UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Pronóstico de la carga eléctrica total en plantas de ciclo combinado (gas-vapor) usando máquinas de vectores de soporte (SVM's).

TESIS

Que para obtener el título de

Ingeniero Industrial

P R E S E N T A

Abraham Alvarado Martínez

DIRECTOR DE TESIS

Dr. Wulfrano Gómez Gallardo

Ciudad Universitaria, Cd. Mx., 2019

Índice

Índice de Ilustraciones

Índice de Tablas

Agradecimientos

A mi madre, por ser el mayor ejemplo que tengo en la vida, por ser la persona que me ha enseñado a nunca rendirme, por acompañarme durante los momentos más sombríos que he atravesado, y por mostrarme amor y apoyo incondicional aun cuando no lo merecía.

A mi padre, por demostrarme que las dificultades en la vida sólo te hacen más fuerte y que el camino convencional no es el único que te lleva a la felicidad, por enseñarme que la familia es el mayor motor por el cual seguir luchando, y por brindarme su apoyo y amor durante toda la vida.

A mi hermano, quien me ha acompañado siempre, sin importar las circunstancias; por ser la persona que más aprecio en el mundo, por demostrarme que la bondad y la sencillez son valores esenciales, y porque sé que en todo momento puedo contar contigo. No puedo expresar lo orgulloso que estoy de ti y de lo alto que estoy seguro que llegarás.

A mis amigos, Jaime y Arturo, por demostrarme que el trabajo en equipo es capaz de lograr grandes cosas, por brindarme su apoyo y confianza cuando más lo necesitaba, y porque más que amigos los considero hermanos.

A mi asesor de tesis, quien soportó toda la indecisión que mostré al realizar este trabajo, por mostrarme su apoyo y enseñarme grandes cosas desde la primera clase que tomé con él, y por abrir mis ojos hacia el camino profesional que he decidido seguir.

Todo lo que soy, todo lo que he hecho, se lo debo a las personas que han estado a mi alrededor, sin las facilidades que me han brindado jamás me habría desarrollado de esta forma. Un individuo puede tener grandes cualidades y atributos, pero sin un entorno tan magnífico como el que me he encontrado, difícilmente logrará explotar por completo su potencial.

Antecedentes

Introducción

Con los cambios recientes en la industria eléctrica en México (Reforma energética de 2013), se abrieron nuevas áreas de oportunidad en una industria que antes era completamente controlada por dependencias gubernamentales. La llegada del sector privado a la generación y comercialización de energía eléctrica representó una modificación total de la estructura del mercado eléctrico. Así, se crearon nuevos reglamentos y se cimentaron nuevas bases para regir las operaciones cotidianas de los actores envueltos en la industria eléctrica. En este trabajo, nuestra atención se centra en el pronóstico de la generación total de energía en una planta de ciclo combinado. Dicha acción es solo un paso en la operación de las plantas generadoras de energía, pero representa un insumo esencial para que el sistema del mercado eléctrico mayorista funcione.

Para abordar el pronóstico de la carga eléctrica total de una planta de generación de ciclo combinado se propone el uso de algoritmos inteligentes, mejor conocidos como Machine Learning. Este tipo de plantas se componen de dos ciclos termodinámicos diferentes: ciclo Brayton y ciclo Rankine. El primero de ellos emplea turbinas de gas para generar la energía, mientras que el segundo ocupa turbinas de vapor. Dado que el ciclo Brayton emplea aire ambiental como fluido de trabajo, las propiedades de éste influyen en el rendimiento de la turbina de gas. Esto hace de la generación de energía en una planta de ciclo combinado una variable estocástica cuando la planta trabaja a su máxima capacidad, puesto que se puede condicionar su operación cuando labora a una capacidad menor, convirtiendo a la variable en una determinística.

El impacto de las plantas de ciclo combinado en México es amplío. En el año 2018, constituyeron el 36.5% de la capacidad instalada para la generación de energía eléctrica en el país, sumando un total de 83 diferentes centrales. Pero, en lo que hay que prestar mayor atención es que del total de energía eléctrica generada durante ese año, más de la mitad provino de este tipo de plantas (51%) (SENER, 2019). De ahí, la importancia de afinar su sistema de pronóstico.

En este trabajo se propone el uso del algoritmo de Máquinas de Vectores de Soporte (SVM, por sus siglas en inglés) para desarrollar un modelo de regresión que permita realizar estimaciones altamente exactas de la carga eléctrica total de una planta de ciclo combinado a partir de mediciones de las condiciones ambientales del lugar de operación de dicha planta. Los resultados que se obtuvieron fueron muy favorables. No sólo se obtuvo el modelo con una alta exactitud y generalización, sino que se desarrolló toda una metodología para poder llegar a él.

Este escrito consiste en siete partes distintas. La primera es la que actualmente se está leyendo. En los “Antecedentes” se pueden encontrar las razones por las que se inició esta investigación, el objetivo que se planteó cumplir, las hipótesis que se propusieron sobre la forma

de abordar este problema y las necesidades que se deseaban cumplir con la obtención del modelo de regresión.

El capítulo de “Marco Contextual” se compone de un resumen de cómo se constituye actualmente el mercado eléctrico en México. En él se pueden observar los principales cambios que existieron en la industria eléctrica del país con la aceptación de la Reforma energética, el papel que ahora juegan los generadores independientes dentro del mercado eléctrico, los organismos que regulan dicho mercado y la estructura del nuevo mercado eléctrico mayorista.

El capítulo del “Marco Teórico” es mucho más extenso que la anterior. Éste consiste en varios subcapítulos que tratan de temas muy distintos entre sí. El primer subcapítulo nos introduce al mundo de las plantas de ciclo combinado. En éste se puede ver a detalle cómo se conforma una de estas plantas y se dan características de su funcionamiento. El segundo subcapítulo “Fundamentos de Machine Learning” es mi sección favorita del marco teórico. Para toda persona interesada en Machine Learning, puede tomar este capítulo como referencia para comenzar su viaje en el mundo de la ciencia de datos. En este subcapítulo se abarcan temas desde la conceptualización de Machine Learning hasta aspectos metodológicos que se emplean en el análisis de datos. En mi opinión, este subcapítulo es muy bueno para comprender los conceptos claves de Machine Learning, pero quien quiera hacer análisis de este tipo debe conjuntarlo con práctica.

El siguiente subcapítulo del capítulo II, se enfoca en un algoritmo en especial: las SVM’s. Este subcapítulo comienza dando un entendimiento conceptual del algoritmo y termina con la formulación matemática del mismo. El nivel de profundidad que se alcanza en este subcapítulo es alto, pero hay ciertas partes en donde se simplifican los supuestos con el objetivo de no llenar este escrito con la demostración del algoritmo. El último subcapítulo de esta sección se centra en un estudio realizado anteriormente sobre el mismo problema que aborda este trabajo. De él se extrae posteriormente la base de datos empleada en este escrito y se usa como referencia para comparar los resultados obtenidos.

La “Presentación de Caso” es el capítulo más importante de todo el escrito. En él se inicia recordando los motivos de la investigación, así como los beneficios que conlleva. Luego se da paso a la descripción de la metodología seguida para obtener el modelo deseado. Se continua con la presentación de los resultados y su análisis, en donde se dan detalles muy precisos y justificados de cómo obtener un modelo SVM de alta exactitud y generalización para cualquier problema que se quiera afrontar, no solamente el de este trabajo. Y, una vez que se obtuvo un modelo que cumpliera con lo deseado, se dan recomendaciones futuras de cómo emplear dicho modelo y acciones que se podrían tomar para mejorar la metodología.

En las “Conclusiones” se hace el análisis de si el trabajo valió la pena, si se cumplió con el objetivo, si se comprobaron las hipótesis y, sobre todo, se responde a la pregunta de si la metodología propuesta es aplicable por las plantas de ciclo combinado en México.

Las secciones de “Referencias” y “Anexos” muestran los escritos que me ayudaron a forjar este trabajo y sustentarlo. En especial, en la sección de “Anexos” se tienen documentos que explican temas a los que no se les dedicó mucho espacio en el escrito y contiene el código completo que se empleó para obtener el modelo de regresión.

Sin más por el momento, les agradezco el interés en leer este trabajo y los dejo continuar para que conozcan la motivación detrás de él.

Planteamiento del Problema y Justificación

En gran parte de los mercados energéticos, incluyendo a México, es necesario que las plantas generadoras de electricidad envíen el pronóstico de su producción a los centros de transmisión de la red eléctrica. Esta acción tiene dos fines: el primero es que el centro cuente con la información necesaria para realizar un plan de acción que satisfaga la demanda esperada; y el segundo se relaciona con el proceso de pago, ya que el centro no recibirá más energía que la pronosticada considerando una cierta tolerancia hacia arriba y, a la vez, exigirá a la planta la energía prometida igualmente con una tolerancia, ahora hacia abajo. Esto significa que para que el ingreso por mega watt de una planta generadora de energía eléctrica pueda ser maximizado, el pronóstico de salida debe ser altamente exacto, dado que la energía producida de más se perderá por el hecho de que no puede ser almacenada y si se produce menos energía, la planta tendrá que extender su jornada laboral o pagar una multa.

En el caso específico de México, en el año 2013 se aprobó la Reforma Energética que modifica gran parte del esquema de adquisición y aprovechamiento energético del país. Esto dio pie a la creación de la Ley de la Industria Eléctrica en 2014, dentro de la cual uno de los principales cambios fue abrir el mercado de generación de energía eléctrica a productores independientes, sin que el Estado perdiera control sobre el sistema eléctrico nacional (transmisión y distribución). Además, toda la regulación del mercado de energía eléctrica fue concedida al Centro Nacional de Control de Energía (CENACE). De esta forma, los productores independientes están sujetos al marco comercial establecido por la CENACE, el cual que se divide de la siguiente forma: Subastas de largo plazo, Subastas de mediano plazo y Mercado de corto plazo. En cada uno de los esquemas de venta, las plantas de generación ofrecen la cantidad de energía que son capaces de producir y, mediante un modelo de programación lineal, el CENACE decide cuánta energía comprar a cada una.

La energía que el sistema eléctrico nacional adquiere está basada en sus pronósticos de demanda. Tanto las subastas a largo como a mediano plazo cubren la energía base que el país necesita de manera cotidiana. Sin embargo, conforme pasa el tiempo y se adquiere más información, el pronóstico de la demanda de energía eléctrica se modifica respecto al inicial. Por tanto, entra en juego el mercado de corto plazo que se encarga de empatar la demanda de energía eléctrica con su suministro, en caso de que el pronóstico inicial haya sido menor a la demanda

requerida. Esta acción es de extrema importancia dado que la electricidad es un recurso indispensable para prácticamente todas las actividades productivas.

La segunda problemática que atiende este trabajo es proveer de un modelo de pronóstico lo suficientemente confiable para la generación de energía de sistemas termodinámicos extremadamente complejos. El modelado preciso para plantas de generación eléctrica a partir de leyes termodinámicas requiere del uso de muchas ecuaciones no lineales que a veces son imposibles de resolver. Este tipo de análisis demanda un alto tiempo y poder de cómputo y sufre de pérdida de exactitud con el paso del tiempo, dado que no considera el envejecimiento de la maquinaria.

En el caso de las plantas de generación de ciclo combinado, uno de sus insumos es aire ambiental (ciclo Brayton). Este tipo de plantas tienen mecanismos reguladores para su generación de energía, sin embargo, cuando se ocupa cerca de o a su máxima capacidad, las propiedades del aire que emplea son factores determinantes para la carga eléctrica que será capaz de suministrar. Dado que la cantidad de aire que se consume es enorme como para darle un tratamiento previo (el cual resultaría bastante costoso), la generación de energía se ve impactada directamente por las condiciones ambientales.

Objetivo

Desarrollar una metodología centrada en la obtención de un modelo de pronóstico capaz de realizar estimaciones altamente exactas de la carga eléctrica total de una planta de generación de ciclo combinado. Esto mediante el uso del algoritmo inteligente conocido como Máquinas de Vectores de Soporte (SVM's) y la medición de las condiciones ambientales del lugar de operación, de modo que la planta pueda ofrecer, a la red de distribución eléctrica nacional, la cantidad de energía que verdaderamente es capaz de producir en determinado momento y, por tanto, disminuir sus pérdidas.

Hipótesis

H1: Las características del algoritmo de Máquinas de Vectores de Soporte lo hacen capaz de generar un modelo de pronóstico de la carga eléctrica total de una planta de generación altamente exacto, de emplear menores recursos computacionales comparado con su modelado a partir de ecuaciones termodinámicas y de adaptarse al transcurso del tiempo.

H2: La producción energética total de una planta de ciclo combinado es dependiente de ciertas condiciones ambientales del lugar de operación[1].

[1] Dichas condiciones tienen la ventaja de que su medición es relativamente sencilla.

I. Marco Contextual

"La electricidad es un bien preferente que no admite sustitutos, que presenta una demanda casi perfectamente inelástica al precio, que no es almacenable a gran escala (lo que prohíbe crear provisiones a mediano y largo plazo), que es prácticamente imposible (técnica y económicamente) satisfacer la demanda eléctrica con una sola tecnología y que, además, incide en la mayoría de los procesos productivos de la economía" (Payan, Díaz, & Cossío, 2016).

1.1 Antecedentes

El 27 de septiembre de 1960 el presidente Adolfo López Mateos anunció la nacionalización de la industria eléctrica[2]. Esta acción implicaba que la nación era la única responsable de generar, conducir, transformar, distribuir y abastecer energía eléctrica como servicio público. A partir de ese momento se asignaron dos organismos descentralizados con el fin de proveer electricidad al país: la Comisión Federal de Electricidad (CFE) y Luz y Fuerza del Centro (LFC), ambos regidas por la Ley de Servicio Público de Energía Eléctrica (LSPEE).

Tiempo después, en el año de 1992, se generó una primera reforma al sector eléctrico. En ésta, constitutivamente se permitía la participación privada en la forma de Productores Independientes de Energía (PIE) y Pequeños Productores (PP). Las funciones que ambos tenían se limitaban a vender energía a CFE, exportarla y autoabastecerse. Por lo que no fungían como competidores para las empresas paraestatales.

Sin embargo, a medida que la población mexicana evolucionó (cambiaron sus necesidades tecnológicas), diversos factores comenzaron a poner en duda la viabilidad del sector para satisfacer la demanda eléctrica. A continuación, se presentarán dichos factores.

- Demanda creciente de energía: Según las últimas proyecciones del Consejo Nacional de Población (CONAPO, 2018), México tendrá una población de 139.3 millones de personas en el año 2032, cuyo consumo bruto eléctrico se pronostica de 492,165 GWh (SENER, 2018). Esto representa un incremento de alrededor del 59% comparado con el consumo eléctrico del año 2017.
- Alta dependencia de insumos fósiles: En 2013, la matriz eléctrica mexicana estaba conformada en un 81.7% por tecnologías que usan combustibles fósiles para la generación eléctrica. En ese mismo año, esta actividad fue el segundo sector que más emisiones de gases de efecto invernadero (GEI) liberó a la atmósfera (USAID, 2013).
- Redirección del gasto: Puesto que CFE era responsable de administrar todas las actividades dentro de la cadena de valor, el gasto que ejercía era considerable. En este caso, los gastos

[2] Acción que quedó establecida en el artículo 27 constitucional.

por generación de energía equivalían a prácticamente la mitad (45.3%) de los ingresos recaudados. Por lo que se propuso que una parte de la cadena de valor dejara de ser parte exclusiva del Estado, y, así, CFE pudiera liberar recursos y emplearlos en destinos más productivos (CIEP, 2017).

Debido a estos factores, se requería de un sector "abierto a la competencia, moderno, regulado, competitivo, seguro y sustentable" (CIEP, 2017). Y fue así como, el 11 de agosto de 2014, se abrogó la LSPEE y entraron en vigor la Ley de la Industria Eléctrica (LIE) y la Ley de la Comisión Federal de Electricidad, las cuales sentaron las bases de lo que hoy es el Mercado Eléctrico Mayorista (MEM).

1.2 Generalidades de la LIE

Acorde al glosario de términos eléctricos de la Secretaría de Energía (SENER), las cuatro actividades principales de la industria eléctrica son (SENER, 2018):

1. *Generación: Producción de energía eléctrica por el consumo de alguna otra forma de energía (viento, sol, gas, vapor, calor geotérmico, movimiento del agua, etc.).*
2. *Transmisión: Es la conducción de energía eléctrica, desde las plantas de generación o puntos de interconexión (punto donde se entrega energía entre dos entidades), hasta los puntos de entrega para su distribución.*
3. *Distribución: Es la conducción de electricidad, desde el/los puntos de entrega de la transmisión, hasta los puntos de suministro a los usuarios.*
4. *Comercialización: Es el conjunto de actos y trabajos para proporcionar energía eléctrica a cada usuario.*

Con la LIE, dos de estas actividades se abrieron a la participación privada: la generación[3] y la comercialización[4]. El resto siguen siendo catalogadas como áreas estratégicas y permanecen como actividades exclusivas del Estado[5]. De forma que, con la reforma energética, la industria eléctrica se dividió en sus cuatro principales actividades, de modo que se realicen de forma independiente entre sí[6].

En este sentido, se modificó totalmente la estructura de la industria eléctrica. El esquema de ésta quedó definido por las siguientes categorías:

1. Generadores.
2. Operador, regulador y productos del mercado.
3. Comercializadores.

[3] Artículos 17 a 25 de la LIE.
[4] Artículos 45 a 58 de la LIE.
[5] Segundo párrafo, artículo 2 de la LIE.
[6] Artículo 8 de la LIE.

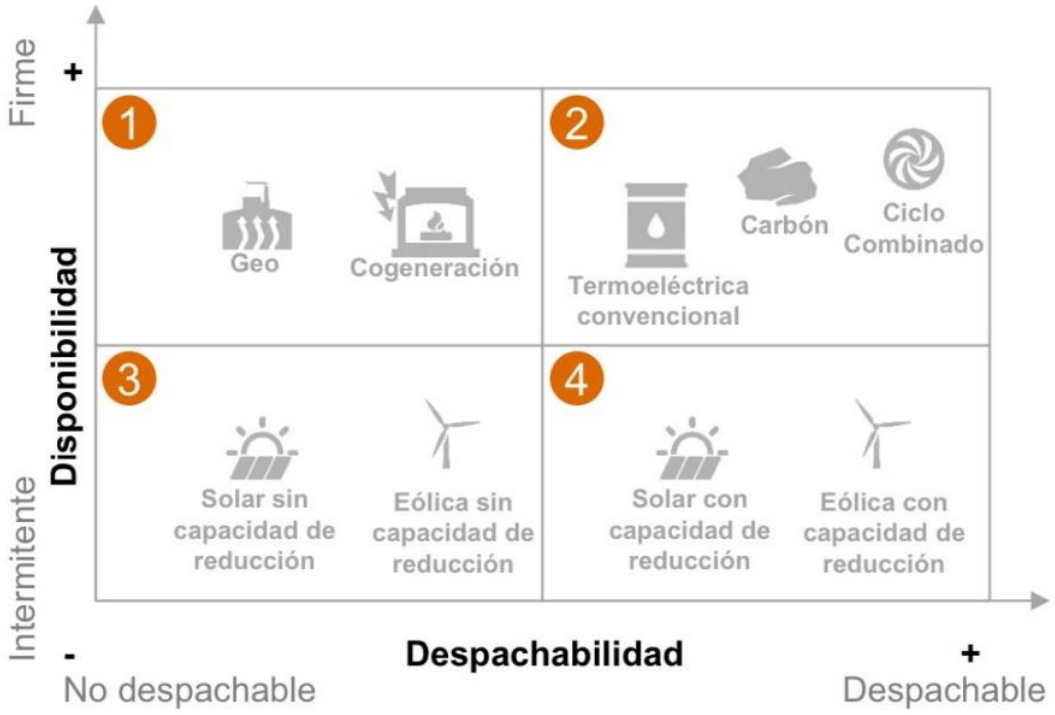
4. Usuarios.

1.3 Generadores

El artículo 17 de la LIE dicta que cualquier central eléctrica con capacidad mayor o igual a 0.5 MW puede ser considerada generadora dentro del nuevo mercado, sujeta a diversas aprobaciones por parte de la Comisión Reguladora de Energía (CRE). Estos podrán comerciar, no sólo la energía eléctrica que produzcan, sino también los productos asociados que se les entreguen a consecuencia de su generación.

Debido a la gran variedad que existe de centrales eléctricas, éstas deben registrarse con un estatus según su grado (firme o intermitente) y su despachabilidad. Esa categorización se puede observar en la siguiente gráfica (Ilustración 1):

Ilustración 1: Estatus para registro de Centrales Eléctricas



Gráfica recuperada de (PwC, 2015).

A continuación, se da una explicación más detallada de cada uno de los estatus (PwC, 2015):

1. *Firme no despachable: Fuente firme que no tiene la capacidad de controlar su nivel de producción en tiempo real (ciertas instalaciones de cogeneración o geotérmica). Dichas unidades no están exentas de seguir instrucciones del CENACE cuando se requiere por confiabilidad; sin embargo, en el despacho económico se asumirá que su producción está fija en el último valor medido o en el valor pronosticado.*

2. *Firme despachable: Fuente que tiene la capacidad de seguir instrucciones de despacho en tiempo real hasta su capacidad instalada (Ciclo Combinado, Termoeléctrica convencional o Carboeléctrica).*
3. *Intermitente no despachable: Fuente intermitente que no tiene la capacidad de controlar su nivel de producción en tiempo real (eólica o solar sin la capacidad de reducir generación mediante instrucciones automáticas de despacho). Al igual que la firme no despachable, dichas unidades no están exentas de seguir instrucciones del CENACE cuando se requiere por confiabilidad; sin embargo, en el despacho económico se asumirá que su producción está fija en el último valor medido o en el valor pronosticado.*
4. *Fuente que tiene la capacidad de seguir instrucciones en tiempo real hasta una capacidad intermitente (eólica o solar con la capacidad de reducir generación mediante instrucciones automáticas de despacho).*

Ahora bien, cada generador está obligado a ofertar su capacidad total a la Unidad de Vigilancia del Mercado. "Los representantes de las Unidades de Central Eléctrica ofrecerán la totalidad de las capacidades disponibles para producir energía eléctrica y Servicios Conexos en dichas unidades, a menos que cuenten con una exención de la Unidad de Vigilancia del Mercado o no se encuentren disponibles, total o parcialmente, debido a una salida programada por mantenimiento, salida forzosa, reducción de potencia u otro motivo aprobado por el CENACE" (SENER, 2018).

1.4 Productos del Mercado y Organismos Reguladores

Además de la energía, otros productos pueden ser negociados en el mercado para permitir el cumplimiento de las obligaciones de los participantes y el adecuado funcionamiento del sistema eléctrico. A éstos se les conoce como productos asociados (Tabla 1).

El primer producto asociado es la Potencia. La Potencia es la obligación que el generador tiene de asegurar la disponibilidad de electricidad para el futuro y, por tanto, tiene valor comercial (CIEP, 2017). "Este producto tiene como objetivo asegurar la disponibilidad de la producción física y ofrecer la energía correspondiente en el mercado en tiempo real y el mercado del día en adelanto" (PwC, 2015).

El segundo producto se conoce como Certificados de Energía Limpia (CELS). Un CEL es un título que acredita la producción de energía limpia (fuentes cuyas emisiones y/o residuos no rebasen 100 kg de CO_2 equivalente). Los generadores que produzcan energía limpia recibirán un CEL por cada MWh generado. Estos tendrán un valor comercial, pues a los suministradores, usuarios calificados, usuarios por abasto aislado y a los CIL que no generen energía limpia se les exigirá que un porcentaje de su consumo provenga de fuentes limpias. Para respaldar esto, deben adquirir CELS equivalentes al monto que exija la SENER (CIEP, 2017).

El tercer producto es la demanda controlable, que se refiere a la demanda que los suministradores y/o usuarios calificados ofrecen reducir en un momento determinado, por orden del CENACE. Por otra parte, el siguiente producto se conoce como Derechos Financieros de Transmisión (DFT). Estos otorgan el derecho y la obligación de cobrar o pagar la diferencia entre los precios marginales locales (PML) de inyección y retiro. En otras palabras, son coberturas de precio en diferentes nodos del sistema. El titular del derecho paga o cobra la diferencia del precio que en verdad resulte de la venta de energía.

El último producto son los servicios conexos. Son servicios que garantizan la calidad, continuidad y seguridad del Sistema Eléctrico Nacional (SEN). Dentro de estos se encuentran los arranques de emergencia; regulación del voltaje, frecuencia y potencias; reservas operativas; entre otros.

Tabla 1: Productos ofertados en el mercado

Productos ofertados en Mercado

Producto	*¿Qué es?*
Potencia	El requerimiento de Potencia es una herramienta de **Confiabilidad** que tiene como objetivo cumplir **requisitos mínimos de planificación de reservas.**
Certificados de Energía Limpia	Título emitido por la CRE que acredita la producción de un monto determinado de energía eléctrica a partir de Energías Limpias y que sirve para cumplir los requisitos asociados al consumo de los Centros de Carga
Servicios conexos	Los Servicios Conexos del MEM buscan garantizar la confiabilidad del Sistema Eléctrico Nacional y **pueden o no estar incluidos en el mercado.** Representan una obligación para los participantes del mercado.
Derechos Financieros de Transmisión	Son títulos de crédito para **pagos financieros**, no otorgan derecho físico a usar la red. Derecho a cobrar la **diferencia del valor de los Componentes de Congestión Marginal entre un nodo origen y uno destino.**

Tabla recuperada de (PwC, 2015).

Una vez explicados los productos que se pueden comercializar en el nuevo esquema de la industria eléctrica, se proseguirá con los organismos que se encargan de regularlos a ellos y al mercado en general. El primer organismo es el Centro Nacional de Control de Energía (CENACE). Cuando se abrieron los sectores de generación y comercialización de la industria eléctrica, el mercado eléctrico se transformó radicalmente: de tener precios semifijos, cuya variación simplemente dependía de las modificaciones en el precio de los insumos; a un mercado libre que se rige por las leyes de la oferta y la demanda. No obstante, la electricidad es un insumo prioritario, y, por tanto, el Estado se debe cerciorar que el precio de la energía y servicios eléctricos no sucumban ante la volatilidad que los mercados contemporáneos experimentan. Por

tanto, de acuerdo con el artículo 94 de la LIE, la CENACE se encarga de calcular el precio de los productos y servicios del Mercado Eléctrico Mayorista (MEM). Esto a partir de la información de mercado que el organismo pueda recabar. De esta forma, la CENACE se encarga de la operación del MEM.

La CENACE cumple varias funciones como procurar la seguridad de despacho, la calidad y la continuidad del SEN, así como garantizar el acceso abierto y no discriminatorio a la Red Nacional de Transmisión (RTN) y a las Redes Generales de Distribución (RGD). Sin embargo, sus funciones principales se centran, como se mencionó en el párrafo anterior, en la regulación y operación del MEM. Dentro de sus principales funciones dentro del MEM se encuentran:

- Asignar la energía generada por las centrales eléctricas a centros de carga en donde exista demanda.
- Recibir ofertas (de energía y productos asociados) de los generadores y calcular los precios de equilibrio.
- Llevar a cabo subastas para celebrar contratos de cobertura eléctrica entre los generadores y los centros de carga.
- Coordinar el programa de mantenimiento de las centrales eléctricas de los generadores (dado que debe mantener la continuidad y confianza del sistema eléctrico).

Por otro lado, el segundo organismo que se encarga de la regulación del MEM es la Comisión Reguladora de Energía (CRE). Una de sus principales funciones es otorgar los permisos necesarios para participar en el MEM[7]. Dentro de esos permisos se encuentran los siguientes:

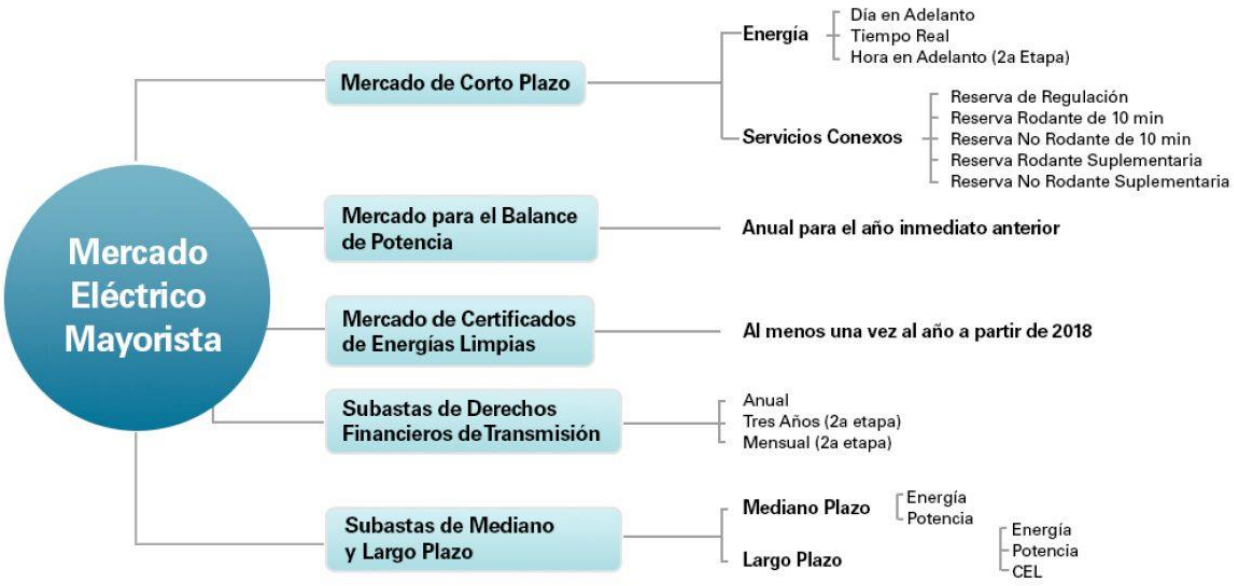
- Permiso para generar energía eléctrica.
- Permiso para suministrar energía.
- Emite (o niega) permisos de importación de electricidad.
- Define los modelos contractuales.
- Regula las tarifas de transmisión y distribución.

