PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

IMPLEMENTACIÓN DE UN MODELO MACHINE LEARNING DISTRIBUIDO EN COMPUTADORES DE BAJO COSTO Y CONSUMO DE ENERGÍA

DAMIR EXZAEL ALIQUINTUI PEREIRA
JOSE MATIAS CAMPUSANO CASTILLO

Profesor Guía: **HECTOR ALLENDE CID**
Profesor Correferente: **WENCESLAO PALMA MUÑOZ**

INFORME DE PROYECTO
PARA OPTAR AL TÍTULO PROFESIONAL DE
INGENIERO CIVIL INFORMÁTICA

DICIEMBRE 2018

Resumen

Debido al constante crecimiento de los datos e información que brindan los diferentes sistemas, se hace necesario buscar nuevas formas de crear herramientas de aprendizaje automatizados que cumplan con un buen nivel de precisión y, así mismo, cumplir esto de la manera más rápida posible y sin comprometer la seguridad e integridad de los datos.

Debido a esto surgen nuevas soluciones e innovaciones tecnológicas para satisfacer este tipo de problemas. Así es como *blockchain* surge como una estructura de datos que permite el almacenaje de información de manera distribuida, y al mismo tiempo, es capaz de verificar las transacciones realizadas manteniendo la información actualizada en cada uno de los equipos conectados. Todo esto funciona utilizando una arquitectura Peer to Peer (P2P), donde cada nodo cumple la función de validar las transacciones realizadas, entregando mayor seguridad a los datos.

Así es como esta tesis busca aplicar herramientas de aprendizaje de ensamblado, basado en otra investigación, en un sistema que contienen datos de forma distribuida, constituido principalmente de nodos con una capacidad de computo baja, conectados a través de la tecnología *Ethereum*, la cual se basa en la estructura de datos conocida como *blockchain*. De esta forma se busca conseguir entrenar un conjunto de modelos los cuales permitan predecir el comportamiento de los datos, sin dejar de lado el contexto de donde provienen ni comprometer la seguridad de los datos al ser procesados por este sistema distribuido y con un sistema P2P que permite escalar esta solución de forma fácil y rápida.

Es así como este estudio permitirá el desarrollo de máquinas de aprendizaje automatizado de forma distribuida utilizando *blockchain*. Esto, en sistemas asequibles para cualquier interesado en el tema y, a su vez, servirá de guía para la implementación en dichos sistemas. Por lo que se espera que se puedan realizar futuros desarrollos prácticos, experimentos y mejoras para la solución propuesta.

Palabras-claves: *machine learning, sistemas distribuidos, vecindario, aprendizaje automatizado, blockchain, ethereum, peer to peer.*

Abstract

Due to the constant growth of the data and information provided by the different systems, It is necessary to look for new ways to create automated learning tools that comply with a good level of precision and, likewise, meet this as quickly as possible and without compromising the security and integrity of the data.

Due to this new solutions and technological innovations arise to satisfy this type of problems. This is how *blockchain* arises as a data structure that allows the storage of information in a distributed way, and at the same time, it is able to verify the transactions made keeping the information updated in each of the connected equipment. All this works using an architecture P2P, where each node fulfills the function of validating the transactions carried out, giving greater security to the data.

This is how this thesis seeks to apply assembly learning tools, based on other research, in a system that contains data in a distributed form, consisting mainly of nodes with a low computing capacity, connected through technology *Ethereum* , which is based on the data structure known as *blockchain*. In this way we seek to train a set of models which allow us to predict the behavior of the data, without leaving aside the context from which they come or compromising the security of the data when processed by this distributed system and with a system P2P that allows you to scale this solution easily and quickly.

This is how this study will allow the development of automated learning machines in a distributed way using *blockchain*. This, in systems accessible to anyone interested in the subject and, at the same time, will serve as a guide for the implementation in said systems. So it is expected that future practical developments, experiments and improvements for the proposed solution can be made.

Keywords: *machine learning, distributed systems, neighborhood, automated learning, blockchain, ethereum, peer to peer.*

Índice

Resumen I

Abstract II

Siglas V

Lista de Figuras VI

Lista de Tablas VII

1. Introducción 1

2. Marco Teórico 3

3. Descripción del Algoritmo 7
- **3.1. Fase 1: Entrenamiento Local** 8
- **3.2. Fase 2: Modelo y Transmisión de información** 8
- **3.3. Fase 3: Generación de la predicción** 9

4. Arquitectura 10
- **4.1. Nodo** 10
  - 4.1.1. API REST 10
  - 4.1.2. GETH 11
- **4.2. Comunicación entre Nodos** 11

5. Implementación 12
- **5.1. Herramientas** 12
  - 5.1.1. API REST 12
  - 5.1.2. Blockchain 12
- **5.2. Smartcontract** 13
  - 5.2.1. Métodos del SmartContract 13
- **5.3. API REST** 14
  - 5.3.1. ETL 15
  - 5.3.2. GETH 15
- **5.4. Models** 15

6. Experimentación 16
- **6.1. Procedimiento** 16
- **6.2. Resultados** 17

7. Conclusiones 20

Anexos 25

A:. Resultados con 2 clusters..... 25
B:. Resultados con 3 clusters..... 28
C:. Entrenamiento local 31

Siglas

DML Distributed Machine Learning. 3, 4, 12, 20

IoT Internet of Things. 1

IT Information Technology. 1

ML Machine Learning. 1–3, 7, 20

P2P Peer to Peer. 4–6, 11–13

PML Parallel Machine Learning. 3

POA Proof of Authority. 5, 12

POS Proof of Stake. 5

POW Proof of Work. 5

Lista de Figuras

3.1. Flowchart completo 7
3.2. Flowchart de la Fase 1 8
3.3. Flowchart de la Fase 2 9
3.4. Flowchart de la Fase 3 9
4.1. Arquitectura Lógica de un Nodo 10
4.2. Comunicación entre nodos..... 11
5.1. Estructura P2P..... 13
5.2. Estructura API..... 14
6.1. Score por iteración para 100.000 datos y función (4) 17
6.2. Comparación de R2 score para función (4) con 1.000, 10.000, 100.000 datos 18
6.3. Comparación de R2 score para funciones generadoras (1), (2), (3) 18
6.4. Comparación de R2 Score para 2 y 3 vecindades con 100.000 datos y función (4) 19

Lista de Tablas

1. single MSE con 1000 datos 25
2. distribuido MSE con 1000 datos 25
3. single R2 con 1000 datos 25
4. distribuido R2 con 1000 datos 25
5. single MSE con 10000 datos 26
6. distribuido MSE con 10000 datos 26
7. single R2 con 10000 datos 26
8. distribuido R2 con 10000 datos 26
9. single MSE con 100000 datos 26
10. distribuido MSE con 100000 datos 27
11. single R2 con 100000 datos 27
12. distribuido R2 con 100000 datos 27
13. single MSE con 1000 datos 28
14. distribuido MSE con 1000 datos 28
15. single R2 con 1000 datos 28
16. distribuido R2 con 1000 datos 29
17. single MSE con 10000 datos 29
18. distribuido MSE con 10000 datos 29
19. single R2 con 10000 datos 29
20. distribuido R2 con 10000 datos 30
21. single MSE con 100000 datos 30
22. distribuido MSE con 100000 datos 30
23. single R2 con 100000 datos 30
24. distribuido R2 con 100000 datos 30
25. Resultados utilizando 1000 datos y 8 iteraciones en el modelo local 31
26. Resultados utilizando 1000 datos y 100 iteraciones en el modelo local 31

1 Introducción

Gracias a las nuevas tecnologías junto con las grandes capacidades que se posee hoy en día para el almacenamiento y generación de datos, van surgiendo nuevas problemáticas a abordar. Entre lo que se puede destacar es el tratamiento y la obtención de información de este gran volumen de datos, no obstante, no existe solo una forma de abordar dicho enfoque, lo que da a lugar a diversas investigaciones afines en este tema. Esto da origen a la investigación acá propuesta, la cual consiste en abordar esto a través de un sistema de aprendizaje automatizado implementado sobre una arquitectura distribuida.

Así es como se puede observar que redes sociales, tales como *Facebook*, *Twitter*, entre otras, son las principales generadoras de datos, siendo la primera que, solo utilizando sus aplicaciones de mensajería, manejan un volumen de 60 billones de mensajes diarios. No obstante, los datos no solo provienen de redes sociales o interacciones de usuarios con dichas aplicaciones; el Internet of Things (IoT) hoy está más presente que nunca alrededor del mundo. Esto incluye sensores de temperatura, movimiento, sonidos, y de diferente índole, que dan vida hoy en día a las conocidos sistemas inteligentes, las cuales tienen como objetivo principal facilitar la vida de los usuarios automatizando los procesos cotidianos.

Se estima que para el 2020 el universo digital llegue a pesar 40.000 exabytes, duplicando su tamaño por cada año que pasa. Así mismo, la inversión en hardware, software, servicios y telecomunicaciones en Information Technology (IT) crecerá un 40 % desde el 2012 al 2020 y la inversión dedicada a *Big Data* y *Cloud Computing* crecería considerablemente más rápido [11].

Junto con todo este crecimiento de datos, empiezan a aparecer nuevas problemáticas a ser resueltas, dentro de las cuales se encuentra la necesidad de un aprendizaje automático de estos datos, de forma que ya no solo se cuente con los obtenidos y almacenados, si no que también puedan ser utilizados para la predicción y la toma de decisiones, abriendo con esto los estudios relacionados con el aprendizaje de computadores, mejor conocido hoy en día como Machine Learning (ML).

Dentro de las ramas que aparecen dentro de ML, se pueden apreciar distintos enfoques y estrategias que se utilizan para lograr el objetivo descrito anteriormente. No obstante, cada una de estas aproximaciones presentan nuevos desafíos a ser resueltos. Es así como la arquitectura clásica del ML, que está enfocada en un sistema monolítico, es decir, donde el conjunto de datos será almacenado y procesado en la memoria de un solo sistema, se ve fuertemente afectado cuando el conjunto de datos empieza a crecer o está distribuido en diferentes fuentes, dificultando así la tarea del aprendizaje al tener que considerar el transporte de dichos datos a este nodo central.

Por otro lado, *Blockchain* es una tecnología que a tenido un fuerte auge, pese a que nace desde las criptomonedas, hoy en día se le está dando un uso mucho más versátil aprovechando las fortalezas que ofrece a la hora de manejar datos distribuidos y validar las transacciones hechas por los distintos nodos que pertenecen a una red.

Uno de los principales focos que se estudiará en esta tesis, es cuando la fuente de los datos se encuentran distribuidos ya sea topológicamente o producto de su arquitectura lógica y física. Es aquí cuando hay que considerar diferentes opciones para el tratamiento de dichos datos, sin perder de vista el contexto semántico de estos, y, por otro lado, considerando también la capacidad de cómputo que poseen los nodos encargados de generar este aprendizaje automático.

Con este desafío, se pretende lograr un gran avance en ML Distribuido usando una arquitectura basada en *Blockchain*, enfocado a computadoras de bajo consumo energético. Así se pretende poder llevar esta tecnología a un uso mucho más cotidiano y accesible para cualquier interesado en el área, sin perder la exactitud y velocidad de aprendizaje que se han ganado con todos los años de investigación que se han visto últimamente. También se tendrá en cuenta un fuerte enfoque a la seguridad y confidencialidad de los datos e información que se transmitan en este sistema distribuido.

Cabe destacar, que a la fecha que se escribe este documento, el uso de ML sobre una arquitectura con *Blockchain* es un tema que no ha sido abordado a gran detalle, habiendo solo ideas y conceptos. Es por esto que es posible considerar esta propuesta como una prueba de concepto, validando que es una solución viable y, así, abrir nuevas ramas de estudio enfocadas a este tema, convergiendo ambas tecnologías sacando lo mejor de cada una de ellas.

