que emplea un cifrado criptográfico para garantizar su titularidad, asegurar la integridad de las transacciones y controlar la creación de unidades adicionales. Estas monedas no existen de forma física: se almacenan en una cartera digital. El valor de las criptomonedas varía en función de la oferta, de la demanda y del compromiso de los usuarios. Este valor se forma en ausencia de mecanismos eficaces que impidan su manipulación, como los presentes en los mercados regulados de valores. En muchas ocasiones los precios se forman también sin información pública que las respalde.

Cada día aumenta la cantidad de personas y analistas que consideran las criptomonedas como uno de los activos más rentables para invertir, debido a esto, muchos inversores buscan la manera de hacer un pronóstico ya sea de precio o rentabilidad que pueda presentar la criptomoneda en un futuro para obtener ganancias. Como es bien sabido, las criptomonedas se caracterizan por una alta volatilidad, por eso su precio puede aumentar o también disminuir bruscamente. Por lo tanto, aquellos que decidan invertir en criptomonedas deben saber aprovechar la volatilidad a su favor vendiendo o comprando en los instantes de tiempo precisos y fundamentalmente limitando lo más posible el riesgo. Para hacer un pronóstico de criptomonedas preciso se deben considerar algunos procesos y mecanismos que ayuden a comprender qué está sucediendo dentro del mercado. Una parte del análisis, en el cual no se centra completamente este trabajo, se puede basar en el conocimiento de ciertos factores macroeconómicos, políticos y financieros. Por lo tanto, como existen tantas consideraciones en este mercado, se debe tener cuidado para no sacar conclusiones apresuradas.

En el presente trabajo se busca encontrar un modelo predictivo que permita realizar un pronóstico, desde el punto de vista de la analítica de datos y el aprendizaje automático, de la rentabilidad de una de las tantas criptomonedas existentes, teniendo en cuenta las particularidades mencionadas de este mercado.

Palabras clave — criptomonedas, rentabilidad, modelos de pronóstico, aprendizaje automático, series de tiempo.

Repositorio GitHub: https://github.com/juanda9709/Crypto_Forecasting-

I. DESCRIPCIÓN DEL PROBLEMA

Hoy en día, el concepto de criptomonedas se viene divulgando bastante en la economía mundial, en las personas naturales e incluso en las organizaciones como bancos o asociaciones de inversión. Se estima que el número de criptomonedas que existe actualmente gira en torno a 10 mil según CoinMarketCap (2022), las cuales son ofrecidas en las principales, medianas y especializadas plataformas de intercambio que se encuentran en todo el mundo.

El valor de una criptomoneda y el cambio de este en el tiempo está altamente interconectado entre las diferentes criptomonedas, por ejemplo, Bitcoin históricamente ha sido un importante impulsor de los cambios de precios en otras criptomonedas, pero igualmente otras monedas también afectan el mercado.

El mercado de las criptomonedas, aunque es de los más populares, tiene la particularidad de ser muy volátil debido a la gran especulación e inversión de las personas. De acuerdo con lo anterior, los precios que fluctúan rápidamente han convertido en millonarios a unos pocos afortunados y han causado pérdidas aplastantes a otros.

1. Problema de negocio

Las criptomonedas se han convertido en un mercado extremadamente popular y volátil, que ofrece tanto ganancias como pérdidas masivas a los inversores. Estos activos se negocian ampliamente en las plataformas de intercambios de cifrado (Exchanges), con un volumen promedio de $41 mil millones de dólares negociados diariamente entre los años 2020 y 2021, según CryptoCompare (al 25 de julio de 2021).

Con este proyecto se quiere desarrollar una solución que logre disminuir las pérdidas a las que está expuesto un inversor en el mercado de criptomonedas. Específicamente, se desea construir un modelo predictivo que permita anticipar de alguna manera la variación en los precios de estos criptoactivos a corto plazo con el fin de prevenir pérdidas al realizar una inversión, y al mismo tiempo, encontrar buenas oportunidades de compra.

2. Aproximación desde la analítica de los datos

Una tarea fundamental en el sector financiero es predecir cómo se comportarán los precios de un activo en un futuro próximo. Se usarán los datos comerciales en un determinado rango de tiempo de algunas criptomonedas como datos de entrenamiento para generar un modelo predictivo con cual se desea predecir si el precio de la criptomoneda subirá o bajará, y en qué medida, es decir, el rendimiento de los activos.

Para analizar los cambios de precio de un activo, se puede tratar con la diferencia de precio. Sin embargo, las diferentes criptomonedas exhiben diferentes escalas de precios, por lo que sus rendimientos no son fácilmente comparables. Se puede resolver este problema calculando el cambio porcentual en el precio, también conocido como rendimiento, este rendimiento coincide con el cambio porcentual en el capital invertido. La predicción de los rendimientos de las criptomonedas es una tarea de pronóstico muy desafiante desde el punto de vista de la analítica de datos debido a su volatilidad, la naturaleza no estacionaria de los datos, la manipulación del mercado y que las condiciones del mercado cambian muy rápidamente.

El principal objetivo es construir modelos de aprendizaje automático para predecir los rendimientos de determinadas criptomonedas apuntando a un futuro cercano (horizonte de 15 minutos) y enfocándose en mayor medida en prevenir las posibles pérdidas.

En términos generales, el objetivo es modelar una serie temporal utilizando variables generadas a partir del comportamiento pasado de la serie y que sirven como entrada para los modelos predictivos. Teniendo en cuenta las características de la variable respuesta, este problema se aborda como un problema de regresión donde el rendimiento que se desea predecir es un valor continuo y depende de los valores que las variables toman en instantes anteriores.

3. Origen de los datos

Los datos con los cuales se trabaja en este proyecto se obtuvieron directamente desde la plataforma Kaggle (son de libre acceso y se pueden obtener desde el siguiente link: https://www.kaggle.com/c/g-research-crypto-forecasting), este conjunto de datos fue respaldado en esta plataforma por la empresa G-Research con el objetivo de realizar una competencia en la cual se quiere obtener el modelo que haga la mejor predicción del rendimiento de todas las criptomonedas presentes en la base de datos.

El conjunto de datos consta de millones de filas con registros de información comercial de 14 criptomonedas populares que abarcan un intervalo de tiempo desde inicio de 2018 hasta septiembre de 2021. Cada fila de datos de comercio de estas criptomonedas corresponde a cada minuto durante el espacio de tiempo mencionado y con base en esto se diseñarán los respectivos modelos de pronóstico para el rendimiento de las criptomonedas.

4. Variable Objetivo

El objetivo del modelo es predecir el rendimiento logarítmico en un futuro cercano para el precio de la criptomoneda. Los rendimientos se utilizan ampliamente en finanzas pero se prefieren los rendimientos logarítmicos para el modelado matemático de series temporales. En las finanzas este tipo de rendimientos facilitan la interpretación de los resultados cuando se quieren observar en

un determinado intervalo dado que son aditivos a lo largo del tiempo. Para calcular el rendimiento logarítmico, simplemente se puede tomar el logaritmo de la relación entre los dos precios: el actual y el futuro (dependiendo del horizonte de tiempo que se está considerando).

Cada fila del dataset contiene la variable "Target", la cual es el objetivo de la predicción. Esta variable surge calculando los retornos logarítmicos de cada criptomoneda en un intervalo de 15 minutos a futuro. A continuación, se presentan con más detalle las fórmulas utilizadas para el cálculo de esta variable:

$$R^a(t) = log(P^a(t+16) \ / \ P^a(t+1))$$

Donde “a” representa la criptomoneda sobre la cual se está realizando el cálculo y “P” representa el precio de cierre de la criptomoneda en ese minuto. Los rendimientos de los criptoactivos están altamente correlacionados, siguiendo en gran medida el mercado criptográfico general. Como se quiere probar la capacidad para predecir los rendimientos de activos individuales, se realizan operaciones adicionales para eliminar en cierta medida la influencia de los demás activos del mercado a los rendimientos individuales, por lo tanto, para obtener la variable objetivo se tiene lo siguiente:

$$M(t) = \frac{\sum_a w^a R^a(t)}{\sum_a w^a}$$

$$\beta^a = \frac{\langle M \cdot R^a \rangle}{\langle M^2 \rangle}$$

$$\text{Target}^a(t) = R^a(t) - \beta^a M(t)$$

Donde “R” corresponde al cálculo resultado de la fórmula anterior, las variables “Beta” y “M(t)” corresponden a parámetros calculados basados en un valor de peso “w” asignado a cada criptomoneda dependiendo de su comportamiento en el mercado (más adelante se presenta esta información).

5. *Métrica de Machine Learning (Desempeño del modelo)*

La métrica para la evaluación del desempeño del modelo de machine learning será mediante el coeficiente de correlación de Pearson, la cual es la que se sugiere y con la que se evalúa la competencia de Kaggle de donde se tomaron los datos. En esta métrica los resultados más cercanos a 1 indican un mejor desempeño del modelo. Esto permite comparar nuestro modelo con otros modelos existentes que participen en la competencia.

