Estimación del Coste del Gas en transacciones de Ethereum mediante Deep Learning

Antonio Arias Sánchez
Máster Universitario en Ingeniería de Telecomunicación UOC-URL
TFM Área de Telemática

José López Vicario
Xavier Vilajosana Guillén

Septiembre 2021 - Enero 2022

FICHA DEL TRABAJO FINAL

Título del trabajo:	Estimación del Coste del Gas en transacciones en Ethereum mediante Deep Learning
Nombre del autor:	Antonio Arias Sánchez
Nombre del consultor/a:	José López Vicario
Nombre del PRA:	Xavier Vilajosana Guillén
Fecha de entrega (mm/aaaa):	01/2022
Titulación:	Máster Universitario en Ingeniería de Telecomunicación UOC-URL
Área del Trabajo Final:	Telemática
Idioma del trabajo:	Español
Palabras clave	Deep Learning, Ethereum, Gas Price

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados y conclusiones del trabajo.*

En el presente trabajo se estudia la aplicación de técnicas de Deep Learning a la predicción del precio del Gas en Ethereum.

Después de una breve introducción al campo del Deep Learning y de la Blockchain Ethereum, se revisan algunos trabajos ya existentes con esta temática.

Se plantea a continuación el problema de la predicción del precio a partir de los principales parámetros que pueden encontrarse en la propia Blockchain Ethereum, y que parecen relevantes para el mismo.

Se ha identificado una fuente para obtener datos históricos, y se han creado scripts para la captura y preparación de datasets desde la misma. Se ha realizado un estudio preliminar de los datos así obtenidos.

Se han propuesto varios modelos y Redes Neuronales que se han entrenado a partir de los anteriores. Se han comparado los resultados obtenidos, comprobando las capacidades predictivas de varios de los modelos.

Finalmente, se presentan las conclusiones obtenidas y sugieren posibles áreas de estudio adicionales.

Abstract (in English, 250 words or less):

This work studies the application of Deep Learning techniques to Ethereum blockchain Gas price prediction.

After a brief introduction to Deep Learning and Ethereum Blockchain, a literature review and search of related works is performed.

Then, a Machine Learning scenario is proposed, where the gas price is to be predicted from several Ethereum blockchain parameters that seem relevant.

A data source from where to get historical datasets has been identified, and scripts have been prepared to retrieve and prepare them. A preliminary data analysis is then performed on them.

Subsequently, several models and Artificial Neural Networks are proposed, created and trained with the previous datasets, and their results compared and commented.

Lastly, conclusions, and possible additional study lines, are presented.

Índice

1. Introducción 1
   1.1. Contexto y justificación del Trabajo 1
   1.2. Objetivos del Trabajo 2
   1.3. Enfoque y método seguido 4
   1.4. Planificación del Trabajo 4
   1.5. Breve sumario de productos obtenidos 6
   1.6. Breve descripción de los otros capítulos de la memoria 6
2. Estado del Arte: Deep Learning 8
   2.1. Inteligencia Artificial. Breve repaso histórico 8
   2.2. Machine Learning 9
   2.3. Deep Learning 10
   2.4. Tensor Flow y Keras 14
3. Estado del Arte: Blockchain y Ethereum 16
   3.1. Tecnología de Blockchain. Bitcoin. 16
   3.2. Ethereum 17
   3.3. Transacciones de Ethereum: Coste de Gas y Tiempo de Ejecución 18
   3.4. Problemas de escalabilidad. Transición a Ethereum 2.0 19
4. Estado del Arte: Deep Learning aplicado a Ethereum 21
5. Predicción del precio del Gas en Ethereum 22
   5.1. Planteamiento del Trabajo 22
   5.2. Origen de datos: Ethereum in BigQuery 22
   5.3. Obtención del Dataset 23
   5.4. Análisis Exploratorio de los Datos. Vista anual. 25
   5.5. Análisis Exploratorio de los Datos. Vista de 5 minutos. 31
   5.6. Limpieza y Preparación de Datos. 32
   5.7. Creación de los Datasets de Entrenamiento, Validación y Test. 34
   5.8. Predicción de series temporales mediante Ventana Deslizante 34
   5.9. Establecimiento de Baseline 36
   5.10. Predicción mediante modelo de Regresión Lineal 36
   5.11. Predicción mediante Red Neuronal Densa 38
   5.12. Predicción mediante Redes Neuronales Recurrentes (RNN) 40
   5.13. Comparativa de Rendimientos 41
6. Conclusiones 44
7. Glosario 46
8. Bibliografía 49
9. Anexos 52
   9.1. Scripts Python 52
   9.2. Datasets 52
   9.3. Otros 52

Lista de figuras

Figura 1. Pioneros de la IA: Charles Babbage (1860), Alan Turing (1951) y John McCarthy (1974) 8
Figura 2. Programación Clásica vs. Aprendizaje Automático [6] 9
Figura 3. Deep Learning vs. Machine Learning vs. AI [11] 11
Figura 4. Esquema de una Red Neuronal Artificial (RNA) [12] 11
Figura 5. RNN de una sola neurona a lo largo del tiempo [12] 12
Figura 6. Habilitadores del progreso en Deep Learning [12] 13
Figura 7. Relación entre Keras y TensorFlow. [6] 14
Figura 8. Función de Activación RELU (Rectified Linear Unit) [6] 15
Figura 9. Transacción de Bitcoin, vista como una transición de estados. [20] 16
Figura 10. El proceso de Minado y generación de Bloques en Bitcoin. [20] 16
Figura 11. Cambio de estados asociados a una Transacción o Mensaje de Ethereum. [20] 18
Figura 12. Esquema E-R de Google BigQuery (parcial) [30] 23
Figura 13. Valores mínimo, medios y máximos del Gas, Ene-Nov 2021, resolución horaria. 26
Figura 14. Valores mínimo, medios y máximos del Gas, Ene 2021, resolución 1 minuto 27
Figura 15. Valores mínimo y medio del Gas, y Base Fee. Ene-Nov 2021, resolución horaria. .. 27
Figura 16. Dificultad, tamaño de bloque, velocidad de transacciones y bloques. Ene-Nov 2021, resolución horaria. 28
Figura 17. Gas usado y Límite de gas, por bloque. Ene-Nov 2021, resolución horaria. 29
Figura 18. Ether total transmitido. Ene-Nov 2021, resolución horaria 29
Figura 19. Matriz de Correlación entre las 12 features. Ene-Nov 2021, resolución horaria 30
Figura 20 FFT del Precio Mínimo del Gas. Ene-Nov 2021, resolución horaria 31
Figura 21. Valores mínimo, medios y máximos del Gas, Sep-Nov 2021, resolución 5 minutos. 32
Figura 22. Matriz de Correlación entre las 12 features. Sep-Nov.2021, resolución 5 minutos... 32
Figura 23. Distribución normal de cada feature. Sep-Nov 2021, resolución 5 minutos 33
Figura 24. Distribución normal de cada feature tras eliminar Outliers (> 3·σ). Sep-Nov 2021, resolución 5 minutos 33
Figura 25. Parámetros de una Ventana Deslizante 35
Figura 26.Evolución de las entradas y salidas de un modelo al que se aplica una Ventana Deslizante. 35
Figura 27. Predicciones del Modelo Baseline, basado en la repetición del último valor disponible. 36
Figura 28. Predicciones del Modelo de Regresión Lineal. 37
Figura 29. Entrenamiento del modelo de Regresión Lineal 38
Figura 30. Predicciones de Perceptrón Multicapa 39
Figura 31. Entrenamiento del Perceptrón Multicapa 39
Figura 32. Evolución de las entradas, salidas y estado interno usando una Red Neur. Recurrente (RNN) 40
Figura 33. Predicciones de Red Neuronal Recurrente basada en LSTM 41
Figura 34. Entrenamiento de la Red RNN 41
Figura 35. Comparativa de resultados de los modelos. Ventana de 24 horas para intervalos de 5 minutos 42

1. Introducción

1.1. Contexto y justificación del Trabajo

Las tecnologías de **Deep Learning** y las basadas en **Blockchain** son dos áreas de plena actualidad dentro de las disciplinas TIC. Ambas tecnologías han pasado, en distintos momentos, por las primeras fases del "**Hype Cycle**" [1] que Gartner definió como ciclo de vida de una nueva tecnología, y se encuentran ahora en lo que se denomina el "**Slope of Enlightenment**", en donde, tras pasar por ciclos de euforia y depresión, la tecnología sigue madurando y dando frutos prácticos.

La **Inteligencia Artificial** es una disciplina veterana [2] que, como veremos, ha experimentado igualmente momentos de euforia y de decepción en su trayectoria desde sus inicios. En el presente, vuelve a suscitar fuertes esperanzas tras los crecientes éxitos cosechados por las técnicas de Machine Learning, una vez que la potencia de cómputo disponible ha conseguido elevarse a un nivel suficiente.

Las tecnologías basadas en **Blockchain** son mucho más recientes. Suponen una aplicación de la criptografía para disponer de un registro de eventos a la vez confiable y descentralizado, que resuelve una amplia gama de problemas. La primera, y disruptiva, aplicación de esta idea fue la creación de una moneda virtual descentralizada, el **Bitcoin**, que desató un furor especulativo y espejismos similares a la era .com. Sin embargo, **nuevas aplicaciones** de este paradigma se siguen sucediendo, y prometen revolucionar otros campos además de las finanzas, como los seguros, la logística y las propias técnicas de computación distribuida.

Fruto de nuestro interés en estas dos tecnologías surge **el presente trabajo**, en el cual se tratará **de aplicar las técnicas de Deep Learning** en alguno de los **problemas** prácticos **que aparecen en el uso de cadenas de bloques**.

Dos son las Blockchains públicas más importantes de la actualidad, **Bitcoin** y Ethereum. Bitcoin fue la fundadora de la tecnología y su aplicación principal es la de servir como dinero descentralizado [3]. **Ethereum** apareció más tarde, con unos objetivos más ambiciosos, pues además de permitir el intercambio monetario, permite construir sobre ella nuevas aplicaciones mediante una serie de "Smart Contracts" que pueden ser programados a voluntad [4].

En ambos casos, la **operativa básica de los usuarios** consiste en la **ejecución de transacciones**, que deben quedar incorporadas a la cadena de bloques como prueba de su validez irrefutable. Para que esta incorporación sea efectiva, los usuarios deben incluir un pago en la transacción, destinado a incentivar a los nodos que participan en el mantenimiento de la red. En función de la cantidad ofrecida, y del estado de carga de la red en cada momento, la transacción puede tardar más o menos tiempo en quedar confirmada.

Debido a la popularización de las redes Blockchain, el tráfico en las mismas ha ido aumentando, y ello ha provocado saturación y **aumento tanto del tiempo como de los costes necesarios para la confirmación de transacciones**.

Por tanto, una necesidad inmediata para el uso eficiente de estas redes es la capacidad de **predecir el tiempo y coste de insertar una nueva transacción** en la misma. El **propósito general del presente trabajo** es **aplicar técnicas de Deep Learning a esta predicción**, y comparar los resultados con otras técnicas actualmente utilizadas.

1.2. Objetivos del Trabajo

Los objetivos concretos del Trabajo han sido los siguientes:

Objetivo 1
Descripción
Iniciarse en el campo de la Inteligencia Artificial, específicamente en las áreas de Machine Learning y **Deep Learning.** Se deberá adquirir conocimiento de los principales algoritmos y modelos de Redes Neuronales empleados en Deep Learning. Se deberá aprender a usar las principales herramientas existentes actualmente, concretamente los frameworks TensorFlow y Keras, y el lenguaje de programación Python
Seguimiento
Se ha conseguido una primera toma de contacto y un conocimiento general de los principales conceptos en el campo de Deep Learning, así como capacidad operativa para manipular datos y definir y entrenar Redes Neuronales empleando TensorFlow y Keras sobre el entorno Google Colab. Para ello, se han realizado tutoriales, como **[5]**, y consultado obras de referencia como **[6]**, **[7]** y **[8]**

Objetivo 2
Descripción
Introducirse en el Estado del Arte de la **tecnología** Blockchain. Estudiaremos Bitcoin y **Ethereum** como dos ejemplos relevantes de la misma. Se estudiará en concreto la problemática de la predicción del coste y el tiempo de confirmación de transacciones, un problema práctico que se presenta en las distintas cadenas de bloques, y que tiene impacto en la adopción de las mismas.
Seguimiento
Se ha ampliado el conocimiento personal en estas áreas, especialmente en la estructura de bloques y transacciones de Ethereum, y en la problemática del coste de las mismas. Se han consultado obras de referencia como **[3]** y **[4]**, y material disponible on-line, citado a lo largo del trabajo.

Objetivo 3
Descripción
Planteamiento del Problema de predicción del Coste del Gas en **Ethereum como un caso de uso de una Red Neuronal.** Se planteará la obtención del Coste del Gas como un problema de predicción de una secuencia temporal mediante una red neuronal, en donde se seleccionará un conjunto de parámetros observables de la cadena de bloques como entradas, y el coste como la variable a predecir.

Seguimiento
Se ha definido un Dataset con un conjunto de features como base para el entrenamiento de varios tipos de redes neuronales, aplicando el método de Ventana Deslizante, y obteniendo resultados para cada distintas configuraciones y meta parámetros.

Objetivo 4
Descripción
Obtención de un Dataset adecuado al planteamiento anterior, consistente en un histórico de parámetros de la cadena de bloques, incluyendo entre ellos los costes de transacciones.
Seguimiento
Se ha encontrado en Google BigQuery una buena fuente para el histórico de datos de la Blockchain de Ethereum. Se han preparado scripts para la obtención de Datasets a partir de la misma, extrayendo las features que se han considerado más relevantes, dentro de rangos de fechas y resoluciones temporales configurables. Igualmente, se ha realizado procesamiento y limpieza de estos datos, como elimación de Outliers

Objetivo 5
Descripción
Definición, entrenamiento y test de una Red Neuronal con capacidad predictiva del coste del gas sobre la red Ethereum.
Seguimiento
Se han definido, implementado y entrenado varias modelos básicos y Redes Neuronales, en concreto modelo de base de Referencia, Modelo de Regresión Lineal, Red Neuronal Densa y Red Neuronal Recurrente. Se han entrenado y testado con diferentes configuraciones tanto del modelo como del dataset de entrada, y obtenido y comparado métricas de su eficacia.

Objetivo 6
Descripción
Comparar los resultados **con otros trabajos existentes** que aborden problemas similares, e identificar otros posibles problemas susceptibles de ser analizados mediante Deep Learning.
Seguimiento
Se han identificado, resumido y comparado con el presente, varios otros trabajos académicos con objetivos similares.

Objetivo 7
Descripción
Cumplir con los requisitos académicos para **constituir un adecuado Trabajo Final del Máster Universitario** en Ingeniería de Telecomunicación.
Seguimiento
Se redacta la presente Memoria y Presentación de acuerdo con las indicaciones del Aula y del Tutor. Se presentarán y defenderán delante del Tribunal indicado.

1.3. Enfoque y método seguido

Los **métodos** que se han utilizado para abordar el trabajo han sido:

- **Investigación bibliográfica**, identificando y consultando las principales fuentes para comprender el Estado del Arte.
- **Realización de Cursos** y Tutoriales, para conseguir un adecuado manejo práctico de las herramientas a usar.
- **Obtención de Datasets** con el histórico de datos necesarios.
- Definición, entrenamiento y test de modelos y **Redes Neuronales** con capacidad predictiva sobre los datasets anteriores.
- Los dos puntos anteriores han implicado el **desarrollo de código y scripts**, que se dejan disponibles tal como se indica en el Anexo.

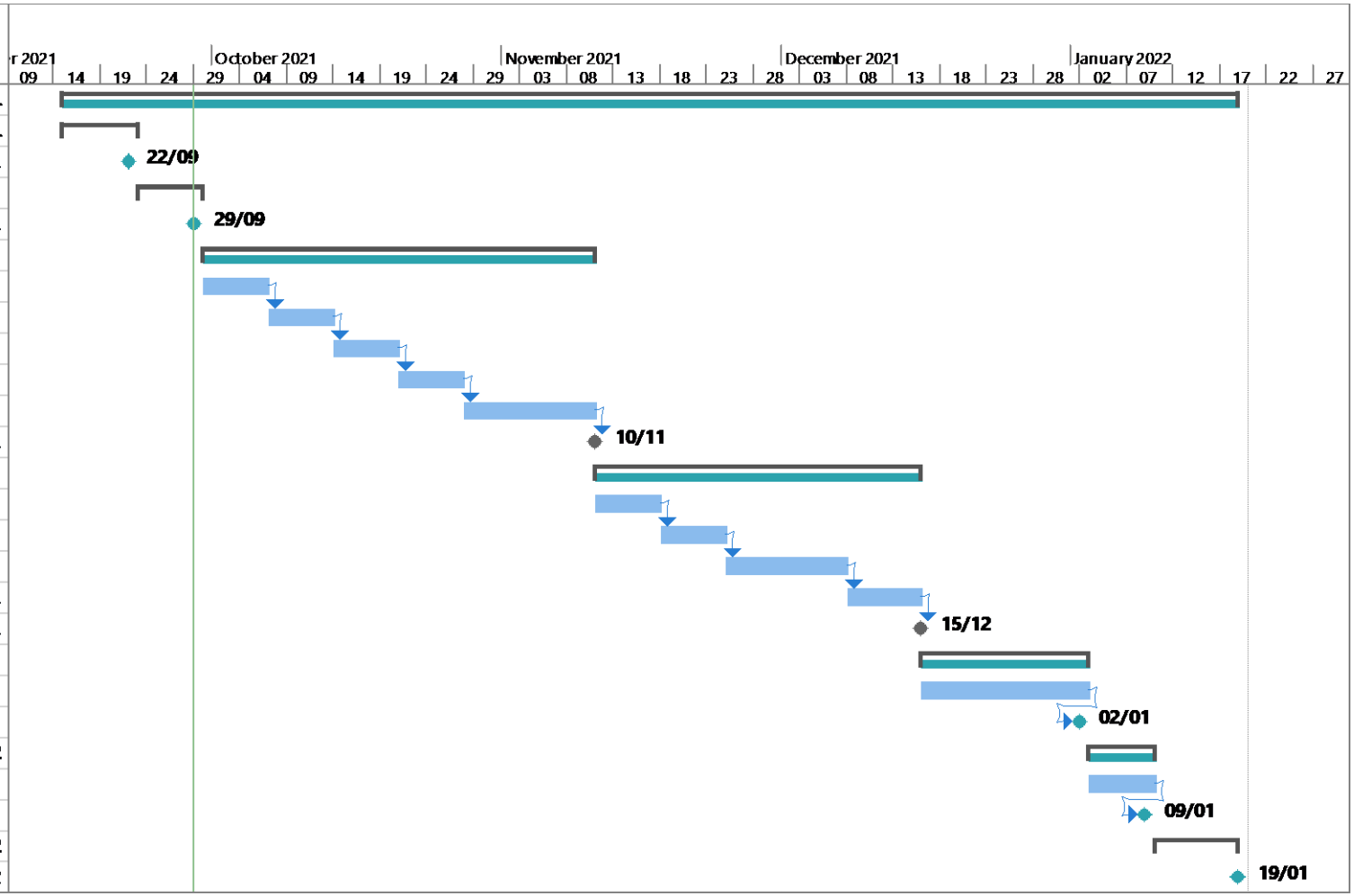
1.4. Planificación del Trabajo

Los principales **Hitos del Trabajo** vienen marcados por la planificación académica, y han sido los siguientes:

Hito	Fecha
Decisión y Definición de la Temática	22.Sep.2021
Planificación del Trabajo	29.Sep.2021
Primera Entrega del Proyecto (60% Alcance)	10.Nov.2021
Segunda Entrega del Proyecto (100% Alcance)	15.Dic .2021
Entrega de la Memoria Final	2.Ene.2022
Entrega de la Presentación	9.Ene.2022
Defensa ante Tribunal	19.Ene.2022

Las **Tareas y Cronograma** seguidos se presentan en el siguiente Gantt:

ID	Nombre de tarea	Start
1	**TFM**	**Wed 15/09/21**
2	**Elección Temática**	**Wed 15/09/21**
3	Comunicación Temática	Wed 22/09/21
4	**PEC1: Planificación**	**Thu 23/09/21**
5	PEC1	Wed 29/09/21
6	**PEC2: Primera Entrega**	**Thu 30/09/21**
7	Revisión Estado Arte	Thu 30/09/21
8	Planteamiento Problema	Thu 07/10/21
9	Generación Dataset	Thu 14/10/21
10	Entrenamiento Red Neuronal	Thu 21/10/21
11	Conclusiones y Memoria	Thu 28/10/21
12	Entrega PEC2	Wed 10/11/21
13	**PEC3: Segunda Entrega**	**Thu 11/11/21**
14	Revisión Bibliografía	Thu 11/11/21
15	Comparación con otras solucione	Thu 18/11/21
16	Replanteo y Alternativas	Thu 25/11/21
17	Conclusiones y Memoria	Wed 08/12/21
18	Entrega PE3	Wed 15/12/21
19	**Memoria**	**Thu 16/12/21**
20	Redacción Memoria	Thu 16/12/21
21	Entrega Memoria	Sun 02/01/22
22	**Presentación**	**Mon 03/01/22**
23	Preparación Presentación	Mon 03/01/22
24	Entrega Presentación	Sun 09/01/22
25	**Defensa Tribunal**	**Mon 10/01/22**
26	Defensa	Wed 19/01/22



1.5. Breve sumario de productos obtenidos

Los principales productos obtenidos tras la ejecución del Proyecto son:

- **Código Python desarrollado para crear los Dataset y definir, entrenar y comparar modelos y Redes neuronales.**

  Se mantendrán disponibles en un repositorio público, tal como se indica en los Anexos.