Es así como este documento comenzará presentando el marco teórico, donde se hablará de las bases que motivan el desarrollar la investigación. Aquí se explicará brevemente el estado del arte, en que consisten las tecnologías y herramientas a utilizar, el motivo por el cual se han elegido y que es lo que se pretende lograr con ellas. Luego se pasará a la descripción del algoritmo a utilizar, junto con un desglose de sus fases y un pseudocódigo que explicará paso a paso para así poder entender de mejor forma este algoritmo.

Una vez explicado las bases, se continuará con la arquitectura utilizada para efectuar la investigación. Luego, se describirán el *hardware* y el *software* utilizados, además de hacer una representación gráfica de estos y sus interacciones entre ellos. A continuación, se pasará a la sección de experimentación, donde se explicará cuales fueron los ambientes creados, así como también los resultados obtenidos. Para finalizar, se concluirá está tesis con una síntesis de los resultados obtenidos, las ventajas de está implementación y posibles aplicaciones prácticas. Y finalmente se hablará de las posibles investigaciones a desarrollar en esta área.

2 Marco Teórico

Esta tesis, tal como se habló en el apartado anterior, busca la integración de los sistemas distribuidos y el aprendizaje de computadores conectados mediante *blockchain*. Es así como se hace necesario resaltar los beneficios de cada uno de estos sistemas, así como también la sinergia obtenida al combinarlos de manera adecuada. Por lo que a continuación se describen los conceptos importantes de lo que se sustenta este documento así como también un estado del arte de cada tema.

El ML es una rama de la inteligencia artificial permite utilizar las capacidades computacionales para analizar y predecir nueva información a partir de datos obtenidos anteriormente. Esto surge debido a que ha ocurrido un gran avance en la capacidad de generar y almacenar datos, lo cual conlleva a que la capacidad humana no sea capaz de procesar tanta información. Es por esto que ha adquirido una gran popularidad en los últimos años, ya que ha demostrado ser una herramienta que permite prestar valor a diferentes campos tales como los negocios, la medicina, investigación, entre muchos otros.

Por otro lado, los sistemas distribuidos tienen una arquitectura que consta de una serie de computadores, llamados nodos, separados generalmente de forma física, pero que están comunicados entre ellos. Dentro de sus principales ventajas, se encuentra la capacidad de realizar diferentes procesos de manera concurrente y paralela, permitiendo así acelerar el tiempo de cómputo total. Adicional a esto, permite el manejo de un gran volumen de datos ya que estos también pueden ser o estar distribuidos entre dichos nodos.

Como se mencionó en la introducción, a través del tiempo ha aumentado de forma significativa la cantidad de datos y, a su vez, nuevos métodos para analizarlos. Es por esto que surge la necesidad de utilizar sistemas distribuidos con el fin de aumentar la eficiencia en el procesamiento de la información y sacar provecho de su capacidad para realizar tareas de forma paralela. Es así como surgen conceptos como Distributed Machine Learning (DML), donde el cómputo y los datos son traspasados a través de mensajes entre los nodos y Parallel Machine Learning (PML), dónde se asume que el procesamiento de los núcleos comparten la misma memoria. Cabe destacar que ambos apuntan a mejorar la eficiencia en el análisis de información.

Es importante considerar el avance que ha tenido el procesamiento de los datos a través del tiempo. Donde en un principio se comenzó a trabajar con conjuntos de datos monolíticos, es decir, centralizados en un nodo, pero con el tiempo se hizo necesario comenzar a utilizar los conjuntos de datos distribuidos. No obstante, el costo de pasar la gran cantidad de datos desde un nodo a otro provocaba un cuello de botella y una posible brecha de seguridad ya que dicha información podría ser interceptada generando grandes pérdidas. Siendo de esta manera, un punto importante el identificar qué información intercambiar entre estos.

La mayoría de los DML trataban al conjunto de datos distribuidos como una sola tabla virtual por lo que se intenta obtener el mejor modelo para esta gran cantidad de datos. Esto era un problema debido que algunos Dataset locales, es decir, que está almacenado en unos de los nodos distribuidos, pueden tener alguna relación con el contexto de donde se están obteniendo,

generando así que la distribución de estos datos cambie para cada subconjunto [26]. Es por esto que se comienza enfocar en la combinación de modelos predictivos y así mejorar la precisión del modelo. Así es como comienzan nuevas investigaciones sobre un nuevo concepto llamado como "sistema de aprendizaje distribuido basado en el método de ensamblado" [9] [17] [18]. Por lo que surgieron novedosos métodos para la los datos de entrenamiento como *simple average, stacking, majority voting, winner-takes-all or boosting*. Por lo que la idea de estos métodos es reducir el costo de comunicación entre los nodos y mejorar el sistema de predicción.

Existen una gran cantidad de investigaciones en relación a problemas de clasificación distribuida [8] [20] [23], data clustering distribuido [6] [10] [15] y problemas de regresión distribuida [28] [24] [22], siendo este último el método en el que se enfocará la tesis. Además, complementario al método de regresión, se utilizarán distintos métodos para la obtención de la descripción de distribución de los daos y cálculo de vecindades y sus respectivas distancias, el cual está explicado en [4]. Este algoritmo consiste en que los modelos locales de cada nodo son entrenados con sus datos locales, luego se comparte información característica de su conjunto de datos y del modelo generado, con lo cual se pueden calcular la similitud de cada nodo con los demás, construyendo así los vecindarios. De esta forma, el resultado final del modelo será la respuesta de los modelos que pertenecieron al mismo vecindario los cuales serán los datos de entrada para un algoritmo de agrupación jerárquica que está inspirado en el *modelo de apilado de Wolpert* [27].

Actualmente, existen estudios similares que buscan llevar DML a dispositivos de bajo consumo, entre ellos se encuentra Google, uno de las empresas más grandes dentro de la innovación tecnológica. Dentro de sus investigaciones [16], ellos buscan entrenar con alta precisión un modelo cuyos datos estén distribuidos en un gran número de clientes los cuales posean una conexión inestable y lenta a internet, donde dichos dispositivos serían principalmente teléfonos móviles.

El problema de este tipo de soluciones es que su arquitectura es una red centralizada por lo que si el nodo principal falla, dejará de funcionar la red. Es por esto que es necesario poder general una arquitectura P2P descentralizada para mejorar la escalabilidad, eliminando la dependencia que tienen los nodos, aumentando la disponibilidad. Como solución a este problema se encuentra *Blockchain*, el cual es una estructura de datos basada en una arquitectura P2P descentralizada de alta disponibilidad, capaz de mantener la consistencia e integridad de los datos.

Como se ha mencionado, *Blockchain* [14] consiste en almacenar información de manera distribuida, donde cada nodo maneja la misma información y, por lo tanto, puede validar cualquier tipo de transacción que se quiera efectuar dentro de su cadena de bloques. Esta cadena consiste en un conjunto de bloques conectados, donde cada bloque conoce su predecesor, exceptuando el bloque inicial llamado génesis el cual cumple la función de configurar la cadena. Además contiene información dentro de si y una huella digital que se genera en base a la información contenida. Como cada nodo posee la misma cadena, una vez que se genera un bloque nuevo, este tiene que ser validado por la misma cadena local, verificando la conexión y las huellas digitales de sus demás bloques, también debe ser validado por los otros nodos pertenecientes a la red, de esta forma se mantiene una red distribuida, consistente y a prueba de alteraciones

que puedan ser realizadas por terceros.

Adicionalmente, dentro de la plataforma de *Ethereum*, la cual proporciona la creación y administración de una cadena de bloques, existe el concepto de *SmartContract*, el cual define un contrato y sus reglas escritas como funciones de manera que se pueden realizar transacciones sobre este mismo [19]. Para lograr esto, es necesario generar la transacción del contrato, y una vez que este es agregado a la cadena, todos los nodos podrán acceder a las funciones descritas por este contrato y por ende, cualquier tipo de transacción que se efectué sobre este, podrán ser validados por todos los nodos pertenecientes a la red.

Por último, para entender *Blockchain* hay que comprender como es la generación de estos bloques. Para esto es necesario que los nodos estén generando bloques (concepto conocido como "minado"), proceso el cual consiste en tomar las transacciones hechas por los nodos, y agregar estas a un bloque para posteriormente pasar a una validación hecha por la red. Existen distintos tipos de validaciones, tales como Proof of Work (POW), Proof of Authority (POA) y Proof of Stake (POS) las cuales serán explicadas a continuación.

- POW consiste en la resolución de un algoritmo, donde la complejidad inicial de este viene definido por el Génesis e irá aumentando a través de la generación de bloques. En este tipo de minados, los nodos compiten para resolver el algoritmo y poder generar un bloque nuevo.
- POA consiste en que dentro del nodo génesis se definen cuentas habilitadas para generar bloques, a diferencia de método anterior, aquí los nodos no compiten, simplemente se elige uno de los habilitados y este tomará las transacciones hechas y las inyectará a un bloque. No obstante, la red sigue validando que sea una cadena válida.
- POS es similar al POA, con la única diferencia que el nodo elegido es quien tiene mayor influencia en la red, es decir, quien tiene mayor cantidad de ether.

Dentro de las soluciones e investigaciones sobre *blockchain* que se pueden destacar, se encuentran la creación de criptomonedas, donde hoy en día existen hasta la fecha 1628 de estas [1] donde las más influyentes son *Bitcoin, Ripple, LiteCoin, Dash Coin, Ether, Stellar*, entre otras [7, 12, 21]. No obstante, este no es el único uso que se le puede dar a esta tecnología ya que su manera de distribuir los datos permite distintos usos, es así como *Storj* [25] la utiliza para el almacenamiento de información P2P, incrementando de esta forma la seguridad, privacidad y control de estos datos.

Otra solución que se puede encontrar hoy en día es *Blockstack* [2], el cual propone un *framework* para crear una red para aplicaciones centralizadas, así los usuarios no necesitan confiar en servidores remotos, el cual se divide en tres capas, *blockchain*, red P2P y de almacenamiento, cambiando así hasta los protocolos utilizados dentro de esta red. Por último, cabe mencionar que se encuentran desarrollos en el área de la salud, como lo hace *MedRec* [5], el cual mantiene información descentralizada de registro médicos electrónicos, entregando acceso, desde cualquier lugar y momento, a esta información, permitiendo administrar la autenticación, confidencialidad, responsabilidad e intercambio de información en la red utilizando *blockchain*.

Dado esto, se encuentra latente la necesidad de poder realizar análisis y procesamiento logrando el menor costo posible y de forma descentralizada. Así, esta tesis se enfocará en utilizar un algoritmo ya propuesto en [4] pero adaptado a una arquitectura distribuida P2P de bajo nivel a través de la tecnología de *blockchain* usando la la herramienta de código libre *Ethereum*, y, con esto, analizar el procesamiento de los datos entre un conjunto de dispositivos de bajo consumo energético, como son las conocidas *Raspberry*.

3 Descripción del Algoritmo

El algoritmo propuesto en [4], busca entrenar un modelo de ML utilizando Dataset distribuidos. De esta forma se mantiene la distribución de cada uno de los conjuntos de datos, no se exponen estos ya que son procesados por cada uno de sus nodos y reduce el tráfico y tiempo de comunicación ya que solo se envían los parámetros necesarios para hacer la predicción.

La propuesta se divide en tres fases. La primera consta de el proceso de entrenamiento local que efectuará cada nodo, la siguiente se enfocará en la comunicación y traspaso de parámetros entre los nodos así como la generación de vecindades a través de un proceso de *clustering*; finalmente se efectuará la predicción que dará el resultado final utilizando las predicciones generadas por los modelos que pertenezcan a la misma vecindad. Cabe destacar que el proceso de entrenamiento local puede ser efectuado por cualquier tipo de aprendizaje automatizado, así como la generación de vecindad puede utilizar distintos tipos de algoritmos que permitan esto.