6. *Métrica De Negocio*

En el negocio el rendimiento es la ganancia o pérdida respecto a una inversión durante un período de tiempo, expresada como una proporción de la inversión original. Comprende cualquier cambio en el valor que el inversor recibe de su inversión independientemente de su tamaño, en este caso se mide como un porcentaje de la cantidad invertida.

Una de las propiedades importantes de los rendimientos logarítmicos para este proyecto es que son simétricos, mientras que los rendimientos ordinarios no lo son. En los rendimientos ordinarios los porcentajes positivos y negativos de igual magnitud no se anulan entre sí y dan lugar a un cambio neto, pero los rendimientos logarítmicos de igual magnitud, pero signos opuestos se anulan entre sí. Por ejemplo, esto significa que una inversión de $100 que produce un rendimiento ordinario del 50% seguido de un rendimiento ordinario de -50% dará como resultado $75, mientras que una inversión de $100 que produce un rendimiento logarítmico del 50% seguido de un rendimiento logarítmico de -50 % volverá a $100.

La métrica de negocio para este proyecto será el error SMAPE (Symmetric mean absolute percentage error) la cual indicará si el modelo cumple con los objetivos planteados.

SMAPE (Symmetric mean absolute percentage error)

El SMAPE es una medida que sirve para calcular la precisión de los pronósticos basada en errores porcentuales. Está definido de la siguiente manera:

$$\text{SMAPE} = \frac{100\%}{n} \sum_{t=1}^{n} \frac{|F_t - A_t|}{(|A_t| + |F_t|)/2}$$

Donde “F” corresponde al valor de pronóstico entregado por el modelo y “A” corresponde al valor actual o real en este instante de tiempo.

Esta versión del SMAPE, propuesto por primera vez por Makridakis y Hibbon (2000), es el error promedio de los pronósticos realizados en un horizonte determinado. Dos de sus ventajas es que evita el problema de volverse muy grande o indeterminado cuando se tienen valores muy cercanos de cero (lo que sucede en este conjunto de datos) y que sus valores no crecen infinitamente, sino que fluctúan entre 0 y 200% (Rink, 2021).

Igualmente, se deben tener en cuenta las desventajas que puede presentar. Por cómo está definido, Goodwin y Lawton (1999) señalaron y demostraron que SMAPE penaliza más las subestimaciones que las sobreestimaciones. Cabe aclarar que sobrestimar quiere decir que el pronóstico es superior a la cantidad real mientras que subestimar indica que el pronóstico es menor a la cantidad real.

Por otro lado, la característica principal de su uso en este proyecto es que este error castiga en mayor medida los pronósticos en los cuales las predicciones tienen signo opuesto al valor real y precisamente este tipo de errores son los que se quieren prevenir con el desarrollo del modelo. El objetivo final es tener un modelo en el cual este error porcentual sea lo más pequeño posible, teniendo en cuenta que se considerará como aceptable un resultado de SMAPE menor al 30%.

II. DATOS

1. *Datos Originales*

Este conjunto de datos contiene la información sobre el historial de información financiera para 14 criptomonedas diferentes. Cada fila del dataset corresponde a los datos del mercado para cada criptomoneda en un intervalo de tiempo de un minuto. La primera columna contiene el índice de tiempo donde fueron tomados los datos, la segunda identifica la criptomoneda correspondiente, las siguientes 6 columnas son los datos comerciales que tuvo la moneda en ese minuto y la última columna es el objetivo de la predicción:

	timestamp	Asset_ID	Count	Open	High	Low	Close	Volume	VWAP	Target
0	1514764860	2	40.0	2376.580000	2399.5000	2357.1400	2374.590000	19.233005	2373.116392	-0.004218
1	1514764860	0	5.0	8.530000	8.5300	8.5300	8.530000	78.380000	8.530000	-0.014399
2	1514764860	1	229.0	13835.194000	14013.8000	13666.1100	13850.176000	31.550062	13827.062093	-0.014643
3	1514764860	5	32.0	7.659600	7.6596	7.6567	7.657600	6626.713370	7.657713	-0.013922
4	1514764860	7	5.0	25.920000	25.9200	25.8740	25.877000	121.087310	25.891363	-0.008264
5	1514764860	6	173.0	738.302500	746.0000	732.5100	738.507500	335.987856	738.839291	-0.004809
6	1514764860	9	167.0	225.330000	227.7800	222.9800	225.206667	411.896642	225.197944	-0.009791
7	1514764860	11	7.0	329.090000	329.8800	329.0900	329.460000	6.635710	329.454118	NaN
8	1514764920	2	53.0	2374.553333	2400.9000	2354.2000	2372.286667	24.050259	2371.434498	-0.004079
9	1514764920	0	7.0	8.530000	8.5300	8.5145	8.514500	71.390000	8.520215	-0.015875
10	1514764920	1	235.0	13835.036000	14052.3000	13680.0000	13828.102000	31.046432	13840.362591	-0.015037
11	1514764920	5	10.0	7.656800	7.6569	7.6567	7.656700	3277.475494	7.656749	-0.014534
12	1514764920	7	1.0	25.897000	25.8970	25.8970	25.897000	1.468019	25.897000	-0.029902
13	1514764920	6	192.0	738.507500	745.1400	732.4900	738.260000	232.793141	738.268967	-0.004441
14	1514764920	9	212.0	225.133333	227.3800	222.4500	224.856667	3640.502706	224.446625	-0.012991

Fig. 1. Muestra del conjunto de datos

A continuación, se da una descripción más detallada de las características de cada variable:

Tabla 1. Descripción de variables

Nombre de variable	Descripción	Tipo de dato
TIMESTAMP	Marcas de tiempo que indican el minuto al cual corresponden los datos de esa fila. (UNIX timestamp).	INT
ASSET_ID	ID que identifica a cuál criptomoneda corresponden los datos de la fila.	INT
COUNT	Número total de operaciones que se realizaron en ese minuto.	FLOAT
OPEN	Precio de la criptomoneda al comienzo del minuto (USD).	FLOAT
HIGH	Precio más alto alcanzado durante el intervalo de tiempo (USD).	FLOAT
LOW	Precio más bajo alcanzado durante el intervalo de tiempo (USD).	FLOAT
CLOSE	Precio de la criptomoneda finalizar el minuto (USD).	FLOAT
VOLUME	Cantidad de criptomonedas negociadas durante el minuto.	FLOAT
VWAP	Precio medio del activo durante el último minuto.	FLOAT
TARGET	Rentabilidad logarítmica de la criptomoneda en un horizonte de 15 minutos.	FLOAT

Este conjunto de datos principal disponible en la plataforma es llamado “train.csv” tiene un tamaño de 2.82 GB y cuenta con un total de 24.236.806 registros.

Igualmente, se cuenta con una tabla de referencia llamada “asset_details.csv” disponible en el mismo repositorio de la competencia de Kaggle. En esta se presenta más información acerca de las diferentes criptomonedas del conjunto de datos; se cuenta con un identificador de cada criptomoneda, su respectivo nombre y el peso asignado para el cálculo de la variable objetivo “Target”. Específicamente, en la competencia los pesos asociados a cada criptomoneda están determinados por el logaritmo de la capitalización de mercado de cada producto (en USD), en un punto fijo en el tiempo. Se asignaron pesos para dar más relevancia a las criptomonedas con mayores volúmenes de mercado para garantizar que las criptomonedas más pequeñas no impacten de manera desproporcionada en los modelos.

A continuación, se presenta la tabla mencionada:

	Asset_ID	Weight	Asset_Name
0	2	2.397895	Bitcoin Cash
1	0	4.304065	Binance Coin
2	1	6.779922	Bitcoin
3	5	1.386294	EOS.IO
4	7	2.079442	Ethereum Classic
5	6	5.894403	Ethereum
6	9	2.397895	Litecoin
7	11	1.609438	Monero
8	13	1.791759	TRON
9	12	2.079442	Stellar
10	3	4.406719	Cardano
11	8	1.098612	IOTA
12	10	1.098612	Maker
13	4	3.555348	Dogecoin

Fig. 2. Identificación y peso asignado a cada criptomoneda

2. Datasets

El conjunto de datos con el que se trabaja corresponde a una serie temporal, por tanto, los conjuntos de entrenamiento y pruebas se construyen considerando estas características. Una serie de tiempo o serie temporal es una secuencia de datos, medidos en determinados momentos del

tiempo, ordenados cronológicamente, espaciados de manera uniforme y usualmente dependientes entre sí. El principal objetivo al trabajar con una serie de tiempo es hacer pronósticos.