- **Conjunto de Datasets**, referidos al problema descrito, y que empleados para el entrenamiento y test de los modelos y Redes Neuronales.

  Igualmente se mantendrán disponibles, tal como se indica en los Anexos.

- **Bibliografía consultada.**

  Se incorporará en la presente memoria, y como fichero digital disponible tal como se indica en los Anexos.

- **Memoria y Presentación.**

  Memoria del Proyecto, de acuerdo a lo descrito en el siguiente apartado; y Presentación, como resumen de esta.

1.6. Breve descripción de los otros capítulos de la memoria

Los siguientes capítulos de la memoria, y sus contenidos, son los siguientes:

- **Estado del Arte: Deep Learning.**

  En este apartado, tras un breve repaso histórico de la evolución de la Inteligencia Artificial como disciplina técnica, se presenta un resumen del campo del Deep Learning, y los principales conceptos usados en el presente trabajo.

  Igualmente se presentan las principales herramientas empleados en el mismo, concretamente las librerías Tensor Flow y Keras.

- **Estado del Arte: Blockchain y Ethereum**.

  Del mismo modo, se realiza un breve repaso de estas tecnologías, centrándose en la red Ethereum.

  Se pasa rápidamente a describir el problema objetivo: la predicción del coste óptimo del Gas para las transacciones sobre dicha la red Ethereum.

- **Predicción del precio del Gas en Ethereum**.

  En este apartado se describe el trabajo técnico que realizaremos para conseguir los objetivos propuestos. Los principales aspectos tratados son:

  - Planteamiento del problema de predicción y elección de las técnicas y tipo de redes neuronales más adecuadas.

- Creación, obtención y preparación de los Datasets de trabajo.
- Creación de modelos y redes neuronales y entrenamiento de los mismos.
- Testeo, validación y comparativa de resultados.

- **Conclusiones.**

  Se realiza un análisis de los resultados y revisión del cumplimiento de los objetivos iniciales. Se señalan líneas de trabajo que parecen interesantes a futuro, que no han podido ser exploradas en el proyecto.

  Se realiza un breve estudio de cierre del Proyecto, con las principales lecciones aprendidas, seguimiento de la planificación temporal, del alcance, y de los recursos empleados.

- **Glosario, Bibliografía y Anexos**.

2. Estado del Arte: Deep Learning

2.1. Inteligencia Artificial. Breve repaso histórico

La idea de reproducir la inteligencia humana mediante máquinas es casi tan antigua como la Humanidad, o al menos, existe desde los comienzos del pensamiento filosófico y científico. En un primer momento, fue la **Ingeniería Mecánica** la encargada de explorar las posibilidades de distintos ingenios para simular algunos procesos mentales, fundamentalmente los razonamientos matemáticos o lógicos, como hacía la "Analytical Engine" de Charles Babbage. A partir de los **años 1940s**, con el nacimiento de la electrónica y el ordenador digital, el testigo lo retomó la **Ingeniería Electrónica**. **Alan Turing**, el padre del moderno ordenador y líder del Proyecto Enigma, fue uno de los pioneros del campo. [9]

El nacimiento de la moderna disciplina de la Inteligencia Artificial (IA) se suele fijar en el **verano de 1956, en el Dartmouth College**, cuando un grupo de 10 investigadores se reunieron para un seminario de 6 semanas, con el objetivo de representar el conocimiento humano de manera automática. En ella, fue John McCarthy quien fijó el nombre actual de IA, que se puede definir como *"el esfuerzo de automatizar tareas intelectuales normalmente realizadas por humanos"*

Figura 1. Pioneros de la IA: Charles Babbage (1860), Alan Turing (1951) y John McCarthy (1974)

En los **primeros años**, los esfuerzos se centraron en construir sistemas capaces de resolver **problemas en dominios acotados y bien definidos.** Algunos de sus resultados más conocidos fueron el "Logic Theorist", capaz de demostrar teoremas, "SHRDLU", brazo robótico capaz de ejecutar órdenes sobre un mundo de bloques, o "ELIZA", programa de simulación del diálogo de un psicoterapeuta.

Muchos de estos sistemas empleaban árboles de búsqueda para explorar distintas posibilidades, lo que provocaba dificultades a la hora de escalar a problemas más complicados, por la conocida como "**explosión combinatoria**". Los esfuerzos se encaminaron a encontrar **técnicas heurísticas**, para reducir los campos de búsqueda y hacer abordables problemas insolubles mediante una exploración exhaustiva. En todo caso, a mediados de los años 70 se fueron haciendo evidentes las limitaciones de las técnicas y hardware disponibles hasta entonces, y sobrevino el conocido como primer "invierno de la IA".

A partir de los años 80s, el gobierno de Japón lanzó un Proyecto para los **Ordenadores de 5ª Generación**, una iniciativa de investigación público-privada, dotada de gran financiación, que devolvió el entusiasmo perdido. La principal aportación de esta etapa fueron los **Sistemas Expertos**, que trataban de codificar el conocimiento de un campo concreto por medio de reglas.

Las técnicas empleadas hasta este momento se conocen como "**IA Simbólica**", basadas en reglas codificadas a mano, o bien "**GOFAI**" (Good Old Fashioned Artificial Intelligence). Estos sistemas funcionaron bien en algunos campos limitados, pero eran difíciles de mantener y escalar, supusieron una nueva decepción, y provocaron otro "segundo invierno". [10]

2.2. Machine Learning

A partir de los años 90s se introdujeron nuevas técnicas, como las Redes Neuronales - la llamada "**IA conexionista**"-, o los Algoritmos Genéticos, todas ellas centradas en el paradigma del **"Aprendizaje Automático"** o "Machine Learning" [6], término que ya había sido acuñado en 1.959 por Arthur Samuel. [11]

La manera habitual de emplear ordenadores para conseguir objetivos útiles se basa en la "programación", en donde un equipo de desarrolladores humanos escribe código indicando cómo el ordenador debe ejecutar alguna tarea. El Aprendizaje Automático, en cambio, suministra al ordenador una larga lista de ejemplos, de tal manera que éste sea capaz de inferir las reglas necesarias para conseguir resultados similares. Una comparativa entre ambos métodos se presenta en la siguiente ilustración:

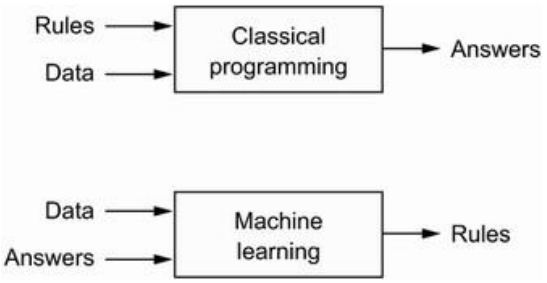


Figura 2. Programación Clásica vs. Aprendizaje Automático [6]

Básicamente, un sistema de Machine Learning necesita los siguientes **elementos**:

- Colección de Elementos de Entrada, o "**Inputs**". Por ejemplo, en reconocimiento de voz, tendríamos muestras de audio.
- Colección de Elementos de Salida deseados, o "**Etiquetas**". En el mismo ejemplo, transcripciones de audio a texto.
- Sistema de **medida**, que determine la calidad de la salida generada respecto a la deseada. Servirá para dirigir el entrenamiento del sistema, de manera que se acerque cada vez más a la salida deseada.

Otra forma de interpretar lo que el sistema hace es buscar reglas de representación y/o transformación para mapear puntos desde el espacio de entradas al de salidas, actuando el sistema de medida como señal de feedback para conseguir cada vez mejores resultados.

Dentro de Machine Learning, se pueden distinguir tres tipos principales de **aproximaciones**, aunque se han ido proponiendo muchas otras variantes:

- **Aprendizaje Supervisado**. Al sistema se le proporcionan entradas y etiquetas, y el objetivo es obtener reglas de transformación para obtener las etiquetas.

- **Aprendizaje No Supervisado.** Al sistema sólo se le proporcionan entradas, y el sistema debe tratar de encontrar patrones o estructuras dentro de los datos.

- **Aprendizaje Reforzado.** Similar al Supervisado, si bien no existen colecciones de datos previas. El sistema es expuesto a un sistema dinámico y cambiante, en donde debe conseguir ciertos objetivos, cuya consecución es medida por una señal de feedback o refuerzo. Un ejemplo es la conducción autónoma.

Todas estas aproximaciones se basan en la construcción de un **Modelo** susceptible de ser entrenado, validado y, finalmente, empleado para la resolución de algún tipo de problema. Estos modelos pueden ser construidos basados en multitud de **técnicas** que se han ido desarrollando desde los años 1950, entre otras:

- **Análisis de Regresión.** Técnicas clásicas en Estadística, que buscan estimar los coeficientes de un conjunto de ecuaciones que relacionan los valores de entrada con los de salida.

- **Árboles de Decisión**. En donde se busca construir un árbol en donde las observaciones quedan representadas en las ramas, y los valores de salida en la hojas.

- **Redes Bayesianas**. Representan un conjunto de variables aleatorias y sus dependencias condicionales a través de un grafo acíclico dirigido.

- **Algoritmos Genéticos**. Técnica surgida en los años 70 e inspirada en la evolución biológica. Estos algoritmos hacen evolucionar una población de individuos sometiéndola a acciones aleatorias, así como también a una selección de acuerdo con algún criterio, en función del cual se decide cuáles son los individuos más adaptados, que sobreviven, y cuáles los menos aptos, que son descartados. Se han aplicado con éxito a multitud de problemas no solubles analíticamente. [12]

- **Redes Neuronales Artificiales.** (RNAs, ó Artificial Neural Networks, ANNs**)** Sistemas inspirados de nuevo en la biología, aparecidos inicialmente al tratar de modelar el supuesto funcionamiento de los cerebros biológicos. Son la tecnología predominante en el presente. Constituyen la llamada “IA Conexionista”, y forman la base del Deep Learning, que se discute a continuación.

2.3. Deep Learning

Una de las primeras redes neuronales implementadas fue el **Perceptrón**, inventado en 1958. Estaba orientado al reconocimiento de imágenes, y constaba de una única capa de neuronas. Aunque inicialmente parecía un diseño prometedor, pronto se demostró que era incapaz de reconocer bastantes patrones, ni tampoco implementar una simple operación XOR. Esto hizo detener algunos años el avance en ANNs [13]

Más adelante, sin embargo, se demostró que añadiendo varias capas al Perceptrón, su capacidad de predicción aumentaba. Con este **Perceptrón Multicapa** (multilayer perceptrón, MLP) se marca el inicio del Deep Learning, en donde el término ‘Deep’ hace referencia a esta existencia de múltiples capas superpuestas, que pueden alcanzar gran profundidad..

La relación entre esta disciplina, y las anteriores, se puede por tanto representar como en la siguiente figura.

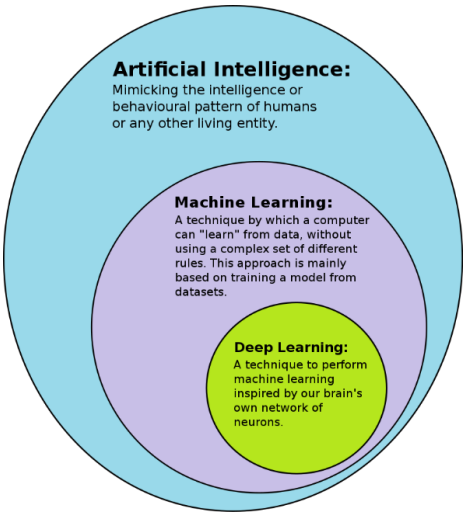


Figura 3. Deep Learning vs. Machine Learning vs. AI [11]

Las **Redes Neuronales Artificiales** (RNAs ó ANNs, Artificial Neural Networks) surgieron como una interpretación aproximada del procesamiento de información que ocurre en los cerebros biológicos. Se componen de varias capas de Neuronas Artificiales. Cada una de ellas tiene varias entradas y una salida. La salida puede dispararse o no (tiene una salida binaria), en función de los valores de las entradas, de un valor umbral configurable, y de la llamada "**función de activación**".

Existen al menos tres capas en una RNA, una **Capa de entrada**, a la que llegan los datos de entrada, una **Capa de salida**, que proporciona los datos de clasificación (labels) o predicción, y una o varias capas intermedias, llamadas **Capas ocultas,** que van transformando progresivamente la información de entrada en la de la salida, tal como se ilustra en la siguiente figura:

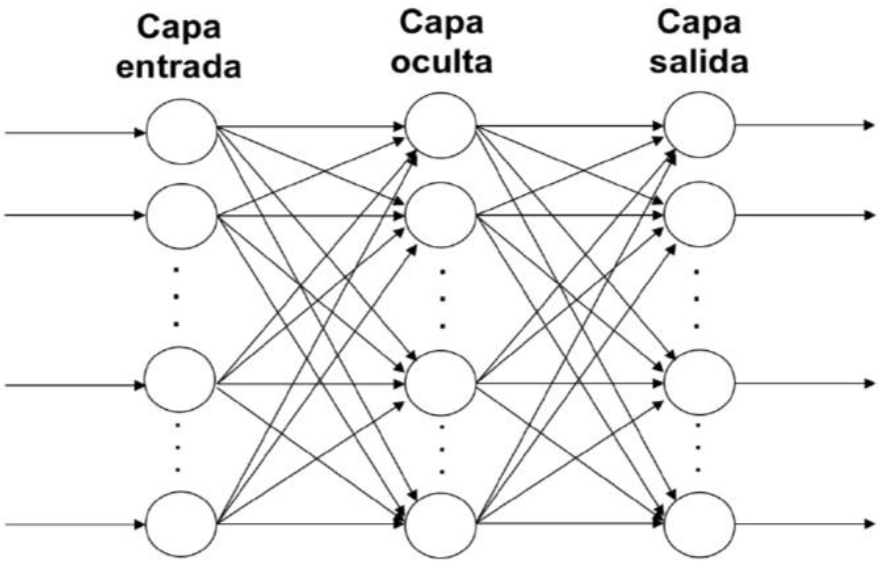


Figura 4. Esquema de una Red Neuronal Artificial (RNA) [12]

Las redes neuronales modernas se pueden componer de un gran número de capas ocultas superpuestas, número relacionado con la profundidad de la red, de donde se origina el término Deep.

Dos elementos importantes en el Deep Learning fueron el desarrollo de los algoritmos de **Backpropagation** y **Stochastic Gradient Descent** [14] [15]**.** Ambos algoritmos son

utilizados para el entrenamiento de las Redes Neuronales, ajustando los valores de disparo de las distintas neuronas en función de los datos de entrenamiento.

En la actualidad, existen un gran número de **tipos de Redes Neuronales**, cada una de las cuales se ha demostrado eficiente para un tipo de problema determinados. Las diferencias entre ellas se basan en el tipo de neuronas, el modo de conexión entre ellas, y las funciones de activación. Aunque la taxonomía es abundante, los tipos principales son:

- **Perceptrón Multicapa** (MLP, Multi-Layered Perceptron), comentado anteriormente. RNA con varias capas conectadas densamente entre sí, por lo que se conoce también como Modelo Denso.

- **Redes Neuronales Convolucionales** (CNN, Convolutional Neural Networks). Contienen varias capas especiales, encargadas de calcular una función convolución. Son muy utilizadas, con excelentes resultados, para problemas como el reconocimiento de imágenes y la visión por ordenador.

- **Redes Neuronales Recurrentes (RNN, Recurrent Neural Networks).** Estas redes emplean neuronas que son capaces de mantener un estado interno, función de los datos de entrada ya vistos por la red, y que se puede incorporar a la predicción de salida, además de los nuevos datos de entrada. Esto hace que estas redes sean especialmente eficaces en el procesado de series temporales, en donde la salida depende tanto de los datos de entrada como del contexto histórico. Serán el tipo de redes que se emplearán en el presente trabajo.

  Existen varias variantes, como las RNN standard, las LSTM y las GRU. [16] Su funcionamiento es como se representa en la siguiente figura.

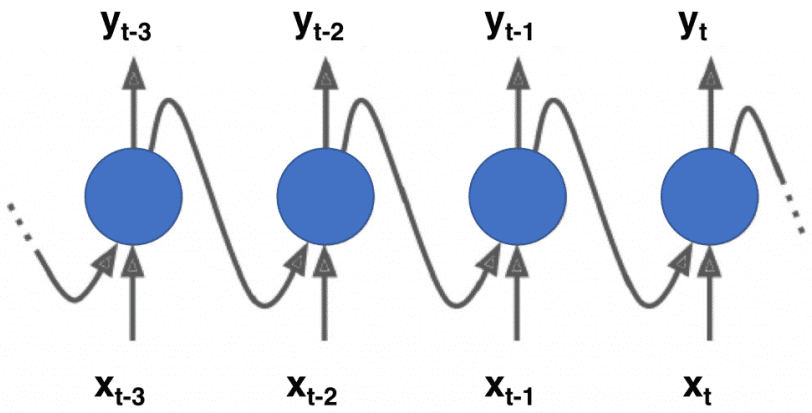


Figura 5. RNN de una sola neurona a lo largo del tiempo [12]

Las técnicas englobadas como **Deep Learning** son en la actualidad las responsables de un nuevo auge de la IA, con resultados espectaculares en varios campos, como son:

- **Reconocimiento de Voz Automático** (ASR, Automated Speech Recognition). Se han conseguido importantes avances en la transcripción de audio a texto, y la construcción de interfaces vía voz.

- **Procesado de Lenguaje Natura**l (NLP, Natural Language Propcessing). En este campo se engloban logros como la traducción automática de cada vez mayor calidad, o la generación automática de textos.

- **Visión por Computador** (CV, Computer Vision). Es posible el reconocimiento facial, la descripción del contenido de fortografías, o la generación de imágenes y vídeo con rostros artificiales y/o basados en el de personajes reales con total realismo.

La mayoría de las técnicas descritas fueron ya desarrolladas a partir de los años 1950, pero los **grandes éxitos del Deep Learning** sólo se han logrado **a partir de la última década**. Son tres los principales vectores que **lo explican**:

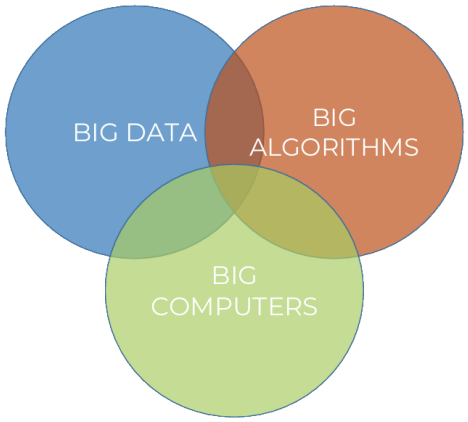


Figura 6. Habilitadores del progreso en Deep Learning [12]

- **Avances en Potencia de cómputo**. La potencia de cómputo necesaria para entrenar con eficiencia una Red Neuronal no ha comenzado a estar disponible a gran escala hasta recientemente. Varios factores han ayudado a ello:
  - En primer lugar, la progresión exponencial en la potencia de los semiconductores, simplemente dejando funcionar la **Ley de Moore**, más la creación de arquitecturas de **computación en paralelo**
  - En segundo lugar, la aparición de **Hardware de propósito específico**. En primer lugar, las tarjetas **GPUs (Graphic Processing Units),** desarrolladas para la síntesis de imágenes, resultaron muy adecuadas para las aplicaciones Deep Learning. En este sentido, se dice que **"la Industria del Videojuego financió los grandes progresos en IA**". En la actualidad, se produce hardware aún más optimizado, son las llamadas **TPUs (Tensor Processing Units**)
  - En tercer lugar, la democratización en el acceso a esta potencia de cómputo, gracias a la **Computación en la Nube**, ya sea para el acceso a infraestructura bajo demanda, bien en las nuevas ofertas de **AI-as-a-Service**.