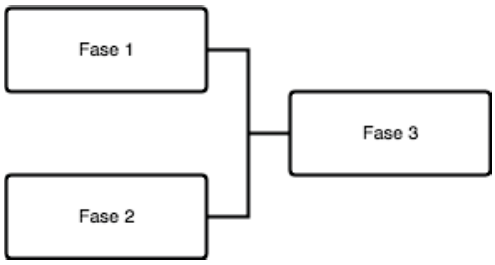


Figura 3.1 Flowchart completo

Para comenzar, se tendrá k nodos denominados N_k, donde cada uno de ellos tendrá su propio conjunto de datos D_k. Estos conjuntos tendrán *n-dimensiones* las cuales poseen las mismas características, no obstante, como cada nodo posee su propio conjunto, estos pueden tener distintas distribuciones, es por esto que es importante mantenerlos separados en sus respectivos nodos. A continuación, se puede realizar el proceso de extracción de características de distribución de sus datos a través de los momentos estadísticos o con histogramas.

Para obtener los momentos estadísticos se utiliza el generador de momentos estadísticos, el cual hace innecesaria la comunicación de los nodos para obtener este tipo de descripción de distribución de los datos. En cambio, para obtener el histograma es necesario buscar los mínimos y máximos de todas las *n-dimensiones* entre todos los nodos, obteniendo así los vectores *n-minimo* y *n-maximo*. Estos vectores serán compartidos con el resto de los nodos y así cada uno tendrá los valores globales de los mínimos y máximos para cada una de las características.

Una vez obtenido este vector con los valores globales, se utilizarán para crear un histograma por cada característica en cada uno de los nodos, así se tendrá *k-histogramas*. Cada dimensión estará dividida por el número de tramos, por lo que cada histograma tendrá las mismas dimensiones y el vector generado tendrá una dimensión de n^d donde n es el número de tramos y d es la cantidad características que posee el conjunto de datos. Estos histogramas representarán la frecuencia de los valores que se encuentran en los conjuntos de datos, estos utilizarán

como valores límites los *n-minimo* y *n-maximo* globales obtenidos anteriormente y se dividirá en una cantidad de tramos definidas por el usuario. La cantidad de tramos a dividir el histograma influirá de manera significativa en la precisión y costo que posea el algoritmo.

Es posible utilizar un algoritmo de agrupación para definir las vecindades tales como *Hierarchical Clustering algorithm* y *Kmean algorithm* junto a la medida de las distancias las como *Manhattan distance* y *Euclidian distance*. Una vez completado esto, se podrá pasar a la fase de predicción. Cada nodo debe entrenar de manera local su modelo de aprendizaje, y luego compartir los parámetros de este al resto de los nodos, de esta forma cada uno de los nodos tendrá el modelo propio y de los demás nodos. De esta forma, cuando un nuevo dato al cual se le deba generar una predicción entre a alguno de los nodos, este nodo podrá efectuar la predicción con su modelo y el de todos los nodos que pertenezcan a su vecindad, generando así n diferentes *outputs*.

Finalmente, para generar la predicción final, se utilizarán estos n *outputs* como datos de entrada para para obtener el promedio diferenciado por pesos de estos y así llegar a la respuesta final.

A continuación se explicará cada una de estás fases, así como el algoritmo propuesto.

3.1 Fase 1: Entrenamiento Local

Con k conjunto de datos distribuidos, en cada nodo N_i con $i = 1, ...k$, se utiliza un algoritmo de aprendizaje para generar un modelo. Como se mencionó anteriormente, se puede utilizar cualquier tipo de aprendizaje que no sea del tipo *lazzy training* debido a que no se debe compartir el *dataset*. Los parámetros de estos modelos serán compartidos con los demás nodos y además se considerará que el vector de entrada será de n dimensiones que representarán las características de la entrada x_i y se obtendrá una respuesta $\hat{y}_i$ del modelo creado.

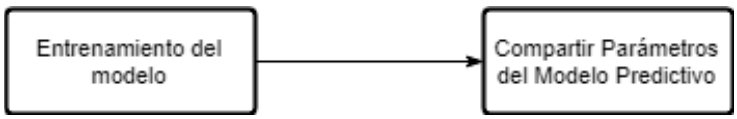


Figura 3.2 Flowchart de la Fase 1

3.2 Fase 2: Modelo y Transmisión de información

Para que los nodos puedan compartir una descripción de sus datos, existen dos alternativas propuestas. En el primer caso, cada nodo N_i recibe el máximo y mínimo de cada característica de todos los nodos, guardando así los valores globales. Con estos límites cada nodo generará un histograma de n-dimensiones para luego ser compartidos entre todos los nodos. La otra opción consiste en que se pueden obtener los momentos estadísticos de cada nodo para luego compartir-

los entre ellos, de esta forma los k-momentos estadísticos generados, podrán describir de manera precisa la distribución del *Dataset*.

Se calcula el vector de distancia con respecto a cada uno de los nodos generando así un vector con las k distancias. Utilizando un algoritmo de *clustering*, se generan los vecindarios de los nodos creando así un vector binario h_i con k dimensiones, donde los índices que posean un valor igual a 1 serán los nodos que poseen una distribución similar correspondiente a su vecindario.

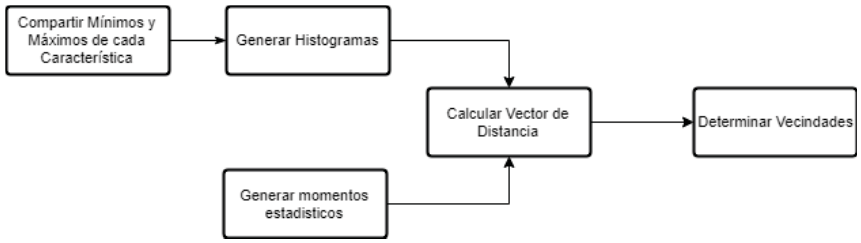


Figura 3.3 Flowchart de la Fase 2

3.3 Fase 3: Generación de la predicción

La fase de predicción consiste en tomar las predicciones hechas por los diferentes modelos, ponderar cada una de estas mediante las distancias que posea entre ellas y el nodo al cual se le este solicitando la predicción, y, por último, sumar estas ponderaciones para obtener un resultado final, siempre respetando y entregando una mayor ponderación a la predicción obtenida por el nodo local.

Siendo el nodo N_j al que se requiere predecir un valor, y y_j es la predicción entregada por el modelo local. Además, como ya que cada nodo N_i posee una copia de los modelos y parámetros del resto de los nodos, se obtienen todos los resultados y_i de cada uno de los modelos que pertenecen a la vecindad del nodo N_j. Estos resultados se ponderan a través de la fórmula $\hat{d}_{ij} = 1 - d_{ij}/\sum d_{ij}$, obteniendo así un vector con las distancias entre el nodo local con los nodos vecinos. Por último, el valor final está dado por $y = c * y_j + (1-c) * \sum \hat{d}_{ij} * y_i$, donde se le da una ponderación de c al nodo local y $(1-c)$ a las predicciones entregadas por los otros modelos.

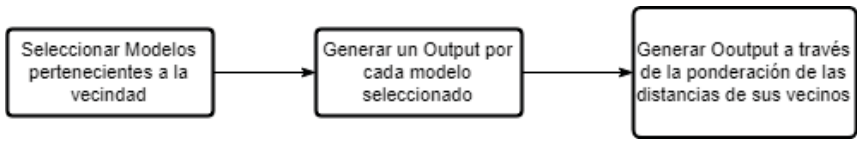


Figura 3.4 Flowchart de la Fase 3

4 Arquitectura

A continuación se presentará una descripción y representación de la arquitectura propuesta de la solución. Está se representará en tres maneras, una lógica a nivel de nodo, otra lógica a nivel de *cluster* y por último una arquitectura física para mantener una alta disponibilidad.

4.1 Nodo

Un nodo, el cual es una instancia de la solución, está compuesto por dos componentes principales, una *API REST*, la cual cumple la función de escuchar mensajes enviados por el protocolo *HTTP*, los cuales realizarán llamadas a funciones relacionadas con el algoritmos propuestos y una instancia de *Ethereum*, plataforma encargada del manejo de *BlockChain*.

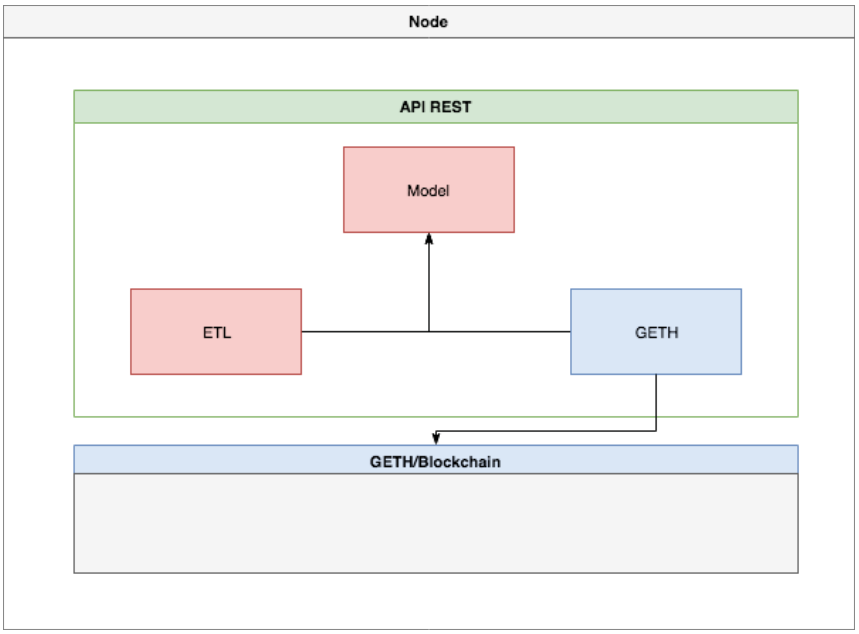


Figura 4.1 Arquitectura Lógica de un Nodo

4.1.1 API REST

La API REST contendrá 3 módulos, que se relacionarán entre ellos, no obstante, solo *ETL* y *Models* tendrán sus métodos expuestos por *HTTP*. A continuación se listaran y se describirá cada uno de ellos.

- ETL: Encargado del preprocesamiento de los datos, separación entre datos de entrenamiento y de prueba, además de obtener los momentos estadísticos de cada uno de los *Dataset*.

- GETH: Módulo encargado de la conexión y comunicación con la plataforma de *Ethereum*.
- Models: Encargado del entrenamiento y predicción, es una clase abstracta que puede utilizar cualquier tipo de modelo. Es quien recopila información obtenida por los otros módulos para hacer la predicción final.

4.1.2 GETH

Como se habló anteriormente, *GETH* es la plataforma encargada de manejar la cadena de bloques. Es está la que establece las conexiones P2P y permite las transacciones. Para que funcione adecuadamente, cada nodo debe poseer su propia cuenta en dicha plataforma, la cadena de bloque debe partir del mismo bloque y estar pareado con el resto de los nodos. Es necesario que esta plataforma esté corriendo en segundo plano y que este adecuadamente configurada ya que es solo un medio para que la *API REST* efectúe las transacciones y obtenga los datos desde la cadena.

4.2 Comunicación entre Nodos

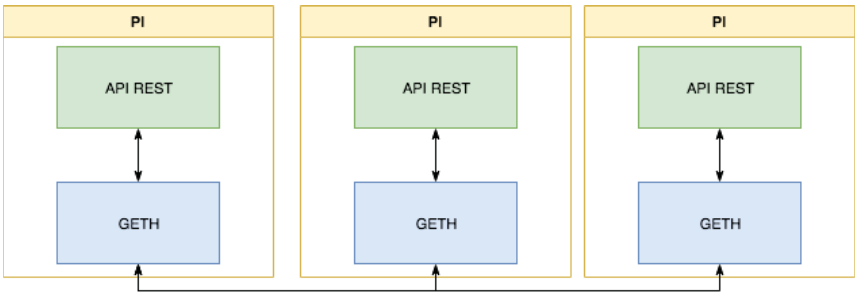


Figura 4.2 Comunicación entre nodos

La figura 4.2 muestra como es la comunicación entre cada uno de los nodos. La *API REST* expone sus métodos *HTTP*, por lo cual pueden ser accedido y consultados desde cualquier interfaz que posea acceso. No obstante, los parámetros de dicho nodo son almacenados en la cadena de bloque, la cual se distribuye entre todos los nodos que pertenezcan a la red a través de *GETH*.