El proceso de pronosticar consiste en predecir el valor futuro de una serie de tiempo, bien sea modelando la serie únicamente en función de su comportamiento pasado (autorregresivo) o empleando variables externas (exógenas). El conjunto de datos cuenta con información desde enero del 2018 hasta septiembre de 2021, para generar los conjuntos de entrenamiento y prueba primero que todo se extraen los datos de la criptomoneda de interés. Después de esto se escogen los datos con los que se va a trabajar en términos de días (tomando siempre desde el último dato que se tiene hacia atrás) y la proporción de estos datos que se usarán para crear el conjunto de prueba. A continuación, se presenta un ejemplo de partición del conjunto de datos donde se seleccionó las últimas 12 horas de las que se tiene de información y un 30% de estos datos para conformar el conjunto de test:

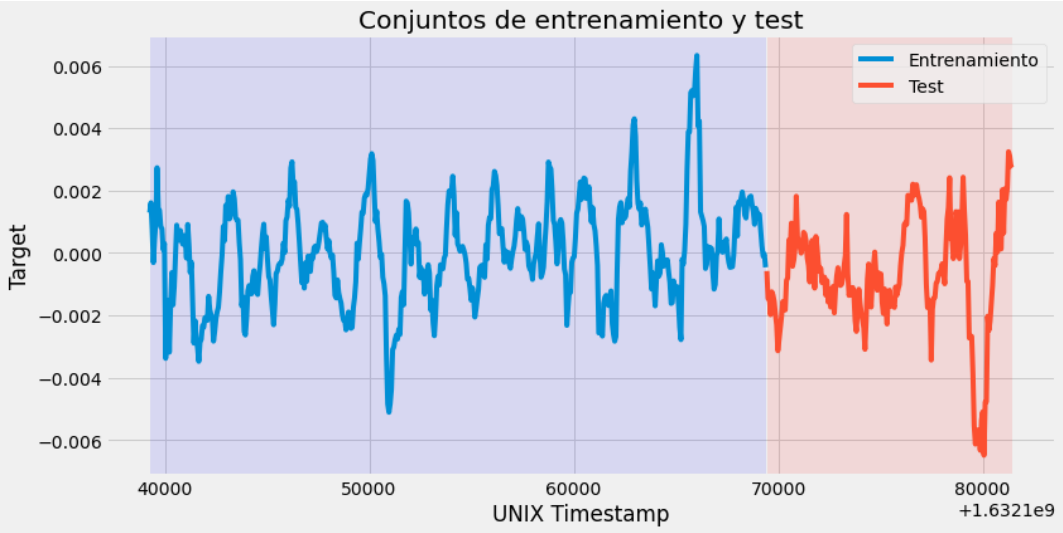

Fig. 3. Conjuntos de entrenamiento y prueba

Para el entrenamiento de los modelos se realiza el proceso de validación cruzada donde el objetivo es entrenar inicialmente con un pequeño subconjunto de datos, con estos pronosticar los puntos de datos posteriores seleccionados para validación y luego verificar la precisión de este pronóstico. Posteriormente, los mismos datos de validación anteriores se incluyen como parte del siguiente subconjunto de datos de entrenamiento y se pronostican los siguientes datos escogidos

para validación. Esta técnica permite evaluar igualmente el rendimiento del modelo durante el proceso de entrenamiento. En la siguiente figura se puede observar gráficamente en qué consiste la construcción de los conjuntos de datos para la validación cruzada que se usa en el proyecto:

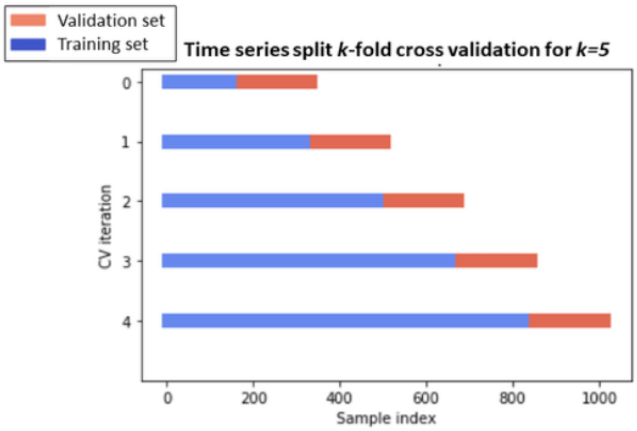

Fig. 4. Validación cruzada en conjunto de entrenamiento.

3. *Análisis Descriptivo*

Se presenta información básica acerca del conjunto de datos donde se observa que se tiene un total de 24.236.806 registros y cada uno con sus 10 variables asociadas:

```
RangeIndex: 24236806 entries, 0 to 24236805
Data columns (total 10 columns):
 #   Column     Dtype
---  ------     -----
 0   timestamp  int64
 1   Asset_ID   int64
 2   Count      float64
 3   Open       float64
 4   High       float64
 5   Low        float64
 6   Close      float64
 7   Volume     float64
 8   VWAP       float64
 9   Target     float64
```

Fig. 5. Descripción del conjunto de datos

El conjunto de datos cuenta con información de 14 criptomonedas, en la siguiente figura se observa cómo es la distribución respecto al número de registros que se tienen para cada moneda:

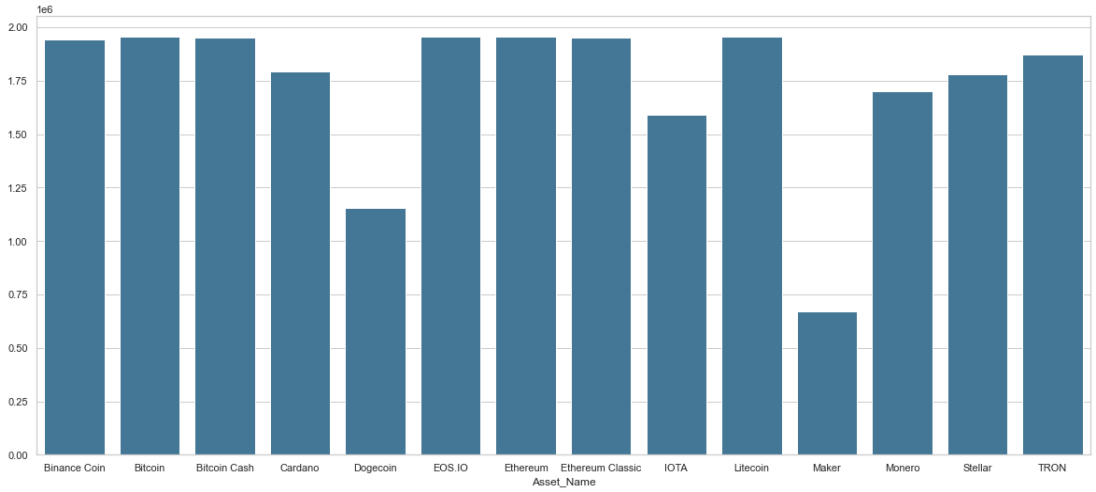

Fig. 6. Cantidad de registros por criptomoneda

En cuanto a valores nulos se tienen 750.347 en la variable "Target" y 9 en la variable "VWAP". Los registros nulos de la variable "Target" están distribuidos entre las diferentes criptomonedas como se muestra a continuación:

	Criptomoneda	Cantidad Nulos
0	Bitcoin Cash	4861
1	Binance Coin	13415
2	Bitcoin	304
3	EOS.IO	2302
4	Ethereum Classic	9326
5	Ethereum	340
6	Litecoin	521
7	Monero	239603
8	TRON	21014
9	Stellar	61816
10	Cardano	18731
11	IOTA	193106
12	Maker	40025
13	Dogecoin	144974

Fig. 7. *Valores nulos en variable "Target" para cada criptomoneda*

Los valores nulos en esta variable se deben a la forma en que esta se calcula (ver fórmula en numeral 2.4) y es causado principalmente por la ausencia de información en determinados minutos en la base de datos.

Los pocos valores nulos que se presentan en la variable “VWAP” corresponden solo a la criptomoneda Maker y se deben a que en esos instantes de tiempo la variable “Volume” tiene un valor de 0, por lo tanto, al calcular el valor de “VWAP” se obtiene un valor indeterminado:

timestamp	Asset_ID	Count	Open	High	Low	Close	Volume	VWAP	Target
1592173560	10	2.0	501.00	501.00	501.00	501.00	0.0	NaN	NaN
1592173620	10	4.0	501.00	501.00	501.00	501.00	0.0	NaN	NaN
1592426160	10	2.0	529.77	529.77	529.77	529.77	0.0	NaN	0.007356
1592823720	10	2.0	503.60	503.60	503.60	503.60	0.0	NaN	-0.005672
1593008940	10	2.0	484.16	484.16	484.16	484.16	0.0	NaN	-0.004281
1593013920	10	2.0	480.00	480.00	480.00	480.00	0.0	NaN	NaN
1593014040	10	2.0	480.00	480.00	480.00	480.00	0.0	NaN	NaN
1593014340	10	6.0	479.07	479.07	479.07	479.07	0.0	NaN	NaN
1593016440	10	4.0	478.00	478.00	475.00	478.00	0.0	NaN	NaN

