

**UNIVERSIDAD DE CASTILLA-LA MANCHA**

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

**GRADO EN INGENIERÍA INFORMÁTICA**

TECNOLOGÍA ESPECÍFICA DE COMPUTACIÓN

**TRABAJO FIN DE GRADO**

Predicción de precios de instancias “*spot*” en la nube de Amazon

**Autor**: Luis Mendoza Montero

**Directoras**: María del Carmen Carrión Espinosa

María Blanca Caminero Herráez

Diciembre, 2018

DEDICATORIA

*A mi familia, en especial a mis padres, hermanos y sobrinos por su apoyo constante.*

*A mi abuela por su preocupación e interés en todo momento.*

*A mis amigos y compañeros de carrera por estar ahí durante todos estos años juntos.*

*A mis profesores durante estos años por todas sus enseñanzas y tutoras del trabajo por toda su ayuda y esmero.*

AGRADECIMIENTOS

*A todos los profesores de la facultad y en especial a Blanca y a Carmen por su tiempo y dedicación.*

*A mis compañeros de Opinator, gracias a ellos he conseguido mucha información relevante para completar este documento.*

DECLARACIÓN DE AUTORÍA

 Yo, Luis Mendoza Montero con DNI 49217658G, declaro que soy el único autor del Trabajo Fin de Grado titulado Predicción de precios de instancias “*spot*” en la nube de Amazon y que el citado trabajo no infringe las leyes en vigor sobre propiedad intelectual y que todo el material no original contenido en dicho trabajo está apropiadamente atribuido a sus legítimos autores.

Albacete, a 4 de diciembre de 2018

Fdo.: Luis Mendoza Montero

RESUMEN

Hoy en día, se ofrecen numerosos servicios de alquiler de computación y almacenamiento disponibles en la nube. Estos servicios son ofertados por proveedores públicos de *IaaS*. *Amazon Web Services,* del que dedicaremos a hablar en este trabajo, es uno de estos principales proveedores junto con *Microsoft Azure*. Además, *Amazon* fue pionera de la computación en la nube. *Amazon EC2* es un servicio que ofrece instancias donde cada una de estas aporta unas características computacionales específicas en función de su tipo. En este trabajo, se desarrollará el estudio sobre las instancias *spot*. El precio de las instancias *spot* varía y puede afectar a las prestaciones y disponibilidad de las aplicaciones que las usan. Las instancias se organizan atendiendo a las diferentes regiones que *Amazon* tiene ubicadas por distintos territorios a nivel mundial compuestas por diferentes zonas de disponibilidad.

El objetivo del *TFG* será aplicar distintas técnicas de análisis sobre el comportamiento de los datos históricos que obtendremos para predecir el precio. De esta forma, se ofrecerá al usuario un precio *(bid)* para que sus instancias *spot* no se interrumpan, ya que estas son ofertadas mediante un sistema de subasta. El *bid* será el precio que le será entregado al usuario para que pueda contratar una instancia *spot* sin que el sistema llegue a interrumpírsela en algún instante durante su periodo de contratación. El usuario ha de pujar por el uso de las instancias superando o igualando el precio que el sistema le impone.

Se analizará el comportamiento de los datos gracias al esbozo de gráficas y a la ejecución del coeficiente de *Gini* como medidas de análisis previas a la predicción. Concretamente, para realizar la predicción de los precios, se utilizarán dos técnicas de suavizado exponencial llamadas *Simple Exponential Smoothing* y *Double Exponential Smoothing*.

Realizaremos una serie de experimentos donde se podrá analizar el estudio y la aplicación de los algoritmos anteriores a datos reales y luego evaluaremos los resultados de estos mediante unas métricas de validación que propondremos. Estas métricas nos servirán para realizar una comparativa de la eficiencia entre los dos algoritmos de predicción para cada experimento.

Finalmente, extraeremos algunas conclusiones sobre el trabajo realizado y propondremos algunas ideas para mejorarlo.

**Índice de Contenido**

[**1. INTRODUCCIÓN** 17](#_Toc531645361)

[**1.1** **Esquema inicial** 17](#_Toc531645362)

[**1.2** **Motivación** 19](#_Toc531645363)

[**1.3** **Objetivos** 20](#_Toc531645364)

[**1.4** **Metodología y temporización** 21](#_Toc531645365)

[**1.5** **Estructura de la memoria** 23](#_Toc531645366)

[**2. CARACTERÍSTICAS DEL ENTORNO DE APLICACIÓN** 25](#_Toc531645367)

[**2.1** **Cloud Computing** 25](#_Toc531645368)

[***2.1.1*** ***¿Qué es Cloud Computing?*** 25](#_Toc531645369)

[***2.1.2*** ***Características generales*** 25](#_Toc531645370)

[***2.1.3*** ***Modelos de despliegue*** 26](#_Toc531645371)

[***2.1.4*** ***Tipos de contratación*** 28](#_Toc531645372)

[**2.2** **Amazon Web Services** 30](#_Toc531645373)

[***2.2.1*** ***¿Qué es Amazon Web Services?*** 30](#_Toc531645374)

[***2.2.2*** ***Infraestructura interna*** 31](#_Toc531645375)

[***2.2.3*** ***Instancias de Amazon EC2*** 34](#_Toc531645376)

[***2.2.4*** ***Infraestructura global*** 38](#_Toc531645377)

[**3. MÉTODOS DE ANÁLISIS Y PRONÓSTICO** 41](#_Toc531645378)

[**3.1** **Medidas de pronóstico y series temporales** 41](#_Toc531645379)

[**3.2** **Coeficiente de Gini** 43](#_Toc531645380)

[**3.3** **Suavizado Exponencial** 46](#_Toc531645381)

[***3.3.1*** ***Simple Exponential Smoothing*** 46](#_Toc531645382)

[***3.3.2*** ***Double Exponential Smoothing*** 49](#_Toc531645383)

[**4. FUNCIONAMIENTO DEL SISTEMA** 51](#_Toc531645384)

[**4.1** **Entorno de trabajo** 51](#_Toc531645385)

[***4.1.2*** ***Jupyter Notebook*** 51](#_Toc531645386)

[***4.1.3*** ***SQLite*** 52](#_Toc531645387)

[**4.2** **Preparación de los datos** 53](#_Toc531645388)

[***4.2.1*** ***Credenciales de acceso*** 54](#_Toc531645389)

[***4.2.2*** ***Obtención de datos desde la nube*** 54](#_Toc531645390)

[***4.2.3*** ***Preprocesado de datos*** 55](#_Toc531645391)

[**4.3** **Parámetros de entrada** 61](#_Toc531645392)

[**4.4** **Análisis de gráficas** 62](#_Toc531645393)

[**4.5** **Obtención del modelo** 62](#_Toc531645394)

[***4.5.1*** ***Entrenamiento de los datos (Training)*** 62](#_Toc531645395)

[***4.5.2*** ***Evaluación de los datos (Test)*** 63](#_Toc531645396)

[**5. ANÁLISIS DE RESULTADOS** 65](#_Toc531645397)

[**5.1** **Experimento 1 – Demanda instancia *spot*: Predicción Diaria** 65](#_Toc531645398)

[***5.1.1*** ***Análisis temporal*** 66](#_Toc531645399)

[***5.1.2*** ***Cálculo del coeficiente de Gini*** 68](#_Toc531645400)

[***5.1.3*** ***Eficiencia de los algoritmos de predicción*** 68](#_Toc531645401)

[**5.2** **Experimento 2 – Demanda instancia *spot*: Predicción Semanal** 74](#_Toc531645402)

[***5.2.1*** ***Análisis temporal*** 74](#_Toc531645403)

[***5.2.2*** ***Cálculo del coeficiente de Gini*** 76](#_Toc531645404)

[***5.2.3*** ***Eficiencia de los algoritmos de predicción*** 77](#_Toc531645405)

[**5.3** **Experimento 3 – Demanda instancia *spot*: Predicción Mensual** 83](#_Toc531645406)

[***5.3.1*** ***Análisis temporal*** 83](#_Toc531645407)

[***5.3.2*** ***Cálculo del coeficiente de Gini*** 85](#_Toc531645408)

[***5.3.3*** ***Eficiencia de los algoritmos de predicción*** 85](#_Toc531645409)

[**6. CONCLUSIONES Y PROPUESTAS DE MEJORA** 93](#_Toc531645410)

[**6.1** **Conclusiones** 93](#_Toc531645411)

[**6.2** **Propuestas de mejora** 94](#_Toc531645412)

[**ANEXOS** 95](#_Toc531645413)

[**ANEXO A. Tipos de servicios de *AWS*** 95](#_Toc531645414)

[**ANEXO B. Especificaciones de las instancias *EC2*** 99](#_Toc531645415)

[**ANEXO C. Regiones y zonas de disponibilidad** 102](#_Toc531645416)

[**ANEXO D. Contenido del CD** 105](#_Toc531645417)

[**BIBLIOGRAFÍA** 107](#_Toc531645418)

**Índice de Figuras**

[Figura 1. Esquema de trabajo 19](file:///C:\Users\Luis\Desktop\TFG\Memoria\TFG%20-%20Luis%20Mendoza%20Montero.docx#_Toc531644000)

[Figura 2. Objetivos 21](file:///C:\Users\Luis\Desktop\TFG\Memoria\TFG%20-%20Luis%20Mendoza%20Montero.docx#_Toc531644001)

[Figura 3. Temporización 22](file:///C:\Users\Luis\Desktop\TFG\Memoria\TFG%20-%20Luis%20Mendoza%20Montero.docx#_Toc531644002)

[Figura 4. Modelos de despliegue 26](#_Toc531644003)

[Figura 5. Tipos de contratación [22] 30](file:///C:\Users\Luis\Desktop\TFG\Memoria\TFG%20-%20Luis%20Mendoza%20Montero.docx#_Toc531644004)

[Figura 6. Infraestructura Amazon EC2 [19] 33](file:///C:\Users\Luis\Desktop\TFG\Memoria\TFG%20-%20Luis%20Mendoza%20Montero.docx#_Toc531644005)

[Figura 7. Lanzamiento de una instancia [39] 34](file:///C:\Users\Luis\Desktop\TFG\Memoria\TFG%20-%20Luis%20Mendoza%20Montero.docx#_Toc531644006)

[Figura 8. Ciclo de vida de una instancia [40] 35](#_Toc531644007)

[Figura 9. Configuración de las regiones [44] 39](#_Toc531644008)

[Figura 10. Mapa de regiones [44] 39](#_Toc531644009)

[Figura 11. Metodología de Análisis 41](#_Toc531644010)

[Figura 12. Curva de Lorenz [46] 44](file:///C:\Users\Luis\Desktop\TFG\Memoria\TFG%20-%20Luis%20Mendoza%20Montero.docx#_Toc531644011)

[Figura 13. Índice libreta de Jupyter 52](#_Toc531644012)

[Figura 14. Estructura de tablas BBDD 53](#_Toc531644013)

[Figura 15. Atributos de una tabla 53](#_Toc531644014)

[Figura 16. Línea de comandos para credenciales 54](#_Toc531644015)

[Figura 17. Timestamp sin preprocesar 56](#_Toc531644016)

[Figura 18. Timestamp preprocesado por horas 58](file:///C:\Users\Luis\Desktop\TFG\Memoria\TFG%20-%20Luis%20Mendoza%20Montero.docx#_Toc531644017)

[Figura 19. Gráfica horas sin preprocesar 58](#_Toc531644018)

[Figura 20. Gráfica horas preprocesadas 59](#_Toc531644019)

[Figura 21. Entrenamiento y Test 64](#_Toc531644020)

[Figura 22. Gráfica ap-northeast-1 (Experimento 1) 66](#_Toc531644021)

[Figura 23. Gráfica ap-southeast-1 (Experimento 1) 67](#_Toc531644022)

[Figura 24. Gráfica Entrenamiento Simple (Experimento 1) 69](#_Toc531644023)

[Figura 25. Gráfica Test Simple (Experimento 1) 70](#_Toc531644024)

[Figura 26. Gráfica Entrenamiento Double (Experimento 1) 72](#_Toc531644025)

[Figura 27. Gráfica Test Double (Experimento 1) 73](#_Toc531644026)

[Figura 28. Gráfica us-east-2 (Experimento 2) 75](#_Toc531644027)

[Figura 29. Gráfica us-west-1 (Experimento 2) 76](#_Toc531644028)

[Figura 30. Gráfica Entrenamiento Simple (Experimento 2) 78](#_Toc531644029)

[Figura 31. Gráfica Test Simple (Experimento 2) 79](#_Toc531644030)

[Figura 32. Gráfica Entrenamiento Double (Experimento 2) 80](#_Toc531644031)

[Figura 33. Gráfica Test Double (Experimento 2) 81](#_Toc531644032)

[Figura 34. Gráfica eu-central-1 (Experimento 3) 83](#_Toc531644033)

[Figura 35. Gráfica ap-northeast-1 (Experimento 3) 84](#_Toc531644034)

[Figura 36. Gráfica Entrenamiento Simple (Experimento 3) 86](#_Toc531644035)

[Figura 37. Gráfica Test Simple (Experimento 3) 87](#_Toc531644036)

[Figura 38. Gráfica Entrenamiento Double (Experimento 3) 89](#_Toc531644037)

[Figura 39. Gráfica Test Double (Experimento 3). 90](#_Toc531644038)

[Figura 40. Localización de regiones en América del Norte [44] 102](#_Toc531644039)

[Figura 41. Localización de regiones en América del Sur [44] 102](#_Toc531644040)

[Figura 42. Localización de regiones en Europa, Oriente Medio y África [44] 103](#_Toc531644041)

[Figura 43. Localización de regiones en Asia Pacífico [44] 103](#_Toc531644042)

**Índice de Tablas**

[Tabla 1. Entrenamiento Simple (Experimento 1) 69](#_Toc531644226)

[Tabla 2. Test Simple (Experimento 1) 70](#_Toc531644227)

[Tabla 3. Entrenamiento Double (Experimento 1) 71](#_Toc531644228)

[Tabla 4. Test Double (Experimento 1) 72](#_Toc531644229)

[Tabla 5. Ahorro Económico Double (Experimento 1) 74](#_Toc531644230)

[Tabla 6. Simple vs Double (Experimento 1) 74](#_Toc531644231)

[Tabla 7. Entrenamiento Simple (Experimento 2) 77](#_Toc531644232)

[Tabla 8. Test Simple (Experimento 2) 78](#_Toc531644233)

[Tabla 9. Ahorro Económico Simple (Experimento 2) 79](#_Toc531644234)

[Tabla 10. Entrenamiento Double (Experimento 2) 80](#_Toc531644235)

[Tabla 11. Test Double (Experimento 2) 81](#_Toc531644236)

[Tabla 12. Ahorro Económico Double (Experimento 2) 82](#_Toc531644237)

[Tabla 13. Simple vs Double (Experimento 2) 82](#_Toc531644238)

[Tabla 14. Entrenamiento Simple (Experimento 3) 86](#_Toc531644239)

[Tabla 15. Test Simple (Experimento 3) 87](#_Toc531644240)

[Tabla 16. Ahorro Económico Simple (Experimento 3) 88](#_Toc531644241)

[Tabla 17. Entrenamiento Double (Experimento 3) 88](#_Toc531644242)

[Tabla 18. Test Double (Experimento 3) 89](#_Toc531644243)

[Tabla 19. Simple vs Double (Experimento 3) 90](#_Toc531644244)

[Tabla 20. Características instancias de uso general 99](#_Toc531644245)

[Tabla 21. Características instancias optimizadas para informática 100](#_Toc531644246)

[Tabla 22. Características instancias optimizadas para memoria 100](#_Toc531644247)

[Tabla 23. Características instancias para informática acelerada 101](#_Toc531644248)

[Tabla 24. Características instancias optimizadas para almacenamiento 101](#_Toc531644249)

[Tabla 25. Regiones en América del Norte 102](#_Toc531644250)

[Tabla 26. Regiones en América del Sur 103](#_Toc531644251)

[Tabla 27. Regiones en Europa, Oriente Medio y África 103](#_Toc531644252)

[Tabla 28. Regiones en Asia Pacífico 104](#_Toc531644253)

# **INTRODUCCIÓN**

En este capítulo de introducción, vamos a hablar acerca de los objetivos del *TFG*, así como la metodología y el proceso que se ha llevado a cabo durante el trabajo.

## **Esquema inicial**

*Amazon Web Services (AWS)* [1] es una plataforma que ofrece un conjunto de servicios fiables, escalables y económicos que se ofrecen en la nube pública [2] con el fin de adquirir potencia de cómputo, almacenamiento de bases de datos y otras funcionalidades de gran ayuda para el mundo empresarial. Por tanto, ofrece un completo servicio de infraestructura en la red muy beneficioso para el ámbito de las tecnologías de la información. Esta agrupación de servicios de *Amazon* permite un acceso de prueba gratuito, aunque con limitaciones a algunos recursos.

En este *TFG* nos centraremos en el servicio web de *Amazon Elastic Compute Cloud (Amazon EC2)* que proporciona servicios *on-line* del tipo *IaaS*. Los datos que necesitaremos para el desarrollo del trabajo serán los que *Amazon* pone a su disposición en la nube y que nosotros obtendremos gracias a los mecanismos que se irán desarrollando durante la fase de trabajo. Estos datos estarán distribuidos geográficamente en varias regiones divididas a su vez por varios *data centers* (zonas de disponibilidad). *Amazon* tiene varios *data-centers* repartidos por diferentes lugares geográficos. Por ejemplo, existe una región en Europa que contiene las zonas de disponibilidad en Fráncfort, Irlanda, Londres y París [4]. Las instancias que este servicio ofrece se clasifican por diferentes familias con unas especificaciones computacionales particulares para máquinas virtuales (herramientas *software* que simulan un computador virtual como si se tratase de uno real) según el sistema operativo (*SUSE Linux*, *Unix* o *Windows*) [5]. Las familias se estructuran en función de unas características generales (de uso general, optimizadas para cómputo, optimizadas para memoria, aceleración *hardware* y optimizadas para almacenamiento.) Dentro de cada una de ellas, existen distintas subcategorías atendiendo al número de *cores*, a la memoria o al almacenamiento entre otras características en función de la categoría a la que pertenezcan [6]. Cabe mencionar que no todas las categorías están disponibles en todos los *data center* de los que habíamos hablado anteriormente.

Como pionera de la computación en la nube, el servicio de *Amazon* *EC2* utiliza tres opciones de compra diferentes para mejorar la administración del rendimiento: instancias reservadas, instancias bajo demanda e instancias de subasta o también denominadas puntuales o *spot*. También existen los *hosts* dedicados, pero de estos hablaremos más adelante.

* Con las instancias reservadas, los consumidores incurren en un coste fijo relativamente alto, pero tienen acceso garantizado al pedido, y las instancias no son anuladas una vez que son aprovisionadas.
* Con las instancias bajo demanda, los consumidores incurren en un alto coste fijo y no tienen acceso garantizado al pedido, pero las instancias no son anuladas una vez que son aprovisionadas.
* Con las instancias puntuales (*spot*), los usuarios envían solicitudes de una instancia *spot* especificando el precio máximo por hora que están dispuestos a pagar por los recursos requeridos. Las instancias *spot* se basan en subasta, y pueden llegar a ser interrumpidas.

El tipo de instancias en las que se basará este *TFG* será con las instancias *spot* y el proceso que se lleva a cabo frente a una solicitud será el siguiente:

* El usuario fija el precio máximo a pagar por una instancia o instancias que desea desplegar en una región específica (también llamado *bid*).
* Siempre y cuando el precio que determina el sistema para las instancias solicitadas por los usuarios sea inferior al precio especificado por el mismo, las instancias se desplegarán.
* En cuanto el precio de las instancias del sistema supere al ofrecido por el usuario, entonces las instancias se darán de baja con un preaviso de dos minutos.

Estas instancias *spot* permiten disponer unas especificaciones computacionales a un precio inferior al resto de instancias. Al ser susceptibles de ser interrumpidas debido a la fluctuación de su precio, la aplicación que las utilice debe ser diseñada para poder funcionar con este tipo de interrupciones.

Analizar el historial de precios de las instancias *spot* para una cantidad tan grande de mercados, entender la dinámica de precios dentro y entre ellos, y determinar el mejor tipo de instancia, zona de disponibilidad, intervalo de tiempo y precio de oferta no es sencillo. Por tanto, los usuarios más potenciales necesitan ayuda debido a la gran cantidad de historial de precios *spot* que está disponible para ellos. Pueden beneficiarse de un servicio que analiza e interpreta el historial de precios con el fin de cuantificar la volatilidad y variabilidad de cada tipo de instancia. Esto se realiza, bien identificando patrones de precios y tendencias dentro y entre las instancias, o bien calificando el riesgo asociado con las pujas de las instancias *spot*. Gracias a un enfoque estadístico basado en distintas métricas y algoritmos, podremos establecer una aproximación con respecto al precio de una determinada instancia *spot*.

Con los datos ya almacenados en la base de datos y una vez preparados para ser ejecutados en los algoritmos, hay que especificar qué clase de métricas estadísticas emplearemos para que los resultados obtenidos sean adecuados para la aproximación de nuestro objetivo. Algunos ejemplos de algoritmos que podemos utilizar son el coeficiente de *Gini* [7] que es muy útil ya que mide la desigualdad entre los valores de los precios [8]. También pueden ser de gran ayuda métodos estadísticos de promedio móvil [10] o cualquier otro modelo de pronóstico [11].

Para una mejor comprensión de la elaboración del *TFG* realizado se adjunta también en este documento en la *Figura 1* una visión global de todos los elementos que intervienen a lo largo del estudio:

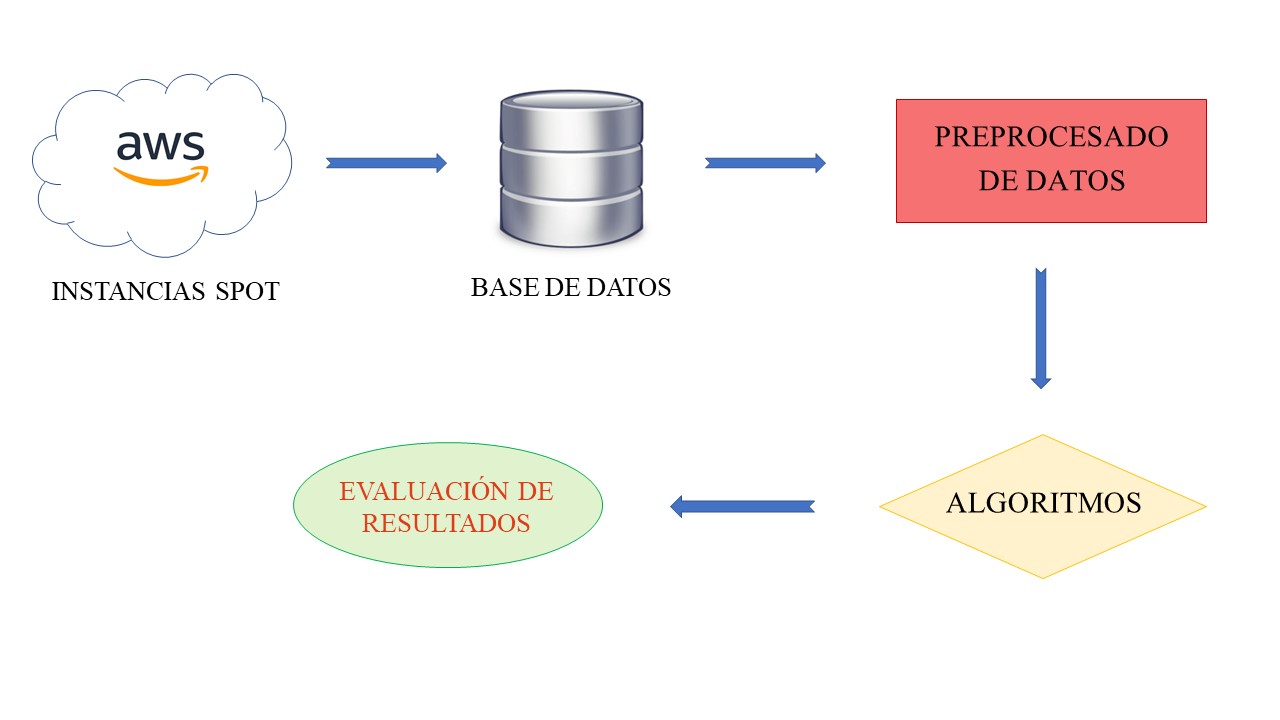


Figura 1. Esquema de trabajo

## **Motivación**

Hoy en día, son muchas las empresas que deciden contratar instancias que el servicio de *Amazon EC2* les ofrece para su negocio gracias a los beneficios que obtienen al decidir levantar sus propios servicios desde la nube, sin necesidad de gastar ningún coste en instalación ni mantenimiento de estos, como los que conllevaría hacerlo de forma tradicional.

Como veremos más adelante existen varios tipos de instancias, unas más baratas que otras según las necesidades de los usuarios y además varios tipos de contratación, los cuales presentan unas ventajas e inconvenientes específicos frente a otros tipos. Las instancias reservadas y bajo demanda son más caras que las instancias *spot*, aunque el inconveniente de estas últimas es que los usuarios pueden perder la instancia debido a que estas pueden ser interrumpidas por el sistema.

La principal motivación de este trabajo es ayudar a los usuarios a establecer un precio máximo a pujar (*bid*). De esta forma podrán ahorrarse costes al decantarse por las instancias *spot* intentando que dichas instancias no sean interrumpidas.

Gracias a los conocimientos adquiridos en la intensificación de Computación, se aprenderán métodos estadísticos y algoritmos para analizar datos. Además, hablaremos de numerosos conceptos relacionados con *Cloud Computing* [27]y en concreto de *Amazon Web Services*.

El objetivo y la gran motivación del desarrollo de este trabajo nace de esta pregunta: ¿Podemos llegar a calcular de algún modo el mejor *bid* para un usuario que quiera contratar una instancia con la opción de *spot* durante un determinado tiempo, para que le suponga un ahorro económico frente a la opción de reservarla por un precio por hora mayor?

A pesar de que el precio por hora que cuesta una instancia reservada cueste no mucho más que el de las instancias *spot*, puede suponer un importante ahorro económico con el paso del tiempo. Por ejemplo, si una instancia reservada nos cuesta 0’8 $ la hora y una instancia *spot* nos cuesta 0’68 $, si un usuario necesita cuatro semanas la instancia, optar por una instancia reservada le supondría un total de 537’6 $ las cuatro semanas. Si el usuario optase por una instancia *spot*, saldría por 436’8 $ en el mejor de los casos (si los 0’68 $ por cada hora es igual o superior al precio otorgado por la máquina).

La respuesta a esta pregunta la dejaremos propuesta para los siguientes apartados y capítulos que nos quedan.

## **Objetivos**

El objetivo primordial de este trabajo consistirá esencialmente en obtener una predicción sobre el precio de una instancia *spot* por medio de una serie de medidas estadísticas que serán implementadas y ejecutadas por medio de los datos que obtendremos del historial de los precios. Se necesitarán unas credenciales de acceso que nos permitirán acceder y utilizar los servicios de la *AWS* y descargar el conjunto de precios disponibles atendiendo a su categoría, familia y región a la que pertenecen.

Para una mejor comprensión, la finalidad del trabajo se subordinará en cinco objetivos parciales que se detallarán a continuación tal como se muestra en la *Figura 2*:

* **Estudio de las características de la plataforma pública *AWS*:** antes de empezar a trabajar, analizaremos nuestro entorno de desarrollo, así conoceremos mejor el dominio del problema. Se estudiarán previamente algunos conceptos relacionados con la computación en la nube, en concreto aquellos servicios que *Amazon* nos ofrece y la relación con los datos que necesitaremos durante todo el progreso.
* **Conocimiento y aplicación de medidas estadísticas:** será necesario inferir y hacer un exhaustivo estudio acerca de los algoritmos y métricas más eficientes para lograr unos resultados aceptables, de manera que nos permita pronosticar el patrón de coste óptimo de las instancias *spot*.
* **Adquisición, almacenamiento y preprocesado de los datos extraídos:** necesitaremos obtener los datos históricos por medio de la ejecución de un *script* que se implementará en función del desarrollo del trabajo, una vez que se aclaren los conceptos previos sobre computación en la nube y *AWS*.
* **Desarrollo, implementación y ejecución de algoritmos estadísticos:** se estudiará qué algoritmos decidiremos emplear para realizar la predicción de los precios. Estos se implementarán y se validará por el conjunto de datos que hemos obtenido.
* **Evaluación de los resultados:** por último, nos centraremos en hacer un análisis en vista de los resultados obtenidos que ayudarán a predecir el precio de las instancias *spot*.

Figura 2. Objetivos

## **Metodología y temporización**

Todos los puntos propuestos para el evolutivo del trabajo constan de una serie de tareas que se especificarán detalladamente a continuación y que resumen cada uno de los temas a tratar. Estas servirán de ayuda para un seguimiento contínuo del trabajo. El orden de la realización de estos puntos es importante para conseguir que el funcionamiento sea el adecuado y el esperado. Así prevendremos algunos errores o fallos inesperados que pudieran llegar a aparecer.