- **Disponibilidad del Big Data.** Igualmente importante para el progreso del Deep Learning ha sido la disponibilidad de grandes volúmenes de datos, necesarios para el entrenamiento de las redes neuronales a gran escala. Esto ha sido posible gracias al abaratamiento de los medios de almacenamiento, los avances en técnicas de recuperación, como los sistemas No-SQL, y la posibilidad de recopilación y acceso a través de la Internet y el crowd-sourcing.

- **Disponibilidad de Algoritmos y Técnicas.** Finalmente, el desarrollo progresivo de potentes frameworks de trabajo Open Source, y el espíritu colaboración y publicación en abierto de la comunidad investigadora en AI han sido también decisivos para los avances en las últimas décadas.

2.4. Tensor Flow y Keras

Uno de los principales frameworks utilizados en la actualidad en ciencia de datos es el compuesto por **Tensor Flow**, al que se ha incorporado, en sus últimas versiones, la librería **Keras**, y que utilizan el lenguaje **Python**. [7] [5]

TensorFlow es una librería orientada a construir todo tipo de soluciones de Machine Learning, desarrollada por Google, y liberada como código abierto en 2017. **Keras** fue desarrollada como una interfaz sobre TensorFlow y otros frameworks para facilitar la implementación de Redes Neuronales, por F. Chollet, ingeniero de Google [6]. En la actualidad, sólo existe implementación para Tensor Flow, en la que está integrada desde su versión 2.0 de 2019.

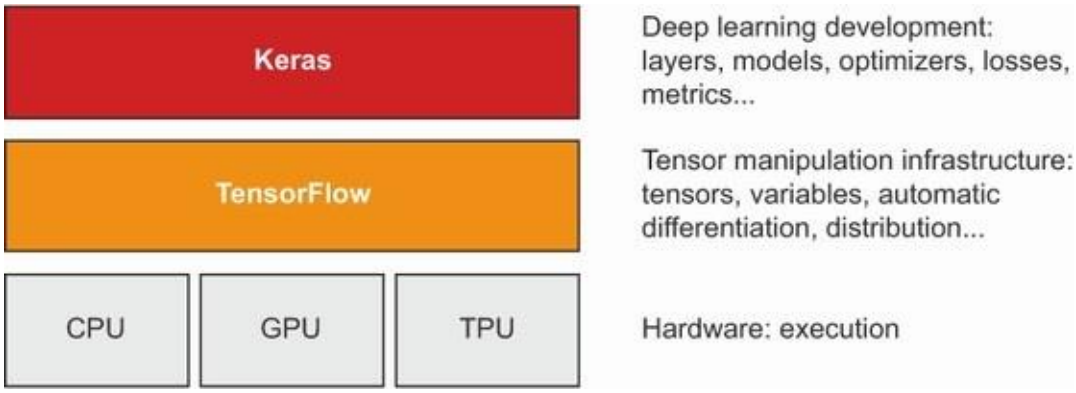


Figura 7. Relación entre Keras y TensorFlow. [6]

El nombre TensorFlow hace referencia a los **Tensores**, estructura de datos que puede verse como una generalización de Vectores y Matrices a n dimensiones, y que son las estructuras de datos básicas para las computaciones de Machine Learning. El paso de los datos por las distintas capas de una Red Neuronal se puede representar como un flujo de datos con estado (**Stateful Data Flow**), de donde viene la segunda parte del término.

TensorFlow admite un gran número de implementaciones, sobre distintos sistemas operativos y arquitecturas, y es capaz de ejecutarse en sistemas distribuidos y compuestos de procesadores, GPUs y TPUs.

Para el desarrollo y la ejecución del código del presente trabajo se ha empleado **Google Collab**, un entorno de desarrollo ya preparado en la nube de Google, y que permite escribir y ejecutar **cuadernos Jupyther** en Python, que permiten integrar texto y código.

Dentro de Keras, la estructura de datos principal es la **clase Sequential**, que permite **crear una RNA básica**. Este nombre hace referencia a que un modelo se considera una secuencia de capas, a través de las cuales fluyen los datos, en forma de tensores, para ir gradualmente transformando la entrada en salida, una vez que la red ha sido entrenada.

Un modelo se construye a base de añadir **capas**, de distintos tipos y características. Algunas capas no tienen estados, pero la mayoría sí, contienen los **pesos,** obtenidos a partir del entrenamiento, que constituyen el conocimiento adquirido por la red.

Para construir una Red Neuronal a partir de la clase Sequential, se deberá en primer lugar **elegir**:

- El **tipo y número de capas** a emplear. El tipo más común de capa es el **tipo Dense**, que representa una capa de neuronas conectadas cada una con todas las entradas y todas las salidas.

- En segundo lugar, el **número de** elementos o **neuronas de cada capa**. El número de unidades escogido se corresponde con el número de pesos a entrenar en cada capa, y representa el número de grados de libertad de la misma. A mayor número, más información se podrá mantener, pero con mayor coste de cómputo y riesgo de **overfitting**.

- En tercer lugar, la **función de activación** escogida para la capa. La función de activación define la salida final de la neurona en función de la entrada y el peso, y normalmente es una función no lineal. Si no fuera por la función de activación, todo el tratamiento de la red neuronal sería una mera transformación lineal, con una capacidad predictiva muy limitada. Son las no linealidades introducidas por las funciones de activación de cada capa lo que explican las capacidades predictivas de las RNAs. [17]

  Una de las funciones de activación más usadas es la **Rectified Linear unit**, o **RELU** [18], que se ha demostrado ser útil en diversos tipos de RNAs, y tiene la siguiente función de transferencia:

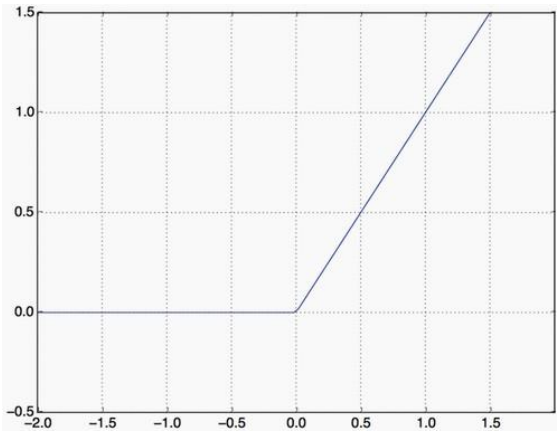


Figura 8. Función de Activación RELU (Rectified Linear Unit) [6]

  Otra función interesante es la **Softmax**, o **Normalized Exponential Function,** que se utiliza en los **problemas de clasificación**. En ellos se emplea en la última capa, para asignar probabilidades a cada una de las posibles salidas, siendo cada una de éstas una de las posibles categorías resultantes, y sumando los valores de todas las salidas 1.

Como ser verá más adelante, una vez construida el modelo de RNA, éste debe a continuación **compilarse** y **entrenarse**, para después poder emplearlo para **realizar predicciones**.

3. Estado del Arte: Blockchain y Ethereum

3.1. Tecnología de Blockchain. Bitcoin.

La tecnología de cadenas de bloques (**Blockchain**) nacen, a partir de un famoso paper publicado por **Satoshi Nakamoto** en 2009 [19]. En él, describe cómo implementar una moneda electrónica, a la que llamó **Bitcoin**, soportada sobre una red peer-to-peer en donde no sea necesario un tercero de confianza. Ésta figura, la de una **autoridad central** o fuente de verdad, siempre había sido necesaria en cualquier sistema financiero, siendo su función la de garantizar la validez de cualquier transacción. [3]

Mediante una aplicación novedosa de la criptografía, Nakamoto consiguió eliminar este elemento de su implementación, determinándose la validez de cualquier transacción mediante el **consenso** de los nodos en una red de pares (**peer-to-peer**).

El **funcionamiento** resumido es como sigue. Los **usuarios** de bitcoin disponen cada uno de una o varias **claves privadas**, de las que se generan un clave pública y una **dirección**. Estas direcciones pueden recibir o enviar bitcoins Para enviar, el remitente debe generar una **transacción**, firmarla con su clave privada, y enviarla a la red, junto con una comisión. Técnicamente, esto puede verse como una transición de estado, donde las distintas direcciones cambian su saldo de bitcoins:

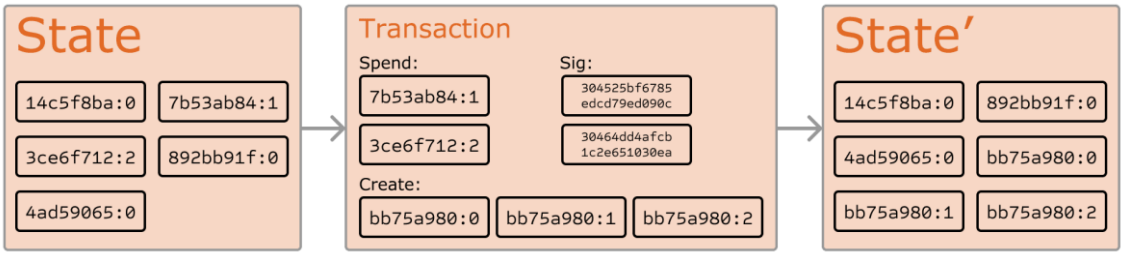


Figura 9. Transacción de Bitcoin, vista como una transición de estados. [20]

Mediante un proceso descentralizado, las distintas transacciones se van agrupando en **bloques**, que se van sumando de forma irrefutable a la cadena de bloques. De este modo, la **cadena de bloques** en su conjunto constituye un **registro indeleble** de todas las transacciones ocurridas, que cualquiera puede verificar de manera independiente.

Para evitar alteraciones interesadas, los bloques se validan criptográficamente de manera pública y descentralizada, mediante un proceso de computación intenso. Este proceso se conoce como **minado**. Para garantizar la neutralidad, se incentiva la participación del mayor número posible de terceros mediante el pago de recompensas.

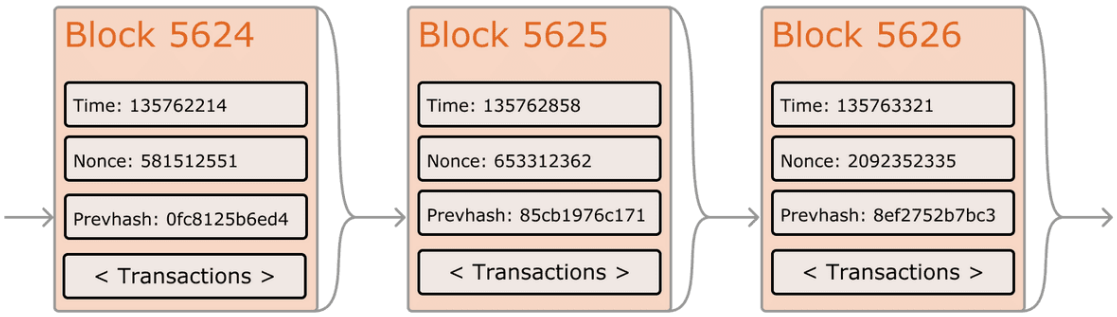


Figura 10. El proceso de Minado y generación de Bloques en Bitcoin. [20]

Esta innovación ha supuesto una auténtica revolución, y el Bitcoin, y las redes Blockchain en general, pronto se han convertido en un fenómeno que algunos han considerado equivalente al nacimiento de la propia Internet o de la WWW. A partir de entonces, han proliferado muchas otras monedas similares, y más importante, nuevas aplicaciones de la idea original de Blockchain. En cualquier caso, el Bitcoin original sigue siendo la más importante de las Blockchains públicas y cryptomonedas en existencia.

3.2. Ethereum

Ethereum surgió a partir de un **paper** publicado en 2013 por **Vitalik Buterin**, uno de los posteriores fundadores del proyecto en 2015 [20]. En el paper, proponía utilizar la tecnología Blockchain, que había demostrado ya su utilidad, para desarrollar una **plataforma** que **incorporase un lenguaje de programación** Turing completo que permitiese incorporar **Decentralized Applications** o **Smart Contracts**. Estos contratos permiten implementar todo tipo de activos digitales, donde cada desarrollador es libre de definir la lógica que deben controlarlo.

Gracias a esta capacidad, Ethereum se ha convertido en una **plataforma sobre la que** se han ido construyendo diversidad de **otros proyectos**. Algunos de ellos son:

- **Tokens ERC-20**. Es un estándar que implementa un API para la definición y manejo de Tokens, y que es ampliamente utilizado para todo tipo de criptomonedas. El API ofrece métodos para el envío y la consulta de saldos de tokens, y simplifica la construcción de nuevas monedas, ya que elimina la necesidad de un nuevo desarrollo para la mayoría de operaciones comunes a todo activo que deba comportarse como una moneda digital.

- **Non-fungible Tokens (NFTs).** Se trata de tokens únicos e indivisibles, cada uno de los cuales puede representar un objeto único, como arte digital, coleccionables, elementos de un juego, etc. Su introducción representó una oleada de especulación no exenta de polémica. En definitiva, cada token almacena simplemente un número de serie, y una URL u otro tipo de puntero que apunta fuera de la cadena de bloques, para terminar de definir su contenido.

- **Decentralized Finance (DeFi).** Un caso de uso que se refiere al uso de herramientas financieras tradicionales implementadas mediante contratos. Así, se han creado depósitos, préstamos, opciones y futuros. Al estar implementados mediante código desplegado en la red, permiten replicar de manera automática el comportamiento de estos objetos jurídicos, pero sin necesidad de instituciones centralizadas que garanticen su comportamiento, como bancos, cámaras de compensación, compañías de seguros, etc.

El **funcionamiento** resumido de Ethereum es similar al descrito para bitcoin, con la diferencia en que, en cada transacción, un usuario puede, o bien transferir Ethereum -como en bitcoin-, o bien crear o interaccionar con un contrato.

Cada usuario de la red se relaciona con esta mediante una **cuenta**, que contiene:
- El **balance de ether** de la cuenta.
- **Código del contrato**, si existe.
- **Espacio de almacenamiento**, inicialmente vacío.
- El **nonce**, un contador de transacciones emitidas, usado para asegurar que cada una se procese una sola vez.

Existen **dos tipos de cuentas**:

- **Cuentas controladas externamente**. Son empleadas por los usuarios del sistema, y están controladas mediante sus claves privadas. No contienen código. Su uso implica la creación y firma de una transacción.
- **Cuentas de contratos**. Están controladas por el código que contienen. Se activan al recibir una transacción o mensaje, que le hará cambiar su almacenamiento interno o enviar otro mensaje a otro contrato.

Una **transacción** de Ethereum consiste en un conjunto firmado de datos enviado desde una cuenta externa a otra cuenta, y que contiene:

- Dirección de la **cuenta receptora**.
- **Firma digital** que identifica a la cuenta emisora.
- **Cantidad de ether** que debe enviarse a la receptora, si es el caso.
- Campo opcional de **datos**, que contiene los parámetros a pasar al contrato, si la cuenta receptora es una de contrato.
- **Parámetros de gas**, que se verá más adelante.

Además de transacciones, existen también **mensajes**, similares a estas, pero que son enviadas desde cuentas de contratos.

El efecto de una transacción o mensaje puede interpretarse como un cambio de estados de los elementos de la Blockchain, como se puede ver en la figura:

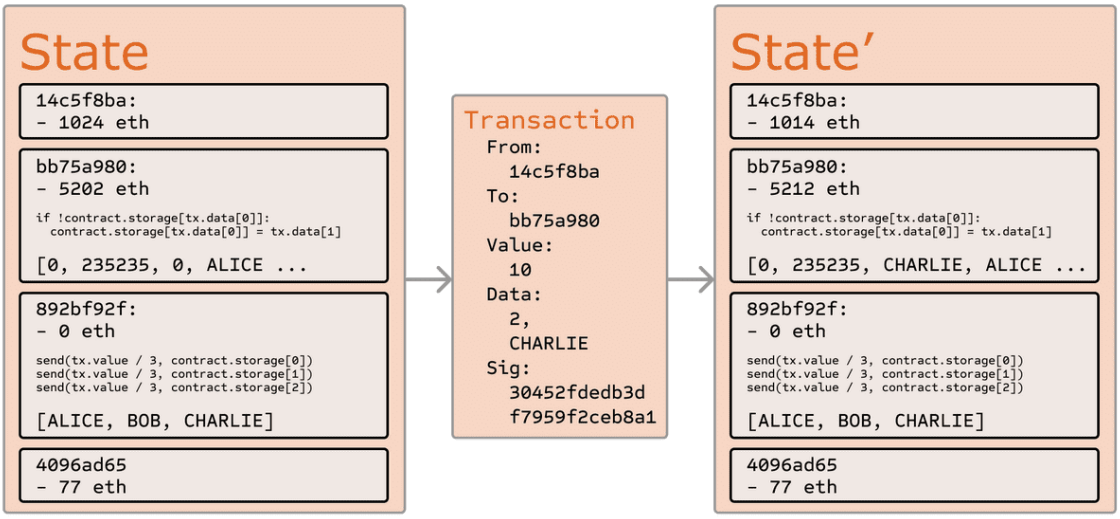


Figura 11. Cambio de estados asociados a una Transacción o Mensaje de Ethereum. [20]

3.3. Transacciones de Ethereum: Coste de Gas y Tiempo de Ejecución

Como hemos visto, sobre la red Ethereum podemos realizar **dos tipos de transacciones**: o bien simplemente **transferir Ether**, como en Bitcoin, **o** bien, **ejecutar un contrato -o crearlo-.** Esto último implica que cada uno de los nodos debe ejecutar el código asociado al contrato para calcular el estado final de la transacción. Dado que es posible ejecutar cualquier algoritmo (la VM de Ethereum es “Turing completa”), ello podría provocar problemas como sobrecarga, explosión exponencial o bucles infinitos, bien intencionadamente o bien por bugs en el contrato. Las consecuencias podrían ser fatales para la red en conjunto.

Para evitarlo, se exige que el emisor de cada transacción pague una "tasa de red" (**network fee**) que cubra el coste computacional, y que además estará siempre acotada. En caso de superar el límite superior, la transacción fallará. [21]

Esta tasa de red tiene varios **objetivos**, por un lado, evitar problemas en la red e incentivar una programación eficiente de los los contratos, y por otra parte, incentivar a los mineros para que contribuyan a mantener la red operativa, ya que las fees son recogidas por los mineros.

Este **mecanismo de fees** funciona del siguiente modo:

- El coste de cada transacción se mide en **Gas,** que se debe pagar mediante Ether, a un precio variable que se calcula como se indicará.

- Al lanzar una transacción, el remitente puede especificar varios parámetros:

    - **Gas Price**: El precio en **WEIs** ($1\ wei = 10^{-9}\ gwei = 10^{-18}\ Ether$) al que el remitente está dispuesto a pagar el gas.
    - **Gas Limit**. El total máximo de unidades de gas que está dispuesto a pagar para la transacción.

- Para que la **transacción sea válida**, el coste real de gas que suponga la ejecución de la transacción debe ser menor o igual al Gas Limit. Cada nodo ejecuta la transacción y va calculando el gas acumulado. Si se supera el límite, se aborta la transacción y se invalida la misma.

- Una transacción donde sólo se ejecute transferencia de Ether entre dos direcciones tiene un coste fijo de 21000 Gas. Para las demás transferencias, en donde se ejecute código de algún contrato, el de la misma coste **se calculará al irse ejecutando**, siguiendo unas reglas precisas que definen el gas gastado en cada operación de bytecode.

- A la hora de decidir qué **transacciones** se incorporarán **en el siguiente bloque**, cada minero ordena las transacciones válidas por orden del precio del gas ofrecido decreciente, y va incorporando desde la mayor hasta la menor hasta alcanzar la capacidad máxima del bloque.

Por lo tanto, cuanto mayor sea el precio del gas ofrecido, antes se incorporará una transacción a la cadena de bloques. Al mismo tiempo, habrá un **precio óptimo del gas**, que será el menor que en cada momento asegure la inclusión en el bloque.

Por lo tanto, **ser capaz de predecir este precio óptimo** tiene un doble beneficio, pues permite que la transacción sea ejecutada con rapidez, y a la vez evita un gasto innecesario.