Fig. 8. Valores nulos en variable “VWAP” en el conjunto de datos

4. Análisis de datos descriptivo de criptomoneda de interés – Bitcoin

Para el desarrollo del proyecto se consideró trabajar únicamente con el Bitcoin. Para esta moneda, se tiene un total de 1.956.282 registros que van desde el primero de enero de 2018 hasta el 21 de septiembre de 2021.

```
Int64Index: 1956282 entries, 1514764860 to 1632182400
Data columns (total 9 columns):
 #   Column    Dtype
---  ------    -----
 0   Asset_ID  int64
 1   Count     float64
 2   Open      float64
 3   High      float64
 4   Low       float64
 5   Close     float64
 6   Volume    float64
 7   VWAP      float64
 8   Target    float64
dtypes: float64(8), int64(1)
```

Fig. 9. Descripción del subconjunto de datos correspondiente al Bitcoin

III. PROCESO DE ANALÍTICA

1. *Pipeline Principal*

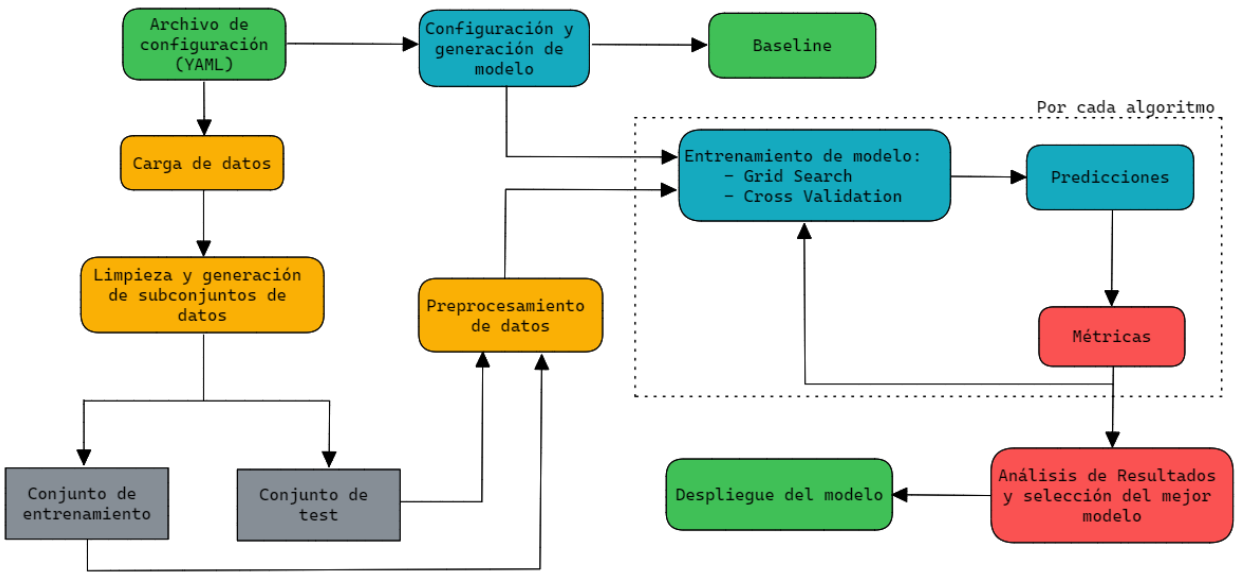

Fig. 10. Pipeline general

Se da una explicación detallada de los trabajos que abarca y en qué consiste cada etapa del flujo de trabajo implementado para este proyecto:

2. *Archivo de configuración (YAML)*

Se dispone de un archivo de configuración (config.yml) desde el que se definen varios parámetros y conjuntos de parámetros que hacen parte del funcionamiento del proceso:

- **hyperparams:** Se definen los pasos del pipeline de preprocesamiento y los hiperparámetros específicos, dependiendo del algoritmo, con los que se desea entrenar el modelo (una sola iteración).
- **data:** Se define la ruta desde donde se leerán los datos, el ID de la criptomoneda objetivo (ID=1 en caso del Bitcoin), la cantidad de días de información con los que se quiere trabajar y la proporción de estos datos que se dejarán como conjunto de test.
- **metrics:** Se definen las métricas a calcular una vez se generan las predicciones del modelo utilizando con el conjunto de test.

- **search:** Se definen los pasos del pipeline de preprocesamiento y el conjunto de hiperparámetros (dependiendo del algoritmo), que conformarán el objeto GridSearch con el que se realiza la búsqueda exhaustiva de hiperparámetros y donde se quiere encontrar la mejor combinación de estos para el modelo.
- **export:** Se indica la ruta del directorio (se crea en caso de que no exista) donde se guarda un archivo con el pipeline del modelo serializado y en un archivo adicional los respectivos hiperparámetros con los que fue entrenado.
- **reports:** Se indica la ruta del directorio en el cual, una vez realizadas las predicciones, se almacena un archivo YAML que contiene los resultados del modelo para cada métrica considerada en el archivo de configuración.

3. Carga de datos, limpieza y generación de conjuntos de entrenamiento y test

Con la información indicada en el archivo de configuración se procede con la lectura de datos y se realiza un proceso de limpieza a este conjunto de datos crudos.

Inicialmente se extraen los datos asociados a la criptomoneda de interés y se verifica la cantidad de valores nulos, donde se tienen 304 en la variable “Target”. El manejo que se da a estos valores nulos es imputarlos con un valor de cero, que para este problema finalmente significa que no habrá ningún rendimiento. Estos 304 valores representan un porcentaje despreciable del conjunto de datos, por lo cual, realizar esta imputación no representa ningún impacto a considerar en el desarrollo del proyecto.

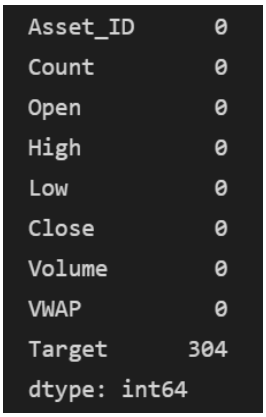

Fig. 11. Valores nulos en variable “Target” para el Bitcoin

En este problema los datos faltantes no solo están representados por campos con valores nulos, sino por la ausencia de filas en determinados instantes de tiempo. Se comprueba si faltan datos realizando la diferencia de tiempo entre dos filas consecutivas y se observa que existen vacíos en los datos ya que la diferencia entre dos filas consecutivas no siempre es de 60 segundos, cuando debería serlo dado el formato que se tiene en la estampa de tiempo:

```
60     1956136
120         78
180         12
240         11
420          9
Name: timestamp, dtype: int64
```

Fig. 12. Diferencia de tiempo (segundos) entre filas consecutivas

Se hace un proceso de reindexado de las filas, buscando tener un ordenamiento constante (minuto a minuto) en la estampa de tiempo y generar correctamente los conjuntos de entrenamiento/test en términos del tiempo (cantidad de días hacia atrás).

Ahora con los datos preparados se hace la partición de estos teniendo en cuenta la proporción que se asigna al conjunto de prueba. Se generan entonces dos "split" que se identificaran en el proceso como "train" y "test", el primero utilizado en la etapa de entrenamiento de los modelos y el segundo para la etapa de evaluación.

Es importante mencionar que, aunque la criptomoneda escogida es el Bitcoin, los pasos aquí realizados se pueden aplicar a cualquiera de las criptomonedas que se tienen en el conjunto de datos, con esto se podría explorar y ver el desempeño de este proceso con otras monedas.

4. *Preprocesamiento de datos*

Se usa la librería de python TSFRESH la cual fue desarrollada para realizar ingeniería de características a partir de series temporales y otros datos secuenciales. El uso de esta herramienta se divide en tres partes dentro del pipeline del modelo:

- **Generación de ventanas de tiempo para extracción de características:** Como el objetivo es realizar un pronóstico aproximado del rendimiento de la criptomoneda en los siguientes 15 minutos, una forma de abordar este problema es considerar una ventana de tiempo con un tamaño fijo que abarque 15 registros y a la cual se le aplica un desplazamiento minuto a minuto sobre la serie de tiempo ordenada. Cada una de estas ventanas desplazadas contiene la información del mercado correspondiente a los 15 minutos inmediatamente anteriores al valor a predecir. Se obtiene un DataFrame extendido con muchos segmentos de 15 registros, donde cada uno de estos segmentos es debidamente identificado con la estampa de tiempo correspondiente al valor que se quiere predecir. La longitud de este nuevo DataFrame depende directamente del tamaño del conjunto de datos seleccionado en etapas previas.

- **Extracción de características:** Se procede con la extracción de características para los datos que se tienen en cada ventana de tiempo. Con la generación de estas nuevas variables se busca tener más información y poder explicar de la mejor manera la variable objetivo mediante los modelos predictivos. La herramienta cuenta con tres diccionarios predefinidos que contienen diferentes conjuntos de características para extraer de los datos: ComprehensiveFCParameters, MinimalFCParameters y EfficientFCParameters.

  En teoría, se podría hallar un número ilimitado de variables con esta librería creando un diccionario de características a calcular con múltiples combinaciones de funciones y parámetros, pero para efectos prácticos se utilizaron los diccionarios predefinidos en la herramienta. Más adelante se explican las particularidades de cada uno y los rendimientos que presentaron al utilizarlos durante los experimentos. Finalmente, después del proceso de extracción se obtiene un conjunto de datos con la misma cantidad de registros inicial pero ahora con las características calculadas de los 15 minutos previos a ese instante de tiempo.