Para el desarrollo del trabajo se contemplará una de las metodologías ágiles más conocidas y empleadas en la actualidad por numerosas empresas como es *SCRUM* [12]. Para equilibrar la organización del trabajo se han realizado reuniones quincenales con las directoras del trabajo, estableciendo un *sprint planning* de quince días, que será el plazo indicado para realizar las tareas propuestas. En estas reuniones se han revisado los estados de las tareas y se han propuesto nuevas ideas para llevarlas a cabo durante el proceso de desarrollo tal y como se muestra en la *Figura 3*:

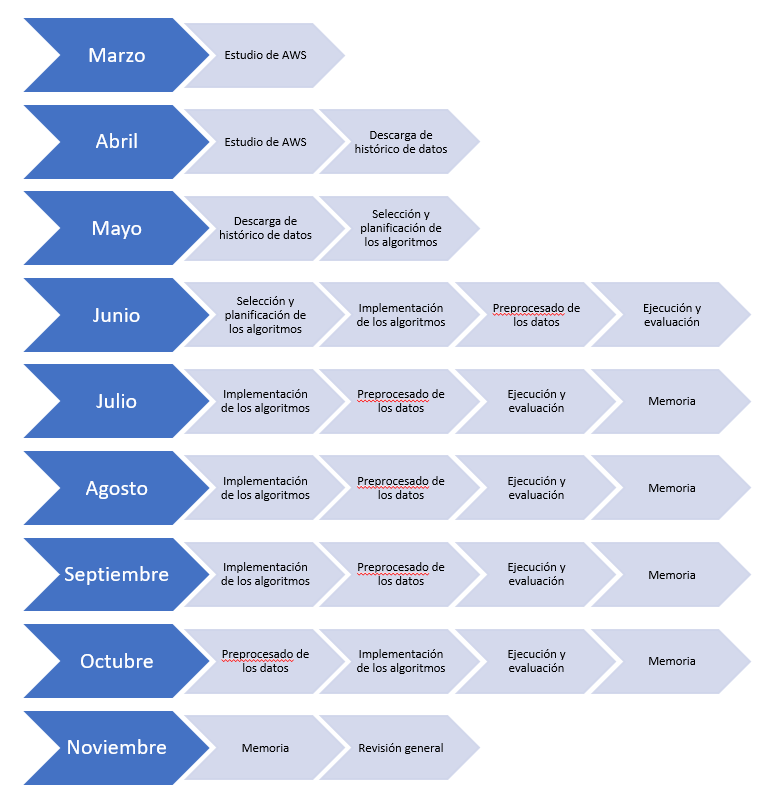
* **Estudio de *AWS* y *Cloud Computing*:** Primero de todo, vamos a estudiar el dominio del problema y los diferentes conceptos que nos encontramos como familiarizarnos con el entorno de los servicios que ofrece *Amazon Web Services* y en concreto con *Amazon* *EC2*. También vamos a analizar qué es *Cloud Computing* y cómo funciona el sistema de oferta y demanda de las instancias *spot*, así como preparar todo nuestro entorno de trabajo a nivel de *software*.
* **Descarga de histórico de datos:** Lo siguiente que debemos realizar es implementar un *script* para descargar toda la información acerca de la dinámica de precios de las instancias y almacenar dicha información en base de datos.
* **Selección y planificación de los algoritmos:** Una vez conseguida toda la información necesaria y preparada la base de datos, es hora de investigar acerca de los algoritmos que puedan llegar a ser de utilidad para estimar el precio de una instancia atendiendo a que los resultados obtenidos sean de utilidad y coherentes a la definición del problema.
* **Preprocesado de los datos:** Es muy importante que antes de someter los datos a las ejecuciones de los algoritmos sean preprocesados de una forma eficaz evitando valores nulos que puedan entorpecer los resultados tras las ejecuciones.
* **Implementación de los algoritmos:** Ya seleccionados los algoritmos que necesitaremos, vamos a implementar cada uno de ellos para que una vez que sean ejecutados al indicarles los parámetros, nos devuelvan los resultados y que estos sean desde un punto de vista computacional, óptimos y coherentes.
* **Ejecución y evaluación:** Por último, tras la ejecución de los algoritmos, esbozaremos unas gráficas que ayuden a analizar visualmente cuáles han sido estos resultados y comprobar si se ajustan al resultado esperado pudiendo así sacar una serie de conclusiones que pondrán el broche de oro al proceso de este trabajo.
* **Memoria de trabajo:** durante el proceso del trabajo se elaborará un informe que abarcará los contenidos del estudio con detalle.

Figura 3. Temporización

## **Estructura de la memoria**

La estructura de la memoria contemplará un total de siete capítulos:

* ***Capítulo 1***. Introducción: esta parte habla sobre una breve introducción con los principales objetivos, la metodología de desarrollo y la planificación del trabajo.
* ***Capítulo 2***. Características del entorno de aplicación: en este capítulo se abordarán los diferentes conceptos relacionados con la computación en la nube sobre *AWS*.
* ***Capítulo 3***. Métodos de análisis y pronóstico: se hablará acerca de los métodos y algoritmos que aplicaremos para realizar la predicción de los precios.
* ***Capítulo 4***. Funcionamiento del sistema: este apartado contendrá la explicación detallada de nuestro sistema.
* ***Capítulo 5***. Análisis de resultados: analizaremos varios ejemplos de los experimentos desarrollados.
* ***Capítulo 6***. Propuestas de mejora y conclusiones: se propondrán algunas propuestas para un trabajo futuro de nuestro proyecto que puedan ser interesantes de investigar y las conclusiones del trabajo en general.
* ***Anexos y bibliografía***: al final de la memoria se recogen los Anexos A, B, C y D que hablarán sobre “*Tipos de servicios de AWS*”, “*Especificaciones de las instancias EC2*”, “*Regiones y zonas de disponibilidad*” y “*Contenido del CD*” respectivamente. Finalmente se concluye la memoria con la bibliografía de interés consultada durante la elaboración del trabajo.

# **CARACTERÍSTICAS DEL ENTORNO DE APLICACIÓN**

En este capítulo, vamos a ver algunos de los conceptos básicos sobre la computación en la nube junto a las características más significativas de *Amazon Web Services*.

## **Cloud Computing**

### ***¿Qué es Cloud Computing?***

La informática en la nube nos proporciona una serie de servicios informáticos para el uso de numerosos recursos computacionales como bases de datos, servidores web, aplicaciones empresariales entre otras muchas cosas. De esta forma se pueden ahorrar muchos costes por todos estos servicios ya que permite pagar solo por la utilización de estos ajustándose a las necesidades de los usuarios (pago por uso), normalmente para negocios. Incluso los usuarios más comunes en la red hacen uso de todos estos servicios utilizando la gran diversidad de herramientas *software* disponibles que nos encontramos en la nube [2] [3] [27].

Algunos ejemplos de los servicios más conocidos son los que se mencionan a continuación:

* **Correo electrónico:** *Outlook o Gmail.*
* **Almacenamiento y compartición de ficheros:** *Dropbox, Google Drive, One Drive, Box.*
* **Gestores de tareas:** *Jira, Nozbe, Do It Tomorrow*.
* **Comercio electrónico:** *Amazon, Ebay.*
* **Antivirus:** *Baidu Antivirus, Panda Cloud Antivirus, HitmanPro*
* **Reproductores de música:** *CloudPlayer.*

Estos son algunos ejemplos de los más conocidos y utilizados por los usuarios, pero podemos encontrar una infinidad de ejemplos más [28].

En definidas cuentas, podemos definir *Cloud Computing* como un conjunto de recursos informáticos que el usuario tiene disponibles y puede acceder a ellos pagando o incluso gratis en cualquier momento y en cualquier lugar por medio de Internet [23].

### ***Características generales***

Una vez que conocemos el concepto de *Cloud Computing*, hablaremos acerca de las características más generales que diferencian la computación en la nube de la computación tradicional [3] [24]:

* **Pago por uso:** También llamado servicio bajo demanda. Los usuarios solicitan los servicios a través de la red pagando exclusivamente por el tiempo de uso de estos.
* **Acceso desde la red:** Todos los servicios están disponibles en Internet y los usuarios pueden acceder a ellos con un dispositivo con acceso a la red.
* **Escalabilidad:** Atendiendo a las necesidades de los usuarios, los recursos en la nube pueden subir o bajar la calidad o cantidad de estos dinámicamente.
* **Recursos compartidos:** Normalmente los recursos son reservas comunes para todos los usuarios (compartición de *hardware* y *software*) a no ser que se contraten servicios privados de la nube.
* **Servicio supervisado:** Aquellos recursos consumidos por los usuarios cuentan con un control automatizado por el proveedor en la nube de forma totalmente transparente para ellos.

### ***Modelos de despliegue***

Existen tres modelos de servicio o de despliegue en base a las necesidades que requieran los usuarios, en particular las empresas u organizaciones. Estos tres modelos son la nube pública, la nube privada y la nube híbrida. En la *Figura 4* se puede ver un ejemplo de estos tres modelos.

Se detallarán cada uno de estos despliegues junto con algunas ventajas y desventajas que presentan cada uno de ellos: [25]

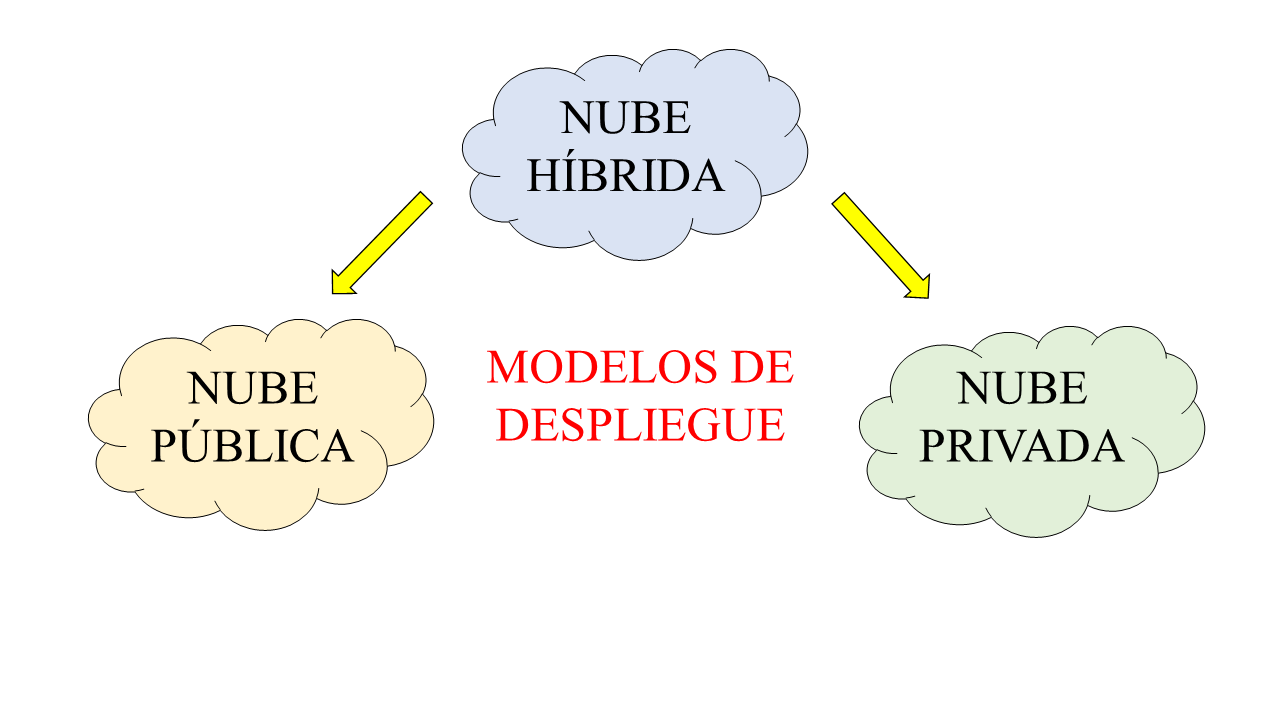


Figura 4. Modelos de despliegue

#### **Nubes públicas**

Este servicio se da cuando los usuarios, empresas u organizaciones llegan a compartir los recursos. Concretamente, un proveedor de un servicio ofrece ese mismo servicio a más de un usuario por lo que comparten un mismo recurso físico [21].

La compartición de recursos puede llevarse a cabo en disco o diferentes infraestructuras de red en un mismo servidor. Además, se requiere la aceptación por parte de un contrato con las condiciones propuestas por el proveedor [21] [25].

Los contratos con los que el proveedor y el cliente declaran el mutuo acuerdo de aceptación por ambas partes suelen ser innegociables por lo que hay que ceñirse a las condiciones que propone el proveedor. Excepcionalmente hay otras grandes organizaciones que pueden acordar otras políticas de uso [21] [3].

#### **Nubes privadas**

El segundo modelo de despliegue que se nos presenta es el de la nube privada. Este modelo tiene la particularidad de que a diferencia de la nube pública donde los recursos eran compartidos entre los clientes, en esta se ofrecen única y exclusivamente para uso propio o para nuestra empresa. Por tanto, los recursos que necesitemos están a nuestra plena disposición sin tener que preocuparnos por tener que compartirlo con nadie y por tanto ganaremos un importante incremento de flexibilidad y de libertad [3].

Los recursos privados que ofrece este modelo también pueden ser llamados como recursos dedicados porque todas las instalaciones que contempla son asumidas por un mismo usuario de la plataforma y por supuesto no suelen existir terceros de por medio [21].

Este tipo de contratos a diferencia de los de la nube pública son negociables total o parcialmente de cara al acuerdo del proveedor con el cliente o incluso hay empresas que pueden gestionar su propio servicio privado.

#### **Nubes híbridas**

El último de estos modelos de implementación en la nube del que hablaremos será la nube híbrida. La nube híbrida combina los servicios de la nube pública con los de la nube privada gestionando los servicios como una única administración. También puede considerarse como una aplicación privada con una mejora de servicios de la computación en la nube y en la infraestructura de esta para que así los clientes potenciales puedan mantener el control de las aplicaciones. Estas aplicaciones pueden estar relacionadas con la infraestructura entre los recursos que se ubican dentro y fuera de la nube [3].

Es importante tener en cuenta de cara a la contratación con el proveedor que la nube híbrida no exenta a los usuarios de pagar los costes de mantenimiento de la nube privada, pero sí que permite reducir la complejidad y el coste desde un punto de vista considerable.

Este modelo es un gran aliado de las empresas ya que, para un futuro próximo, los desarrollos del *software* serán gestionados mediante la nube privada a la vez que se podrán obtener recursos de los proveedores públicos más destacables [21].

Para terminar, existen otros dos modelos más además de los anteriores de los que hablaremos brevemente [25]:

* ***Nubes combinadas***: son una combinación desde dos o más nubes públicas o privadas que pueden ser gestionadas por proveedores y usuarios de manera que estos pueden modificar sus servicios ofrecidos por alguna de las nubes públicas de manera sencilla.
* ***Nubes comunitarias***: sirven para aquellas empresas que tienen negocios en común y llegan a un acuerdo para compartir un servicio de la computación en la nube de manera conjunta obteniendo así, mejor seguridad y privacidad.

### ***Tipos de contratación***

Empezaremos a hablar ahora acerca de los tipos de contratación o de servicio que dispone *Cloud Computing* para los usuarios donde podemos destacar tres tipos de opciones. Por un lado, tenemos *Software* como Servicio (*SaaS*), por otro Plataforma como Servicio (*PaaS*) y por último Infraestructura como Servicio (*IaaS*). Cada uno de estos modelos de servicio en la computación en la nube ofrece un determinado control, flexibilidad y gestión administrativa por diferentes métodos de implementación. Con el paso del tiempo, estos modelos se han ido desarrollando y evolucionando en función de las necesidades más cruciales de los usuarios. Para determinar en base a la necesidad o necesidades que se requieran para todos los usuarios, vamos a estudiar con detalle cada uno de estos tipos [3].

* + - 1. ***Software como Servicio (SaaS)***

El primer modelo que se nos presenta es el denominado *Software* como Servicio o *Software as a Service,* el cual, unos proveedores de *software* ofrecen a los usuarios los servicios desarrollados por ellos atendiendo a sus necesidades y que además se encargan de ejecutar y administrar estos servicios. Dicho de otro modo, este *Software* como Servicio, podríamos considerarlo como aplicaciones de usuario finales donde dichos usuarios no tienen por qué preocuparse de administrar ni de cómo gestionar el servicio de la infraestructura que contempla, tan solo en saber cómo utilizar, o mejor dicho consumir el *software* en condiciones [21].

Para este modelo, el usuario paga por la utilización, así como por el almacenamiento, la seguridad, la localización, entre otras cosas, para que las aplicaciones funcionen sin problemas a excepción de algunas configuraciones básicas que sí han de controlar, como el funcionamiento de la herramienta en sí [25] [26].

Normalmente este tipo de aplicaciones suelen ser de correo electrónico, para comunicaciones, videojuegos, etc.

Entre los ejemplos más destacados de *SaaS* podemos encontrar: *Google Aps*, *Gmail*, *Moodle*, *WordPress*, *Dropbox*, etc.

* + - 1. ***Plataforma como Servicio (PaaS)***

El segundo modelo de servicio del que hablaremos con detalle será la Plataforma como Servicio (*PaaS*), el cual presenta algunas diferencias claras en comparación con el anterior. La principal diferencia que presenta la Plataforma como Servicio es que permite a los usuarios desarrollar aplicaciones por ellos mismos para así mejorar el nivel de producción en la computación en la nube, ya que el modelo anterior solo permitía hacer uso de los servicios. Por otro lado, los usuarios no podrán encargarse de gestionar la parte de la infraestructura, aunque sí podrán utilizar las aplicaciones disponibles acerca de las configuraciones sobre las herramientas que necesite [21].

El objetivo de este modelo permite mejorar la eficiencia de las aplicaciones ya que el usuario simplemente ha de ceñirse a la parte del desarrollo e implementación sin tener que preocuparse por la parte del almacenamiento, mantenimiento, recursos ni nada por el estilo [25] [26].

Las aplicaciones que suelen cubrir la Plataforma como Servicio son la implementación de herramientas, bases de datos, aplicaciones para inteligencia artificial, etc.

Entre algunos de los ejemplos más comunes podemos contemplar: *Velneo*, *Abiquo.com*, *Windows Azure* (aunque también es de *IaaS*), *Google App Engine* y *Amazon* *beanstalk2* y muchos más.

* + - 1. ***Infraestructura como Servicio (IaaS)***

El último de estos servicios es la Infraestructura como Servicio (*IaaS*) que contempla todo lo relacionado con el desarrollo de las TIC. Se utilizan servidores externos en lugar de tener uno ya centralizado dentro de nuestra organización o incluso optar por simplemente gestionar un centro de datos particular. Se podría decir que el usuario paga por el uso de los recursos necesarios virtuales ya sea capacidad de cómputo, almacenamiento de las bases de datos, velocidad o tiempo de procesamiento de la CPU [21].

Con este tipo de servicio, los usuarios pueden reducir sus problemas de gestión e incluso pueden llegar a obtener un importante ahorro en costes ya que simplemente pagan por los recursos que desean utilizar atendiendo a las restricciones de los proveedores.

Además, tiene otra gran ventaja la cual permite obtener un grado de escalabilidad óptimo puesto que el usuario puede contratar nuevos recursos de los servicios disponibles en base a sus necesidades [25].

Este quizás sea el tipo de contratación más importante de las tres que hemos visto porque permite al usuario obtener recursos a un nivel económico adecuado. Además, puede adquirir una gran variedad de estos recursos atendiendo a sus necesidades y pagando por ellas según el uso que requieran emplear [26].

El pilar fundamental de este trabajo que es *Amazon Web Services* y del que hablaremos en capítulos posteriores se incluye dentro de esta modalidad.

Dentro de los servicios que la Infraestructura como Servicio ofrece podemos destacar las máquinas virtuales, comunicaciones de red, cortafuegos, etc.

Además de *Amazon Web Services,* también podemos añadir otros ejemplos de este modelo como *Google Compute Engine*, *Microsoft Azure,* *IBM Cloud*, *Rackspace*, entre otra gran variedad de ejemplos.

Cualquiera de estos tres modelos de contratación que ofrece *Cloud Computing* nos aporta una infinidad de ventajas con respecto a los servicios más comunes que no podemos disfrutar en la nube. En la *Figura 5*, podemos apreciar una ilustración gráfica acerca de los diferentes ejemplos sobre los tipos de contratación.

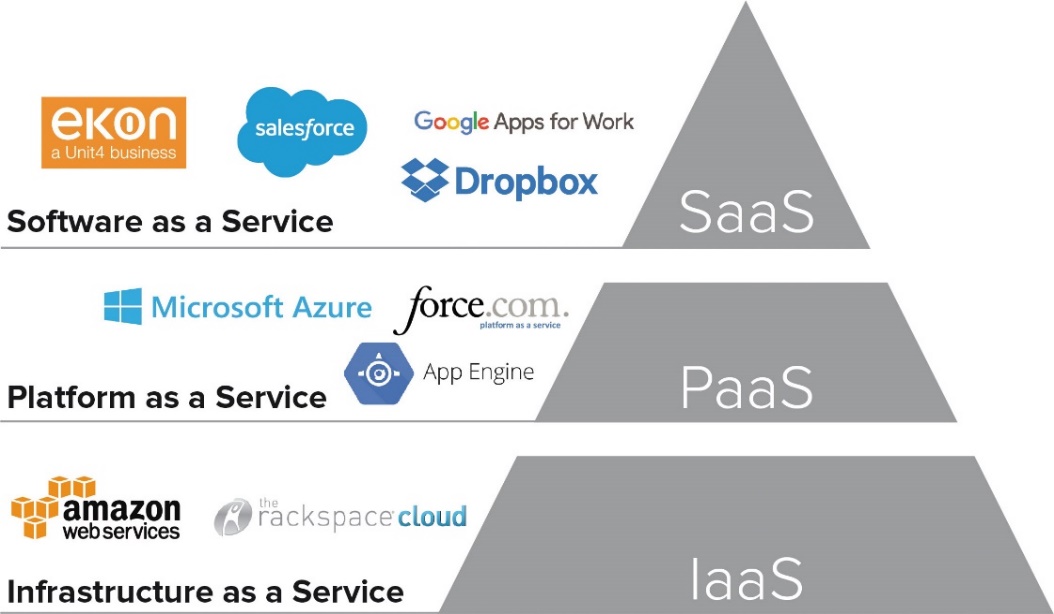


Figura 5. Tipos de contratación [22]

## **Amazon Web Services**

### ***¿Qué es Amazon Web Services?***

Hemos estudiado los conceptos básicos que compone *Cloud Computing* junto con algunos ejemplos de los servicios más comunes que nos podemos encontrar en este mismo capítulo. Nuestro entorno de trabajo estará guiado por una plataforma ya archiconocida por la gran mayoría de los usuarios en la nube. Además, al ser un servicio que aporta grandes rangos de fiabilidad, seguridad y confianza a los consumidores, toma la primera posición en la cúspide de los mercados. A lo largo del tiempo, desde el año 2006 que empezó a desplegar sus servicios, ha adquirido un importante nivel de popularidad y fama debido a la confortabilidad y satisfacción de sus potenciales clientes, esta plataforma es *Amazon Web Services* o *AWS* [18] [29].

*AWS* ofrece servicios de infraestructura web ubicados en la nube donde gracias a esta mecánica, los usuarios pueden obtener todo tipo de recursos de IT, servidores o almacenamiento de forma rápida sin tener que reservar dichos servicios previamente. Está basado en el modelo de compra *pay-as-you-go* que quiere decir que un usuario paga por lo que va a utilizar en lugar de preocuparse por el coste de la instalación del servicio o del mantenimiento, como veíamos entre las ventajas que presentaba *Cloud Computing*. Como ya hemos comentado, *AWS* ocupa el nivel más alto entre las plataformas desplegables en la nube ofreciendo sus servicios a más de un millón de clientes con negocios esparcidos por todo el mundo de hasta 190 países diferentes [30].

En sus inicios, *Amazon* ofrecía productos orientados al comercio electrónico como *CDs*, libros físicos y digitales, electrodomésticos, dispositivos electrónicos, etc. Este tipo de negocio supuso un gran impacto económico para *Amazon*. Esto era debido al pico de demanda de sus centros dimensionales que se producían en épocas navideñas mientras que el resto del año disponía de muchos recursos ociosos y generaban muchos costes de mantenimiento.

*AWS* oferta recursos aptos para informática relacionados con el mundo empresarial y para los usuarios. Así nació *Amazon Web Services* apoyándose en el modelo de contratación de infraestructura como servicio (*IaaS*), que como ya hemos visto anteriormente, utiliza servidores interconectados por redes que ofrecen acceso a los recursos disponibles para los usuarios.

Hoy en día son muchas las organizaciones y sobre todo *startups* y jóvenes emprendedores que deciden abrir las puertas de su negocio gracias a los recursos que oferta *AWS* y debido a los numerosos beneficios económicos que conlleva invertir sus planes de proyectos en esta plataforma. Por ejemplo, la factura de una empresa que cubre sus servicios y gastos con *AWS* puede llegar a ahorrar hasta más de un 90% el precio de sus acciones que si invierte por cuenta propia los gastos de instalación de su propia infraestructura. Incluso se ahorra contratar personal encargado de mantener estos servicios del cual se encargan los propios proveedores, siempre y cuando disponga de personal adecuado. *Amazon Web Services* es una plataforma que actúa como proveedor de servicios web, dentro de ellos, hablaremos con detalle acerca de ***Amazon* *Elastic Compute Cloud* (*EC2*)** [31].

### ***Infraestructura interna***

En este apartado vamos a ver los diferentes medios de acceso a estos servicios que *AWS* dispone y presenta a los usuarios. También veremos con detenimiento los servicios más destacados de *AWS* cuya utilidad está dividida atendiendo a la categoría, como la capacidad de cómputo, de almacenamiento, los servicios de red, etc. Entraremos en profundidad a hablar acerca del servicio de *Amazon EC2* ya que dispone de las instancias con las que trabajaremos en el Capítulo 3 de este documento asociado a la experimentación del *TFG*. Intentaremos enfocar y compaginar el uso de todos estos servicios que tienen relación con *Amazon EC2,* ya que no tienen por qué ser independientes y además algunos son necesarios para poder realizar sus funciones [32] [33].

#### **Tipos de acceso**

*AWS* ofrece tres posibles formas para acceder a este tipo de servicios para sus usuarios [34]:

* **Interfaz de línea de comandos**: gracias a una interfaz sencilla que presenta esta herramienta, los usuarios podrán hacer uso de los servicios de *AWS* por medio de línea de comandos.
* **Kit de desarrollo *software*:** los usuarios también pueden acceder por medio de *SDKs* que ofrecen una *API* compatible con el lenguaje de programación que utilicen para implementar y diseñar sus aplicaciones con *AWS*. Por ejemplo, *AWS CodeCommit* para facilitar la gestión de repositorios como *Git* o *AWS CodePipeline* para realizar actualizaciones tras cada modificación de código.
* **Interfaz de usuario**: la interfaz de usuario (*AWS Management Console*) es una herramienta web que permite a los usuarios visualizar los recursos disponibles en la nube o buscar los servicios adecuados a sus condiciones. También permite gestionar la configuración de la cuenta de *AWS* o sus credenciales (más adelante las utilizaremos) incluso desde la aplicación diseñada para dispositivos móviles.

#### **Amazon Elastic Compute Cloud (EC2)**

Para finalizar esta sección, vamos a definir qué es *Amazon EC2* y cuál es su principal función y los beneficios que ofrece. Más adelante hablaremos acerca de las instancias que este servicio ofrece para estudiar los tipos y las características de estas.

*Amazon EC2* es uno de los servicios web de *AWS* que permite ofrecer capacidad de cómputo y almacenamiento escalable en la nube a un reducido coste por medio de instancias. Estas instancias de tamaño flexible son servidores privados virtuales que funcionan sin necesidad de invertir en *hardware* o en infraestructura para la adquisición de los mismos [30].

Con *Amazon EC2* los usuarios cuentan con la ventaja de que éste se encargará de desplegar los servidores virtuales a las máquinas *host* donde los usuarios podrán ejecutar las aplicaciones desde la nube [31].

Proporciona una interfaz web con información sobre los recursos computacionales disponibles para poder obtener las instancias en apenas unos minutos. Además, gracias al modelo *pay-as-you-go* muy característico de la computación en la nube, los usuarios pagarán solamente por lo que usarán [32] [20].