1.5 Mercado Eléctrico Mayorista y Contratos de Cobertura Eléctrica

El MEM y los Contratos de Cobertura Eléctrica (CCE) son los esquemas en que se comercializa la energía eléctrica desde que la LIE entró en vigor. Ambos se crearon para garantizar la confiabilidad y la continuidad del SEN a largo, mediano y corto plazo. En términos generales la estructura de comercialización es la descrita por la siguiente imagen (Ilustración 2):

[7] Artículo 130 de la LIE.

Ilustración 2: Estructura del Mercado Eléctrico Mayorista



Mapa recuperado de (CENACE, 2017).

Los contratos de cobertura eléctrica se dan en tres modalidades: Libre decisión, Subastas a largo plazo y Subastas a mediano plazo. Todas estas tienen el objetivo de "reducir o eliminar la exposición a la fluctuación de los precios de energía y productos derivados en el corto plazo para los consumidores" (CENACE). En la cobertura de libre decisión los participantes del mercado tienen la libertad de fijar pagos, términos y condiciones que les convengan. Por otra parte, las subastas a largo y mediano plazo se rigen de diferente manera. En ellas gana el participante que ofrezca la energía generada lo más barato posible, con el fin de que la electricidad entregada al consumidor sea la de menor valor. Las diferencias que se encuentran entre estas subastas es el horizonte que contemplan. La Subasta de largo plazo es de quince años de cobertura para energía y Potencia, mientras que es de veinte años para CELS. Por su parte, la Subasta de mediano plazo es de únicamente tres años de cobertura para energía y Potencia.

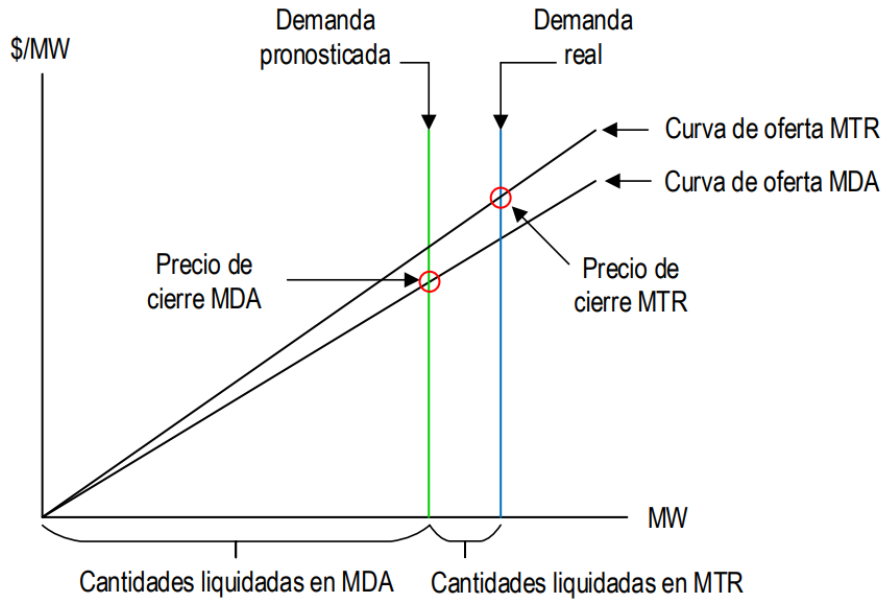
Por otra parte, el mercado de energía Spot (corto plazo) se rige por reglas totalmente distintas. La característica especial de este mercado es que la oferta está dada por los costos variables. Esto significa que aquella central que tenga los menores costos variables será elegida primero para despachar su energía, y así sucesivamente hasta satisfacer la demanda. Sin embargo, el precio que se les pagará será el de la última central utilizada para cumplir con la demanda. Así, los generadores tienen incentivos para reducir sus costos variables.

También se encuentra el mercado de balance de potencia. Este es un mercado auxiliar en el que aquellas entidades que no cuenten con el mínimo de Potencia requerida por la CRE, encuentren generadores con exceso de ella y no la hayan comprometido antes en contratos de cobertura eléctrica.

Por último, tocaremos el mercado de corto plazo con un detalle mayor. Éste se divide en tres modalidades: Mercado del Día en Adelanto (MDA), Mercado en Tiempo Real (MTR) y Mercado de la Hora en Adelanto (MHA). Cabe mencionar que éste último y el MDA tienen una estructura similar, por lo que simplemente se abordará el MDA. La existencia de este mercado está ligada a la imprecisión natural que se tiene al momento que la CENACE realiza pronósticos de demanda de energía eléctrica a mediano y largo plazo. Como es de esperarse, a medida que transcurre el tiempo, se tiene una mayor información y, por tanto, se modifica la demanda esperada de consumo eléctrico en las diferentes zonas de potencia del país. En caso de que la nueva demanda sea menor a la pronosticada en un inicio, se busca redirigir la energía o simplemente almacenarla por un corto periodo de tiempo. No obstante, si la demanda resulta mayor que la esperada, se recurre al mercado de corto plazo (Ilustración 3).

En el mercado de corto plazo se comercializan dos tipos de productos: energía y servicios conexos. La planificación de este mercado es que "diariamente, el CENACE evaluará los requerimientos de energía y Servicios Conexos para los siguientes 7 días con la finalidad de identificar a las Unidades de Central Eléctrica que serán requeridas para la eficiencia del mercado y la confiablidad del sistema" (CENACE, 2016). Para garantizar estas situaciones "el CENACE recibirá Ofertas de Compra y Ofertas de Venta de energía y Servicios Conexos correspondientes al Mercado del Día en Adelanto durante el periodo de recepción de Ofertas el cual estará disponible 7 días previos al Día de Operación y hasta las 10:00 horas del día anterior al Día de Operación" (CENACE, 2016).

Ilustración 3: Precios y demandas en el mercado de corto plazo

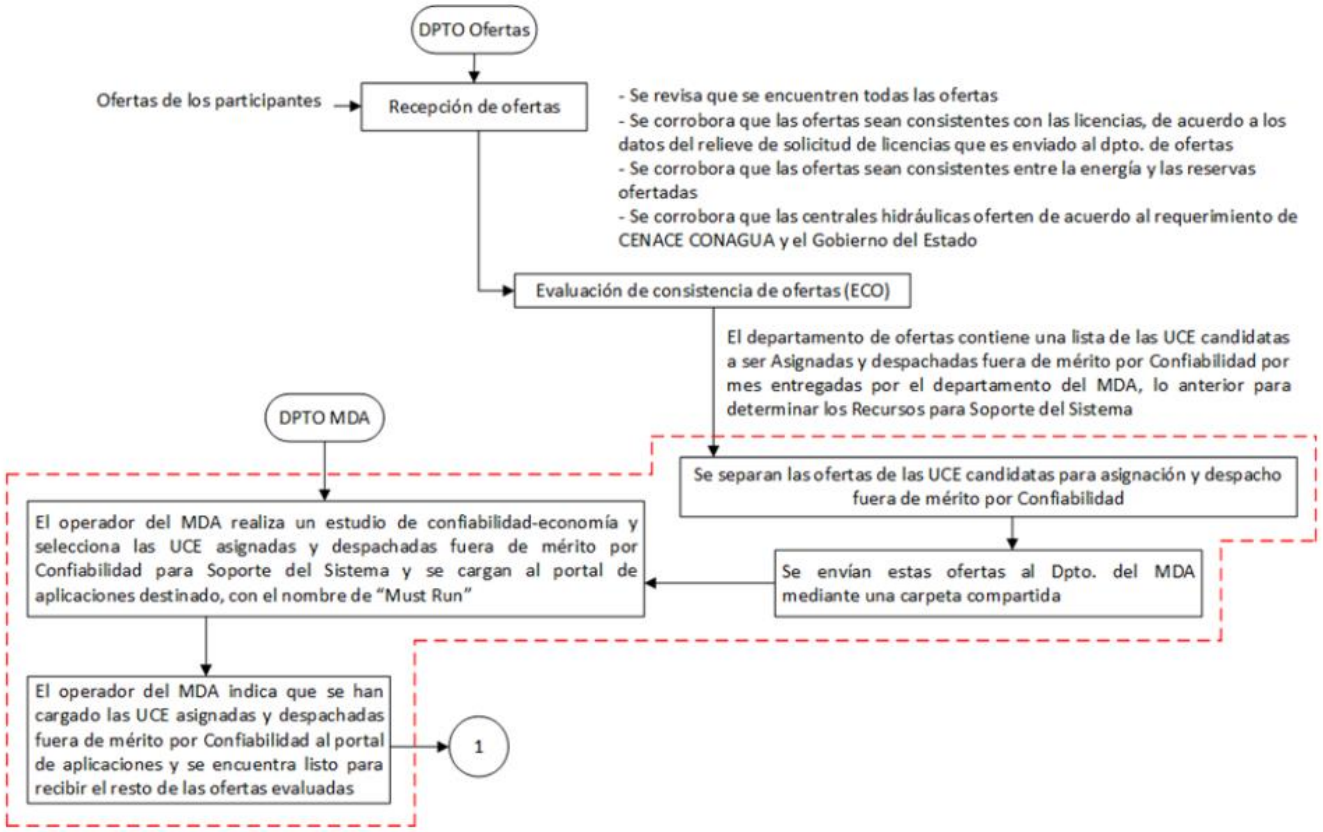


Gráfica recuperada de (CRE, 2018).

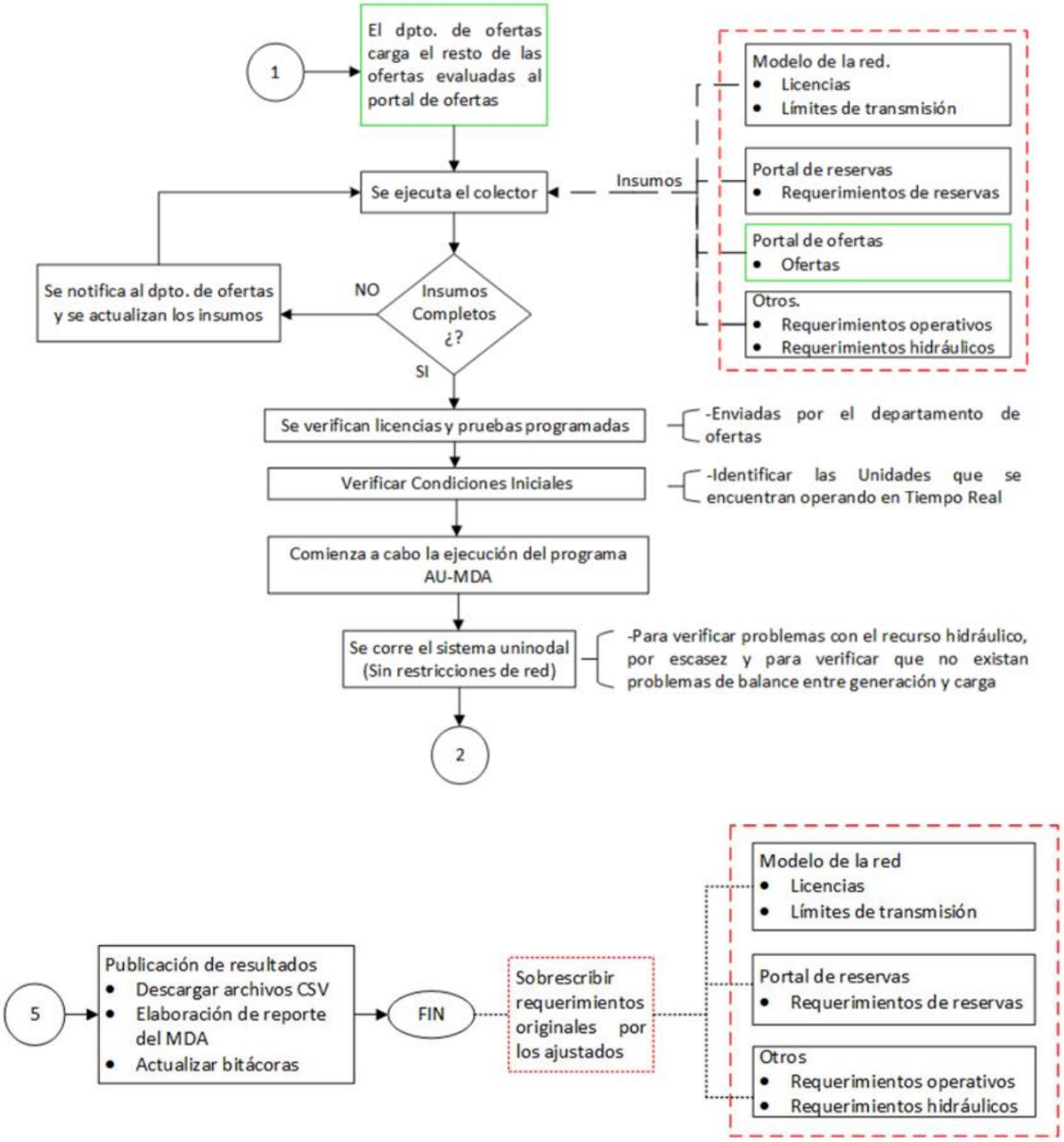
Por otra parte, “el objetivo del Mercado de Tiempo Real es ajustar las diferencias entre las transacciones realizadas en el Mercado del Día en Adelanto y las condiciones del mercado en tiempo real. Sólo se permitirá ajustar las Ofertas de Venta por diferencias que resulten de cambios en las capacidades disponibles de generación y de las capacidades para el suministro de los Servicios Conexos” (CENACE, 2016).

En el siguiente diagrama se muestra de forma gráfica y a grandes rasgos el funcionamiento día a día del MDA (Ilustración 4), y en los anexos 1 y 2 se describe de forma más detallada su operación según el manual expedido por la CENACE.

Ilustración 4: Diagrama de procesos para el Mercado de Energía de Corto Plazo[8]



[8] Se excluyeron partes del proceso debido a que describían casos especiales en la planeación del mercado, se mantuvieron las fases más generales en el funcionamiento.



Fuente: Elaboración del MIM, 2017.

Diagrama recuperado de (CRE, 2018).

Una vez que se revisó el impacto de la reforma energética en la estructura del mercado eléctrico del país, se puede notar que los cambios han sido profundos. La existencia de competencia en el mercado eléctrico conlleva grandes beneficios, pero la electricidad es un suministro base para el funcionamiento de la sociedad, por lo que se espera que los organismos regulatorios no caigan en irregularidades que desemboquen en un daño mayúsculo para la nación.

II. Marco Teórico

2.1 Generalidades sobre Plantas de Ciclo Combinado

Con el objetivo de obtener eficiencias térmicas más altas, en la época moderna se han formulado modificaciones innovadoras en las centrales eléctricas convencionales. La modificación más popular es el ciclo combinado de gas y vapor o simplemente: Ciclo Combinado (CC). En éste un ciclo de potencia de gas suministra energía a un ciclo de potencia de vapor. Así mismo, dentro de los ciclos combinados de gas y vapor, aquél que mayor uso tiene es el que combina el ciclo de turbina de gas (Brayton) y el ciclo de turbina de vapor (Rankine). Esta configuración ha demostrado tener una eficiencia térmica más alta que cualquiera de los ciclos ejecutados individualmente.

Ahora, con el propósito de ofrecer un mayor entendimiento sobre el funcionamiento del CC, se abordarán el ciclo Brayton y el ciclo Rankine de forma individual, para luego cerrar con su aplicación en conjunto y sus beneficios. Se comenzará con el ciclo Brayton.

El ciclo Brayton fue propuesto por George Brayton alrededor de 1870. Aunque originalmente se usó para un motor reciprocante que quemaba aceite, actualmente se utiliza en turbinas de gas donde los procesos de compresión y expansión ocurren en una misma maquinaria rotatoria. Las turbinas de gas operan generalmente en un ciclo abierto, en el que se introduce aire fresco en condiciones ambiente a un compresor. Dentro de éste la temperatura y presión del aire se elevan. El camino del aire a alta presión continúa hacia una cámara de combustión, en ésta se quema combustible a presión constante. Los gases que resultan de este proceso ahora están a alta temperatura y a alta presión. Estos gases se dirigen a la turbina, donde se expanden hasta regresar a la presión atmosférica, lo cual genera potencia. Finalmente, los gases de escape que salen de la turbina se expulsan del ciclo (no se recirculan), por lo que el ciclo se clasifica como uno abierto (Çengel & Boles, 2015). La ilustración 5 muestra una representación gráfica del ciclo. Sin embargo, por regulaciones, los gases no son expulsados directamente a la atmósfera, sino que se les debe dar un tratamiento para procurar que contaminen sustancialmente menos el ambiente que si se dejaran libres los gases. Se debe recordar que en 2013 el uso de combustibles fósiles para la generación de energía fue el segundo contaminante de GEI (USAID, 2013).

Por otra parte, el circuito de aire puede modificarse de abierto a uno cerrado. En este caso, los procesos de compresión y expansión se mantienen iguales, pero el proceso de combustión se sustituye por uno de adición de calor a presión constante desde una fuente externa. Así mismo, el proceso de escape se reemplaza por uno de rechazo de calor a presión constante hacia el aire ambiental. Una representación gráfica de este proceso se puede observar en la ilustración 6.

Ilustración 5: Ciclo Brayton circuito abierto

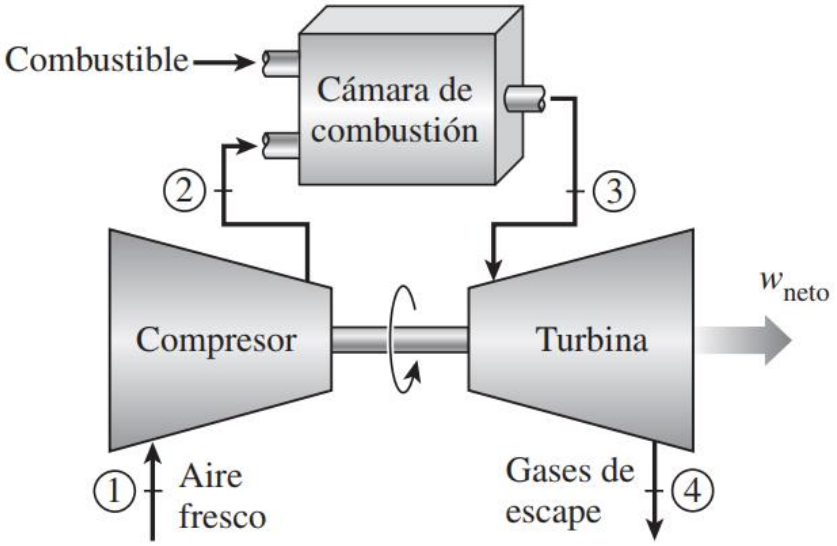


Diagrama recuperado (Çengel & Boles, 2015).

Ilustración 6: Ciclo Brayton circuito cerrado

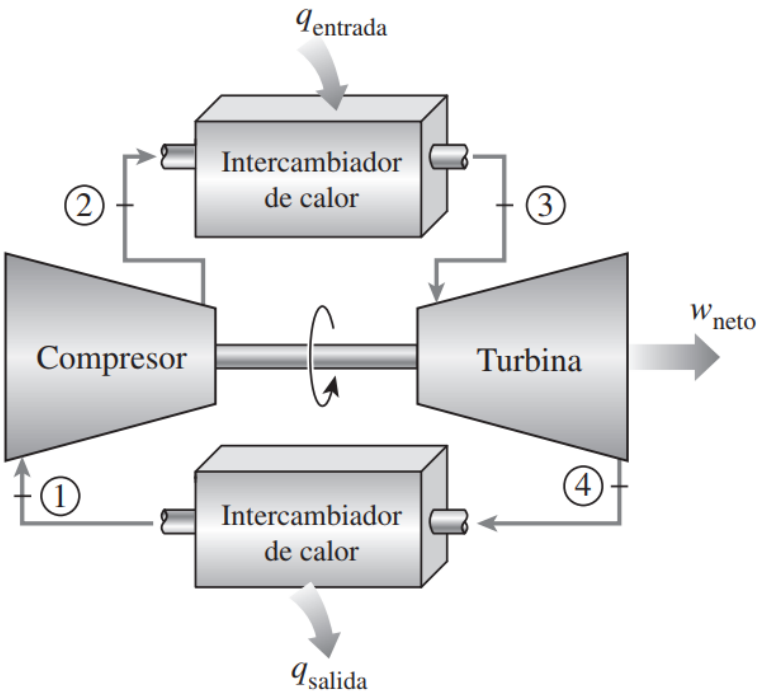


Diagrama recuperado (Çengel & Boles, 2015).

El aire que se emplea en el ciclo Brayton cumple con dos funciones: suministrar el oxígeno necesario para la combustión del combustible y funge como refrigerante, manteniendo la temperatura de diversos componentes dentro de los límites seguros. Para que la segunda función se pueda dar, se extrae más aire del necesario para la combustión entera del combustible. En una turbina de gas una relación de masa de aire y combustible de 50 o más es muy común.

Las dos áreas principales de aplicación de turbinas de gas son la propulsión de aviones y la generación de energía eléctrica. En el segundo caso, la generación se puede dar como unidad independiente, o bien, en conjunto con las centrales eléctricas de vapor en el lado de alta temperatura. En estas centrales los gases de escape sirven como fuente de calor para el vapor.

En las centrales eléctricas de ciclo Brayton el compresor y la turbina se encuentran unidos por un mismo eje. Por lo que existe una relación directa entre el trabajo del compresor y el trabajo de la turbina (relación del trabajo de retroceso). Esta condición del sistema es muy importante, dado que usualmente más de la mitad de la salida de trabajo de la turbina se utiliza para activar el compresor (Çengel & Boles, 2015). De esta forma, una central cuya relación del trabajo de retroceso es alta requiere una turbina más grande para suministrar los requerimientos energéticos adicionales que el compresor representa.

Una vez cubierto el ciclo Brayton, se abordará la segunda fase del CC: el ciclo Rankine. El ciclo Rankine es el modelo idealizado para las centrales eléctricas de vapor. Está compuesto por los siguientes procesos (ver ilustración 7):

- 1-2 Compresión isoentrópica en una bomba.
- 2-3 Adición de calor a presión constante en una caldera.
- 3-4 Expansión isoentrópica en una turbina.
- 4-1 Rechazo de calor a presión constante en un condensador.

En el ciclo Rankine, el agua entra a la bomba como líquido saturado y se comprime isoentrópicamente hasta la presión de operación de la caldera. Luego, el agua entra a la caldera como líquido comprimido (a alta presión) y sale como vapor sobrecalentado. La caldera es un gran intercambiador de calor en donde el calor que se origina en los gases de combustión, reactor nuclear u otras fuentes se transfiere al agua a presión constante. A la caldera junto con el sobrecalentador reciben el nombre de generador de vapor. Una vez que el vapor es sobrecalentado, entra a la turbina donde sufre una expansión isoentrópica y produce trabajo al hacer girar el eje conectado a un generador eléctrico. Al salir de la turbina, la presión y la temperatura del vapor llegan al punto de un vapor húmedo con alta calidad. De ahí, entra al condensador donde el agua se condensa a presión constante. El condensador es un gran intercambiador de calor que transfiere la energía hacia un sumidero de calor que puede ser un lago, un río o la atmósfera. Así, el agua sale como líquido saturado y entra de nuevo a la bomba, lo que completa el ciclo (Çengel & Boles, 2015).

Ilustración 7: Ciclo Rankine

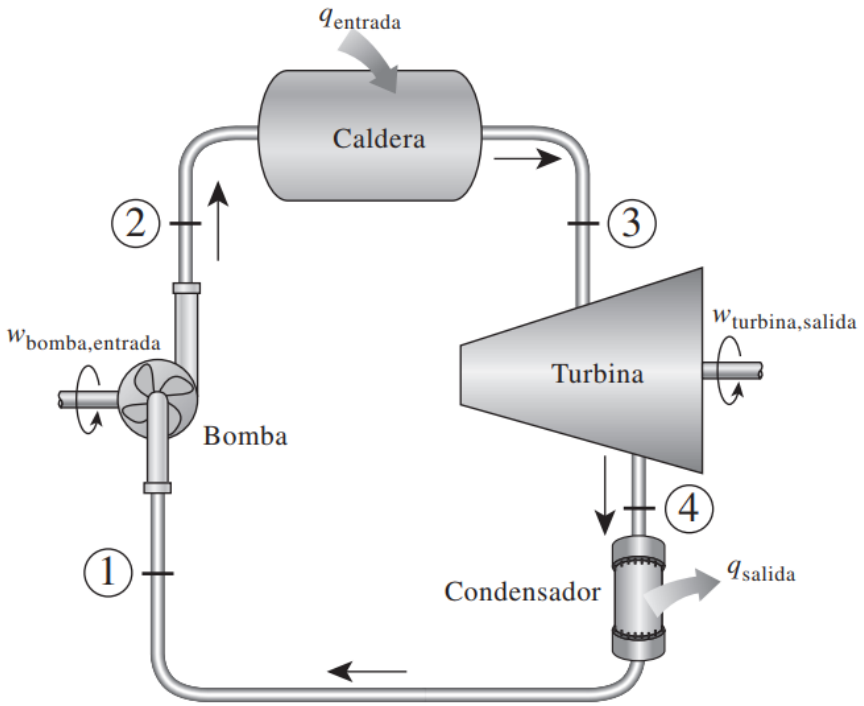


Diagrama recuperado (Çengel & Boles, 2015).

Una vez que se han cubierto las bases tanto del ciclo Rankine como del ciclo Brayton, se revisará la aplicación que ambos tienen en conjunto. Para empezar, los ciclos característicos de una turbina de gas operan a temperaturas considerablemente más altas que las de vapor. La temperatura máxima del agua al entrar en la turbina de vapor se acerca a los 620 °C, mientras que los gases de combustión en la turbina de gas alcanzan los 1425 °C. En realidad, el valor de la temperatura de los gases es mayor a los 1500 °C cuando salen de los quemadores, pero se pierde energía hasta que llegan a la turbina. Debido a la temperatura promedio más alta a la cual se suministra el calor, los ciclos con turbina de gas tienen un potencial mayor para eficiencias térmicas más elevadas. Sin embargo, el gas sale de la turbina a temperaturas muy altas (arriba de los 500 °C), por lo que se cancela cualquier ganancia de potencial en la eficiencia térmica.

Por tanto, desde el punto de vista de la ingeniería, conviene aprovechar los gases de escape a alta temperatura de las turbinas de gas como fuente de energía de un ciclo con un intervalo de temperatura inferior, como lo es un ciclo de potencia de vapor. Así se crea el CC. En este ciclo, la energía se recupera de los gases de escape y se transfiere al vapor mediante un intercambiador de calor que funge como caldera. Por lo general, más de una turbina de gas es necesaria para suministrar suficiente calor al vapor (Ilustración 8).

Ilustración 8: Ciclo combinado de gas y vapor

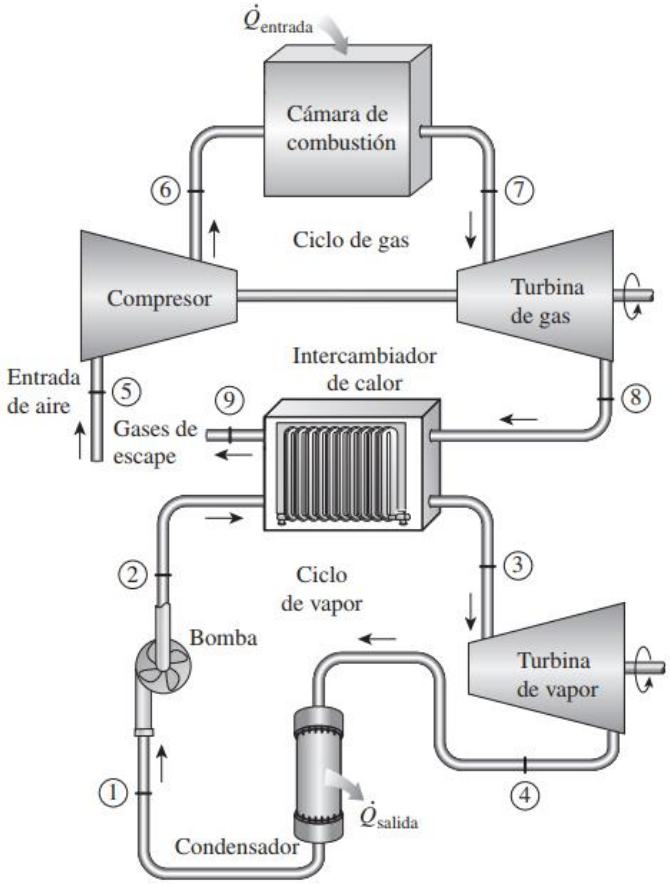


Diagrama recuperado (Çengel & Boles, 2015).

Debido a los recientes desarrollos tecnológicos para las turbinas de gas, se ha logrado que el CC resulte muy atractivo desde el punto de vista económico, puesto que el ciclo aumenta la eficiencia sin incrementar mucho el costo inicial. Es por ello que muchas centrales eléctricas nuevas operan con el CC, mientras que centrales inicialmente de vapor o de turbina de gas han optado por convertirse en centrales de CC. El resultado de esta conversión es que sean reportado eficiencias térmicas muy por encima del 40 por ciento. Y, en algunas centrales modernas, el CC ha permitido tener eficiencias por arriba del 60 por ciento (Çengel & Boles, 2015).