- **Selección/Filtro de características:** En el proceso de selección de características se busca identificar y seleccionar dentro de todos los atributos sólo los más relevantes con

el fin de reducir el tamaño del conjunto de datos y a su vez el costo computacional. Para esta selección de características, la librería implementa el algoritmo “Distributed and parallel time series feature extraction for industrial big data applications” (hrist, M., Kempa-Liehr, A.W. and Feindt, M.). Este algoritmo evalúa cada vector de características de forma individual e independiente con respecto a su importancia para predecir el rendimiento de la criptomoneda. El resultado de estas operaciones es un vector de valores p que cuantifica la importancia de cada característica para predecir la variable objetivo, este vector de valores p se evalúa mediante el procedimiento de Benjamini-Yekutieli (2001) para decidir qué características conservar.

5. *Escalamiento de datos*

El proceso de escalar los datos y la técnica con la que se haga puede depender de varios factores. En general, los algoritmos de aprendizaje automático que se basen sobre la hipótesis que los datos poseen una distribución Gaussiana requieren que estos sean estandarizados. De igual manera, cuando los datos tienen las características con órdenes de magnitud muy diferentes también suele ser necesario escalarlos.

Durante el desarrollo del proyecto se experimentó con dos técnicas de escalamiento de datos de la librería Scikit-learn para ver cual daba mejores resultados (en cada caso teniendo en cuenta el modelo utilizado): Normalización y Estandarización. Normalizar consiste en escalar los datos desde sus valores originales a un rango entre 0 y 1, por lo tanto, se requiere conocer cuál es el valor mínimo y máximo de la variable para realizar la operación. Por otra parte, estandarizar se refiere a escalar la distribución de los datos removiendo la media de los valores observados y haciendo que su desviación estándar sea 1. En los modelos basados en árboles de decisión no es estrictamente necesario realizar una estandarización o normalización de las variables. En la sección de resultados se presentan las métricas obtenidas durante la evaluación de estos modelos después de aplicar o no este paso de estandarización.

6. *Configuración/construcción de los modelos*

Utilizando la librería Scikit-Learn se procede a construir un objeto tipo pipeline que contendrá los pasos a ejecutar en cada experimento dependiendo de la información indicada en el

archivo de configuración. Para esto se adaptaron todos los pasos indicados en la sección del preprocesamiento de forma que sean compatibles con los métodos de entrenamiento y predicción de la librería. En el archivo de configuración se deben indicar en orden los pasos del preprocesamiento que se quieren realizar así como el modelo predictivo a entrenar. Adicionalmente, junto con el modelo se deben especificar sus respectivos hiperparámetros o conjunto de hiperparametros cuando se desea realizar una búsqueda exhaustiva de estos.

7. *Baseline*

En este paso se entrena y se evalúa un primer modelo básico que servirá como punto de referencia para comparar resultados de los demás modelos que se van explorando al aplicar diferentes técnicas y algoritmos buscando mejorar el desempeño. En el numeral 5 se explica con más detalle cómo se construyó este modelo.

8. *Entrenamiento, predicciones y métricas*

En este paso se ejecuta el pipeline y se realiza el entrenamiento del modelo especificado en el archivo de configuración. Una vez se tiene el modelo entrenado se realizan las predicciones para el conjunto de datos de prueba y con estas predicciones se calculan las métricas con las que se evalúa su desempeño y comportamiento.

En cuanto a las métricas de desempeño, para el cálculo del coeficiente de correlación de Pearson se empleó la función que dispone la librería Numpy y para calcular el error SMAPE se empleó la librería SKTIME. En esta última se tiene la particularidad de que se utiliza la misma definición de función para el cálculo del MAPE y SMAPE, la diferencia radica en que para calcular el SMAPE se debe considerar el argumento “symmetric = True”. Esta librería trabaja con la definición formal de SMAPE, para la cual se obtienen valores de error entre el 0 y 200%.

9. *Análisis de resultados y selección de modelo*

En este punto se evalúan los resultados obtenidos en cada experimento y basados en esto se selecciona el modelo final que presente los mejores resultados en cuanto a desempeño y rendimiento. El modelo con mejor desempeño se define con los resultados de las métricas

escogidas para el problema de negocio y por el lado del rendimiento se considera el tiempo y costo computacional que se requiere para realizar las predicciones.

La métrica de negocio definida para el proyecto será la principal característica para definir si un modelo es mejor que otro. En los casos que un modelo que requiera mucho tiempo de ejecución presente un desempeño mejor que otro modelo que tiene menos costo, pero esta diferencia no sea muy significativa, se puede entrar a evaluar la viabilidad de escoger el modelo con menor tiempo de ejecución.

10. Despliegue de modelo

En este paso se procede a exponer el modelo de aprendizaje automático como un servicio al cual se pueda acceder mediante una API. Una API sirve como intermediaria entre dos aplicaciones independientes que se comunican entre sí. Al desarrollar un modelo de aprendizaje automático se busca que este se encuentre disponible para que otras personas lo consuman. Para hacer esto se crea una API, utilizando el framework Fast API, que actúe como punto de entrada al modelo y mediante esta herramienta sea posible comunicarse e interactuar a través de solicitudes HTTP con el servicio que en este caso es el modelo final.

IV. MODELOS

1. Modelos Autorregresivos

Los modelos de autorregresión, como bien dice su nombre, son modelos que se regresan en sí mismos, es decir, se utilizan para realizar pronósticos del valor futuro de una misma variable en determinados momentos del tiempo ordenados cronológicamente. La principal adaptación que se necesita hacer para aplicar modelos de aprendizaje automático a problemas de pronóstico es transformar la serie temporal en un matriz en la que cada valor está asociado a la ventana temporal que le precede.

Una vez que los datos se encuentran re ordenados de esta forma, se puede entrenar cualquier modelo de regresión para que aprenda a predecir el siguiente valor de la serie. A los datos ordenados de esta manera se añaden también las nuevas características extraídas en los pasos de preprocesamiento con el fin de aportar información relevante al modelo para realizar las predicciones.

Para la creación de este tipo de modelos se emplea la librería SKFORECAST, específicamente un objeto del tipo "ForecasterAutoreg". A este objeto se entregan como argumento cualquier algoritmo de aprendizaje automático que funcionará como regresor y una ventana temporal que indica cuántos datos de la misma variable objetivo tomará hacia atrás en cada iteración para realizar las predicciones.

2. KNN (K Nearest Neighbors) Regressor

Este algoritmo es un método de aproximación simple no paramétrica basado en la regla del vecino más cercano, que consiste en estimar el valor de un dato desconocido a partir de las características del dato más próximo. El algoritmo KNN se adapta fácilmente a la regresión de funciones con valores continuos, mediante una medida de distancia en el espacio de características se determinan los k datos más cercanos al nuevo valor para aproximar una función. Esta función corresponde al promedio de los k valores más cercanos; si se considera el promedio aritmético (todos los datos dentro del grupo tienen igual relevancia) (Morales, 2008).

Es importante mencionar que todos los datos deben estar normalizados para evitar que las características en el conjunto de entrada con valores más altos dominen el cálculo de la distancia.

El objeto que se usó para este ejercicio es KNeighborsRegressor que pertenece a la librería Scikit-Learn.

3. *Random Forest Regressor*

Los algoritmos de árboles de decisión se pueden interpretar como árboles que comienzan con una raíz y siguen divisiones basadas en resultados variables hasta que se alcanza un nodo final y se da un resultado.

Los métodos de ensamble consisten en usar múltiples modelos, entrenados con los mismos datos, promediando los resultados de cada modelo y finalmente encontrando un resultado predictivo más preciso. Por otro lado, Bootstrapping es el proceso de muestrear aleatoriamente subconjuntos de un conjunto de datos en un número determinado de iteraciones y un número determinado de variables. Estos resultados luego se promedian juntos para obtener un resultado más poderoso.

El algoritmo de Bootstrapping Random Forest es un método de ensamble de árboles de decisión creados aleatoriamente a partir de los datos, promediando los resultados para generar un nuevo resultado que normalmente conduce a predicciones/clasificaciones sólidas. ajustar la profundidad para realizar las divisiones en cada decisión de los árboles de decisión.

El objeto que se usó para este ejercicio es RandomForestRegressor que pertenece a la librería Scikit-Learn.

4. *LGBM (Light Gradient Boosting Machine)*

Gradient Boosting Decision Tree (GBDT) es un método de ensamble de aprendizaje automático basado en árboles de decisión ampliamente utilizado, debido a su eficiencia, precisión e interpretabilidad. Este tipo de algoritmos pueden presentar algunos problemas cuando el número

de datos aumenta considerablemente, relacionados principalmente con el balance entre precisión y eficiencia.