Los beneficios que presenta contratar los servicios de las instancias *EC2* son los siguientes [19] [33]:

* **Escalabilidad**: los usuarios pueden contratar una o varias instancias de servicio al mismo tiempo, pudiendo aumentar o disminuir el tiempo de uso por el que contratarán dichas instancias, ya que el pago se efectúa por el uso por hora en función del tiempo que requiera el usuario.
* **Control**: los usuarios contarán los privilegios de administrador *root* sobre sus instancias pudiendo acceder e interactuar libremente con cada una de ellas. Pueden hacer varias operaciones funcionales con las instancias como reiniciarlas incluso de forma remota con las *APIs* disponibles de *AWS* o incluso detenerlas.
* **Flexibilidad**: los usuarios también tendrán la opción de elegir la instancia que mejor le convenga según las necesidades, ya que existen varios tipos de instancia (como veremos en los próximos apartados) de acuerdo con unas determinadas especificaciones computacionales como la configuración de la memoria, la CPU, almacenamiento o incluso para un sistema operativo en concreto.
* **Integración**: *Amazon EC2* es equiparable y puede estar incorporado en otros servicios de almacenamiento de *AWS* como *Amazon S3*, *Amazon RD*S o *Amazon VPC* necesarios para conseguir un funcionamiento adecuado.
* **Asequibilidad**: el precio a pagar por cada una de las instancias será relativamente bajo al pagar simplemente por el uso que efectúen los usuarios sobre dichas instancias.
* **Fiabilidad**: los servicios de las instancias que se llevan a cabo en los centros de datos tienen un alto rango de confianza en relación con los usuarios a la hora de ofrecer las instancias atendiendo a las necesidades de cada uno de ellos.
* **Seguridad**: este beneficio es una de las mayores preocupaciones de *AWS* ya que ha de ofrecer sus servicios de la forma más fiable posible para ganarse la confianza de los usuarios, quedando estos bien protegidos ante cualquier problema de seguridad. Para establecer un plan de seguridad óptimo, *EC2* trabaja a su vez con *VPN* para asegurar la protección de los recursos que se ofrecen a los clientes.
* **Usabilidad**: otra de las ventajas con la que cuenta *EC2* es que a la hora de comenzar a utilizar sus servicios ofrece varias formas de acceso rápido y fácil para los usuarios menos expertos en el ámbito del uso de las tecnologías de la información en la nube como las que ya hemos visto en el apartado *2.2.2.1*.

En la *Figura 6*, podemos ver cómo están relacionados algunos de los servicios de *AWS* con *Amazon EC2.* Este servicio suele trabajar con otros servicios de colas que necesitan de una base de almacenamiento como *EBS* o *S3*. El usuario puede interactuar con estos servicios por medio de *AWS Management Console*. También utiliza el servicio de monitorización de *CloudWatch* [20].

Una vez estudiado con detenimiento cuáles son las bases en las que se apoya *Amazon Web Services* y *Amazon EC2*, podremos empezar a hablar acerca de las instancias que este último dispone para los usuarios y cómo se organizan atendiendo a la infraestructura global que presentan.

Para entender mejor cuál es el funcionamiento de los servicios más importantes para *Amazon EC2*, puede consultarse el Anexo A al final de este documento.

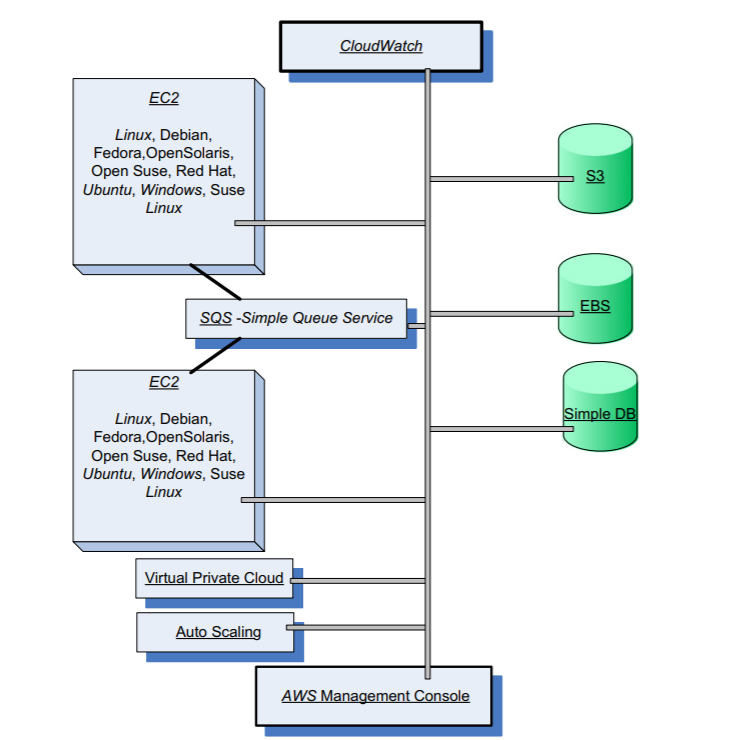
Con esto empezaremos a entrar en la parte más técnica del trabajo y donde se verán conceptos relacionados acerca de las técnicas de predicción de precios que posteriormente desarrollaremos para finalizar toda la parte relacionada con el estado del arte.

Figura 6. Infraestructura Amazon EC2 [19]

### ***Instancias de Amazon EC2***

*Amazon EC2* facilita a los usuarios un acceso mediante aplicación web donde se ofertan una serie de instancias para dichos usuarios. Para entender mejor la dinámica de las instancias, vamos a ver en el siguiente apartado qué son las imágenes de máquina de *Amazon* (*AMIs)* y su relación con las instancias [39].

#### **¿Qué son las instancias de Amazon EC2?**

Una instancia es un servidor privado virtual ubicado en la nube con características informáticas que son utilizadas para ayudar a dar soporte *hardware* a máquinas locales. Dichas instancias pueden ser lanzadas por medio de las *AMIs*. Estas imágenes están compuestas por una serie de elementos *software* como un sistema operativo, las aplicaciones que utiliza y sus servidores de gestión asociados dispuestas para uso público. Desde una única *AMI* se pueden lanzar mínimo una instancia o copia virtual de la *AMI* que actúa como un servidor virtual en cada uno de los *hosts* en caso de que se requiera varias instancias con la misma configuración. En caso contrario, si se requieren instancias de diferentes configuraciones, se pueden hacer uso de otras *AMIs* sin problemas y de este modo tener varias instancias de diferentes imágenes de máquina. La *Figura 7* muestra el comportamiento de estas *AMIs* cuando lanzan a ejecución las copias o instancias [39] [42].

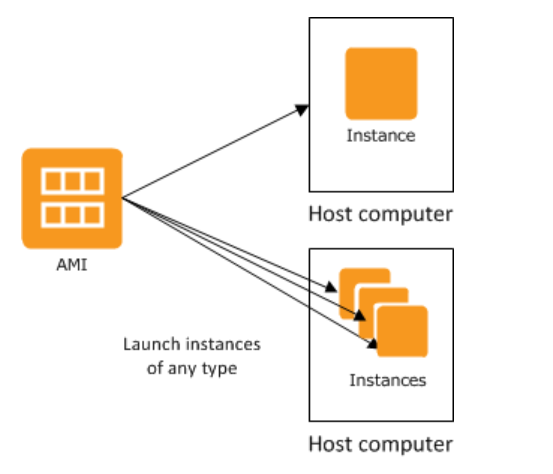
Las *AMIs* también cuentan con un control de cuentas de usuarios de *AWS*. También existe la posibilidad de crear *AMIs* personalizadas.

Figura 7. Lanzamiento de una instancia [39]

De cara al ciclo de vida de una instancia, esta puede llegar a pasar por varios estados, tal como muestra la *Figura 8*. Puede pasar por el estado de **pendiente** si es lanzada por primera vez desde la *AMI* hasta llegar a ejecutarse. En **ejecución** si están trabajando en un determinado *host*. **Detenidas o interrumpidas** en caso de fallo o provocadas por el sistema o **finalizadas** si termina el periodo de contratación de la instancia. Aunque pueden volver a lanzarse nuevamente desde la misma *AMI* [40] [42].

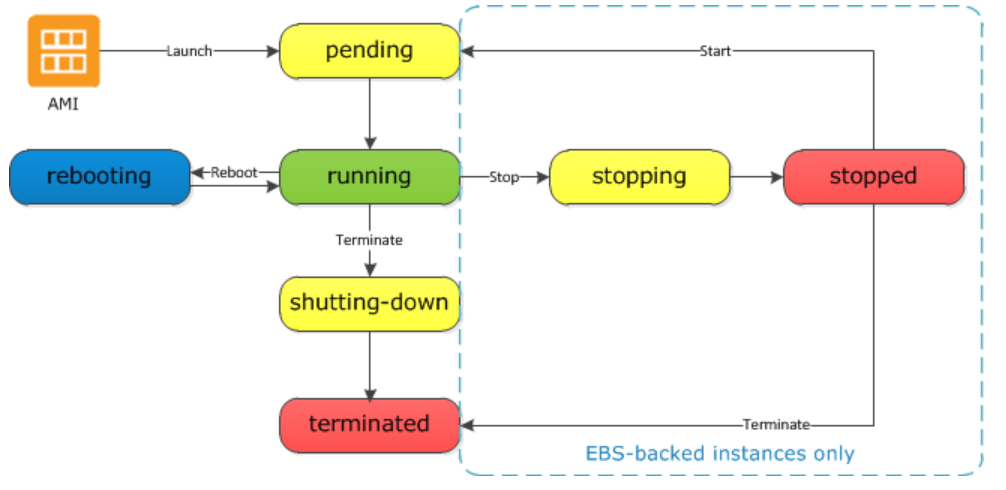


Figura 8. Ciclo de vida de una instancia [40]

Una vez que la instancia es asignada a un *host*, esta actúa como si se tratase de una máquina local normal en base a sus características computacionales o de almacenamiento, pero desde la nube.

Si el usuario ya no necesita los servicios de instancia de una *AMI*, puede darla de baja fácilmente.

Cuando un usuario solicita acceso a una de estas instancias, lo hace en función de los siguientes parámetros [41]:

* *Tipo de instancia*: las instancias se dividen en varios tipos atendiendo a la categoría o familia a las que pertenece en base a sus características computacionales específicas. Más tarde estudiaremos estos tipos y características de instancias con mayor profundidad.
* *Región y zona de disponibilidad*: las instancias pueden estar alojadas en diferentes regiones por todo el mundo, y cada una de estas dispone de varias zonas de disponibilidad, tal y como se explican en el Capítulo 1. Luego veremos cómo se organiza esta infraestructura global.
* *Número de instancias*: como ya hemos mencionado, un usuario puede contratar más de una instancia para su *host* sin problema.
* *Precio*: los usuarios han de pagar por las instancias en función del uso que les den a estas. Los precios varían según el tipo de instancia que contraten.

#### **Opciones de contratación**

Ya conocido cual es el funcionamiento de las instancias virtuales de *Amazon EC2*, vamos a ver ahora que tipos de modalidad de contratación tenemos a la hora de adquirir las instancias. Destacamos cuatro modelos de contratación u opciones de compra que describiremos para que los usuarios puedan seleccionar a su gusto cuál de estos tipos le viene mejor para obtener en su *host* una o varias de instancias que desee [37] [38]:

* ***Instancias puntuales (spot)***: las instancias de este tipo de contratación, también llamadas a partir de ahora *spot,* son nuestro primordial objetivo con las que trabajaremos en la parte de experimentación del trabajo. Permite a los usuarios pujar por la capacidad computacional de los recursos disponibles para una instancia ofrecida por el sistema.

Tienen el inconveniente de que no ofrecen ninguna garantía de que la disponibilidad de estos recursos sea permanente con el tiempo. Esto es porque el sistema se encargará automáticamente de cambiar el precio mínimo a pagar por el uso de estas instancias de forma dinámica en función de la disponibilidad de esos recursos.

Si la disponibilidad de los recursos es lo suficientemente abundante, los usuarios podrán pujar por adquirir recursos de la instancia que demandan. Pero el precio mínimo por hora que pujan los usuarios por una instancia (también llamado *bid*) ha de ser mayor o igual al precio que indica el sistema para llegar a adquirir dicha instancia para su máquina [39].

En caso de que el sistema compruebe que sus recursos se están llenando, el sistema subirá el precio de las instancias obligando a los usuarios a subir la puja de la instancia si quieren mantenerla activa. Si el sistema llega a subir el precio por hora de una instancia *spot* a un precio mayor que el que estamos pagando, el sistema nos informará con un preaviso de dos minutos para hacernos saber que nuestra instancia se interrumpirá si no igualamos o superamos el precio impuesto por el sistema [39].

Este tipo de instancias suelen emplearse para aplicaciones con capacidad intrínseca de adaptación para adaptarse a la ejecución con recursos variables y pueden llegar a aparecer y desaparecer durante el tiempo de vida del servicio.

* ***Instancias reservadas***: con este modo de contratación, los usuarios podrán reservar las instancias durante un periodo de tiempo que soliciten sin tener que preocuparse de la interrupción de dichas instancias. Por tanto, una vez pagada la instancia durante el tiempo que la usarán, tendrán acceso garantizado a ella (no han de preocuparse por el cambio del precio periódico que indicaban las instancias *spot*). El coste asociado a estas instancias es hasta un 75% más barato que las instancias bajo demanda. El tipo de aplicaciones que usan esta modalidad de pago está pensado para aquellas que requieran funcionamiento a largo plazo.
* ***Instancias bajo demanda (on-demand)***: tienen un coste también muy alto y además los usuarios no tienen acceso garantizado a ellas, es decir, obtienen la instancia una vez que esté disponible, aunque tampoco pueden llegar a ser interrumpidas una vez que se adquieran. No será necesario pagar inicialmente por el uso de las instancias y está pensado para aplicaciones con uso a corto plazo.

El coste por hora por el uso de las instancias *spot* suele ser inferior al coste por hora de las instancias reservadas y bajo demanda (hasta un 90% más barato que las instancias bajo demanda).

* ***Hosts dedicados***: los *hosts* dedicados permiten a los usuarios utilizar licencias de *software* vinculadas a estos servidores aptos para almacenamiento de instancias *EC2*. Se paga por la hora que se dedique a un *host* dedicado que está activado, pero no por el uso de este.

Como podemos comprobar, cada una de estas opciones tienen sus claras ventajas y sus inconvenientes, pero sobre todo puede causar controversia entre los usuarios a la hora de elegir la opción de compra. Si reserva una instancia, podrá tenerla el tiempo que desee, pero le saldrá muy caro, si decide pujar por una instancia *spot*, le saldrá más barato, pero se arriesga a que llegue a ser interrumpida por el sistema en algún momento [35] [36].

#### **Categorías y tipos**

Para finalizar este apartado dedicado a las instancias de *Amazon EC2* vamos a ver qué tipos de instancias se nos presentan atendiendo a las características de cómputo y de almacenamiento que poseen a la hora de adoptarse a las necesidades de los usuarios. Estas instancias están compuestas por la unión de varios recursos optimizados. En función de lo que los consumidores requieran, se orientarán por un tipo de instancia o de otra. Estas instancias se clasifican en cinco familias o categorías para englobarlas de manera general. Vamos a ver pues cuáles son cada una de estas categorías y las instancias que tienen cada una de ellas.

Todas y cada una de las instancias tienen diversos tipos de tamaño. Por ejemplo, de la primera instancia de la que hablaremos podemos destacar los modelos t3 en ***nano*, *micro*, *small*, *medium*, *large*, *xlarge* o *2xlarge***. Conforme mayor sea el tamaño de la instancia en base a sus especificaciones computacionales, mejores serán las especificaciones informáticas, desde *nano* hasta *2xlarge* o incluso en algunas instancias hasta *24xlarge* [35] [36].

##### **USO GENERAL**

Dentro de la categoría de uso general nos encontramos con cuatro tipos de instancias, T3, T2, M4 Y M5. Estas instancias son aptas para el uso de bases de datos no muy extensas de tamaño pequeño y mediano, para procesos cuya utilización requiere memoria externa, memoria caché, servidores para control de versiones, micro servidores y la utilización de *clústers* entre otras cosas [35] [36].

##### **OPTIMIZACIÓN INFORMÁTICA**

Las instancias para optimización informática están reguladas para el uso de recursos informáticos con rendimiento óptimo, servicios web, técnicas y aplicaciones para inteligencia artificial, procesamiento basado en lotes e incluso para investigación y modelado científico e ingeniería. Dentro de esta categoría nos encontramos con instancias de tipo C5 y C4 [35] [36].

##### **OPTIMIZADAS PARA MEMORIA**

En esta categoría nos encontramos con instancias que trabajan con bases de datos medianas y de alto rendimiento y relacionales, análisis de inteligencia artificial como *Big Data* o para el uso de aplicaciones empresariales con grandes volúmenes de datos. En base a estos requisitos podemos apreciar instancias de tipo R5, R4, X1e, X1, de memoria elevada y Z1d [35] [36].

##### **INFORMÁTICA ACELERADA**

Para informática acelerada podemos destacar el uso de memorias de alto rendimiento para gráficos, renderización de imágenes en 3D, texturas avanzadas, técnicas de modelado molecular, grandes bases de datos, dinámica computacional mejorada y técnicas de inteligencia artificial como aprendizaje automático y mecánicas de análisis para financia empresarial. Las instancias que entran dentro de esta familia son P3, P2, G3, y F1 [35] [36].

##### **OPTIMIZADAS PARA ALMACENAMIENTO**

Por último, la familia de instancias optimizadas para almacenamiento como su nombre indica aportarán una gran eficacia para bases de datos de un tamaño muy grande, memoria, BBDD transaccionales y técnicas de análisis, procesado y procesado de datos a gran escala. Estas instancias son H1, I3 y D2 [35] [36].

Para más información acerca de las especificaciones de las instancias de *Amazon EC2* se puede consultar la página oficial de *AWS* y al final de este documento en el Anexo B.

### ***Infraestructura global***

Vamos a comentar algunos conceptos relacionados con la estructura global que *AWS* presenta alrededor del mundo. Este apartado es una clave de información muy importante para los usuarios puesto que han de conocer dónde pueden desplegar sus instancias atendiendo al centro de datos que se encargue de gestionar la instancia en concreto.

Las instancias de las que hemos estado hablando hasta ahora, se encuentran dispersadas por diferentes regiones independientes unas de otras repartidas globalmente por cuatro de los cinco continentes.

Estas regiones son localizaciones geográficas compuestas por un conjunto de zonas de disponibilidad las cuales representan los diversos puntos o centros de datos ubicados en cada una de las regiones. Al ser regiones independientes, los recursos están asociados a cada región específica [43].

Las zonas de disponibilidad o *data centers* que dividen a cada región, están conectadas a uno o hasta cinco centros de datos, sin embargo, dos zonas de disponibilidad no pueden compartir un mismo centro de datos, tal y como puede visualizarse en la *Figura 9*.

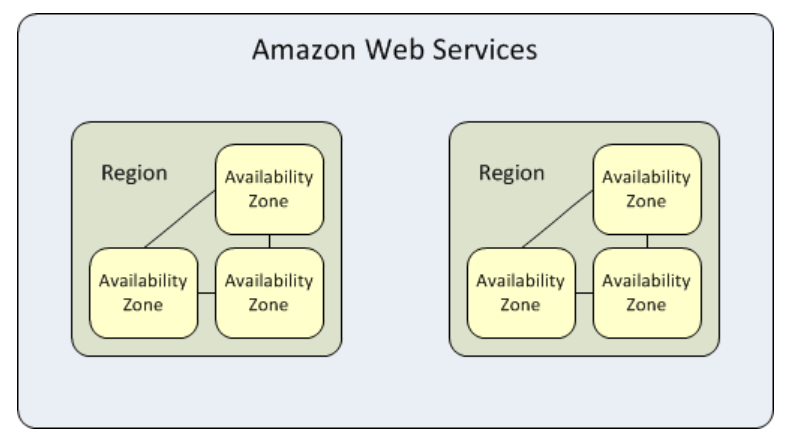


Figura 9. Configuración de las regiones [44]

Normalmente las zonas de disponibilidad se suelen anotar con el nombre de la región seguido de una letra. Por ejemplo, para la región de *us-east-1*, nos encontramos con seis zonas de disponibilidad denominadas por *us-east-1a, us-east-1b, us-east-1c, us-east-1d, us-east-1e* y *us-east-1f.* Así podemos denotar cada zona de disponibilidad de forma aislada por medio de la letra que representa dentro de su región [44].

Próximamente, se desarrollará una nueva región en África y así los cinco continentes del planeta contarán con instancias virtuales de *Amazon EC2*.

Para nuestro trabajo, los usuarios pueden elegir diversos tipos de instancias *spot* diferentes repartidas por diecinueve regiones con unas cincuenta y cinco zonas de disponibilidad y estas pueden ser ejecutadas hasta en dos sistemas operativos (*Linux* y *Windows*), aunque *Linux* ofrece dos distribuciones diferentes (*SUSE Linux* o *Unix*).

En el mapa de la *Figura 10*, podemos apreciar gráficamente cuales son las regiones disponibles y que aparecen rodeadas con un circulo naranja. El número de dentro representa el número de zonas de disponibilidad que tiene cada región y las que no tienen número representan aquellas que están en proceso de serlo.

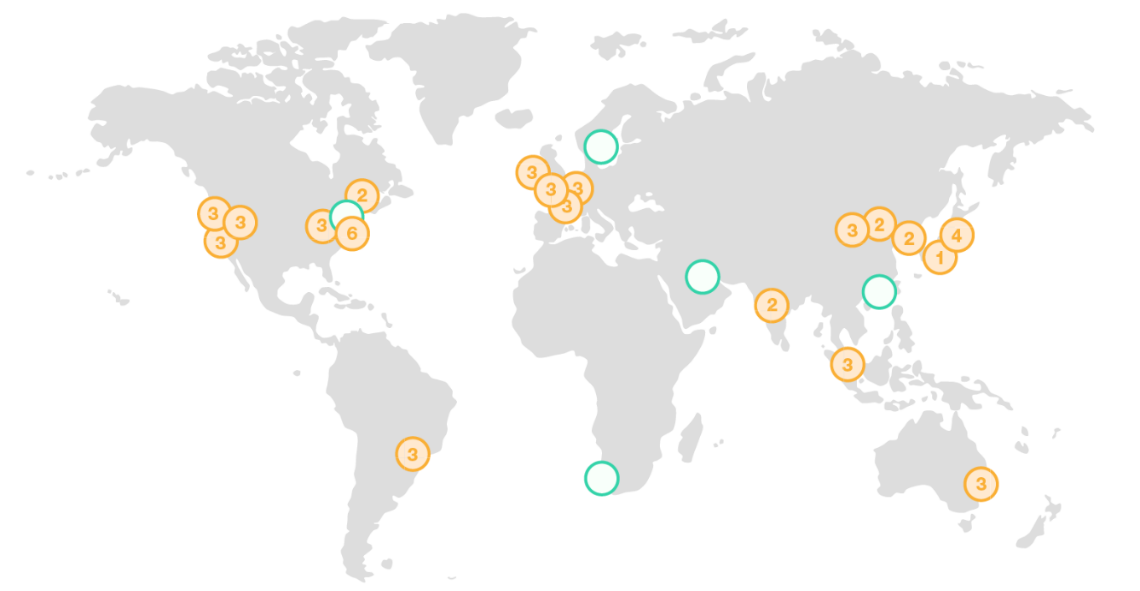


Figura 10. Mapa de regiones [44]

En el Anexo C, tendremos la información sobre cada una de las regiones repartidas por el mundo y las zonas de disponibilidad asociadas con mayor detalle [44].

# **MÉTODOS DE ANÁLISIS Y PRONÓSTICO**

En este capítulo, hablaremos acerca de las medidas de pronóstico. Haremos una breve introducción sobre las series temporales y detallaremos los algoritmos que necesitaremos para realizar el pronóstico. Para ello, podemos visualizar la *Figura 11* para conocer mejor el proceso que necesitaremos para llevar a cabo el análisis de predicción de nuestro sistema.

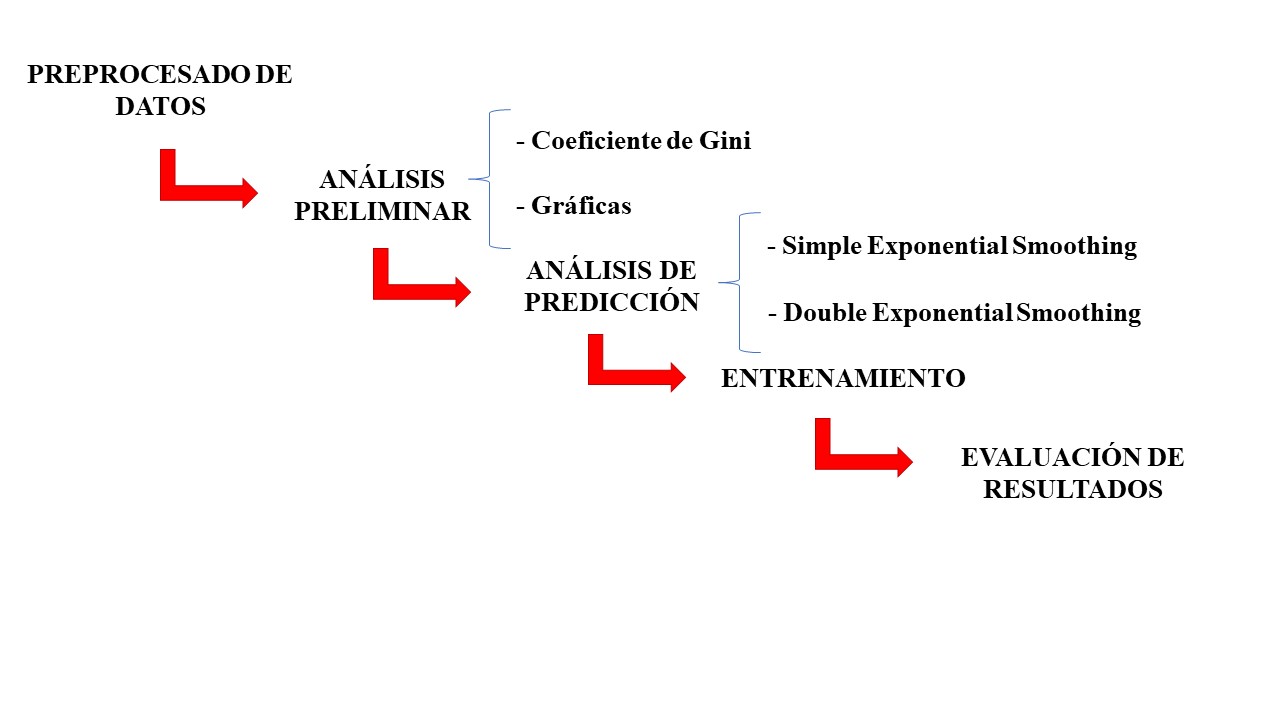


Figura 11. Metodología de Análisis

## **Medidas de pronóstico y series temporales**

En este capítulo, hablaremos acerca de los métodos de análisis y pronóstico que vamos a emplear para llevar a cabo una estimación del precio *bid* más apropiado para los consumidores de instancias de *Amazon EC2.*

Una vez que ya conocemos los conceptos básicos de *Cloud Computing* y las instancias de *AWS*, vamos a desarrollar uno de los subobjetivos del *TFG*, que será hacer un análisis de datos mediante una serie de técnicas y algoritmos. Recordamos que de entre los tipos de contratación que ofrece *Amazon EC2* podemos encontrarnos con instancias reservadas, instancias bajo demanda, *host* dedicados o instancias *spot*.

Normalmente, los usuarios con negocios de *start-ups* no expertos y que acaban de comenzar a ser emprendedores, no suelen conocer muy bien cuáles son las ventajas de contratar servicios en la nube. Es importante conocer los beneficios que obtienen al decidir contratar los servicios de la nube de *Amazon* para instalar sus propios servidores virtuales y así ahorrar los costes que supone la instalación y el mantenimiento de estos. Como ya hemos visto también, existen numerosos tipos y familias de instancias del servicio de *Amazon EC2*, cada una con unas especificaciones computacionales particulares en función de las necesidades de los consumidores [45].

Hemos comentado que las instancias reservadas tenían la ventaja de que una vez que eran contratadas durante un periodo de tiempo, el sistema no las podía interrumpir como con las de tipo *spot* y por tanto se les podía dar uso durante el tiempo de contratación sin problemas. Ahora bien, estas instancias también tienen un precio por hora superior al de las instancias *spot*, pero el problema que presentaban estas últimas es que el precio variaba dinámicamente. Por tanto, los usuarios tenían que pujar por ellas atendiendo al precio que el sistema actualizaba arriesgando así que si el precio que pujaban (*bid*) era inferior al que la máquina establecía, les interrumpía las instancias y supone un riesgo importante de cara al negocio.

Una de las posibilidades que tenemos para minimizar la probabilidad de que las instancias *spot* sean interrumpidas, sería tratar de predecir el precio de subasta adecuado. Para ello, podemos hacer uso de técnicas de inteligencia artificial y análisis de datos.

En el mundo de la inteligencia artificial podemos encontrarnos una infinidad de tipos de datos y algoritmos que han facilitado la vida a la sociedad desde los últimos años hasta la actualidad, gracias a que están desarrollados para dar resultados con soluciones óptimas. Las técnicas de inteligencia artificial y análisis de datos se han convertido en un elemento clave para nuestro día a día. No cabe la menor duda de que el futuro que nos depara estará plagado de muchas más técnicas y algoritmos sofisticados y mejorados con un gran rendimiento computacional [45].

Existen numerosos tipos de técnicas de aprendizaje automático que funcionan en muchas aplicaciones que ofrece la inteligencia artificial. Para nuestro caso, nos interesa hablar acerca de los algoritmos de predicción que se pueden emplear para el estudio de series temporales. Estos algoritmos son muy utilizados en el ámbito de *Big Data*. Uno de los más utilizados es el de regresión.

Antes de comenzar a hablar de los algoritmos de predicción, vamos a comentar algunos conceptos básicos sobre series temporales que debemos de conocer.