Es acá donde, a través de los *SmartContact*, es posible controlar el acceso a los datos almacenados en la cadena, así como también la modificación y trazabilidad de estos. Es así como se permite que cada nodo pueda funcionar de manera autónoma y al mismo tiempo se puede nutrir con la información que esté siendo compartida por los distintos participantes de la red, para así mejorar la predicción que pueda entregar el modelo local.

5 Implementación

En este apartado, se verá la implementación actual que se desarrolló para efectuar las primeras pruebas de concepto, se hablará de las distintas herramientas utilizadas, así como también de los métodos utilizados en el *Smartcontract* y las interacciones que tiene con la *API REST*. A través de esto se podrá apreciar la convergencia de todas las tecnologías para lograr establecer una nueva forma de realizar DML.

5.1 Herramientas

5.1.1 API REST

Para el desarrollo de la *API*, se utilizó *django 2*, el cual es un *Framework* de *Python 3* para la realización de aplicaciones web, junto a este se utiliza *django REST Framework 3*, el cual proporciona clases y métodos para el desarrollo de una *API REST* en *django*, entregando de esta manera todas las herramientas necesarias para la construcción de esta solución.

Para el desarrollo de los modelos de aprendizaje, procesamiento de datos y cálculos estadísticos se están utilizando las bibliotecas *sklearn, scipy, numpy y pandas*, las cuales proporcionan las herramientas necesarias para hacer todo lo antes mencionado, facilitando así el desarrollo y permitiendo concentrar el esfuerzo en la convergencia de las distintas tecnologías.

5.1.2 Blockchain

Como se mencionó anteriormente, la plataforma utilizada para la creación y manejo de la cadena de bloques fue *Go-Ethereum (GETH)*, una implementación de *Ethereum* utilizando el lenguaje de programación *Go* el cual permite la creación de un nodo génesis, la inicialización de la cadena de bloques y la conexión a otros nodos dentro de la red, permitiendo así utilizar esta herramienta directamente para mantener los nodos conectados.

Para comenzar, la creación del nodo génesis fue establecida como POA ya que es una red privada donde solo los nodos previamente establecidos podrán aceptar las transacciones realizadas por el resto. Además, se establece una serie de nodos los cuales estarán habilitados para realizar transacciones dentro de la red, esto con el fin de mantener un ambiente controlado y así poder realizar los estudios pertinentes.

Ya que es una red P2P, las conexiones entre los nodos son bidireccionales y puede poseer la estructura que se estime conveniente. Los nodos encargados de la generación de bloques estarán conectados entre ellos y todo el resto de los nodos encargados de entrenar modelos estarán conectados a estos, de esta manera se obtiene una arquitectura robusta para la creación

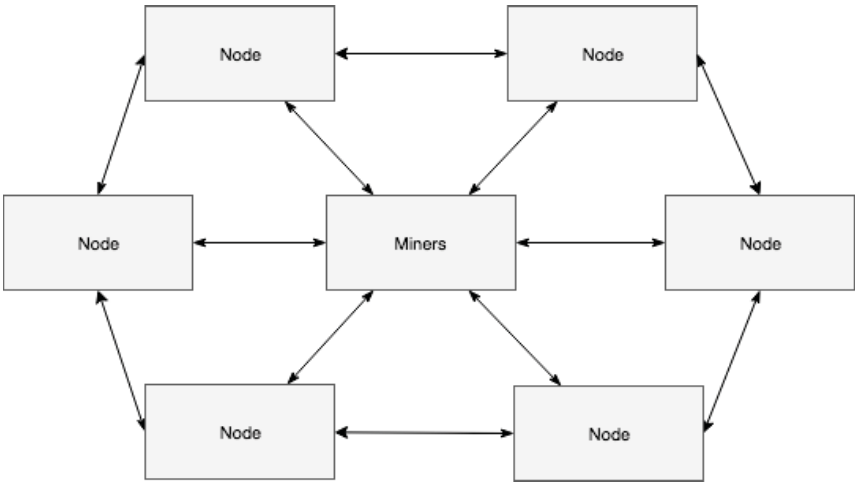


Figura 5.1 Estructura P2P

y propagación de los bloques.

5.2 Smartcontract

Ya que el algoritmo propuesto necesita compartir la descripción de distribución del *Dataset*, debido a que con esto puede saber quienes son sus vecinos, se decidió implementar un *SmartContract* que pudiera almacenar esa información. Así también, es necesario destacar que se ha establecido ciertas restricciones necesarias para respetar los objetivos propuestos de esta tesis, es por esto que solo los nodos dueños de un *dataset* pueden modificar sus valores dentro del *SmartContract*, no obstante, estos pueden ver los parámetros de todos los nodos. De esta forma, los nodos solo pueden modificar su propia información en la cadena de bloques y al momento de hacer una predicción, podrán consultar los parámetros de los demás nodos para mejorar la predicción.

5.2.1 Métodos del SmartContract

A continuación se explicarán los métodos mas importantes que contiene el contrato utilizado para la implementación del proyecto. Cabe destacar que debido a que los contratos no almacenar valores decimales, se guardan dos tipos de valores enteros, donde se puede tener el valor final de la manera $decimal = valor1 * 10^{valor2}$.

- **constructor**: Encargado de definir los valores iniciales para la construcción del contrato, los cuales son el nombre del contrato, cantidad de características y momentos estadísticos que se extraerán del *dataset*. Como resultado cada cuenta solo podrá modificar solo su propia información, es decir, momentos estadísticos y una variable de tipo bytes con los

valores necesario del modelo entrenado. Además, permite la modificación de los valores iniciales del contrato solo por la cuenta que creó este.

- **setNode**: Recibe un arreglo de los momentos estadísticos que corresponden al $valor1$, un arreglo de los momentos estadísticos que corresponden al $valor2$ y una variable en bytes que corresponde a los parámetros del modelo del nodo que está realizando la llamada a la función. Cabe destacar que no se guardarán la información del nodo si es que el tamaño de los arreglos no corresponden al valor $count\_features * count\_moments$.
- **getNodeStatisticsMoments**: Encargado de obtener los momentos estadísticos de un nodo específico, el cual entregará dos arreglos de tamaño $count\_features * count\_moments$. Por lo que el primer arreglo será el $valor1$ y el segundo arreglo el $valor2$, de acuerdo a la fórmula $decimal = valor1 * 10^{valor2}$, el cual corresponde a los momentos estadísticos de cada nodo.
- **getNodeModel**: Encargado de obtener los parámetros del modelo de un nodo específico el cual estará en bytes.
- **getNodesList**: Encargado de obtener la lista de nodos que ya se han guardado sus parámetros en el contrato, cabe destacar que es un arreglo de las cuentas que utiliza cada nodo.

5.3 API REST

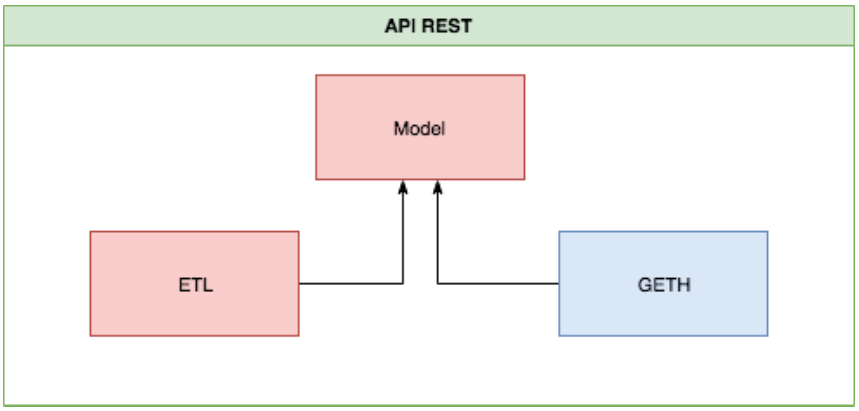


Figura 5.2 Estructura API

Para trabajar la *API REST* se definieron métodos para cada uno de los modelos descritos en la arquitectura, a continuación se especificará cada uno de ellos y en que consiste.

5.3.1 ETL

Modulo encargado del pre-procesamiento de los datos, una vez inicializada la *API*, lee el *dataset*, obtiene los momentos estadísticos y lo separa en un conjunto de entrenamiento y otro de pruebas para que puedan ser utilizados por los modelos de aprendizaje

- **GET**: Método que retorna una descripción detallada del *Dataset*.

5.3.2 GETH

Módulo que se comunica con la plataforma *GETH*, establece la conexión con este y se configura para poder utilizar los *SmartContract* así como también para poder hacer transacciones hacia este utilizando la información del nodo local.

- **Node**: Clase encargada de la conexión con *GETH*, incluye métodos para comunicarse con los *smartcontract* y hacer las transacciones necesarias para obtener y enviar parámetros a este.
- **GET**: Método que proporciona información sobre la cadena de bloques a la cual se está conectada.

5.4 Models

Módulo que entrena el modelo y efectúa las predicciones, la información obtenida por los otros dos módulos converge en este y es quien se encarga de hacer las predicciones finales. Debido a que la predicción final utilizando la suma ponderada puede mezclar las predicciones de cualquier modelo, esta clase podría utilizar cualquier tipo de aprendizaje para entrenar y hacer su predicción local.

- **Model**: Clase que almacena la información del modelo local, contiene las métodos para entrenar y predecir.
- **GET**: Entrega información detallada sobre el modelo actual.
- **POST**: Recibe datos a predecir y efectúa la predicción. Se comunica con la cadena de bloques para determinar la vecindades y procede a hacer la suma ponderada entregando una predicción final.
- **PUT**: Método encargado de volver a entrenar el modelo, una vez tiene los nuevos parámetros del modelo, los comparte con la cadena de bloques para que así el resto de los nodos tengan acceso a su información.

6 Experimentación

En este apartado se realizará una comparación entre la predicción del modelo local con el ensamblado. Para esto se tomará en cuenta parámetros como la cantidad de datos, cantidad de vecindades, cantidad de nodos necesarios para que el ensamblado funcione de forma más eficiente.

Para los experimentos se utilizó solo el modelo *MLP Regression* para simplificar el problema. Además, se utilizó como clustering el algoritmo *KMean* y para la distancia *Euclidian distance*, los cuales corresponden a la Fase 3.

Se utilizarán datos sintéticos donde se generarán 1,000, 10,000 y 100,000 datos en cada nodos, dividiendo estos en 80 % de entrenamiento y 20 % para las pruebas. La distribución de los datos serán definidos por las funciones (1, 2, 3, 4) descritas en [3].

$$y_1 = \frac{\sin X_1 * \sin x_2}{x_1 * x_2} + \varepsilon \tag{1}$$

$$y2 = 0{,}01 * x_1 + 0{,}02 * x_2 + 0{,}9 * x_3 + \varepsilon \tag{2}$$

$$y_3 = 10 * \sin \pi * x_1 * x_2 + 20 * (x_3 - 0{,}5)^2 + 10 * x_4 + 5 * x_5 + \varepsilon \tag{3}$$

$$y_4 = 0{,}79 + 1{,}27 * x_1 * x_2 + 1{,}56 * x_1 * x_4 + 3{,}42 * x_2 * x_5 + 2{,}06 * x_3 * x_4 * x_5 + \varepsilon \tag{4}$$

Para el calculo de las vecindades junto con la distancia que tienen entre ellas se utiliza KMean [29]. Esto permite agrupar los nodos que sean de semejante distribución para luego realizar el aprendizaje de ensamblado utilizando solo estos modelos. Por lo tanto, se realizan los experimentos con 2 y 3 vecindades debido a que se comprobara el funcionamiento de este sistema dentro de una red de 6.