3.4. Problemas de escalabilidad. Transición a Ethereum 2.0

Un problema importante en la actualidad para el empleo de la red Ethereum, que se extiende a otras criptomonedas como Bitcoin, es la **congestión de la red**. El éxito de algunas aplicaciones alojadas en la red ha hecho evidentes los problemas de escalabilidad de la red; la congestión produce, además de retardos en el procesamiento de las transacciones, un **aumento del precio del gas**, debido a la competición entre los usuarios para hacer que sus transacciones ocupen alguno de los escasos slots disponibles en los bloques. [22]

Esto ha provacado la aparición de las llamadas **redes Blockchain de tercera generación**, que persiguen solucionar estos problemas de escalabilidad, como Cardano, Polkadot, o Ethereum 2.0. [23]

Ethereum 2.0 es un conjunto de propuestas de mejoras sobre Ethereum, que se irán introduciéndo progresivamente sobre la red, con el fin de solucionar estos problemas de escalabilidad, así como aumentar su seguridad y escalabilidad. [24] Entre otras, está previsto el empleo de **Shard Bockchains,** o el paso de **Proof-Of-Work (PoW)** a **Proof-of-Stake (PoS). Shard Blockchains** se refiere a la existencia de múltiples cadenas parciales que funcionan en paralelo, consiguiendo de este modo mayor capacidad.

Proof-of-Work se refiere al actual método para asegurar la validez de los nuevos bloques: los mineros deben invertir bastante trabajo ejecutado desafíos criptográficos para generar un nuevo bloque. Hay bastante controversia debido a la gran cantidad de potencia de cómputo -y desperdicio energético- que este método supone a nivel agregado mundial. Por el contrario, **Proof-of-Stake** elimina los cálculos intensivos, introduciendo un mecanismo que incentiva igualmente la existencia de múltiples participantes externos en el proceso de validación que bloquean una cantidad de fondos importante como garantía de su ejecución imparcial de las transacciones. Su incentivo es recibir compensaciones por su participación, y la posibilidad de perder sus fondos bloqueados en caso de tratar de cometer fraude.

Estas mejoras se están introduciendo progresivamente mediante forks, o cambios en el código en ejecución en los nodos. Así, el día 5 de Agosto de 2021 entró en producción el llamado **“fork London”,** que entre otros cambios, introdujo la EIP-1559. [25].

La **EIP-1559** pretende optimizar el empleo de gas en las transacciones, y simplificar su gestión para los usuarios. También introduce un elemento deflacionario en la red, al destruir (“quemar”) parte del gas gastado, en lugar de destinarlo íntegramente a remunerar a los mineros.

De relevancia para el presente trabajo es que además se introduce un **nuevo mecanismo para el cálculo de coste** en gas de las transacciones. Así, la red establece un precio base en función de la ocupación de la misma en cada momento, y los usuarios, en lugar de indicar un precio del gas, deben indicar una prima sobre éste más un valor máximo absoluto. Se espera de este modo mantener un precio más acotado, y evitar el gasto excesivo en las transacciones.

4. Estado del Arte: Deep Learning aplicado a Ethereum

El **Deep Learning** se ha **aplicado** en numerosos trabajos de **predicción en Ethereum**. Sin embargo, en la gran **mayoría** se intenta predecir las **tendencias del precio** del Ethereum -u otras cripto monedas-, en términos de USD o EUR, de manera similar al de otros activos financieros, como acciones o bonos, como por ejemplo en [26] o en [27]

Sin embargo, son menos los estudios similares al presente, en donde se trate de predecir parámetros que, si bien pueden tener importancia económica, son relativos al propio funcionamiento de la red, como en nuestro caso el coste del gas.

Se relacionan a continuación algunos de los **principales trabajos** encontrados en la literatura:

En [28], se ha desarrollado un modelo para predecir simplemente el **éxito o fracaso de una transacción dada** – considerando éxito si la transacción finalmente queda incorporada a la cadena. Se trata por tanto de un modelo sencillo de clasificación binaria, que, a partir de una serie de features similares a las de nuestro trabajo, determina el éxito de la transacción. Como conclusión, se determina que los valores más relevantes para la predicción son el gas usado, el ofrecido y su precio.

En [29], se propone un modelo para predecir el **tiempo medio en bloques que una transacción** dada tardaría en ser incluida en el Blockchain. Se trata también de un modelo de **clasificación**, con varias categorías en función del número de bloques (1, 2, 3-4 o 5 ó más). Se utilizan varios features, similares a los del presente trabajo, y se comparan los resultados de varios algoritmos de clasificación.

En [30], se propone un modelo para **predecir el precio mínimo del Gas** necesario para incluir una transacción en el siguiente bloque, empleando para ello un **modelo de Regresión basado en Machine Learning**. Se emplean 5 features: difficulty, block gas limit, transaction gas limit, ether price, and miner reward.

En [31], se propone un modelo de Deep Learning, similar al del presente trabajo, para **predecir el precio del Gas**. Se propone además un algoritmo para **recomendar** un precio a emplear, en función de la predicción anterior y el número de bloques máximo que el usuario esté dispuesto a esperar para tener validación de su transacción. El modelo está basado en el uso de Gate Recurrent Units (GRU), un tipo de red neuronal recurrente, que trata de reducir su coste computacional. Se emplean sólo 5 features, entre ellas el precio ETH/USD, y se toman valores en períodos de 5 minutos en 10 días de Noviembre de 2019.

El **presente trabajo** propone una aproximación similar al del último paper, pues se tratará de realizar predicciones del precio mínimo del gas empleando varios tipos de RNAs. A diferencia de éste, se empleará un conjunto más amplio de features, pero sólo relacionadas con el estado de la red (no se utilizará el precio del ETH, por ejemplo, que resulta un valor externo a la red), y experimentará con diferentes conjuntos de meta-parámetros, hasta encontrar aquellos que ofrezcan mejores resultados.

5. Predicción del precio del Gas en Ethereum

5.1. Planteamiento del Trabajo

El **objetivo** del presente trabajo será estudiar las capacidades de **predicción del precio óptimo del gas, aplicando técnicas de Deep Learning**. Como se ha visto, esta predicción puede aplicarse para reducir el coste de las transacciones ejecutadas sobre Ethereum, o ajustarlo en función del tiempo de espera medio deseado.

Se trata en general de un **problema importante para el empleo práctico de** Ethereum – o cualquier otra **Blockchain**-. De hecho, como se ha comentado, la implantación de la EIP-1559 en el fork "London" en Agosto de 2021, persigue ayudar en este ajuste del precio del gas. Para ello, se incorpora dentro de la red un precio de referencia estimado en función del estado observado.

El precio del gas está **determinado**, de manera general, **por la ocupación de red** en cada momento. No existe una métrica directa para esta ocupación, pero sí que hay una serie **parámetros**, de medición más o menos directa, que pueden condicionar, o estar **correlacionados con la misma**. Se tratará por tanto de escoger de entre estos parámetros aquellos que sean más relevantes para ayudar en la predicción.

5.2. Origen de datos: Ethereum in BigQuery

Una de las mejores fuentes de datos para trabajar sobre la cadena de bloques de Ethereum es **Ethereum in BigQuery** [32], ofrecida por Google en su plataforma cloud, en donde diariamente se importan todos los datos de la cadena y se ponen a disposición de los usuarios interesados.

Estos datos se presentan estructurados en una serie de tablas, cuyos contenidos se describen en detalle en la documentación anterior.

De todas las tablas y columnas disponibles, se resumen a continuación todos los elementos que se han considerado más relevantes para la determinación del precio del Gas:

- Tabla **transactions.** Contiene una entrada por cada transacción válida.
  - **gas_price.** El coste pagado por cada transacción. Su valor es lo que interesa predecir – y reducir.
  - **value**. Cantidad de Ethereum transmitido en la transacción. El agregado puede resultar significativo como una medida del tráfico en cada momento; aunque esto sería más significativo en el caso de la red de Bitcoin, pues en la red Ethereum prima la ejecución de contratos y el intercambio de tokens asociados a los mismos.
  - **block_hash.** Clave externa que referencia el bloque en que está incluida la transacción.
- Tabla **blocks**. Contiene una entrada por cada bloque de la cadena.
  - **Difficulty**. Dificultad del bloque, que la red va ajustando en función de la potencia total de los mineros, para conseguir una adecuada cadencia de los

bloques, que en la actualidad está en torno a un bloque cada 12-14 segundos - o bien, entre 250-300 bloques por hora -. [33] [34]

- **Size**. Tamaño del bloque en bytes
- **gas_limit**. Máximo gas permitido en el bloque (fijado por la red)
- **gas_used**. Gas total usado por todas las transacciones del bloque.
- **transaction_count**. Número de transacciones existentes en el bloque.
- **base_fee_per_gas**. De acuerdo con la EIP-1559 tras el fork London, coste base del gas fijado por la red en cada momento. Como veremos, es muy cercano al precio que se paga en cada una de las transacciones del bloque.

En la siguiente figura, puede verse un corte del Esquema Entidad-Relación, tal como se documenta en [35]

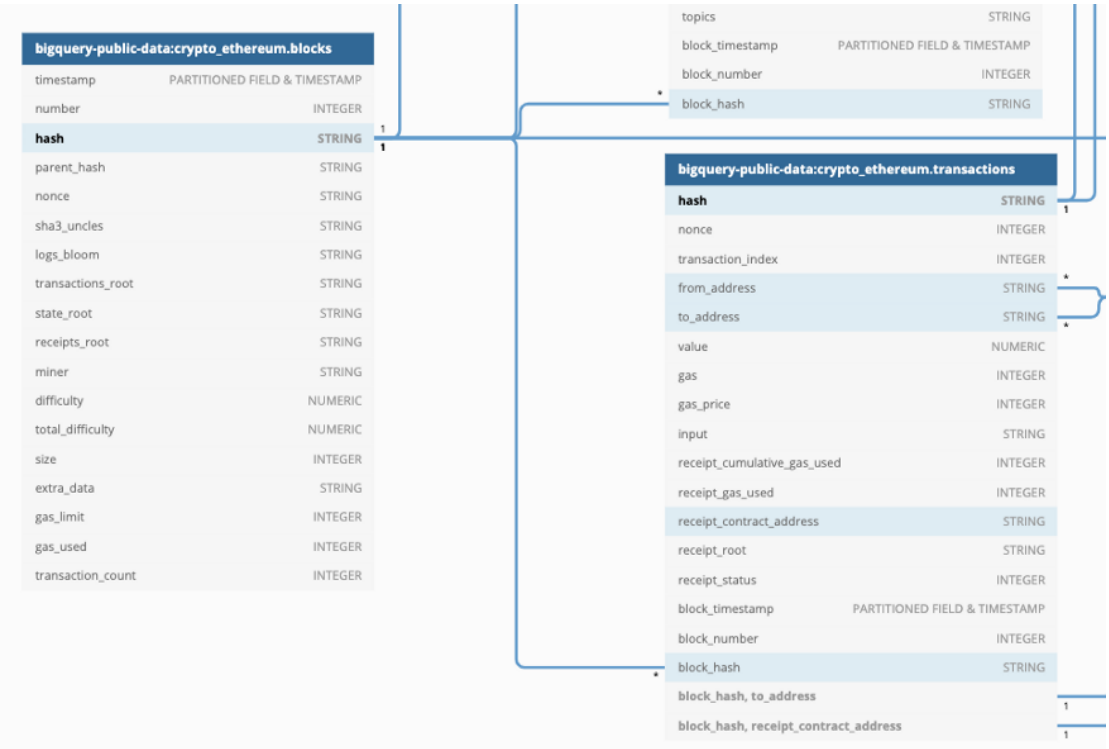


Figura 12. Esquema E-R de Google BigQuery (parcial) [30]

5.3. Obtención del Dataset

Las tablas anteriores contienen la **información de la Blockchain** como una secuencia de **bloques** y **transacciones**.

La red Ethereum está diseñada para generar un bloque cada 12-14 segundos, con mecanismos que aumentan o reducen la dificultad y el tamaño de cada bloque para mantener esta cadencia. Así, cada hora se generan hasta 300 bloques nuevos, cada uno conteniendo entre 100-400 transacciones (en Noviembre de 2021).

Pero, para predecir el coste del gas, resulta más interesante obtener **valores a intervalos regulares de tiempo**.

Se ha preparado por ello **una *query*** que obtiene los valores de cada parámetro –rangos, medios o agregados- por unidad de tiempo y dentro de un intervalo de fechas, todo ello configurable.

```
# Preparar query sobre Ethereum BigQuery
#   anidando busqueda en bloques y transacciones
#   y agrupando por unidades de tiempo
query = """
SELECT

  AVG(base_fee_per_gas/POWER(10,9)) AS avg_base_fee_per_gas,
  AVG(difficulty)                   AS avg_difficulty,
  AVG(transaction_count)            AS avg_transaction_count,
  AVG(gas_limit)                    AS avg_gas_limit,
  AVG(gas_used)                     AS avg_gas_used,
  AVG(size/POWER(10,3))             AS avg_size,
  COUNT (*)                         AS num_blk,
  """+TIME_AGREGATION+"""           AS time,

  AVG(trns.avg_trn_gas_price)       AS avg_gas_price,
  MIN(trns.min_trn_gas_price)       AS min_gas_price,
  MAX(trns.max_trn_gas_price)       AS max_gas_price,
  SUM(trns.sum_trn_tx_ether)        AS sum_tx_ether,
  SUM(trns.num_trn_blk)             AS num_trn,

FROM
  `bigquery-public-data.crypto_ethereum.blocks` AS blocks
   LEFT JOIN
      (SELECT
        block_number,
        AVG(gas_price/POWER(10,9))     AS avg_trn_gas_price,
        MIN(gas_price/POWER(10,9))     AS min_trn_gas_price,
        MAX(gas_price/POWER(10,9))     AS max_trn_gas_price,
        SUM(value/POWER(10,18))        AS sum_trn_tx_ether,
        COUNT (*)                      AS num_trn_blk,
      FROM
        `bigquery-public-data.crypto_ethereum.transactions`
      GROUP BY block_number) AS trns
   ON blocks.number = trns.block_number
GROUP BY time
HAVING DATE(time)>='"""+DATE1+"""' AND
       DATE(time)<= '"""+DATE2+"""'
ORDER BY time """
```

La **query** anterior **funciona del siguiente modo**:

En primer lugar, en el SELECT anidado, se toman todas las **transacciones** que pertenecen al mismo bloque, y de ellas se extraen el valor mínimo, medio y máximo del gas, así como la suma de todo el Ether transferido en las mismas, y el número de transacciones contenidas en el bloque.

Estos valores se añaden a los que se pueden leer directamente de cada **bloque**, en el SELECT externo, en donde se toman todos aquellos que corresponden a cada intervalo de tiempo.

Los **intervalos de tiempo** se pueden definir con ayuda de los operadores de tiempo de BiqQuey [36], por ejemplo:

- Para definir intervalos de una hora:

```
TIME_AGREGATION = 'TIMESTAMP_TRUNC (timestamp, HOUR)'
```

- Para definir intervalos de 5 minutos:

```
TIME_AGREGATION = 'TIMESTAMP_TRUNC(TIMESTAMP_SUB(timestamp, INTE
RVAL MOD(EXTRACT(MINUTE from timestamp), 5) MINUTE) ,MINUTE)'
```

Finalmente, para cada intervalo, se leen los siguientes valores, que constituyen los **'features'** disponibles para los distintos modelos :

Feature	Descripción
avg_gas_price	Media del **precio del gas** para todas las transacciones del intervalo
min_gas_price	Valor mínimo del precio del gas para todas las transacciones del intervalo
max_gas_price	Valor máximo del precio del gas para todas las transacciones del intervalo
avg_base_fee_per_gas	**Valor base del precio del gas,** recomendado por la red, medio de todos los bloques del intervalo.
avg_difficulty	**Dificultad** media de todos los bloques del intervalo.
avg_transaction_count	Número medio de **transacciones por bloque** del intervalo.
avg_gas_limit	Número medio del **límite máximo de gas** permitido **por bloque** para el intervalo. Este valor es el máximo permitido para la suma de todas las transacciones de un bloque - no debe confundirse con el que cada usuario indica en cada transacción iniciada.
avg_gas_used	Número medio del del total de **gas usado por bloque** en el intervalo
avg_size	**Tamaño medio de cada bloque** en el intervalo, en kBytes
sum_tx_ether	Suma total del **ether intercambiado** en todas las transacciones del intervalo
num_trn	Número **total de transacciones** ocurridas en el intervalo de tiempo.
num_blk	Número **total de bloques** generados en el intervalo de tiempo.
time	**Timestamp** de inicio del bloque temporal al que corresponden las 12 features anteriores.

A partir de esta consulta se han creado los Datasets empleados para entrenar y probar distintos modelos, tal como se describirá más adelante.

5.4. Análisis Exploratorio de los Datos. Vista anual.

Se ha realizado en primer lugar un análisis previo de los datos así recogidos.

En primer lugar, para tener una visión general de los datos, se ha realizado una **consulta de largo plazo**, para el **período de 11 meses** desde el 1.Ene.2019 hasta el 30.Nov.2021, **con resolución horaria**, y se ha analizado el comportamiento de los distintos features identificados anteriormente.

En la **¡Error! No se encuentra el origen de la referencia.**Figura 13 se representa, en primer lugar, los **valores mínimos, medios y máximos del gas** para todo el rango:

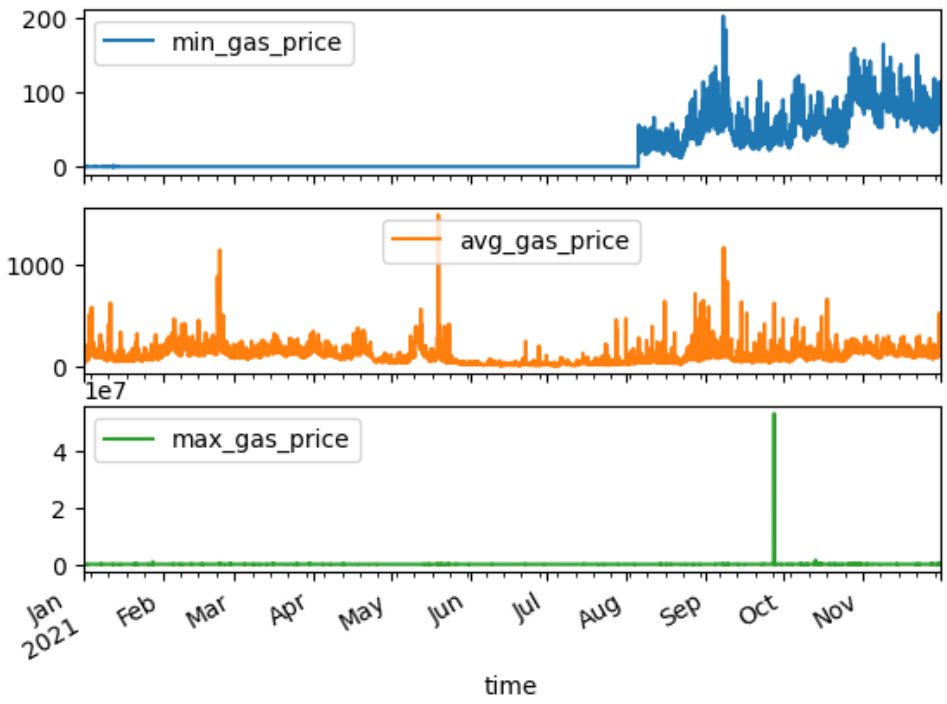


Figura 13. Valores mínimo, medios y máximos del Gas, Ene-Nov 2021, resolución horaria.

Se observa en primer lugar como el valor mínimo, para períodos de 1 hora, resulta ser cero hasta el comienzo del fork London, en Agosto de 2021. Esto quiere decir, que, hasta este momento, durante cada hora, al menos una transacción consigue ejecutarse prácticamente a coste cero. El valora medio, no obstante fluctúa bastante. Y el máximo, presenta un pico muy fuerte en Oct de 2021 – seguramente un 'outlier' debido a algún error en alguna transacción.

Para verificar esto, se ha realizado un **zoom sobre Enero de 2021**, usando **períodos de 1 minuto**. Se comprueba, en la Figura 14, como a esta resolución, el precio mínimo ya no se mantiene en cero, oscila aunque más lentamente que el medio. En el máximo, de nuevo se encuentra un valor de pico mucho mayor que el resto de valores.

Como se ha visto en la Figura 13, el valor del mínimo no alcanza cero a partir de la entrada en funcionamiento del fork London, opuesto que, como se ha explicado, la red ya introduce el 'Base Fee', un valor mínimo para todas las transacciones, que se introduce en la feature llamada 'avg_base_fee_per_gas', como puede observarse en la Figura 15.