Cuando hablamos de una **serie temporal**, nos referimos a un conjunto de datos u observaciones secuenciales obtenidas a lo largo de un periodo determinado de tiempo. Dicho de otra forma, una serie temporal, también llamada periodo, son muestras de un valor obtenidas a intervalos de tiempo regulares. Es importante que se respete el orden temporal de estas observaciones de forma secuencial para establecer las medidas estadísticas adecuadas. Las series temporales pueden ser discretas o continuas [52].

Entre esas medidas estadísticas podemos apreciar el nivel de **tendencia** de los datos, que utilizaremos posteriormente para nuestros casos de estudio. También se pueden estudiar medidas de **estacionalidad** o periodicidad entre ciclos que se repiten a lo largo del tiempo. La tendencia es el suavizado de la pendiente de un periodo serie en un tiempo prolongado, mientras que la estacionalidad, son oscilaciones que se repiten durante un determinado periodo continuamente.

Para estudiar las series temporales de tiempo, lo primero que se ha de hacer es esbozar la gráfica correspondiente para analizar algunas de las medidas que hemos comentado anteriormente. Después, para nuestro caso, como queremos hacer una **predicción** de valores futuros, debemos de tener en cuenta el comportamiento de los datos pasados para realizar una predicción de los datos futuros.

Cabe mencionar que los algoritmos de predicción no son cien por cien fiables. Hoy en día es imposible realizar predicciones exactas ya que hablamos de cifras estadísticas, cálculos relativos y aproximaciones. Por tanto, a la hora de evaluar los resultados, no obtenemos quizás los esperados. A pesar de ello, sí que nos ayudan a estimar valores cuando trabajamos con datos muy cercanos entre ellos y con poca dispersión o variabilidad entre ellos. Nos enfocaremos en técnicas de aplicación de minería de datos como el **entrenamiento** y la **evaluación** de los datos de los que luego hablaremos detenidamente. El entrenamiento se aplica sobre un conjunto de datos, para obtener un **modelo** óptimo (configuración de parámetros) para un algoritmo que posteriormente se evaluará con el **conjunto de prueba o de *test***.

En los próximos apartados detallaremos con más información el funcionamiento de estos algoritmos y en concreto hablaremos acerca de las técnicas de ponderado exponencial [52].

Es de esperar que estos métodos sean muy adaptables cuando trabajamos con series de tiempo ya que vamos a predecir valores según el comportamiento de las observaciones pasadas.

## **Coeficiente de Gini**

La primera medida que estudiaremos para realizar nuestro análisis previo a la predicción de precios será el coeficiente de *Gini* [46]. El coeficiente de *Gini* es una medida estadística descriptiva para el análisis de grandes distribuciones de datos el cual permite medir la dispersión de los estos. Este tipo de métricas ofrecen una alternativa más prometedora en términos del valor de los datos como la frecuencia o el promedio, ya que el resultado se centra más en la variabilidad que en el valor esperado de los datos [9].

Esta medida de distribución fue creada por el estadístico de procedencia italiana Corrado *Gini* en el año 1912 para medir índices de desigualdad entre valores de una distribución. El resultado esperado es un valor en el intervalo cerrado de [0,1] donde un valor de cero nos indica la máxima igualdad entre los valores y un valor de uno la desigualdad total entre los componentes. Además, este método estadístico es aplicable a una gran cantidad de campos científicos como en ingeniería, sociología, medicina, etc.

El caso más común donde se suele aplicar esta medida es para medir los índices de desigualdad de riqueza entre los países y poblaciones, de forma que cuando una población en concreto tiene un coeficiente de *Gini* muy cercano a cero quiere decir que el nivel de riqueza de la población está muy igualado. Sin embargo, si el índice se corresponde con un valor muy próximo a 1, el nivel de riqueza de la población será muy distante entre los individuos.

El coeficiente de *Gini* está basado en la que se conoce como curva de *Lorenz* (ver *Figura 12*), la cual es una representación en forma gráfica de una distribución, donde el eje *de* abscisas representa las proporciones acumuladas de una población, y el eje de ordenadas representa los ingresos acumulados de dicha población. La línea recta en diagonal de color verde representa la igualdad perfecta cuando un porcentaje concreto de la población recibe el mismo porcentaje de ingresos (el 10% de la población recibe un 10 % de ingresos, el 20% de la población un 20% de ingresos y así sucesivamente) [46].



Figura 12. Curva de Lorenz [46]

A medida que disminuye la desigualdad, el área denominada como A desaparece puesto que el coeficiente de *Gini* es cero y viceversa. Conforme aumenta la desigualdad de los ingresos entre los individuos de la población, el coeficiente de *Gini* toma el valor de uno. Por tanto, podemos considerar el coeficiente de *Gini* como el nivel de proporción que existe entre la línea verde de la igualdad absoluta y la línea de la curva de *Lorenz* de color rojo, de forma que lo podemos calcular con la *Ecuación 1*:

Ecuación 1. Coeficiente de Gini 1 [46]

Esta fórmula nos indica que el coeficiente de *Gini* (*G*) es igual al área bajo la línea de igualdad perfecta entre la suma de A junto con el área bajo la curva de *Lorenz* (B).

La fórmula anterior viene a ser lo mismo que la *Ecuación 2*:

Ecuación 2. Coeficiente de Gini 2 [46]

Ya que A + B = 0’5 debido a la partición simétrica que contempla la diagonal de la igualdad perfecta.

Cuanto mejor es la igualdad, el área de A disminuye mientras que la curva de *Lorenz* (línea roja) se va acercando a la línea de igualdad perfecta (diagonal verde), entonces el área bajo la curva de *Lorenz* (B) irá aumentando. A la inversa, si la curva de *Lorenz* se separa de la diagonal perfecta, la desigualdad irá en aumento. Por tanto, el área B irá desapareciendo y solo quedaría el área de A.

Esta medida es muy útil sobre todo en campos relacionados con la economía para medir el nivel de ingresos que reciben los individuos de una población. De esta manera, se puede estimar una aproximación del nivel de riqueza que existe en una población entre unos individuos con otros [46].

Podríamos utilizar el coeficiente de *Gini* para medir la desigualdad de los precios de las instancias *spot*, así el coeficiente de *Gini* nos orientará acerca de la dispersión y la variabilidad de estos precios. Por ejemplo, para una instancia específica de una región, zona de disponibilidad y sistema operativo concreto, podríamos medir el nivel de desigualdad entre los precios con estas especificaciones. Así, podemos comprobar que los precios máximos a pagar que nos impone la máquina son muy distantes unos de otros (coeficiente de *Gini* alto) o, al contrario, que el precio de las instancias ha cambiado muy poco (coeficiente de *Gini* bajo) en un periodo determinado.

Para nuestro ejemplo, consideraremos la *Ecuación 3* también válida como las anteriores pero aplicada a series temporales[38]:

Ecuación 3. Coeficiente de Gini 3 [38]

Donde *n* representa el número de muestras tomadas a lo largo de un periodo. El valor *µ* es la media de los precios. El sumatorio calcula la suma de todas las diferencias de los valores medidos en dicho intervalo.

Vamos a considerar una clasificación de la variación de los precios en función del valor del coeficiente de *Gini* siguiendo el siguiente criterio:

* *Gini* bajo (entre 0 y 0’2): nos indica que el valor de los precios presenta una tendencia homogénea con respecto a la igualdad de estos y por tanto las desviaciones son muy pequeñas entre los precios.
* *Gini* medio (entre 0’2 y 0’6): no hay una tendencia clara de cara a la igualdad o desigualdad de los precios, por tanto, no se puede analizar ni mucha ni poca variabilidad entre los precios.
* *Gini* alto (entre 0’6 y 1): los valores de los precios son muy distantes y por tanto hay mucha variabilidad entre los precios.

El coeficiente de *Gini* nos aporta mucha información acerca de la variabilidad de los precios y los picos más claros de estos. Esto podría hacernos pensar que un coeficiente de *Gini* muy alto, quiere decir que los precios han variado mucho entre ellos con el paso del tiempo y como consecuencia de ello, han podido ser muchos los usuarios que han demandado la instancia en particular. Por tanto, la disponibilidad de los recursos ha ido decrementando y es por ello por lo que los precios han subido mucho. En caso contrario, si el coeficiente de *Gini* es muy bajo, nos hace suponer que los precios apenas han cambiado. Esto es debido a que posiblemente no ha habido apenas demanda por parte de los usuarios y por tanto hay disponibilidad de recursos y el precio por hora de la instancia no aumentaría [46].

## **Suavizado Exponencial**

A continuación, vamos a estudiar dos algoritmos que nos ayudarán a estimar el mejor precio que puede pujar un usuario para adquirir una instancia *spot* durante el periodo de tiempo que necesite, sin que el sistema llegue a interrumpírsela. Estos algoritmos son técnicas de suavizado exponencial o *Exponential Smoothing*. Cabe destacar que existen tres tipos de algoritmos de suavizado exponencial, el *Simple Exponential Smoothing* propuesto por *Brown* en 1959, el *Double Exponential Smoothing* por *Holt* en 1957 y el *Triple Exponential Smoothing* por *Winters* en 1960. De estos tres tipos de algoritmos utilizaremos los dos primeros, ya que el *Triple Exponential Smoothing* [50]se utiliza cuando los datos presentan componentes estacionarias entre periodos de tiempo. En nuestro caso no hemos encontrado ningún tipo de estacionalidad ni de ciclo entre los datos durante el desarrollo del *TFG*. Por tanto, lo descartaremos en principio para hacer nuestros análisis. Aunque más adelante volveremos a hablar un poco más de él. El funcionamiento de estos métodos de suavizado exponencial se describirá a continuación.

Los pronósticos que dan como resultado, se estiman a partir de promedios ponderados (pesos) que se asocian a las observaciones pasadas, y el valor de estos pesos va decreciendo a medida que se van contemplando observaciones muy antiguas. Dicho de otra forma, cuanto más reciente sea la observación anterior, el peso será mayor y cuanto más antigua sea la observación, menor será el peso asociado a la misma [45].

### ***Simple Exponential Smoothing***

El primero de los algoritmos de suavizado exponencial del que hablaremos es el que se conoce como *Simple Exponential Smoothing* o suavizado exponencial simple, que como su nombre indica, es el más simple de las técnicas de suavizado. Este método es ideal para llevar a cabo pronósticos que no presentan ninguna tendencia clara ni ningún tipo de estacionalidad. Esto lo podemos apreciar gracias al esbozo de una gráfica que represente la dinámica de los precios y podamos analizar las tendencias de estos.

Por un lado, vamos a utilizar la fórmula básica que utilizaremos con los datos de las observaciones pasadas que estudiaremos para posteriormente poder hacer el pronóstico. Para cualquier periodo *t,* el valor suavizado *St* viene dado por la ecuación general del *Simple Exponential Smoothing (Ecuación 4)*:

Ecuación 4. Simple Exponential Smoothing 1 [47]

Con los siguientes valores iniciales:

Donde *St* es el valor de la observación suavizada e *yt* el valor de la observación real.

El valor de se corresponde con el peso asociado a las observaciones pasadas cuyo valor se encuentra entre 0 y 1 y t 2.

Para la primera observación, inicializaremos el valor suavizado con el mismo valor real que presenta la observación. Para el valor suavizado de la segunda observación nos quedaremos con el valor suavizado de la primera. A partir de la tercera observación, podemos emplear la fórmula general para calcular los suavizados de las observaciones pasadas que estimemos. También se podría establecer el mismo criterio que hemos utilizado para inicializar las dos primeras observaciones, pero con las cuatro o cinco primeras, aunque siempre lo haremos inicializando únicamente las dos primeras [47].

Este algoritmo se llama exponencial porque a medida que vamos mirando las observaciones pasadas, el valor de *alpha* va disminuyendo exponencialmente hasta un punto en que casi es insignificante para las observaciones más antiguas. Es por ello por lo que en función del peso que le asociemos a *alpha*, les daremos mayor importancia a las observaciones pasadas más recientes que a las más antiguas para estimar el valor del suavizado. Como el valor de *alpha* está comprendido entre 0 y 1, si el valor es muy cercano a cero según la fórmula, les daremos mayor peso a las observaciones más lejanas, pero si el valor es muy cercano a uno, les daremos mayor importancia a las observaciones más recientes.

Para ver esto con mayor detalle, proponemos un ejemplo en la *Ecuación 5* para calcular el valor suavizado de la quinta observación. Aplicando y expandiendo la ecuación general, se obtendría recursivamente de la siguiente forma:

Ecuación 5. Simple Exponential Smoothing 2 [47]

#### **Error Cuadrático Medio (ECM)**

Para calcular el valor de *alpha*, lo haremos mediante un barrido de parámetros. El parámetro *alpha* va tomando distintos valores y se trabaja con una muestra de entrenamiento y nos quedamos con aquel valor que minimice el error cuadrático medio [47].

Para calcular el error cuadrático medio, lo haremos con la *Ecuación 6*:

Ecuación 6. Error Cuadrático Medio [47]

Donde *n* es el número de observaciones, *Si* el valor de suavizado de la observación *i* e *yi* el valor real de la observación actual *i*.

En resumen, el error cuadrático medio es la diferencia del valor del suavizado frente a los datos reales que se pretenden suavizar. De esta forma nos quedaremos con el valor de *alpha* que mejor nos convenga para hallar la mejor configuración de parámetros para posteriormente poder hacer la estimación con las observaciones futuras.

#### **Pronóstico de valores**

Una vez visto esto, entraremos a la parte del pronóstico, también llamada *forecasting,* donde detallaremos cómo se van a pronosticar los valores de las observaciones o periodos futuros una vez que tengamos la mejor parametrización en relación con el valor de *alpha* [47].

Lo primero que necesitamos saber es que la estimación de los valores futuros no se hará exactamente igual que lo hemos hecho para estimar el valor de suavizado de las observaciones pasadas. En este caso necesitamos predecir no el día siguiente, si no varios días, semanas o meses posteriores según el caso que estemos trabajando [47].

Ahora necesitaremos aplicar otra técnica para hallar los valores futuros en vista de las observaciones pasadas. Por tanto, analizando el comportamiento de estos datos, obtendremos el pronóstico de la observación posterior que indiquemos gracias a la *Ecuación 7*:

Ecuación 7. Simple Exponential Smoothing 3 [47]

Donde *t* es un valor mayor que 0.

Ahora bien, esta fórmula es apropiada para pronosticar el valor de la observación posterior, pero si queremos pronosticar más de una observación futura, necesitaremos partir desde un origen y por tanto debemos modificar la fórmula anterior convirtiéndola en lo que se conoce como técnica de *bootstrapping* (ver *Ecuación 8*):

Ecuación 8. Simple Exponential Smoothing 4 [47]

La única diferencia respecto a la anterior es que, a la hora de predecir más de un periodo posterior, no nos vale con la fórmula primera si no que debemos de partir desde un origen que será el último valor *yt* ahora llamado *yorigin*. Este lo tomaremos de nuestro conjunto de datos como estático para predecir los valores que queramos tomando como base ese origen (el último valor real que tenemos).

También puede escribirse como se muestra en la *Ecuación 9*:

Ecuación 9. Simple Exponential Smoothing 5 [47]

Donde es el error (valor real – valor predicho) para el periodo *t*. Esto quiere decir que el valor del nuevo pronóstico es igual al anterior por el error asociado en ese periodo.

Con la fórmula general de *Simple Exponential Smoothing* podríamos haber predicho los periodos futuros suponiendo que el valor real que desconocemos (*St*+ 1) podemos considerarlo como si fuera el pronosticado del periodo anterior (St + 1 = yt). De esta forma, a partir de ahí podríamos haber seguido con la fórmula general de *Simple Exponential Smoothing* para predecir con el valor de suavizado. A pesar de ello, la fórmula de *bootstraping* es muy eficaz a la hora de evaluar los valores pronosticados con los reales porque a diferencia de la de *Simple Exponential Smoothing*, los valores no suelen dispersarse mucho y se suele obtener un error cuadrático medio menor que si aplicamosla fórmula general del *Simple Exponential Smoothing* [47] [48].

Hasta aquí hemos visto todo lo referente que necesitamos saber acerca del *Simple Exponential Smoothing* para llevarlo a cabo cuando tengamos series de valores irregulares y sin tendencias claras. Pero si tenemos tendencias ligeras entre los valores de los datos ya sean crecientes o decrecientes, esta técnica no suele ser muy eficiente y obtendremos resultados muy dispares y con mucho sesgo (diferencia entre el valor esperado y el obtenido realmente). Veremos esto mejor cuando lleguemos a la parte del Capítulo 5 [47] [48].

### ***Double Exponential Smoothing***

El suavizado exponencial simple no da resultados del todo óptimos cuando existe una tendencia clara en el comportamiento de los datos, tal y como se podría observar en las gráficas. En nuestros datos tendremos muchos casos donde apreciaremos pendientes ascendentes y descendientes de precios del sistema. Analizaremos cuales podrán ser los posibles resultados gracias a este algoritmo, que nos ayudará a pronosticar los valores futuros atendiendo a las tendencias de los datos pasados [49].

Si recordamos el *Simple Exponential Smoothing* solo teníamos como entrada el valor de *alpha* que indicaba el peso que se les asignaba a las observaciones pasadas. Pero para este algoritmo, además de *alpha*, tendremos el parámetro *beta* que es el que ayudará a medir la tendencia de los datos, por tanto, el *Double Exponential Smoothing* es como una extensión del *Simple*.

Las ecuaciones para el suavizado de los valores del conjunto de entrenamiento, son la *Ecuación 10* y la *Ecuación 11*:

Ecuación 10. Double Exponential Smoothing 1 [49]

Ecuación 11. Double Exponential Smoothing 2 [49]

Donde *St* es una estimación del nivel de la serie de tiempo, *bt* denota la estimación de la tendencia en el periodo *t*, es el parámetro para el suavizado para las observaciones pasadas y es el parámetro de suavizado que mide la tendencia. Los valores de *y* son las observaciones al igual que en el *Simple Exponential Smoothing*.

Para inicializar el parámetro de *b0*, podemos hacerlo tal y como dicta la *Ecuación 12*:

Ecuación 12. Double Exponential Smoothing [49]

El parámetro de *S0* lo inicializamos tomando el mismo valor que *yt*, al igual que con el *Simple Exponential Smoothing*:

La *Ecuación 13* muestra cómo calcular los valores de suavizado en el entrenamiento de los datos:

Ecuación 13. Double Exponential Smoothing 6 [49]

Donde *Ft+1* es el pronóstico para el periodo posterior el cual es la suma de *St* + *bt* del periodo u observación en la que nos encontramos.

#### **Error Cuadrático Medio (ECM)**

Al igual que hacíamos con el *Simple Exponential Smoothing*, debemos de calcular los parámetros, en este caso de *alpha* y de *beta*, que minimicen el error cuadrático medio aplicando la misma fórmula de la *Ecuación 6* [49].

#### **Pronóstico de valores**

Al igual que nos pasaba con el *Simple Exponential Smoothing*, a la hora de pronosticar los *m* valores posteriores para evaluarlos tenemos que hacerlo con la *Ecuación 14*, tomando como referencia el último valor de *Ft* que tenemos de los datos de entrenamiento:

Ecuación 14. Double Exponential Smoothing 7 [49]

De esta forma podemos pronosticar los *m* valores posteriores partiendo del valor del último valor *Ft*.

Para hallar con la configuración óptima de parámetros, podemos utilizar el error cuadrático medio y quedarnos con la parametrización de *alpha* y *beta* que minimice dicho error [49].

# **FUNCIONAMIENTO DEL SISTEMA**

En este capítulo, hablaremos acerca del entorno de trabajo donde hemos realizado todo nuestro sistema de predicción para entender mejor su dinámica. En concreto, hablaremos acerca del desarrollo y el funcionamiento del sistema desde un punto de vista más técnico.

## **Entorno de trabajo**

Todo el código que ha sido creado para implementar nuestro sistema se ha realizado con el lenguaje de programación *Python 3* [13]*.* Hemos de instalar el *plugin* necesario en nuestro equipo y configurarlo para poder utilizarlo dentro del entorno de *Jupyter Notebook* [15] del que hablaremos un poco más en detalle a continuación. Podemos descargar el *plugin* de *Python* desde su página oficial.

### ***Jupyter Notebook***

*Jupyter Notebook* es una herramienta de programación para *Python* [13] muy intuitiva e interactiva, ya que nos permite trabajar con bloques de código para estructurarlo de una forma mucho más atractiva y visual para los desarrolladores. Este entorno funciona mediante la creación de libretas o *scripts* en formato *ipynb* que podremos guardar en un directorio de nuestro equipo añadiendo un nombre a cada libreta. Dentro de cada una de estas libretas podemos crear bloques de código normales y bloques para comentarios de forma que en nuestro caso nos ayudarán a dividir diferentes secciones de trabajo, como la carga de paquetes, la especificación de parámetros de entrada, el preprocesado de datos, la ejecución de los algoritmos, etc.

Todo nuestro trabajo para desarrollar el sistema de precios consta de trece libretas:

- Tres de ellas para esbozar las gráficas de los precios donde cada una de ellas serán tres libretas para días, semanas o meses dependiendo del caso a estudiar.

- Para el coeficiente de *Gini* que serán también tres *scripts* dependiendo del periodo que se analice (días, semanas o meses).

- Para el *Simple Exponential Smoothing*. Estas serán igual que con las anteriores, una para días, otra para semanas y otra para los meses.

- Para el *Double Exponential Smoothing*, igual que los anteriores casos, tres para diferentes tipos de periodos, en función con lo que se pretenda trabajar.

- La última libreta será el *script* para la descarga e inserción de los datos, pero este es un caso particular y hablaremos de él más adelante.

De las trece libretas que hemos comentado, tendremos nada más abrir una de ellas, un índice que nos especificará dónde se encuentra cada una de las partes en las que se estructura el código. Por ejemplo, para la ilustración que se muestra al abrir la libreta de “*Double Exponential Smoothing – Días.ipynb*” (ver ejemplo en la *Figura 13*), tendremos un apartado para cargar paquetes, otro donde debemos especificar los parámetros de entrada, otro para el preprocesado de datos del tipo de periodo en concreto (días, semanas o meses), otro para llevar a cabo el entrenamiento de los datos y finalmente otro para la evaluación del pronóstico. Pueden consultarse el nombre de los ficheros de las libretas en el Anexo D de este documento.

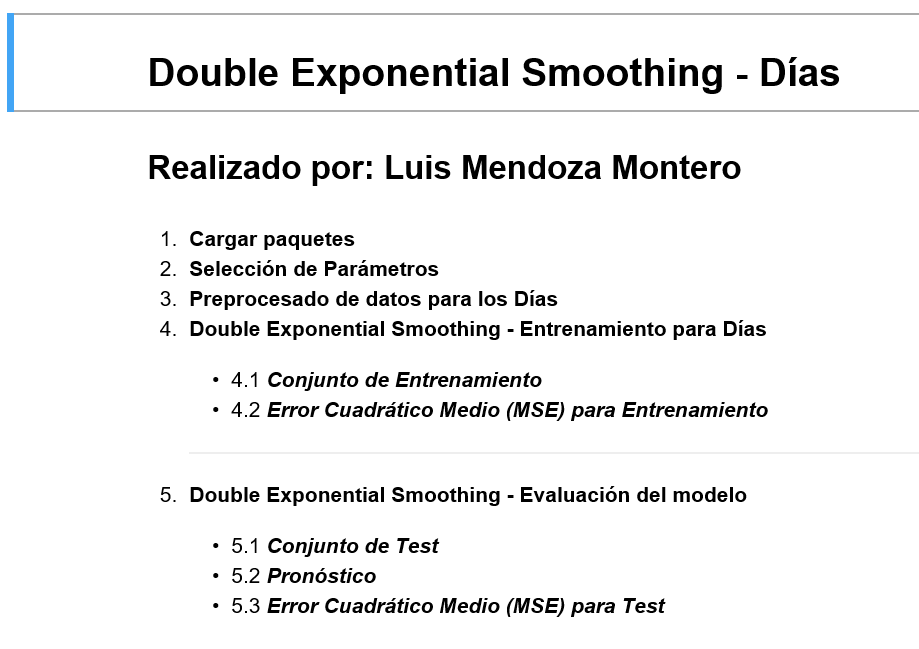


Figura 13. Índice libreta de Jupyter

Al ejecutar la libreta una vez indicados todos los parámetros requeridos, se ejecutarán cada uno de los bloques, tanto de texto como de código, de forma secuencial desde el primero hasta el último sin verse afectados los bloques de código negativamente.

### ***SQLite***

Para almacenar y estructurar los datos que necesitaremos para los experimentos, utilizaremos el gestor de bases de datos relacionales *SQLite* [17]. A continuación, describiremos la estructura de nuestra base de datos.

La base de datos representa un tamaño de instancia en particular, y cada una de las tablas representa la categoría y la región a la que pertenece la instancia. El formato de la tabla será “*categoría\_región*”. Por ejemplo, para referirnos a la información del tipo de instancia *c4.2xlarge* para la región *ap-northeast-1*, la encontraremos en la tabla *C4\_ap\_northeast\_1*. La *Figura 14* detalla la organización de las tablas.

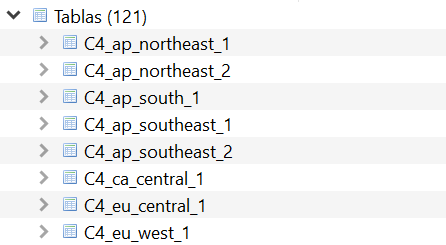


Figura 14. Estructura de tablas BBDD

Dentro de cada tabla nos encontramos con cinco columnas que nos aportarán el resto de información que necesitamos como la zona de disponibilidad, el sistema operativo, el tipo de instancia, el precio y el *timestamp* con la fecha y hora de los cambios del precio registrados por el sistema (visualizar *Figura 15*).

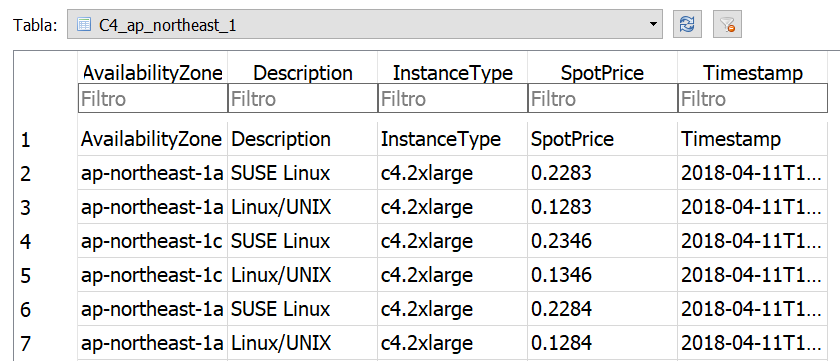


Figura 15. Atributos de una tabla

Para poder manipular la información de la base de datos podemos descargar la aplicación de escritorio “*DB Browser for SQLite* [17]*”* que nos permitirá interactuar con la base de datos de una forma visual y rápida. Podemos encontrar nuestro fichero de base de datos con toda la información que hemos trabajado en la documentación del CD con el nombre *BBDD.db*.

El tamaño de la BBDD aumentará conforme vayamos adquiriendo más datos del histórico de precios. En total tenemos 121 tablas y cada tabla con unos 4000 registros aproximadamente. Nuestra base de datos mantiene un histórico de fechas del febrero hasta el 30 de septiembre de 2018. En total 8 meses. El tamaño de la BBDD es de 90’9 MB.

## **Preparación de los datos**

Antes de comenzar con el apartado de experimentos, vamos a dedicar una parte de este capítulo a hablar acerca de los datos con los que vamos a trabajar. Para empezar, necesitamos acceso a *Amazon Web Services* por medio de una cuenta que crearemos desde la plataforma (es totalmente gratuita durante primer año). Una vez que tengamos acceso a *Amazon*, gracias a un *script*, podremos acceder al histórico de datos pasados para poder almacenarlos en nuestra base de datos. Esto lo podemos hacer a través de un *API* para obtener la lista de precios también llamada *API* de consulta. Por último, hablaremos sobre el preprocesado o limpiado previo que llevaremos a cabo con los datos, ya que como veremos después, estos datos no son perfectos recién descargados de la plataforma y presentan algunas disconformidades que pueden hacer que los resultados de los algoritmos no sean correctos.

Veamos entonces cómo acceder a la plataforma de *Amazon* para obtener el histórico de precios de las instancias.

### ***Credenciales de acceso***

Si recordamos los apartados anteriores, hablamos sobre las distintas formas de acceso a *Amazon* que podría ser mediante (un kit de desarrollo software o una interfaz para los usuarios). Nosotros lo haremos por medio de línea de comandos gracias a la herramienta *AWS CLI* y a las credenciales que nos facilitará interactuar con los servicios de *AWS* durante un año de forma gratuita [34]. Dichas credenciales las podemos conseguir registrando una cuenta en *AWS*. Además, la herramienta de *AWS CLI* está basada en un *SDK* que *Amazon* ofrece para acceder a sus servicios por medio de *Python* [13]. Esta es la librería *boto* [14] que utilizaremos también cuando accedamos a los datos de *Amazon* por medio del *script*” *Obtener e insertar Datos.ipynb*”.