El modelo de aprendizaje a utilizar dentro de estos experimentos es un *Multilayer Perceptron* [13] para regresiones. Cabe destacar que la solución propuesta permite utilizar distintos tipos de modelos de aprendizaje, pero el tiempo de entrenamiento varia entre cada uno de estos. Así, es conveniente utilizar el modelo antes mencionado debido a que tiene un bajo tiempo de entrenamiento, para así centrar el procesamiento en la precisión de la predicción obtenida en cada experimento.

6.1 Procedimiento

A continuación se presenta, a modo de resumen, el procedimiento que se realizará por cada una de las configuraciones definidas anteriormente, los cuales se realizaron para obtener los resultados expuestos más adelante:

1. Se realiza una combinación de pruebas, para encontrar la mejor configuración posible para la MLP para las determinadas funciones. Se determinó que la mejor configuración consiste en 1 capa oculta con 18 unidades ocultas. Adicionalmente, y para dificultar el aprendizaje del modelo, solo se le permitió a la red realizar 8 iteraciones.

2. Generar un nuevo conjunto de datos en cada uno de los nodos con las siguientes características

3. Separar el conjunto de datos en **80 %** para entrenamiento y **20 %** para las pruebas.

4. Entrenar cada modelo de *Multilayer Perceptron* con sus datos locales.

5. Subir parámetros (momentos estadísticos y coeficientes del modelo) a la cadena de bloques.

6. Almacenar la predicción con los datos de pruebas locales de cada nodo, utilizando su modelo local y el ensamblado.

6.2 Resultados

Para poder identificar los distintos nodos fueron identificados con nombres de Capitales, siendo 6 la cantidad de nodos identificados como Nairobi, Tokyo, Dublin, Denver, Moscu y Berlin. Los experimentos se repetirán 10 veces para cada una de las configuraciones propuestas, donde se promediaran los resultados de estas diez iteraciones con el fin de acercarse más a un problema real. Los resultados obtenidos, que están publicado en el ANEXO A y el ANEXO B, demuestran que en la mayoría de los experimentos existe una mejora en el aprendizaje distribuido como se ve en la figura 6.1

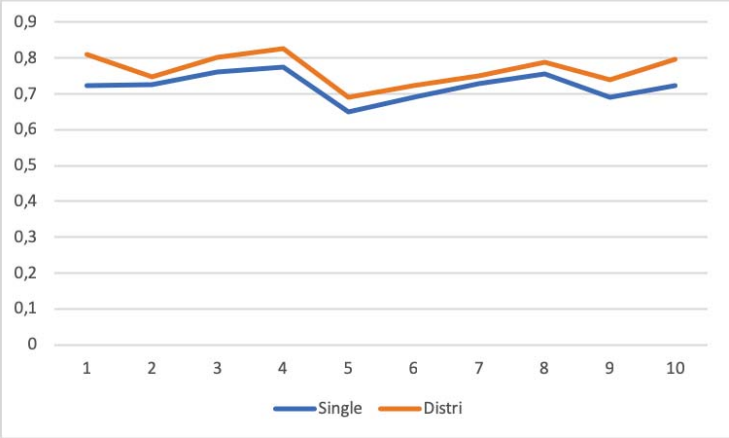


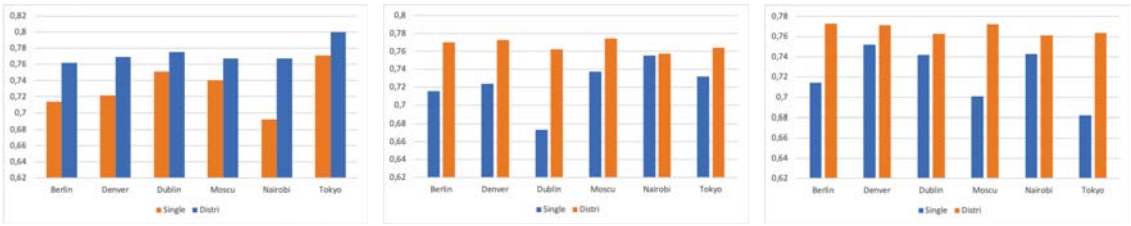
Figura 6.1 Score por iteración para 100.000 datos y función (4)

De esta manera, se pueden observar distintos puntos. Dentro de lo que se puede desprender a través de los resultados obtenidos es que el incremento de la mejora del modelo distribuido

sobre el modelo local se puede atribuir al conjunto de datos (fig: 6.2), los tipos de funciones generadoras utilizadas (fig: 6.3) y la cantidad de vecindades . A través de esta información se puede deducir que la mejora del aprendizaje distribuido sobre el modelo local está presente en cada uno de los casos pero la intensidad de está mejora se ve afectada por diferentes factores. Mientras más datos y más características se posean (Definidas por las funciones generadoras), la diferencia entre el modelo local y el aprendizaje distribuido se incrementará.

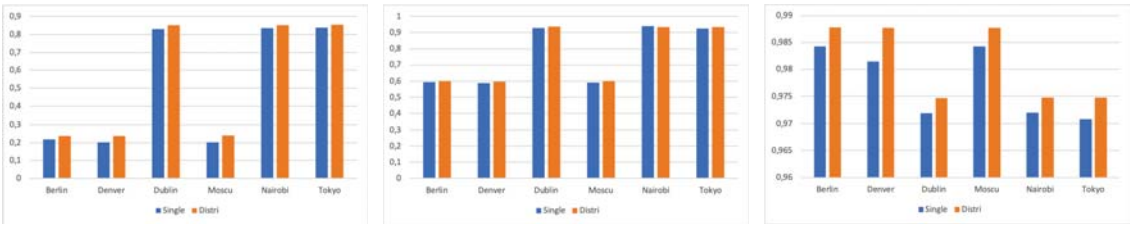
En cuanto a las vecindades, se efectuaron pruebas con distintos conjuntos de datos (1.000, 10.000, 100.000) para predecir con 2 y 3 vecindades, lo cual se puede apreciar en la figura 6.4 donde se aprecia que con es posible escalar a distintas vecindades, siempre y cuando se posean los nodos suficientes para alimentar el aprendizaje distribuido.

Por último, se determinó que hay un nivel de incidencia relacionada a la efectividad del modelo local, es decir, cuando el modelo local no sea suficientemente efectivo, el aprendizaje distribuido va a ser una buena herramienta para mejorar la precisión de los resultados, en cambio, cuando el modelo local entrega buenos resultados, el aprendizaje distribuido ayudará en menor medida a la mejora de la precisión de los resultados finales.



(a) R2 para 1.000 datos (b) R2 para 10.000 datos (c) R2 para 100.000 datos

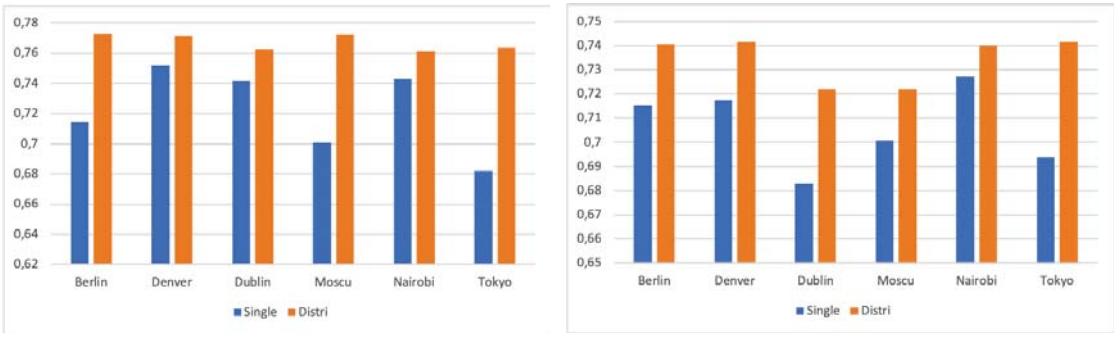
Figura 6.2 Comparación de R2 score para función (4) con 1.000, 10.000, 100.000 datos



(a) R2 para función (1) (b) R2 para función (2) (c) R2 para función (3)

Figura 6.3 Comparación de R2 score para funciones generadoras (1), (2), (3)

Otros resultados a destacar son los expuestos en el ANEXO C, en el cual se exponen los resultados obtenido realizando 8 iteraciones Tabla 25 y 100 iteraciones Tabla 26 para el entrenamiento de los modelos locales. Con estos resultados se puede confirmar, de otra manera, que existe una mejora en utilizando el entrenamiento distribuido cuando con 8 iteraciones, en cambio, no me mejoran con 100 iteraciones. Esto es debido a que el *machine learning* distribuido es utilizado para cuando el entrenamiento local no es suficientemente bueno. Es decir, si con el



(a) R2 para 2 vecindades (b) R2 para 3 vecindades

Figura 6.4 Comparación de R2 Score para 2 y 3 vecindades con 100.000 datos y función (4)

entrenamiento local se llega a una alta precisión, no será muy necesario utilizar el método de aprendizaje ensamblado.

7 Conclusiones

Dentro de este documento, se presentó una implementación de un sistema que permite y facilita el aprendizaje distribuido, utilizando una arquitectura de *Blockchain*. donde hay que resaltar que lo propuesto fue implementado utilizando equipos de bajo consumo energético, logrando realizar un aprendizaje utilizando los modelos locales, pero mejorando sus predicciones mediante los datos almacenados en la cadena de bloques. Es así coma lo convergencia de estas tecnologías abre puertas para seguir investigando posibles mejoras que saquen el mejor provecho de ambas ramas.

Como se pudo observar en los resultados obtenidos, destacando una mejora sustancial cuando los modelos locales no son suficientemente eficaces, el algoritmo propuesto ayuda a mejorar los resultados finales al utilizar la convergencia de distintos modelos y entrenamientos, logrando una colaboración entre estos pero respetando siempre la integridad y confidencialidad de los datos pertenecientes a cada nodo. Es así como un aprendizaje local realizado por una maquina podrá entregar mejores predicciones al utilizar los demás modelos que fueron entrenados con diferentes conjuntos de datos, siempre y cuando estos posean una distribución similar.

Por otro lado, el estudio que se hizo sobre *Blockchain* permitió entender que esta estructura facilita el almacenamiento de los datos de manera distribuida y permite validar las transacciones hechas por los distintos participantes de la red logrando así mantener una consistencia de los datos. Es así mismo que la plataforma *Ethereum* monta una arquitectura distribuida privada, donde los nodos participantes pueden realizar transacciones y, de esta forma, compartir sus parámetros a la cadena de bloques, haciendo que estos valores puedan ser accedidos por cualquier usuario que pertenece a la red. Además, el uso de *SmartContract* es un medio para poder realizar interacciones entre los nodos, estableciendo reglas y funciones que no pueden ser alteradas por estos. De esta forma se construyó una red que valida las transacciones, manteniendo la consistencia de los datos y ayuda a conservar la privacidad de la información, limitando la interacción que se tiene entre los nodos ya que estos solo se comunican con la cadena de bloques. Por último, la plataforma es sencilla de acceder, gracias a que *Ethereum* entrega las herramientas necesarias tanto para constituir la cadena de bloques, establecer la conexión P2P entre los nodos y manejar las transacciones que se efectúen en esta red.

Realizando la implementación propuesta, fue posible combinar dos tecnologías que no han sido utilizadas en conjunto anteriormente, demostrando que este enfoque no solo es un concepto si no que puede ser aplicado en los desarrollos e investigaciones del DML y, junto con esto, abrir nuevas posibles futuras mejoras tanto en la implementación como en la predicción de esta disciplina.