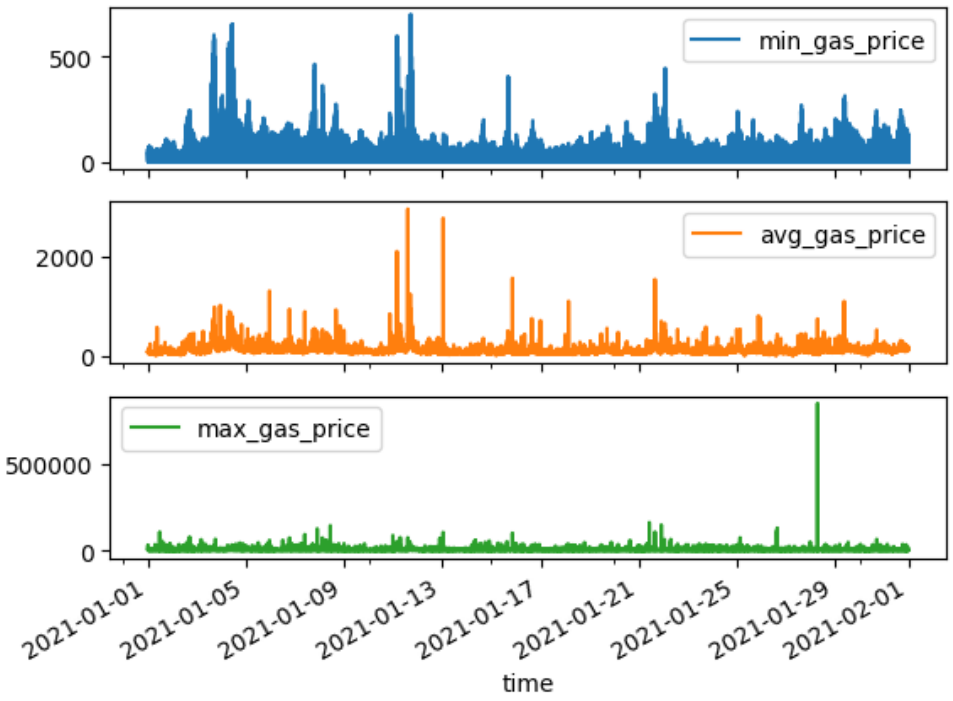


Figura 14. Valores mínimo, medios y máximos del Gas, Ene 2021, resolución 1 minuto

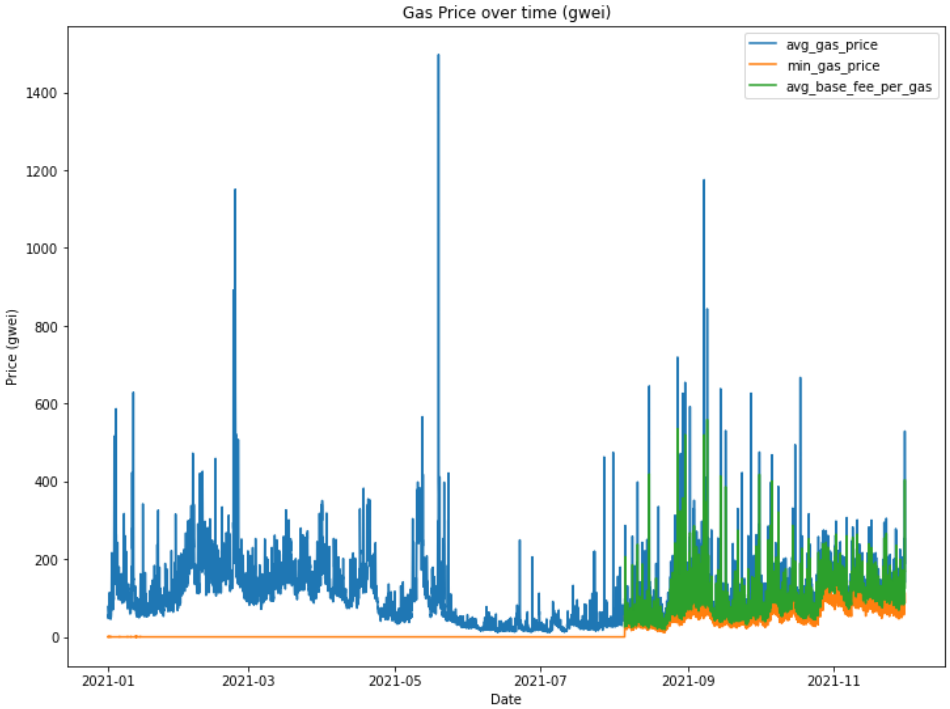


Figura 15. Valores mínimo y medio del Gas, y Base Fee. Ene-Nov 2021, resolución horaria.

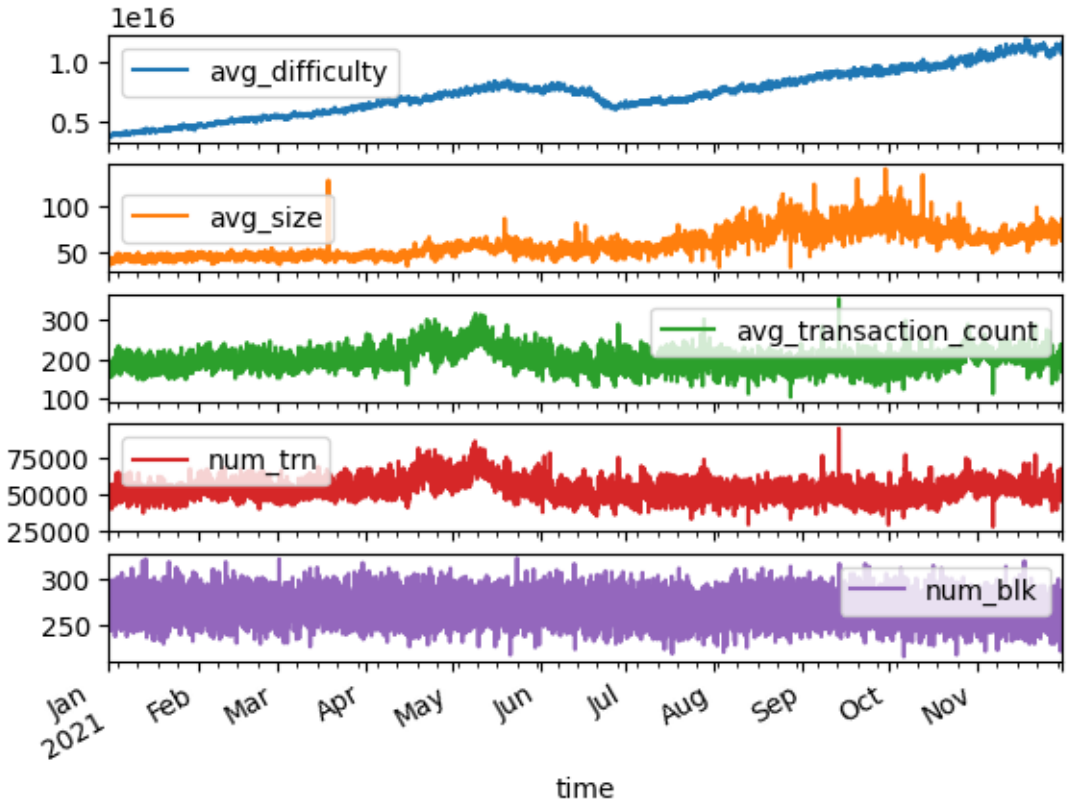


Figura 16. Dificultad, tamaño de bloque, velocidad de transacciones y bloques. Ene-Nov 2021, resolución horaria.

Se representan en la Figura 16 cuatro **features relacionadas con la ocupación de la red:**

En primer lugar, la **dificultad de la red** es un valor que ésta va ajustando para mantener la cadencia de bloques a la velocidad estándar, que debe estar entre 250 y 300 bloques por hora. Como se puede ver en la cuarta feature, **número de bloques**, efectivamente esta velocidad se mantiene dentro de estos valores.

Se aprecia que la dificultad de la red tiene una tendencia ascendente, lo que puede significar que la velocidad a la que más mineros añaden su potencia de cómputo y compiten por las recompensas, es mayor que el crecimiento del tráfico.

Se observa también que el **número medio de transacciones por bloque** y el **número total de transacciones por unidad de tiempo** oscilan de manera similar -tal como es de esperar si el número de bloques por hora se mantiene más o menos uniforme-. No se aprecia un aumento en el número de transacciones por hora en todo el período.

En la Figura 17 se han representado otras dos features que también informan sobre la ocupación de la red. En este caso, el **Gas usado por bloque**, y el **Límite** del mismo, fijado por la red. El gas usado por bloque indica el total de gas de todas las transacciones contenidas en cada uno, que siempre debe estar por debajo del Límite.

Se puede observar cómo el gas usado por bloque ha ido creciendo a lo largo del año, cada vez que los incrementos del límite lo han permitido, hasta estabilizarse a partir del fork London. Ya que, como se ha visto, el número de transacciones se ha mantenido estable, se puede concluir que ha sido el total de **gas medio por transacción** es lo que **ha ido creciendo** a lo largo del año.

A su vez, en la Figura 16 podemos observar cómo el tamaño del bloque en kBytes también ha ido creciendo. Por análogo razonamiento, se puede concluir que el **tamaño medio por transacción ha ido creciendo** igualmente.

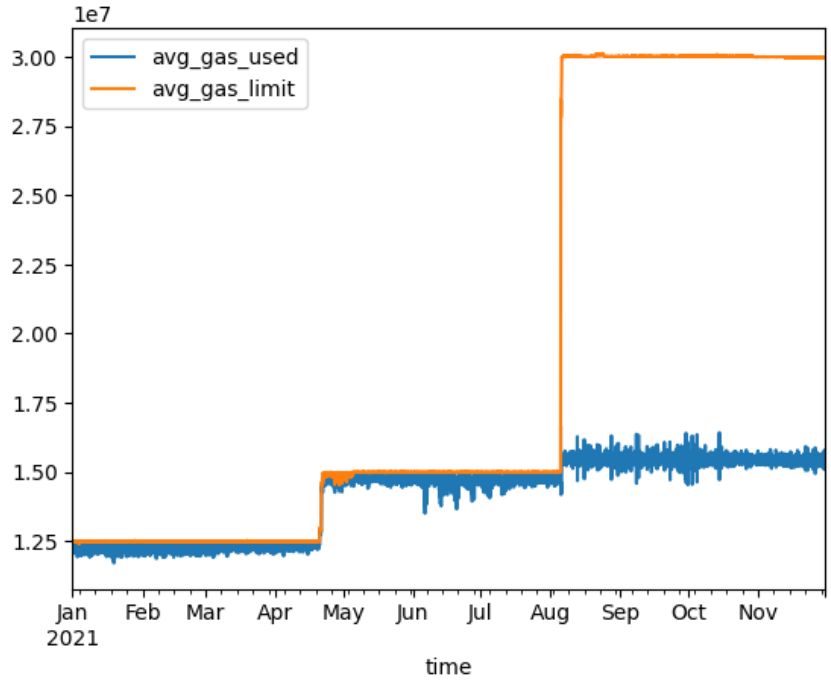


Figura 17. Gas usado y Límite de gas, por bloque. Ene-Nov 2021, resolución horaria.

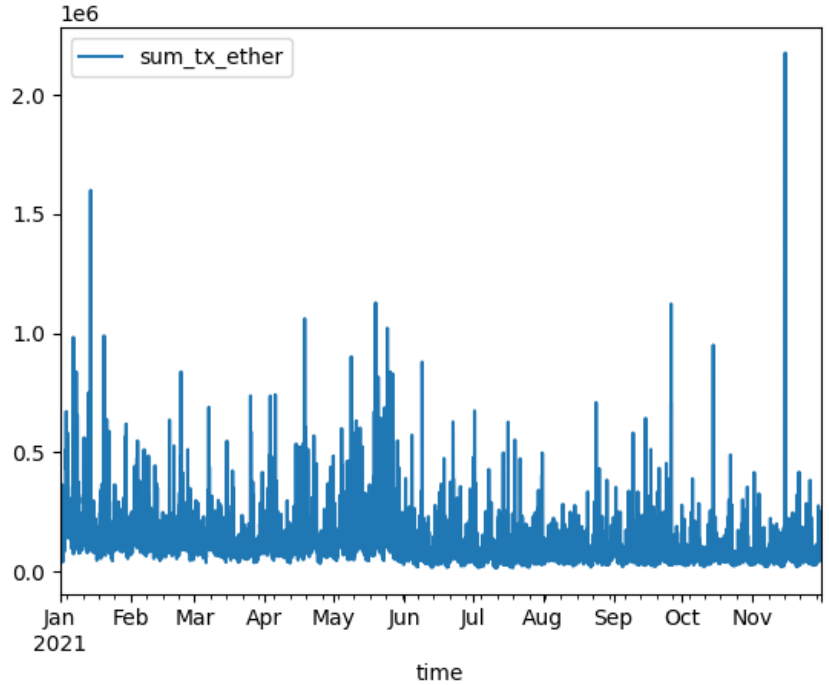


Figura 18. Ether total transmitido. Ene-Nov 2021, resolución horaria

Finalmente, en la Figura 18, tenemos una representación del **Ether total transmitido** por unidad de tiempo. En ella no vemos una tendencia clara. Hay una enorme variación del mismo, sin ninguna tendencia, ni al alza ni a la baja.

Estos últimos resultados, parecen indicar un **aumento de la complejidad de las transacciones** ejecutadas, lo que se corresponde a un uso de la red Ethereum cada vez más orientada hacia la ejecución de contratos frente al mero intercambio de Ether. Contratos que, a su vez, parecen ser cada vez de mayor coste computacional.

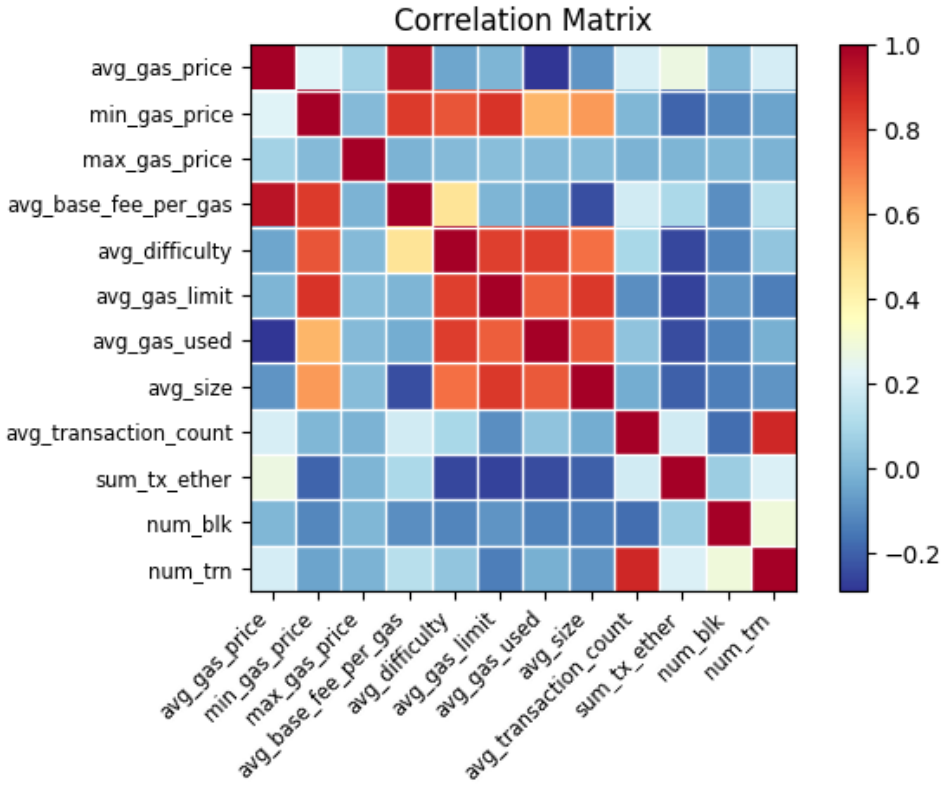


Figura 19. Matriz de Correlación entre las 12 features. Ene-Nov 2021, resolución horaria

En la Figura 19, se ha representado la **Matriz de correlación** entre todas las features obtenidas.

Observamos que la correlación es importante entre la Dificultad, Gas Usado y Límite, y Tamaño, magnitudes relacionadas con la complejidad de las transacciones, como hemos visto.

Existe también entre el Precio del Gas y el Precio Base, lo que indica que este último, introducido tras el fork London, aproxima bien el precio que finalmente se paga por transacción – como se ha visto, supone de hecho una costa inferior.

Para comprobar la presencia de **componentes estacionales** en los datos, en la Figura 20 se ha representado una transformación FFT del precio Mínimo del Gas. [37]

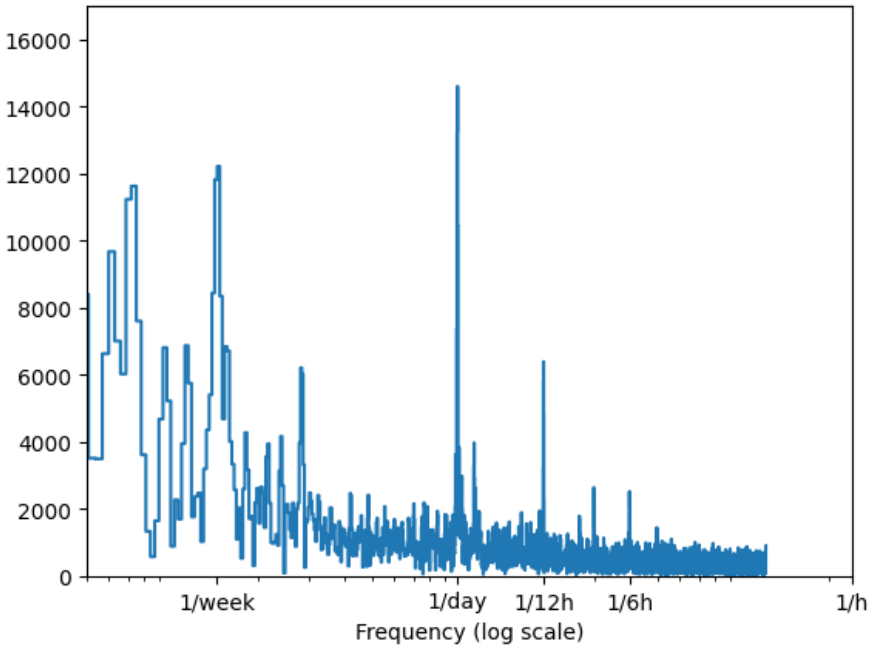


Figura 20 FFT del Precio Mínimo del Gas. Ene-Nov 2021, resolución horaria

Se pueden apreciar factores estacionales con frecuencia diaria y semanal, y algo menor con frecuencias de 12, 8 y 6 horas

5.5. Análisis Exploratorio de los Datos. Vista de 5 minutos.

Se analiza a continuación un **Dataset** que comprende los datos de **tres meses**, entre el 1.Sep.2021 y el 30.Nov.2021, agrupados **con resolución de 5 minutos**. Este Dataset, con mayor resolución temporal, resultará para adecuado para el objetivo de entrenamiento y predicción, en donde se ha establecido el objetivo de predecir con 1 hora de antelación el coste del gas.

Se han repetido los análisis anteriores a este nuevo set, y se señalan los resultados más relevantes.

En la Figura 21, se observa un comportamiento similar en los precios: el **precio máximo** oscila fuertemente y contienen valores 'outliers', que corresponden probablemente a errores. Por ello, se ha decidido eliminar esta feature de las empleadas en la predicción.

En la Figura 22 se ve de nuevo la **Matriz de correlación.** Puede observarse que, a esta resolución, descienden las correlaciones obervadas anteriormente. Puede deducirse por tanto que éstas ocurren para valores medios de la red en períodos más amplios, pero si nos fijamos en el corto plazo, en donde la red no se ha estabilizado, los valores pueden ayudarnos en las predicciones.