2.2 Fundamentos de Machine Learning

A través de la historia, el ser humano ha tenido la idea de mecanizar el proceso del pensamiento. La posibilidad de colocar una mente dentro de un cuerpo mecánico ha sido objeto de discusión desde la época de los antiguos griegos. De ellos es que nace uno de los mitos más resonantes en el contexto de la Inteligencia Artificial (IA) moderna: *Talos*, el hombre mecánico. Así, filósofos griegos, indios y chinos trabajaron en formas de materializar esta idea a lo largo del primer

milenio. Durante el siglo diecisiete, otros personajes como Gottfried Leibniz, Thomas Hobbes y René Descartes se enfocaron en el supuesto de racionalizar todo pensamiento como simples símbolos matemáticos. De esta forma, la visión de la IA ha existido por un largo tiempo, sin embargo, su aplicación es relativamente nueva.

La hipótesis sobre la que está construida la IA es que la mecanización del pensamiento es posible (Mueller & Massaron, 2016). El primer paso hacia la IA moderna fue dado por Alan Turing en su publicación "Computing Machinery and Intelligence" en 1950. En ésta, Turing exploró la idea de cómo determinar si una máquina puede pensar. Esta publicación, a la vez, continuo con el juego de la imitación (*Imitation Game*) que involucra tres jugadores. El juego consiste en tres participantes: dos humanos y una computadora. El jugador A (humano) y el jugador B (computadora) deben convencer al jugador C (humano) que cada uno es humano. Obviamente, el jugador C no puede ver a los otros dos jugadores, sino que la interacción se da de forma que físicamente no pueda descalificar a la computadora. Si el jugador C es incapaz de diferenciar quién es humano y quién no de forma repetida, la computadora gana.

Por otra parte, se encuentra Machine Learning (ML) que se sustenta en algoritmos para analizar grandes bases de datos. En la actualidad, ML no tiene la capacidad de alcanzar el nivel de IA que muchos añoran. Aun los mejores algoritmos son incapaces de pensar, sentir, tener consciencia de sí mismos y de ejercer libre albedrío. Lo que ML puede hacer es realizar análisis predictivos mucho más rápidos que cualquier humano, lo cual lo convierte únicamente en una herramienta. La relación que mantienen Machine Learning y la Inteligencia Artificial es que el primero dota de la capacidad de aprendizaje a la segunda. El concepto de IA es mucho más amplio que solamente la fase de aprendizaje. La mayor confusión que existe entre ML e IA proviene de la confusión entre aprendizaje e inteligencia. Las personas tienden a creer que cuando un algoritmo mejora en su trabajo (aprendizaje) lo hace porque es consciente de los cambios que realiza (inteligencia). Sin embargo, no hay nada que apoye esta visión sobre ML (Mueller & Massaron, 2016).

A diferencia de la IA, el concepto de ML es mucho más reciente. En primer lugar, no se podría hablar de ML sin la existencia de las computadoras. Uno de los primeros algoritmos que logró tener la capacidad de aprender por sí solo fue el perceptrón. Frank Rosenblatt ideó el perceptrón cuando trabajaba en el *Cornell Aeronautical Laboratory* con patrocinio de la naval de Estados Unidos en 1957. Rosenblatt era un psicólogo y pionero en el campo de la IA, cuyo proyecto era crear una computadora que pudiera aprender a través de prueba y error, tal y como los humanos lo hacen.

La idea se llevó a cabo exitosamente. El perceptrón en un inicio no se concibió solamente como un algoritmo, sino como software corriendo en un hardware especializado (una computadora dedicada). Éste era capaz de hacer reconocimiento de imágenes mucho más avanzado que cualquier software de la época. El éxito inicial del algoritmo condujo a su creador a decir que el perceptrón era la primera de una nueva generación de computadoras capaces de caminar, hablar,

ver, escribir e incluso reproducirse y ser conscientes de sí mismos. Sin embargo, el perceptrón nunca logró cumplir con esas expectativas. Pronto evidenció capacidades limitadas, incluso en el reconocimiento de imágenes (su especialidad). La decepción que generó el fracaso del perceptrón inició el Invierno de la Inteligencia Artificial y el abandono temporal de las investigaciones sobre la AI. Fue hasta la década de los 80's que el Conexionismo revivió el interés por ML.

Machine Learning es la ciencia de programar computadoras de modo que puedan aprender a partir de datos (Géron, 2017). Y así como la IA tiene en su núcleo una hipótesis, en el núcleo de ML se encuentra la hipótesis de que se puede representar la realidad utilizando una función matemática que el algoritmo no conoce inicialmente, pero que, después de haber visto (analizado) algo de información, puede formular, o bien, adivinar. De esta forma, se sostiene que se puede expresar la realidad y toda su complejidad en términos de una función matemática desconocida (función de mapeo). Por tanto, los algoritmos de ML tienen la tarea de hallar esa función y hacerla útil y conveniente (Mueller & Massaron, 2016).

Para comprender mejor el funcionamiento de ML, se ejemplificará con diagramas de flujo las diferencias existentes entre la programación tradicional y la programación empleando ML. Cabe mencionar que el proceso lógico y estructural en las dos formas es el mismo: se tiene un problema, se analiza el problema, se propone una generalización o solución del problema y se evalúa dicha solución. En caso de que la evaluación resulte negativa, o bien, insuficiente de acuerdo con el objetivo inicial, se analizan los errores de la propuesta y se inicia de nuevo. Cuando la evaluación es positiva o cumple con los límites establecidos, se lanza el programa.

Ilustración 9: Secuencia de programación tradicional

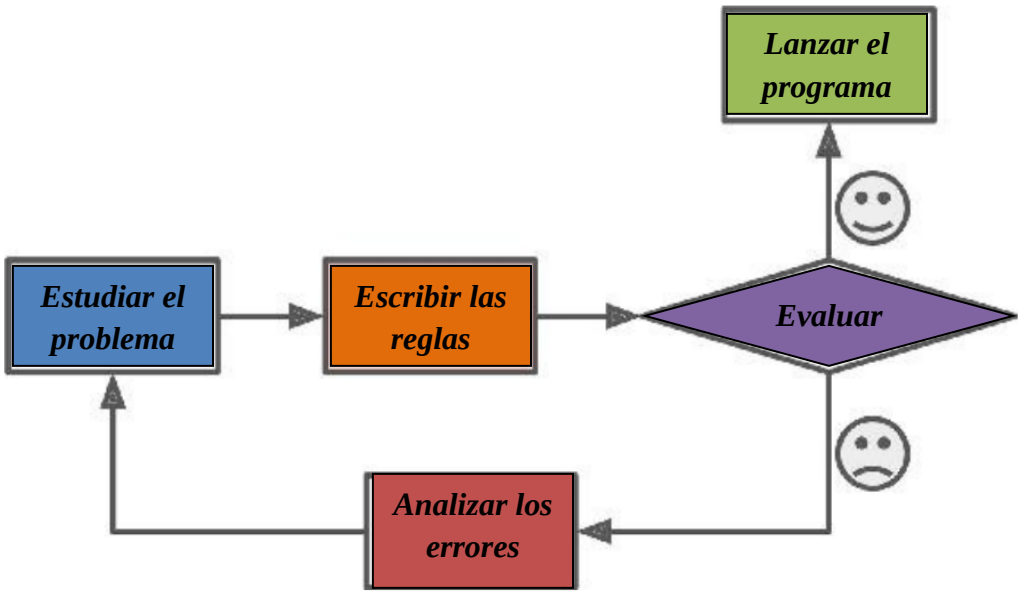


Diagrama recuperado de (Géron, 2017) con traducción propia.

Ilustración 10: Secuencia de programación con Machine Learning

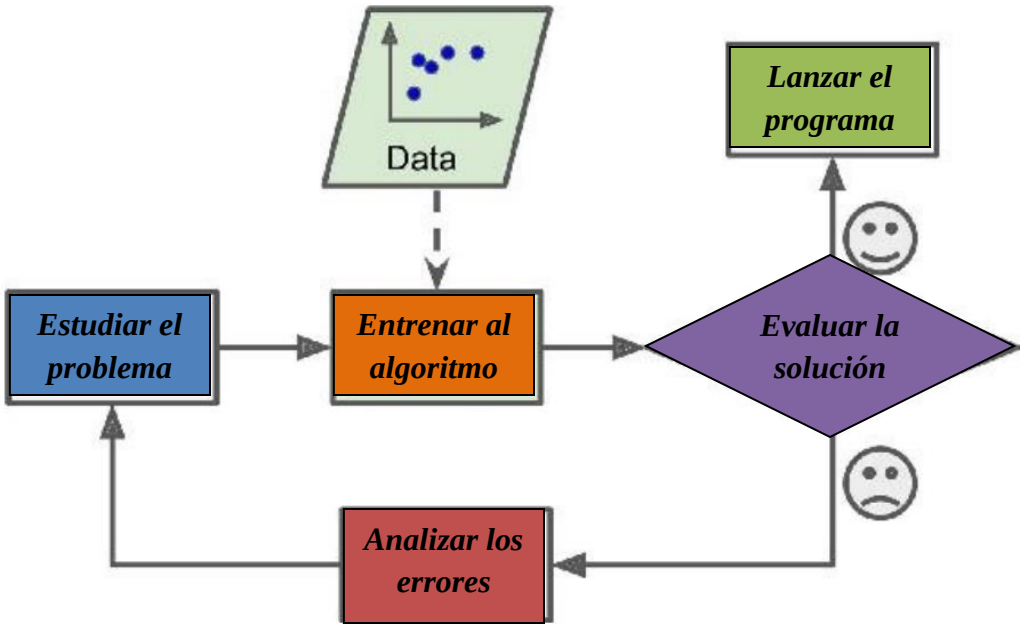


Diagrama recuperado de (Géron, 2017) con traducción propia.

Al comparar los dos diagramas (Ilustraciones 9 y 10), se observa que existe una sola diferencia entre ambos. En la secuencia tradicional, quien realiza la propuesta de solución es una persona; mientras que, en la secuencia con ML, la propuesta de solución es dada por una computadora a través del algoritmo. Ahora, la sustitución de la propuesta humana no significa un acto de holgazanería o irresponsabilidad, el intelecto humano se sigue empleando un paso más adelante al interpretar el modelo al que el algoritmo llegó y usarlo con cautela. Esto es porque muchos de los problemas que se abordan conllevan consecuencias altas y el algoritmo no tiene manera de discernir lo correcto de lo incorrecto en el plano ético, simplemente optimiza a partir de la información que se le es dada.

Por otra parte, la complejidad del mundo moderno y la interconexión de la mayor parte de los diferentes sectores productivos han generado relaciones profundas que el análisis humano es incapaz de percibir en una primera instancia. A eso se le une el crecimiento inadvertido de información disponible, que conlleva a la creación de bases de datos masivas, las cuales son demasiado grandes para que una persona pueda comprenderlas en su totalidad. Todos estos factores son los que hicieron necesaria una herramienta como ML.

Existe una gran variedad de algoritmos a los cuales se les puede calificar como ML, todos ellos son muy diversos en cuanto a complejidad y difieren en las bases matemáticas que los sustentan. Sin embargo, hay ciertos aspectos generales que se presentan en todos los algoritmos y a partir de los cuales se puede hacer una clasificación de éstos. Aquí se presentarán cuatro características que diferencian a los algoritmos de ML cuando se analizan como sistemas.

La forma principal por la que se puede separar a los algoritmos de ML es a través de su forma de aprendizaje. Existen tres maneras en que los algoritmos aprenden a partir de una base de datos:

- Aprendizaje supervisado: En este aprendizaje el algoritmo aprende de datos que se le proveen como ejemplo junto a una respuesta asociada a éstos, de forma que luego pueda predecir la respuesta correcta al enfrentarse con nuevos ejemplos. Las respuestas pueden ser valores numéricos o bien una cadena de caracteres, como una clase o categoría. El aprendizaje supervisado es parecido al aprendizaje humano al mando de un maestro. El maestro provee de buenos ejemplos para que el alumno los reconozca y, de ahí, genere un conjunto de reglas generales de esos ejemplos específicos.
- Aprendizaje no supervisado: En este aprendizaje el algoritmo aprende de ejemplos que no tienen asociados ninguna respuesta, por lo que debe de determinar los patrones de los datos por su cuenta. Este tipo de algoritmo reestructura los datos en nuevas características que pueden representar una clase o una nueva serie de valores no correlacionados. Son útiles ya que ofrecen un entendimiento más profundo en el significado de bases de datos complejas, y también apoyan en la creación de nuevas entradas para algoritmos de aprendizaje supervisado.
- Aprendizaje por refuerzo: En este aprendizaje se le presentan al algoritmo ejemplos sin una respuesta asociada, nuevamente. Sin embargo, a la respuesta que el algoritmo propone se le asigna una retroalimentación positiva o negativa, de acuerdo con el objetivo del programa. Este aprendizaje se emplea en aplicaciones en donde el algoritmo debe hacer decisiones, las cuales conllevan consecuencias. De esta forma, el algoritmo aprende por prueba y error, cuando tiene una retroalimentación positiva aprende qué hacer y cuando es negativa qué no hacer, por lo que muchos de estos algoritmos son genéticos y van mejorando sus predicciones y desempeño de generación en generación.

La siguiente clasificación de algoritmos aplica únicamente al aprendizaje supervisado, aunque en el aprendizaje no supervisado se encuentren los mismos principios y simplemente por tecnicidades se les nombre diferente. Esta separación se basa en el tipo de respuesta que es capaz de otorgar el algoritmo. La primera forma de predicción es la regresión. En ésta, los algoritmos intentan estimar la función de mapeo (f) de una base de datos a partir de variables de entrada (x) y tener variables de salida (y) numéricas o continuas. Por otro lado, los algoritmos de clasificación intentan estimar la función de mapeo (f) a partir de variables de entrada (x) y tener variables de salida (y) discretas o categóricas (Garbade, 2018).

Dado que las predicciones de estos algoritmos tienen naturaleza distinta, la forma de evaluar su desempeño es diferente también. Los algoritmos de clasificación se evalúan a partir de su exactitud, es decir, cuántas predicciones clasifica de manera correcta. Por otra parte, los algoritmos de regresión se evalúan usando la *root mean squared error* (RMSE), o en español, la raíz del error cuadrático medio. Estos métodos de evaluación son aplicables solo a sus respectivos

algoritmos, por lo que un problema de clasificación no se puede evaluar con la RMSE (Brownlee, 2017).

Completando lo que se mencionó, para aprendizaje no supervisado el tipo de predicción es prácticamente el mismo, solo que, al no ocupar ejemplos con respuesta, se les nombra de forma distinta. Cuando el algoritmo agrupa los datos debido a sus características se le conoce como *clustering*. Cuando da una respuesta continua, se le conoce como asociación (Garbade, 2018).

La siguiente forma de caracterización de los algoritmos de ML se basa en si el sistema es capaz de aprender incrementalmente de un flujo de datos entrantes o no:

- Aprendizaje por lote (*Batch*): En este aprendizaje el sistema es incapaz de aprender incrementalmente. Generalmente esto consumirá una gran cantidad de tiempo y recursos de cómputo. El proceso se lleva de la siguiente forma, primero el sistema es entrenado, se evalúa y se lanza a trabajar sin la posibilidad de aprender nada más. Si se requiere que el algoritmo aprenda de más información, se necesita entrenar una nueva versión del sistema desde cero con la base de datos completa (no sólo los nuevos datos), detener el anterior sistema y reemplazarlo con el nuevo. Afortunadamente, el proceso de entrenar, evaluar y lanzar el sistema de ML puede automatizarse fácilmente, por lo que un sistema por lote se puede adaptar al cambio. Sin embargo, en sistemas que necesitan adaptarse rápidamente a datos cambiantes, la respuesta de este aprendizaje puede resultar muy lenta.
- Aprendizaje en línea: En este aprendizaje el sistema se entrena de forma incremental alimentándolo secuencialmente con datos ejemplo, ya sean individualmente o en pequeños grupos. Cada paso de aprendizaje es rápido y barato, por lo que el sistema puede aprender de los nuevos datos tan pronto como se generan. El aprendizaje en línea es muy útil cuando se tienen sistemas que reciben un flujo continuo de información y requieren adaptarse rápidamente al cambio o ser autónomos. También, si se tienen recursos de cómputo limitados es ideal, pues tan pronto se emplea la información, ésta puede ser desechada; así como entrenar una gran base de datos resulta más demandante que hacerlo por porciones pequeñas.

  Los sistemas de aprendizaje en línea tienen un parámetro distintivo conocido como tasa de aprendizaje. La tasa de aprendizaje determina la velocidad a la que el sistema se adapta a los cambios en la información. Si se tiene una tasa alta, el sistema se adaptará rápidamente a los nuevos datos, pero de la misma forma olvidará los antiguos. Si se tiene una tasa baja, el sistema tendrá una mayor inercia y aprenderá más lento, lo que significa que es menos propenso a secuencias de datos no representativos. Uno de los principales problemas de este aprendizaje es que, al alimentar los sistemas con mala información, su desempeño caerá gradualmente, por lo que se debe monitorear detalladamente y detener el aprendizaje cuando se vea una caída en el desempeño.

La última clasificación de los algoritmos de ML se basa en la forma en que logran una generalización de las bases de datos. Unos de los principales objetivos de ML es ser capaz de hacer predicciones correctas. Esto significa que, dado un número determinado de ejemplos de entrenamiento, el sistema debe ser capaz de generalizar reglas para predecir ejemplos que no ha visto antes. El verdadero objetivo de los sistemas de ML es desempeñarse bien en información nueva. Así, existen dos enfoques principales para la generalización:

- Aprendizaje basado en ejemplos: En este aprendizaje el sistema generaliza a partir de una medida de similitud. El proceso es el siguiente: se le dota una base de datos al sistema, éste analiza la base y encuentra reglas generales, cuando se le presenta un nuevo caso, recurre a una medida de similitud para determinar la salida. Esto significa que el algoritmo se basa en la memoria para generar una salida, pues asignará la misma salida que tiene el ejemplo que más se parezca al nuevo caso.
- Aprendizaje basado en modelos: Este aprendizaje se basa en crear un modelo matemático a partir de los ejemplos con el objetivo de hacer predicciones con él posteriormente. Existen muchos algoritmos que construyen una función a partir de los datos de entrenamiento. A diferencia del aprendizaje basado en ejemplos, los modelos se basan en distintas variables que afectan a la variable objetivo, por lo que detrás hay una implicación de causalidad y no solamente de similitud. El proceso completo para este aprendizaje es estudiar los datos, seleccionar un modelo adecuado para ellos, entrenarlo con los ejemplos de entrenamiento y aplicarlo a nuevos datos para hacer predicciones (llamado inferencia), esperando que se haya realizado una buena generalización.

Una vez que abordamos los tipos de sistemas en ML, podemos continuar con los principales problemas a los que se enfrenta este enfoque de comprensión de la realidad. Se ha observado hasta ahora que hay dos componentes principales para el funcionamiento de ML: el algoritmo que se emplea y los datos que lo entrenan. Por lo tanto, si se tiene un problema se deberá a uno de estos dos.

El primer problema con el que se puede encontrar es una cantidad insuficiente de datos de entrenamiento. Para la mayor parte de los algoritmos de ML es necesaria una gran cantidad de datos si se desea que funcione apropiadamente. Aun para resolver problemas muy simples, generalmente, se necesitan miles de ejemplos, y para problemas complejos (como reconocimiento de imágenes o voz) se pueden necesitar de millones de ejemplos (a menos de que se pueda reusar partes de un modelo ya existente). En un artículo publicado en 2001 por Michel Banko y Eric Brill (investigadores de Microsoft), se estudió el efecto de la disponibilidad de datos contra el uso de algoritmos complejos. Se concluyó que, con los suficientes datos, hasta los algoritmos más sencillos son capaces de igualar en resultados a los más complejos (Géron, 2017). Por lo que se pone en duda si la investigación debe estar enfocada en crear mejores algoritmos o en mejorar la disponibilidad de datos. Claro que, actualmente, el conseguir información es muy costoso, por lo que mejorar los algoritmos resulta conveniente.

El segundo problema que se presenta en ML es la presencia de datos no representativos en las bases de datos. Con el fin de hacer una buena generalización, es crucial que los datos de entrenamiento sean representativos de los nuevos casos para los que se quiere generalizar. Esto resulta muy difícil de apreciar. En bases de datos pequeñas los datos no representativos suelen tener un efecto alto, pero aún en las bases grandes puede existir un sesgo si el método de muestreo tiene fallas.

El siguiente problema son los datos de calidad pobre. Si la información de entrenamiento está llena de errores, *outliers* y ruido; será difícil detectar los patrones subyacentes para el sistema de ML, por lo que su desempeño será pobre. De esta forma, es totalmente valioso dedicar tiempo a limpiar los datos de entrenamiento. Los *outliers* se pueden descartar por completo o tratar de arreglar. Si hay datos faltantes en una variable se puede ignorar la variable (si faltan muchos datos), ignorar los ejemplos que tienen faltantes o rellenar los datos faltantes.

Otro problema que se tiene es contar con características (variables) irrelevantes. El sistema sólo es capaz de aprender si se tienen suficientes características relevantes y pocas o ninguna irrelevantes. Para obtener mejores características se puede hacer lo siguiente:

- Seleccionar características: Elegir las mejores y más útiles características para entrenar de las ya disponibles.
- Extraer características: Combinar características existentes para producir una más útil.
- Crear nuevas características con la búsqueda de nuevos datos.

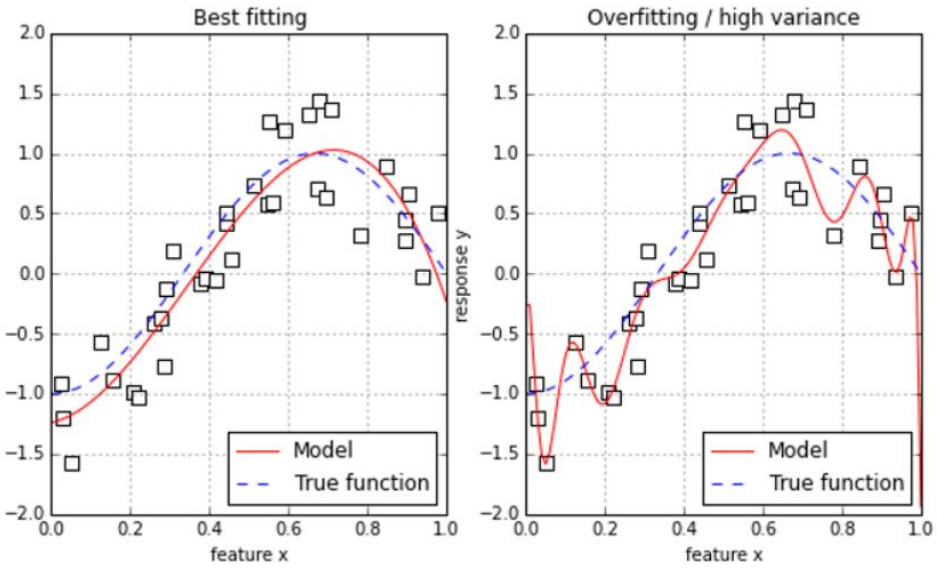
Ahora que tocamos los problemas con la información disponible, es hora de seguir con los problemas que pueden presentar los algoritmos de usar los incorrectos o incorrectamente.

El que es probablemente el principal problema con los algoritmos se conoce como sobreajuste (*overfitting*). La pronta generalización es uno de los problemas a los que nos enfrentamos día a día los humanos, y, desafortunadamente, ML también puede caer en este error si no se supervisa detalladamente. El *overfitting* significa que el modelo se desempeña bien con los datos de entrenamiento, pero no es capaz de hacer una buena generalización. Esta situación ocurre cuando el modelo es demasiado complejo comparado con el tamaño y el ruido de los datos de entrenamiento. Cuando se exploran las propiedades de un modelo con *overfitting* se puede observar que sus estimaciones tienen una variancia más grande que un modelo equilibrado (Ilustración 11). Dentro de las posibles soluciones se encuentran: simplificar el modelo seleccionando uno con menores parámetros, reduciendo el número de atributos de la base o restringiendo el modelo; recolectar más información de entrenamiento; y reducir el ruido en la base de entrenamiento.

Se conoce como regularización al acto de restringir un modelo para hacerlo más simple y reducir el riesgo de *overfitting*. La cantidad de regularización que se aplica durante el proceso de aprendizaje se puede controlar a partir de un hiperparámetro. Un hiperparámetro es un parámetro

del algoritmo de aprendizaje, no del modelo. El tema de los hiperparámetros se tocará más adelante.

Ilustración 11: Comparación de un modelo con overfitting y uno optimizado



Gráfica recuperada de (Mueller & Massaron, 2016).

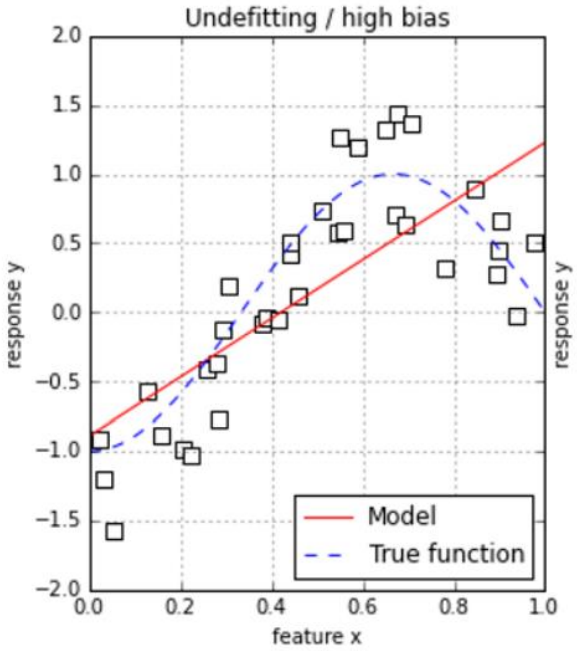
El segundo problema que se presenta con los algoritmos es lo contrario al *overfitting* y se conoce como *underfitting*. Éste ocurre cuando el modelo es demasiado simple para aprender la estructura subyacente de los datos. Contrario al *overfitting*, cuando se analiza un modelo con *underfitting* se observa que las estimaciones se caracterizan por tener un alto sesgo (Ilustración 12). Para solucionar este problema se puede hacer lo siguiente: elegir un modelo más poderoso con más parámetros, alimentar al algoritmo con mejores características (variables), y/o reducir las restricciones en el modelo (reducir la regularización de los hiperparámetros).

Detrás de la optimización de la mayor parte de los algoritmos de ML se encuentra una función interna conocida como la función de costo. Esta función es una función de evaluación que mide qué tan bien un algoritmo mapea la función objetivo que intenta adivinar (Mueller & Massaron, 2016). De esta forma, la función de evaluación compara las predicciones del algoritmo contra la respuesta que en verdad se dio en el mundo real. Al hacer esta comparación, la función de costo determina el nivel de error del algoritmo. La importancia de esta función reside en que le transmite al algoritmo lo que en realidad es significativo que aprenda. Así, la función de costo recompensa o penaliza las predicciones según sea el objetivo de aprendizaje.

En este momento es necesario adentrarse más al proceso de entrenamiento del algoritmo. Este consiste en tres fases: aprendizaje, validación y prueba. En un mundo perfecto, al presentarle toda la información disponible al algoritmo sobre un determinado problema, se podría probar

poner a prueba con datos de los que no ha aprendido antes y, de esta forma, corroborar su utilidad. Sin embargo, esperar por información fresca no siempre es factible en términos de tiempo y costos.

Ilustración 12: Ejemplo de un modelo con underfitting



Gráfica recuperada de (Mueller & Massaron, 2016).

La primera respuesta a este problema fue dividir aleatoriamente las bases de datos en dos conjuntos conocidos como el conjunto de entrenamiento y el conjunto de prueba. Como indican su nombre, el algoritmo se entrena con el primer conjunto y con el segundo se prueba su generalización hacia nuevos datos. Cabe señalar que la división de la base incluye tanto las variables de entrada como sus respuestas correspondientes. Las separaciones más comunes van del 20 al 30 por ciento de la base de datos para el conjunto de prueba y del 70 al 80 para el conjunto de entrenamiento (Mueller & Massaron, 2016) (Géron, 2017).

Este primer método es muy útil, no obstante, cuando es necesario afinar el algoritmo de aprendizaje se emplea otro enfoque. En este caso, se adiciona un tercer conjunto conocido como el conjunto de validación. Este conjunto equivale al 20 por ciento de la base, modificando el de entrenamiento a 70 por ciento y el de prueba al 10 por ciento (Mueller & Massaron, 2016). Existe un problema conocido como *snooping*, el cual se refiere a modificar los hiperparámetros del algoritmo para obtener un mejor rendimiento en el conjunto de prueba, en lugar del conjunto de entrenamiento. Cuando se hace esto, se moldea al algoritmo hacia el conjunto de prueba, en lugar de una buena generalización. Por lo que el conjunto de validación entra en escena al ser una

prueba piloto antes de la prueba definitiva con el conjunto de prueba. El conjunto de validación otorga una prueba de generalización al algoritmo entrenado con el conjunto de entrenamiento empleando datos desconocidos para éste, diferenciándose de la fase de prueba en el sentido que permite modificar los hiperparámetros del algoritmo. Así, se tiene una fase de prueba sin overfitting y se refuerza la generalización del sistema.

Es de suma importancia realizar las divisiones de la base de datos de forma aleatoria, debido a que de otra forma la fase de prueba no será confiable. El primer caso que se puede presentar es que los datos tengan un orden significativo, lo cual resultará en una sobreestimación. El segundo caso es que la distribución de la base difiera mucho de una sección a otra, lo que resulta en una subestimación. Esto es aplicable a menos que se emplee una serie de tiempo en el que el orden de la base es fundamental. A lo que esto nos lleva es a comprar estadísticamente los conjuntos de prueba y entrenamiento y cerciorarse que sus distribuciones sean similares. De otra forma, pueden no ser compatibles para la predicción la una de la otra.

Ahora, cuando se emplea esta separación de la base de datos se introduce un sesgo al proceso de prueba, debido a que se reduce el tamaño del conjunto de entrenamiento. Al separar la base, ejemplos útiles para el entrenamiento pueden quedar fuera. Y, aunque se verifiquen estadísticamente los conjuntos de prueba y entrenamiento, una combinación desafortunada de datos puede hacer parecer que son similares, cuando no lo son. Por lo que hacer esta separación resulta muy riesgosa para la presunción de generalización.