Ya que se demostró con esta prueba de concepto que se puede hacer DML usando una arquitectura basada en *Blockchain* utilizando sistemas de bajo consumo energético, como trabajo futuro queda la experimentación, agregar más métodos de aprendizaje para el entrenamiento de los modelos locales, agregar más métodos para la Fase 3 como regresión lineal y redes neuronales, y finalmente pulir las interacciones que se tiene entre ML y la cadena de bloques. Por otro

lado el desarrollo de una plataforma que permita administrar lo expuesto con una interfaz más amigable para los usuarios, respetando siempre los objetivos de este documento, es decir, que sea accesible a computadores de bajo consumo y que se respete siempre el contexto de los datos con la privacidad de estos.

Referencias

[1] Coin Market Cap cryptocurrency market cap rankings, charts, and more. Accessed: 2018-06-12.

[2] Muneeb Ali, Ryan Shea, Jude C. Nelson, and Michael J. Freedman. Blockstack: A new internet for decentralized applications. 2017.

[3] Héctor Allende-Cid. *Distributed Machine Learning with Context-Awareness for the Regression Task*, pages 305–322. Springer International Publishing, Cham, 2017.

[4] Héctor Allende-Cid, Héctor Allende, Raúl Monge, and Claudio Moraga. Discrete neighborhood representations and modified stacked generalization methods for distributed regression. *J. UCS*, 21(6):842–855, 2015.

[5] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman. Medrec: Using blockchain for medical data access and permission management. In *2016 2nd International Conference on Open and Big Data (OBD)*, pages 25–30, Aug 2016.

[6] Maria-Florina F Balcan, Steven Ehrlich, and Yingyu Liang. Distributed k-means and k-median clustering on general topologies. In *Advances in Neural Information Processing Systems*, pages 1995–2003, 2013.

[7] HolotescuAleksander Berentsen and Fabian Schar. A short introduction to the world of cryptocurrencies. *Federal Reserve Bank of St. Louis Review*, pages 1 – 16, 01 2018.

[8] Doina Caragea, Adrian Silvescu, and Vasant Honavar. Analysis and synthesis of agents that learn from distributed dynamic data sources. In *Emergent neural computational architectures based on neuroscience*, pages 547–559. Springer, 2001.

[9] Nitesh V Chawla, Lawrence O Hall, Kevin W Bowyer, and W Philip Kegelmeyer. Learning ensembles from bites: A scalable and accurate approach. *Journal of Machine Learning Research*, 5(Apr):421–451, 2004.

[10] Ittay Eyal, Idit Keidar, and Raphael Rom. Distributed data clustering in sensor networks. *Distributed computing*, 24(5):207–222, 2011.

[11] John Gantz and David Reinsel. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. *IDC iView: IDC Analyze the future*, 2007(2012):1–16, 2012.

[12] Sufian Hameed and Sameet Farooq. The art of crypto currencies. *International Journal of Advanced Computer Science and Applications*, 7(12), 2016.

[13] Simon S Haykin, Simon S Haykin, Simon S Haykin, and Simon S Haykin. *Neural networks and learning machines*, volume 3. Pearson Upper Saddle River, 2009.

[14] Carmen Holotescu. Understanding blockchain technology and how to get involved. 04 2018.

[15] Dino Ienco, Albert Bifet, Indrė Žliobaitė, and Bernhard Pfahringer. Clustering based active learning for evolving data streams. In *International Conference on Discovery Science*, pages 79–93. Springer, 2013.

[16] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[17] Andreas Lattner, Alexander Grimme, and Ingo Timm. An evaluation of meta learning and distribution strategies in distributed machine learning, 01 2010.

[18] Aleksandar Lazarevic and Zoran Obradovic. The distributed boosting algorithm. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 311–316. ACM, 2001.

[19] Loi Luu, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena, and Aquinas Hobor. Making smart contracts smarter. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 254–269, New York, NY, USA, 2016. ACM.

[20] Christopher Moretti, Karsten Steinhaeuser, Douglas Thain, and Nitesh V Chawla. Scaling up classifiers to cloud computers. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 472–481. IEEE, 2008.

[21] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 03 2009.

[22] Diego Peteiro-Barral and Bertha Guijarro-Berdiñas. A survey of methods for distributed machine learning. *Progress in Artificial Intelligence*, 2(1):1–11, 2013.

[23] Miguel Rodríguez, Diego M. Escalante, and Antonio Peregrín. Efficient distributed genetic algorithm for rule extraction. *Applied Soft Computing*, 11(1):733 – 743, 2011.

[24] Ognjen Rudovic, Vladimir Pavlovic, and Maja Pantic. Context-sensitive dynamic ordinal regression for intensity estimation of facial action units. *IEEE transactions on pattern analysis and machine intelligence*, 37(5):944–958, 2015.

[25] S. J 3. Wilkinson. Storj a peer-to-peer cloud storage network. 2016.

[26] Rüdiger Wirth, Michael Borth, and Jochen Hipp. When distribution is part of the semantics: A new problem class for distributed knowledge discovery. In *Proceedings of the pkdd 2001 workshop on ubiquitous data mining for mobile and distributed environments*, pages 56–64, 2001.

[27] David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241 – 259, 1992.

[28] Yan Xing, Michael G Madden, Jim Duggan, and Gerard J Lyons. Context-based distributed regression in virtual organizations. In *Parallel and Distributed computing for Machine Learning. 7th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 03), Cavtat-Dubrovnik, Croatia*, 2003.

[29] Jyoti Yadav and Monika Sharma. A review of k-mean algorithm. *International Journal of Engineering Trends and Technology (IJETT)–Volume*, 4, 2013.

Anexos

A: Resultados con 2 clusters

Tabla 1 single MSE con 1000 datos

Iteracion	Berlin	Denver	Dublin	Moscu	Nairobi	Tokyo
1	140158.2247	170582.1153	92.04268507	99448.8244	450.8302363	406.3404521
2	167046.4222	160973.2269	445.8957113	95228.76592	415.4575495	188.3640757
3	101111.3298	217073.9871	372.8890146	143779.8146	442.436698	554.0858169
4	122601.5002	79887.89114	564.6218782	117004.5904	576.2643537	107.0571233
5	86806.67034	93982.61276	432.025434	154108.0652	221.4724512	259.3599804
6	178158.3465	155821.0962	445.149954	105511.7907	555.950349	496.6010671
7	173374.7907	148058.0048	173.2728406	145349.0958	545.7347165	158.8222813
8	151787.91	119975.4454	182.2148564	90451.72435	212.5895661	330.3781557
9	190738.1418	185709.6238	321.1770314	165214.96	341.4805172	115.9876654
10	55914.29863	125037.0252	227.7162367	265515.1009	379.8890616	536.2111233
Final	136769.76+-44253.58	145710.1+-41833.05	325.7+-151.79	138161.27+-52089.75	414.21+-129.12	315.32+-174.49

Tabla 2 distribuido MSE con 1000 datos

Iteracion	Berlin	Denver	Dublin	Moscu	Nairobi	Tokyo
1	122813.0929	162513.3118	220.2042985	85992.71789	304.9482825	220.0177316
2	83157.413	158208.5827	387.7229645	89885.47464	372.1174152	221.4835278
3	139384.6258	122760.6254	321.6034704	150722.5093	359.9093048	476.2733591
4	95756.19206	107369.123	264.8274673	99273.32969	385.8480309	288.6858644
5	45680.92322	75357.91025	290.7108178	72379.48825	321.5220312	265.7931308
6	150661.2162	127258.0855	668.7901125	119470.0168	290.2825014	189.3623726
7	149867.7964	131258.8782	203.3148075	145899.3667	261.0640214	212.534318
8	129086.1611	90922.25385	195.787254	104263.6815	165.199706	343.8483343
9	165872.7796	160704.9975	250.4061044	156601.7557	312.2182497	147.5150344
10	76791.96151	91667.72656	311.2816924	191313.7187	329.6592318	454.2661503
Final	115907.22+-38831.02	122802.15+-31328.73	311.46+-138.78	121580.21+-38040.35	310.28+-63.6	281.98+-110.83

Tabla 3 single R2 con 1000 datos

Iteracion	Berlin	Denver	Dublin	Moscu	Nairobi	Tokyo
1	0.65836775	0.751153925	0.923792306	0.721642792	0.65633134	0.54370986
2	0.602227116	0.738307931	0.656453942	0.697656747	0.736658986	0.868827324
3	0.811796745	0.51842133	0.713104864	0.738903665	0.607129547	0.67707522
4	0.656474881	0.83246566	0.49657312	0.756540393	0.594598884	0.925957619
5	0.748350986	0.794411278	0.614680446	0.73976415	0.83832973	0.80170762
6	0.679910458	0.663897409	0.837979297	0.770357982	0.666812173	0.644698188
7	0.705821091	0.760617953	0.850377045	0.768925562	0.608429292	0.86981011
8	0.728633876	0.734055441	0.899550667	0.823881596	0.817561624	0.834061996
9	0.673933459	0.710676273	0.73094246	0.749996247	0.807840052	0.875848468
10	0.868570773	0.70998446	0.789747896	0.63371766	0.591724986	0.67083214
Final	71.3409+-7.9287	72.1399+-8.5172	75.132+-13.5384	74.0139+-5.0105	69.2542+-9.9009	77.1253+-12.7247

Tabla 4 distribuido R2 con 1000 datos

Iteracion	Berlin	Denver	Dublin	Moscu	Nairobi	Tokyo
1	0.700646085	0.762924737	0.817679572	0.759306427	0.767537403	0.752936433
2	0.801984601	0.742802377	0.701273879	0.714621244	0.764130469	0.845763652
3	0.740556865	0.727655536	0.752563181	0.726296109	0.680411385	0.722424822
4	0.731694496	0.774834272	0.763875133	0.793435063	0.728556484	0.800340339
5	0.867572858	0.835153162	0.740717667	0.877775783	0.765295623	0.796789187
6	0.729313385	0.725507179	0.756581253	0.739978485	0.826030156	0.864517419
7	0.745707293	0.787778993	0.824435485	0.768050747	0.812683671	0.825781248
8	0.769220017	0.798456438	0.89206863	0.796988357	0.858230267	0.827296371
9	0.716440755	0.749631883	0.790228927	0.763029772	0.824306689	0.842102024
10	0.819496831	0.787382456	0.712591285	0.736079657	0.645708073	0.721136302
Final	76.2263+-5.2338	76.9213+-3.4428	77.5202+-5.7421	76.7556+-4.7175	76.7289+-6.7393	79.9909+-5.1534

Tabla 5 single MSE con 10000 datos

Iteracion	Berlin	Denver	Dublin	Moscu	Nairobi	Tokyo
1	144342.6654	13685.6605	707.1284083	159197.0132	211.7638131	620.6830939
2	110988.1305	133624.1171	383.5221728	143208.2866	230.3661861	313.3753163
3	198122.0825	170983.428	470.9692222	28848.46217	249.6525045	641.476628
4	228199.1823	179940.536	311.1257787	151405.857	265.7103112	623.0733545
5	30651.88893	132228.9233	472.3917819	116003.8208	556.2267937	121.954442
6	183770.464	166028.1497	480.2008023	171125.3411	578.8964157	527.064313
7	166757.2853	150794.603	594.3761505	80112.13004	521.9569192	253.925006
8	80901.65016	151908.0667	459.7141421	188001.7082	299.0008447	239.4872531
9	173823.8146	186541.0886	330.4098871	141715.8685	206.224184	196.2675776
10	155744.0807	144203.6164	214.1187655	200885.6034	187.4827777	217.2113309
Final	147330.12+-58645.71	142993.82+-49036.81	442.4+-142.2	138050.41+-51608.45	330.73+-156.72	375.45+-203.78