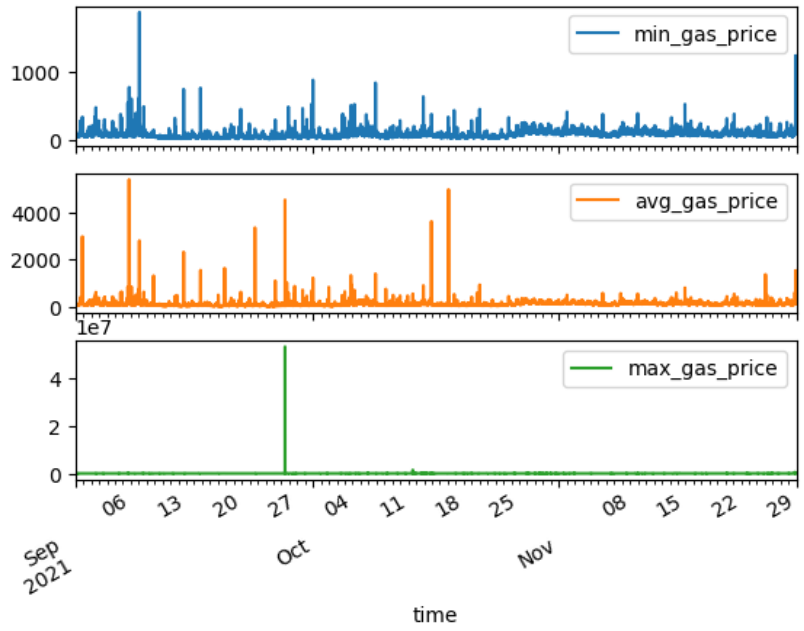


Figura 21. Valores mínimo, medios y máximos del Gas, Sep-Nov 2021, resolución 5 minutos.

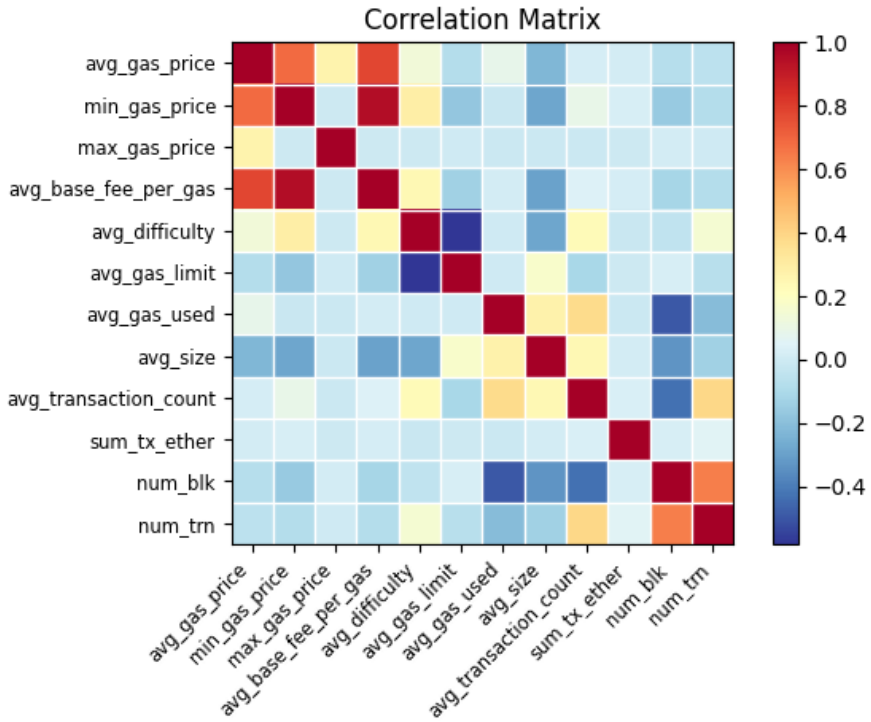


Figura 22. Matriz de Correlación entre las 12 features. Sep-Nov.2021, resolución 5 minutos

5.6. Limpieza y Preparación de Datos.

En la Figura 23 se ha representado una Distribución normal de cada una de las features. Como se puede ver, algunas de ellas presentan una gran dispersión fuera de la normal, especialmente el Precio Máximo del Gas, por las razones ya comentadas.

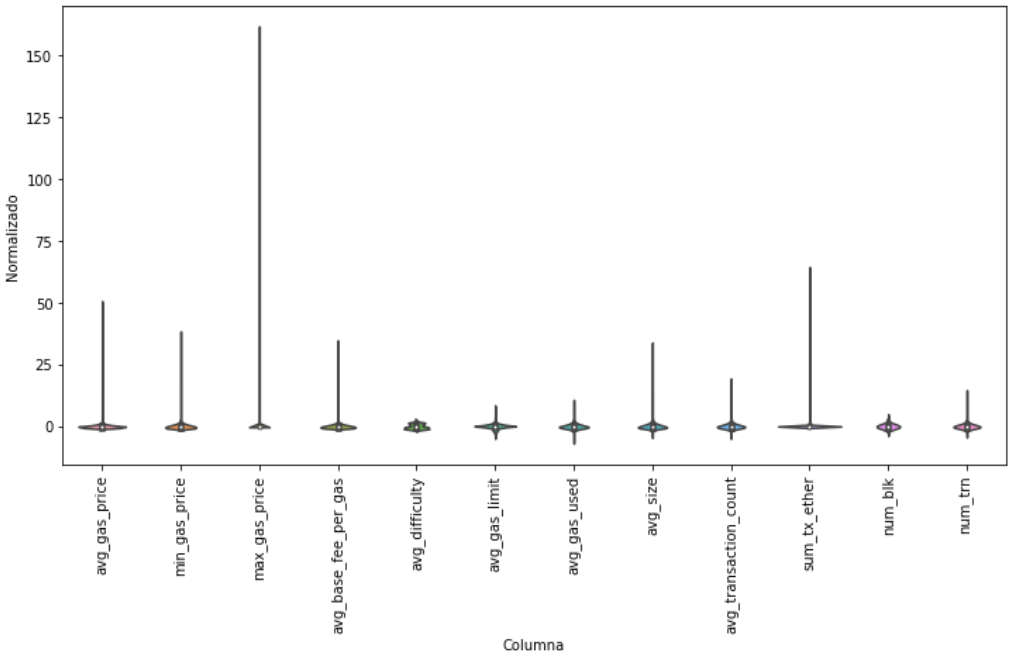


Figura 23. Distribución normal de cada feature. Sep-Nov 2021, resolución 5 minutos

Para que estos datos puedan ser útiles para entrenar un modelo de Deep Learning, es conveniente **limpiar los valores extremos (Outliers).** Así, se ha decidido eliminar todos aquellos valores por fuera de 3 desviaciones típicas hacia arriba o debajo del valor normal. El resultado puede comprobarse en la Figura 24.

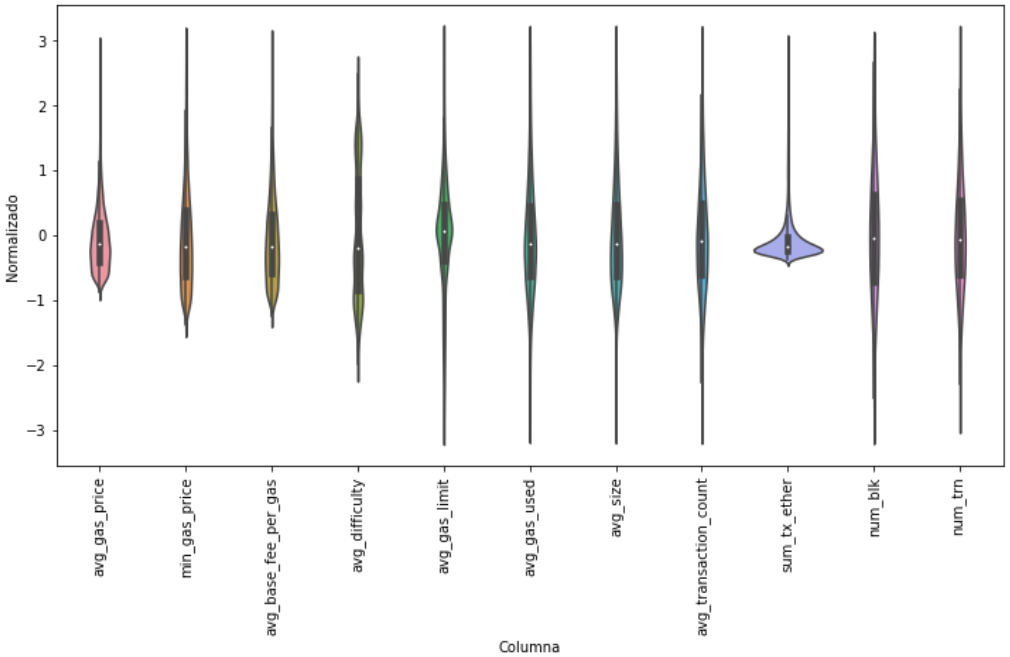


Figura 24. Distribución normal de cada feature tras eliminar Outliers (> 3·σ). Sep-Nov 2021, resolución 5 minutos

Igualmente, se eliminaron todas aquellas filas en donde faltase alguno de los valores de las features. En total, **se han eliminado aproximadamente un 5%** de las muestras.

5.7. Creación de los Datasets de Entrenamiento, Validación y Test.

Para el entrenamiento de RNNs, la práctica habitual es, en primer lugar, **normalizar todas las features**, lo que no afecta a las predicciones pero sí optimiza el procesamiento numérico de los modelos.

En segundo lugar, es necesario **dividir** los datos disponibles **en tres datasets**:

- **Dataset de Entrenamiento**. Serán los datos con que se le alimente la red neuronal durante el proceso de entrenamiento, que considerá, en esencia, en ajustar los parámetros internos de la misma para ajustar las entradas a los valores de salida proporcionados, realizando sucesivas pasadas del algorimo de back-propagation, - denominadas '**epoch'-**

- **Dataset de Validación.** Un excesivo número de epochs puede dar lugar al fenómeno de sobreajuste u **overfitting,** pues la red se adapta excesivamente a los datos concretos y no al patrón general que se desea capturar. Para evitarlo, el proceso de entrenamiento utiliza este segundo conjunto para probar la red sobre él, y dejar de entrnar cuando el resultado para el mismo ya no mejora, o incluso empeora.

- **Dataset de Prueba.** Finalmente, la capacidad de la red así entrenada debe validarse con un tercer subconjunto de datos, que no haya sido visto por la misma durante su entrenamiento.

Para el **caso** de predicción **de series temporales**, es recomendable que las muestras que forman los tres datasets, sean tomadas en el mismo orden en que se han descrito – se desea predecir el comportamiento futuro a partir de lo visto en el pasado, y no al contrario. Una **proporción** razonable **para dividir** el dataset inicial, y que se ha adoptado en el presente trabajo, es un tomar el 70% inicial como datos de Entrenamiento, un 20% como datos de Validación, y el 10% final como datos de Test.

5.8. Predicción de series temporales mediante Ventana Deslizante

Para la predicción de series temporales es habitual emplear la **técnica de ventana deslizante** [38], en donde se van tomando N muestras de todas las entradas para predecir otras 'M' muestras futuras de las 'labels'.

Para ello, se define una **ventana con las siguientes características**:

- Número de muestras de la entrada
- Número de muestras a predecir
- Offset temporal entre ambos rangos de muestras
- Qué 'features' se consideran 'inputs', 'labels', o de ambos tipos.

Por ejemplo, para una secuencia de muestras horarias, en donde queremos predecir un valor dentro de 24 h, a partir de las 24 horas precedentes, tendríamos:

- Número de muestras de la entrada: 24
- Número de muestras a predecir : 1
- Offset temporal entre ambos rangos de muestras: 24

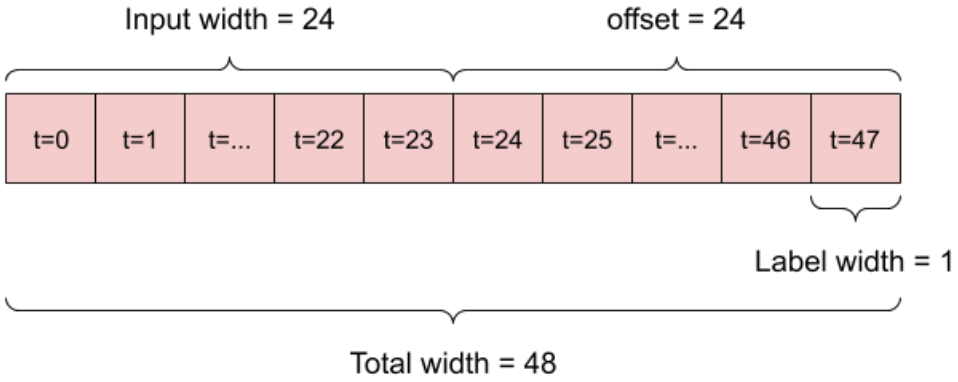


Figura 25. Parámetros de una Ventana Deslizante

Para el presente trabajo, se ha optado por definir inicialmente una **ventana de 24 horas**, con el objetivo de **predecir el valor para al cabo de 1 hora**. Para una resolución de 5 minutos, se deben tomar por tanto 24*12=288 muestras de entrada, para predecir una muestra con un offset de 60.

Como inputs se han tomado todas las 'features', incluyendo el 'min_gas_price', que será también nuestra 'label' (o valor a predecir), tal como se esquematiza a continuación:

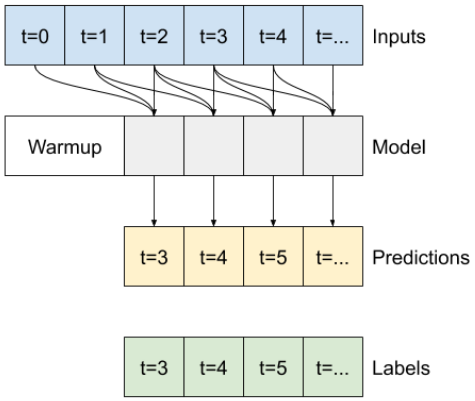


Figura 26.Evolución de las entradas y salidas de un modelo al que se aplica una Ventana Deslizante.

Para facilitar la manipulación del Dataset, se ha reutilizado el código disponible en [38], en donde se define una clase Python, llamada **WindowGenerator**, que facilita:

- Definición de ventanas a partir de los parámetros anteriores.
- Creación eficiente de lotes de entrenamiento, validación y test aplicando la ventana secuencialmente a los datasets iniciales.
- Representación gráfica de ejemplos de la ventana, comparando los valores de test con los generados por un modelo Keras previamente entrenado.

Esta técnica de ventana puede aplicarse para entrenar y validar cualquier modelo Keras que deseemos aplicar a nuestros datos, como veremos a continuación.

5.9. Establecimiento de Baseline

Antes de crear un modelo para predecir, es interesante establecer un **nivel de referencia**, basado en una **predicción sencilla**, a partir de la cual se pueda medir la mejora que los distintos modelos entrenados representen sobre la misma.

Así, dado que el precio del gas es una variable continua, que va evolucionando con el tiempo en función del estado de la red, una predicción básica razonable es simplemente repetir el último valor del gas.

Se ha creado un modelo básico que realiza esta predicción a partir de cada lote entregado. Se representa a continuación ejemplo de sus predicciones, con ayuda de la clase anterior, y para la ventana descrita de 24 horas con offset de 1 hora:

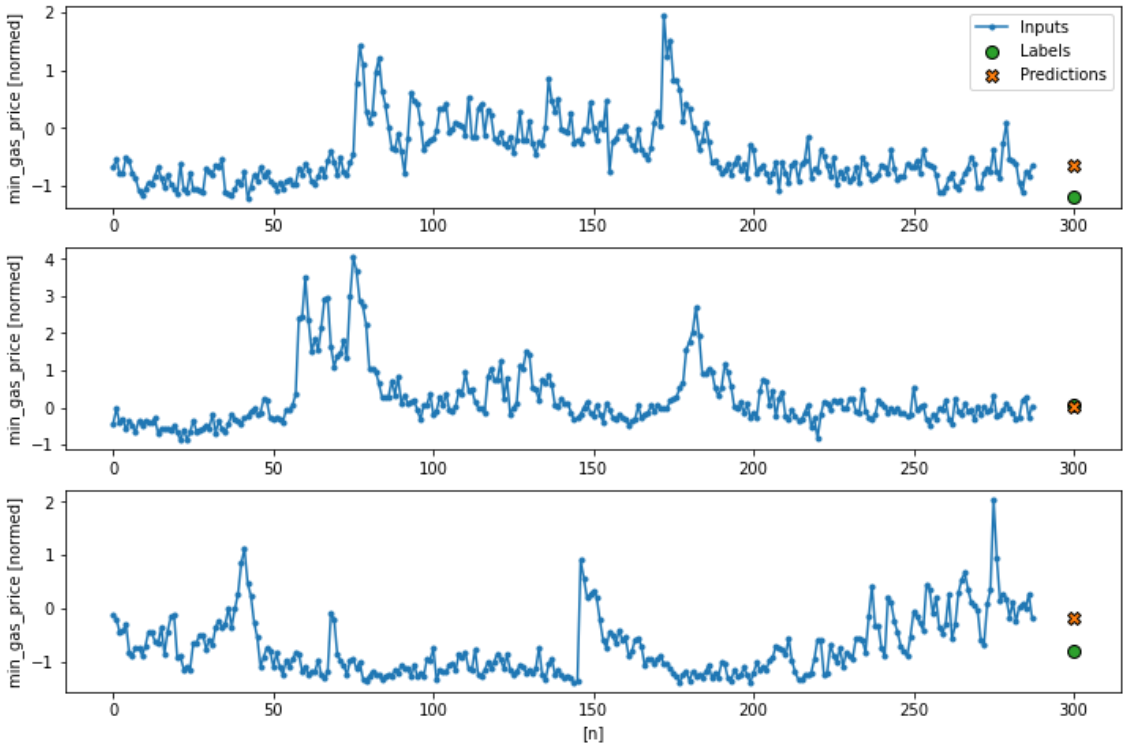


Figura 27. Predicciones del Modelo Baseline, basado en la repetición del último valor disponible.

5.10. Predicción mediante modelo de Regresión Lineal

Un **Modelo de Regresión Lineal** trata de aproximar el valor de una variable independiente (label en términos de Machine Learning) construyendo una expresión lineal sobre un conjunto de variables dependientes (features).

Es uno de los modelos más básicos que se pueden emplear en Machine Learning, y es por tanto interesante estudiar su comportamiento con los datos.

El modelo se puede implementar en Keras mediante una única capa de tipo Dense:

```
multi_linear_model = tf.keras.Sequential([
    # Take the last time-step.
    # Shape [batch, time, features] => [batch, 1, features]
    tf.keras.layers.Lambda(lambda x: x[:, -1:, :]),
```

```
    # Shape => [batch, 1, out_steps*features]

    tf.keras.layers.Dense(OUTPUT_LEN*num_features,
                          kernel_initializer=tf.initializers.zeros()),
    # Shape => [batch, out_steps, features]
    tf.keras.layers.Reshape([OUTPUT_LEN, num_features])
])
```

El código anterior construye el modelo mediante varias capas, que realizan lo siguiente:

- La primera capa **Lambda** simplemente realiza una transformación del Tensor de entrada, tomando la última muestra temporal de cada batch de entrenamiento. El tensor entregado por la clase Windows consta de 3 dimensiones: batch, time, features, ordenados según el convenio utilizado habitualmente en Machine Learning.
- La siguiente capa, de tipo **Dense**, es la única propiamente del modelo, pues la primera y la última sólo realizan transformaciones. Tiene el mismo número de elementos que las features de entrada, y no tiene función de activación. Esto permite implementar una función lineal de las entradas, como corresponde a una Regresión Lineal.
- La última capa simplemente realiza una transformación de dimensiones

Se representa a continuación ejemplo de sus predicciones.

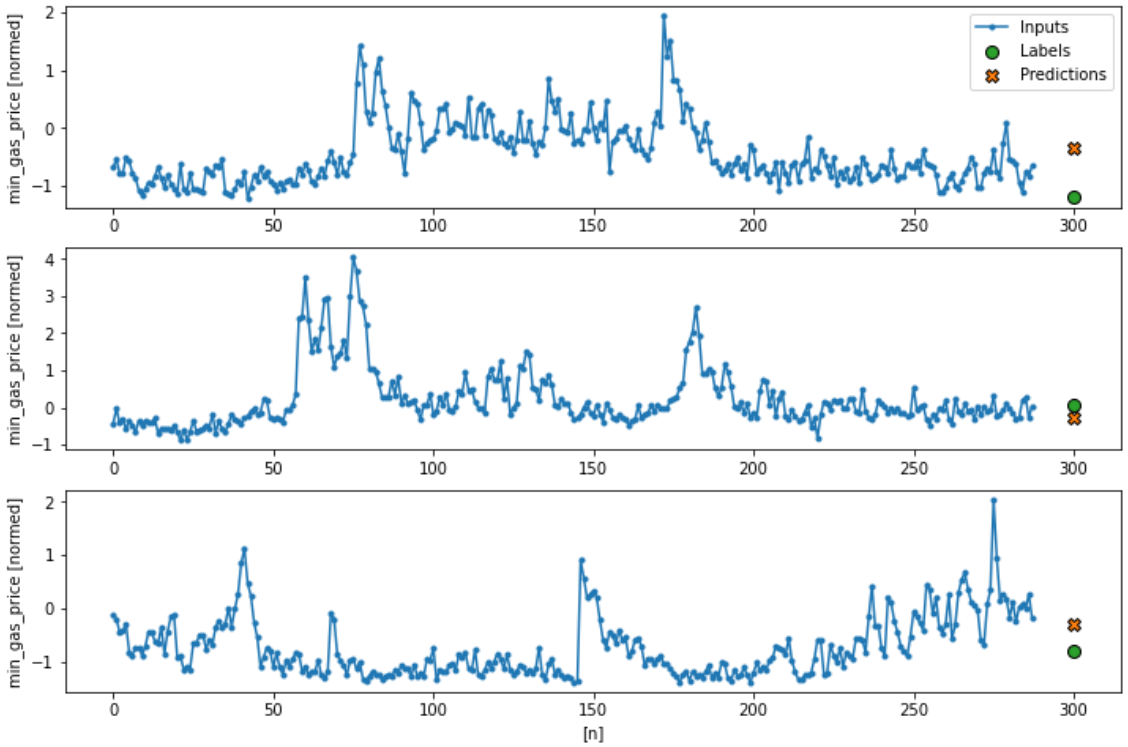


Figura 28. Predicciones del Modelo de Regresión Lineal.