Una vez que tengamos las credenciales, obtendremos un nombre de usuario, una contraseña, un ID de acceso, una clave secreta y una *url* donde podremos iniciar sesión con esta información para acceder a *Amazon*.

Después, necesitaremos abrir la consola e ingresar el siguiente comando como se muestra en la *Figura 16*:

**$ *py aws configure***

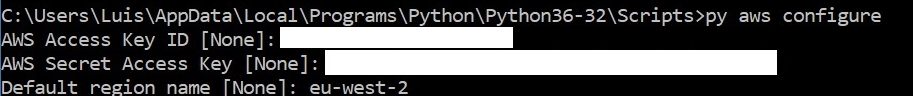


Figura 16. Línea de comandos para credenciales

*AWS CLI* nos pedirá que escribamos el ID y la clave secreta de acceso para que de esta forma podamos acceder a los datos de *Amazon* por medio del *script* del que hablaremos a continuación.

### ***Obtención de datos desde la nube***

Antes de empezar a realizar ejecuciones de nuestras libretas de *Jupyter* [15] necesitamos ejecutar el *script* “*Obtener e Insertar Datos.ipynb*”. Como ya hemos mencionado anteriormente, en la parte donde añadimos la carga de paquetes y librerías al comienzo de todos los bloques, hemos de cargar la librería de *boto* [14]. De esta forma tendremos acceso garantizado a *Amazon*.

Los parámetros que especificaremos al inicio de la libreta serán el tamaño de instancia, la familia o tipo al que queremos acceder y la región en la que se encuentra:

Una vez que los datos son descargados con éxito, serán insertados y almacenados automáticamente en nuestra BBDD.

Debemos tener en cuenta que *Amazon* permite obtener los datos del histórico de los precios que el sistema ha ido cambiando desde los 90 días anteriores hasta el momento de la ejecución del *script*. Por tanto, para almacenar todos los datos que tenemos en la BBDD hemos de ir ejecutando e insertando los datos periódicamente para que vayamos aumentando los datos con los que trabajar en los experimentos.

El *script* está diseñado para que, por cada ejecución de este, los nuevos datos sean almacenados y apilados sobre los anteriores sin que estos últimos sufran ningún tipo de modificación ni pérdida. Si ejecutamos el *script* más de una vez y descargamos datos que ya estaban insertados en la BBDD, los registros repetidos no serán insertados de nuevo, puesto que los datos que ya estaban registrados.

En el mismo código hemos indicado cuáles son los parámetros que debemos añadir en nuestra base de datos sobre cada uno de los registros que almacenemos. Como tenemos cinco columnas por cada tabla, la información que nos devolverá *Amazon* del sistema será la zona de disponibilidad referente a la región, la descripción del sistema operativo, el tipo de instancia, el *timestamp* con la fecha, hora, minuto y segundo exacto en el que se ha cambiado el precio y el precio.

Por eso, cada tabla de nuestra base de datos contendrá esos cinco atributos para diferenciar por cada registro el tipo de instancia, zona en la que se encuentra (recordemos que la región la especifica el propio nombre de la tabla junto a la categoría asociada), con qué sistema operativo y el precio que ha alcanzado en un instante dado (*timestamp*).

El *script* realiza la descarga y la inserción de los datos al mismo tiempo durante la ejecución.

Como ya hemos comentado antes, se ha seleccionado un tamaño de instancia *2xlarge* como muestra del tamaño de instancia estático para nuestra experimentacióndebido a que está disponible para todas las categorías y así se puede tener más información que comparar. Debido a que nuestra base de datos es exclusiva para estudiar los casos de *2xlarge*, se sugiere crear otra base de datos para otro tamaño de instancia que se quiera analizar para no colapsar la BBDD. Las tablas se estructurarían de la manera que hemos mencionado anteriormente para que el formato de estas mantenga la consistencia con nuestro sistema y sea adaptable para otras configuraciones de tamaños diferentes. Hay que tener en cuenta que, si queremos crear otra base de datos para otros tamaños, hemos de especificárselo en nuestros *scripts* y cambiar el nombre de *BBDD.db* por el de nuestra nueva base de datos en la variable *conexion*.

Frecuentemente, los datos suelen presentar mucha información irrelevante para nuestros casos. Datos duplicados o redundantes, datos con mucho ruido, inconsistentes y muy poco fiables, por lo que la fase de análisis de datos a la hora de obtener los resultados puede llegar a ser un verdadero problema. En general, tenemos datos denominados datos sucios o no limpios.

Cuando nos referimos a que los datos no están limpios o están sucios es que contienen valores perdidos, mucho ruido entre ellos o errores. Normalmente estos datos tienen un nivel de calidad muy bajo para ser ejecutados por los algoritmos.

### ***Preprocesado de datos***

Este apartado lo enfocaremos a hablar sobre el preprocesado previo que ha de realizarse sobre los datos descargados para poder hacer un análisis de estos.

El preprocesado de los datos es uno de los puntos más importantes para el análisis de datos. Es una de las claves fundamentales para que los algoritmos sean ejecutados eficientemente y con éxito. La representación, la estructura, la organización y la calidad de los datos son los objetivos primordiales para realizar la fase del preprocesado correctamente.

Los datos que están recién descargados desde la nube suelen presentar valores “sucios” como los que ya hemos comentado antes. Cuando ejecutamos el *script* de descarga e inserción de los datos, podemos comprobar cómo están almacenados en la tabla correspondiente. Vamos a ver esto en la *Figura 17* después de realizar la descarga de la tabla *C4\_ap\_northeast\_1* para el sistema operativo *SUSE Linux* y para la zona de disponibilidad *ap-northeast-1a* con algunos de los datos registrados.

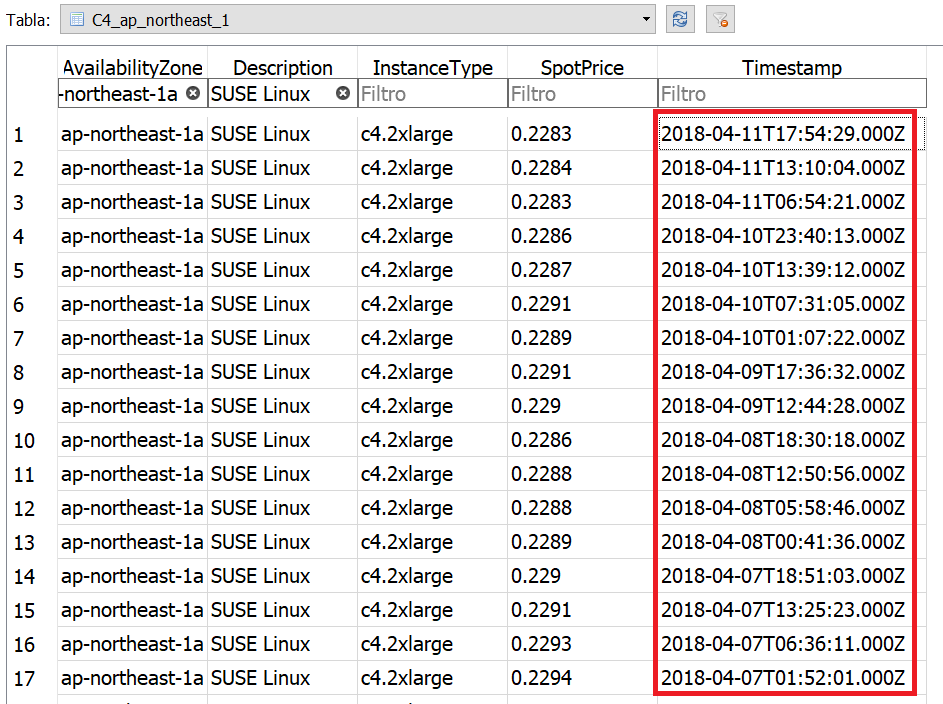


Figura 17. Timestamp sin preprocesar

Como podemos comprobar en la columna donde aparece el *timestamp*, tenemos la fecha, la hora, el minuto y el segundo exacto cuando el sistema actualiza los precios automáticamente. La máquina del sistema suele actualizar el valor del precio cada vez que esta varía. Por tanto, cuando realiza la actualización, nos muestra el instante en el que cambia el precio que deben pujar los usuarios para obtener la instancia. La máquina suele actualizar el precio constantemente cada vez que este cambia debido a la cantidad de demanda de los recursos que requieran los usuarios.

Ahora bien, si nosotros queremos establecer para nuestro algoritmo una predicción de los *n* días posteriores, por ejemplo, necesitamos saber cuál ha sido el comportamiento del precio durante las 24 horas que tiene cada día. Pero como podemos ver en la *Figura 17*, el sistema solo nos actualiza los registros de los precios cada vez que estos cambian. Por ese motivo, necesitamos identificar el estado de las horas que hay entre medias de las actualizadas. Para el correcto funcionamiento de los algoritmos, necesitamos una muestra de datos con una temporalidad periódica. Es decir, el mismo número de muestras por periodo temporal y además que estén distribuidas uniformemente a lo largo del tiempo.

Para solventar este problema de los datos perdidos o sin valor, podemos aplicar distintas técnicas de limpieza como ignorar registros sin valores. Aun así, hay que tener especial cuidado, ya que en nuestro caso necesitamos los valores de los precios en las horas que faltan y por tanto necesitamos rellenar esos datos de algún modo. Por lo tanto, vamos a completar los valores de los precios de las horas que nos faltan aplicando la metodología de preprocesado que se especificará en el siguiente apartado.

#### **Preprocesado por horas**

Para empezar, vamos a realizar un preprocesado de datos de manera general para todos los casos. Este preprocesado consistirá en organizar los datos que tenemos por horas para así, partiendo de estas horas, poder organizar los datos por días, por semanas o por meses. Para ello, aplicaremos los criterios que se mencionan a continuación.

Los precios de las instancias *spot* pueden permanecer constantes sin cambiar nada durante un determinado periodo de tiempo. Como estos precios solo se registran cuando cambian o de vez en cuando pasado un tiempo, nuestros datos pueden no contener información sobre los precios en todas las horas del día. Por lo tanto, vamos a rellenar los datos faltantes para que no contengan valores nulos (*NaNs*). Sustituimos el valor de los datos faltantes añadiendo el último precio registrado anterior y así secuencialmente para todos los días.

Una vez que tengamos todas las horas de cada día rellenadas con el precio anterior actualizado del sistema, insertaremos los datos en una tabla nueva de la BBDD llamada “*Preprocesados*”. Esta tabla la tomaremos como base para pasar nuestros datos ya preparados para poder realizar otros preprocesados por días, por semanas o por meses.

Por lo tanto, los datos con los que trabajaremos no serán los de cualquier tabla que tenemos en la BBDD, si no los de la tabla “*Preprocesado”s* que contendrá la información de los datos sin valores nulos en las horas y preparados para ser ejecutados por los algoritmos. Así ya tendremos las 24 horas de cada día con sus valores actualizados tal y como se muestra en la *Figura 18*. Si se da el caso de que, en una hora específica, el sistema haya actualizado los precios más de una vez, tomaremos el valor máximo del precio que indique uno de esos instantes para esa hora.

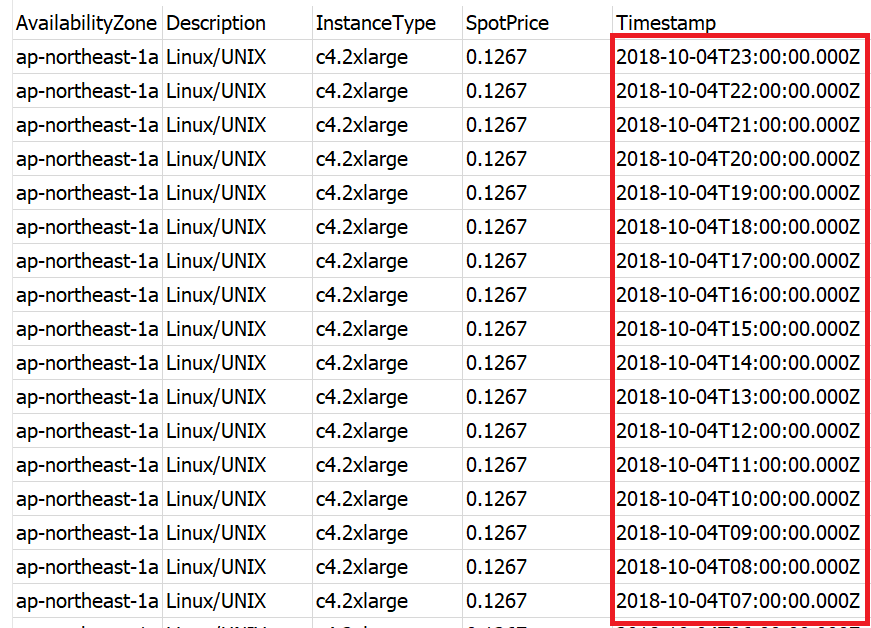


Figura 18. Timestamp preprocesado por horas

Vamos a mostrar un ejemplo más visual del preprocesado por horas mediante dos gráficas, una con datos preprocesados y otra sin preprocesar.

Figura 19. Gráfica horas sin preprocesar

En la gráfica de la *Figura 19*, podemos apreciar como en el eje de abscisas se representa el tiempo transcurrido por un intervalo de horas para un día en concreto. En el eje de ordenadas tenemos el valor del precio en dólares *USD* por cada hora, entre los 0’3338 $ a los 0’235 $. Se puede comprobar como el precio desciende desde los 0’2346 $ para las *1:58:44* y va disminuyendo hasta las *18:32:15* con un valor de 0’2342 $ la hora. El sistema simplemente ha actualizado el precio en cinco instantes de tiempo distintos en un día. Es una mala representación de los datos, puesto que no tenemos información acerca de las horas que hay entre medias de las registradas. Por ello, vamos a realizar un preprocesado sobre esos mismos datos y compararemos los resultados. Además, no cumple los requisitos de una serie temporal.

Figura 20. Gráfica horas preprocesadas

En la gráfica de la *Figura 20*, vemos como los periodos por horas en el eje de abscisas quedan mucho más organizado que en la *Figura 19* puesto que estamos estructurando los precios siguiendo el criterio de las horas para un día en concreto. En el eje de ordenadas tenemos el mismo rango de valores para los precios que en la anterior gráfica. Sin embargo, el precio desde las *00:00:00* hasta las *06:00:00* es de 0’2349 $ que viene del día anterior que no contemplábamos en la *Figura 19*. Esto es porque se corresponde con el precio registrado del día anterior. Por tanto, todas las horas quedan cubiertas y contempladas con el precio que les corresponde.

#### **Preprocesado por días**

Ahora que tenemos los precios de las instancias *spot* preprocesdos por horas, podremos obtener los precios por días si queremos hacer la estimación por medio de nuestros algoritmos. También nos serviría hacer la estimación con una granularidad por rango de horas.

Nuestra mecánica de preprocesado por días será muy sencilla. Simplemente trataremos de *parsear* el campo de *timestamp* para todas las horas que tiene un día y registraremos el precio máximo entre esas horas para ese día. De esta forma, obtendremos el precio máximo por día para que podamos establecer un único precio para nuestro conjunto de datos por días.

Para esto, obtendremos los 24 registros de cada día de la BBDD y calcularemos el precio máximo de esos 24 registros. Después crearemos un registro por cada día obtenido del *timestamp* con el precio correspondiente. Obtendremos una tabla con la simplificación de datos que teníamos por horas, pero ahora con los días.

En nuestra tabla de datos tendremos una columna de “Periodo (*Días*)” con la fecha por días *parseada* (modificada del *timestamp*). Y otra de “Precio Sistema” que será el precio máximo que ha llegado a actualizar el sistema para cualquier instancia.

#### **Preprocesado por semanas**

Para preprocesar los datos por semanas, hemos de partir de los anteriores preprocesados, tanto por horas como por días. Una vez que tengamos lo datos preprocesados por días, hemos de calcular el precio máximo por cada semana para añadirlo a nuestra tabla de datos y pasársela a los algoritmos de predicción.

Para ello, teniendo en cuenta la fecha de inicio y la fecha de fin de los datos de entrenamiento que especificamos por parámetros, el sistema nos mostrará automáticamente las semanas que hay desde el primer día hasta el último con el precio máximo por semana.

El procedimiento para calcular el precio máximo por semana es muy sencillo. De la misma manera que calculábamos el precio máximo por días, lo haremos con las semanas. Como tenemos el conjunto de datos por días, sacamos el precio máximo de cada siete días y así sacamos el precio máximo por semana.

Nuestro conjunto de datos quedará con una columna denominada “Periodo (*Semanas*)” donde tendremos el número de la semana específica en función de la fecha del parámetro de inicio hasta el parámetro de fecha de fin. La otra columna “Precio Sistema” contendrá el precio máximo por cada semana.

#### **Preprocesado por meses**

El último de los casos de preprocesados que consideraremos será con los meses. Así, los usuarios ya podrán hacer la estimación del precio para su instancia tanto por días, como por semanas como por meses.

Para realizar el preprocesado por meses será algo más complicado que los anteriores. Para este caso, debemos de preprocesar los datos por días y después habrá que *parsear* el *timestamp* por la fecha y en concreto por el mes, para obtener todos los registros de un mes y calcular el máximo precio alcanzado por el sistema.

Para hacer el *parseo* de los números de las fechas del *timestamp* con el nombre de los meses, simplemente tratamos de que en función del número de mes que aparezca, añadamos el nombre del mes correspondiente. Por ejemplo, para *2018-03-16* al preprocesar por días, nos quedamos con el *03* y lo cambiamos por “Marzo” y así para los meses que hagan falta. Al tener los registros de los días correspondientes del mes específico, obtenemos el valor del precio máximo de todos esos días para el mes en particular.

De esta manera, nos quedará en una columna llamada “Periodo (*Meses*)” con el nombre del mes en concreto, y una columna llamada “Precio Sistema” que como siempre mostrará el precio máximo por cada uno de esos meses.

Con esto, ya tendremos los datos preparados para analizarlos con los algoritmos de nuestro sistema y poder hacer la estimación de un precio óptimo para los usuarios sin que los datos tengan valores nulos o estén sucios. Para nuestro caso, estos casos de preprocesados son suficientes para nuestro volumen de datos, puesto que solamente hemos conseguido información desde febrero hasta septiembre. Pero si hubiéramos adquirido más volumen de datos, podríamos haber preprocesado los datos por años para completar aún más nuestro sistema de predicción.

En el próximo apartado hablaremos acerca de los parámetros que los usuarios necesitarán rellenar en nuestro sistema correctamente para que nuestro sistema no entre en conflicto y reconozca bien cada uno de los parámetros.

## **Parámetros de entrada**

Los parámetros de entrada que nos tendremos para nuestro sistema de predicción según a la parametrización del tipo de libreta que realicemos la ejecución, serán los siguientes:

* **Fecha de inicio y fin:** estos parámetros sirven para especificar cuantos datos históricos, en rango de tiempo, se pretenden observar hacia atrás. Así podremos entrenarlos y dar con el modelo óptimo en función del error (más adelante detallaremos esto último). Los días que tenemos disponibles en la base de datos son desde el 1 de febrerohasta el 1 de octubre de 2018*,* por tanto, las especificaciones del usuario no han de sobrepasar el rango de fechas.
* Por otro lado, se debe especificar el periodo que queremos estimar en el futuro (rango de tiempo a pronosticar). Este parámetro se llama **posteriores**, aparecerá tanto en el *Simple* como en el *Double Exponential Smoothing,* ya sea para los días, semanas o meses posteriores que deseamos estimar.
* Los usuarios también deberán indicar **el tipo de sistema operativo** que puede ser para *Windows*, *SUSE Linux* o *Linux Unix*.
* También hay que especificar **la región** que estará asociada a la tabla que deseemos estudiar de la BBDD.
* La **zona de disponibilidad** de la región con la que queramos acceder también ha de ser referenciada como parámetro.
* **Otros parámetros**: en función del tipo de algoritmo que seleccionemos, les pasaremos unos parámetros específicos para ese algoritmo u otro en función del caso o del experimento a analizar.

Estos serán los parámetros necesarios para la ejecución de los *scripts* en nuestras libretas de *Jupyter* [15] para que la ejecución de estos funcione correctamente. En el caso de que el *script* no reconozca el valor de algún parámetro, el sistema fallará. Por ejemplo, a la hora de especificar la fecha de inicio o fecha de fin fuera del rango de datos que tenemos en la BBDD, el *script* nos informará sobre el fallo y se interrumpirá la ejecución del código en el bloque donde se trate el parámetro en concreto.

Para finalizar la parte donde explicamos el funcionamiento de nuestro sistema, vamos a hablar brevemente cómo realizamos el entrenamiento y la evaluación de los datos para la ejecución de los algoritmos y pasaremos al apartado final de la memoria donde haremos una serie de experimentos reales con los datos. Pero antes, comentaremos cómo realizar el estudio para analizar el comportamiento de los datos con las gráficas.

## **Análisis de gráficas**

Existen numerosos métodos de predicción para estimar valores gracias a distintas métricas de análisis y algoritmos para evaluar pronósticos, pero además de eso, podemos apreciar otras medidas previas no tan exactas pero que nos ayudarán mucho a la hora de saber qué algoritmo utilizar para predecir valores.

En este caso, el primero de los análisis que llevaremos a cabo en la experimentación será hacer una gráfica que nos ayudará a captar una visión general del comportamiento de los datos para después establecer y seleccionar los parámetros necesarios conforme al rango de valores que decidamos entrenar y posteriormente evaluar.

Para ello, lo primero que necesitamos, será tener los datos de los precios según el día, de la semana o del mes que tengamos disponibles en la BBDD. Sobre esos datos ejecutaremos una serie de *scripts* que nos darán como resultado la gráfica correspondiente a los precios que la máquina ha ido cambiando y actualizando de forma dinámica.

Gracias a la gráfica de los precios del sistema, antes de hacer cualquier estimación, podremos obtener información sobre qué tendencias tienen los valores de los precios con el paso del tiempo, cuáles son los picos que presenta, si tienen pendiente ascendente, descendiente o constante o si podemos hallar algún tipo de ciclo en intervalos de tiempo repetidos.

La representación gráfica por tipo de instancia, sistema operativo y zona de disponibilidad nos permitirá hacer un análisis de tendencia sobre los precios, por lo que nos ayudará a seleccionar los datos que analizaremos para predecir valores a posteriori.

## **Obtención del modelo**

Para llevar a cabo una buena técnica de predicción y aplicación de algoritmos, además del preprocesado previo de los datos, vamos a explicar con más detalle en qué consiste el entrenamiento y la evaluación o *test* de la que hablábamos en el Capítulo 3.

### ***Entrenamiento de los datos (Training)***

Cuando ya tengamos preparados los datos para ser ejecutados por los algoritmos, hemos de hacer un entrenamiento con estos datos para obtener un modelo adecuado para el algoritmo en cuestión. El entrenamiento de los datos consiste en hacer varias ejecuciones de los algoritmos con diferentes valores de parámetros (barrido de valores) para obtener un modelo.

El modelo que seleccionaremos será aquel cuya ejecución del algoritmo correspondiente con la parametrización adecuada, minimice el valor del error cuadrático medio con respecto al resto de ejecuciones. Tal y como ya hemos mencionado en los apartados del Capítulo 3, hemos de calcular el error cuadrático medio de cada una de las ejecuciones que consideremos atendiendo a la parametrización. Por ejemplo, para el *Simple Exponential Smoothing*, tenemos como parámetro el *alpha*, pero debemos saber qué valor de *alpha* nos da el menor error con respecto al precio real que registra el sistema.

Para ello, gracias a la librería *pandas* de *Python* [16], podremos trabajar con conjuntos de datos voluminosos agrupándolos por ***dataframes***. De esta manera, todos los datos que vayamos a entrenar y ejecutar en los algoritmos serán pasados de la BBDD a *dataframes* con el nombre de *dfTrain* para saber el conjunto de datos que utilizaremos para entrenar.

El lenguaje *Python* [13] también permite convertir los datos de *dataframe* a *csv* de forma que podremos obtener los datos en una hoja de cálculo. De esta forma, gracias a la función *solver* de la herramienta, podremos obtener automáticamente el valor de los parámetros que minimicen el ECM. Por tanto, es un atajo automatizado para llevar a cabo el proceso de barrido de valores para obtener el valor de *alpha* óptimo.

Los datos que tomaremos para nuestro conjunto de entrenamiento serán los que especificamos por los parámetros del sistema *f\_inicio* y *f\_fin*. Además, *pandas* [16] también nos permite visualizar el *dataframe* de entrenamiento (*dfTrain*) con una tabla que nos creará al ejecutar el código del *script*.

Los algoritmos que requieren hacer un entrenamiento de los datos serán el *Simple* y el *Double Exponential Smoothing.* Es decir, los algoritmos que utilizaremos para realizar el pronóstico.

Una vez que hagamos el entrenamiento de los datos y obtengamos el modelo con el menor ECM, la última parte que nos queda será la evaluación de nuestro modelo por medio del conjunto de datos de prueba, de los que hablaremos en el apartado que viene a continuación.

### ***Evaluación de los datos (Test)***

La fase final que nos queda por hablar de nuestro sistema de cara a cada experimento será la evaluación de nuestro modelo. Al igual que hemos hecho con el entrenamiento, dejaremos un conjunto de datos para evaluar que denominaremos por *dataframe* como *dfTest*. Estos datos los obtendremos gracias al parámetro “*posteriores*”, que nos indicará los *n* periodos que siguen a los datos de entrenamiento, los cuales queremos evaluar o predecir. Si por ejemplo tenemos veinte días para entrenar y el usuario quiere el pronóstico para el precio de los cinco días siguientes, en nuestro conjunto de prueba tendremos los precios de esos cinco días posteriores. Nuestro sistema predecirá gracias a nuestro modelo de entrenamiento los cinco días posteriores de nuestras observaciones pasadas (datos de entrenamiento). Lógicamente sin tener en cuenta cual será el comportamiento real de los precios de esos cinco días. Para una mejor comprensión del funcionamiento del entrenamiento y evaluación de los datos, visualizar la *Figura 21*:

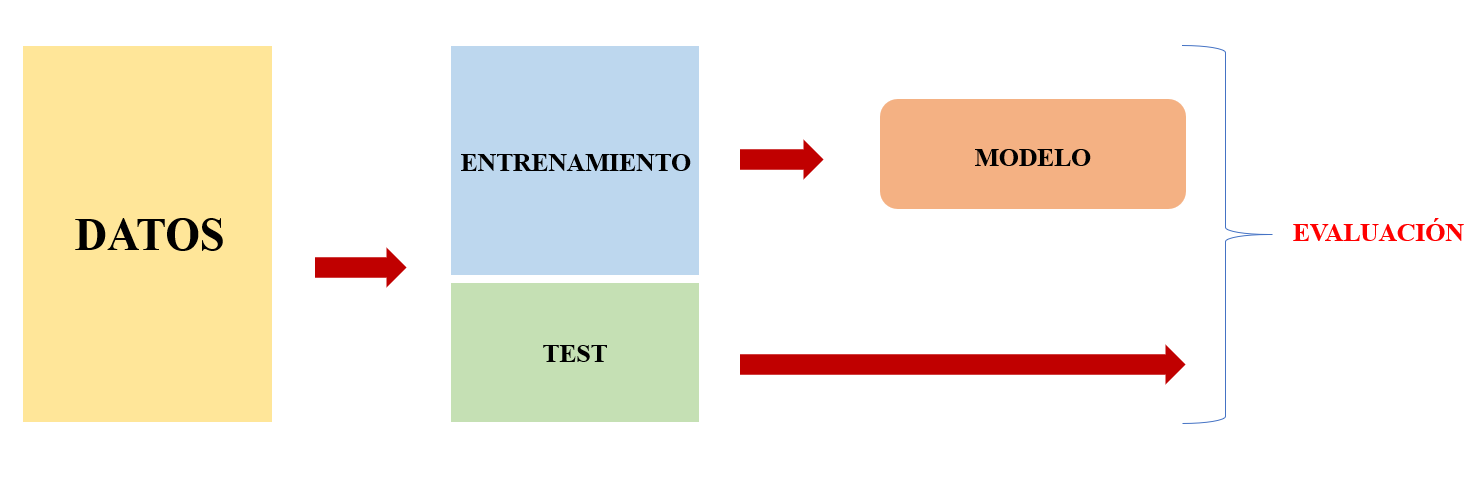


Figura 21. Entrenamiento y Test

Estas son las métricas de evaluación que utilizaremos en el trabajo:

* **Error Cuadrático Medio**: una vez pronosticados los precios de los *n* periodos posteriores de un algoritmo, calcularemos el ECM correspondiente del conjunto de *test* y lo compararemos con el otro algoritmo como una medida de error.
* **Servicio Cumplido**: para entregar al usuario el precio adecuado conforme a sus necesidades, nos quedaremos con el precio máximo de los *n* periodos posteriores estimados y el máximo de los precios posteriores reales. Una vez que tengamos los precios máximos, el sistema los comparará y si el valor del precio máximo de nuestra estimación ha sido igual o superior al real, el sistema responderá con un mensaje de que el pronóstico ha sido óptimo. Si el pronóstico ha sido óptimo, significa que el usuario podrá hacer uso de la instancia durante el periodo que necesita sin que el sistema llegue a interrumpírsela. En caso contrario, si el valor del precio máximo del sistema es superior al pronosticado, el sistema nos informará de que el pronóstico no ha sido correcto. Si el pronóstico no ha sido correcto, significa que si el usuario puja durante su periodo de contratación el precio que le hemos asignado, el sistema llegará a superar su valor en algún momento durante su periodo de contratación e interrumpirá la instancia.
* **Ahorro en coste**: en caso de que la estimación del precio de la instancia *spot* haya sido correcta en función de la anterior evaluación, compararemos ese precio con el de una instancia reservada o bajo demanda con las mismas características para demostrar al usuario el ahorro que le ha supuesto contratar su instancia como una instancia *spot*. Los *hosts* dedicados nos los consideraremos para la comparativa puesto que el precio de estos mucho mayor que el del resto, ya que integran recursos *hardware* exclusivos.