La respuesta a este problema se encuentra en una técnica llamada validación cruzada en k iteraciones o, en inglés, *cross-validation based on k-folds*. Esta técnica se basa en una separación aleatoria de igual forma, pero esta vez separa los datos en un número k de *folds*[9] del mismo tamaño. Luego, cada *fold* es ocupado como conjunto de prueba mientras que los otros se emplean como conjuntos de entrenamiento. Cada iteración usa un diferente *fold* como prueba, lo que produce una estimación de error. Y así se continua con cada *fold*, probándolo con el algoritmo entrenado con los demás *folds* y generando una nueva estimación de error, hasta que todos los *folds* se usan como prueba. Al final, todas las estimaciones se promedian y se obtiene su error estándar. En la ilustración 13 se muestra de manera gráfica el proceso.

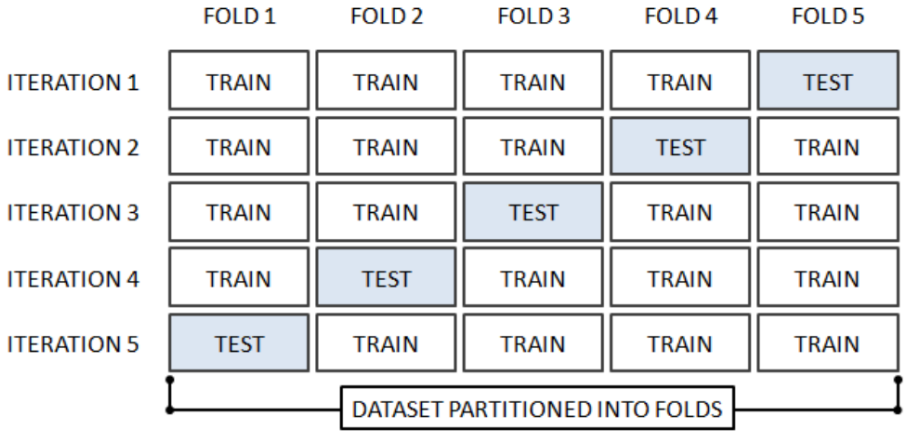
Las ventajas de esta técnica se enuncian a continuación:

- Funciona bien independientemente del número de ejemplos, debido a que, al incrementar el número de *folds*, se incrementa el tamaño del conjunto de entrenamiento (lo que reduce el sesgo) y se reduce el de prueba.
- Las diferencias en la distribución de un *fold* no importan demasiado, dado que se emplea solo una vez como conjunto de prueba y las demás ocasiones se mezcla con otros para formar parte del conjunto de entrenamiento.

[9] Porciones de datos

- Se evalúan todas las observaciones, por lo que se pone a prueba completamente la hipótesis de ML empleando toda la información que se tiene.
- Ocupando la media de los resultados se mejora el desempeño predictivo. Además, la desviación estándar de los resultados indica el nivel de variación que se puede esperar de nuevos datos reales. Se interpreta que, cuando el nivel de variación después de *cross-validation* es muy alto, el algoritmo no es capaz de mapear correctamente la función con las condiciones dadas.

Ilustración 13: Representación gráfica del funcionamiento de cross-validation



Gráfica recuperada de (Mueller & Massaron, 2016).

Una vez que se tiene un método confiable de aprendizaje y la hipótesis de ML se ha validado, se pasa a la afinación de los hiperparámetros. Los hiperparámetros son diferentes a los parámetros internos de los algoritmos en el sentido de que definen una hipótesis a priori, cuando los otros parámetros la especifican a posteriori. Es decir, los parámetros internos interactúan con los datos y activan un proceso de optimización de modo que encuentran qué valor de un parámetro arroja mejores resultados. Los hiperparámetros se eligen al inicio y no se modifican en el proceso, sino que simplemente al final del proceso se puede medir su utilidad. A la vez, no todos los algoritmos de ML requieren de una afinación intensiva de los hiperparámetros, pero los más complejos sí lo necesitan.

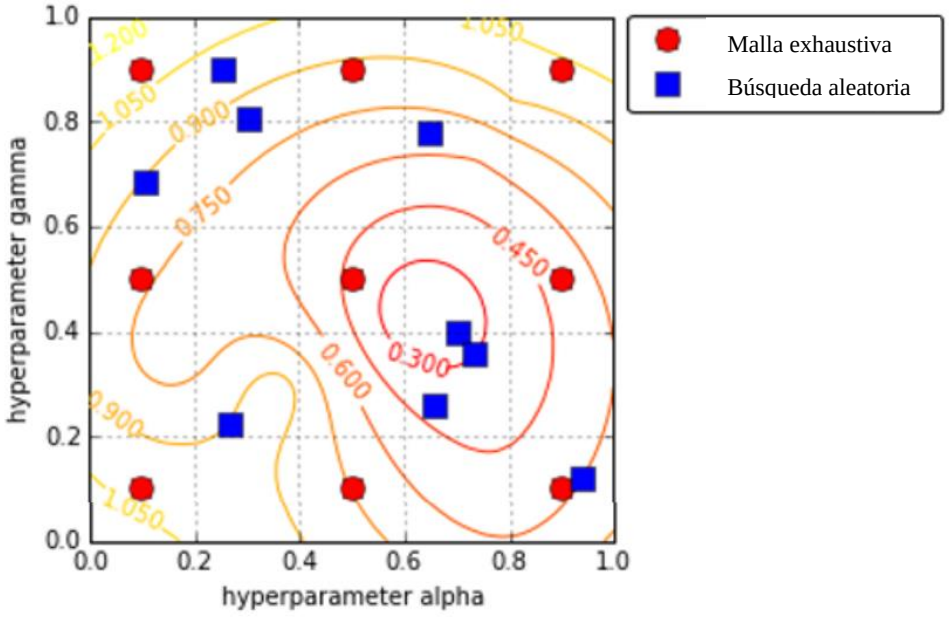
Dado que existe un número enorme de combinaciones posibles para los hiperparámetros de un algoritmo, se vuelve difícil decidir dónde empezar por optimizar. Dentro de un espacio de optimización se contienen combinaciones de valores que se desempeñan mejor o peor. Y aun cuando se tiene una buena combinación de hiperparámetros, no se puede estar seguro de que es la mejor opción[10]. El primer acercamiento que se tuvo para resolver este problema se conoce como búsqueda de malla o, en inglés, *grid search*. Una *grid search* es una técnica en la que se fijan

[10] Se puede estar en un mínimo local de la función de error.

distintos valores a los hiperparámetros de un algoritmo dentro de su respectivo espacio, se generan todas las posibles combinaciones entre ellos, y se prueban todas las combinaciones sobre los datos especificados usando *cross-validation*. Al final se elige el que muestre un mejor desempeño. Las ventajas que esta técnica presenta son que permite un muestreo sistemático de los posibles valores que se pueden introducir en un algoritmo y detectar el mínimo global cuando éste aparece. Sin embargo, tiene las desventajas de ser computacionalmente intensivo y de consumir un alto tiempo. Estas desventajas se pueden desplazar al considerar que las pruebas se pueden correr en paralelo en equipos de cómputo modernos. Pero, el hecho de que sea una técnica sistemática e intensiva aumenta las posibilidades de incurrir en errores que muestran resultados positivos en validaciones que son falsas causadas por el ruido presente en la base de datos.

Una *grid search* es conveniente cuando se exploran pocas posibles combinaciones, pero cuando el espacio de búsqueda de los hiperparámetros es grande, es preferible utilizar una búsqueda aleatoria. En lugar de gastar energía en probar combinaciones ligeramente diferentes con desempeños muy similares, se emplea esa energía en buscar muchas distintas combinaciones compuestas por un valor aleatorio diferente para cada hiperparámetro en cada iteración. Una búsqueda aleatoria prueba con menos combinaciones que una *grid search*, pero cubre mejor los rangos de cada hiperparámetro (Ilustración 14). Si ciertos parámetros son más significativos para el desempeño del algoritmo, como suele suceder, la estrategia a tomar es la búsqueda aleatoria por sobre la *grid search*.

Ilustración 14: Comparación de una grid search contra una búsqueda aleatoria en una función de error



Gráfica recuperada de (Mueller & Massaron, 2016).

Las ventajas que presenta una búsqueda aleatoria sobre la *grid search* son:

- Si se deja correr una búsqueda aleatoria por mil iteraciones, ésta probará mil valores distintos para cada hiperparámetro, en lugar de unos cuantos en una *grid search*.
- Se tiene un control mejor de los recursos que se quieren invertir en la mejora de los hiperparámetros al elegir el número de iteraciones, en lugar de calcular el número combinaciones muy parecidas entre sí.

Para que una búsqueda aleatoria sea exitosa se requieren de 15 a 60 iteraciones, aunque la cantidad de hiperparámetros determina si son necesarias más.

2.3 Máquinas de Vectores de Soporte (SVM)

Una vez que se adentró en las bases de Machine Learning y se revisaron sus aspectos generales, es hora de enfocarse en un algoritmo específico llamado Máquinas de Vectores de Soporte, o en inglés, *Support Vector Machines* (SVM). Las SVM's fueron creadas en la década de los 90s por el matemático Vladimir Vapnik y sus colegas al trabajar en los laboratorios de AT&T. En un inicio, los expertos de ML se mostraban escépticos ante este algoritmo, dado que no había ningún otro enfoque al que se pareciera. Sin embargo, ganó popularidad gracias a los resultados muy favorables que obtuvo en problemas de reconocimiento de imagen, como reconocimiento de escritura, los cuales eran muy complicados para la comunidad de ML del momento (Mueller & Massaron, 2016).

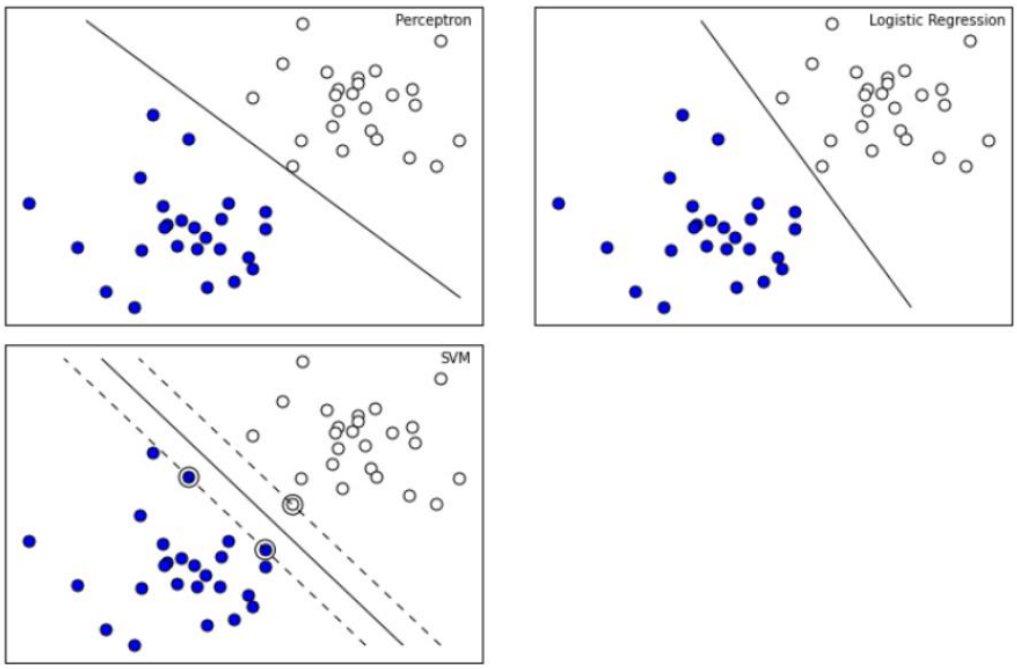
Hoy en día, este algoritmo es usado ampliamente por los científicos de datos, quienes lo aplican para una gran variedad de problemas, desde el diagnóstico médico hasta el reconocimiento de imágenes y la clasificación de textos. No obstante, esta técnica es limitada al usarla con *big data* debido a su falta de escalabidad cuando hay demasiados ejemplos y características.

Desde el punto de vista teórico, el aprendizaje a partir de vectores de soporte está basado en ideas simples y ofrece una demostración clara de lo que se trata el aprendizaje a través de ejemplos. Y al mismo tiempo, es capaz de alcanzar altos desempeños en aplicaciones prácticas. Por lo que surge la pregunta si este algoritmo es una de las mayores intersecciones entre la teoría del aprendizaje y la practicidad. Algunos algoritmos de aprendizaje pueden identificar los factores adecuados para aprender correctamente partiendo de un enfoque estadístico. En las aplicaciones del mundo real, por otra parte, se requieren de algoritmos más complejos y robustos, pero cuyo análisis teórico se vuelve mucho más difícil. Las SVM's tienen las dos características: construyen modelos que son suficientemente complejos para afrontar problemas reales y, a la vez, lo suficientemente simples para poder ser analizados matemáticamente (Schölkopf, 1998).

Las SVM's son modelos muy poderosos y versátiles capaces de realizar clasificaciones lineales y no lineales, regresiones y hasta detección de *outliers*. Para tener una mayor comprensión

de su funcionamiento es conveniente demostrarlo gráficamente como se muestra a continuación (Ilustración 15):

Ilustración 15: Comparación entre diferentes clasificadores lineales



Gráfica recuperada de (Mueller & Massaron, 2016).

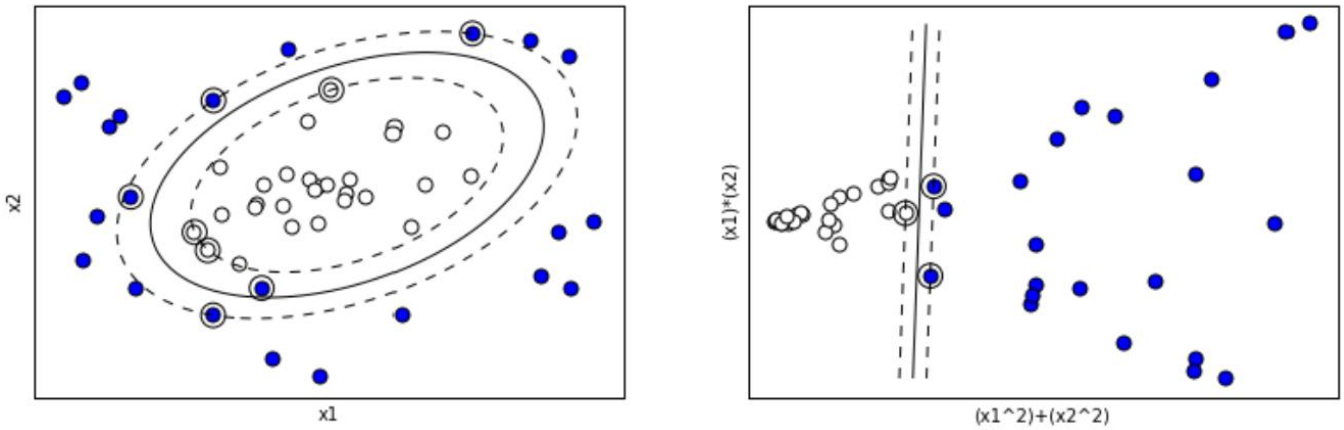
Se puede notar que las dos categorías son claramente separables linealmente. En las primeras dos gráficas se tienen un modelo de perceptrón y en el otro una regresión logística y ambos son capaces de separar las dos clases correctamente. Sin embargo, sus fronteras de decisión están tan pegadas a los datos de entrenamiento que, al momento de emplearlos con nuevos datos, no se desempeñarán tan bien. Por último, se tiene la gráfica representando la SVM, la línea sólida representa la frontera de decisión. Esta línea no sólo separa las dos clases correctamente, sino que, además, se mantiene tan separada de los ejemplos de entrenamiento más cercanos como es posible. De esta forma, los clasificadores SVM tratan de acomodar el margen (las líneas punteadas en la gráfica) más ancho posible entre las diferentes categorías. Al mismo tiempo, se debe notar que al añadir nuevos ejemplos fuera de ese margen no afecta en nada la frontera de decisión, ya que está completamente determinada o soportada por los ejemplos ubicados en los bordes del margen. A estos ejemplos de entrenamiento se les denomina vectores de soporte.

Comenzaremos por revisar los clasificadores lineales. Un tipo especial de clasificador es el de margen rígido. En éste, todos los ejemplos se encuentran fuera del margen definido por los vectores de soporte. Este clasificador tiene dos problemas: sólo funciona con categorías que son linealmente separables por completo y son muy sensibles a las anomalías que pueda presentar la base de datos. Para evitar estos problemas se ocupa una alternativa más flexible conocida como

clasificador de margen suave. Su objetivo es encontrar un buen balance entre mantener el margen más ancho posible y limitar las violaciones de margen, que son los ejemplos que se permiten dentro del margen o incluso en el lado incorrecto de la frontera de decisión. La generalización de este segundo clasificador suele ser mucho mejor que la del primero que se mencionó.

Ahora tocaremos a los clasificadores no lineales. El problema que se presenta en la mayoría de los casos del mundo real es que las clases no se pueden dividir linealmente. La primera solución que se dio para solucionar este problema fue usar una expansión polinomial, de modo que, en espacios dimensionalmente mayores, las categorías se puedan separar linealmente (Ilustración 16). Sin embargo, esta propuesta requiere de un alto nivel de cómputo, lo cual la hace muy demandante en recursos y tiempo. Por suerte, se encontró una mejor opción.

Ilustración 16: Transformación de variables a espacios dimensionales mayores

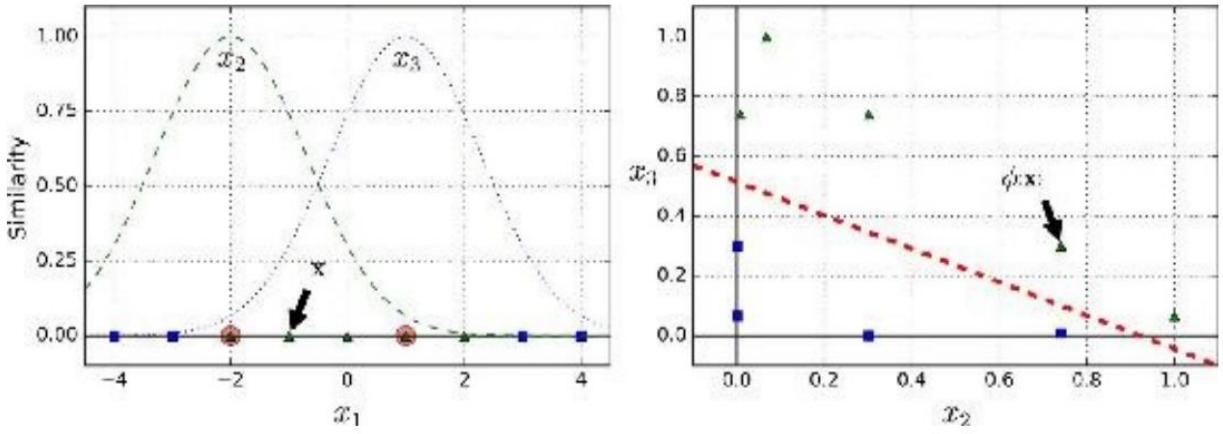


Gráfica recuperada de (Mueller & Massaron, 2016).

Esta técnica es matemáticamente milagrosa y se conoce como el *truco del kernel*. Algunos de los resultados que se obtienen con esta técnica equivalen a añadir muchas nuevas transformaciones polinomiales como variables de entrada, aun contando polinomios de alto grado, sin la necesidad de añadirlos verdaderamente. Por lo que no existe una explosión combinatoria del número de variables, ya que nunca son añadidas. Si la transformación que sufren las variables de entrada es simplemente de un cambio en el grado polinomial, el *kernel* se conoce como polinomial.

Para problemas no lineales también se pueden añadir variables provenientes de una función de similitud que mide qué tanto cada ejemplo se parece a un punto de referencia. En la ilustración 17 se muestran dos categorías diferentes contenidas en un espacio unidimensional (triángulos y cuadrados). En este caso, se escogen dos puntos de referencia -2 y 1 y a cada ejemplo se le aplica una función de similitud (función de base radial gaussiana, RBF). El resultado es un espacio bidimensional en el que son separables las categorías.

Ilustración 17: Uso de RBF para generar un espacio en el que las categorías sean linealmente separables



Gráfica recuperada de (Géron, 2017).

El enfoque más simple para el uso de esta técnica es que cada ejemplo de la base de datos se convierta en un punto de referencia. De esta forma, se crean muchas dimensiones incrementando las probabilidades de que el conjunto de entrenamiento sea linealmente separable. El problema con esta técnica es que se terminan generando tantas nuevas variables que resulta demasiado demandante. No obstante, así como para expansiones polinomiales es posible emplear el *truco del kernel*, para funciones de similitud también es posible. En especial, el *kernel* más usado es la función de base radial gaussiana. Este truco permite tener los mismos resultados que crear muchas nuevas variables sin tener que añadirlas verdaderamente.

Con esto se ha acabado la revisión conceptual de las SVM's, ahora partiremos a un ámbito más formal de su concepción y formulación matemática. He considerado que una introducción informal era necesaria para comprender las bases conceptuales del algoritmo, puesto que de tomar como referencia únicamente sus fundamentos matemáticos se vuelve muy complicado su entendimiento. Y, sobre todo, es innecesario que en una técnica tan intuitiva como esta, presentar las matemáticas sean un impedimento más que un complemento. Por lo que se comienza con un raciocinio más formal del algoritmo.

Detrás de la idea de las SVM's se encuentra un método lineal en un espacio de variables con dimensiones altas que mantienen una relación no lineal con el espacio de variables de entrada. Aún más, aunque se puede pensar como un algoritmo lineal en un espacio de altas dimensiones, en la práctica no se involucra ningún cálculo dentro ese espacio. Con el uso de *kernels*, todos los cálculos necesarios se realizan directamente sobre el espacio de las variables de entrada. Éste es el giro característico de las SVM's: se trata de algoritmos complejos empleados para el reconocimiento de patrones no lineales (clasificación), regresión, detección de outliers e identificación de características (aprendizaje no supervisado), pero se pueden simplificar a un modelo lineal. Todo esto que se acaba de describir se abordará a continuación partiendo desde el inicio de la teoría de aprendizaje.

Para hacer más sencilla la comprensión del funcionamiento del algoritmo, se empleará el ejemplo de una clasificación binomial (0,1). Partiendo de las bases, en la clasificación binomial se intenta estimar una función tal que $f: R^N \rightarrow \{0,1\}$ al utilizar datos de entrenamiento compuestos por variables de entrada $\boldsymbol{x_i}$[11] y respuestas y_i,

$$(\boldsymbol{x_1}, y_1), \dots, (\boldsymbol{x_l}, y_l) \in R^N \times \{0,1\},$$

tal que f clasifique correctamente nuevos ejemplos $(\boldsymbol{x}, y)$, es decir, $f(\boldsymbol{x}) = y$ para ejemplos $(\boldsymbol{x}, y)$. Se hace la suposición de que los nuevos ejemplos provienen de la misma distribución de probabilidad $P(\boldsymbol{x}, y)$ subyacente que los datos de entrenamiento. Si no se impone ninguna restricción sobre la clase de funciones a utilizar para hacer las estimaciones, aun cuando una función se desempeñe bien en el conjunto de entrenamiento, es decir, $f(\boldsymbol{x_i}) = y_i$; no significa que generalice con ejemplos nunca vistos. Por lo que si no tiene ninguna información adicional acerca de f, entonces los valores en el conjunto de entrenamiento no llevan ninguna información acerca de los valores en nuevos conjuntos. De esta forma, el aprendizaje es imposible y minimizar el error en el entrenamiento no significa un error esperado pequeño en el conjunto de prueba.

La teoría del aprendizaje estadístico muestra que es crucial restringir la clase de funciones que el algoritmo de ML puede implementar. Se debe elegir una función con la capacidad adecuada para la cantidad de datos de entrenamiento disponibles. De esta forma, para diseñar algoritmos de aprendizaje se debe encontrar o llegar a una clase de funciones cuya capacidad sea calculable. Los clasificadores SVM están basados en la clase de los hiperplanos:

$$(\boldsymbol{w}^T \cdot \boldsymbol{x}) + b = 0, \qquad \boldsymbol{w} \epsilon R^N, b \epsilon R,$$

que corresponden a funciones de decisión

$$f(\boldsymbol{x}) = hardlim\big((\boldsymbol{w}^T \cdot \boldsymbol{x}) + b\big).$$

Y de ahí, se puede demostrar que el hiperplano óptimo, aquel con el margen máximo de separación entre las dos clases, tiene la capacidad más baja.

Pero antes de llegar a encontrar el hiperplano óptimo, se definirá la función de decisión. El vector $\boldsymbol{w}$ se conoce como el vector de pesos de las variables de entrada y el valor b es llamado sesgo. Así, para la clasificación binomial, si el resultado de la expresión $(\boldsymbol{w}^T \cdot \boldsymbol{x}) + b$ es positivo, la clase predicha $\hat{y}$ es la clase positiva (1), de lo contrario pertenece a la clase negativa (0). Esto se expresa de la siguiente forma:

$$\hat{y} = \begin{cases} 0 & si\ (\boldsymbol{w}^T \cdot \boldsymbol{x}) + b < 0, \\ 1 & si\ (\boldsymbol{w}^T \cdot \boldsymbol{x}) + b \geq 0 \end{cases}$$

Como ya hemos mencionado, la expresión $(\boldsymbol{w}^T \cdot \boldsymbol{x}) + b$ constituye un hiperplano, en el que el vector $\boldsymbol{w}$ determina la pendiente y el valor b su desplazamiento. De esta forma, entrenar un

[11] El uso de negritas se refiere a vectores, las minúsculas son valores singulares y las mayúsculas representan matrices.

clasificador lineal SVM significa encontrar los valores de $\boldsymbol{w}$ y b que hacen al margen lo más ancho posible al evitar violaciones del margen o limitándolas.

La pendiente de la función de decisión equivale a la norma del vector de pesos. Al hacer menos pronunciada la pendiente de la función de decisión, el margen se hace más grande. Esto significa que al hacer más pequeño el vector de pesos, el margen entre las dos categorías se ampliará. Por ello, se desea minimizar $\boldsymbol{w}$ para hacer el margen más grande.

Sin embargo, si también se quiere evitar cualquier violación del margen, se requiere de una función de decisión que sea mayor que μ[12] para todos los ejemplos de entrenamiento positivos, y menor que $-\mu$ para los ejemplos de entrenamiento negativos. Si se define a $t^{(i)} = -\mu$ para los ejemplos negativos ($y^{(i)} = 0$) y $t^{(i)} = \mu$ para los ejemplos positivos ($y^{(i)} = 1$), entonces se puede expresar la siguiente restricción para todos los ejemplos $t^{(i)}\left(\boldsymbol{w}^T \cdot \boldsymbol{x}^{(i)} + b\right) \geq 1$.

De esta forma, el clasificador de margen rígido se puede expresar como el siguiente problema optimización con restricciones:

$$minimizar_{\boldsymbol{w},b} \quad \frac{1}{2}\boldsymbol{w}^T \cdot \boldsymbol{w}$$

$$sujeto\ a\ t^{(i)}\left(\boldsymbol{w}^T \cdot \boldsymbol{x}^{(i)} + b\right) \geq 1 \quad para \quad i = 1,2,\dots,m$$

Se emplea $\frac{1}{2}\boldsymbol{w}^T \cdot \boldsymbol{w}$, el cual es igual a $\frac{1}{2}\boldsymbol{w}^2$, en lugar de minimizar $\boldsymbol{w}$. Esto se debe a que se obtendrá el mismo resultado, ya que los valores de $\boldsymbol{w}$ y b que minimizan su valor, también minimizan a la mitad de su cuadrado. Pero se diferencian en que $\frac{1}{2}\boldsymbol{w}^2$ tiene una derivada simple ($\boldsymbol{w}$), mientras que $\boldsymbol{w}$ no es diferenciable en $\boldsymbol{w} = 0$. Y se debe tener en cuenta de que los problemas de optimización funcionan mucho mejor con funciones diferenciables.

Ahora, para tener un clasificador de margen suave es necesario introducir una variable de holgura $\zeta^{(i)} \geq 0$ a cada ejemplo. $\zeta^{(i)}$ mide cuánto se le permite al i-ésimo ejemplo violar el margen. Lo que significa que se cuenta con dos objetivos en conflicto: hacer la variable de holgura lo más pequeña posible para reducir el número de violaciones del margen, y minimizar al vector de pesos para hacer al margen lo más ancho posible. Aquí es donde el hiperparámetro C (relacionado con la función de costo) entra en escena, pues es él quien se encarga en regular el intercambio entre estos dos objetivos.

Así, el problema de optimización con restricciones es:

$$minimizar_{\boldsymbol{w},b,\zeta} \quad \frac{1}{2}\boldsymbol{w}^T \cdot \boldsymbol{w} + C\sum_{i=1}^{m}\zeta^{(i)}$$

$$sujeto\ a\ t^{(i)}\left(\boldsymbol{w}^T \cdot X^{(i)} + b\right) \geq 1 - \zeta^{(i)}\ y\ \zeta^{(i)} \geq 0 \quad para \quad i = 1,2,\dots,m$$

[12] Representa la suma de la posición de la frontera de decisión y el ancho del margen.