Para entrenar los modelos, se ha utilizado la siguiente función:

```
# Función de compilación y entrenamiento
MAX_EPOCHS = 20
```

```
def compile_and_fit(model, window, patience=4):
  early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                                    patience=patience,
                                                    mode='min')

  model.compile(loss=tf.losses.MeanSquaredError(),
                optimizer=tf.optimizers.Adam(),
                metrics=[tf.metrics.MeanAbsoluteError(),
                         tf.metrics.MeanSquaredError()])

  history = model.fit(window.train, epochs=MAX_EPOCHS,
                      validation_data=window.val,
                      callbacks=[early_stopping])
  return history
```

El método **compile** prepara el modelo; en él se especifica la función de pérdida que debe emplear el algoritmo de Backpropagation, el error medio cuadrático, así como las métricas a utilizar para medir la calidad de la aproximación, donde al anterior se añade el error absoluto.

El método **fit** es el que se encarga del entrenamiento. Se le entregan los datasets de entrenamiento y validación, así como el número máximo de Epochs, y un método 'early stopping' para evitar el over-fitting, que hará que el entrenamiento se detenga cuando detecte un aumento del error de validación entre epochs.

En la gráfica siguiente se puede ver el histórico del entrenamiento del modelo, en donde se representan el error de entrenamiento y de validación. Se puede apreciar como prácticamente tras un solo paso los resultados del modelo ya no mejoran.

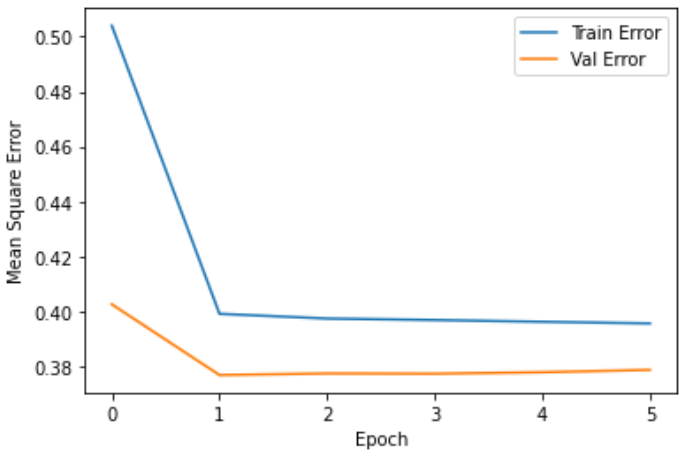


Figura 29. Entrenamiento del modelo de Regresión Lineal

5.11. Predicción mediante Red Neuronal Densa

El modelo más básico de Red Neuronal sería, como se ha visto anteriormente, un Perceptrón Multicapa, con al menos dos capas de tipo Dense:

```
multi_dense_model = tf.keras.Sequential([
    # Take the last time step.
```

```
    # Shape [batch, time, features] => [batch, 1, features]
    tf.keras.layers.Lambda(lambda x: x[:, -1:, :]),
    # Shape => [batch, 1, dense_units]
    tf.keras.layers.Dense(512, activation='relu'),
    # Shape => [batch, out_steps*features]
    tf.keras.layers.Dense(OUTPUT_LEN*num_features,
                          kernel_initializer=tf.initializers.zeros()),
    # Shape => [batch, out_steps, features]
    tf.keras.layers.Reshape([OUTPUT_LEN, num_features])
])
```

El modelo contiene **dos capas densas** – además de la primera y última de adaptación, similares al caso anterior-. La primera capa consta de 512 elementos y función de activación ReLu.

Se representa a continuación ejemplo de sus predicciones.

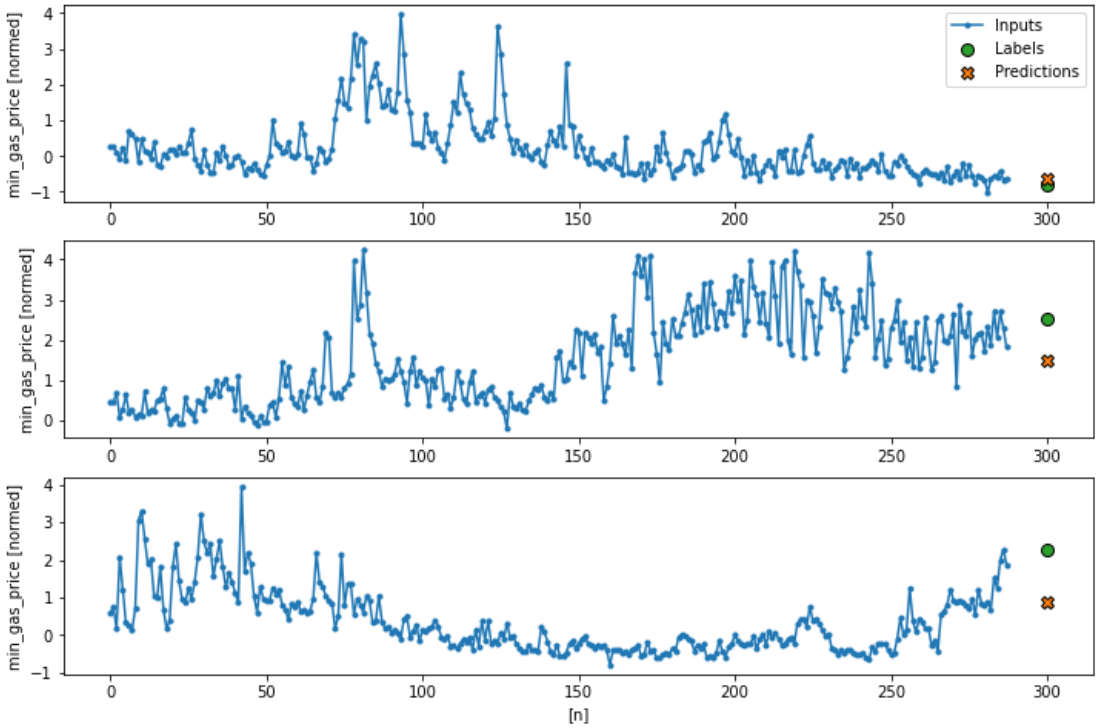


Figura 30. Predicciones de Perceptrón Multicapa

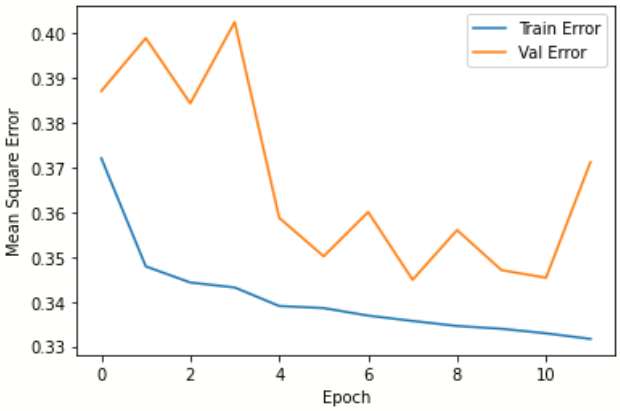


Figura 31. Entrenamiento del Perceptrón Multicapa

En el histórico de entrenamientos se puede apreciar como los errores de entrenamiento y validación continúan descendiendo hasta aproximadamente la iteración 10. A partir de ella, la disminución del error de entrenamiento ya sólo indica overfitting, pues el error de validación comienza a aumentar.

5.12. Predicción mediante Redes Neuronales Recurrentes (RNN)

Para el caso de series temporales, las **Redes Neuronales Recurrentes (RNN)** son especialmente adecuadas, ya que permiten que el modelo, una vez entrenado, mantenga un estado interno en función de la evolución de la entrada.

De este modo, el modelo no se limita a predecir un valor que sería siempre el mismo para una entrada dada, sino que variará con el histórico de los pasos anteriores, ya que puede mantener un contexto no limitado al ancho de la ventana deslizante.

Una manera de representar esta evolución sería así:

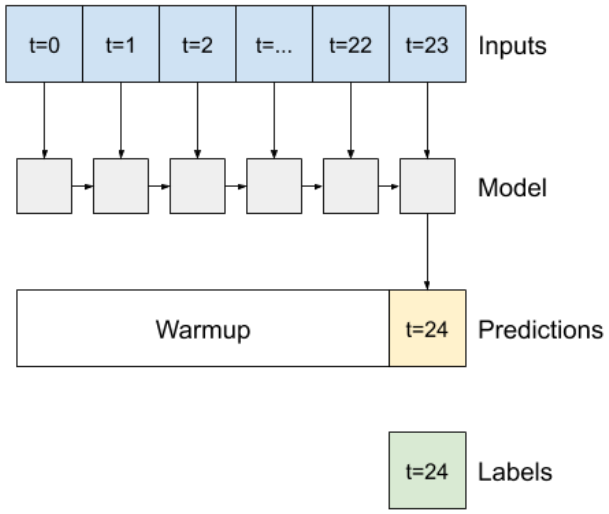


Figura 32. Evolución de las entradas, salidas y estado interno usando una Red Neur. Recurrente (RNN)

Donde, a diferencia de la figura anterior, se pretende representar que el estado interno del modelo depende del estado anterior en t-1.

El modelo se ha construido del siguiente modo, empleando uno de los tipos más comunes de capa RNN, la Long shor-term memory (LSTM).

```
multi_lstm_model = tf.keras.Sequential([
    # Shape [batch, time, features] => [batch, lstm_units].
    tf.keras.layers.LSTM(32, return_sequences=False),
    # Shape => [batch, out_steps*features].
    tf.keras.layers.Dense(OUTPUT_LEN*num_features,
                          kernel_initializer=tf.initializers.zeros()),
    # Shape => [batch, out_steps, features].
    tf.keras.layers.Reshape([OUTPUT_LEN, num_features])
])
```

Se representa a continuación ejemplo de sus predicciones, y resultado del entrenamiento.

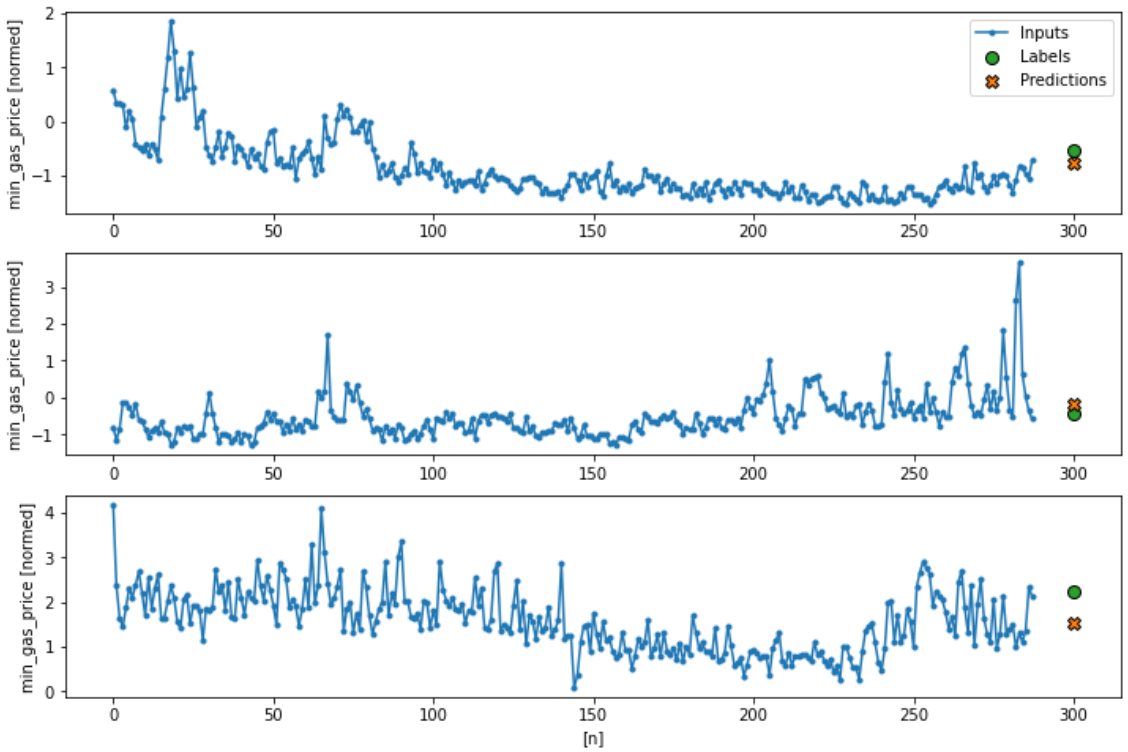


Figura 33. Predicciones de Red Neuronal Recurrente basada en LSTM

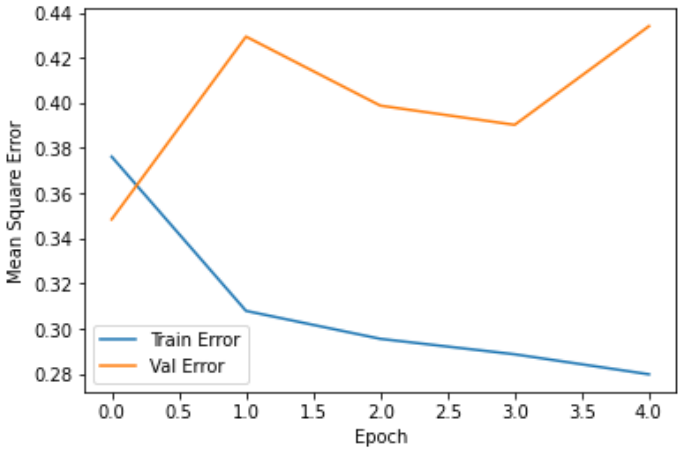


Figura 34. Entrenamiento de la Red RNN

5.13. Comparativa de Rendimientos

En la siguiente figura tenemos una **comparativa de los resultados** de los modelos descritos anteriormente. Se ha empleado el Error Absoluto Medio, calculado como la media de las diferencias absolutas entre los valores predichos y los reales, para las secuencias de validación y test.

Se puede observar cómo los **mejores resultados se obtienen para el Perceptrón Multicapa**. La red RNN, en cambio, no consigue mejorar sus resultados, pese a su teórica mayor capacidad para predecir series numéricas. Esto parece sugerir que no existe correlación temporal en las series – habría más información que obtener del resto

de features en el mismo instante de tiempo, para lo cual el Perceptrón tiene más capacidad.

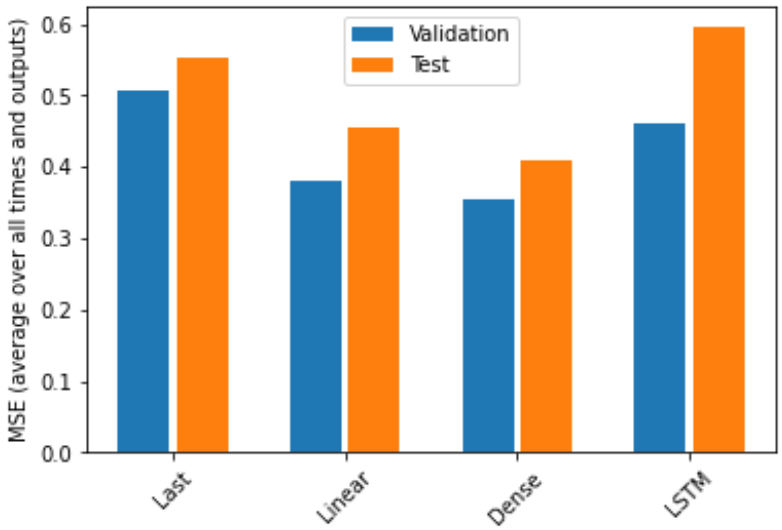


Figura 35. Comparativa de resultados de los modelos. Ventana de 24 horas para intervalos de 5 minutos

Se han **repetido** los entrenamientos con **distintas combinaciones de meta-parámetros**, y se ha encontrado que, empleando intervalos de un minuto frente a cinco – y reduciendo la ventana a 12 horas, la red LSTM mejora su comportamiento frente a los restantes modelos, si bien los errores resultan ser mayores.

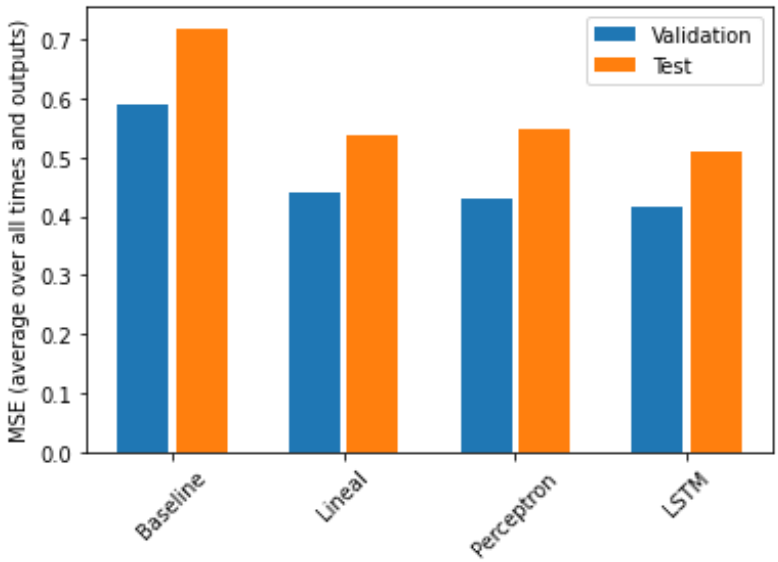


Figura 36. Comparativa de resultados de los modelos. Ventana de 12 horas para intervalos de 1 minuto

Es interesante señalar que inicialmente, se trató de predecir **el valor medio del precio en lugar del precio mínimo**. En todos los casos se obtuvieron resultados bastante malos, que mejoraron en poco respecto al Benchmark, como se puede apreciar en el siguiente gráfico.

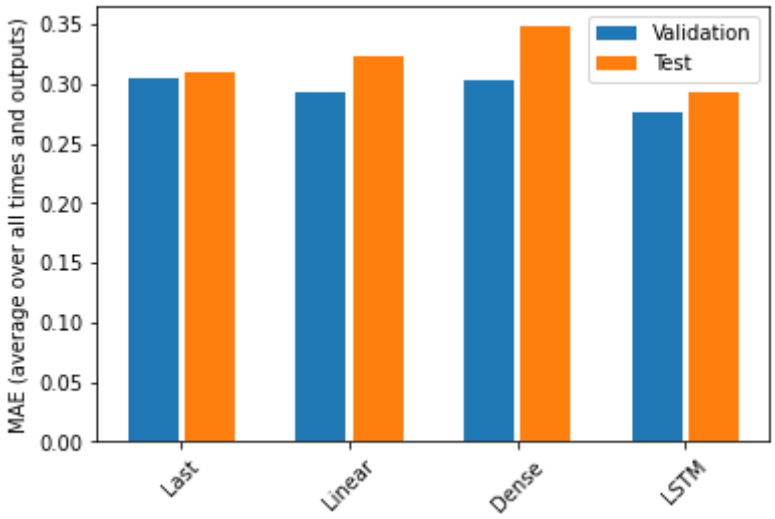


Figura 37. Comparativa de resultados al tratar de predecir el precio medio en lugar del mínimo

Por ello, se decidió pasar a predecir el **valor mínimo del precio,** momento a partir del cual los resultados mejoraron claramente.