LGBM es un framework desarrollado por Microsoft que usa algoritmos basados en árboles de decisión, con la eficiencia como objetivo principal. Está optimizado para que los árboles crezcan en la dirección de los mejores nodos (ayudando a una mejor administración de memoria). A diferencia de XGBoost, por ejemplo, que utiliza algoritmos basados en la clasificación previa, LightGBM usa algoritmos basados en histogramas para agilizar el entrenamiento y reducir el uso de memoria. Este algoritmo puede ser utilizado tanto para problemas de clasificación como para regresión.

El objeto que se usó para este ejercicio es LGBMRegressor que pertenece a la librería de LightGBM.

V. METODOLOGÍA

1. *Baseline*

El modelo baseline consistió en el entrenamiento de un algoritmo sencillo de tipo RandomForest con 30 estimadores y con la particularidad de que en la etapa de preprocesamiento de datos no se hizo ninguna extracción de características ni escalamiento, solo se realizó una limpieza de los valores nulos y se eliminó la columna que identifica la criptomoneda, mientras que el resto de columnas se mantuvieron iguales.

Tampoco se consideró su uso como un modelo de tipo autorregresivo, sino que se asumió inicialmente como un problema de regresión común donde el ordenamiento en el tiempo de la variable objetivo y de las otras características no influyen en la predicción.

El resultado de esta primera ejecución tuvo las siguientes métricas de desempeño:

Tabla 2. Desempeño del modelo Baseline

Modelo	Métrica	Valor
RandomForest Regressor	Coeficiente correlación de Pearson	0.006
	SMAPE	165%

En la siguiente figura se presenta una comparación entre un segmento de valores de predicciones y un segmento de valores reales:

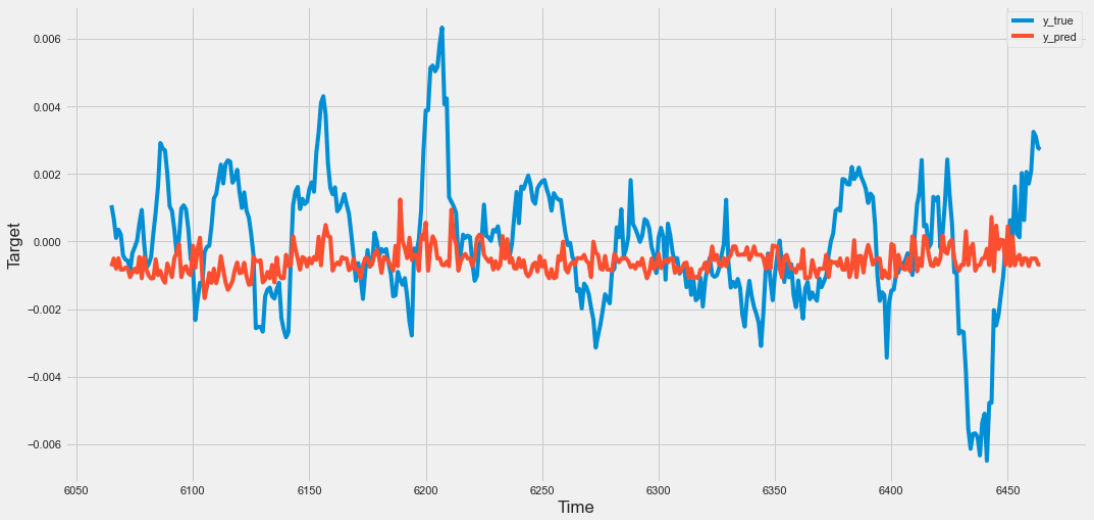

Fig. 13. Predicción vs Valor Real - Modelo Baseline

Se observa que efectivamente este modelo no presenta un buen desempeño y servirá solo como punto de referencia para iniciar con la búsqueda de un modelo predictivo óptimo que cumpla con los objetivos planteados en el proyecto.

2. *Validación*

Dentro de los primeros pasos del pipeline principal del proceso se tienen definidas las particiones de los datos entre los conjuntos de entrenamiento y prueba.

Durante el proceso de entrenamiento y búsqueda de hiperparámetros se realiza igualmente un proceso de validación denominado backtesting. El proceso de backtesting consiste en evaluar el comportamiento de un modelo predictivo al aplicarlo de forma retrospectiva sobre datos históricos. Por lo tanto, es una estrategia de validación que permite observar la capacidad predictiva del modelo.

Existen varias maneras de realizar backtesting al trabajar con series de tiempo, para el proyecto se utilizó el tipo de backtesting más conocido como time series cross-validation, donde el conjunto de entrenamiento se va moviendo a través del tiempo para ir abarcando todos los datos.

Por otro lado, el conjunto de prueba se utiliza solo en el momento de medir el desempeño del modelo. En el proceso de entrenamiento estos datos no se tienen en cuenta para la extracción de características o al generar ventanas de autorregresión. Separando de esta forma los conjuntos de datos se evita que se filtre información del conjunto de prueba al proceso de entrenamiento y así se puede evitar tener métricas erradas.

3. *Iteraciones y evolución*

Se realizaron ejecuciones similares a la que se hizo con el modelo baseline utilizando otros algoritmos de regresión, de estas ejecuciones se obtuvieron resultados muy similares en todos los modelos probados.

Posteriormente considerando modelos autorregresivos, se hizo un proceso de varias iteraciones donde se trabajó con los tres diccionarios de características estándar que ofrece la librería TSFRESH para la extracción de variables exógenas:

- **ComprehensiveFCParameters:** Incluye todas las funciones sin parámetros y todas las funciones con parámetros, cada una con diferentes combinaciones de estos.

- **EfficientFCParameters:** Incluye la mayoría de funciones que el diccionario anterior, exceptuando las que están marcadas con un atributo que indica que la característica requiere un alto costo computacional.

- **MinimalFCParameters:** Incluye solo un puñado de funciones y es la que requiere menos costo computacional.

En cada iteración se probó con los diferentes conjuntos de características y modelos, llegando a la conclusión que, aunque los dos primeros grupos de funciones contienen muchas más características y se puede pensar que aportarán mejor información, en realidad para este problema la información que aportan no es muy relevante en comparación con el elevado costo computacional que requieren. Los siguientes experimentos ejecutados fueron realizados con el diccionario "MinimalFCParameters" que calcula para cada ventana de tiempo características de las variables como mínimos, máximos, media, desviación estándar, entre otros.

Ahora en este punto, el trabajo fue dirigido a una búsqueda exhaustiva centrada solo en los modelos. Los modelos utilizados fueron los descritos en el numeral 4.2, aquí se fueron considerando varias opciones tanto para la configuración del modelo autorregresivo como para los hiperparámetros propios de cada regresor. Aunque ninguno de los experimentos a lo largo del trabajo tuvo un desempeño que se considerara bueno, se pudo mejorar la métrica de error en la medida de lo posible llegando a algunas conclusiones interesantes.

Un hallazgo importante que se dio es que la cantidad de días tomados para hacer el entrenamiento no afectaba significativamente los resultados obtenidos para cada modelo, incluso cuando se disminuyeron los días tomados para el entrenamiento y prueba del modelo fue cuando se presentaron de los mejores resultados.

4. Herramientas

Se presentan las principales herramientas y librerías de python utilizadas junto con una breve descripción de las mismas:

Tabla 3. Herramientas utilizadas

Herramienta/Librería	Descripción	Uso
Visual Studio Code	Editor de código fuente	En este se escribieron en python todos los módulos de código que hacen parte del proyecto.
Typer	Librería para la creación de aplicaciones tipo CLI con Python 3.6 p superior.	Se utilizó para desarrollar y trabajar las diferentes etapas del proyecto como una aplicación tipo CLI (Interfaz de línea de comandos).
Scikit-Learn	Librería de Python de aprendizaje automático y modelado estadístico.	Se utilizó para la creación de los modelos de aprendizaje automático, pipeline principal y adaptación de funcionalidades de otras librerías a este proyecto.
Skforecast	Librería de Python para el uso de regresores de scikit-learn como pronosticadores de varios pasos.	Se utilizó para la creación de los modelos autorregresivos para pronóstico.
Tsfresh	Librería de Python para la extracción automática de características de series temporales.	Se utilizó para la generación de las ventanas de tiempo y su extracción de características.
Sktime	Librería de Python para una amplia gama de tareas de aprendizaje automático de series temporales.	Se utilizó únicamente para el cálculo de la métrica de negocio (SMAPE).
FastAPI	Es un framework de alto rendimiento para crear APIs con Python 3.6 o superior	Se utilizó para crear una API básica a través de la cual se pueda consumir el modelo.