Los problemas de los márgenes rígido y suave son problemas de optimización cuadrática convexa con restricciones lineales. Este tipo de problemas se conocen como problemas de Programación Cuadrática (QP, por sus siglas en inglés). La formulación general de este problema es la siguiente:

$$minimizar_{\boldsymbol{p}} \quad \frac{1}{2}\boldsymbol{p}^T \cdot H \cdot \boldsymbol{p} + \boldsymbol{f}^T \cdot \boldsymbol{p}$$

$$sujeto\ a\ A \cdot \boldsymbol{p} \leq \boldsymbol{b}$$

$$donde \begin{cases} \boldsymbol{p}\ es\ un\ vector\ de\ dimensión\ n_p\ (n_p = número\ de\ parámetros) \\ H\ es\ una\ matriz\ de\ n_p \times n_p \\ \boldsymbol{f}\ es\ un\ vector\ de\ dimensión\ n_p \\ A\ es\ una\ matriz\ de\ n_c \times n_p\ (n_c = número\ de\ restricciones) \\ \boldsymbol{b}\ es\ un\ vector\ de\ dimensión\ n_c \end{cases}$$

Se nota que la expresión $A \cdot \boldsymbol{p} \leq \boldsymbol{b}$ define un número n_c de restricciones. Para aplicar este modelo al problema del margen rígido, se deben ingresar los parámetros QP de la siguiente forma:

- $n_p = n + 1$, en donde n es el número de variables de entrada (el +1 se debe al término de sesgo).
- $n_c = m$, donde m es el número de ejemplos de entrenamiento.
- H es una matriz identidad de $n_p \times n_p$, con la excepción de contener un cero en su esquina izquierda superior para ignorar el sesgo.
- $\boldsymbol{f} = 0$, es un vector de dimensión n_p lleno de ceros.
- $\boldsymbol{b} = 1$, es un vector de dimensión n_c lleno de unos.
- $\boldsymbol{a}^{(i)} = -t^{(i)}\dot{\boldsymbol{X}}^{(i)}$, donde $\dot{\boldsymbol{X}}^{(i)}$ es igual a $\boldsymbol{x}^{(i)}$ con una variable de sesgo adicional $\dot{\boldsymbol{X}}_0 = 1$ y $\boldsymbol{a}^{(i)}$ es el vector que contiene los elementos de la i-ésima fila de A.

Así que una forma de entrenar al clasificador de margen rígido es usar un solucionador de problemas QP ingresando los parámetros anteriores. El vector $\boldsymbol{p}$ resultante contendrá al término de sesgo $b = p_0$ y al vector de pesos $w_i = p_i$ para $i = 1,2,\dots,m$. El problema del margen suave se puede resolver de manera similar. Sin embargo, para poder usar el *truco del kernel*, le echaremos un vistazo a un problema de optimización con restricciones diferente.

Al tener un problema de optimización con restricciones, conocido como el problema primal, es posible expresar un problema diferente pero estrechamente relacionado con el primero, conocido como el problema dual. La solución del problema dual normalmente provee de un límite inferior a la solución del problema primal, pero, bajo ciertas condiciones, puede tener la misma solución que el problema primal. Afortunadamente, el problema detrás de las SVM's cumple con estas condiciones: la función objetivo es convexa, y las desigualdades que conforman las restricciones son funciones convexas, continuas y diferenciables. Por lo que se puede elegir entre

resolver el problema dual o el primal, ambos tendrán el mismo resultado. A continuación, se mostrará cómo se obtiene el problema dual a partir del problema primal.

Para entender qué es la dualidad, primero es necesario entender el método de los multiplicadores de Lagrange. La idea general es transformar un problema de optimización con restricciones en uno sin restricciones al mover las restricciones dentro de la función objetivo. Así que se propondrá un ejemplo para comprender el método. Suponga que se desea encontrar los valores de x y y que minimizan la función $f(x,y) = x^2 + 2y$, sujeta a la restricción de igualdad $3x + 2y + 1 = 0$. Al usar el método de multiplicadores de Lagrange, se inicia definiendo una nueva función llamada función de Lagrange: $g(x,y,\alpha) = f(x,y) - \alpha(3x + 2y + 1)$. Cada restricción (en este caso solamente hay una) es tomada del problema original para integrarse en la nueva función multiplicada por una nueva variable llamada multiplicador de Lagrange.

Joseph-Louis Lagrange demostró que si $(\hat{x},\hat{y})$ es una solución del problema de optimización con restricciones, entonces debe existir una $\hat{\alpha}$ tal que $(\hat{x},\hat{y},\hat{\alpha})$ sea un punto estacionario de la función de Lagrange (un punto estacionario es un punto en donde todas las derivadas parciales son iguales a cero). En otras palabras, las soluciones para el problema de optimización con restricciones se encuentran en los puntos estacionarios de la función de Lagrange. Para obtener las soluciones se resuelve un sistema de ecuaciones.

Sin embargo, este método sólo es aplicable para restricciones de igualdad. Por suerte, bajo ciertas condiciones regulatorias (las cuales son respetadas por las SVM's), este método se puede generalizar para restricciones de desigualdad. La función de Lagrange generalizada para el problema del margen rígido se muestra a continuación, las variables $\alpha^{(i)}$ se conocen como los multiplicadores de Karush-Kun-Tucker (KKT) y deben de ser iguales o mayores a cero.

$$\mathcal{L}(\boldsymbol{w}, b, \alpha) = \frac{1}{2}\boldsymbol{w}^T \cdot \boldsymbol{w} - \sum_{i=1}^{m} \alpha^{(i)}\left(t^{(i)}\left(\boldsymbol{w}^T \cdot \boldsymbol{x}^{(i)} + b\right) - 1\right),\ donde\ \alpha^{(i)} \geq 0\ para\ i = 1,2,\dots,m$$

Al igual que con el método de los multiplicadores de Lagrange, se calculan las derivadas parciales de la nueva función y se buscan sus puntos estacionarios. Las condiciones que hacen posibles la generalización del método de Lagrange son las condiciones de KKT. De esta forma, si existe una solución para el problema, es necesario que se encuentre en los puntos estacionarios $\left(\hat{\boldsymbol{w}}, \hat{b}, \hat{\alpha}\right)$ que respeten las condiciones de KKT:

- Los puntos estacionarios deben respetar las restricciones: $t^{(i)}\left(\hat{\boldsymbol{w}}^T \cdot \boldsymbol{x}^{(i)} + \hat{b}\right) \geq 1$ para $i = 1,2,\dots,m$
- Corroborar que $\hat{\alpha}^{(i)} \geq 0\ para\ i = 1,2,\dots,m$.
- Los puntos estacionarios se componen de $\hat{\alpha}^{(i)} = 0$ o la i-ésima restricción debe ser una restricción activa, es decir, que respeta la igualdad $t^{(i)}\left(\hat{\boldsymbol{w}}^T \cdot \boldsymbol{x}^{(i)} + \hat{b}\right) = 1$. Esta condición se conoce como la condición de holgura complementaria. Lo que esta condición implica es

que $\hat{\alpha}^{(i)} = 0$ o el i-ésimo ejemplo se encuentra en el margen, por lo que es un vector de soporte.

Como se mencionó, las condiciones de KKT son necesarias para que un punto estacionario sea solución de un problema de optimización con restricciones. Bajo ciertas condiciones, las condiciones de KKT son también condiciones suficientes. Los problemas de optimización SVM cumplen con esas condiciones, por lo que cualquier punto estacionario que cumpla con las condiciones de KKT es solución garantizada del problema de optimización con restricciones.

Ahora calcularemos las derivadas parciales de la función de Lagrange generalizada respecto a $\boldsymbol{w}$ y a b:

$$\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}, b, \alpha) = \boldsymbol{w} - \sum_{i=1}^{m} \alpha^{(i)}\, t^{(i)}\, \boldsymbol{x}^{(i)}$$

$$\frac{\partial}{\partial b} \mathcal{L}(\boldsymbol{w}, b, \alpha) = -\sum_{i=1}^{m} \alpha^{(i)}\, t^{(i)}$$

Al igualar estas derivadas a cero, se obtiene:

$$\widehat{\boldsymbol{w}} = \sum_{i=1}^{m} \hat{\alpha}^{(i)}\, t^{(i)}\, \boldsymbol{x}^{(i)}$$

$$\sum_{i=1}^{m} \hat{\alpha}^{(i)}\, t^{(i)} = 0$$

Cuando sustituimos estos términos dentro de la definición de la función de Lagrange generalizada, desaparecen algunos términos y se obtiene la función:

$$\mathcal{L}\left(\widehat{\boldsymbol{w}}, \hat{b}, \alpha\right) = \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha^{(i)} \alpha^{(j)} t^{(i)} t^{(j)} \boldsymbol{x}^{(i)^T} \cdot \boldsymbol{x}^{(j)} - \sum_{i=1}^{m} \alpha^{(i)}\,,$$

$$donde\ \alpha^{(i)} \geq 0\ para\ i = 1,2, \ldots, m$$

Por tanto, el objetivo ahora es encontrar el vector $\hat{\alpha}$ que minimice esta función y que cumpla con $\hat{\alpha}^{(i)} \geq 0$ para todos los ejemplos de entrenamiento. Este problema de optimización con restricciones es el problema dual que se buscaba. Es pertinente recordar que, aunque usar el método de multiplicadores de Lagrange generalizado ingresa las restricciones a la función objetivo, surge una nueva restricción $\alpha^{(i)} \geq 0$.

Una vez que se encontró el $\hat{\alpha}$ óptimo empleando un solucionador de QP, se puede calcular el vector de pesos $\widehat{\boldsymbol{w}}$, empleando la expresión que se encuentra arriba. Para calcular $\hat{b}$, es posible usar el hecho de que un vector de soporte cumple con $t^{(i)}\left(\widehat{\boldsymbol{w}}^T \cdot \boldsymbol{x}^{(i)} + b\right) = 1$, por lo que si el k-

ésimo ejemplo es un vector de soporte ($\alpha_k > 0$), se puede calcular $\hat{b} = 1 - t^{(k)}\left(\hat{\boldsymbol{w}}^T \cdot \boldsymbol{x}^{(k)}\right)$. Sin embargo, comúnmente se prefiere calcular el sesgo promedio con todos los vectores de soporte, debido a que da un resultado más estable y preciso.

$$\hat{b} = \frac{1}{n_s} \sum_{\substack{i=1 \\ \hat{\alpha}^{(i)} \geq 0}}^{m} \left[1 - t^{(i)}\left(\hat{\boldsymbol{w}}^T \cdot \boldsymbol{x}^{(i)}\right)\right]$$

El problema dual se resuelve más rápido que el primal cuando el número de ejemplos de entrenamiento es más pequeño que el número de variables de entrada. Pero, más importante, hace que el *truco del kernel* sea posible. En seguida se explicará el efecto de los *kernels* sobre la base de datos. Se debe recordar que, para problemas no lineales, los ejemplos se llevan a un espacio dimensionalmente mayor para que puedan ser separados linealmente.

Para presentar el *truco del kernel* se empleará un ejemplo. Suponga que se le quiere aplicar una transformación polinomial de segundo grado a un conjunto de entrenamiento de dos dimensiones, y luego entrenar un clasificador lineal SVM con el conjunto transformado. Se muestra a continuación la transformación, donde ϕ es la función de mapeo (transformación):

$$\phi(\boldsymbol{x}) = \phi\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} {x_1}^2 \\ \sqrt{2}\, x_1 x_2 \\ {x_2}^2 \end{pmatrix}$$

El vector transformado es tridimensional en lugar de bidimensional, como el original. Ahora, se observará qué le ocurre a un par de vectores bidimensionales, $\boldsymbol{a}$ y $\boldsymbol{b}$, si se le aplica una transformación polinomial de segundo grado y luego se calcula el producto punto de los vectores transformados:

$$\phi(\boldsymbol{a})^T \cdot \phi(\boldsymbol{b}) = \begin{pmatrix} {a_1}^2 \\ \sqrt{2}\, a_1 a_2 \\ {a_2}^2 \end{pmatrix}^T \cdot \begin{pmatrix} b^2 \\ \sqrt{2}\, b_1 b_2 \\ {b_2}^2 \end{pmatrix} = {a_1}^2 {b_1}^2 + 2\, a_1 b_1 a_2 b_2 + {a_2}^2 {b_2}^2$$

$$= (a_1 b_1 + a_2 b_2)^2 = \left(\begin{pmatrix} a_1 \\ a_2 \end{pmatrix}^T \cdot \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}\right)^2 = (\boldsymbol{a}^T \cdot \boldsymbol{b})^2$$

De esta forma, se observa que el producto punto de los vectores transformados es igual que el cuadrado del producto punto de los vectores originales: $\phi(\boldsymbol{a})^T \cdot \phi(\boldsymbol{b}) = (\boldsymbol{a}^T \cdot \boldsymbol{b})^2$.

Si se le aplica la transformación ϕ a todos los ejemplos de entrenamiento, entonces el problema dual contendrá el producto punto $\phi\left(\boldsymbol{x}^{(i)}\right)^T \cdot \phi\left(\boldsymbol{x}^{(j)}\right)$. En el caso de que ϕ sea una transformación polinomial de segundo grado, se puede reemplazar este producto punto de los vectores transformados por $\left(\boldsymbol{x}^{(i)^T} \cdot \boldsymbol{x}^{(j)}\right)^2$ simplemente. Esto significa que en realidad no se tienen

que transformar todos los ejemplos de entrenamiento, basta con sustituir el producto punto entre vectores por su cuadrado en la expresión del problema dual. El resultado obtenido será el mismo que si se transformara todo el conjunto de entrenamiento, pero este truco hace que el proceso completo sea computacionalmente más eficiente por mucho. Esta es la esencia del *truco del kernel*.

En ML, un *kernel* es una función que es capaz de calcular el producto punto $\phi(\boldsymbol{a})^T \cdot \phi(\boldsymbol{b})$ empleando únicamente los vectores originales $\boldsymbol{a}$ y $\boldsymbol{b}$; sin tener que calcular o si quiera saber de la transformación ϕ. A continuación, se muestran los kernels más comunes.

- Lineal: $K(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{a}^T \cdot \boldsymbol{b}$.
- Polinomial: $K(\boldsymbol{a}, \boldsymbol{b}) = (\gamma \boldsymbol{a}^T \cdot \boldsymbol{b} + \boldsymbol{r})^d$.
- Función de base radial gaussiana: $K(\boldsymbol{a}, \boldsymbol{b}) = \exp(-\gamma \|\boldsymbol{a} - \boldsymbol{b}\|^2)$.
- Sigmoide: $K(\boldsymbol{a}, \boldsymbol{b}) = \tanh(\gamma \boldsymbol{a}^T \cdot \boldsymbol{b} + r)$.

Ahora, tomaremos un enfoque más formal en el *truco del kernel* revisando el teorema de Mercer. De acuerdo con el teorema de Mercer, si una función $K(\boldsymbol{a}, \boldsymbol{b})$ respecta unas cuantas condiciones matemáticas llamadas condiciones de Mercer: K debe ser continua y simétrica en sus argumentos, es decir, $K(\boldsymbol{a}, \boldsymbol{b}) = K(\boldsymbol{b}, \boldsymbol{a})$; entonces existe una función ϕ capaz de mapear a $\boldsymbol{a}$ y $\boldsymbol{b}$ a otro espacio (posiblemente con dimensiones mucho mayores) tal que $K(\boldsymbol{a}, \boldsymbol{b}) = \phi(\boldsymbol{a})^T \cdot \phi(\boldsymbol{b})$. Por lo que se puede usar a K como un kernel ya que se sabe que ϕ existe, aun cuando ϕ no se conoce. En el caso de la función de base radial gaussiana, se puede demostrar que ϕ mapea cada ejemplo de entrenamiento a un espacio dimensional infinito, así que es un alivio no tener que mapearlos en realidad.

Todavía nos queda un punto por tocar y es cómo regresar de la solución dual a la primal con el uso de *kernels*, pues en este caso se encuentran ecuaciones que contienen a $\phi(\boldsymbol{x}^{(i)})$. De hecho, $\widehat{\boldsymbol{w}}$ debe tener el mismo número de dimensiones que $\phi(\boldsymbol{x}^{(i)})$ por la ecuación $\widehat{\boldsymbol{w}} = \sum_{i=1}^{m} \hat{\alpha}^{(i)}\, t^{(i)}\, \phi(\boldsymbol{x}^{(i)})$, este número puede ser enorme o incluso infinito, por lo que no se puede calcular. Entonces, ¿cómo se pueden hacer predicciones sin conocer $\widehat{\boldsymbol{w}}$? Bueno, lo que se puede hacer es introducir la ecuación anterior de $\widehat{\boldsymbol{w}}$ en la función de decisión para nuevos ejemplos $\boldsymbol{x}^{(n)}$, lo que resulta en un producto punto entre dos vectores de entrada. Y cuando se tiene ese producto punto, se emplea nuevamente el truco del *kernel*.

La función de decisión es el hiperplano óptimo $f(\boldsymbol{x}) = (\boldsymbol{w}^T \cdot \boldsymbol{x}) + b$, pero que en este caso se modifica a:

$$h_{\widehat{\boldsymbol{w}},b}\left(\phi(\boldsymbol{x}^{(n)})\right) = \widehat{\boldsymbol{w}}^T \cdot \phi(\boldsymbol{x}^{(n)}) + \hat{b} = \left(\sum_{i=1}^{m} \hat{\alpha}^{(i)}\, t^{(i)}\, \phi(\boldsymbol{x}^{(i)})\right)^T \cdot \phi(\boldsymbol{x}^{(n)}) + \hat{b}$$

$$= \sum_{i=1}^{m} \hat{\alpha}^{(i)}\, t^{(i)} \left(\phi\left(\boldsymbol{x}^{(i)}\right)^T \cdot \phi\left(\boldsymbol{x}^{(n)}\right) \right) + \hat{b} = \sum_{\substack{i=1 \\ \hat{\alpha}^{(i)} > 0}}^{m} \hat{\alpha}^{(i)}\, t^{(i)}\, K\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(n)}\right) + \hat{b}$$

Se aprecia que, al tener $\alpha^{(i)} \neq 0$ únicamente para los vectores de soporte, realizar predicciones para nuevos vectores $\boldsymbol{x}^{(n)}$ involucra solamente el cálculo del producto punto entre los vectores de soporte y los nuevos ejemplos, no con el conjunto de entrenamiento completo. Claro que antes de hacer las predicciones se necesita calcular el término de sesgo $\hat{b}$, empleando el mismo truco:

$$\hat{b} = \frac{1}{n_s} \sum_{\substack{i=1 \\ \hat{\alpha}^{(i)} \geq 0}}^{m} \left[1 - t^{(i)} \left(\hat{\boldsymbol{w}}^T \cdot \phi\left(\boldsymbol{x}^{(i)}\right) \right) \right] = \frac{1}{n_s} \sum_{\substack{i=1 \\ \hat{\alpha}^{(i)} \geq 0}}^{m} \left[1 - t^{(i)} \left(\left(\sum_{j=1}^{m} \hat{\alpha}^{(j)}\, t^{(j)}\, \phi\left(\boldsymbol{x}^{(j)}\right) \right)^T \cdot \phi\left(\boldsymbol{x}^{(i)}\right) \right) \right]$$

$$= \frac{1}{n_s} \sum_{\substack{i=1 \\ \hat{\alpha}^{(i)} \geq 0}}^{m} \left[1 - t^{(i)} \sum_{\substack{j=1 \\ \hat{\alpha}^{(j)} \geq 0}}^{m} \hat{\alpha}^{(j)}\, t^{(j)} K\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}\right) \right]$$

Hasta ahora, la mayor restricción que se ha presentado es considerar solamente el caso de la clasificación. No obstante, es posible obtener una generalización para un modelo de regresión con este algoritmo ($y \epsilon R$). En este caso, el algoritmo intenta construir una función lineal en el espacio de las variables de entrada tal que los ejemplos de entrenamiento se coloquen dentro una distancia $\varepsilon > 0$ de la función lineal. De manera similar a la clasificación, este problema se puede formular como un problema QP y emplear transformaciones con *kernels*. Por lo que es un algoritmo versátil para problemas lineales y no lineales.

De forma simplificada, la regresión SVM busca acomodar tantos ejemplos de entrenamiento como sea posible dentro del margen que acompaña a la curva de regresión. El ancho del margen está definido por el hiperparámetro ε (Ilustración 18).

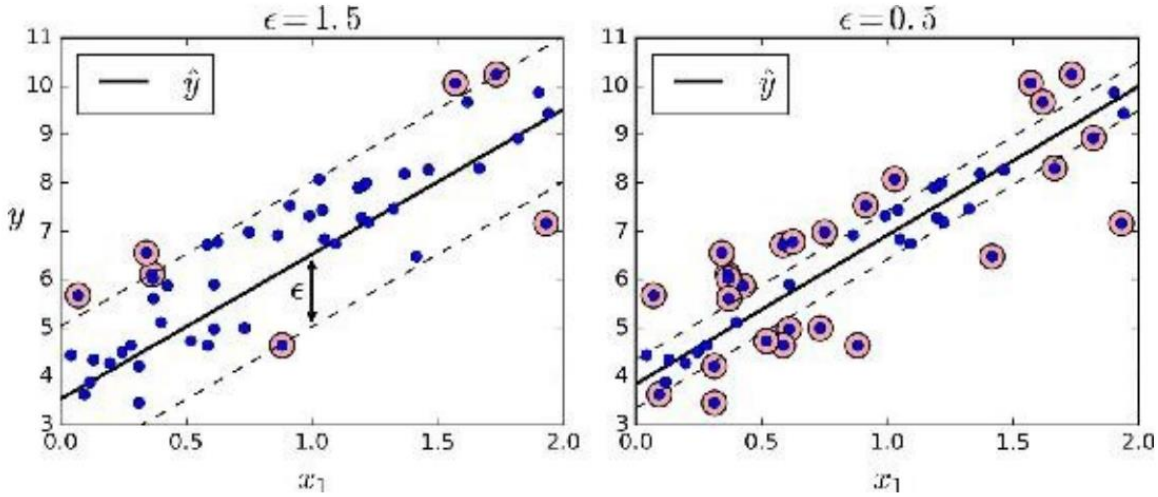
La formulación de este problema es muy similar al de clasificación, solo que incluye unas diferencias significativas. Se tienen datos de entrenamiento compuestos por variables de entrada $\boldsymbol{x_i}$ y respuestas y_i, y se trata de encontrar la función y su minimización:

$$f(\boldsymbol{x}) = (\boldsymbol{w}^T \cdot \boldsymbol{x}) + b$$

$$minimizar_{\boldsymbol{w}} \quad \frac{1}{2} \boldsymbol{w}^T \cdot \boldsymbol{w}$$

$$sujeto\ a \left| y^{(i)} - \left(\boldsymbol{w}^T \cdot \boldsymbol{x}^{(i)} + b \right) \right| \leq \varepsilon \quad para\ i = 1,2, \ldots, m$$

Ilustración 18: Ejemplos de regresión SVM



Gráfica recuperada de (Géron, 2017).

Este es el problema del margen rígido para regresión. Sin embargo, como es muy raro que alguna base de datos se alinee con este modelo, se introducen dos variables de holgura para cada punto: $\xi^{(i)}$ y $\xi^{*(i)}$ (Ilustración 19). Por lo que se tiene el siguiente problema primal:

$$minimizar_{\boldsymbol{w},\xi,\xi^*} \quad \frac{1}{2}\boldsymbol{w}^T \cdot \boldsymbol{w} + C\sum_{i=1}^{m}\left(\xi^{(i)} + \xi^{*(i)}\right)$$

$sujeto\ a$:

$$y^{(i)} - \left(\boldsymbol{w}^T \cdot \boldsymbol{x}^{(i)} + b\right) \leq \varepsilon + \xi^{(i)},$$

$$\left(\boldsymbol{w}^T \cdot \boldsymbol{x}^{(i)} + b\right) - y^{(i)} \leq \varepsilon + \xi^{*(i)},$$

$$\xi^{(i)} \geq 0 \text{ y } \xi^{*(i)} \geq 0 \; para\; i = 1,2,\dots,m$$

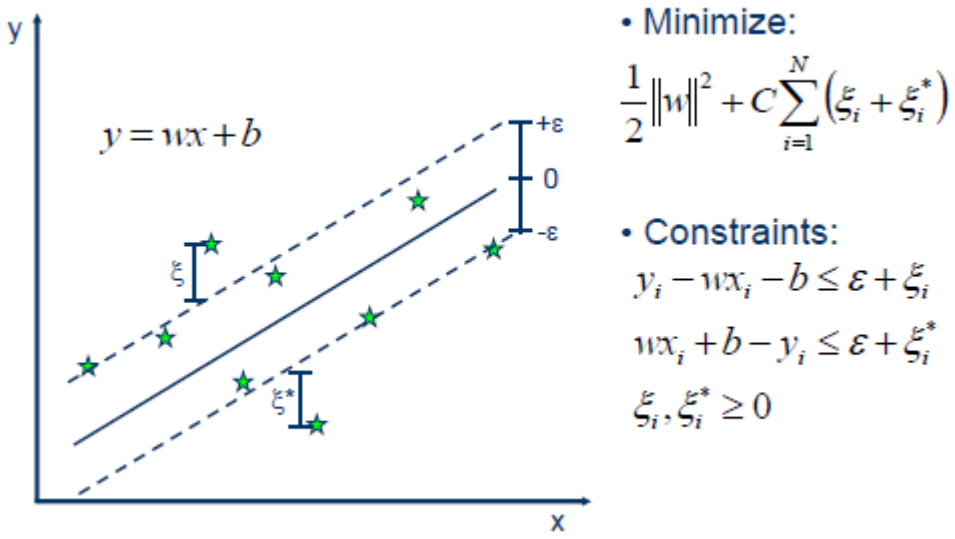
El hiperparámetro C es un número con valor positivo que controla la penalización impuesta a las observaciones que se salen del margen épsilon y previene el *overfitting*, por lo que es un parámetro de regularización. Este hiperparámetro determina el balance entre la rectitud de la función $f(\boldsymbol{x})$ y la magnitud de las desviaciones mayores que ε son permitidas.

La función de costo para este algoritmo se conoce como *ε-insensitive*. Esta función ignora los errores que se encuentran a una distancia menor que ε del valor de respuesta ($y^{(i)}$) al tratarlos como cero. El costo se mide de acuerdo con la distancia entre el valor observado $y^{(i)}$ y el margen ε. Se expresa matemáticamente:

$$L_\varepsilon = \begin{cases} 0 & si\ |y - f(x)| < \varepsilon \\ |y - f(x)| - \varepsilon & si\ |y - f(x)| \geq \varepsilon \end{cases}$$

Ahora, abordaremos el problema dual de la regresión con SVM. Nos saltaremos algunos pasos porque son muy similares a la clasificación. Lo que se debe tomar en cuenta es que esta vez existen dos restricciones en el problema, por lo que se requieren de dos multiplicadores de Lagrange: α y α^*. En este caso, todos los ejemplos dentro del margen cumplen con $\alpha^{(i)} = 0$ y $\alpha^{*(i)} = 0$. Si ninguno de los dos multiplicadores de Lagrange es cero, el ejemplo de entrenamiento es un vector de soporte.

Ilustración 19: Regresión SVM con holgura



Gráfica recuperada de (MathWorks, s.f.).

Evitando la formulación del problema dual para regresión lineal, se entrará de lleno a la formulación dual para regresión no lineal, es decir, con el uso de *kernels*:

$$\mathcal{L}(\hat{\boldsymbol{w}}, \hat{b}, \alpha, \alpha^*) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}(\alpha^{(i)} - \alpha^{*(i)})(\alpha^{(j)} - \alpha^{*(j)})\, K(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) + \varepsilon\sum_{i=1}^{m}(\alpha^{(i)} + \alpha^{*(i)}) - \sum_{i=1}^{m} y^{(i)}(\alpha^{(i)} - \alpha^{*(i)})$$

$sujeto\ a$:

$$\sum_{i=1}^{m}(\alpha^{(i)} - \alpha^{*(i)}) = 0,$$

$$0 \le \alpha^{(i)} \le C,$$

$$0 \le \alpha^{*(i)} \le C\ para\ i = 1,2,\dots,m$$

Y de igual forma, el vector de pesos se presenta como una combinación lineal del conjunto de entrenamiento:

$$\widehat{\boldsymbol{w}} = \sum_{i=1}^{m} \left(\alpha^{(i)} - \alpha^{*(i)}\right) \boldsymbol{x}^{(i)}$$

La función de decisión para este caso es la siguiente:

$$h_{\widehat{\boldsymbol{w}},b}\left(\phi\left(\boldsymbol{x}^{(n)}\right)\right) = \sum_{i=1}^{m} \left(\alpha^{(i)} - \alpha^{*(i)}\right) K\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(n)}\right) + b$$

$sujeta\ a$:

$$\alpha^{(i)}\left(\varepsilon + \xi^{(i)} - y^{(i)} + f\left(\boldsymbol{x}^{(i)}\right)\right) = 0,$$

$$\alpha^{*(i)}\left(\varepsilon + \xi^{*(i)} + y^{(i)} - f\left(\boldsymbol{x}^{(i)}\right)\right) = 0,$$

$$\xi^{(i)}\left(C - \alpha^{(i)}\right) = 0 \quad y \quad \xi^{*(i)}\left(C - \alpha^{*(i)}\right) = 0 \ \ para\ i = 1,2,\ldots,m$$

Por último, en esta sección se hablará de los hiperparámetros en la regresión SVM y sus efectos. El primer hiperparámetro es ε, como ya se mencionó, determina el ancho del margen entorno a la curva de regresión. El segundo hiperparámetro es C. Un valor pequeño de C provoca un margen más amplio, lo que puede representar un sobreajuste. Un valor más grande de C genera un margen más estrecho, pero tiene una mayor cantidad de violaciones de margen. Por último, el hiperparámetro γ es exclusivo del *kernel* de la función de base radial gaussiana. Incrementar el tamaño de γ resulta en una curva de campana más estrecha, por lo que la frontera de decisión se vuelve más irregular. Un valor pequeño de γ genera curvas de campana más amplias, por lo que es poco influida por ejemplos particulares. De esta forma, si el modelo sufre de *overfitting*, se debe reducir el valor de γ; si sufre de *underfitting*, se debe aumentarlo.