Una explicación plausible es que el valor medio contiene muchas más variaciones aleatorias que el valor mínimo. Esto a su vez es un indicio de que la mayoría de transacciones no tienen un coste óptimo – los usuarios están pagando de más frente al valor mínimo de cada bloque, que sería el óptimo para cada transacción.

Conocer el **valor mínimo** futuro tiene además una **mayor utilidad** que el medio, ya que la **optimización del gasto** se consigue acercándose lo más posible a aquél que a este último.

6. Conclusiones

Se presentan a continuación algunas conclusiones de los resultados obtenidos en el presente trabajo, así como

- En el presente trabajo se ejecutado con éxito el **ciclo de vida típico de un Proyecto de Deep Learning** para la predicción del precio del gas:
    - Se ha definido el problema inicial en términos solubles por una RNA.
    - Se han definido varias modelos de Machine Learning, incluyendo varios tipos de RNAs, y se han escogido sus meta-parámetros.
    - Se ha encontrado una buena fuente de datos y preparado scripts para preparar y limpiar los Datasets.
    - Se ha realizado el entrenamiento y validación de los distintos modelos.
    - Se han comparado los resultados obtenidos entre ellos.

- Se ha comprobado como **los modelos de Red Neuronal**, en general, **consiguen** un **resultado** predictivo a partir de los datos de entrada **mejor que los modelos más simples**. No obstante, existe un límite de capacidad de predicción; si se complican lo modelos, añadiendo más elementos o capas, sólo se consigue llegar antes al overfitting.

- Lo anterior parece sugerir que existe poca información relevante para el precio en todas las features empleadas. Algunas **otras features que podrían estudiarse** son:
    - Evolución del **precio de Ethereum o de los contratos** más negociados en cada momento, como indicadores del grado de interés por parte de los usuarios en cada momento. Como se ha comentado, esta feature es externa a la red, pero sí que se utiliza en otros trabajos similares.
    - Número de **transacciones pendientes de confirmación** en cada momento. Sin duda será una métrica interesante, pero es de difícil obtención, ya que no queda reflejada de ningún modo en la cadena de bloques, y sería necesario encontrar alguna fuente externa.

- Una observación interesante ha sido comprobar la **importancia de la escala de tiempo** a que se toman las muestras y se realizan las predicciones, pues algunas features que presentan alta correlación a escalas mayores, dejan de tenerla al examinar intervalos menores de tiempo. Estas **features tienen correlación a largo plazo** debido a la estabilización de la red; **a corto plazo**, sin embargo, la red se encuentra en estado de desequilibrio, y las features **pueden tener** por tanto **un valor predictivo**, precisamente porque pueden indicar estos desequilibrios, cuya corrección puede ser predicha por el modelo.

- Se ha comprobado también como **en un Proyecto de Data Science**, la parte que **más dedicación** requiere es la **obtención y procesado de los Datasets**. La parte de construcción, entrenamiento y aplicación de modelos, con el Estado del Arte actual de los frameworks empleados es relativamente sencillo de realizar.

- Del mismo modo, se ha constatado **la importancia** de manipular y **comprender las características del Dataset** del problema a **la hora de escoger los** modelos y sus características, los llamados **meta-parámetros** de la red: número y tipo de capas, elementos de las mismas, funciones de activación, métricas y optimizadores, etc,

  Si es importante exponer a la red a una amplia muestra de datos para entrenar el modelo y afinar de este modo sus parámetros, igualmente debe antes el diseñador del modelo analizarlos para seleccionar los mejores meta-parámetros.

De hecho, la **optimización automática de estos meta-parámetros** es un tema de investigación actual. [39] [40]

- Finalmente, tras la ejecución del presente trabajo, se considera **muy ajustado el plazo de un cuatrimestre** para ponerse al día desde cero, y adquirir la soltura operativa necesaria con las herramientas de Deep Learning empleadas. Por ello, se han dejado de trabajar **otros aspectos que se consideran interesantes**, como la incorporación de más features a las usadas, o la ejecución de mayor número de pruebas con otros modelos y meta-parámetros.

- Igualmente, ha quedado sin poder analizarse la **relación entre el precio del gas propuesto y el retraso en la ejecución de las transacciones**, aspecto complementario al del precio, y de importancia práctica en la actualidad, tal como se puede comprobar en varios de los artículos de investigación referenciados en el Estado del Arte.

- Otro aspecto que podría ser interesante **analizar** es el proceso de **cálculo del precio base del gas**, incorporado a la red tras el **fork London** ya descrito [25], y su comparativa con la predicción **mediante Deep Learning**.

7. Glosario

Bitcoin
Primera implementación de una tecnología de blockchain, que constituye una criptodivisa capaz de alamcenar e intercambiar valor económico, 1, 2, 16, 17, 18, 19, 22

Blockchain
Cadena de bloques, tecnología que permite el mantenimiento de un registro confiable de manera descentralizada mediante empleo de técnicas criptográficas, 1

Deep Learning
Disciplina perteneciente a la IA basada en la construcciones de RNAs de varias capas de profundidad, i, iii, 1, 2, 3, 6, 8, 10, 11, 12, 13, 21, 22, 33

Ethereum
Blockchain que fue pionera en la introducción de Aplicacones Distribuidas además de moneda digtal, i, iii, 1, 2, 3, 6, 16, 17, 18, 19, 20, 21, 22, 23, 24, 30

Hype Cycle
Ciclo de vida de una tecnología, desde el punto de vista de su adopción, uso y difusión por los medios., 1

IA conexionista
Técnicas de IA basadas en la creación dredes neuronales, basadas en neuronas interconectadas entre sí, 9

IA Simbólica
Técnicas de la IA basadas en la codificación del conocimiento mediante símbolos manipulables mediante reglas, 9

Labels
Valores de salida de un modelo de Machine Lerning. Hace referencia a las etiquets de salida de un problema de clasificación, 9

Machine Learning
Técnicas de la IA que persiguen enseñar a un ordenador a resolver un problmea suministrando ejemplos de resouliciones, frente a la Computación cásica, que describe los pasos necesarios para la reolución del problema, 9

Perceptrón
Uno de los primeros intentos de creación de una Red Neuronal Artifical. Evolucionó hasta el Perceptrón Multicapa, tipo de RNA muy empleado en la actualidad., 10

Redes Neuronales Artificiales
Modelos de IA inspirados en la biología que byuscan convertir un conjunto de entradas en una salida de manera similar ac omo funcionan los cerebros biológicos, 10

Smart Contracts
Aplicaciones distribuidas que se ejeutn dentro de una blockchain, como Ethereum u otras posteriores, 1

transacción
Unidad de ejecución dentro de una blockchain, en donde un suaurio ordena el envió o de divisa o ejecucióno de un contrato desde una cuenta que controla, 1

AI
Artificial Intelligence, ó Inteligencia Artificial

ANN
Artificial Neural Network, ó Red Neuronal Artificial

Bitcoin
Primera implementación de una tecnología de blockchain, que constituye una criptodivisa capaz de almacenar e intercambiar valor económico.

Blockchain

Cadena de bloques, tecnología que permite el mantenimiento de un registro confiable de manera descentralizada mediante empleo de técnicas criptográficas.

Bloque

Cada uno de los elementos consecutivos y firmados que contienen toda la evolución de estados de una Blockchain. Normalmente cada bloque es un conjunto de transacciones, más una firma digital que garantiza su autenticidad y posición en la cadena.

Deep Learning

Disciplina perteneciente a la IA, basada en la construcción de RNAs de varias capas de profundidad

ERC-20

Estándar de un tipo de Contrato Inteligente que facilita la implantación de Tokens de tipo monetario en la red Ethereum

Ether

Unidad básica monetaria dentro de la red Ethereum. Equivale a 1e18 weis

Ethereum

Blockchain que fue pionera en la introducción de Aplicaciones Distribuidas además de moneda digital.

Feature

En Machine Learning, cada una de las características de una serie de datos, que puede ser actuar como entrada y/o salida al modelo

Fork

En la Red Ethereum, introducción de una nueva versión de código que implica un cambio funcional importante.

Función de activación

Función no lineal que suele ser el paso final de procesado de muchas capas de RNAs. La introducción de una no linealidad es clave para la capacidad de predicción de las RNAs

GPU

Graphical Processing Unit. Hardware diseñado para el procesado de gráficos. Sus capacidades de cálculo en paralelo lo hacen muy adecuado para problemas de Machine Learning.

Gwei

Giga wei. Unidad monetaria en la red Ethereum, empleada para establacer el precio del Gas, y que equivale a 1e+9 weis ó 1e-9 Ethers

Hype Cycle

Ciclo de vida de una tecnología, desde el punto de vista de su adopción, uso y difusión por los medios.

IA

Inteligencia Artificial

IA conexionista

Técnicas de IA basadas en la creación dredes neuronales, basadas en neuronas interconectadas entre sí.

IA Simbólica

Técnicas de la IA basadas en la codificación del conocimiento mediante símbolos manipulables mediante reglas.

Labels

Valores de salida de un modelo de Machine Lerning. Hace referencia a las etiquets de salida de un problema de clasificación,.

LSTM

Long shor-term memory. Tipo de RNN utilizada para la predicción de series temporales

Machine Learning

Técnicas de la IA que persiguen enseñar a un ordenador a resolver un problmea suministrando ejemplos de resouliciones, frente a la Computación cásica, que describe los pasos necesarios para la reolución del problema.

Minería
Actividad llevada a cabo por nodos participantes en una red blockchain, durante la cual ejecutan operaciones criptográficas con objeto de asegurar la seguridad de la misma (Proof of Work), recibiendo a cambio un pago en criptomoneda

NFT
Non Fungible Token. Tipo de Tokens con propiedades de unicidad e indivisibilidad, que permiten representar la propiedad de objetos o activos únicos dentro de una Blockchain.

Overfitting
Sobreentrenamiento. Fenómeno que ocurre al entrenar en exceso una RNA, más allá del punto en que la red deja de aprender el patrón general de la secuencia para comenzar a aprender los detalles del conjunto particular de datos usados en el entrenamiento.

Perceptrón
Uno de los primeros intentos de creación de una Red Neuronal Artifical. Evolucionó hasta el Perceptrón Multicapa, tipo de RNA muy empleado en la actualidad.

PoF
Proof of Work, vease Minería

PoS
Proof of Stake. Actividad llevada a cabo por nodos participantes en una red blockchain para asegurar su funcionamiento, a cambo de una comisión. Se asegura su correcto comportamiento mediante el Stake, o fondos bloqueados susceptibles de perderse en caso de intentar cometer fraude.

Red Neuronal Artificial
Modelos de IA inspirados en la biología que buscan convertir un conjunto de entradas en una salida de manera similar ac omo funcionan los cerebros biológicos.

RELU
Rectified Linear Unit. Un tipo de función de activación común en muchas RNAs

RNA
Red Neuronal Artifical

RNN
Recurrent Neural Network. Red Neuronal Recurrente. Tipo de RNA especialmente adecuada para predicción de series temporales.

Smart Contract
Contrato inteligente, o Aplicación Distribuida. Aplicación que se ejecuta dentro de la red blockchain, asegurando su comportamiento de acuerdo a la lógica programad, y garantizando su validez mediante el entorno criptrográfico de la red

Smart Contracts
Aplicaciones distribuidas que se ejeutn dentro de una blockchain, como Ethereum u otras posteriores,.

Softmax
Normalized Exponential Function. Un tipo de función de activación común en problemas de clasificación.

TPU
Tensor Processing Unit. Hardware diseñado específicamente para abordar problemas de Machine Learning, basados en la manipulación de tensores. Puede contemplarse como la evolución de las GPUs

Transacción
Unidad de ejecución dentro de una blockchain, en donde un suaurio ordena el envió o de divisa o ejecucióno de un contrato desde una cuenta que controla.

Wei
Unidad monetaria en la red Ethereum, empleada para establacer el precio del Gas, y que equivale a 1e-18 Ethers

8. Bibliografía

[1] Gartner, «Hype Cycle Research Methodology,» 2021. [En línea]. Available: https://www.gartner.com/en/research/methodologies/gartner-hype-cycle.

[2] B. G. Buchanan, «A (Very) Brief History of Artificial Intelligence,» *AI Magazine Volume,* vol. 26, nº 4, 2005.

[3] A. M. Antonopoulos, Mastering Bitcoin: Programming the Open Blockchain, 2 ed., Sebastopol, CA: O'Reilly, 2017.

[4] A. M. Antonopoulos y G. Wood, Mastering Ethereum: Building Smart Contracts and DApps, S.l: Stanford University Press, 2018, p. 384.

[5] TensorFlow, Udacity, «Course: Intro to TensorFlow for Deep Learning,» 2021. [En línea]. Available: https://www.udacity.com/course/intro-to-tensorflow-for-deep-learning--ud187.

[6] F. Chollet, Deep Learning with Python, 2 ed., Shelter Island, NY: Manning, 2021.

[7] J. Torres, Python Deep Learning. Introducción práctica con Keras y TensorFlow 2, Barcelona: Marcombo, 2020.

[8] I. Goodfellow, Y. Bengio y A. Courville, Deep Learning, MIT Press, 2016.

[9] N. Bostrom, Superintelligence: Paths, Dangers, Strategies, Paperback ed., New York: Oxford University Press, 2016, p. 415.

[10] Wikipedia, «History of artificial intelligence,» 2021. [En línea]. Available: https://en.wikipedia.org/wiki/History_of_artificial_intelligence.

[11] Wikipedia, «Machine Learning,» 2021. [En línea]. Available: https://en.wikipedia.org/wiki/Machine_learning.

[12] J. Anguera Pros y A. Andújar Linares, «Optimización de antenas mediante Algoritmos Genéticos,» de *Diseño y Aplicaciones de Antenas. Módulos Didácticos.*, Barcelona, Fundació UOC, 2014.

[13] Wikipedia, «Perceptron,» 2021. [En línea]. Available: https://en.wikipedia.org/wiki/Perceptron.

[14] J. A. Morán Moreno y J. C. Socoró Carrié, «Filtros Adaptativos,» de *Procesado Avanzado. Módulos Didácticos.*, Barcelona, Fundació UOC, 2014.

[15] J. Brownlee, «Difference Between Backpropagation and Stochastic Gradient Descent,» 2021. [En línea]. Available: https://machinelearningmastery.com/difference-between-backpropagation-and-stochastic-gradient-descent/.

[16] S. Hasan, «Recurrent Neural Network and it's variants….,» 2020. [En línea]. Available: https://medium.com/analytics-vidhya/recurrent-neural-network-and-its-variants-de75f9ee063.

[17] Wikipedia, «Activation Function,» [En línea]. Available: https://en.wikipedia.org/wiki/Activation_function. [Último acceso: 2021].

[18] Wikipedia, «Rectified Linear Unit activation function,» [En línea]. Available: https://en.wikipedia.org/wiki/Rectifier_(neural_networks). [Último acceso: 2021].

[19] S. Nakamoto, «Bitcoin: A Peer-to-Peer Electronic Cash System,» 2009. [En línea]. Available: https://bitcoin.org/bitcoin.pdf.

[20] V. Buterin, «Ethereum Whitepaper,» 2013. [En línea]. Available: https://ethereum.org/en/whitepaper/.

[21] E. Conner, «Understanding Ethereum Gas, Blocks and the Fee Market,» 2019. [En línea]. Available: https://medium.com/@eric.conner/understanding-ethereum-gas-blocks-and-the-fee-market-d5e268bf0a0e.

[22] A. I. Sanka y R. C. Cheung, «A systematic review of blockchain scalability: Issues, solutions, analysis and future research,» *Journal of Network and Computer Applications,* vol. 195, 2021.

[23] ledger.com, «The Blockchain Generations,» May 2021. [En línea]. Available: https://www.ledger.com/academy/blockchain/web-3-the-three-blockchain-generations.

[24] ethereum.org, «The Eth2 Vision,» [En línea]. Available: https://ethereum.org/en/eth2/vision/. [Último acceso: 2021].

[25] A. Tiwari, «Ethereum's London hard fork sets ETH on a more deflationary path,» 2021. [En línea]. Available: https://cointelegraph.com/news/ethereum-s-london-hard-fork-sets-eth-on-a-more-deflationary-path.

[26] T. Zoumpekas, E. Houstis y M. Vavalis, «ETH analysis and predictions utilizing deep learning,» *Expert Systems with Applications,* vol. 162, p. 113866, 12 2020.

[27] D. Kumar y S. K. Rath, «Predicting the Trends of Price for Ethereum Using Deep Learning Techniques,» *Advances in Intelligent Systems and Computing,* vol. 1056, pp. 103-114, 2020.

[28] V. C. Oliveira, J. Almeida Valadares, J. E. A. Sousa, A. Borges Vieira, H. Soares Bernardino, S. Moraes Villela y G. Dias Gonçalves, «Analyzing Transaction Confirmation in Ethereum Using Machine Learning Techniques,» *ACM SIGMETRICS Performance Evaluation Review,* vol. 48, nº 4, pp. 12-15, 5 2021.

[29] H. J. Singh y A. S. Hafid, «Transaction Confirmation Time Prediction in Ethereum Blockchain Using Machine Learning,» 11 2019.

[30] F. Liu, X. Wang, Z. Li, J. Xu y Y. Gao, «Effective GasPrice Prediction for Carrying Out Economical Ethereum Transaction,» *Proceedings - 2019 6th International Conference on Dependable Systems and Their Applications, DSA 2019,* pp. 329-334, 1 2020.

[31] S. M. Werner, P. J. Pritz y D. Perez, «Step on the Gas? A Better Approach for Recommending the Ethereum Gas Price,» *arXiv,* pp. 161-177, 3 2020.

[32] Google Cloud, «Ethereum in BigQuery: a Public Dataset for smart contract analytics,» 2021. [En línea]. Available: https://cloud.google.com/blog/products/data-analytics/ethereum-bigquery-public-dataset-smart-contract-analytics.

[33] P. Siriwardena, «The Mystery Behind Block Time,» 2017. [En línea]. Available: https://medium.facilelogin.com/the-mystery-behind-block-time-63351e35603a.

[34] P. Wackerow, «Ethereum Blocks,» 2021. [En línea]. Available: https://ethereum.org/en/developers/docs/blocks/.

[35] R. Kiamil, «Full relational diagram for Ethereum public data on Google BigQuery,» 2021. [En línea]. Available: https://medium.com/google-cloud/full-relational-diagram-for-ethereum-public-data-on-google-bigquery-2825fdf0fb0b.

[36] Google Cloud, «BiqQuery Timestamp functions,» [En línea]. Available: https://cloud.google.com/bigquery/docs/reference/standard-sql/timestamp_functions#timestamp_trunc. [Último acceso: 2021].

[37] J. Korstanje, «Fourier Transform for Time Series,» 2021. [En línea]. Available: https://towardsdatascience.com/fourier-transform-for-time-series-292eb887b101.

[38] TensorFlow.org, «Time series forecasting,» Nov 2021. [En línea]. Available: https://www.tensorflow.org/tutorials/structured_data/time_series.

[39] T. Hartmann, «Meta-Modelling Meta-Learning,» 2019. [En línea]. Available: https://medium.com/datathings/meta-modelling-meta-learning-34734cd7451b.

[40] J. Brownlee, «What Is Meta-Learning in Machine Learning?,» 2021. [En línea]. Available: https://machinelearningmastery.com/meta-learning-in-machine-learning/.

[41] Wikipedia, «Deep learning,» [En línea]. Available: https://en.wikipedia.org/wiki/Deep_learning. [Último acceso: 2021].

9. Anexos

Se ha creado un repositorio disponible en la URL de más abajo, en donde se dejará disponible el material indicado.

https://github.com/antarias/tfm-deepeth

9.1. Scripts Python

Se dejan disponibles los principales scripts empleados en el presente trabajo, en formato Colab Notebook. Para su correcta ejecución deben copiarse a una carpeta de Google Drive, y otorgarles permisos de acceso a la misma. Los scripts y su contenido son los siguientes:

0.config.py
Fichero de configuración incluido por los demás scripts, que define el rango de fechas, resolución horaria y nombre del Dataset a emplear.

1.Preparar Dataset.ipynb
Script que captura el Dataset según los parámetros definidos a partir de BigQuery, y lo almacena en Google Drive

2. Limpiar y Preparar.ipynb
Script que lee el Dataset anteriormente generado, lo limpia de outliers y elementos vacíos, y lo almacena de nuevo en Google Drive

3. Analisis Dataset.ipynb
Script que realiza el Análisis preliminar de datos del dataset seleccionado.

4. Entrenar y Probar.ipynb
Script que crea, entrena y compara los resultados de varios modelos con el dataset indicado.

9.2. Datasets

Se almacenan los datasets capturados y usados en el trabajo.

9.3. Otros

Bibliografía, copia de la presente Memoria y Presentación del Proyecto.