VI. RESULTADOS

1. Métricas

En general, las métricas obtenidas en la gran mayoría de iteraciones y pruebas realizadas no mejoraron mucho respecto a lo obtenido en el modelo baseline. Estos valores se movieron en torno a un valor de SMAPE de 140% a 150%. A continuación, se presentan los resultados de los experimentos considerados más relevantes durante el desarrollo del proyecto y que tuvieron un mejor desempeño:

Tabla 4. Resultados de mejores modelos

Modelo	Métricas	
	Coeficiente de correlación de Pearson	SMAPE
K Neigborhs Regressor	0.324	Validación: 124% Test: 130%
Light GBM Regressor	0.110	Validación: 105% Test: 120%
Random Forest Regressor	0.271	Validación: 113% Test: 114%

Tabla 5. Configuración de los modelos

Modelo	Configuración
K Neigborhs Regressor	algorithm: auto leaf_size: 30 metric: manhattan metric_params: null n_jobs: null n_neighbors: 2 p: 2 weights: distance
Light GBM Regressor	boosting_type: dart class_weight: null colsample_bytree: 1.0 importance_type: split learning_rate: 0.1 max_depth: -1 min_child_samples: 20 min_child_weight: 0.001 min_split_gain: 0.0 n_estimators: 125 n_jobs: -1 num_leaves: 31 objective: null random_state: null reg_alpha: 0.0 reg_lambda: 0.0 silent: warn subsample: 1.0 subsample_for_bin: 200000 subsample_freq: 0
Random Forest Regressor	bootstrap: true ccp_alpha: 0.0 criterion: squared_error max_depth: null max_features: 1.0 max_leaf_nodes: null max_samples: null min_impurity_decrease: 0.0 min_samples_leaf: 1 min_samples_split: 2 min_weight_fraction_leaf: 0.0 n_estimators: 50 n_jobs: null oob_score: false random_state: null verbose: 0 warm_start: false

En las siguientes figuras se presenta una comparación entre un segmento de valores de las predicciones realizadas y un segmento de valores reales para cada uno de estos modelos:

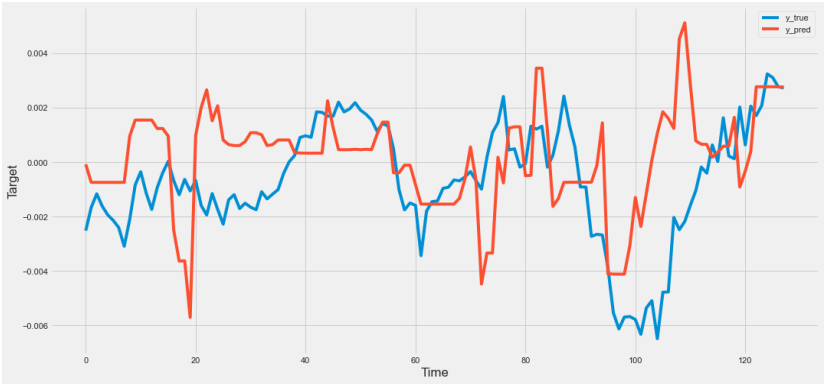

Fig. 14. Predicción vs Valor Real - K Neigborhs Regressor

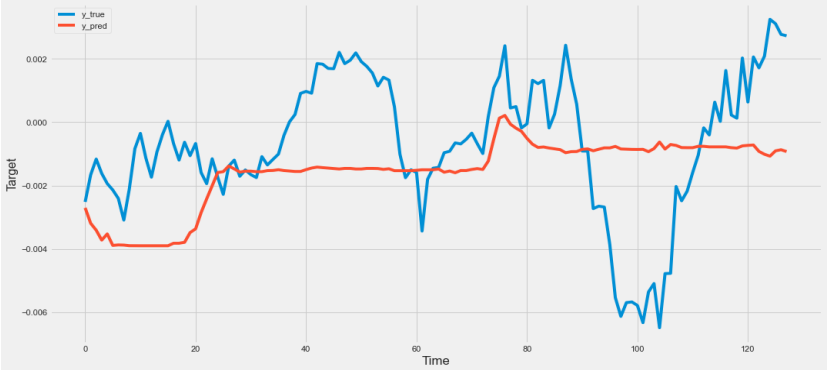

Fig. 15. Predicción vs Valor Real - Light GBM Regressor

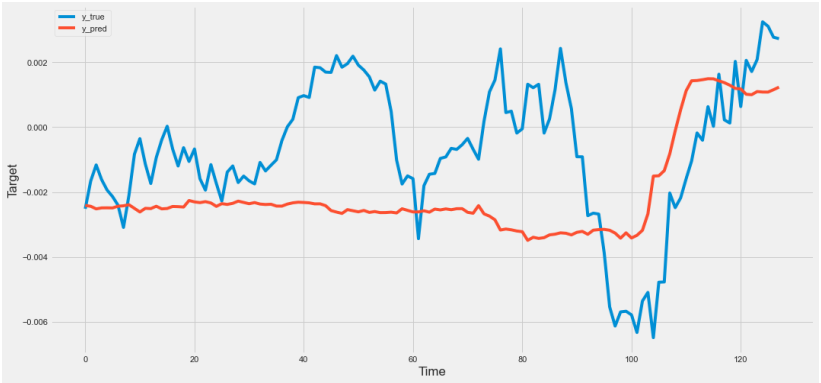

Fig. 16. Predicción vs Valor Real - Random Forest Regressor

2. Evaluación cualitativa

En los resultados de los modelos se observa que no hay casos de overfitting, sus valores de métricas en la etapa de validación son muy cercanos a su desempeño con el conjunto de prueba. Durante el desarrollo se presentaron varios casos de estos casos, donde se solucionaron principalmente ajustando en la estructura del modelo autorregresivo el número de pasos a predecir (entre más pequeño era este hiperparámetro, mayor overfitting se presentaba).

Los resultados obtenidos no son útiles para la finalidad que se propuso desde el inicio del proyecto, incluso los mejores resultados de SMAPE no son menores si quiera al 100% de error, por lo tanto, los valores predichos por los modelos no son confiables para tenerlos en cuenta cuando se quiera realizar una inversión en esta criptomoneda.

3. Consideraciones de producción

Con el framework FastAPI se desarrolló un esquema de API básico para simular el proceso de un posible despliegue y acceder al modelo seleccionado como un servicio. Con el uso de este Framework se dispone de una documentación y una interfaz web interactiva a través de la cual es posible comunicarse con la API y enviar datos al modelo, de acuerdo con el esquema definido en el entrenamiento, para que realice sus predicciones y verificar que funciona correctamente. En las siguientes figuras se puede ver un ejemplo de esta interfaz:

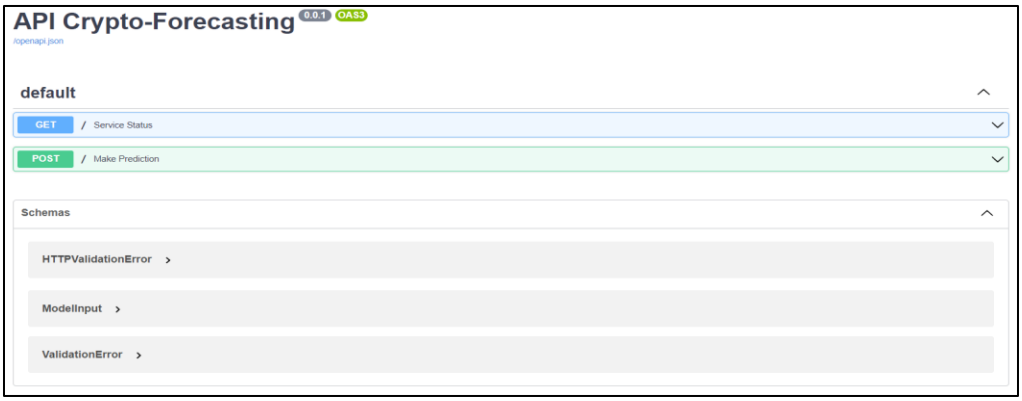

Fig. 17. Interfaz interactiva

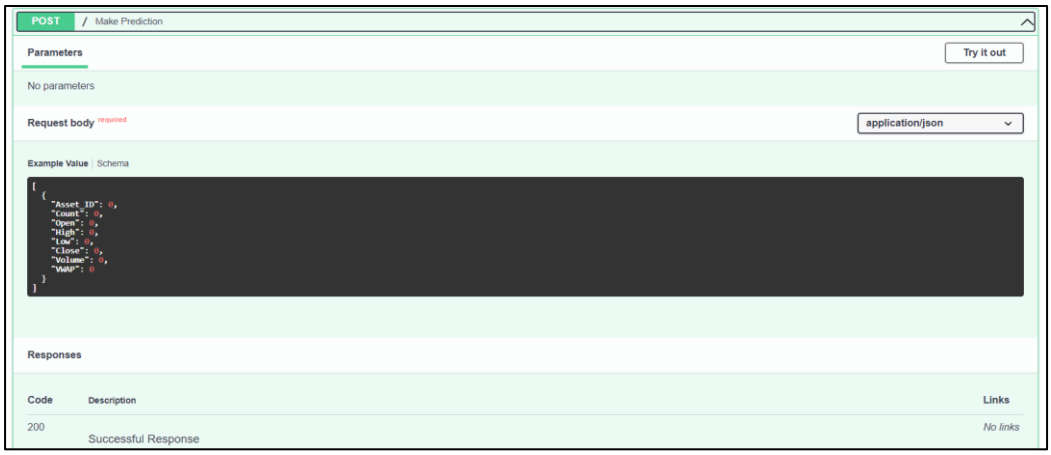

Fig. 18. Ejemplo de esquema para envío de datos.