2.4 Pronóstico de la Carga Total de una Planta de CC usando ML

En este apartado se revisará el estudio particular de Tüfekci, quien empleó diferentes técnicas de ML para predecir la carga eléctrica total de una planta de CC en Turquía. El artículo es muy basto, debido a la gran cantidad de algoritmos que comparó y los resultados que obtuvo fueron bastante favorables. Cabe mencionar que éste es el único meta-estudio (empleando diferentes algoritmos de ML) que existe para el pronóstico de la carga eléctrica total de una planta generadora.

Para poder hacer análisis de sistemas precisos con enfoque termodinámicos, se deben hacer una gran cantidad de supuestos, de modo que estos supuestos equilibren la imprevisibilidad de la solución. Sin estos supuestos, los análisis termodinámicos de aplicaciones reales se componen de miles de ecuaciones no lineales, cuyas soluciones son casi imposibles de calcular, o bien,

requieren de una carga computacional muy alta acompañada de un alto tiempo. Por lo que, para evitar estos problemas, se han empleado enfoques de ML como alternativas, particularmente para analizar sistemas con patrones de entrada-salida arbitraria (Tüfekci, 2014).

En el artículo se resume el uso de varias técnicas de ML en una planta de CC con dos turbinas de gas, una de vapor y dos sistemas de recuperación de calor. El pronóstico de la carga eléctrica total de una planta de generación ha sido considerado como un problema real crítico dentro del modelado con ML. El predecir la carga eléctrica total de una planta con capacidad instalada definida es importante para su operación económica y su eficiencia. También es útil para lograr maximizar el ingreso que se tiene por mega watt hora (MW h). La confiabilidad y sustentabilidad de una turbina de gas depende altamente del pronóstico de su generación de energía, sobre todo si se tienen restricciones de alta rentabilidad y responsabilidades contractuales.

Por esta razón, es necesario elaborar un modelo predictivo confiable capaz de pronosticar la producción de energía neta del día siguiente en intervalos de una hora. Los dos objetivos principales de Tüfekci fueron determinar la mejor combinación de parámetros de la base de datos que tenía disponible para obtener la mayor precisión posible, y luego comparar la precisión los diferentes métodos de regresión que empleó para determinar cuál era el más exitoso al pronosticar la carga eléctrica total de la planta de CC (Tüfekci, 2014).

La generación de energía de una turbina de gas depende principalmente de parámetros ambientales como la temperatura ambiente, presión atmosférica y la humedad relativa; dado que su fluido de trabajo es aire. Por otra parte, la generación de una turbina de vapor se ve directamente influida por la presión de escape de la turbina. Puesto que se trata de que el vapor libere la mayor cantidad de energía posible, reducir la alta presión con la que llega el vapor a la turbina a la menor posible (incluso por debajo de la presión atmosférica), significa que se logrará tener una eficiencia energética más alta.

En la literatura, los efectos de las condiciones ambientales sobre la generación de energía de una planta han sido estudiados ampliamente. Ejemplo de ello son los estudios por separado de Kesgin y Fast, que emplearon redes neuronales artificiales para predecir la generación de energía de una planta y los efectos que tienen en ésta parámetros como la temperatura ambiental, la presión atmosférica, la humedad relativa, y la velocidad y dirección del viento.

Las variables que Tüfekci ocupó para su estudio se presentan a continuación. Todos los datos de las variables de entrada y la respuesta objetivo corresponden al promedio por hora que se midieron en los sensores que colocó alrededor de la planta.

- Temperatura ambiente: Esta variable fue medida en grados Celsius y varía desde 1.81 °C hasta 37.11 °C.
- Presión atmosférica: Esta variable fue medida en milibares y varía de 992.89 a 1033.30 mbar.

- Humedad relativa: Esta variable fue medida en porcentajes, los cuales fueron del 25.56% al 100.16%.
- Presión de escape del vapor: Esta variable fue medida en centímetros de mercurio y varía de 25.36 a 81.56 cm Hg.
- Carga eléctrica total: La carga eléctrica total es la variable objetivo, fue medida en mega watts y varía de 420.26 a 495.76 MW.

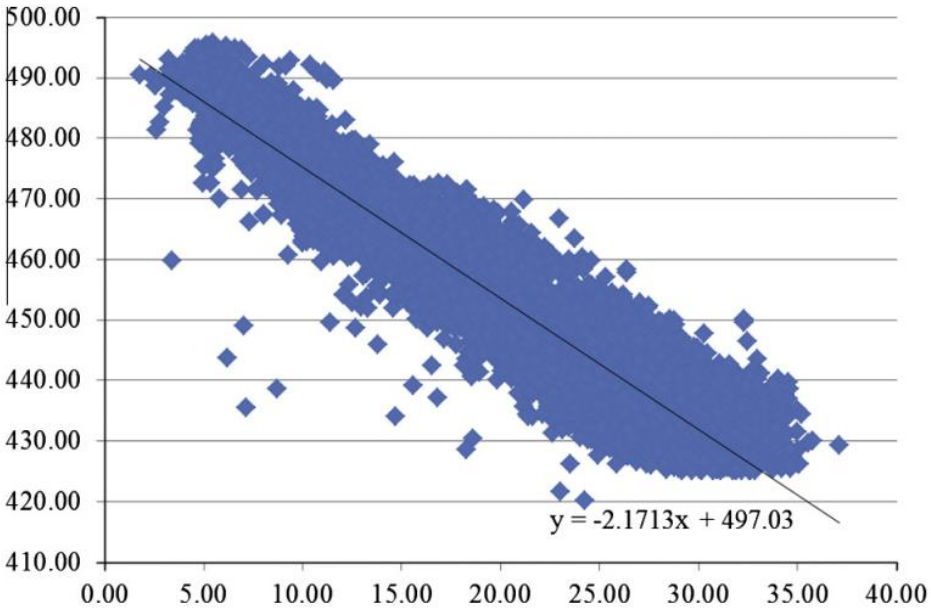
Lo primero que se tiene que notar en la carga eléctrica total es que la capacidad instalada de la planta es de 480 MW, pero el rango de valores que presentó la planta a su máxima capacidad no gira entorno a ese valor, por lo que se deben apreciar los efectos de las condiciones ambientales en el sistema.

Ahora, se procederá a ver los efectos que se detectaron en el artículo en cuanto a las condiciones ambientales sobre la carga eléctrica total. Primero, se menciona que la presión atmosférica y la humedad relativa no tuvieron una correlación suficientemente fuerte con la variable de salida para hacer una predicción individual. Sin embargo, cuando las demás variables permanecen constantes, se tiene una relación directamente proporcional entre estas variables y la variable de salida. Se continuará con una evaluación más detallada de cada variable.

- El efecto de la temperatura ambiente en el desempeño de las plantas de generación es el más estudiado (como en los artículos de Arrieta, De Sa y Erdem). Esto se debe a que parece ser el más significativo. En la base de datos empleada por Tüfekci, mostró una correlación de -0.95 con la variable de respuesta (Ilustración 20). Esta relación inversa proviene de la disminución en la densidad del aire de entrada en la turbina con temperaturas más bajas.
- El efecto de la presión atmosférica es el segundo más influyente (estudiado también por Arrieta). No tiene una correlación lo suficientemente fuerte (0.57) para hacer predicciones por su cuenta, pero cuando otros factores permanecen constantes, mantiene una relación proporcionalmente positiva con la carga total (Ilustración 21). Esto se debe a que tiene una influencia similar a la de la temperatura con el aire de entrada en la turbina de gas, en la cual modifica su densidad.
- Cuando otras variables permanecen constantes, el efecto que la humedad relativa tiene sobre la carga total es proporcionalmente positiva (estudiado por Arrieta y Lee). Sin embargo, por sí sola no es una variable con correlación fuerte (0.39) respecto a la variable de salida. Una humedad relativa más grande incrementa la temperatura del gas de escape de la turbina de gas, lo que incrementa la generación de energía de la turbina de vapor (Ilustración 22).
- El efecto de la presión de escape de la turbina de vapor consiste en una relación inversa con la carga eléctrica total de la planta (Ilustración 23). Esto se debe a que, si el vapor sale de la turbina a una menor presión, habrá una mayor cantidad de energía transferida que si

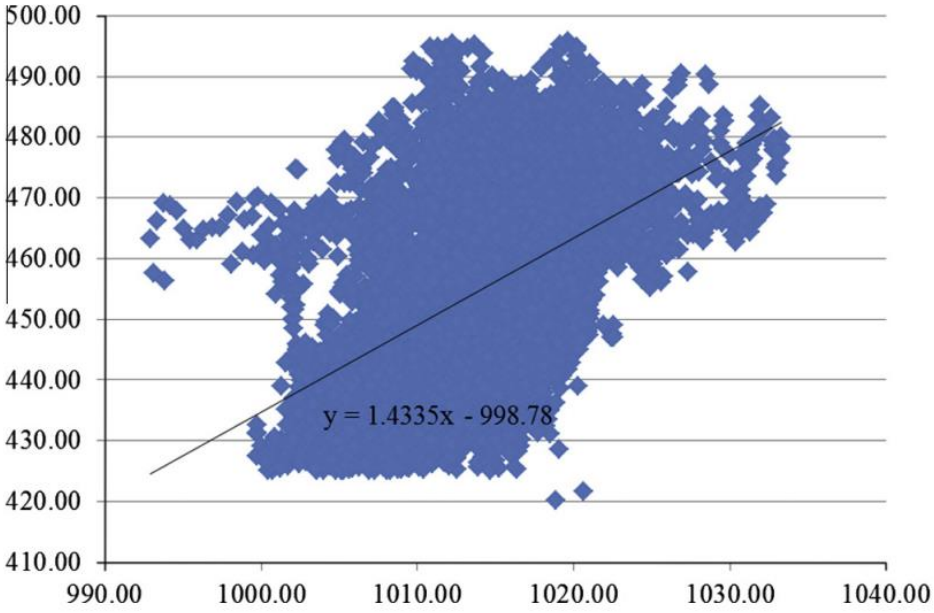
lo hace a una presión mayor. Por esta razón, la correlación que muestra es fuerte (-0.87) respecto a la variable de salida.

Ilustración 20: Diagrama de dispersión de temperatura ambiente (eje x, °C) vs carga eléctrica total (eje y, MW)



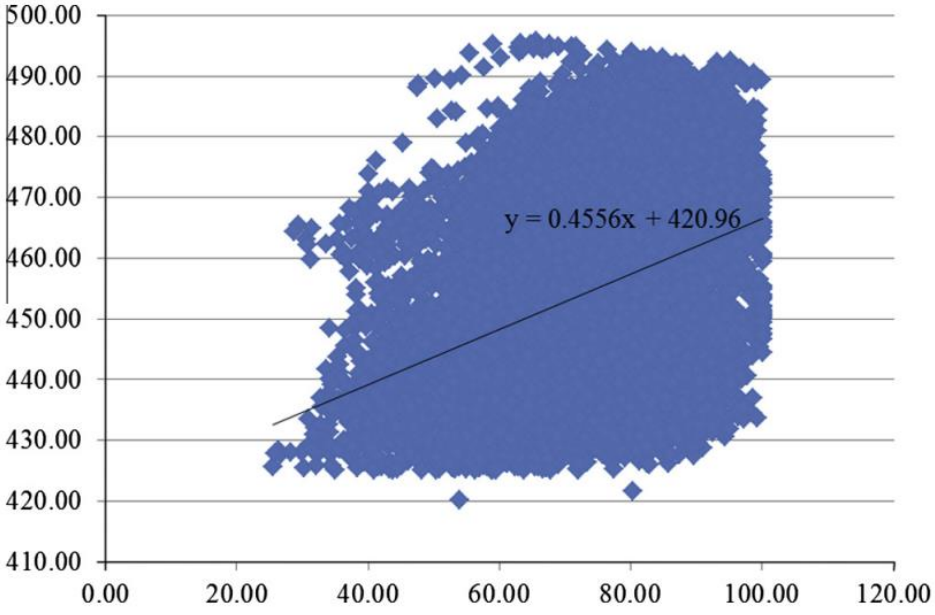
Gráfica recuperada de (Tüfekci, 2014).

Ilustración 21: Diagrama de dispersión de presión atmosférica (eje x, mbar) vs carga eléctrica total (eje y, MW)



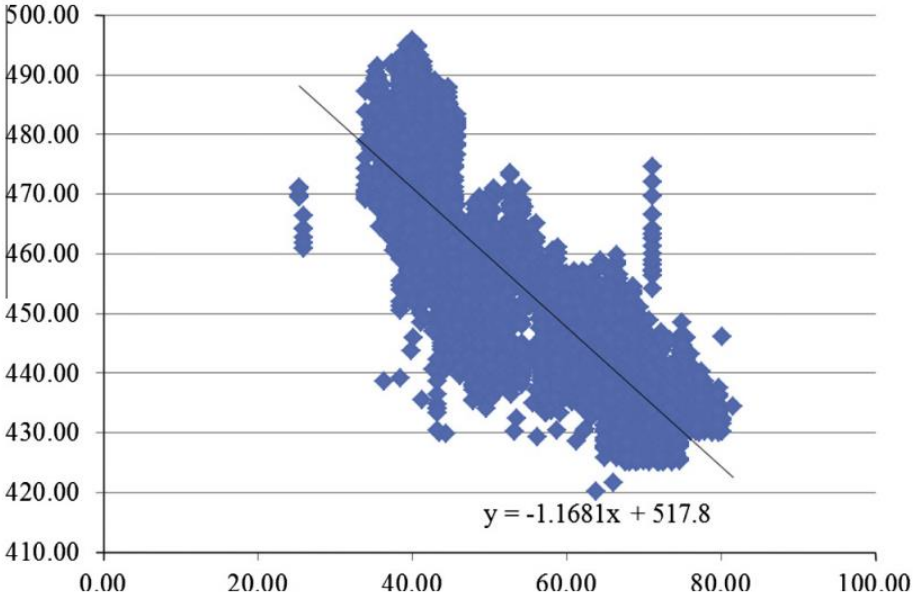
Gráfica recuperada de (Tüfekci, 2014).

Ilustración 22: Diagrama de dispersión de humedad relativa (eje x, %) vs carga eléctrica total (eje y, MW)



Gráfica recuperada de (Tüfekci, 2014).

Ilustración 23: Diagrama de dispersión de presión de escape de la turbina de vapor (eje x, cm Hg) vs carga eléctrica total (eje y, MW)



Gráfica recuperada de (Tüfekci, 2014).

Una vez que se observaron los efectos de las variables de entrada sobre la variable objetivo, se conocerán los distintos algoritmos que Tüfekci usó para hacer los modelos de regresión. En total ocupó quince algoritmos distintos del paquete WEKA, pero debido a que no es el propósito de este trabajo estudiarlos todos, solo se mencionarán. Sin embargo, el autor separó los algoritmos en distintos grupos según sus características. Esta clasificación sí se tocará.

La primera categoría es la de funciones. Esta categoría se basa en el uso de modelos matemáticos. La segunda categoría es la de los algoritmos de aprendizaje flojo. Éstos guardan el conjunto de entrenamiento en la memoria y encuentran los datos relevantes de la base de datos para responder a una pregunta particular. Se basan en la similitud para hacer las predicciones. La tercera categoría son los algoritmos de meta-aprendizaje. Estos combinan diferentes tipos de algoritmos para obtener mejores resultados. La cuarta categoría es la de los algoritmos basados en reglas. Estos usan reglas de decisión para poder hacer la regresión. La última categoría es la de los algoritmos basados en árboles de decisión. Éstos usan estructuras de árboles para hacer las predicciones. En la tabla 2 se muestran todos los algoritmos con sus respectivas categorías y abreviaturas:

Tabla 2: Algoritmos utilizados por Tüfekci para el problema de regresión[13]

Categoría.	**Algoritmo.**	**Abreviatura.**
Funciones	Regresión Lineal Simple	SLR
	Regresión Lineal	LR
	Least Median Square	LMS
	Perceptrón Multicapas	MLP
	Red Neuronal de Función de Base Radial	RBF
	Pace Regression	PR
	Regresión con Vectores de Soporte y *Kernel* Polinomial	SMOReg
Algoritmos de Aprendizaje Flojo	IBk Linear NN Search	IBk
	KStar	K*
	Aprendizaje Ponderado Localmente	LWL
Algoritmos de Meta-aprendizaje	Regresión Aditiva	AR
	Bagging REP Tree	BREP

[13] Algunos algoritmos permanecen en inglés, ya que no hay referencias de sus nombres correspondientes en español.

Algoritmo basado en Reglas	Modelo de Reglas de Árbol	M5R
Algoritmos basados en Árboles de Decisión	Modelo de Regresión de Árbol	M5P
	Árboles REP	REP

Tabla basada en (Tüfekci, 2014).

Una vez que se mencionaron los algoritmos utilizados, se va a describir el proceso que Tüfekci siguió para cumplir con sus objetivos. Primero fue entrenando cada algoritmo con su respectivo conjunto de entrenamiento, aplicando *cross-validation* y luego probando en el conjunto de prueba. Inició con una sola variable (temperatura ambiente) y fue aumentándolas para ver la capacidad predictiva de cada caso. Al final notó que emplear todas las variables dan un mejor resultado que emplear una o unas cuantas variables de entrada. Usó el RMSE y el error absoluto medio (MAE, por sus siglas en inglés) para medir el desempeño de los algoritmos. A continuación, se muestran los algoritmos con mejores resultados (Tabla 3):

Tabla 3: Resultados de los algoritmos utilizados por Tüfekci para el problema de regresión

Categories	Regression methods	AT		AT–V		AT–V–RH		AT–V–AP–RH		Mean	
		MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Functions	LMS	4.285	5.433	3.912	4.968	3.619	4.580	3.621	4.572	3.859	4.888
	SMOReg	4.284	5.433	3.913	4.968	3.620	4.585	3.620	4.563	3.859	4.887
Lazy-learning algorithms	K*	4.260	5.381	3.628	4.634	3.358	4.331	2.882	3.861	3.532	4.552
Meta-learning algorithms	BREP	4.074	5.208	3.035	4.026	2.952	3.934	2.818	3.787	3.220	4.239
Rule-based algorithm	M5R	3.985	5.085	3.418	4.419	3.265	4.217	3.172	4.128	3.460	4.462
Tree-based learning algorithms	M5P	3.982	5.086	3.362	4.359	3.229	4.178	3.140	4.087	3.428	4.428
	REP	4.089	5.229	3.258	4.339	3.214	4.291	3.133	4.211	3.424	4.518

Tabla recuperada de (Tüfekci, 2014).

Como se puede apreciar, el algoritmo que mejor se comportó fue el BREP, tanto en el conjunto con todas las variables de entrada como en el promedio de los resultados. Sin embargo, el error que presentan todos estos algoritmos es muy bajo al compararlo con la magnitud de los datos de la carga eléctrica total. El problema que tiene el BREP es su complejidad, pero debido a que se adapta tan bien al problema, no es un inconveniente.

En general, se nota que los algoritmos que se basan en una medida de similitud (K^*) o, bien, combinan diferentes métodos de aprendizaje (BREP), son los que obtienen mejores resultados, pero a la vez, consumen más recursos computacionales. Luego, los algoritmos basados en reglas y decisiones ponderadas se ubican en segundo puesto. Finalmente, los algoritmos basados en funciones son los que peor se desempeñan, pero, a la vez, son aquellos a los que más se les puede modificar en términos de hiperparámetros y que mejor balancean la complejidad con los recursos computacionales utilizados.

A pesar de tener un modelo tan viable como el BREP, Tüfekci dio un paso más y ocupó una técnica llamada aprendizaje en conjunto. En este se entrenan distintos algoritmos y se obtiene

su modelo. Cuando se presentan nuevos ejemplos, se corre cada algoritmo y se obtiene una respuesta por cada uno. Luego se usa un sistema de voto que, en clasificación, se adopta la respuesta más repetida, mientras que en regresión se promedian los resultados ya sea con una ponderación preestablecida, o bien, con el mismo peso para cada método.

Los algoritmos que se ocuparon fueron los mejores siete por su desempeño individual. Lo que se puede notar es que esta técnica no arrojó los resultados esperados, debido a que el BREP tuvo un mejor desempeño por sí solo que la combinación con los demás. Así que no resultó conveniente usar esta técnica, puesto que se requiere de una mayor cantidad de recursos computacionales y los resultados no mejoran en general. En la tabla 4 se muestran dichos resultados.

Tabla 4: Resultados del aprendizaje en conjunto

Categories	Regression methods	AT		AT–V		AT–V–RH		AT–V–AP–RH		Mean	
		MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
The best single method	BREP	4.074	5.208	3.035	4.026	2.952	3.934	2.818	3.787	3.220	4.239
Voting first 2 methods	BREP, M5P	4.000	5.111	3.147	4.121	3.027	3.971	2.911	3.848	3.271	4.263
Voting first 3 methods	BREP, M5P, M5R	3.986	5.091	3.213	4.185	3.085	4.023	2.972	3.904	3.314	4,301
Voting first 4 methods	BREP, M5P, M5R, REP	3.989	5.095	3.217	4.190	3.036	3.980	2.948	3.886	3.298	4.288
Voting first 5 methods	BREP, M5P, M5R, REP, K*	4.003	5.099	3.202	4.163	3.052	3.985	2.855	3.783	3.278	4,258
Voting first 6 methods	BREP, M5P, M5R, REP, K*, SMOReg	4.001	5.095	3.248	4.212	3.091	4.015	2.912	3.830	3.313	4,288
Voting first 7 methods	BREP, M5P, M5R, REP, K*, SMOReg, LMS	4.012	5.108	3.303	4.270	3.137	4.046	2.977	3.887	3.358	4.328

Tabla recuperada de (Tüfekci, 2014).

Finalmente, nos queda hablar de la generalización del estudio, es decir, la aplicación que tiene en escenarios distintos al del autor. En este caso, se ocuparon datos recolectados durante seis años en una planta de Turquía, lo cual es una muestra bastante significativa de la población. El autor no modificó los hiperparámetros de los modelos por querer evitar el *overfitting* de los mismos, aun cuando pudo haber obtenido mejores resultados y regularizar los modelos para evitar el *overfitting*. También, para que la presunción de generalización tenga efecto en nuevos ejemplos, éstos deben tener la misma distribución de probabilidad subyacente. Por lo que la información debe ser generada con el mismo o un proceso similar al que se ocupa para hacer la primera base de datos. Al emplear un modelo para una planta distinta, se deben corroborar que los estadísticos sean similares a los de la base de Tüfekci para ocupar los modelos directamente. Si esto no ocurre, debido a condiciones totalmente diferentes, se debe generar una nueva base de datos, aunque se puede ocupar su metodología.

Dado los resultados que se obtuvieron, Tüfekci alienta que para una aplicación similar se empleen las cuatro variables que le funcionaron a él y probar en primer lugar los algoritmos que mejor se desempeñaron en su base. El modelo que Tüfekci desarrolló es usado en la planta de generación de donde obtuvo los datos inicialmente. Se usa para predecir la producción de energía del día siguiente para cada hora del día. Las condiciones que se emplean son las pronosticadas por el instituto meteorológico local. Se menciona que, para mejorar las variables de entrada, se debe desarrollar otro modelo que pronostique de mejor forma las condiciones ambientales específicas

de la planta y no del área en que se encuentra. Así como ampliar el modelo para plantas de distinta naturaleza que las plantas de ciclo combinado.

Con esto damos fin al marco teórico, en donde se estudiaron la composición de una planta de ciclo combinado, las bases sobre las cuales se cimienta Machine Learning, el caso específico de las SVM's y el uso de ML para el pronóstico de la carga total de energía de una planta de ciclo combinado. Ahora, se dará paso al caso de estudio, en donde se empleará gran parte de estos conceptos para desarrollar un modelo de pronóstico.

III. Presentación de Caso

3.1 Introducción y Presentación de la Base de Datos

Para iniciar este capítulo, se abordará de nuevo el objetivo de este trabajo:

> "Desarrollar una metodología centrada en la obtención de un modelo de pronóstico capaz de realizar estimaciones altamente exactas de la carga eléctrica total de una planta de generación de ciclo combinado. Esto mediante el uso del algoritmo inteligente conocido como Máquinas de Vectores de Soporte (SVM's) y la medición de las condiciones ambientales del lugar de operación, de modo que la planta pueda ofrecer, a la red de distribución eléctrica nacional, la cantidad de energía que verdaderamente es capaz de producir en determinado momento y, por tanto, disminuir sus pérdidas.

De esta forma, lo que se plantea es finalizar con un modelo que permita mejorar la planificación de la planta de CC, con el objetivo de que la energía que genera se convierta siempre en una fuente de ingreso, y no en un desperdicio o en causa de una multa. Estos dos efectos negativos son casos de interés para el mercado de corto plazo. En este mercado, los pronósticos de generación se envían al CENACE máximo una semana (MDA) o unas horas (MTR) antes de que se asigne la generación a cada planta para determinada hora del día. Si el pronóstico mandado es erróneo pueden ocurrir dos situaciones:

- Si el pronóstico es menor a la energía que verdaderamente se generó, la energía adicional muy seguramente se desperdiciará. Existen dos razones por las que esto pasará, primero porque el contrato convenido con el Sistema Eléctrico Nacional no pagará más energía que la asignada, aun cuando exista una mayor capacidad que la prometida; y, segundo, debido a la incapacidad de almacenar esa energía por un largo periodo de tiempo. Esto significa una pérdida por oportunidad, ya que, de tener un pronóstico más exacto en esta situación, las posibilidades de que sea comprada más energía aumentan, y así los ingresos serían mayores. Ahora, también es cierto que los excedentes se pueden buscar colocar en otro sitio dentro del mercado eléctrico, sin embargo, depender de esta situación aumenta considerablemente el riesgo de la operación, dada la incertidumbre asociada a que este evento se llegue a concretar.
- Si el pronóstico es mayor a la energía que verdaderamente se generó y se le asigna tal cantidad de energía, la planta estará incumpliendo con el contrato y deberá pagar multas. Estas multas se asocian a los gastos adicionales que tiene el CENACE por contar con menos energía que la prevista. Hay que aclarar que una cosa es que el CENACE compre energía en el mercado de corto plazo por cambios en su pronóstico de la demanda (del cual ellos son totalmente responsables) y otra es que tenga que comprarla porque algún generador no cumplió con sus responsabilidades. En este caso, imaginemos que una planta envía su pronóstico para el MDA, y a la hora de entregar la energía que se le asigno es

incapaz de hacerlo. El CENACE, en este caso, recurrirá al MTR, lo cual contiene el costo de operación de este mercado y la diferencia de precio en la energía que tiene que pagar comparado con el MDA. Las multas o pagos con energía en otros periodos son las sanciones que la planta que incumplió debe saldar, por lo que la energía que genera no es 100% un ingreso, sino que le puede generar gastos.

Ahora, es muy importante mencionar que el CENACE asigna la energía mediante un modelo de programación lineal, dentro del cual consideran muchos factores para decidir cómo repartir la generación de energía que requiere en determinado instante. Obviamente, la oferta de cada planta es uno de los insumos más importantes, pero también se toma en cuenta la confiabilidad que cada planta tiene conforme su historial, por lo que incumplir reiteradamente se penalizará con una menor demanda hacia dicha planta.

Si cambiamos un poco el esquema, al tomar a los mercados de mediano y largo plazo, tener un pronóstico certero a corto plazo también resulta muy importante. Estos mercados operan en contratos de 3 a 15 años y se basan en la capacidad instalada de la planta. Sin embargo, si existen factores que impidan llegar siempre a ese valor, tener un pronóstico basado en dichos factores resulta indispensable para poder generar estrategias con las cuales se puedan cubrir los términos del contrato. A la vez, si se pronostica con anticipación una generación mayor a la que el contrato establece, se puede buscar aplicar al mercado de corto plazo y aprovechar esa energía.

Se mencionó en el marco contextual que los generadores están obligados a proporcionarle a la Unidad de Vigilancia del Mercado la capacidad total de que disponen para poder realizar las asignaciones energéticas correspondientes. Para el caso de las plantas de CC, al ser un generador firme y despachable, se le exigirá siempre una cantidad de energía un poco menor a su capacidad instalada. No obstante, como veremos más adelante en la base de datos, la generación verdadera de estas plantas puede ser muy variable.

La propuesta de este trabajo es emplear SVM’s para predecir la carga eléctrica total de una planta de CC. Las razones por las cuales se escogió a este algoritmo provienen de las características del sistema que se analiza. Las plantas de CC son sistemas termodinámicos muy complejos, altamente no lineales y cuyo modelado tradicional (con ecuaciones diferenciales) resulta demandante en tiempo y computacionalmente; y a veces no se puede encontrar una solución con este enfoque. Para una industria que requiere de actualizaciones constantes, por ser sistemas dinámicos, este tipo de soluciones no son adecuadas. El uso de ML facilita resolver muchas de estas complicaciones, en primer lugar, es capaz de lidiar con una cantidad enorme de datos. Las plantas de CC generan datos a cada instante, por lo que el uso de técnicas convencionales (análisis de serie de tiempo, modelos de regresión lineal múltiple, entre otros) no es lo más recomendable para el análisis de estos datos. Además, en las plantas de CC los factores que influyen en la generación de energía mantienen relaciones profundas que son muy difíciles de detectar.

Dicho esto, si volvemos al último capítulo del marco teórico, en su investigación, Tüfekci, dividió todos los algoritmos que empleó en cinco categorías dependiendo de su estructura interna (Tabla 5):

Tabla 5: Fragmento de la Tabla 3

Categories	Regression methods	AT–V–AP–RH	
		MAE	RMSE
Functions	LMS	3.621	4.572
	SMOReg	3.620	4.563
Lazy-learning algorithms	K*	2.882	3.861
Meta-learning algorithms	BREP	2.818	3.787
Rule-based algorithm	M5R	3.172	4.128
Tree-based learning algorithms	M5P	3.140	4.087
	REP	3.133	4.211

Tabla recuperada de (Tüfekci, 2014).