Es importante que los datos de prueba no sean en proporción mayores que los de entrenamiento, ya que normalmente se suele elegir una proporción alta de los datos para entrenarlos y una proporción más baja de los datos para evaluarlos.

# **ANÁLISIS DE RESULTADOS**

La experimentación de los diferentes casos de prueba que analizaremos serán la parte práctica y real de la que hemos estado hablando durante el funcionamiento del sistema.

Realizaremos varios experimentos para estudiar diferentes posibles casos que puedan llegar a darse en nuestro sistema. Analizaremos un caso de prueba para estimar los días, otro para las semanas y otro para los meses.

Para recapitular cada uno de los puntos vistos en el funcionamiento del sistema, vamos a establecer una serie de pautas a seguir para realizar nuestro proceso de análisis y estimación correctamente en cada experimento:

1. Inicializar los parámetros en los *scripts* de gráficas especificando el tipo y categoría de la instancia, el sistema operativo, la región, la zona de disponibilidad el periodo de las observaciones a entrenar, los parámetros de los algoritmos, etc. Estas serán las especificaciones del usuario.

2. Preprocesar los datos según el caso con el que trabajemos (días, semanas o meses).

3. Graficar los precios reales para medir el comportamiento y la tendencia de los datos.

4. Medir la variabilidad y dispersión de los datos con el coeficiente de *Gini*.

5. Aplicar el entrenamiento de los datos por medio del *Simple* o *Double Expoential Smoothing*.

6. Obtener el modelo para el caso en concreto con el mínimo error cuadrático medio.

7. Evaluar el modelo con las diferentes opciones de evaluación que hemos comentado en la sección *4.5.2*.

8. Obtener el precio óptimo pronosticado para poder entregárselo al usuario.

## **Experimento 1 – Demanda instancia *spot*: Predicción Diaria**

Un usuario quiere contratar una instancia *c4.2xlarge* para uso personal durante siete días. Esta instancia es del tipo “optimizada para informática” y entre sus características más destacables podemos apreciar una CPU virtual de 8 *cores* o una memoria de 15 GiB, un ancho de banda de 750 Mbps y almacenamiento EBS. En su máquina, desea instalar la instancia con un sistema operativo de *SUSE Linux*. Si el usuario quiere contratar la instancia durante la semana del 17 al 24 de marzo, analizaremos los datos partiendo de los días anteriores.

Una vez que tengamos el preprocesado de los datos por días, analizaremos el comportamiento de los precios desde el 1 de febrero que es la fecha mínima que disponemos en nuestra BBDD para observar los datos hasta el día 17 de marzo.

### ***Análisis temporal***

Vamos a realizar un estudio sobre dos regiones que se han seleccionado para realizar un análisis temporal sobre las regiones y las zonas de disponibilidad donde podemos desplegar la instancia. Haremos un análisis sobre las regiones de *ap-northeast-1* y *ap-southeast-1*.

#### **Región de ap\_northeast\_1**

Graficamos los resultados de las dos zonas de disponibilidad para esta región, *ap-northeast-1a* y *ap-northeast-1c*:

Figura 22. Gráfica ap-northeast-1 (Experimento 1)

Como podemos comprobar en la gráfica de la *Figura 22*, tenemos en el eje de abscisas el periodo comprendido entre el día 1 de febrero y el día 17 de marzo de 2018. En el eje de ordenadas tenemos los precios máximos equivalentes a cada día en dólares *USD* desde los 0’205 $ a los 0’255 $.

Para la zona de *ap-northeast-1a,* desde el 1 de febrero hasta el día 28, los precios se mantienen constantes con un precio de 0’218 $. A partir del día 28 de febrero, empiezan a cambiar con el paso de los días. Para el día 16 de marzo, alcanza el máximo precio en nuestro periodo de tiempo con un valor de 0’228 $. Para el día 7 de marzo alcanza un pequeño descenso de los precios con un valor de 0’2219 $.

Para la zona de *ap\_northeast\_1c* podemos apreciar como el comportamiento de los precios ha permanecido constante en el mismo periodo que con la zona de *ap\_northeast\_1a*. A partir del 25 de febrero, los precios ascienden desde los 0’2189 $ a los 0’2381 $. A pesar de que la tendencia de los precios es parecida a la de la otra zona, no ha sido exactamente el mismo comportamiento con respecto a los últimos días.

#### **Región de ap\_southeast\_1**

La otra región que estudiaremos será *ap\_southeast\_1*. Esta región está compuesta por otras tres zonas de disponibilidad (*ap\_southeast\_1a, ap\_southeast\_1b, ap\_southeast\_1c*).

Figura 23. Gráfica ap-southeast-1 (Experimento 1)

En la gráfica de la *Figura 23*, al igual que en la *Figura 22*, tendremos en el eje de abscisas, el periodo comprendido entre el 1 de febrero y el 17 de marzo. En el eje de ordenadas, tenemos los precios registrados del sistema desde los 0’205 $ a los 0’255 $.

Para la zona de *ap-southeast-1a*, podemos comprobar como el precio permanece constante con valor de 0’18 $ hasta el día 18. A partir de este día, los precios, presentan una tendencia ascendente con el paso de los días.

En la zona de *ap-southeast-1b*, podemos destacar que también se presenta una tendencia constante con valor de 0’211 $ hasta el día 24 de febrero. Desde el 24 hasta el 17, los precios tienden a ascender e incluso presentan algunas irregularidades entre los primeros días de marzo. Se puede apreciar un intervalo de subida y de bajada reiterativo que podría parecerse a un periodo estacional.

Como peculiaridad la zona de *ap-southeast-1c* con respecto a las anteriores, podemos destacar que no existen periodos con el valor del precio constante. Además, se puede apreciar como existen muchas irregularidades entre picos superiores e inferiores que oscilan entre el valor de 0’210 $ y 2’60 $. También podemos destacar la existencia de una tendencia ascendente a partir del día 14 de febrero.

Con este análisis las dos regiones seleccionadas, podemos destacar que los precios han permanecido constantes durante casi todo el mes de febrero, a excepción de la zona de *ap\_southeast\_1c*. Así estudiaremos un caso apto donde apreciamos tendencia e irregularidades para hacer una comparación entre la eficiencia de los algoritmos. Sin embargo, se puede apreciar cómo los precios tienden a ascender para las cinco zonas de disponibilidad analizadas a partir de los últimos días del mes de febrero aproximadamente.

Para realizar nuestra predicción, como los resultados de las diferentes zonas de disponibilidad de las regiones han sido muy parecidos, nos quedaremos con la primera de todas. Por tanto, la localización donde estimaremos el precio *bid* para el usuario será en *ap\_northeast\_1a.*

Para este ejemplo en particular, solo vamos a considerar los datos desde el día 28 de febrero donde podemos apreciar precios variantes y dinámicos a la vez que estos presentan una tendencia ascendente clara para poder comparar más adelante tanto *Simple* como el *Double Exponential Smoothing*.

### ***Cálculo del coeficiente de Gini***

El siguiente paso será realizar una ejecución con el coeficiente de *Gini* para medir el nivel de dispersión de los datos. Como ya hemos comentado antes, solo tomaremos en cuenta la fecha de inicio desde el 28 de febrero y la fecha de fin hasta el 17. El resto de los parámetros serán los mismos como el sistema operativo, la región, etc.

El coeficiente de *Gini* nos da como resultado un valor de 0’0071. Este valor como se puede apreciar es casi insignificante, debido a que se encuentra entre el rango de un coeficiente de *Gini* bajo (entre 0 y 0’2). Además, este es muy cercano a 0, por tanto, podemos confirmar que estos datos apenas tienen variación entre ellos para los días 1 y 17 de marzo. A pesar de ser un resultado a priori pequeño, como decíamos anteriormente, en la práctica es natural. Esto es porque normalmente cuando el sistema actualiza los precios, si estamos estudiando el precio de un día para otro no suele dispersarse mucho en comparación de si lo estudiamos por semanas, por meses o por años. Cuanto mayor sea el tiempo que pasa en nuestro periodo, mayor será la probabilidad de que los precios sean desiguales. Por lo tanto, para el coeficiente de *Gini* en este caso, existe mucha igualdad entre los precios durante este periodo de días.

Estos serán los datos que utilizaremos para llevar a cabo la ejecución de los algoritmos.

### ***Eficiencia de los algoritmos de predicción***

Ya realizado un análisis previo gracias al estudio de las gráficas y al coeficiente de *Gini* para conocer un poco mejor el comportamiento de los datos, vamos a realizar la estimación del precio a entregar al usuario. Para ello, realizaremos una predicción con cada uno de nuestros algoritmos de predicción.

#### **Simple Exponential Smoothing**

Vamos a realizar el análisis con el *Simple Exponential Smoothing*. Para ello, lo primero que tenemos que hacer, será entrenar los datos para obtener el modelo. El valor de *alpha* de 1, nos minimizará el ECM que vale 5’55 × 10-7. Por tanto, el modelo tendrá una parametrización de un valor de *alpha* de 1. Mostramos los datos del conjunto de entrenamiento en la *Tabla* *1*.

|  |  |  |
| --- | --- | --- |
| **Periodo (*Días*)** | **Precio Sistema** | **Precio Estimado** |
| 28/02/2018 | 0,2184 | 0,2184 |
| 01/03/2018 | 0,2189 | 0,2184 |
| 02/03/2018 | 0,2201 | 0,2189 |
| 03/03/2018 | 0,2213 | 0,2201 |
| 04/03/2018 | 0,2221 | 0,2213 |
| 05/03/2018 | 0,2227 | 0,2221 |
| 06/03/2018 | 0,2226 | 0,2227 |
| 07/03/2018 | 0,2219 | 0,2226 |
| 08/03/2018 | 0,2228 | 0,2219 |
| 09/03/2018 | 0,2233 | 0,2228 |
| 10/03/2018 | 0,2235 | 0,2233 |
| 11/03/2018 | 0,2241 | 0,2235 |
| 12/03/2018 | 0,2248 | 0,2241 |
| 13/03/2018 | 0,2263 | 0,2248 |
| 14/03/2018 | 0,2267 | 0,2263 |
| 15/03/2018 | 0,2274 | 0,2267 |
| 16/03/2018 | 0,228 | 0,2274 |
| 17/03/2018 | 0,2276 | 0,228 |

Tabla 1. Entrenamiento Simple (Experimento 1)

En la *Figura 24*, podemos visualizar la gráfica correspondiente al entrenamiento del modelo.

Figura 24. Gráfica Entrenamiento (Experimento 1)

Como podemos apreciar en la gráfica, los precios que oscilan entre los 0’218 $ y 0’228 $ aproximadamente, son muy parecidos. Desde el 28 de febrero hasta el 6 de marzo, los precios del sistema están por encima de los estimados. A partir del día 5 hasta el 8 de marzo, ocurre lo contrario, el precio del suavizado supera al del sistema. Desde el día 8 hasta el 16 de marzo, el precio del sistema vuelve a superar al precio de la estimación. Sin embargo, para el día 17 de marzo, el precio estimado es superior al del sistema de nuevo, por lo tanto, deberemos de hacer el pronóstico para el conjunto de *test* y comprobar qué ocurre.

Finalmente, llevaremos cabo la evaluación con las métricas*.* Antes de ello, vamos a mostrar los datos de *test* en la *Tabla 2* con las observaciones futuras ya pronosticadas y esbozaremos la gráfica de la *Figura 25* con ellos.

|  |  |  |
| --- | --- | --- |
| **Periodo (*Días*)** | **Precio Sistema** | **Precio Estimado** |
| 18/03/2018 | 0,227 | 0,2276 |
| 19/03/2018 | 0,2262 | 0,2276 |
| 20/03/2018 | 0,2268 | 0,2276 |
| 21/03/2018 | 0,2265 | 0,2276 |
| 22/03/2018 | 0,2275 | 0,2276 |
| 23/03/2018 | 0,2288 | 0,2276 |
| 24/03/2018 | 0,2295 | 0,2276 |

Tabla 2. Test Simple (Experimento 1)

Figura 25. Gráfica Test Simple (Experimento 1)

Analicemos el comportamiento del pronóstico para el suavizado exponencial simple. Nuestro precio estimado, ha resultado ser una recta constante para los siete días de predicción con un valor de 0’2276 $. Desde el día 18 de marzo hasta el 22, nuestro pronóstico estaría por encima del verdadero que hubiera registrado el sistema. A pesar de ello, Para los días 23 y 24, el sistema supera el precio de nuestra estimación tomando un valor de hasta 0’2295 $ para el último día.

* *Error Cuadrático Medio*: El ECM para el conjunto de *test* tiene un valor de 1’3186 × 10-7.
* *Servicio Cumplido*: Nuestro pronóstico no ha sido bueno porque el precio del sistema, con un valor máximo de 0’2295 $, ha superado al nuestro que valía 0’2276 $. El sistema interrumpiría la instancia al usuario el día 22 de marzo.
* *Ahorro en coste*: Como nuestro pronóstico no ha sido bueno, la instancia *spot* se interrumpiría y por tanto no la podríamos comprar con el precio del resto de instancias para obtener el ahorro que le supondría al usuario.

#### **Double Exponential Smoothing**

Lo primero que debemos hacer para la predicción con este algoritmo, será entrenar el algoritmo por medio del conjunto de entrenamiento para obtener el modelo con el mínimo ECM. Los parámetros de *alpha* y *beta* para la configuración de nuestro modelo serán 0’9568 y 0’0506 respectivamente; obtenidos como resultado al minimizar el ECM con un valor de 3’8555 × 10-7. Así obtenemos los resultados del conjunto de entrenamiento con esta configuración tal y como puede visualizarse en la *Tabla 3*.

|  |  |  |
| --- | --- | --- |
| **Periodo (*Días*)** | **Precio Sistema** | **Precio Estimado** |
| 28/02/2018 | 0,2184 | 0,2184 |
| 01/03/2018 | 0,2189 | 0,2194 |
| 02/03/2018 | 0,2201 | 0,2199 |
| 03/03/2018 | 0,2213 | 0,2211 |
| 04/03/2018 | 0,2221 | 0,2223 |
| 05/03/2018 | 0,2227 | 0,2231 |
| 06/03/2018 | 0,2226 | 0,2237 |
| 07/03/2018 | 0,2219 | 0,2235 |
| 08/03/2018 | 0,2228 | 0,2228 |
| 09/03/2018 | 0,2233 | 0,2236 |
| 10/03/2018 | 0,2235 | 0,2241 |
| 11/03/2018 | 0,2241 | 0,2243 |
| 12/03/2018 | 0,2248 | 0,2249 |
| 13/03/2018 | 0,2263 | 0,2256 |
| 14/03/2018 | 0,2267 | 0,2271 |
| 15/03/2018 | 0,2274 | 0,2275 |
| 16/03/2018 | 0,228 | 0,2282 |
| 17/03/2018 | 0,2276 | 0,2288 |

Tabla 3. Entrenamiento Double (Experimento 1)

Figura 26. Gráfica Entrenamiento Double (Experimento 1)

Observamos en la gráfica de la *Figura 26*, la comparativa de los datos entre los precios del sistema frente a los precios estimados con nuestro algoritmo. El comportamiento de los datos tanto del precio del sistema como el pronosticado, es muy similar entre ellos. El precio del sistema desciende ligeramente en los días 7 y 8 de marzo para los precios del sistema y de nuestro pronóstico respectivamente. Mientras que, para el sistema, el precio llega entre esos días de 0’2227 $ hasta 0’2219 $, para nuestro pronóstico estimado llegan desde 0’2231 $ hasta 0’2235 $. Tanto los precios del sistema como los estimados tienden a ascender, siendo el valor de estos últimos algo mayor que los del sistema debido a que el *Double Exponential Smoothing* tiende a predecir valores con respecto a las tendencias de las observaciones pasadas.

Vamos a realizar la evaluación del modelo con los datos de prueba o de *test*, que serán los siete días posteriores a las observaciones del entrenamiento requeridos por el usuario. La *Tabla 4* muestra el conjunto de *test* con el periodo que comprende estos datos son desde el día 18 hasta el 24 de marzo ambos incluidos:

|  |  |  |
| --- | --- | --- |
| **Periodo (*Días*)** | **Precio Sistema** | **Precio Estimado** |
| 18/03/2018 | 0,227 | 0,2284 |
| 19/03/2018 | 0,2262 | 0,2291 |
| 20/03/2018 | 0,2268 | 0,2298 |
| 21/03/2018 | 0,2265 | 0,2305 |
| 22/03/2018 | 0,2275 | 0,2312 |
| 23/03/2018 | 0,2288 | 0,2319 |
| 24/03/2018 | 0,2295 | 0,2326 |

Tabla 4. Test Double (Experimento 1)

Ahora comprobamos los resultados del esbozo de la gráfica que se nos queda tal y como se muestra en la gráfica de la *Figura 27*.

Figura 27. Gráfica Test Double (Experimento 1)

Los valores de los precios para el sistema entre los días 18 y 21 de marzo han sufrido dos picos mínimos para los días 19 y 21 de marzo y un pico máximo entre medias de estos para el día 20. A partir del 21, vuelven a ascender hasta pasar de los 0’229 $. Mientras que el precio estimado, gracias a nuestro pronóstico en esos días, se mantiene en una línea constante por encima de los valores reales que ha marcado el sistema.

Realizamos pues, la evaluación por medio de nuestras métricas:

* *Error Cuadrático Medio*: Tras calcular el error cuadrático medio sobre nuestro conjunto de *test*, obtenemos un valor de 9’75429 × 10-6.
* *Servicio Cumplido*: El precio máximo real para el sistema se establece en 0’2295 $ para el último día, mientras que, para nuestra estimación, en ese mismo día, hemos predicho un precio para este mercado de instancia de 0’2326 $. Como nuestro precio es superior al propuesto por el sistema, nuestro pronóstico ha sido correcto. Con este precio, el sistema no interrumpirá la instancia al usuario durante su periodo requerido de contratación.
* *Ahorro económico*: Finalmente, antes de entregar el precio a nuestro usuario, vamos a demostrar que para esta instancia *spot*, el usuario puede ahorrarse económicamente una cifra significativa si hubiera decidido optar por cualquiera de los otros tipos de contratación. Gracias al sitio oficial de *AWS*, podemos buscar los precios para la instancia *c4.2xlarge* para *SUSE Linux* en la región de *ap\_northeast\_1* para las instancias reservadas y bajo demanda [51]. Para ello, compararemos los precios de la *Tabla 5*:

|  |  |  |
| --- | --- | --- |
|  | **Tipo de contratación de instancia** | |
| *Reservada* | *Bajo Demanda* |
| **Precio por hora (*$*)** | 0,504 | 0,504 |
| **Ahorro (*%*)** | 53,85 | 53,85 |

Tabla 5. Ahorro Económico Double (Experimento 1)

Como tanto la instancia reservada como bajo demanda tienen el mismo precio, el usuario se podría ahorrar hasta un 53’85 % si optase por una instancia *spot* en lugar de cualquier otra.

Comparemos los resultados de la evaluación entre el *Simple* y el *Double Exponential Smoothing* en la *Tabla 6*.

|  |  |  |
| --- | --- | --- |
|  | ***Simple Exponential Smoothing*** | ***Double Exponential Smoothing*** |
| **ECM** | 1’3186 × 10-7 | 9’75429 × 10-6 |
| **Servicio Cumplido** | NO | SI |
| **Ahorro en coste** | - | 53,85 % |

Tabla 6. Simple vs Double (Experimento 1)

El *Double Exponential Smoothing*, valida correctamente la métrica de ahorro en coste. Sin embargo, para el Simple, como ya hemos comentado, no puede validarse por esta métrica debido a la interrupción de la instancia para este caso. Como conclusión de este caso de prueba, el *Double Exponential Smoothing* saldría ganando puesto que cubre el servicio con el precio adecuado para que el sistema no interrumpa la instancia.

## **Experimento 2 – Demanda instancia *spot*: Predicción Semanal**

Para este caso, se nos presenta un usuario que desea contratar una instancia del tipo *m4.2xlarge* para el sistema *Linux/Unix*. Esta instancia es del tipo “optimizada para informática” y entre sus carcterísticas presenta las siguientes: CPU de 8 *cores*, memoria de 32 GiB, almacenamiento solo EBS y un ancho de banda de 1000 Mbps. El periodo de contratación que desea es de cuatro semanas a partir de la segunda quincena de agosto. Vamos a aplicar las técnicas de nuestro sistema para entregarle a este usuario un *bid* adecuado a sus necesidades para ese periodo de tiempo sin interrupciones.

Como en este caso, el usuario desea pronosticar semanas en lugar de días, emplearemos el mismo procedimiento de técnicas realizado en el anterior experimento, pero esta vez con semanas. Por lo tanto, los datos quedarán preprocesados por semanas.

El paso siguiente, será estudiar la actuación de los datos en diferentes regiones y zonas de disponibilidad. Vamos a observar los datos desde el 1 de febrero que disponemos de datos hasta el 15 de agosto.

### ***Análisis temporal***

Vamos a seleccionar la región de *us-east-2* y *us-west-1* para analizar el comportamiento de cada una de sus zonas de disponibilidad.

#### **Región de us-east-2**

Para esta región tenemos disponibles la zona de *us-east-2a*, *us-east-2b* y *us-east-2c*.

Figura 28. Gráfica us-east-2 (Experimento 2)

Procedemos analizar los resultados de la gráfica de la *Figura 28*. En el eje de abscisas tenemos el periodo en semanas comprendido entre el 1 de febrero y el 15 de agosto. En el eje de ordenadas tenemos los precios desde los 0 $ hasta los 0’25 $.

Para la zona de *us-east-2a* podemos apreciar un intervalo constante de los precios entre la primera semana y la 8 con un valor de 0’07 $ aproximadamente. Luego, el precio del sistema asciende hasta alcanzar un máximo para la semana 13 con valor de 0’02171 $. Tras esta semana, el valor vuelve a descender hasta la semana 17 donde llega hasta los 0’0791 $. Desde la semana 17 a las 21, el precio vuelve a pasar por un intervalo constante que oscila entre los 0’07 $ y 0’08 $ aproximadamente.

Para la zona de *us-east-2b*, el comportamiento de los precios es similar al anterior. Permanece constante con valor de 0’071 $ las 8 primeras semanas, Después, el precio asciende hasta los 0’16 $ más o menos. De la semana 15 a la 23, el valor del precio vuelve a descender.

En la zona de *us-east-2c,* se puede apreciar que, durante las 8 primeras semanas, al igual que con las otras zonas, se presenta un precio constante con valor de 0’071 $. A partir de la semana 9, el precio asciende hasta llegar a un valor de 0’1036 $ para la semana 13. Pasada la semana 13, el precio vuelve a descender hasta un valor de 0’071 $ para las últimas semanas.

Los precios se comportan de manera similar en las tres zonas. La diferencia es que el valor del precio del periodo varía significativamente entre las zonas. El precio para *us-east-2a* es superior al de *us-east-2b* y este es superior al de *us-east-2c.* Por tanto, para esta región, tenemos un periodo con tendencia ascendente y otro periodo con tendencia descendente para las tres zonas de disponibilidad.

#### **Región de us-west-1**

En esta región se encuentran las zonas de *us-west-1a* y *us-west-1b*.

Figura 29. Gráfica us-west-1 (Experimento 2)

Para la *Figura 29*, tendremos como valores en el eje de ordenadas los precios comprendidos entre 0 $ y 0’5 $. En el eje de abscisas tendremos el mismo periodo con las 23 semanas comprendidas entre el 1 de febrero y el 15 de agosto. Pasamos a comentar la tendencia de los precios para las zonas de disponibilidad.

En la zona de *us-west-1b*, vemos como en las 9 primeras semanas, el precio presenta un intervalo entre los 0’13 $ y los 0’17 $ bastante irregular con una pequeña tendencia ascendente. Tras la semana 9, hasta la 13, la tendencia de los precios se presenta con una pendiente descendiente desde los 0’16 $ hasta los 0’11 $ aproximadamente. A partir de esta semana hasta la última que observamos, el precio permanece constante en unos 0’11 $.

Para *us-west-1c*, el precio oscila entre 0’11 $ y 0’13 $, llegando a un pico máximo entre la semana 5 y 6. Después, el precio se mantiene constante en 0’11 $.

Procederemos a estudiar la zona de disponibilidad de *us-east-2a* desde semana 13 para analizar el descenso del precio.

### ***Cálculo del coeficiente de Gini***

El siguiente paso será aplicar el coeficiente de *Gini*. Una vez que ejecutemos el *script* con los parámetros correspondientes, analizaremos el resultado. El coeficiente de *Gini* retornado para este caso es de 0’1817. Este valor es un valor de rango pequeño, pero es mayor que el de nuestro experimento anterior por días. Esto es porque al trabajar con semanas en lugar de con días, el valor de los precios está mucho más variado de una semana a otra que de un día para otro. Por lo tanto, a pesar de ser un valor pequeño y cercano a 0’2, podemos considerar que existe una pequeña desigualdad entre el rango de valores para nuestro conjunto de entrenamiento. Esto es porque en la primera semana tenemos un valor de casi el doble de precio que el de las semanas posteriores y, por tanto, para ese valor en concreto, podemos indicar que toma gran parte de la “riqueza de valores” en comparación con el resto de los precios.

### ***Eficiencia de los algoritmos de predicción***

En esta sección, ejecutaremos y compararemos los resultados mediante los dos algoritmos de predicción que tenemos.

#### **Simple Exponential Smoothing**

Vamos a realizar la predicción aplicando la técnica del suavizado exponencial simple. Primero de todo obtenemos la configuración adecuada de parámetros para nuestro modelo. El resultado después de minimizar el ECM con valor de 0’0008 es un valor de *alpha* de 1, por lo tanto, vamos a darle únicamente peso a la primera observación anterior. Veamos los resultados de nuestro conjunto de entrenamiento en la *Tabla 7*.

|  |  |  |
| --- | --- | --- |
| **Periodo (*Semanas*)** | **Precio Sistema** | **Precio Estimado** |
| Semana 13 | 0,2171 | 0,2171 |
| Semana 14 | 0,1705 | 0,2171 |
| Semana 15 | 0,0915 | 0,1705 |
| Semana 16 | 0,0803 | 0,0915 |
| Semana 17 | 0,0791 | 0,0803 |
| Semana 18 | 0,0784 | 0,0791 |
| Semana 19 | 0,0769 | 0,0784 |
| Semana 20 | 0,0787 | 0,0769 |
| Semana 21 | 0,0876 | 0,0787 |
| Semana 22 | 0,0885 | 0,0876 |
| Semana 23 | 0,0899 | 0,0885 |

Tabla 7. Entrenamiento Simple (Experimento 2)

A continuación, analizaremos la gráfica correspondiente:

Figura 30. Gráfica Entrenamiento Simple (Experimento 2)

En la *Figura 30*, tenemos el comportamiento del entrenamiento de los datos con el *Simple Exponential Smoothing*. En el eje de ordenadas tenemos los precios desde 0 $ hasta 0’25 $. En el eje de abscisas tenemos el periodo comprendido en la semana 13 y la 23 que disponemos en nuestro conjunto de entrenamiento. Como podemos apreciar, tanto para el precio del sistema como el de nuestro pronóstico, la tendencia de los precios es descendente. Para los precios del sistema, durante las semanas 13,14 y 15, se puede apreciar un descenso desde los 0’3 $ hasta los 0’1 $ aproximadamente. Desde esta semana hasta la última el precio varía muy poco entre el intervalo de 0’07 $ y 0’1 $. Para los precios del pronóstico, el descenso se lleva a cabo entre las semanas 13, 14, 15 y 16 desde los 0’2 $ hasta los 0’1 $ al igual que los precios reales. De la semana 16 en delante, el precio varía escasamente entre los 0’05 $ y los 0’1 $. Los resultados son muy similares entre las dos tendencias.

Ahora analizaremos los datos con el conjunto de *test* para después evaluarlo. Veamos los resultados en la *Tabla 8*.

|  |  |  |
| --- | --- | --- |
| **Periodo (*Semanas*)** | **Precio Sistema** | **Precio Estimado** |
| Semana 24 | 0,0899 | 0,0899 |
| Semana 25 | 0,0848 | 0,0899 |
| Semana 26 | 0,0781 | 0,0899 |
| Semana 27 | 0,0777 | 0,0899 |

Tabla 8. Test Simple (Experimento 2)

En la gráfica de la *Figura 31*, podemos analizar visualmente los resultados.