Tabla 6 distribuido MSE con 10000 datos

Iteracion	Berlin	Denver	Dublin	Moscu	Nairobi	Tokyo
1	66387.56859	67056.61004	411.6663081	76912.95079	406.626519	461.6278587
2	116509.0132	109897.926	289.8128496	127407.4011	270.7859197	273.4383564
3	100796.7563	102512.5553	330.7809587	82136.08738	369.3444411	401.9182138
4	178221.1254	174052.2341	343.0258769	165696.5619	335.6781524	315.9042516
5	83633.90647	71839.32409	279.6993516	70696.69492	290.8895386	304.3162857
6	157583.9918	166685.2107	500.9679138	144511.6181	529.0413319	548.0175402
7	129784.3875	118362.1132	424.9519472	124754.3789	381.5630912	392.4376124
8	83111.04675	95553.13161	254.0954368	86513.60868	274.9373769	232.7272544
9	140912.4163	136310.7916	195.129126	139663.8349	185.6799054	176.8795466
10	132419.1017	137611.6493	182.4514115	153992.393	202.7700984	193.4429778
Final	118935.93+-35616.42	117988.15+-36159.43	321.26+-102.3	117228.55+-35077.9	324.73+-102.49	330.07+-119.97

Tabla 7 single R2 con 10000 datos

Iteracion	Berlin	Denver	Dublin	Moscu	Nairobi	Tokyo
1	0.722187319	0.974277913	0.451810236	0.717098535	0.823353313	0.565102183
2	0.766138302	0.722897813	0.735503658	0.740240547	0.825669603	0.769729328
3	0.6208934	0.669869043	0.609659829	0.934910108	0.809883762	0.554991697
4	0.585776084	0.656687803	0.765266991	0.700466446	0.800910854	0.528814646
5	0.944104958	0.725833059	0.642914476	0.765220609	0.58299938	0.905478689
6	0.627219088	0.677329663	0.638057892	0.628778969	0.570710508	0.649010557
7	0.702984581	0.727714875	0.600427246	0.848179629	0.628657252	0.818530778
8	0.834544833	0.718973147	0.671879552	0.633388423	0.800486201	0.825023659
9	0.651804118	0.633593052	0.780423389	0.729566612	0.844902623	0.856506425
10	0.700774499	0.729358419	0.837453674	0.677710854	0.86807718	0.845856164
Final	71.5643+-10.9058	72.3653+-9.4588	67.334+-11.1223	73.7556+-9.4329	75.5565+-11.413	73.1904+-14.2783

Tabla 8 distribuido R2 con 10000 datos

Iteracion	Berlin	Denver	Dublin	Moscu	Nairobi	Tokyo
1	0.872225524	0.87396765	0.68086241	0.863321641	0.660804996	0.676548387
2	0.754505319	0.772099855	0.80013036	0.768901105	0.795081832	0.799075324
3	0.807125409	0.802071064	0.725848123	0.814678889	0.718735544	0.721179331
4	0.676495544	0.667922214	0.741199534	0.672194451	0.748485949	0.761104442
5	0.847489964	0.851046449	0.788572551	0.856917412	0.781921476	0.764138363
6	0.680338707	0.676052686	0.622405082	0.686511936	0.607681308	0.635057115
7	0.7688379	0.786277213	0.714323632	0.763578173	0.728539499	0.719541808
8	0.830026308	0.823228638	0.81863967	0.831294669	0.816542991	0.82996271
9	0.717730719	0.732256194	0.870325333	0.733482464	0.860353593	0.870681247
10	0.745587942	0.741730234	0.861493659	0.752943586	0.857320211	0.862723356
Final	77.0036+-6.814	77.2665+-6.9047	76.238+-7.964	77.4382+-6.669	75.7547+-8.1824	76.4001+-7.8032

Tabla 9 single MSE con 100000 datos

Iteracion	Berlin	Denver	Dublin	Moscu	Nairobi	Tokyo
1	121088.3083	126088.1344	537.2707948	141203.2662	138.7994857	568.0075499
2	105728.7293	168647.8661	396.2021856	162035.8216	284.4819917	428.5341761
3	153620.5189	44522.42332	350.7942152	233576.4441	230.0062813	240.8384371
4	164466.1784	22340.37749	409.0862432	119102.7982	388.690233	246.4016158
5	161596.0497	183985.1536	396.6162572	263901.5527	331.7126929	517.2959357
6	199685.5846	169452.3796	247.0416254	117059.3005	481.2018538	482.4644189
7	110493.8808	118152.3564	362.0998788	125544.8923	443.8342389	538.0537094
8	121348.9405	144175.5962	154.4251316	144686.0779	461.3808967	273.819237
9	166947.8619	165001.7213	551.5889082	219307.4724	148.9078084	417.5053526
10	173164.4094	131292.9529	150.7148954	37794.08369	567.3136461	605.8322633
Final	147814.05+-31232.08	127365.9+-54135.2	355.58+-138.21	156421.17+-66523.37	347.63+-145.74	431.88+-135.82

Tabla 10 distribuido MSE con 100000 datos

Iteracion	Berlin	Denver	Dublin	Moscu	Nairobi	Tokyo
1	73344.09584	75144.5456	323.6423037	77506.10965	323.8575233	322.953186
2	136482.0546	136597.1966	335.3306945	138611.8009	323.7539219	323.3736169
3	118152.1913	118201.898	250.1582616	112034.4907	230.4039208	225.111646
4	71510.96629	72674.01644	280.2276962	74849.80226	296.3309997	267.8096365
5	169945.3279	175344.0775	372.0890714	177744.7105	377.3497134	378.063972
6	141410.7722	134582.7391	385.8888941	141667.4081	382.7772938	380.0272343
7	111789.1864	105843.7497	421.2732659	111570.4765	408.0326934	393.3379862
8	122108.5469	127747.3429	245.2655526	123186.7462	238.5163758	239.8249905
9	163424.8085	155161.463	294.192634	159240.9117	321.5197244	310.5978128
10	72082.09956	76243.30647	364.4856516	77242.0009	345.2010326	360.6979263
Final	118025+-36470.21	117754.03+-35224.74	327.26+-59.37	119365.45+-35751.19	324.77+-58.35	320.18+-59.88

Tabla 11 single R2 con 100000 datos

Iteracion	Berlin	Denver	Dublin	Moscu	Nairobi	Tokyo
1	0.760242974	0.756405363	0.600771103	0.730547428	0.899615118	0.5897907
2	0.798283557	0.668226756	0.72063462	0.690006947	0.78794326	0.682912174
3	0.714429138	0.91653723	0.746019511	0.543921172	0.826858664	0.816982354
4	0.676843976	0.957089068	0.70302462	0.775833181	0.726538614	0.812744618
5	0.676809436	0.640719749	0.7090144	0.492253551	0.757967375	0.619491551
6	0.620203576	0.66490573	0.817624907	0.77738423	0.63723447	0.632945691
7	0.792397032	0.768370401	0.74662377	0.768253557	0.678527945	0.613011605
8	0.757943095	0.72441398	0.887853575	0.71783569	0.64872792	0.792106667
9	0.69230475	0.677989934	0.596726263	0.584266036	0.895115181	0.703148888
10	0.655948362	0.745762129	0.891079128	0.928331599	0.570666274	0.555386995
Final	71.4541+-6.0294	75.2042+-10.6815	74.1937+-10.1854	70.0863+-12.9439	74.2919+-11.1127	68.1852+-9.6317

Tabla 12 distribuido R2 con 100000 datos

Iteracion	Berlin	Denver	Dublin	Moscu	Nairobi	Tokyo
1	0.854777373	0.85482529	0.759511663	0.852098176	0.765774354	0.76676648
2	0.739610277	0.731278574	0.763555603	0.734819777	0.75866943	0.760724248
3	0.780362523	0.778415974	0.818881513	0.78124267	0.826559334	0.828933437
4	0.859489654	0.860409262	0.796569237	0.859123024	0.791517566	0.796475378
5	0.660110959	0.657593762	0.72700927	0.65801927	0.724668535	0.721906696
6	0.731040647	0.733860895	0.715122815	0.73058613	0.71143418	0.71087892
7	0.789963329	0.792500581	0.705217709	0.794049279	0.704459239	0.717096577
8	0.756427895	0.755815945	0.821883559	0.759763319	0.818405695	0.817916312
9	0.698797956	0.697193747	0.784911986	0.69813224	0.773534118	0.77916138
10	0.856783709	0.852361186	0.736588113	0.853527057	0.738757482	0.735288134
Final	77.2736+-6.9064	77.1426+-6.9643	76.2925+-4.1944	77.2136+-6.9057	76.1378+-4.2396	76.3515+-4.2219

B: Resultados con 3 clusters

Tabla 13 single MSE con 1000 datos

Iteracion	Berlin	Denver	Dublin	Moscu	Nairobi	Tokyo
1	1119716.23	687477.546	43183.15809	31232.29278	207.8665946	379.3798704
2	387758.9369	696549.8133	32165.06937	55449.40194	738.6273688	390.8279365
3	643570.5333	439210.8907	50464.78188	47709.86094	251.1568961	418.2694644
4	612310.2338	294847.8236	38283.73145	40864.18621	146.4412464	212.288025
5	394425.7086	1231581.582	43888.46249	26819.03356	385.8775374	640.9665739
6	687717.6107	477831.3532	29150.47241	63693.51107	571.2003364	115.2882049
7	458260.2161	412515.3705	10886.58899	42441.7489	385.7551696	253.5456409
8	1051993.684	542313.4247	21070.96129	60704.33572	332.5133251	288.0593607
9	1039354.845	848582.9479	85888.7333	39681.08608	276.9327549	257.5786305
10	537844.509	411256.8573	52049.24884	35172.19684	620.409422	260.4318946
Final	693295.25+-279165.57	604216.76+-275979.77	40703.12+-20463.31	44376.77+-12370.87	391.68+-193.03	321.66+-144.37

Tabla 14 distribuido MSE con 1000 datos

Iteracion	Berlin	Denver	Dublin	Moscu	Nairobi	Tokyo
1	1071326.045	633797.5321	37388.59068	33747.97368	198.2842891	396.8398252
2	443311.5783	451419.0351	44302.63126	37847.99039	558.8702333	460.5373705
3	633387.1314	431544.4059	44886.1001	45184.37146	364.3004164	236.2924742
4	484868.5209	358708.6245	37309.68139	34768.99756	121.3772921	183.6838393
5	511162.4556	888135.1163	38141.20466	29762.92688	379.3698398	557.4938099
6	580445.9614	542871.0302	41908.54394	43075.34878	356.7796081	156.0639926
7	494513.7673	380567.5241	14870.81406	31989.11077	454.3382234	202.6818735
8	828994.2858	699707.2238	34837.53495	34819.65663	419.6212811	244.7371625
9	681248.5013	833860.6606	82392.50281	37507.65574	212.1310897	318.5651983
10	447612.1695	474346.4301	19970.00304	28053.55228	248.4418212	382.6794373
Final	617687.04+-199890.17	569495.76+-187192.6	39600.76+-18007.41	35675.76+-5427.96	331.35+-134	313.96+-132.29

Tabla 15 single R2 con 1000 datos

Iteracion	Berlin	Denver	Dublin	Moscu	Nairobi	Tokyo
1	0.713428356	0.725019296	0.612405299	0.728055371	0.844280507	0.7136178
2	0.857510888	0.667058308	0.793806063	0.543437575	0.414651599	0.689505447
3	0.797317058	0.820244525	0.740251224	0.681256535	0.834761161	0.598164148
4	0.79466594	0.842723025	0.689544589	0.715004917	0.883997725	0.813395815
5	0.824894361	0.629613702	0.668093799	0.831213202	0.631386212	0.612283369
6	0.733356658	0.782555581	0.741248336	0.526602248	0.702465716	0.924116341
7	0.847557695	0.83136087	0.885689387	0.763994444	0.807755176	0.792271245
8	0.714220277	0.837064112	0.891077622	0.658436269	0.82199479	0.652133263
9	0.621862254	0.666697782	0.61802248	0.566501537	0.760450375	0.830781415
10	0.78779989	0.814925444	0.531075359	0.732581013	0.56512651	0.843827617
Final	76.9261+-7.2993	76.1726+-8.2035	71.7121+-11.7491	67.4708+-10.0813	72.6687+-14.9141	74.701+-10.9516