Como se puede apreciar, en la categoría de funciones (que son algoritmos basados en modelos), las SVM's son el algoritmo que menor medida de error presentan. Los algoritmos basados en modelos tienen una ventaja muy clara sobre las demás categorías y es que son interpretables. Mientras unos algoritmos simplemente memorizan los puntos con los que se les entrena y otros obtienen mejores resultados a partir de la combinación de distintos algoritmos, los algoritmos basados en modelos tienen una estructura muy clara a partir de la cual se pueden sacar conclusiones inmediatamente. Las SVM's se basan en hiperplanos (ya sea para clasificar o para modelos de regresión), por lo cual, con el *kernel* adecuado, es posible extraer este hiperplano de vuelta y observar cuál es el efecto de cada variable de entrada sobre el modelo. Esa es la primera razón por la que se escogieron las SVM's.

La segunda razón es que tienen una cantidad de parámetros disponibles muy grande, por lo cual los modelos se pueden personalizar para cada caso en particular. Se pueden elegir distintos *kernels*, que a la vez contienen distintos parámetros internos que permiten mejorar el desempeño del modelo; a la vez de que se puede elegir entre resolver un problema de clasificación o uno de regresión. Todo esto hace de las SVM's un algoritmo altamente versátil.

Sin embargo, la propiedad que más impacta al momento de decidir usar dicho algoritmo en este trabajo es su capacidad para adaptarse a espacios no lineales. Como se mencionó anteriormente, los sistemas termodinámicos presentan una complejidad muy alta, gran parte de ésta proviene de su no linealidad. Las SVM's son, en esencia, algoritmos lineales n-dimensionales, sin embargo, los *kernels* les proporcionan transformaciones no lineales en espacios dimensionalmente más grandes en donde los conjuntos de datos se pueden comportar linealmente. Otras técnicas como el análisis de componentes principales (PCA, por sus siglas en inglés) se encargan de hacer lo mismo. Lo que diferencia estas dos técnicas es el *truco del kernel*. Mientras que el PCA hace todos los cálculos para las transformaciones no lineales, el *truco del kernel* sustituye la transformación por un producto punto.

Esto nos lleva al último punto de por qué se eligieron a las SVM's. La eficiencia del *truco del kernel* hace muy ligera la computación del modelo, al tener en cuenta todo lo que se debería de calcular en caso de que no se pudiera hacer dicha simplificación. Por lo que se pueden correr muchos modelos distintos en un periodo de tiempo relativamente corto.

Todo lo anterior son los pros de por qué escoger este algoritmo, pero se podrá argumentar que, cuando Tüfekci hizo su meta-análisis con diferentes algoritmos, existieron diferencias muy grandes entre los errores que presentaron las SVM's y el BREP. Es cierto que la diferencia es bastante significativa y que, también, entre el BREP y las SVM's hay otros algoritmos que podrían ser considerados. Sin embargo, para este trabajo emplearé una metodología diferente a la que Tüfekci ocupó respecto a las SVM's, esto se verá más ampliamente en el siguiente capítulo, pero se puede adelantar que dentro de las diferencias más significativas se encuentran el uso de un *kernel* distinto y la búsqueda de combinaciones de hiperparámetros para mejorar el modelo.

Una vez que hemos examinado el propósito y la herramienta que se empleará en este trabajo, nos queda describir la base de datos que será usada para darle sustento a todo el análisis. Al momento de escribir este trabajo, en México no existe una base de datos que sea similar a la que capturó Tüfekci en Turquía. La metodología que él propuso no es empleada dentro del país por dos razones principalmente. La más impactante es la normatividad, puesto que en la LIE las plantas de CC se consideran despachables y firmes, no es necesario hacer un pronóstico sobre su carga total, puesto que se considera que se puede controlar totalmente la cantidad de energía que generan. Sin embargo, como veremos más adelante esto no es así. Actualmente se tienen medidas de tolerancia para combatir la variación natural de la generación de energía en las plantas de CC, pero la propuesta de tener un modelo de pronóstico altamente exacto resulta más económica que éstas y permite explotar al máximo los recursos que se disponen, al poder tener una mayor certeza cuando se trabaja a la máxima capacidad de la planta o muy cerca de ella. La segunda razón proviene de un ámbito más cultural. México no es un pionero tecnológico, las tecnologías llegan al país una vez que han sido probadas en muchos otros lugares. En este sentido, ML apenas se está insertando en la operación del día a día de los sectores productivos del país. Esta barrera le representa a la nación depender de otras culturas e ir modificando sus operaciones sobre la marcha, en lugar de tener un plan de desarrollo al inicio de los proyectos.

Por las razones anteriores, la base de datos que se ocupará será la que capturó Tüfekci. De tal forma que se ha de aclarar que, aunque los resultados obtenidos no estén destinados hacia una aplicación inmediata, lo que se pretende es presentar una metodología que se pueda ocupar en cualquier planta de CC diferente a la que Tüfekci empleó. Todas estas diferencias se harán notar en el capítulo siguiente. Por lo pronto, comencemos con la descripción de la base.

Tüfekci es muy reservado a la hora de especificar cuál fue la planta en la que tomó los datos. Lo que se puede encontrar al revisar varios de sus artículos y exposiciones es que se ubica en Turquía. Ahora, dicha planta se compone de dos turbinas de gas ABB 13E2 con capacidad nominal de 160 MW cada una, de dos recuperadores de calor de presión dual (aire-vapor) cuya

función es generar vapor a partir del excedente de calor de las turbinas de gas y de una turbina de vapor ABB con capacidad nominal de 160 MW. La suma de todo esto nos genera una planta de CC con una capacidad nominal de 480 MW (Tüfekci, 2014).

La base de datos que generó consiste en cuatro variables de entrada y una variable objetivo, ésta fue recolectada en un periodo de seis años (2006-2011). Cada variable se compone de 9568 datos, tomados mientras la planta estaba trabajando a su capacidad máxima (carga total) a lo largo de 674 días distintos. Las variables de entrada corresponden al valor promedio calculado por hora de ciertas condiciones ambientales, el cual fue medido por los sensores ubicados en la planta estratégicamente por Tüfekci. La variable objetivo es la carga eléctrica total de la planta y corresponde a la carga eléctrica de salida promedio en periodos de una hora (Tüfekci, 2014).

Al iniciar el procesamiento de datos, la base de datos consistía de 674 hojas de cálculo distintas, las cuales contenían datos incompatibles y con ruido. Por lo que se limpió la información removiendo los datos incompatibles (cuando la planta operaba a menos de 420.26 MW) y los datos con ruido (que fueron datos en los que los sensores presentaban una interferencia eléctrica). Finalmente, todas las hojas de cálculo se combinaron, eliminando los datos duplicados y revolviendo los datos cinco veces aleatoriamente (Tüfekci, 2014). Así es como extraigo la base de datos: consiste de un archivo con cinco hojas de cálculo, pero que en realidad son los mismos datos, por lo cual solo emplearé la primera hoja.

Basado en otros escritos sobre el efecto de las condiciones ambientales en el rendimiento de las plantas de CC, Tüfekci eligió ciertas condiciones ambientales y colocó diferentes sensores a lo largo de la planta para capturarlas:

- Temperatura ambiente (AT): Esta variable fue medida en grados Celsius y varía desde 1.81 °C hasta 37.11 °C.
- Presión atmosférica (AP): Esta variable fue medida en milibares y varía de 992.89 a 1033.30 mbar.
- Humedad relativa (RH): Esta variable fue medida en porcentajes, los cuales fueron del 25.56% al 100.16%.
- Presión de escape del vapor (V): Esta variable fue medida en centímetros de mercurio y varía de 25.36 a 81.56 cm Hg.
- Carga eléctrica total (PE): La carga eléctrica total es la variable objetivo, fue medida en mega watts y varía de 420.26 a 495.76 MW.

Como ya se describió en el último capítulo del marco teórico, cada una de estas variables tiene un efecto diferente sobre la carga eléctrica total. A continuación, se muestra una descripción estadística más a detalle de la base de datos (Tabla 6). El valor máximo y mínimo de cada variable nos puede dar una idea de cómo se distribuyen los datos, pero se requieren de más datos para tener una verdadera apreciación de ellos.

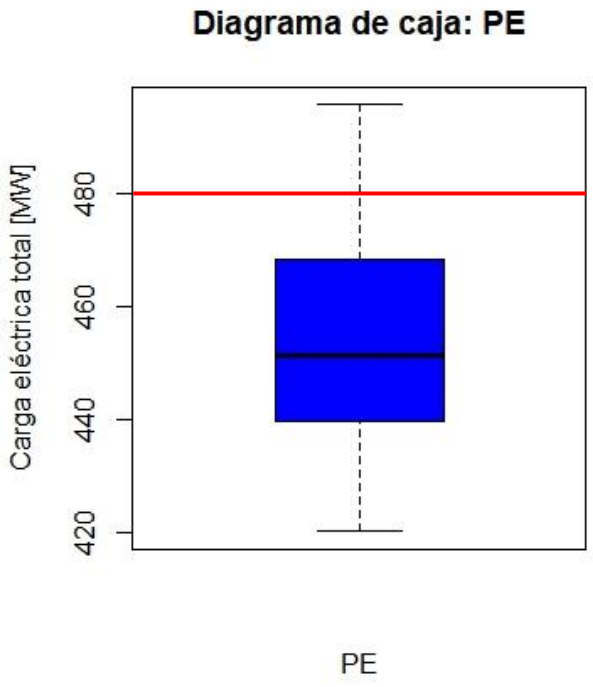
Tabla 6: Estadísticos de la base de datos

	AT	V	AP	RH	PE
Mínimo	1.81	25.36	992.9	25.56	420.3
1er Cuartil	13.51	41.74	1009.1	63.33	439.8
Mediana	20.34	52.08	1012.9	74.97	451.6
Media	19.65	54.31	1013.3	73.31	454.4
3er Cuartil	25.72	66.54	1017.3	84.83	468.4
Máximo	37.11	81.56	1033.3	100.16	495.8
Varianza	55.54	161.49	35.27	213.17	291.28

Elaboración propia con datos de (Tüfekci, 2014).

En este caso, la variable que más nos interesa es PE (la carga eléctrica total), pues es nuestra variable objetivo y, de acuerdo con la regulación actual en México, no debería variar mucho de su capacidad nominal de 480 MW. Para apreciar cómo se comporta se presenta el siguiente diagrama de caja (Ilustración 24):

Ilustración 24: Diagrama de caja de la carga eléctrica total de la planta de CC



Elaboración propia con datos de (Tüfekci, 2014).

Como se puede observar, la planta no está ni cerca de trabajar a su capacidad nominal (marcada por la línea roja). Al trabajar su máxima capacidad, generalmente, no logra llegar a los 480 MW, por ello es que subestimar los efectos que tienen las condiciones ambientales sobre la

eficiencia de la planta de CC es un error. El rango que presenta PE es muy amplio y no es aleatorio, sino que es causado por los factores ambientales. Claro que el hecho de que se aleje de la capacidad nominal puede indicar el desgaste de la planta, sin embargo, aún hay puntos en que supera esa capacidad.

Hasta aquí la introducción y descripción de la base de datos. Ahora nos enfocaremos en la metodología seguida para obtener un modelo balanceado y de alta exactitud.

3.2 Metodología

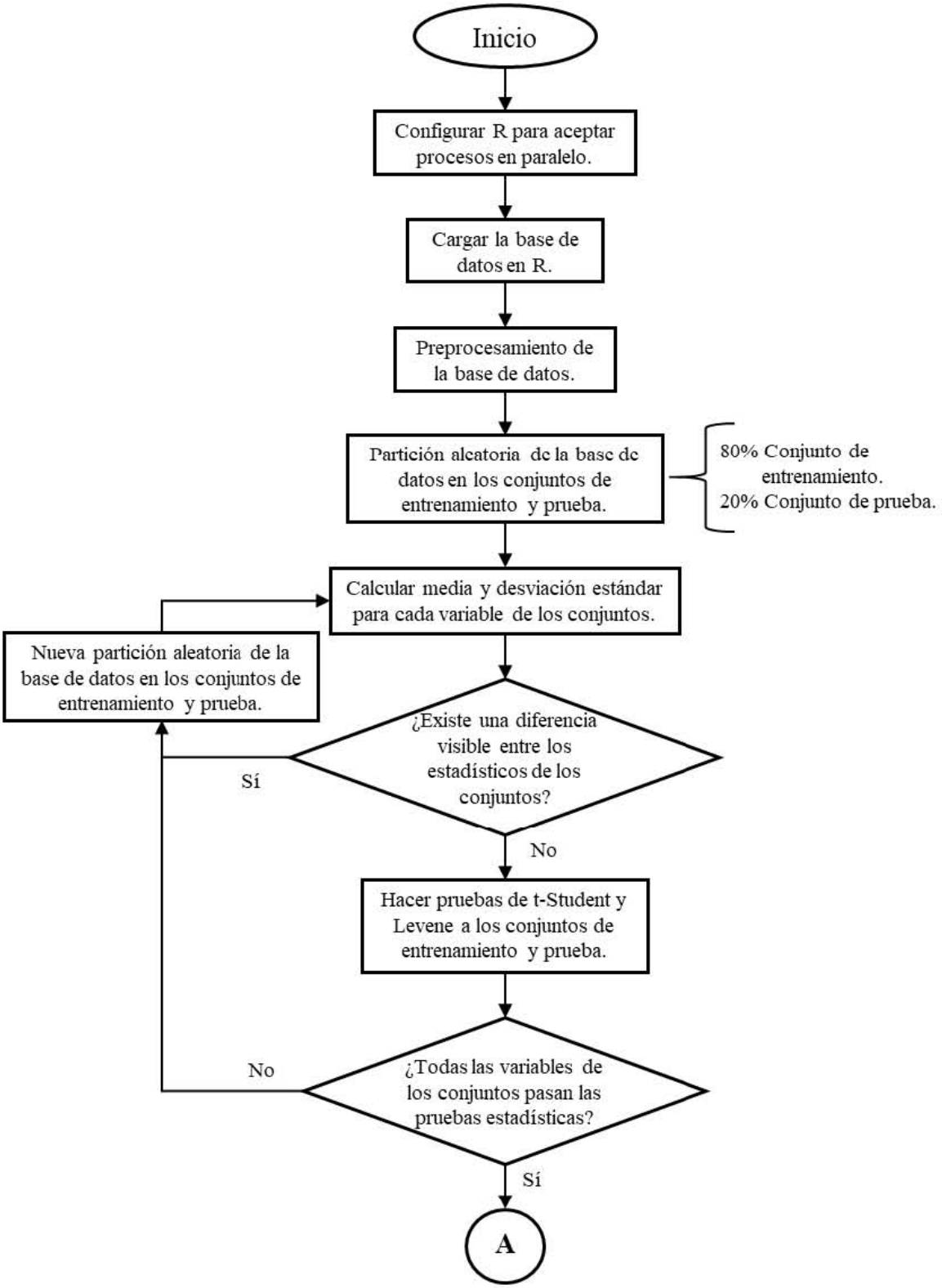
Este capítulo se iniciará haciendo una aclaración. El enfoque de esta metodología es híbrido, en el sentido de que es una combinación entre un enfoque teórico y uno práctico. No se puede considerar puramente teórico debido a que, al iniciar el procedimiento, se tienen muchas suposiciones definidas y, por tanto, no se explora todo el espacio de posibilidades disponibles. También porque existe un paso en particular dentro de la metodología que se aleja un poco de la conceptualización ideal de los procesos de ML. Este paso se denotará al momento de explicar la metodología. Y, por otro lado, tampoco se puede considerar totalmente práctico porque no se persiguen los mejores resultados a toda costa. Se mantiene una base teórica lo suficientemente firme para poder asegurar que los resultados obtenidos cumplen con una buena generalización del problema. Así, se busca un modelo balanceado, pero que, a la vez, tenga una exactitud muy alta.

Lo primero a considerar es el proceso de medición de datos. Para ello, se le recomienda al lector que revise el procedimiento que Tüfekci siguió en su investigación (Tüfekci, 2014). Además, que en el mismo artículo se proporcionaron las referencias hacia otros artículos que igualmente tratan el tema de la medición de condiciones ambientales para plantas de CC. Con estas referencias se puede generar un procedimiento para la recolección de datos de una planta de CC: ubicación de sensores, tipos de sensores, frecuencia de medición de datos, entre otros.

Ahora, comenzaremos con la primera parte de la metodología. La primera acción a realizar una vez que ya se tiene la base de datos es limpiarla. En la sección anterior se presentó cuáles fueron los criterios de Tüfekci para eliminar los datos incompatibles. Los procesos que él realizó son sólo unos cuantos de todos los que existen para el preprocesamiento de datos. Hay tratamientos para datos faltantes, incompatibles, duplicados y disminución de dimensiones. Para una referencia más profunda de estos temas se recomiendan los libros: (Mueller & Massaron, 2016) y (Géron, 2017).

La forma de presentar la metodología será mediante un diagrama de flujo partido en secciones, entre estas secciones se dará una explicación más amplia de los pasos que se reflejan en el diagrama. Es importante aclarar que en el diagrama se verán los pasos principales del proceso, por lo que acciones como llamar librerías y la configuración de parámetros para gráficas no estarán reflejados.

Ilustración 25: Diagrama de flujo de la metodología: Parte 1



Elaboración propia.

Lo primero a extraer de esta sección es que el modelo se desarrollará en el lenguaje de programación R. R es un lenguaje pensado para el análisis de datos desde un punto de vista estadístico, sin embargo, por su índole de *open source*, ha crecido en usuarios y contribuyentes a un ritmo muy alto. Por lo que su evolución natural recayó en ML. En este caso, se ocupa la IDE (plataforma de desarrollo) de RStudio. Estas dos herramientas son libres y bastante potentes.

Ahora, el primer paso que se menciona es configurar R para que pueda aceptar procesos en paralelo. El funcionamiento tradicional de R al realizar cualquier función es emplear un solo núcleo de los disponibles en la computadora. Con las librerías "parallel" y "doParallel" se puede configurar R para realizar diferentes tareas al mismo tiempo en diferentes núcleos del CPU. Las computadoras modernas por lo menos cuentan con dos núcleos en su CPU, por lo que emplear estas funciones puede disminuir a la mitad del tiempo tareas que no sean estrictamente secuenciales. En este caso, se emplea una computadora con las siguientes características:

- CPU: Intel i7-8750H de seis núcleos.
- RAM: DDR4 de 16 GB.
- GPU: NVIDIA GeForce GTX 1050 Ti.

De esta forma, se pueden configurar hasta seis tareas independientes simultáneamente en R. La forma en que R hace esto es generar seis ventanas virtuales de R, una en cada núcleo, y asignarle a cada uno la tarea a realizar. Como se verá más adelante, esta configuración es muy importante para poder llegar a los resultados deseados.

Una anotación importante es mencionar que en lugar de emplear una computación en paralelo se pudieran emplear los núcleos del GPU para realizar los modelos. Estos núcleos están optimizados para hacer tareas de ML. Son una característica de la compañía Nvidia y se les conoce como CUDA. Es posible que el tiempo de cómputo se reduzca usando el GPU, sin embargo, al momento de iniciar este trabajo no se contaba con el equipo actual, por lo que resultaba mejor emplear los núcleos de CPU en paralelo.

Siguiendo con el procedimiento, la continuación es cargar la base de datos a R. La forma más común de hacer esto es importar la hoja de cálculo en la que está la base. Se pueden importar una gran cantidad de tipos de archivo, en este caso es un archivo de Excel (*.xlsx). Como se mencionó en la sección anterior, sólo se importa la primera hoja del archivo de la base de datos, pues las otras cuatro contienen la misma información, sólo que en un orden diferente.

Luego se pasa al preprocesamiento de datos. En este caso, la base de datos fue preprocesada por su generador (Tüfekci), por lo que no tuve que modificarla en nada. Sin embargo, para nuevas bases, este paso es crítico. Como se vio en el capítulo de "Fundamentos de Machine Learning", la calidad de los datos puede exceder la capacidad de los algoritmos complejos: siempre y cuando se tengan datos de calidad, los algoritmos más sencillos pueden igualar el desempeño de algoritmos complejos. También, en este momento me permitiré referenciar el lugar de donde obtuve la base de datos. Esto no fue directamente de Tüfekci, sino

que él permitió el acceso libre a su base mediante el repositorio de la Universidad de California en Irvine (University of California, Irvine, 2014). El "Machine Learning Repository" de esta universidad cuenta con una gran extensión de bases de datos de todo el mundo y de todos los temas (ingeniería, medicina, sociales, etc.). Estas bases se componen de datos reales y la mayor parte de ellas provienen de investigaciones de alto nivel. Para practicar o para tener un marco de referencia, estas bases son muy útiles.

El siguiente paso del proceso es dividir la base de datos en dos conjuntos: el conjunto de entrenamiento y el conjunto de prueba. En el marco teórico se explicó la función de cada uno, el conjunto de entrenamiento será aquel con el cual el algoritmo aprenda de los datos y el conjunto de prueba es un conjunto que se deja fuera de todo el proceso para, al final, medir la generalización del modelo. En este caso, se optó por una partición del 80-20, la cual es la máxima recomendable. Esto se hizo para que, durante la fase de entrenamiento, el algoritmo tenga la mayor cantidad de ejemplos posibles sobre los cuales generar el modelo.

Como se debe recordar, la partición debe ser aleatoria, pues si se le da un orden específico, los conjuntos de datos pueden ser incompatibles. Con la partición de la base se tiene un problema interno, que es el sesgo. Cuando se le priva de ciertos ejemplos, el algoritmo tiende a adaptarse a los ejemplos con los que cuenta. Puede que, en la partición, ciertos ejemplos útiles para el margen de la SVM se queden en el conjunto de prueba, por lo cual no pueda incorporarlos al modelo. De esta forma, se sesga la población y al ser una división aleatoria, no se puede asegurar cuál es el mejor conjunto para entrenar al algoritmo. Nunca se puede asegurar al 100% que, al dividir una población (aunque sea aleatoriamente), sus muestras tendrán las mismas propiedades que ésta.

Sin embargo, sí existen ciertas medidas que se pueden tomar para asegurar la compatibilidad de los conjuntos. La primera es revisar la media y la desviación estándar de cada una de las variables de ambos conjuntos. Al comparar los estadísticos de cada conjunto, no deben ser visiblemente diferentes unos de los otros (propongo más de un 10% de variación). De lo contrario, los conjuntos están sesgados y se debe de rehacer la partición aleatoria. Normalmente, esta prueba es suficiente para quienes se dedican a la ciencia de datos, pero pueden existir combinaciones de datos en los conjuntos que hagan que los estadísticos sean similares, pero que en realidad los conjuntos se distribuyan de una forma muy diferente.

Para eliminar las dudas, lo que se propone es emplear dos pruebas estadísticas que confirmen las similitudes que se pueden encontrar en las medias y desviaciones estándar. La primera prueba es una prueba t de Student para comparar las medias de las variables de los conjuntos. La hipótesis nula de esta prueba es que la diferencia de las medias en dos conjuntos se debe únicamente al ruido natural, por lo que, de no rechazarse esta hipótesis, los conjuntos tienen en realidad la misma media. Esta prueba está diseñada para conjuntos que se distribuyen de forma normal. No obstante, las pruebas clásicas de normalidad se ven alteradas notablemente por el escalamiento en el número de datos, lo que significa que, por poco que se alejen los conjuntos masivos de datos del esquema de una función normal, las pruebas de normalidad asegurarán (con

un error mínimo) que los datos no se distribuyen de esta forma. Pero para este tipo de conjuntos de datos, el teorema del límite central pesa mucho más que tratar de emplear una distribución de probabilidad cualquiera, por lo que la prueba t-Student es adecuada para nuestros conjuntos de prueba y entrenamiento. Al realizar la prueba, si el *p-value* resulta mayor que 0.05, se puede asegurar que las medias de los dos conjuntos son iguales; si resulta menor, la hipótesis nula se rechaza.

La segunda prueba que se realiza es la prueba de Levene. Ésta sirve para determinar homocedasticidad, o bien, que la varianza de dos conjuntos es la misma. A diferencia de la prueba anterior, esta prueba no requiere de normalidad para ser válida. Cuando se realiza la prueba, si el *p-value* resulta mayor que 0.05, se puede asegurar que las varianzas de los dos conjuntos son iguales; si resulta menor, la hipótesis nula se rechaza.

Para este caso, queremos que las hipótesis nulas de ambas pruebas sean verdaderas, de forma que, al no haber distinción entre las medias y varianzas de las variables de los conjuntos de prueba y entrenamiento, se tenga una mayor seguridad de que los conjuntos son compatibles. Si esto no ocurre, se tendrá que dividir de nueva forma la base. Esto se hace para asegurar que la distribución de probabilidad subyacente de los dos conjuntos sea la misma, de lo contrario la hipótesis principal de ML no es válida.

Así, se termina esta primera parte de la metodología, a continuación, se proseguirá con la parte más importante del proceso: el modelo. Cabe mencionar que todo lo que se va a tocar durante este capítulo se hará más claro al observar los resultados en el siguiente capítulo.

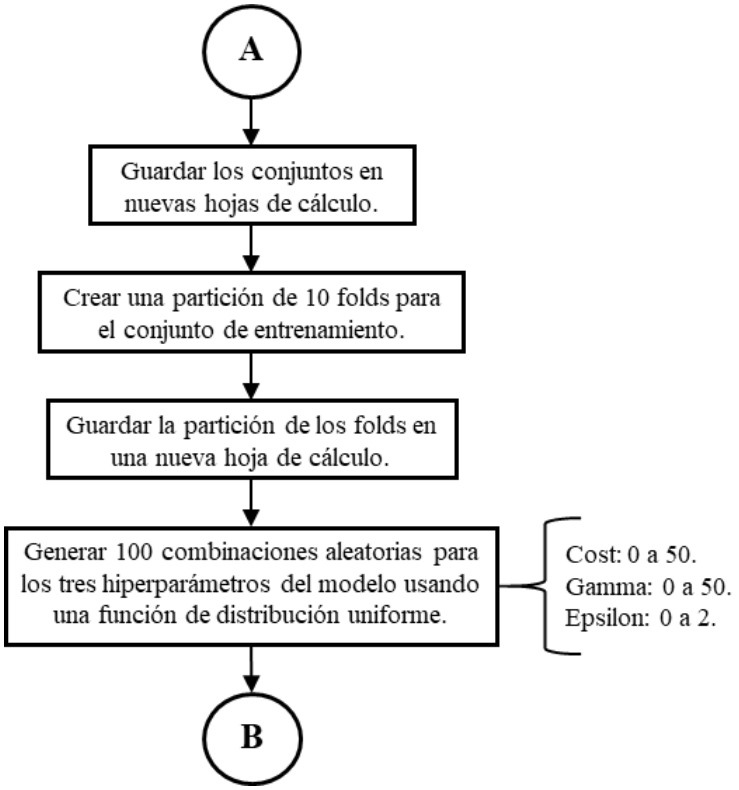
Para generar los modelos SVM's, se empleará la librería e1071 en R. Esta librería fue desarrollada por el grupo E1071 de teoría probabilística de la universidad de TU Wien en Viena. En ella se pueden encontrar diferentes tipos de clasificadores, así como el algoritmo para desarrollar modelos SVM's, ya sea para clasificación o para regresión. En este caso, emplearemos la función svm() con los siguientes parámetros:

- Fórmula: "PE ~ .". Esto significa que queremos un modelo que nos pronostique PE a partir de todas las demás variables de la base de datos. Aquí tomamos las aportaciones de Tüfekci, de modo que ocupamos las cuatro variables de entrada para predecir la variable objetivo, pues en su investigación determinó que era la combinación de variables con mejores resultados.
- Tipo: "eps-regression". Esto le indica al algoritmo que queremos hacer una regresión con el parámetro épsilon. Este tipo de regresión fue el que se estudió en el marco teórico.
- Kernel: "radial". Esto le indica al algoritmo que el *kernel* que emplearemos es una función de base radial gaussiana. Se decidió elegir este *kernel* debido a que es el más flexible dentro de los que están programados en la función. Además, el *kernel* gaussiano es el que mayor uso tiene por su capacidad de adaptarse a espacios de variables con una no linealidad alta. Recordemos que la función de base radial gaussiana tiene la siguiente

fórmula: $K(\boldsymbol{a}, \boldsymbol{b}) = \exp(-\gamma \|\boldsymbol{a} - \boldsymbol{b}\|^2)$. Ésta tiene un solo hiperparámetro modificable: gamma.

Una vez definidos estos parámetros, nos faltaría especificar los hiperparámetros, pero eso se hará más adelante. Por lo pronto continuaremos con nuestro diagrama de flujo:

Ilustración 26: Diagrama de flujo de la metodología: Parte 2



Elaboración propia.

Una vez que se validaron los conjuntos de entrenamiento y prueba con la misma distribución subyacente, lo que se hace es guardar en nuevos archivos estas particiones. El motivo de este paso radica en poder volver a usar esa partición aun cuando se cierre R. Así no se tiene que volver a validar una nueva partición, lo único que se hace es cargar cada conjunto sin necesidad de cargar la base completa de nuevo.

Ya que se tienen los conjuntos de forma independiente, se genera una numeración aleatoria del uno al diez para cada fila del conjunto de entrenamiento. Lo que esto genera es una partición virtual de este conjunto en diez *folds* del mismo tamaño o de tamaños muy cercanos. De esta forma, más adelante se podrá hacer *cross-validation*. Se guarda esta numeración debido a que se ocupará en distintas ocasiones más adelante.