Figura 31. Gráfica Test Simple (Experimento 2)

Para la *Figura 31,* en el eje de ordenadas disponemos del intervalo de los precios entre 0 $ y 0’1 $. En el eje de abscisas tenemos el periodo de las cuatro semanas del conjunto de evaluación. El comportamiento para nuestra predicción permanece constante con un valor de 0’089 $ durante las cuatro semanas. En cuanto a los precios del sistema, se puede ver a partir de la semana 25 se va reduciendo el valor de los precios desde los 0’089 $ hasta alcanzar la cifra de unos 0’079 $ aproximadamente.

Veamos a continuación la evaluación del modelo en cuanto a las métricas:

* *Error Cuadrático Medio*: El ECM para el conjunto de *test* tiene un valor de 7’85 × 10-5.
* *Servicio Cumplido*: Para este caso, el precio máximo durante las cuatro semanas pronosticadas ha sido 0’0899 $ para nuestra predicción. El precio máximo real que ha alcanzado el sistema para ese periodo ha sido de 0’0899 $. Como el precio del pronóstico ha conseguido igualar la cifra del precio máximo del sistema durante estas cuatro semanas, nuestro pronóstico ha sido exitoso y podemos decir que el servicio se ha cumplido.
* *Ahorro en coste*: En la *Tabla 9*, se puede ver el ahorro que supone para el usuario entre el precio de las instancias reservadas y bajo demanda frente al nuestro [51]:

|  |  |  |
| --- | --- | --- |
|  | **Tipo de contratación de instancia** | |
| *Reservada* | *Bajo Demanda* |
| **Precio por hora (*$*)** | 0,4 | 0,4 |
| **Ahorro (*%*)** | 77,53 | 77,53 |

Tabla 9. Ahorro Económico Simple (Experimento 2)

El usuario ahorraría hasta un 77’53 % si decide contratar su instancia con el precio estimado para una instancia *spot*.

#### **Double Exponential Smoothing**

Vamos a aplicar la técnica del *Double Exponential Smoothing*. Tras varias ejecuciones con diferente parametrización, finalmente nos quedamos con un *alpha* de 0’8721 y un valor *beta* de 0’5855 con un ECM de 5’34 ×10-4 para nuestro conjunto de entrenamiento. Los valores de los precios para el conjunto de entrenamiento aparecen en la *Tabla 10*.

|  |  |  |
| --- | --- | --- |
| **Periodo (*Semanas*)** | **Precio Sistema** | **Precio Estimado** |
| Semana 13 | 0,2171 | 0,2171 |
| Semana 14 | 0,1705 | 0,1715 |
| Semana 15 | 0,0915 | 0,1245 |
| Semana 16 | 0,0803 | 0,0327 |
| Semana 17 | 0,0791 | 0,0355 |
| Semana 18 | 0,0784 | 0,0571 |
| Semana 19 | 0,0769 | 0,0702 |
| Semana 20 | 0,0787 | 0,0739 |
| Semana 21 | 0,0876 | 0,0785 |
| Semana 22 | 0,0885 | 0,0914 |
| Semana 23 | 0,0899 | 0,0924 |

Tabla 10. Entrenamiento Double (Experimento 2)

Vamos a ver la diferencia entre el precio del sistema frente al precio pronosticado por nuestro algoritmo en los resultados de la gráfica de la *Figura 32*.

Figura 32. Gráfica Entrenamiento Double (Experimento 2)

Como podemos apreciar en la gráfica, los valores del precio del sistema con el nuestro pronosticado son muy parecidos. Para el precio del sistema, los precios permanecen en un intervalo constante justo después del descenso a partir de la semana 15. Pero para los precios que hemos estimado, debido al suavizado de esa aproximación después del descenso, los precios tienden a bajar hasta llegar a un mínimo para la semana 16 con valor de 0’327 $. Acto seguido, los valores de los precios tanto del sistema como los estimados tienden a permanecer en el mismo rango de precios sobre 0’07 $ y hasta 0’09 $ la hora.

A continuación, vamos a hacer nuestro pronóstico y evaluación para las cuatro semanas futuras después del rango de valores de entrenamiento.

Los precios del sistema y los precios estimados del conjunto de *test* aparecen en la *Tabla 11* y para mayor detalle visual en la gráfica de la *Figura 33*. Las semanas a predecir el precio serán las semanas 24 ,25 ,26 y 27.

|  |  |  |
| --- | --- | --- |
| **Periodo (*Semanas*)** | **Precio Sistema** | **Precio Estimado** |
| Semana 24 | 0,0899 | 0,0924 |
| Semana 25 | 0,0848 | 0,0946 |
| Semana 26 | 0,0781 | 0,0968 |
| Semana 27 | 0,0777 | 0,099 |

Tabla 11. Test Double (Experimento 2)

Figura 33. Gráfica Test Double (Experimento 2)

Los valores de los precios tanto estimados como reales están representados en el eje de ordenadas entre 0 $ y 0’12 $. En el eje de abscisas aparecen las cuatro semanas a pronosticar el precio. Analizando la gráfica, se puede apreciar un leve descenso de los precios reales por parte del sistema desde los 0’0899 $ hasta los 0’0777 $. Sin embargo, para nuestro pronóstico, predecimos que los precios han ascendido desde los 0’0924 hasta los 0’099 $ la hora. Por lo tanto, a simple vista, podemos decir que nuestro pronóstico ha sido óptimo ya que nuestros valores han superado a los del sistema.

Dicho esto, vamos a evaluar los resultados atendiendo a nuestras métricas:

* *Error Cuadrático Medio*: El ECM para el conjunto de *test* vale 2’264 × 10-4.
* *Servicio Cumplido*: El precio máximo durante las cuatro semanas a predecir para el usuario ha sido 0’099 $ para nuestra predicción. Sin embargo, el precio máximo real que ha alcanzado el sistema para ese periodo ha sido de 0’0899 $. Si le hubiéramos entregado al usuario el precio de 0’099 $ para pujar, hubiera salido ganando ya que el sistema no le hubiera interrumpido la instancia en ningún momento.
* *Ahorro en coste*: Vamos a hacer otra evaluación frente al precio del resto de tipos de contratación. Para ello vamos a crear una tabla como en el experimento 1 donde pueda apreciarse la diferencia de precios entre las instancias reservadas, bajo demanda y nuestro pronóstico para las instancias *spot*. Así, teniendo en cuenta la región y el tipo de instancia especificados por el usuario, podemos representar la comparativa de precios en la *Tabla* *12* [51]:

|  |  |  |
| --- | --- | --- |
|  | **Tipo de contratación de instancia** | |
| *Reservada* | *Bajo Demanda* |
| **Precio por hora (*$*)** | 0,4 | 0,4 |
| **Ahorro (*%*)** | 75,25 | 75,25 |

Tabla 12. Ahorro Económico Double (Experimento 2)

Como podemos apreciar, el precio por hora para nuestra instancia *spot* supera económicamente también al precio de las instancias reservadas o bajo demanda. Si el usuario se hubiera decantado por contratar una instancia que no sea *spot*, le saldría mucho más cara la contratación de la instancia debido a las políticas de *Amazon* para periodos de contratación de estas instancias por cada año o incluso tres. Como el usuario simplemente necesitaba cuatro semanas el uso de la instancia, saldría ganado contratando una instancia *spot* por 0’099 $ gracias a la predicción del *Double Exponential Smoothing*.

Finalmente, nos queda por hacer una comparativa entre las métricas de evaluación de los dos algoritmos, esta se puede entender mejor en la *Tabla 13*.

|  |  |  |
| --- | --- | --- |
|  | ***Simple Exponential Smoothing*** | ***Double Exponential Smoothing*** |
| **ECM** | 7’85 × 10-5 | 2’264 × 10-4 |
| **Servicio Cumplido** | SI | SI |
| **Ahorro en coste** | 77,53 % | 75,25 % |

Tabla 13. Simple vs Double (Experimento 2)

Desde el punto de vista del usuario, tanto el *Simple* como el *Double Exponential Smoothing* han valido para que su instancia no se interrumpa con ellos respectivos precios obtenidos durante las cuatro semanas. A pesar de ello, el simple ha conseguido igualar la cifra durante el periodo futuro de contratación, por lo que el usuario no pagaría de más la instancia. De cara al ahorro, el *Double Exponential Smoothing*, tiene casi un 2 % más de ahorro que el *Simple*. Desde el punto de vista computacional, el *Simple* ha obtenido menos error que el *Double* debido a que ha ajustado mejor a los datos de entrenamiento. Para este caso, saldría ganando el *Simple Exponential Smoothing* ya que supera tanto en ECM como en ahorro al *Double*.

## **Experimento 3 – Demanda instancia *spot*: Predicción Mensual**

Para el último de nuestros experimentos, tenemos un usuario que desea contratar una instancia del tipo *i3.2xlarge* con el sistema operativo *SUSE Linux.* Con esta instancia “optimizada para almacenamiento”, el usuario podrá disfrutar de un modelo con 8 *cores* de CPU, 61 GiB de memoria, hasta 10 Gbps de rendimiento en redes y almacenamiento con 1’9 SSD NVMe. El usuario quiere contratar esta instancia durante los meses de agosto y septiembre sin que la máquina le interrumpa la instancia en ningún momento. Vamos a estimar el precio (*bid*) que debería pagar por hora durante dos meses con una instancia *spot,* que sea lo suficientemente alto para que la máquina no se interrumpa y lo suficientemente bajo para que salga más barato que el resto de las instancias.

Para empezar, vamos a recoger de nuestro conjunto de datos, todos los disponibles que tenemos desde el mes de febrero hasta julio para entrenarlos. De esta forma, dejaremos los meses de agosto y septiembre para evaluarlos. Los datos han de estar previamente preprocesados por meses.

### ***Análisis temporal***

Vamos a hacer un análisis sobre las series temporales comprendidas entre los meses de febrero hasta julio para las regiones de *eu-central-1* y *ap-northeast-1*.

#### **Región de eu-central-1**

Para esta región, nos encontramos con tres zonas de disponibilidad que serán *eu-central-1a*, *eu-central-1b* y *eu-central-1c*.

Figura 34. Gráfica eu-central-1 (Experimento 3)

Vamos a hacer algunos comentarios acerca de la gráfica de la *Figura 34* correspondiente a los precios de las zonas de disponibilidad de la región de *eu-central-1*. En el eje de ordenadas tenemos los precios del sistema registrados desde los 0’32 $ hasta los 0’365 $. En el eje de abscisas tenemos el periodo establecido con todos los datos de los precios desde el mes de febrero hasta el mes de julio.

Analizando la zona de *eu-central-1a*, podemos apreciar como durante los meses de febrero y mayo hay una tendencia ascendente desde los 0’325 $ aproximadamente hasta alcanzar el máximo en 0’36 $. Tras el mes de mayo hasta junio, los precios vuelven a descender hasta los 0’32 $. A partir de junio, los precios vuelven a ascender hasta alcanzar u precio entre los 0’35 $ y los 0’355 $. El comportamiento de los precios, por lo tanto, es bastante irregular con respecto a los meses que disponemos.

El comportamiento para la zona de *eu-central-1b* y *eu-central-1c* es prácticamente el mismo. Los precios se mantienen constantes durante todos los meses con un valor de 0’3232 $. No ha habido dinámico durante estos meses para estas dos zonas de disponibilidad.

#### **Región de ap-northeast-1**

En esta región tenemos disponibles las zonas de *ap-northeast-1a*, *ap-northeast-1c* y *ap-northeast-1d*. Vamos a estudiar el comportamiento de las tres zonas apoyándonos en el esbozo de la gráfica de la *Figura 35*.

Figura 35. Gráfica ap-northeast-1 (Experimento 3)

En el eje de ordenadas tenemos los datos de los precios desde los 0’3 $ hasta los 0’6 $ para medir la dinámica de estos. En el eje de abscisas, tenemos el conjunto de meses disponibles desde febrero hasta julio de 2018.

Para la zona de *ap-northeast-1a*, durante los meses de febrero y marzo se pude ver un ascenso de los precios entre los 0’45 $ y los 0’55 $ aproximadamente. El mes de marzo representa el pico máximo de los precios para nuestro conjunto de meses. Después se presenta un descenso hasta el mes de abril hasta los 0’35 $. Luego presenta otro leve descenso hasta el mes de mayo y a partir de este hasta julio, el precio constante sobre aproximadamente los 0’33 $.

En la zona de *ap-northeast-1c*, se puede apreciar otro ascenso durante los meses de febrero y marzo desde una cifra de 0’4 $ hasta los 0’45 $. Entre los meses de marzo y abril, los precios vuelven a descender hasta unos 0’33 $. A partir de abril hasta julio, los precios permanecen en un intervalo constante entre los 0’3 $ y los 0’35 $.

La última zona de *ap-northeast-1d* permanece con un precio constante durante los seis meses que tenemos con un valor entre 0’32 $ y 0’33 $.

Al tener tan pocos datos para el conjunto de entrenamiento, no podemos hacer una ampliación para estudiar alguna parte en concreto del periodo ya que no tendríamos suficientes datos para entrenar. Por lo tanto, estudiaremos este caso con todos los datos que disponemos en nuestra BBDD con los meses de febrero y julio. Dejaremos para el conjunto de evaluación los meses de agosto y septiembre. El conjunto de entrenamiento que seleccionaremos para la predicción serán los datos de la zona de disponibilidad de *ap-northeast-1a*, para así estudiar un comportamiento intermedio entre *ap-northeast-1c y ap-northeast-1d.*

### ***Cálculo del coeficiente de Gini***

El siguiente paso que realizaremos, será hacer una medición con el coeficiente de *Gini*. El coeficiente de *Gini* para los meses de febrero hasta julio tiene un valor de 0’066. Como podemos comprobar, este valor de *Gini* es muy pequeño y cercano a 0, a diferencia del de las semanas que nos retornó un valor muy cercano a 0’2. Al ser un valor tan pequeño, podemos decir que los precios apenas han variado en estos seis meses y por lo tanto esta instancia no ha sufrido muchas demandas durante este periodo de tiempo. Técnicamente, este valor tan cercano a 0 se debe a que simplemente tenemos seis datos para estudiar en nuestro conjunto de entrenamiento, y además no hay casi variación entre estos datos puesto que la mayoría de los meses permanecen en un intervalo constante de precios. Solamente podemos apreciar un máximo para el mes de marzo que desequilibra o descompensa en corta medida el valor constante de los datos.

### ***Eficiencia de los algoritmos de predicción***

Vamos a realizar a continuación la predicción del precio *bid* para el usuario aplicando los dos algoritmos que tenemos. Para empezar, vamos a ver los resultados para el *Simple Exponential Smoothing*.

#### **Simple Exponential Smoothing**

Primero de todo, necesitamos encontrar la mejor configuración de parámetros para obtener el mejor modelo del conjunto de entrenamiento. En este caso, solamente tenemos un parámetro para el *Simple Exponential Smoothing* que es *alpha*. Hacemos varias ejecuciones con diferente parametrización de *alpha* que obtengamos aquella que nos minimice el error cuadrático medio. El parámetro de *alpha* con un valor de 0’655 nos da el menor ECM con un valor de 2’27 × 10-3. Por lo tanto, nos quedaremos un con un *alpha* de 0’655 para nuestro conjunto de entrenamiento y analizaremos los resultados de la predicción del entrenamiento en la *Tabla 14*:

|  |  |  |
| --- | --- | --- |
| **Periodo (*Meses*)** | **Precio Sistema** | **Precio Estimado** |
| Febrero | 0,4027 | 0,4027 |
| Marzo | 0,449 | 0,4027 |
| Abril | 0,3357 | 0,433 |
| Mayo | 0,3342 | 0,3692 |
| Junio | 0,3297 | 0,3463 |
| Julio | 0,3196 | 0,3354 |

Tabla 14. Entrenamiento Simple (Experimento 3)

A continuación, vamos a visualizar los resultados en una gráfica para comentar las diferencias entre los valores del sistema frente a los predichos por nuestro algoritmo:

Figura 36. Gráfica Entrenamiento Simple (Experimento 3)

Analizando la gráfica de la *Figura 36*, apreciamos que, en vista de los resultados, el precio del nuestro suavizado es muy parecido al precio del sistema. Sin embargo, el pico máximo que teníamos en el precio del sistema en el mes de marzo con 0’449 $ no es diferente al pico que muestra la línea de tendencia de la predicción. Para el precio estimado, el pico máximo de los datos se encuentra en el mes de abril con un valor 0’433 $. Esto es debido al suavizado del pronóstico con las ecuaciones de nuestro algoritmo al trabajar con las observaciones pasadas reales para predecir. A partir del mes de abril hasta el mes de mayo conforme a nuestro pronóstico, se presenta un leve descenso de los precios desde los 0’44 a los 0’36 $ aproximadamente. Siguiendo los meses de mayo, junio y julio, el comportamiento de los datos se establece en un intervalo constante parecido al del sistema. Aun así, el precio de nuestro pronóstico suavizado para los meses de entrenamiento queda levemente superior que al precio de los del sistema, por lo que vamos por buen camino para evaluar los meses futuros.

El siguiente paso que toca hacer será la evaluación del modelo. Para ello, vamos a ver en la *Tabla 15* los datos de los precios reales del sistema frente a los precios pronosticados por el algoritmo para los meses de agosto y septiembre:

|  |  |  |
| --- | --- | --- |
| **Periodo (*Meses*)** | **Precio Sistema** | **Precio Estimado** |
| Agosto | 0,3196 | 0,325 |
| Septiembre | 0,3196 | 0,3215 |

Tabla 15. Test Simple (Experimento 3)

Vamos a observar el comportamiento de estos datos visualmente en la gráfica de la *Figura 37* para ver si el resultado ha sido bueno o no, a pesar de que, a simple vista en la tabla, se puede apreciar que efectivamente, ha sido óptimo.

Figura 37. Gráfica Test Simple (Experimento 3)

Para este caso, tal y como podemos observar en la gráfica, nuestra predicción queda por encima de los valores de los precios reales del sistema. Por lo que podemos decir que hemos acertado la predicción. El rango de valores para el eje de ordenadas no empieza desde 0 si no que para mejorar la visualización se ha realizado un aumento en la parte comprendida entre los 0’316 $ y los 0’326 $. En el eje de abscisas simplemente tenemos dos valores para el mes de agosto y de septiembre. El precio del sistema se ha mantenido constante para los meses de agosto y septiembre con un precio de 0’3196 $, tal y como podíamos suponer tras analizar la fase final de los meses de entrenamiento.

Evaluando los resultados con respecto a nuestro sistema de métricas, obtenemos las siguientes conclusiones:

* *Error Cuadrático Medio*: El error cuadrático medio para el conjunto de evaluación entre el sistema y el pronóstico ha sido de 1’6385 × 10-5.
* *Servicio Cumplido*: El precio máximo de nuestro conjunto de *test* estimado ha sido 0’325 $ para el mes de agosto. Como este precio ha sido superior al valor máximo del sistema, que para este caso ha sido igual tanto para agosto como para septiembre (0’3196 $), el precio pronosticado ha sido correcto.
* *Ahorro en coste*: En la *Tabla 16*, evaluaremos los resultados del modelo para este experimento comparando nuestro precio de instancia *spot* con el del resto de tipos de contratación con los mismos requisitos del usuario [51].

|  |  |  |
| --- | --- | --- |
|  | **Tipo de contratación de instancia** | |
| *Reservada* | *Bajo Demanda* |
| **Precio por hora (*$*)** | 0,732 | 0,732 |
| **Ahorro (*%*)** | 55,60 | 55,60 |

Tabla 16. Ahorro Económico Simple (Experimento 3)

El precio para las instancias reservadas y bajo demanda se mantiene igual de precio tanto anualmente como para contratos de tres años, tal y como se puede comprobar en la *Tabla 16*.

#### **Double Exponential Smoothing**

Lo primero que debemos hacer será obtener el modelo mediante el conjunto de entrenamiento para evaluar con el conjunto de *test*. La parametrización que minimiza el ECM para nuestro conjunto de entrenamiento es un valor de *alpha* de 0’5467 y un valor beta de 0’3446con un error de 2’1911 × 10-2. El siguiente paso, será comentar los resultados por medio de la *Tabla 17* y la *Figura 38*.

|  |  |  |
| --- | --- | --- |
| **Periodo (*Meses*)** | **Precio Sistema** | **Precio Estimado** |
| Febrero | 0,4027 | 0,4027 |
| Marzo | 0,449 | 0,4027 |
| Abril | 0,3357 | 0,433 |
| Mayo | 0,3342 | 0,3692 |
| Junio | 0,3297 | 0,3463 |
| Julio | 0,3196 | 0,3354 |

Tabla 17. Entrenamiento Double (Experimento 3)

Figura 38. Gráfica Entrenamiento Double (Experimento 3)

En la gráfica de la *Figura 38* tenemos la representación gráfica de la dinámica de los precios para el conjunto de entrenamiento. El eje de ordenadas indica los precios desde un valor de 0 $ hasta 0’5 $. En el eje de abscisas, podemos observar el periodo entre los meses de febrero y julio. Con respecto al precio del sistema, para el mes de febrero se puede analizar una tendencia ascendente hasta marzo desde los 0’4 $ hasta los 0’45 $. Desde marzo hasta abril, el precio vuelve a descender hasta unos 0’34 $. Del mes de abril hasta julio, el precio continúa en un intervalo constante en 0’33 $. Para el precio de nuestro pronóstico, la tendencia ascendente se presenta desde febrero hasta abril desde los 0’35 $ hasta algo más de 0’4 $. Seguidamente, el valor desciende de nuevo hasta el mes de julio alcanzando un valor aproximado de 0’35 $. Tanto el precio del sistema como el de nuestro pronóstico, convergen a partir de los meses de junio y julio.

Veamos cual ha sido el comportamiento al añadir el conjunto de evaluación o *test* aplicando nuestro modelo. Visualizamos los resultados en la *Tabla 18* y en la *Figura 39*.

|  |  |  |
| --- | --- | --- |
| **Periodo (*Meses*)** | **Precio Sistema** | **Precio Estimado** |
| Agosto | 0,3196 | 0,325 |
| Septiembre | 0,3196 | 0,3215 |

Tabla 18. Test Double (Experimento 3)

Figura 39. Gráfica Test Double (Experimento 3).

En el eje de ordenadas para la *Figura 39* tenemos el precio que representa el conjunto de evaluación del pronóstico con unos valores entre 0’316 $ y 0’326 $. En el eje de abscisas tenemos el periodo entre los meses de agosto y septiembre. Mientras que el precio marcado por el sistema no varía de los 0’3196 $, el precio de nuestra estimación tampoco varía y permanece constante para con 0’3175 $ durante los dos meses.

Si validadnos los resultados aplicando las tres métricas descritas de evaluación obtenemos lo siguiente:

* *Error Cuadrático Medio*: El ECM para el conjunto de evaluación entre el sistema y el pronóstico ha sido de 4’41 × 10-6.
* *Servicio Cumplido*: Como el precio máximo del sistema es de 0’3196 $ y supera al de nuestro pronóstico que toma el valor de 0’3175 $, no hemos acertado la predicción desde el punto de vista de esta métrica.
* *Ahorro en coste*: Al no haber acertado la predicción anterior, si el usuario contratara una instancia *spot* por el precio que, estimado por nuestro sistema, la instancia sería interrumpida y por tanto no necesitamos calcular el ahorro supuesto frente a las instancias reservadas y bajo demanda.

Finalmente, vamos a realizar una comparativa de la evaluación de los dos algoritmos en la *Tabla 19*:

|  |  |  |
| --- | --- | --- |
|  | ***Simple Exponential Smoothing*** | ***Double Exponential Smoothing*** |
| **ECM** | 4’41 × 10-6 | 1’6385 × 10-5 |
| **Servicio Cumplido** | SI | NO |
| **Ahorro en coste** | 56,63 % | - |

Tabla 19. Simple vs Double (Experimento 3)

Comparando el ECM de ambos algoritmos, vemos que el *Simple Exponential Smoothing*, presenta menos error para la evaluación que el *Double*. El servicio cumplido para el *Double Exponential Smoothing* no ha funcionado mientras que para el *Simple* sí. Por lo tanto, en vista de los resultados y de la comparativa, el *Simple Exponential Smoothing* ha sido más eficiente que el *Double* para este caso, debido a que el servicio se ha cumplido correctamente.

# **CONCLUSIONES Y PROPUESTAS DE MEJORA**

## **Conclusiones**

Una vez finalizado todo el desarrollo del *TFG*, podemos concluir la memoria repasando si todos los objetivos de este trabajo se han cumplido. Para empezar, hemos conseguido realizar un estudio sobre las características del entorno y hemos estudiados conceptos sobre la plataforma pública en la nube de *Amazon.* Se ha profundizado en la búsqueda de técnicas y estadísticas aptas para realizar nuestras predicciones con los algoritmos adecuados. Conseguimos implementar el *script* para la descarga de datos históricos desde *AWS*. Los datos descargados han sido ejecutados por medio de los algoritmos adecuadamente, ya que han sido sometidos previamente a un preprocesado o limpieza de estos. Hemos evaluado los resultados de los precios pronosticados para comprobar si el precio que entregamos finalmente al usuario es óptimo o no atendiendo a sus necesidades. Se ha comparado la eficiencia de los dos algoritmos de predicción para cada uno de los casos en función de las tres métricas propuestas.

Podemos destacar la adquisición de nuevos conocimientos sobre conceptos relacionados con la rama de computación, como una de las competencias de la intensificación y otros conceptos sobre *Cloud Computing*. En concreto, se ha aprendido a utilizar la plataforma de *Amazon Web Services*. Desde un punto de vista computacional, se han adquirido conocimientos sobre algoritmos y técnicas de predicción que anteriormente no se habían estudiado durante la carrera.

Otra de las competencias adquiridas durante el proceso de trabajo ha sido el aprendizaje de nuevas técnicas de predicción y diseño de un sistema capaz de hacer pronósticos sobre los precios de las instancias de *AWS*. Los algoritmos de predicción, tanto el *Simple Exponential Smoothing* como el *Double Exponential Smoothing*, nos han sido útiles para hacer un pronóstico óptimo en los experimentos. Sin embargo, cabe mencionar que no son óptimos en todos los casos, tal y como hemos podido verificar en la evaluación de la eficiencia de los algoritmos.

De cara a los experimentos, podemos decir que, gracias a la selección del modelo por fase de entrenamiento de los datos, obtenemos una tendencia de precios estimados lo más cercana posible a la real. De esta forma, los resultados de nuestro pronóstico son muy parecidos a los de las observaciones del sistema, aunque con datos algo más variados. Esto nos beneficia de forma que, si nuestro pronóstico es óptimo, el precio de nuestra predicción con instancias *spot* no es mucho mayor que el del sistema.

Otro dato muy característico apreciado durante la elección de los casos de estudio ha sido que tanto para la distribución de *SUSE Linux* como para *Linux/Unix*, los valores de los precios cambia dinámicamente con el tiempo. Sin embargo, para el caso de *Windows*, apenas pueden notarse cambios durante el periodo de tiempo que tenemos en la base de datos, por eso no se ha contemplado en ninguno de los experimentos.

También hemos verificado, que el precio por hora de una instancia *spot* es menor que el de cualquier instancia reservada o bajo demanda.

## **Propuestas de mejora**

Como ideas de mejora, se proponen otras técnicas muy similares al suavizado exponencial, como la regresión, por ejemplo, aunque finalmente se decidió realizar el estudio aplicando técnicas de suavizado para así estudiar el concepto de predicción con series temporales.

También es muy interesante tomar como medida el *Triple Exponential Smoothing* o *Holt Winters*, que es otro algoritmo del tipo ponderado exponencial. Esta medida solo es apta para cuando tengamos ciclos y periódicos en nuestro conjunto de datos. Pero para nuestro caso, por días, por semanas o por meses no se ha dado ningún caso. Aun así, quizás al preprocesar los datos por años, ya que con nuestro conjunto de datos desde el mes de febrero a septiembre es insuficiente, se podrían presenciar con el paso de los años periodos repetitivos o ciclos para los precios.

Otra manera de trabajar con las predicciones de nuestros algoritmos es realizar un entrenamiento con varios conjuntos de periodos de tiempo diferentes (*hold-out*) y sobre esos datos generalizar el modelo. Con esta generalización, podemos obtener el mejor valor de *alpha* y de *beta* para aplicarlos sobre cualquier caso y poder predecir otras series temporales tomando como base ese modelo general. Para nuestros casos, como simplemente hemos considerado un periodo estático para trabajar y comparar los algoritmos sobre él, obtenemos el modelo adecuado para ese periodo, pero no para otras series temporales distintas. De esta manera, evitaríamos sobreajuste sobre los datos de nuestro conjunto de entrenamiento y podemos generalizar para nuevos datos.

Finalmente, gracias a este trabajo, se ha valorado lo importante que es la inteligencia artificial y la minería de datos para facilitar las formas de trabajar a las personas. Para nuestro caso, hemos creado un sistema capaz de predecir un precio concreto que necesita un usuario para contratar una instancia en la nube. Aun así, existen numerosos tipos de predicción que se pueden contrastar con el nuestro para mejorar el sistema. No cabe duda, que, gracias al avance de la tecnología, aparecerán nuevas técnicas y medidas de predicción con un porcentaje de acierto mayor y con menos sobreajuste para los datos.