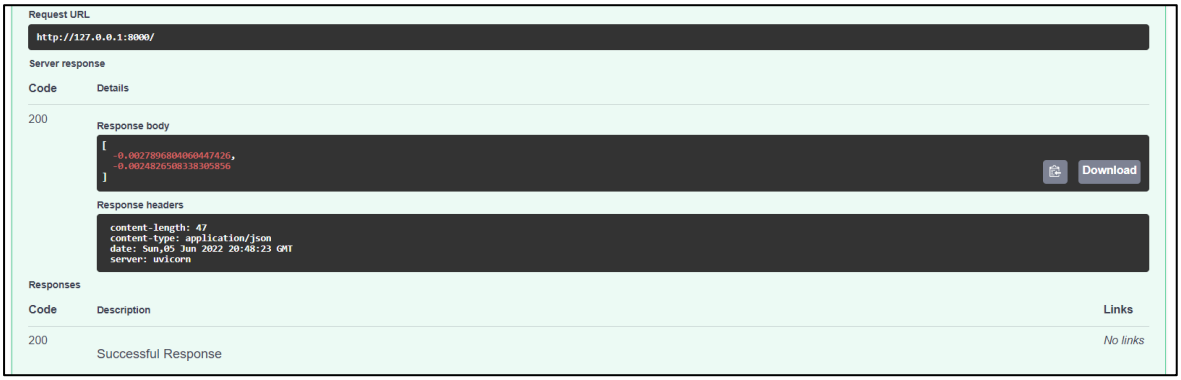

Fig. 19. Ejemplo de predicciones generadas.

VII. CONCLUSIONES

No se obtuvo un desempeño cercano al deseado con ninguno de los modelos desarrollados. Desde el inicio del proyecto se tuvo en cuenta que se podrían presentar resultados como estos debido a la volatilidad del mercado de criptomonedas en general. Partiendo de lo obtenido hasta este punto en la parte del aprendizaje automático, se propone que para un trabajo futuro mejorar el desempeño podría ser enfocado más en el área financiera, donde el estudio a fondo de los mercados y características de estos, que tal vez no se tuvieron completamente en cuenta en el trabajo, podrían ayudar a acercarse más a los resultados deseados.

Para este problema en particular y la forma en que se abordó, aumentar en gran cantidad las características o variables exógenas no está directamente relacionado con obtener un mejor desempeño, solo algunas mostraron real significancia al mejorar las métricas. Un punto importante a tener en cuenta es que al trabajar con la librería TSFRESH para la extracción de características predeterminadas, algunas de ellas pueden presentar errores al calcularlas o valores nulos que pueden interrumpir el proceso de manera inesperada.

No se considera viable llevar el modelo hacia una posible puesta en producción, pues con los resultados obtenidos no es confiable tomarlo como referencia a la hora de realizar inversiones. En cuanto a desempeño de la métrica de negocio (SMAPE), se observa que los algoritmos utilizados basados en métodos de ensamble (RandomForest y LightGBM) presentaron los mejores resultados. Esto puede ser debido a que como su funcionamiento se basa en combinar modelos diferentes, sus errores tienden a compensarse y así mejorar el error en general.

La complejidad en las predicciones del rendimiento de una criptomoneda se basa principalmente en la forma como estas son tratadas, donde el retorno de la inversión proviene de la especulación en torno a las subidas y bajadas de su valor. Debido a este comportamiento, su rendimiento en el mercado en intervalos de tiempo tan pequeños es casi aleatorio y no presenta ninguna estacionalidad, lo que dificulta mucho generar modelos que entreguen predicciones bastante acertadas.

VIII. REFERENCIAS

[1] CoinMarketCap. (2022). Cryptocurrencies. Recuperado el 14 de mayo de 2022 de https://coinmarketcap.com [Precios, gráficos y capitalizaciones de mercado de criptomonedas | CoinMarketCap]

[2] CryptoCompare. (2022). Market Data Reports and Research. Recuperado el 15 de mayo de 2022 de https://data.cryptocompare.com/research [CryptoCompare Market Data Reports and Research]

[3] Cabrera Soto, Marian, & Lage Codorniu, Carlos. (2022). Criptomonedas: ¿qué son y qué pretenden ser?*. Economía y Desarrollo, 166(1), e8. Epub 01 de enero de 2022. Recuperado en 14 de mayo de 2022, de http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S0252-85842022000100008 [Criptomonedas: ¿qué son y qué pretenden ser?]

[4] Banco Central Europeo. (2012). *Virtual Currency Schemes*. Banco Central Europeo. Recuperado el 14 de Mayo de 2022 de https://www.ecb.europa.eu/pub/pdf/other/virtualcurrencyschemesen.pdf

[5] Rink, K. (2021, octubre 21). Time Series Forecast Error Metrics you should know. Towards Data Science. Recuperado en 14 de mayo de 2022, de https://towardsdatascience.com/time-series-forecast-error-metrics-you-should-know-cc88b8c67f27 [Time Series Forecast Error Metrics you should know | Towards Data Science]

[6] Goodwin, Paul, and Richard Lawton. 1999. “On the Asymmetry of the Symmetric MAPE.” International Journal of Forecasting 15(4):405–8. Recuperado el 15 de Mayo de 2022 de: https://doi.org/10.1016/S0169-2070(99)00007-2. [On the asymmetry of the symmetric MAPE - ScienceDirect]

[7] Makridakis, Spyros, and Michèle Hibon. 2000. “The M3-Competition: Results, Conclusions and Implications.” International Journal of Forecasting 16(4). Recuperado el 15 de Mayo de 2022 de: https://doi.org/10.1016/S0169-2070(00)00057-1 [The M3-Competition: results, conclusions and implications - ScienceDirect]

[8] Joaquín Amat Rodrigo. 2021. Skforecast: forecasting series temporales con Python y Scikitlearn J available under Attribution 4.0 International (CC BY 4.0). Recuperado el 15 de Mayo de 2022 de https://www.cienciadedatos.net/py27-forecasting-series-temporales-python-scikitlearn.html

[9] Lewinson, E. (2020, noviembre 1). Choosing the correct error metric: MAPE vs. sMAPE. Towards Data Science. https://towardsdatascience.com/choosing-the-correct-error-metric-mape-vs-smape-5328dec53fac

[10] Lindahl, T., & Sagonas, K. (2005). TypEr: A type annotator of Erlang code. Proceedings of the 2005 ACM SIGPLAN Workshop on Erlang - ERLANG ’05.

[11] Nissi, J., Småros, J., Ylinen, T., & Ala-Risku, T. (2017, noviembre 3). Measuring forecast accuracy: The Complete Guide. RELEX Solutions. https://www.relexsolutions.com/resources/measuring-forecast-accuracy/

[12] Rink, K. (2021, octubre 21). Time Series Forecast Error Metrics you should know. Towards Data Science. https://towardsdatascience.com/time-series-forecast-error-metrics-you-should-know-cc88b8c67f27

[13] Rodrigo, J. A. (s/f). Welcome to skforecast - skforecast docs. Github.Io. Recuperado el 7 de junio de 2022, de https://joaquinamatrodrigo.github.io/skforecast/0.4.3/index.html

[14] Varoquaux, G., Buitinck, L., Louppe, G., Grisel, O., Pedregosa, F., & Mueller, A. (2015). Scikit-learn: Machine learning without learning the machinery. GetMobile Mobile Computing and Communications, 19(1), 29–33. https://doi.org/10.1145/2786984.2786995

[15] Welcome to sktime — sktime documentation. (s/f). Sktime.Org. Recuperado el 7 de junio de 2022, de https://www.sktime.org/en/stable/index.html

[16] FastAPI. (s/f). Tiangolo.Com. Recuperado el 7 de junio de 2022, de https://fastapi.tiangolo.com/

[17] Beheshti, N. (2022, marzo 2). Random forest regression. Towards Data Science. https://towardsdatascience.com/random-forest-regression-5f605132d19d

[18] Besbes, A. (2021, julio 11). How to Deploy a Machine Learning Model with FastAPI, Docker and Github Actions. Towards Data Science. https://towardsdatascience.com/how-to-deploy-a-machine-learning-model-with-fastapi-docker-and-github-actions-13374cbd638a

[19] Morales España, G., Mora Flórez, J., & Vargas Torres, H. (2008). Estrategia de regresión basada en el método de los k vecinos más cercanos para la estimación de la distancia de falla en sistemas radiales. Revista Facultad de Ingeniería Universidad de Antioquia, 45, 100–108. http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0120-62302008000300009

[20] Santander. (2021, agosto 12). ¿Qué son las criptomonedas y cómo funcionan? Santander Bank. https://www.santander.com/es/stories/guia-para-saber-que-son-las-criptomonedas

[21] Sarra, D. (2022, enero 27). How to interpret sMAPE just like MAPE. Medium. https://medium.com/@davide.sarra/how-to-interpret-smape-just-like-mape-bf799ba03bdc