Tabla 16 distribuido R2 con 1000 datos

Iteracion	Berlin	Denver	Dublin	Moscu	Nairobi	Tokyo
1	0.725812971	0.746490496	0.664415011	0.70615093	0.851458917	0.700437817
2	0.837097055	0.784227611	0.715998313	0.688365074	0.557105773	0.634124555
3	0.800524169	0.823382181	0.768965422	0.698129006	0.760322815	0.772991347
4	0.837402649	0.808658559	0.697443482	0.757513993	0.903851938	0.8385393
5	0.77306898	0.732901919	0.711557397	0.812685677	0.637602762	0.662775516
6	0.774948251	0.752958287	0.62800241	0.679845357	0.814155983	0.897277377
7	0.835497789	0.84442137	0.843854501	0.822118361	0.773576147	0.833943691
8	0.774799259	0.789775778	0.819913905	0.804081015	0.775363065	0.70445009
9	0.752148389	0.672480329	0.633571451	0.59024531	0.816504469	0.790715744
10	0.823399979	0.786533761	0.820085271	0.786705033	0.825855704	0.770519814
Final	79.347+-3.9353	77.4183+-4.9782	73.0381+-7.9077	73.4584+-7.4489	77.158+-10.2604	76.0578+-8.4084

Tabla 17 single MSE con 10000 datos

Iteracion	Berlin	Denver	Dublin	Moscu	Nairobi	Tokyo
1	292288.0753	694931.7564	67499.87724	56184.37766	226.6699505	595.2730817
2	999783.7729	757897.5264	55852.67218	41923.71055	144.4809214	585.2487957
3	881399.2908	693792.986	57394.1971	45222.40372	383.4243424	306.4131879
4	700499.1029	538248.8021	48914.39503	36498.99374	545.9964182	234.2732719
5	873202.2316	964406.1574	68344.9065	37105.15077	299.3213075	504.7406963
6	678506.0327	661770.1546	70491.16263	32354.58022	473.6765221	289.265709
7	875470.6265	1070098.185	30078.0628	37620.30381	208.2600312	560.7547469
8	613248.5717	260180.2397	46424.57481	69453.06917	499.0254731	424.7626323
9	1304500.316	1196562.51	28506.09195	60760.70378	632.1044546	298.1220198
10	621503.8362	986204.6545	16945.66887	52013.45339	340.2287274	394.8301454
Final	784040.19+-270542	782409.3+-277276.54	49045.16+-18582.85	46913.67+-12213.31	375.32+-160	419.37+-135.41

Tabla 18 distribuido MSE con 10000 datos

Iteracion	Berlin	Denver	Dublin	Moscu	Nairobi	Tokyo
1	424908.5096	595821.947	25201.31442	26479.96367	204.8109598	204.7394208
2	724437.8336	790024.8128	36345.53545	38697.76098	321.9733311	275.6777838
3	591748.7327	829494.3441	30530.8475	29976.20187	228.3046874	233.7333994
4	763301.1207	774286.85	37869.42575	34899.51991	490.5933338	457.0475748
5	764759.9676	796144.1841	42305.48312	42685.40908	139.8681774	118.0666688
6	602268.1887	569690.1757	43593.57629	43395.63397	309.1615615	291.3160242
7	720232.7373	788967.9188	33685.35782	39775.93733	264.6636482	290.9755217
8	654493.1961	648180.2663	35584.27901	38282.91041	379.7872396	393.5379858
9	705008.1389	694400.2019	43233.03159	40094.32997	235.1819483	259.597722
10	923894.4144	1197075.344	38214.52742	43901.20242	241.2693377	297.0445325
Final	687505.28+-132000.05	768408.6+-175874.85	36656.34+-5830.97	37818.89+-5777.11	281.56+-99.22	282.17+-93.8

Tabla 19 single R2 con 10000 datos

Iteracion	Berlin	Denver	Dublin	Moscu	Nairobi	Tokyo
1	0.913264862	0.670385963	0.748645189	0.891037419	0.789612189	0.868194763
2	0.685321359	0.753252017	0.731453396	0.738811043	0.883047428	0.565823467
3	0.79816082	0.658767915	0.702052413	0.890156205	0.758566456	0.839669706
4	0.579177334	0.801893943	0.744729364	0.778394891	0.610819696	0.739158834
5	0.646026225	0.718743141	0.679973975	0.721921941	0.900979755	0.863798761
6	0.773745524	0.794462935	0.747679897	0.667327816	0.77563743	0.775990797
7	0.762490029	0.68269213	0.769610803	0.766646275	0.774170933	0.749937576
8	0.672680228	0.778483222	0.718066701	0.735122161	0.789051453	0.66099565
9	0.751803433	0.659497781	0.745285866	0.651151242	0.69370761	0.796966784
10	0.481222497	0.748644448	0.754458654	0.711775444	0.82066233	0.696307877
Final	70.6389+-12.1597	72.6682+-5.619	73.4196+-2.6937	75.5234+-8.1341	77.9626+-8.4104	75.5684+-9.5663

Tabla 20 distribuido R2 con 10000 datos

Iteracion	Berlin	Denver	Dublin	Moscu	Nairobi	Tokyo
1	0.823528724	0.810786447	0.836487682	0.831660985	0.84899242	0.858134394
2	0.733992576	0.738133918	0.76659472	0.753048054	0.791343917	0.756413813
3	0.747418316	0.735679797	0.81212244	0.822893713	0.838249573	0.824986302
4	0.723835587	0.73078924	0.776548759	0.773468763	0.685518134	0.690630465
5	0.704386604	0.704122539	0.739001771	0.741549795	0.89566378	0.898410658
6	0.780776032	0.798030863	0.740136743	0.727639195	0.789203415	0.772541234
7	0.732515235	0.735832348	0.782119328	0.779579502	0.79293865	0.788213923
8	0.746018114	0.756817805	0.751689837	0.733015469	0.743386623	0.727500574
9	0.750545785	0.752690439	0.728021086	0.722250134	0.797641053	0.803061007
10	0.660687182	0.649077467	0.735267953	0.756093001	0.807387601	0.806132676
Final	74.037+-4.3145	74.1196+-4.5321	76.6799+-3.5669	76.412+-3.8138	79.9033+-5.7366	79.2603+-6.0735

Tabla 21 single MSE con 100000 datos

Iteracion	Berlin	Denver	Dublin	Moscu	Nairobi	Tokyo
1	292288.0753	694931.7564	67499.87724	56184.37766	226.6699505	595.2730817
2	999783.7729	757897.5264	55852.67218	41923.71055	144.4809214	585.2487957
3	881399.2908	693792.986	57394.1971	45222.40372	383.4243424	306.4131879
4	700499.1029	538248.8021	48914.39503	36498.99374	545.9964182	234.2732719
5	873202.2316	964406.1574	68344.9065	37105.15077	299.3213075	504.7406963
6	678506.0327	661770.1546	70491.16263	32354.58022	473.6765221	289.265709
7	875470.6265	1070098.185	30078.0628	37620.30381	208.2600312	560.7547469
8	613248.5717	260180.2397	46424.57481	69453.06917	499.0254731	424.7626323
9	1304500.316	1196562.51	28506.09195	60760.70378	632.1044546	298.1220198
10	621503.8362	986204.6545	16945.66887	52013.45339	340.2287274	394.8301454
Final	784040.19+-270542	782409.3+-277276.54	49045.16+-18582.85	46913.67+-12213.31	375.32+-160	419.37+-135.41

Tabla 22 distribuido MSE con 100000 datos

Iteracion	Berlin	Denver	Dublin	Moscu	Nairobi	Tokyo
Final	715011.72+-229291.92	715310.26+-232833.96	43079.3+-9409.41	43647.39+-9671.83	358.9+-34.09	355.04+-40.33

Tabla 23 single R2 con 100000 datos

Iteracion	Berlin	Denver	Dublin	Moscu	Nairobi	Tokyo
1	0.894223362	0.753472656	0.557701025	0.643169495	0.837796017	0.55749946
2	0.634503035	0.728421604	0.633193008	0.746211069	0.898927938	0.570703545
3	0.676933361	0.754287956	0.621993231	0.705676457	0.729578322	0.778637414
4	0.754076309	0.79797459	0.684215788	0.762043332	0.599913176	0.834081429
5	0.690060021	0.65308433	0.562456336	0.76532317	0.781972673	0.647065009
6	0.752254227	0.761112637	0.545099094	0.795237432	0.660487913	0.785496045
7	0.6805318	0.610383191	0.807246772	0.759985618	0.847982126	0.575880561
8	0.77360124	0.905613832	0.708436733	0.54998648	0.634791562	0.689574395
9	0.524334055	0.56562063	0.815497368	0.617723239	0.532989857	0.779995987
10	0.773307926	0.643356879	0.892745007	0.660659956	0.747608733	0.719067225
Final	71.5383+-9.8861	71.7333+-10.0399	68.2858+-12.1602	70.0602+-7.9582	72.7205+-11.8623	69.38+-10.1757

Tabla 24 distribuido R2 con 100000 datos

Iteracion	Berlin	Denver	Dublin	Moscu	Nairobi	Tokyo
1	0.854294734	0.853846483	0.618824893	0.621904609	0.738760478	0.738049324
2	0.702557418	0.70610365	0.709146708	0.714211145	0.794098016	0.795970278
3	0.732317167	0.730860872	0.691921794	0.694741848	0.757377401	0.759838054
4	0.787964739	0.793061146	0.752026283	0.747844842	0.742216835	0.74537395
5	0.685742943	0.688825274	0.696704227	0.695645515	0.724129894	0.727419644
6	0.783582369	0.787102331	0.720241925	0.723320869	0.728976045	0.732607957
7	0.675988819	0.674814651	0.787948543	0.789346961	0.764052381	0.760649129
8	0.860781523	0.858482185	0.6585477	0.653050357	0.725989888	0.724514531
9	0.59124882	0.586227412	0.753812264	0.753781181	0.689862	0.686
10	0.731722163	0.735359138	0.829576569	0.826380548	0.733288641	0.744867782
Final	74.062+-8.3364	74.1468+-8.4182	72.1875+-6.1662	72.2023+-6.0902	73.9875+-2.7773	74.1529+-2.8586

C: Entrenamiento local

Tabla 25 Resultados utilizando 1000 datos y 8 iteraciones en el modelo local

node	dist R2	dist MSE	single R2	single MSE
Nairobi	84.2241+-4.905	9025.134+-2695.5999	80.3234+-7.7056	11208.2044+-4100.5663
Tokyo	81.4715+-8.2923	9373.7865+-3495.3419	79.9896+-13.9511	9768.6705+-5752.964
Dublin	83.2476+-3.7024	9405.9194+-3141.771	81.6731+-6.5705	10112.4177+-3538.1534
Moscu	78.6441+-5.8807	356523.5622+-120592.5047	75.2565+-9.1175	412164.9494+-168974.8186
Berlin	75.8362+-6.3816	381352.1092+-135242.4414	74.854+-5.9414	394380.7193+-122778.5434
Denver	74.5012+-6.6484	422102.5048+-135856.9188	75.4136+-8.2887	397686.8527+-114955.9329

Tabla 26 Resultados utilizando 1000 datos y 100 iteraciones en el modelo local

node	dist R2	dist MSE	single R2	single MSE
Nairobi	99.8286+-0.0501	89.9327+-29.7655	99.8441+-0.0312	80.4688+-10.5601
Tokyo	94.5504+-2.4856	2870.729+-1584.6686	99.8321+-0.0262	84.8534+-10.3946
Dublin	94.0286+-2.2893	3253.0357+-1098.1299	99.8206+-0.0795	98.8174+-51.6973
Moscu	99.9258+-0.091	1280.5317+-1671.3721	99.9423+-0.0625	1020.2853+-1165.4989
Berlin	95.3057+-1.5573	74778.9859+-26612.0787	99.9378+-0.0357	1019.5502+-610.7825
Denver	95.0981+-1.8055	87505.6979+-31351.0636	99.9495+-0.0341	961.1431+-789.8685