# **ANEXOS**

## **ANEXO A. Tipos de servicios de *AWS***

***Servicios de computación***

Además de *Amazon* *EC2*, que se incluye dentro de este tipo de servicios existen otros servicios muy interesantes propios de los **servicios de computación** [32]:

* ***Amazon Elastic Container Service (ECS)***: este servicio permite organizar contenedores de tipo *Docker* en un clúster sin necesidad de invertir ningún coste en el gasto de infraestructura y así poder lanzar todo tipo de aplicaciones de manera sencilla y con alto nivel de escalabilidad.
* ***Amazon Elastic Container Registry (ECR)***: parecido al anterior, pero en este caso facilita almacenar, administrar y mejorar la implementación las imágenes de los contenedores *Docker* de forma fiable para el desarrollo de las aplicaciones.
* ***Amazon Lightsail***: ofrece máquinas virtuales, almacenamiento y transferencia de datos entre otras funciones para ayudar a los usuarios a configurar un servidor privado virtual.
* ***Amazon Batch***: permite ejecutar todo tipo de recursos informáticos por lotes para facilitar a los desarrolladores su trabajo atendiendo a las características computaciones específicas que necesiten, como las instancias virtuales que ofrece *Amazon* *EC2*.
* ***Amazon Elastic Beanstalk***: este servicio proporciona una serie de beneficios a los desarrolladores para facilitarles el diseño de aplicaciones web tales como *Java*, .*NET*, *Python* o *Node.js* entre otras. Este servicio interactúa con otros servicios de *AWS* como *Amazon* *EC2* o *Amazon S3*.
* ***Amazon Lambda***: gracias a este servicio, los usuarios podrán hacer ejecuciones de su código sin necesidad de hacer ningún gasto por los servidores, ya que *AWS Lamda* se encargará de hacerlo por ellos.
* ***AWS Auto Scaling***: el servicio de *Auto Scaling* de *AWS* facilita un nivel de escalabilidad eficiente por un bajo coste apoyándose en una interfaz sencilla que ayuda a establecer la monitorización adecuada para los recursos y aplicaciones. Proporciona un escalado automático de las instancias de *Amazon EC2*.

***Servicios de almacenamiento***

A continuación, vamos a ver cómo se lleva a cabo la actividad de los **servicios de almacenamiento**desde el punto de vista de la infraestructura que presentan para que las actividades orientadas al almacenamiento se procesen con éxito:

* ***Amazon Simple Storage Service (S3)***: este es un servicio de almacenamiento orientado a objetos de gran tamaño que está disponible para los usuarios, con él se pueden realizar operaciones como escritura, lectura o borrado para tamaños que ocupan desde un *byte* hasta cinco *terabytes* de espacio. Cada uno de estos objetos están almacenados en un repositorio y pueden ser accedidos mediante la clave única asignada por el desarrollador de este. El repositorio puede ser almacenado por una de las regiones disponibles que el usuario indique. Los objetos están definidos por un nombre especificado por el usuario el cual puede ser global y además se pueden otorgar derechos a otros usuarios haciéndolos públicos.

Hemos mencionado que permite realizar acciones de escritura, lectura o borrado, pero no permite realizar otro tipo de acciones como copiado, renombrado o mover objetos.

Entre algunas de las características que presenta este servicio podemos destacar las siguientes:

* + *Durabilidad*: todos los objetos almacenados persistirán con el tiempo sin sufrir ningún tipo de modificación o daño.
  + *Simplicidad*: gracias al uso de los diferentes medios de acceso que ofrece *AWS* los usuarios podrán acceder y utilizar este servicio de forma fácil y sencilla.
  + *Escalabilidad*: los usuarios pueden utilizar este servicio para almacenar y acceder a sus datos siempre que lo deseen
  + *Seguridad*: todos los datos almacenados podrán ser transferidos mediante técnicas de cifrado y por medio del protocolo *SSL* para garantizar la seguridad.
* ***Amazon Elastic Block Store (EBS)***: este segundo servicio de almacenamiento que trataremos ofrece volúmenes estructurados por bloques que podremos utilizar para el uso de las instancias de *Amazon EC2*. Cada uno de los volúmenes con los que trabajamos se duplica de forma automática dentro de la zona de disponibilidad para garantizar la durabilidad y permanencia de dicho volumen. Una instancia de *Amazon EC2* puede estar asociada a varios volúmenes, pero en viceversa no, un determinado volumen, no puede ser compartido por varias instancias. Este servicio es ideal para el almacenamiento de bases de datos o sistemas de archivos muy complejos.

De cara a las características que presenta podemos destacar las siguientes:

* + Unidad de disco: los usuarios pueden seleccionar entre un disco de estado sólido (*SSD*) o una unidad de disco duro (*HDD*) para mejorar el rendimiento de sus aplicaciones.
  + Cifrado: *Amazon EBS* también permite cifrar el contenido de los datos ya sean estáticos o transferibles.
  + Control de acceso: ofrece a los usuarios la posibilidad de ver quién está autorizado para acceder a los volúmenes de datos.
* ***Amazon Simple DB***: otro servicio de almacenamiento que permite almacenar y hacer consultas de los datos por medio de solicitudes de servicios web, además trabaja con bases de datos relacionales por lo que las funciones de almacenado y de consulta sobre estas bases de datos son totalmente compatibles.

***Servicios de mensajería***

Entre los **servicios de mensajería** que se ofrecen, consideraremos el siguiente:

* ***Amazon Simple Queue Service (SQS)***: este servicio trabaja con una cola de mensajes que integra. Las instancias de *Amazon EC2* pueden enviar y recibir mensajes de tipo *SQS* desde cualquier máquina local con acceso a Internet sin necesidad de instalar ningún tipo de *software*.

El procesamiento de un mensaje actúa de forma que cuando se recibe un mensaje, este queda bloqueado durante su procesamiento hasta que vuelva a estar desbloqueado en un tiempo de espera determinado. Para acceder a estos mensajes se pueden utilizar interfaces diseñadas para consulta de mensajes en las colas. Estas colas pueden ser compartidas por varios usuarios sin perder mensajes pudiendo transferir cualquier volumen de datos como los que ya hemos visto en los servicios de almacenamiento.

***Servicios de monitorización***

Como **servicio de monitorización** podemos destacar:

* ***Amazon CloudWatch***: es un servicio muy interesante y útil para los usuarios ya que permite observar y analizar a los estos las métricas para ayudar a optimar el rendimiento de las aplicaciones y el uso de los recursos. Además, dispone de una interfaz para poder ver gráficos y estadísticas de los recursos. También se puede monitorizar el comportamiento de la latencia de las consultas a los volúmenes *EBS*, el almacenamiento de las instancias de *Amazon EBS* e incluso los mensajes de *Amazon SQS* entre otras cosas más.

***Servicios de red***

Por último, entre **los servicios de red** más destacados, nos encontramos con:

* ***Amazon Virtual Private Cloud (VPC):*** este servicio proporciona un puente de conexión entre la infraestructura existente de una empresa u organización y la nube de *AWS*. La conexión se establece por medio de una red privada virtual hacia una serie de recursos computacionales que quedan aislados de la plataforma. Estos servicios de *VPC* tiene como características la gestión de *firewalls* o de seguridad.

Los usuarios pueden crear en su red virtual privada subredes, cambiar el rango de las direcciones IP tanto en IPv4 como IPv6 de forma que administra a su gusto todo lo relacionado con la configuración de su red. Incluso aprovecha multitud de capas de seguridad para controlar el acceso de las instancias de *Amazon EC2* en cada una de las subredes.

Existen otra multitud más de servicios ofrecidos por *AWS*, pero para nuestro caso, simplemente hemos mencionado aquellos que son particularmente necesarios a la hora de hablar de *Amazon EC2,* ya que todo nuestro trabajo ronda entorno a las instancias que este ofrece además de ser la cúspide de todos los servicios en el mercado.

## **ANEXO B. Especificaciones de las instancias *EC2***

Las especificaciones computacionales de las instancias dependen, no solo de la categoría a la que pertenezcan, sino también en función del tamaño de instancia. Presentamos pues una visión general de las especificaciones y las capacidades mínimas y máximas que pueden tomar según el tamaño de instancia [35].

***Uso general***

|  |  |  |
| --- | --- | --- |
| **Tipo de instancia** | **Tamaños de instancia** | **Especificaciones generales** |
| **T3** | *t3.nano, t3.micro, t3.small, t3.medium, t3.large, t3.xlarge, t3.2xlarge* | CPU virtual: 2-8  Créditos/hora CPU: 6-192  Memoria: 0.5-32 GiB Almacenamiento: Solo EBS |
| **T2** | *t2.nano, t2.micro, t2.small, t2.medium, t2.large, t2.xlarge, t2.2xlarge* | CPU virtual: 1-8  Créditos/hora CPU: 3-81  Memoria: 0.5-32 GiB Almacenamiento: Solo EBS |
| **M5** | *m5.large, m5.xlarge, m5.2xlarge, m5.4xlarge, m5.12xlarge, m5.24xlarge, m5d.large, m5d.xlarge, m5d.2xlarge, m5d.4xlarge, m5d.12xlarge, m5d.24xlarge* | CPU virtual: 2-96  Memoria: 8-384 GiB  Alm. instancia: Solo EBS y SSD NVMe  Ancho de banda EBS: 3500-14000 Mbps |
| **M4** | *m4.large, m4.xlarge, m4.2xlarge, m4.4xlarge, m4.10xlarge, m4.16xlarge* | CPU virtual: 2-64  Memoria: 8-256 GiB  Alm. SSD: Solo EBS  Ancho de banda: 450-10000 Mbps |

Tabla 20. Características instancias de uso general

***Optimización informática***

|  |  |  |
| --- | --- | --- |
| **Tipo de instancia** | **Tamaños de instancia** | **Especificaciones generales** |
| **C5** | *c5.large, c5.xlarge, c5.2xlarge, c5.4xlarge, c5.9xlarge, c5.18xlarge, c5d.large, c5d.xlarge, c5d.2xlarge, c5d.4xlarge, c5d.9xlarge, c5d.18xlarge* | CPU virtual: 2-72  Memoria: 4-144 GiB  Alm. instancia: Solo EBS, y SSD NVMe  Ancho de banda: 3500-14000 Mbps |
| **C4** | *c4.large, c4.xlarge, c4.2xlarge, c4.4xlarge, c4.8xlarge* | CPU virtual: 2-36  Memoria: 3.75-60 GiB  Almacenamiento: Solo EBS  Ancho de banda: 500-4000 Mbps |

Tabla 21. Características instancias optimizadas para informática

***Optimizadas para memoria***

|  |  |  |
| --- | --- | --- |
| **Tipo de instancia** | **Tamaños de instancia** | **Especificaciones generales** |
| **R5** | *c5.large, c5.xlarge, c5.2xlarge, c5.4xlarge, c5.9xlarge, c5.18xlarge, c5d.large, c5d.xlarge, c5d.2xlarge, c5d.4xlarge, c5d.9xlarge, c5d.18xlarge* | *CPU virtual: 2-96*  *Memoria: 16-768 GiB*  *Rendimiento de red: 10-25 Gb*  *Almacenamiento SSD: SSD, SSD NVMe* |
| **R4** | *c4.large, c4.xlarge, c4.2xlarge, c4.4xlarge, c4.8xlarge* | *CPU virtual: 2-64*  *Memoria: 15.25-488 GiB*  *Rendimiento de red: 10-25 Gb*  *Almacenamiento SSD: Solo EBS* |
| **X1e** | *x1e.xlarge, x1e.2xlarge, x1e.4xlarge, x1e.8xlarge, x1e.16xlarge, x1e32.xlarge* | *CPU virtual: 4-128*  *Memoria: 122-3940 GiB*  *Almacenamiento SSD: 1x120 – 2x1920*  *Ancho de banda EBS: 500 – 14000 Mbps* |
| **X1** | *x1.16xlarge, x1.32xlarge* | *CPU virtual: 64-128*  *Memoria: 976-1952 GiB*  *Almacenamiento SSD: 1x1920 – 2x1920*  *Ancho de banda: 7000-14000 Mbps* |
| **Instancias memoria elevada** | *u-6tb1.metal, u-9tb1.metal, u-12tb1.metal* | *Procesadores lógicos: 448*  *Memoria: 6-12 TiB*  *Rendimiento de red: 25 Gbps*  *Ancho de banda: 14 Gbps* |
| **Z1d** | *z1d.large, z1d.xlarge, z1d.2xlarge, z1d.3xlarge, z1d.6xlarge, z1d.12xlarge* | *CPU virtual: 2-48*  *Memoria: 16-384 GiB*  *Rendimiento red: 10-25 Gb*  *Almacenamiento SSD: SSD MVMe* |

Tabla 22. Características instancias optimizadas para memoria

***Informática acelerada***

|  |  |  |
| --- | --- | --- |
| **Tipo de instancia** | **Tamaños de instancia** | **Especificaciones generales** |
| **P3** | *p3.2xlarge, p3.8xlarge, p3.16xlarge* | *GPU: 1-8*  *CPU virtual: 8-64*  *Memoria: 61-488 GiB*  *Memoria GPU: 16-128 GiB*  *GPU punto a punto: NVLink* |
| **P2** | *p2.xlarge, p2.8xlarge, p2.16xlarge* | *GPU: 1-16*  *CPU virtual: 4-64*  *Memoria: 61-732 GiB*  *Memoria GPU: 12-192 GiB* |
| **G3** | *g3.4xlarge, g3.8xlarge, g3.16xlarge* | *GPU: 1-4*  *CPU virtual: 16-64*  *Memoria: 122-488 GiB*  *Memoria GPU: 8-32 GiB* |
| **F1** | *f1.2xlarge, f1.4xlarge, f1.16xlarge* | *FPGA: 1-8*  *CPU virtual: 8-64*  *Memoria: 122- 976 GiB*  *Almacenamiento SSD: 470-4x940 GB*  *Rendimiento redes: 10-25 Gb* |

Tabla 23. Características instancias para informática acelerada

***Optimizadas para almacenamiento***

|  |  |  |
| --- | --- | --- |
| **Tipo de instancia** | **Tamaños de instancia** | **Especificaciones generales** |
| **H1** | *h1.2xlarge, h1.4xlarge, h1.8xlarge, h1.16xlarge* | CPU virtual:8-64  Memoria: 32-256 GiB  Rendimiento redes: 10-25 Gb  Almacenamiento: 1x2000-8x2000HDD GB |
| **I3** | *i3.large, i3.xlarge, i3.2xlarge,i3.4xlarge, i3.8xlarge, i3.16xlarge, i3.metal* | CPU virtual: 2-72  Memoria: 15.25-512 GiB  Rendimiento redes: 10-25 Gb  Almacenamiento: SSD NVMe |
| **D2** | *d2.xlarge, d2.2xlarge, d2.4xlarge, d2.8xlarge* | CPU virtual: 4-36  Memoria: 30.5-244 GiB  Almacenamiento: 3x2000 . 24x2000HDD GB |

Tabla 24. Características instancias optimizadas para almacenamiento

## **ANEXO C. Regiones y zonas de disponibilidad**

En este anexo, podemos apreciar cuales son las regiones que existen junto con sus zonas de disponibilidad [44].

***América del Norte***

******

Figura 40. Localización de regiones en América del Norte [44]

En esta parte del continente contamos con seis regiones disponibles:

|  |  |  |
| --- | --- | --- |
| **Notación** | **Ubicación geográfica** | **Zonas de disponibilidad** |
| us-east-1 | Región de EEUU-Este (Norte de Virginia) | 6 |
| us-west-2 | Región de EEUU-Oeste (Oregón) | 3 |
| us-gov-west-1 | Región AWS GovCloud (EEUU-Oeste) | 3 |
| us-east-2 | Región de EEUU-Este (Ohio) | 3 |
| us-west-1 | Región de EEUU-Oeste (Norte de California) | 3 |
| c-central-1 | Región de Canadá (Central) | 2 |

Tabla 25. Regiones en América del Norte

***América del Sur***



Figura 41. Localización de regiones en América del Sur [44]

En América del Sur simplemente podemos encontrar una sola región con tres zonas de disponibilidad.

|  |  |  |
| --- | --- | --- |
| **Notación** | **Ubicación geográfica** | **Zonas de disponibilidad** |
| sa-east-1 | Región de América del Sur (Sao Paulo) | 3 |

Tabla 26. Regiones en América del Sur

***Europa, Oriente Medio y África***

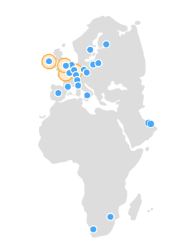


Figura 42. Localización de regiones en Europa, Oriente Medio y África [44]

Aquí podemos apreciar hasta cuatro regiones diferentes:

|  |  |  |
| --- | --- | --- |
| **Notación** | **Ubicación geográfica** | **Zonas de disponibilidad** |
| eu-west-1 | Región de Europa (Irlanda) | 3 |
| eu-west-2 | Región de Europa (Londres) | 3 |
| eu-central-1 | Región de Europa (Fráncfort) | 3 |
| eu-west-3 | Región de Europa (París) | 3 |

Tabla 27. Regiones en Europa, Oriente Medio y África

***Asia Pacífico***



Figura 43. Localización de regiones en Asia Pacífico [44]

Asia es el continente con más regiones y zonas de disponibilidad que podemos encontrar:

|  |  |  |
| --- | --- | --- |
| **Notación** | **Ubicación geográfica** | **Zonas de disponibilidad** |
| ap-southeast-1 | Región de Asia Pacífico (Singapur) | 3 |
| ap-northeast-1 | Región de Asia Pacífico (Tokio) | 4 |
| ap-northeast-3 | Región local de Asia Pacífico (Osaka) | 1 |
| ap-southeast-2 | Región de Asia Pacífico (Sidney) | 3 |
| ap-northeast-2 | Región de Asia Pacífico (Seúl) | 2 |
| ap-south-1 | Región de Asia Pacífico (Mumbai) | 2 |
| cn-north-1 | Región de China (Pekín) | 2 |

Tabla 28. Regiones en Asia Pacífico

## **ANEXO D. Contenido del CD**

**Código**

* *Obtener e Insertar Datos.ipynb*
* *Gráfica – Días.ipynb*
* *Grñafica – Semanas.ipynb*
* *Gráfica – Meses.ipynb*
* *Gini – Días.ipynb*
* *Gini – Semanas.ipynb*
* *Gini – Meses.ipynb*
* *Simple Exponential Smoothing.ipynb*
* *Simple Exponential Smoothing.ipynb*
* *Simple Exponential Smoothing.ipynb*
* *Double Exponential Smoothing.ipynb*
* *Double Exponential Smoothing.ipynb*
* *Double Expoenntial Smoothing.ipynb*

**Base de Datos**

* *BBDD.db*

**Memoria**

* *TFG - Luis Mendoza Montero*

# **BIBLIOGRAFÍA**

* [1] “Instancias *Spot* de *Amazon EC2*”, *Amazon Web Services*, 2018, Acceso: abril de 2018:

<https://aws.amazon.com/es/ec2/spot/>

* [2] *Alicia Rey*, “*Cloud Computing*: El servicio de almacenamiento en la nube”, Gestión de la información, Acceso: abril de 2018:

<http://www.huesca.es/_archivos/ficheros/bibliotecas_2651.pdf>

* [3] *Cloud Computing*, 2017 v.1, “Una guía de aproximación para el empresario”, Instituto Nacional de Ciberseguridad, Acceso: abril de 2018:

<https://www.incibe.es/sites/default/files/contenidos/guias/doc/guia-cloud-computing_0.pdf>

* [4] “*Regions and Availability Zones”*, 2018, Acceso: abril de 2018:

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html>

* [5] “Sistemas Operativos *AWS*”, 2018, Acceso: abril de 2018:

<https://docs.aws.amazon.com/es_es/inspector/latest/userguide/inspector_supported_os_regions.html>,

* [6] “Tipos de instancias de *Amazon EC2*”, 2018”, Acceso: abril de 2018:

<https://aws.amazon.com/es/ec2/instance-types/>

* [7] Jimena Sastre Álvarez, Laura P. Cogua Garzón, J.Manuel Cortés Rivera**, “**Transcripción del Coeficiente de *Gini*”, 2015, Acceso: abril de 2018:

<http://www.icesi.edu.co/cienfi/images/stories/pdf/glosario/coeficiente-gini.pdf>

* [8] Autores desconocidos, “Medición de la desigualdad”, Páginas 3-32, Acceso: abril de 2018:

<http://decon.edu.uy/~mito/nip/desigualdad.pdf>

* [9] Carlos Gradín y Coral Del Río, “Desigualdad, Polarización y Pobreza en la Distribución de la renta en Galicia”, Capítulos 1 y 3, Instituto de Estudios Económicos de Galicia - Fundación P. Barrié de la Maza - nº 11, A Coruña, 2001, Acceso: abril de 2018:

<http://decon.edu.uy/~mito/nip/desigualdad.pdf>

* [10] D.R. Instituto Tecnológico y de Estudios Superiores de Monterrey, “Métodos estáticos para la estimación de ingresos”, México, 2006, Acceso: abril de 2018:

<http://www.cca.org.mx/funcionarios/biblioteca/html/finanzas_publicas/documentos/3/m3_metodos.pdf>

* [11] Dra. Fernanda Villarreal, “Introducción a los modelos de pronóstico”, Universidad Nacional del Sur- Departamento de Matemática, 2016, Acceso: abril de 2018:

<http://www.matematica.uns.edu.ar/uma2016/material/Introduccion_a_los_Modelos_de_Pronosticos.pdf>

* [12] “Qué es *SCRUM*”, Acceso: abril de 2018:

<https://proyectosagiles.org/que-es-scrum/>

* [13] “*Python Official Page*”, Acceso: mayo de 2018:

<https://www.python.org/>

* [14] “Librería *boto Python*”, Acceso: mayo de 2018:

<https://pypi.org/project/boto/>

* [15] “*Project Jupyter*”, Acceso: mayo de 2018:

<http://jupyter.org/>

* [16] “Librería *Pandas*”, Acceso: noviembre de 2018:

<https://pandas.pydata.org/>

* [17] “*SQLite*”, Acceso: noviembre de 2018:

<https://www.sqlite.org/index.html>

* [18] “Información general acerca de los procesos de seguridad”, *Amazon Web Services*, Páginas 4 - 14, 2017, Acceso: septiembre y octubre de 2018:

<https://docs.aws.amazon.com/aws-technical-content/latest/aws-overview/aws-overview.pdf?icmpid=link_from_whitepapers_page>

* [19] Dan C. Marinescu, “*Cloud Computing: Theory and Practice*”, Capítulo 3, Morgan Kaufmann, 2013, Acceso: octubre de 2018:

<https://eclass.uoa.gr/modules/document/file.php/D416/CloudComputingTheoryAndPractice.pdf>

* [20] Kai Hwang – Gregory C.Fox – Jack Dongarra, “*Distributed and Cloud Computing*”, Capítulo 4, 2012, Acceso: octubre de 2018:

<https://eniac2017.files.wordpress.com/2017/03/distributed-and-cloud-computing.pdf>

* [21]” Tipos de *Cloud Computing*”, *Amazon Web Services*, 2018, Acceso: octubre de 2018:

<https://aws.amazon.com/es/types-of-cloud-computing/>

* [22] “5 Pasos previos para la migración en *Cloud*”, Acceso: noviembre de 2018:

<https://www.ekon.es/blog/2017/09/5-pasos-previos-para-la-migracion-al-cloud>

* [23] “¿Qué es la informática en la nube?”, *Amazon Web Services*, 2018, Acceso: octubre 2018:

<https://aws.amazon.com/es/what-is-cloud-computing/>

* [24] “*Cloud Computing* – Una perspectiva para Colombia”, abril de 2010, Acceso: octubre de 2018:

<https://cintel.co/wp-content/uploads/2013/05/16.clud_computing_Cloud-Computing-Mesa-sectorial-1.pdf>

* [25] Beka Kezherashvili, “Computación en la Nube”, Capítulos 2 3 y 4, Universidad de Almería, Acceso octubre de 2018:

<http://www.adminso.es/recursos/Proyectos/PFM/2011_12/PFM_cloud_beka.pdf>

* [26] “Almacenamiento en la nube – *Cloud Computing*”, 20 de enero de 2016, Acceso: octubre de 2018:

<http://tumundoenlanube.blogspot.com/2016/01/almacenamiento-basado-en-la-nubecloud_20.html>

* [27] Ms. Blanca Duarte de Báez, “NUEVAS TECNOLOGÍAS: NUEVOS PARADIGMAS *Cloud Computing* (Computación en la nube) “, julio de 2013, Acceso: octubre de 2018:
* [28] Instituto Nacional de Tecnologías de la Comunicación, “*Riesgos y amenazas en Cloud Computing*”, Capítulo 2, marzo de 2011, Acceso: octubre de 2018:

<https://www.incibe.es/extfrontinteco/img/File/intecocert/EstudiosInformes/cert_inf_riesgos_y_amenazas_en_cloud_computing.pdf>

* [29] “Informática en la nube con *Amazon Web Services*”, *Amazon Web Services*, 2018, Acceso: octubre 2018:

<https://aws.amazon.com/es/what-is-aws/>

* [30] “*Amazon Elastic Compute Cloud* – Guía del usuario de instancias de *Windows*”, *Amazon Web Services*, 2018, Acceso: octubre de 2018:

<https://docs.aws.amazon.com/es_es/AWSEC2/latest/WindowsGuide/ec2-wg.pdf#concepts>

* [31] “Infraestructura básica de *Amazon EC2* para *Windows*”, *Amazon Web Services* 2018, acceso: octubre y noviembre de 2018:

<https://docs.aws.amazon.com/es_es/AWSEC2/latest/WindowsGuide/EC2Win_Infrastructure.html>

* [32] “*Amazon EC2*”, *Amazon Web Services*, 2018, Acceso: octubre y noviembre de 2018:

<https://aws.amazon.com/es/ec2/>

* [33] “Documentación de *Elastic Compute Cloud*”, *Amazon Web Services*, 2018, Acceso: noviembre de 2018:

<https://docs.aws.amazon.com/ec2/index.html#lang/es_es>

* [34] “¿Qué es la *AWS Command Line Interface*?”, *Amazon Web Services*, 2018, Acceso: noviembre de 2018:

<https://docs.aws.amazon.com/es_es/cli/latest/userguide/cli-chap-welcome.html>

* [35] “Tipos de instancias de *Amazon EC2*”, *Amazon Web Services*, 2018, Acceso: noviembre de 2018:

<https://aws.amazon.com/es/ec2/instance-types/>

* [36] “Tipos de instancia “, *Amazon Web Services*, 2018, Acceso: noviembre de 2018:

<https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/instance-types.html>

* [37] Vivel Kumar Singh & Kaushik Dutta, “*Dynamic Price Prediction for Amazon Spot Instances*”, Department od information Systems National University of Singapore, 2015.
* [38] Markus Lumpe – Mohan Baruwal Chhetri – Quoc Bao Vo – Ryszard Kowalczyk, “On Estimation Minimum Bids for Amazon EC2 Spot Instances”, Faculty os Science, Engineering & Technology, Australia, 2017.
* [39] “Instancias y *AMI*”, *Amazon Web Services*, 2018, Acceso: noviembre de 2018:

<https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/ec2-instances-and-amis.html>

* [40] “Ciclo de vida de la instancia”, *Amazon Web Services*, 2018, Acceso: noviembre de 2018:

<https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/ec2-instance-lifecycle.html>

* [41] “*Amazon Elastic Compute Cloud – User Guide for Linux Instances*”, *Amazon Web Services*, 2018, Acceso: noviembre de 2018:

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-ug.pdf>

* [42] “Imágenes de máquina de *Amazon* (*AMI*)”, *Amazon Web Services,* 2018, Acceso: noviembre de 2018:

<https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/AMIs.html>

* [43] “*AWS* para principiantes: regiones y zonas de disponibilidad”, 2018, Acceso: noviembre de 2018:

<https://blog.rackspace.com/es/aws-para-principiantes-regiones-y-zonas-de-disponibilidad>

* [44] “Infraestructura global de *AWS*”, *Amazon Web Services*, 2018, Acceso: noviembre de 2018:

<https://aws.amazon.com/es/about-aws/global-infrastructure/>

* [45] L. Enrique Sucar, “Métodos de Inteligencia Artificial”, UPAEP, Acceso: noviembre de 2018:

<https://ccc.inaoep.mx/~esucar/Clases-MetIA/MetIA-16.pdf>

* [46] Marco Antonio Moreno, “¿Qué es el coeficiente de Gini?”, agosto de 2018, Acceso: noviembre de 2018:

<https://www.elblogsalmon.com/conceptos-de-economia/que-es-el-coeficiente-de-gini>

* [47] “*Single Exponential Smoothing*”, Acceso: noviembre de 2018:

<https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc431.htm>

* [48] “*Simple Exponential Smoothing*”, Acceso: noviembre de 2018:

<https://otexts.org/fpp2/ses.html>

* [49] “*Double Exponential Smoothing*”, Acceso: noviembre de 2018:

<https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc433.htm>

* [50] “*Trend Methods*”, Acceso: noviembre de 2018:

<https://otexts.org/fpp2/holt.html>

* [51] “Precios de *Amazon EC2*”, *Amazon Web Services*, 2018, Acceso: noviembre de 2018:

<https://aws.amazon.com/es/ec2/pricing/>

* [52] JMMArin, “Series Temporales”, Acceso: noviembre de 2018:

<http://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/EDescrip/tema7.pdf>