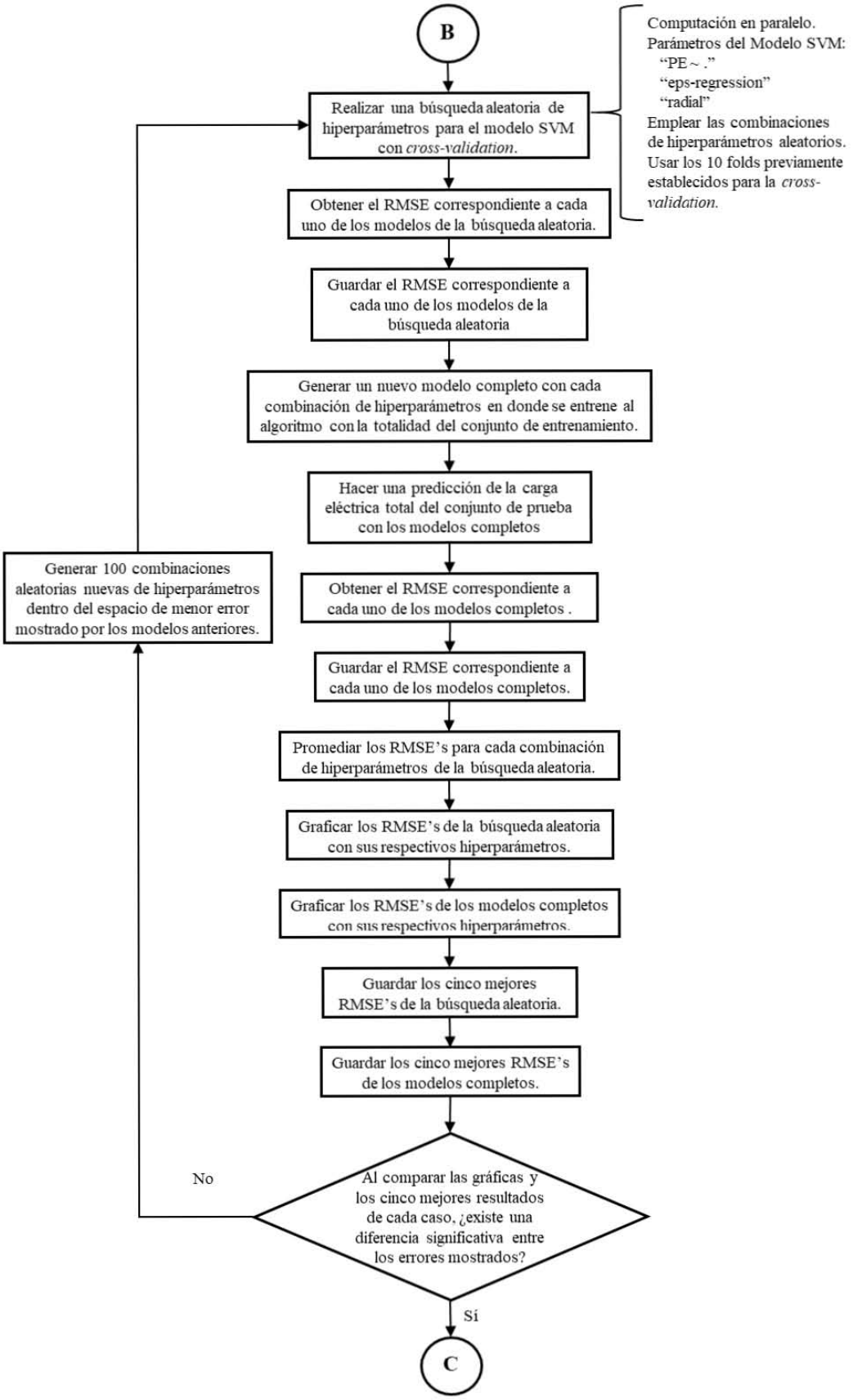
Para terminar la segunda parte de la metodología, se generan aleatoriamente cien combinaciones de hiperparámetros distintas con el fin de encontrar una combinación que resulte en un modelo con desempeño aceptable. Como se recordará, se empleará un modelo SVM con *regresión épsilon* y con la función de base radial gaussiana como kernel, por lo que se tienen los hiperparámetros de épsilon, costo y gamma. En esta primera instancia se abarca un espacio de hiperparámetros muy amplio. Se ha notado que después del valor de 50 para el costo y gamma, los modelos no mejoran mucho en su desempeño, aunque su dominio llega hasta el infinito. En el caso de épsilon, la función svm() únicamente permite valores entre cero y dos.

Ya que se tienen diferentes combinaciones de hiperparámetros, se inicia una búsqueda aleatoria. Como se recordará, una búsqueda aleatoria de hiperparámetros consiste en entrenar muchos modelos de ML con combinaciones diferentes de hiperparámetros con el fin de encontrar el que mejor se desempeña o, bien, la región de hiperparámetros que mejores resultados tiene. Al ser una búsqueda aleatoria, se espera que las combinaciones se distribuyan a lo largo de todo el espacio de hiperparámetros definido, pero también se pueden sesgar hacia un solo sector. Por lo que, si esto ocurre, se deberá generar un nuevo conjunto de combinaciones aleatorias.

Sin embargo, como se mencionó durante el capítulo de “Fundamentos de ML”, lo más conveniente es realizar esta búsqueda de hiperparámetros con *cross-validation*. Esta técnica es útil para garantizar modelos con una generalización mayor, lo cual significa que tendrá resultados muy similares (en cuanto al error) cuando prediga ejemplos que el algoritmo ha observado y cuando prediga ejemplos que no ha observado. *Cross-validation* consiste en separar el conjunto de entrenamiento en *k* particiones iguales. Lo que se hará en este trabajo es dividirlo en los diez *folds* antes mencionados. La manera en que funciona esta técnica es que se toma un *fold* y se separa de todos los demás. Mientras, los otros nueve *folds* se consolidan en un solo conjunto, el cual se usa para entrenar al modelo. Una vez que está listo el modelo, se realiza el proceso de prueba con el *fold* que se separó y se obtiene la medida de error que tuvo el modelo (para este trabajo: RMSE). Y luego se separa otro *fold* y se realiza el mismo proceso hasta que cada *fold* haya sido tomado como conjunto de prueba. En este caso, se tomó diez *folds* porque es el número de particiones más común dentro del medio de la ciencia de datos.

Entonces, se genera una búsqueda aleatoria con *cross-validation* para las cien combinaciones diferentes de hiperparámetros. Suena como una tarea sencilla, sin embargo, cuando ponemos atención a la cantidad de modelos que esto representa, se notará que el total de modelos que corre el programa es de 1000 modelos distintos. Son cien combinaciones de hiperparámetros y por cada una de ellas se tienen diez modelos que se crean para la *cross-validation*. Es aquí donde tiene sentido repartir las tareas asignadas a los diferentes núcleos del CPU. Así, en lugar generar los modelos de forma secuencial, se hace de forma paralela y se divide el tiempo entre el número de núcleos: en este caso, seis.

Ilustración 27: Diagrama de flujo de la metodología: Parte 3



Elaboración propia.

Cuando se termina el proceso de búsqueda aleatoria con *cross-validation* lo que se hace es promediar todos los RMSE's correspondientes a cada una de las combinaciones de hiperparámetros. Con esto se tiene una aproximación al error esperado que puede presentarse cuando se le ingresa un nuevo dato al modelo. Como es de notar, durante el proceso de *cross-validation*, se entrenan diez modelos distintos por cada combinación de hiperparámetros, a todos ellos se les prueba con datos que nunca ha visto antes (el *fold* que es excluido), por lo que se tiene una simulación de lo que ocurriría si al modelo se le alimenta con datos fuera de la base de datos original. De ahí, que el promedio de RMSE's se considere como una medida muy cercana al comportamiento final del modelo.

El proceso descrito hasta ahora es el que pertenece a un ámbito puramente teórico. Lo que se buscaría en este caso sería encontrar la combinación de hiperparámetros que mejor resultado arroje (menor promedio de RMSE). No obstante, cuando esto se hace, el riesgo de caer en el *overfitting* es muy latente. Por esta razón, se decidió que, a la vez de hacer el proceso de búsqueda aleatoria, también se entrenaran modelos con el conjunto de entrenamiento completo. Estos modelos ocupan las mismas combinaciones de hiperparámetros que la búsqueda aleatoria. Una vez entrenados estos modelos, se prueban con el conjunto de prueba y se obtiene su medida de error. Esto significa que el programa, en lugar de estar haciendo los 1000 modelos que se mencionaron antes, está haciendo 1100 modelos distintos. Por esta razón se aseguró anteriormente que el enfoque de esta metodología era híbrido, pues se trata de balancear los modelos de modo que tengan un sustento teórico suficiente, pero evitando que caigan en el *overfitting* al introducir un contrapeso al método tradicional. Esta adición es muy común en la práctica de la ciencia de datos.

Cuando se tienen los errores promedios de la búsqueda aleatoria y los errores de los modelos que ocupan el conjunto de entrenamiento completo (modelos completos), lo que se hace es graficarlos respecto a los hiperparámetros a los que corresponden. Se observa que, entonces, se tendrán cuatro dimensiones que graficar (tres hiperparámetros y la medida de error). Como se sabe, los humanos somos incapaces de representar más de tres dimensiones en ejes, por lo que se tiene que buscar una solución para poder visualizar los datos. A lo que se llegó es a desarrollar una gráfica en que los ejes se compongan de los hiperparámetros e introducir a la medida de error con un código de colores. De esta forma, se pueden apreciar las áreas en las que los hiperparámetros generan un modelo con menor error. Al mismo tiempo, se extrae de los promedios de RMSE's y del error de los modelos completos, las cinco combinaciones de hiperparámetros que menor error presentan.

Finalmente, lo que se propone es hacer una comparación visual de las dos gráficas (los modelos de la búsqueda aleatoria y los modelos completos) y de las combinaciones con menor error. Si los resultados de ambos son muy parecidos, es decir, las áreas en que los hiperparámetros muestran un menor error se corresponden; se tendrá que hacer una nueva búsqueda aleatoria, pero ahora con un espacio de hiperparámetros reducido, enfocado en el área de mejor desempeño. Para determinar estas áreas se deben encontrar los patrones que existen en las gráficas y las

combinaciones con menor error. Este proceso quedará más claro una vez que se revisen los resultados. El proceso de la búsqueda aleatoria debe continuar hasta que el espacio se reduzca tanto que se aprecien diferencias entre los modelos generados por búsqueda aleatoria y con el conjunto de entrenamiento completo.

En caso de que los resultados de la búsqueda aleatoria y los modelos completos no coincidan se proseguirá a una búsqueda de malla en el área en que los dos conjuntos sigan compartiendo los mismos valores. Esto tiene el objetivo de que no se haga un *overfitting* hacia el conjunto de entrenamiento o hacia el de prueba. Se propone elegir un espacio de hiperparámetros en donde exista una variación máxima entre el promedio de RMSE's y los modelos completos del 10% para casos con RMSE's altos. Mientras que, para casos con RMSE's bajos, un 5% ya se considera una diferencia notable. Este criterio está basado en la experiencia propia, sin embargo, se puede elegir otro criterio dependiendo del caso que se presente. La discrepancia en los criterios se debe a que la magnitud del RMSE incrementa o disminuye conforme a la magnitud de la variable sobre la que se mide el error.

Ahora bien, se mencionó que el siguiente paso de la metodología es realizar una búsqueda de malla. Este tipo de búsqueda se diferencia de la búsqueda aleatoria en que se ubica en un espacio más definido dentro de las combinaciones de hiperparámetros. Para realizar una búsqueda de malla, primero se deben predefinir *n* diferentes valores para cada uno de los hiperparámetros. La característica que tienen estos valores es que, para un mismo hiperparámetro, los valores que se eligen forman parte de una secuencia. Normalmente la secuencia que se elige es una progresión aritmética, las cuales comparten una misma diferencia entre todos los valores; pero se puede elegir una diferente. También, cabe aclarar que el número de valores para cada hiperparámetro puede ser distinto, por ejemplo, tener *l* valores para el costo, *m* valores para gamma, y *n* valores para épsilon.

Una vez que se cuenta con los valores predefinidos para los hiperparámetros, el siguiente paso es generar todas las combinaciones de hiperparámetros posibles con dichos valores. De esta forma, si se tienen *l*, *m* y *n* valores, se tendrán $l \times m \times n$ combinaciones diferentes. Con las combinaciones establecidas, se vuelve hacer un proceso muy similar al que se realizó en la búsqueda aleatoria. Se entrenan modelos con *cross-validation* y se obtiene el respectivo RMSE de cada modelo. Luego se obtiene el RMSE promedio para cada combinación de hiperparámetros; se grafica el espacio de error, respecto a los hiperparámetros; y se guardan las cinco combinaciones que mejor desempeño tuvieron.

Al igual que con la búsqueda aleatoria, se entrenan modelos completos a la par de la búsqueda de malla, usando las mismas combinaciones de hiperparámetros. Se obtiene el RMSE de cada modelo, se grafica dicho error respecto a los hiperparámetros, y se guardan los cinco mejores.

Como se ha mencionado antes, el enfoque que tiene esta metodología es teórico-práctico. Por ello, es que se ordena de esta manera la búsqueda de hiperparámetros. En términos generales, la mejor opción para hacer una búsqueda de hiperparámetros es emplear una búsqueda de malla.

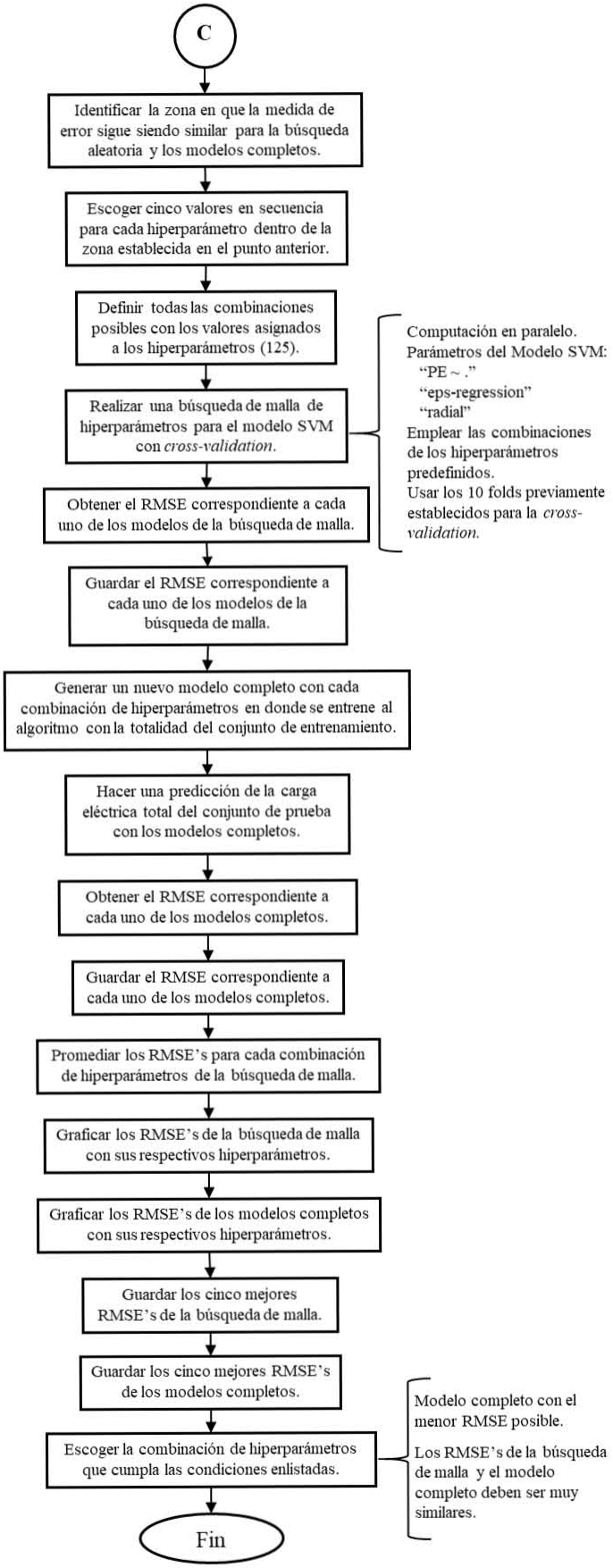
Ésta está diseñada para probar todo el espacio de hiperparámetros disponible. Sin embargo, es tan meticulosa que, cuando el espacio de los hiperparámetros es abierto (no tiene límites), resulta impráctica por la gran cantidad de recursos y tiempo que le tomaría finalizarse. Entonces, mejor se emplea una búsqueda aleatoria con un espacio de hiperparámetros restringido. Así, en lugar de tener muchos modelos con una variación mínima en sus hiperparámetros, se tienen modelos con combinaciones de hiperparámetros muy diferentes entre sí. Esto permite explorar el mismo espacio, pero con un uso de recursos y tiempo menor.

También, es tiempo de revisar con mayor profundidad por qué es que se lleva a cabo el proceso de búsqueda de hiperparámetros y desarrollo de modelos completos simultáneamente. Como se mencionó anteriormente, el motivo principal es evitar el *overfitting*. El proceso de búsqueda de hiperparámetros se lleva a cabo solamente con el conjunto de entrenamiento, lo que significa que, si se optimizara el modelo de esta forma, llegará un momento en que se estará sobreajustando a los ejemplos que se tienen. Cuando se obtienen simultáneamente los modelos completos, se puede monitorear si los resultados son compatibles o existe una diferencia en la distribución del error respecto a las combinaciones de hiperparámetros. Al iniciar con una búsqueda aleatoria en un espacio amplio de hiperparámetros, se espera que los errores de la búsqueda aleatoria y los modelos completos sean muy similares; pero también se busca identificar la zona en que el error es más bajo. Como ya se estableció, el siguiente paso consistiría en otra búsqueda aleatoria, ahora enfocada en la zona que presentó menor error. Se seguiría así hasta que exista una diferencia notable entre los espacios de error de la búsqueda aleatoria y los modelos completos o el desempeño de los modelos ya no mejore. Cuando esto ocurre significa que los modelos se están sobreajustando a los conjuntos de los que se entrenan. Por ello, es necesario encontrar la zona en que sigan manteniendo una medida de error muy cercana. Cuando se tiene esta zona, que debe ser muy restringida, se pasa a la búsqueda de malla, dado que las pequeñas variaciones en los hiperparámetros pueden representar una mejora representativa en el desempeño del modelo.

Así, se aprovechan los recursos disponibles al máximo. Se incrementa el número de modelos al introducir los modelos completos, pero se refuerza la generalización del modelo final. Se incrementa el número de búsquedas aleatorias, pero se evita hacer una búsqueda de malla enorme que consumiría muchos más recursos y tiempo. Y al final, se puede llegar a un espacio de hiperparámetros muy reducido, donde el modelo que se escoja tendrá un sustento muy fuerte.

Regresando a la metodología, una vez que se tiene identificada la zona en que comparten medidas de error muy similares la búsqueda aleatoria y los modelos completos, lo que se recomienda es elegir cinco valores para cada hiperparámetro que se encuentren dentro de esa zona (respetando el orden de secuencia). Se recomienda esta cantidad porque generarán 125 combinaciones distintas, de esta forma, al realizar la búsqueda de malla, el tiempo de computación será muy similar al de la búsqueda aleatoria.

Ilustración 28: Diagrama de flujo de la metodología: Parte 4



Elaboración propia.

Cuando se tienen los cinco mejores modelos y las gráficas de la búsqueda aleatoria y los modelos completos, se busca la zona en que sigan compartiendo una medida de error muy similar. En esta etapa, el espacio de error para cada caso debe ser diferente. Se recomienda, primero, considerar el modelo completo con menor RMSE. Tomando en cuenta este modelo, si, al comparar el error que tuvieron los mismos hiperparámetros en la búsqueda de malla, la diferencia de errores no es grande, el mejor modelo completo será el que se elija finalmente. De lo contrario, se tendrá que elegir el modelo completo con el menor error que no se aleje del error mostrado en la búsqueda de malla.

Y así finaliza la metodología. Ahora se harán notar las diferencias existentes que hay con la metodología que empleó Tüfekci.

En el capítulo anterior se mencionó que existen diferencias entre la metodología empleada en este trabajo y lo que Tüfekci realizó. La primer gran diferencia es que en este trabajo nos enfocamos en un solo algoritmo, mientras que en el otro estudio se hizo un meta-análisis con diferentes algoritmos. Esto cambia muchas perspectivas sobre el enfoque que tiene cada uno. Una de ellas es la percepción que se tiene sobre la búsqueda de hiperparámetros. En el caso de Tüfekci, menciona que no busca modificar los hiperparámetros para evitar el sobreajuste del modelo. En este trabajo, la búsqueda de hiperparámetros es la forma en que se trata de mejorar el desempeño del modelo, aunque claro que se toman medidas para evitar el *overfitting*. En este punto habría que considerar si Tüfekci no modifica los hiperparámetros por el motivo que expresa, o bien, por el arduo trabajo que conlleva hacer una búsqueda de hiperparámetros validada para cada uno de los algoritmos que empleó.

El paquete de modelos que Tüfekci ocupó es el WEKA (Waikato Environment for Knowledge Analysis), el cual es una colección de algoritmos de ML desarrollado por la Universidad de Waikato en Nueva Zelanda. Este paquete se diseñó para el lenguaje Java, contrario a R, que se emplea en este trabajo. En realidad, no existe diferencia en cuanto a la accesibilidad en ambos casos, tanto WEKA como las librerías de R son de acceso libre. Por esta razón, no existe ninguna ventaja o desventaja en cómo se abordó la programación de los modelos.

La siguiente diferencia significativa que se tuvo es directamente sobre el modelo SVM. Mientras que Tüfekci empleó un *kernel* polinomial de segundo grado, en este trabajo se emplea el *kernel* de la función de base radial gaussiana. Esta distinción es representativa porque el nivel de no linealidad que alcanza la función gaussiana es mayor al de la función polinomial. Por tanto, la función gaussiana tiene una capacidad mayor de adaptarse a las curvas n-dimensionales que la función polinomial.

Finalmente, la distinción más importante que se tiene es a la hora de entrenar el algoritmo. Tüfekci emplea una técnica poco tradicional que nunca antes había visto expuesta. Ésta consiste en revolver aleatoriamente la base de datos cinco veces, conformando las cinco hojas de cálculo antes descritas. Una vez que se hizo esto, cada hoja se divide a la mitad para hacer *cross-validation* de

dos *folds*. De esta forma, se toma la mitad de los datos de una hoja de cálculo para entrenar al algoritmo y la otra mitad se emplea como el conjunto de prueba, obteniendo su respectivo RMSE. Y luego se invierten los papeles de dichos conjuntos. Finalmente, se hace esto con cada una de las hojas de cálculo y se promedian los errores. Es muy visible las diferencias que existen con la metodología que se lleva a cabo en este trabajo. En primer lugar, la base se divide en dos conjuntos desde el inicio, por lo que se tiene un conjunto de prueba y otro de entrenamiento. Luego, al conjunto de entrenamiento se le hace una *cross-validation* de diez *folds*. Y al mismo tiempo se hacen modelos completos para evitar el *overfitting*.

Se considera que la metodología aquí presentada tiene un mayor grado de generalización que la que Tüfekci empleó porque los datos que se emplean para validar al modelo (conjunto de prueba), nunca son ingresados al entrenamiento del modelo. Siempre se mantienen como un ente aparte, que bien podrían ser considerados nuevos datos. En cambio, Tüfekci en su entrenamiento emplea los mismos datos una y otra vez con las diferentes combinaciones de la base de datos y los ocupa en su totalidad. Es decir, que no existe ningún dato que haya estado completamente fuera del proceso de entrenamiento. Claro que debe mencionarse que, al momento que divide a la mitad a la base de datos, tiene una desventaja con el procedimiento seguido en este trabajo, debido a que la partición 80-20 para los conjuntos de entrenamiento y prueba le da una mayor cantidad de datos con los cuales aprender al modelo. A la vez, en nuestro caso, para evaluar el modelo se tienen el promedio del proceso de *cross-validation* y el RMSE del modelo completo, mientras que Tüfekci tiene únicamente el primero. Además de que se debe mencionar que dicho promedio viene de una partición en diez *folds* para este trabajo, mientras que el otro viene de una partición de dos *folds*, por lo que la consistencia del modelo es mayor en el esquema que tiene una mayor cantidad de particiones.

Todas estas diferencias que se llevan a cabo simultáneamente en la metodología propuesta tendrán un resultado que se mostrará en el siguiente capítulo. Sin embargo, al realizarse simultáneamente, no se podrá hacer un análisis específico de qué cambios fueron los que mejoraron o empeoraron al modelo respecto a los resultados de Tüfekci. En los párrafos anteriores se describieron los motivos por los que esta metodología debe superar en resultados a la de Tüfekci, pero cuando se evalúen dichas diferencias no se sabrá a ciencia cierta la proporción del efecto de cada uno de estos cambios. Además, de que el objetivo de este trabajo no es superar los resultados que ya se tienen, sino mejorar la generalización de éstos con el uso de mejores prácticas.

3.3 Resultados y Análisis de Resultados

Una vez que se revisó la metodología a seguir, es hora de ponerla en marcha y comenzar a obtener resultados. Los primeros pasos descritos dentro de la metodología son preliminares y como tal no tienen un resultado. Es hasta que se realiza la separación de la base de datos en el esquema 80-20

que podemos comparar los conjuntos de entrenamiento y prueba para corroborar que sean compatibles. De la separación aleatoria, se obtuvo la siguiente tabla comparando los conjuntos:

Tabla 7: Comparación de los estadísticos de los conjuntos de entrenamiento y prueba

	Conjunto de Prueba.		**Conjunto de Entrenamiento.**	
Variables.	***Media.***	***Desviación Estándar.***	***Media.***	***Desviación Estándar.***
AT	19.7191	7.4789	19.6343	7.4462
V	54.5401	12.7660	54.2473	12.6935
AP	1013.2230	6.0211	1013.2681	5.9184
RH	73.4403	14.7197	73.2762	14.5710
PE	454.2014	17.2821	454.4059	17.0137

Elaboración propia.

Como se puede notar, en todas las variables los estadísticos de los conjuntos son muy similares. Solamente tienen diferencias de décimas, que, cuando se considera la cantidad de datos que representan (1913 para el conjunto de prueba y 7655 para el conjunto de entrenamiento), resulta una diferencia insignificante. Entonces, desde este punto se logra observar que los conjuntos serán compatibles. En caso de que las diferencias hubiesen sido más notables, se tendría que volver a separar aleatoriamente la base completa. Aunque las diferencias son muy pequeñas, para asegurarnos que sus estimadores son verdaderamente los mismos se hacen las pruebas estadísticas descritas en la metodología, cuyo resultado es el siguiente:

Tabla 8: Resultados de las pruebas estadísticas para los conjuntos de prueba y entrenamiento

	Variables de los conjuntos.				
Prueba estadística.	*AT* p-value	*V* p-value	*AP* p-value	*RH* p-value	*PE* p-value
t-Student.	0.6569	0.369	0.7687	0.6621	0.6424
Levene.	0.9407	0.4487	0.7283	0.5802	0.6064

Elaboración propia.

Como se puede observar en la tabla, el *p-value* de todas las pruebas estadísticas al comparar cada una de las variables de los dos conjuntos es muy superior a 0.05. Esto representa que con una significancia mayor del 95% se puede asegurar que las diferentes hipótesis nulas no se pueden rechazar. Lo que implica que no existe diferencia entre las medias y varianzas de los conjuntos de prueba y entrenamiento. Una vez más, si el *p-value* de cualquier prueba hubiera sido

menor o igual a 0.05, las hipótesis nulas se rechazarían y se tendría que partir aleatoriamente la base de nuevo.

Ya que se comprobó la compatibilidad de los conjuntos, se prosigue con la primera búsqueda aleatoria. Se debe recordar que para nuestro caso se ejecutan 1100 modelos distintos, con los hiperparámetros inicialmente definidos de la siguiente forma:

- Costo: Distribución uniforme de 0 a 50.
- Gamma: Distribución uniforme de 0 a 50.
- Épsilon: Distribución uniforme de 0 a 2.

El tiempo promedio que tardaron en correr las búsquedas de hiperparámetros (excluyendo la primera) fue de una hora y media. No obstante, esta primera búsqueda aleatoria solamente demoró veinte minutos. Cuando se empezó a usar el equipo, hubo tiempos que llegaron a ser de cuatro horas y media, sin embargo, esta situación se dio porque los drivers de Windows y del BIOS no estaban actualizados y el equipo no funcionaba de forma óptima. También, la versión de R que se tenía instalada no era la más reciente. Por lo que se recomienda que antes de correr algún modelo, se actualice a la versión más reciente del BIOS, de Windows y de R. También, se modificaron ciertos parámetros de la computadora para que el tiempo de computación se redujera. En primer lugar, se deshabilitaron todas las operaciones en segundo plano, para que cuando se estuvieran corriendo los modelos no existieran procesos adicionales que desviaran recursos del CPU. Y, la modificación más importante, se configuró un plan de energía de alto rendimiento para poder ocupar los recursos de la PC al cien por ciento.

De esta forma, por el procedimiento de *cross-validation*, se corrieron 1000 modelos distintos (diez para cada combinación aleatoria de hiperparámetros) y 100 modelos completos que ocuparon la totalidad del conjunto de entrenamiento para la fase de entrenamiento del modelo y la totalidad del conjunto de prueba para la fase de prueba. Cabe recordar que para cada modelo se obtiene un RMSE, y en el caso de la búsqueda aleatoria, los errores de los modelos con los mismos hiperparámetros se promedian. Una vez que se tuvo esto, se extrajeron las siguientes tablas con los mejores modelos:

Tabla 9: Cinco mejores modelos de la búsqueda aleatoria 1

n	Costo.	Gamma.	Épsilon.	RMSE.
56	12.3793	16.9842	0.1295	5.4647
62	11.7694	9.7280	0.1583	4.6648
71	17.5109	7.8404	0.3615	5.2864
81	5.1409	22.5552	0.0216	5.9121
84	17.4456	0.2687	0.2135	3.9602

Elaboración propia.

Tabla 10: Cinco mejores modelos completos de la fase 1

n	Costo.	Gamma.	Épsilon.	RMSE.
56	12.3793	16.9842	0.1295	5.4370
62	11.7694	9.7280	0.1583	4.6715
71	17.5109	7.8404	0.3615	5.2647
81	5.1409	22.5552	0.0216	5.8308
84	17.4456	0.2687	0.2135	4.0245

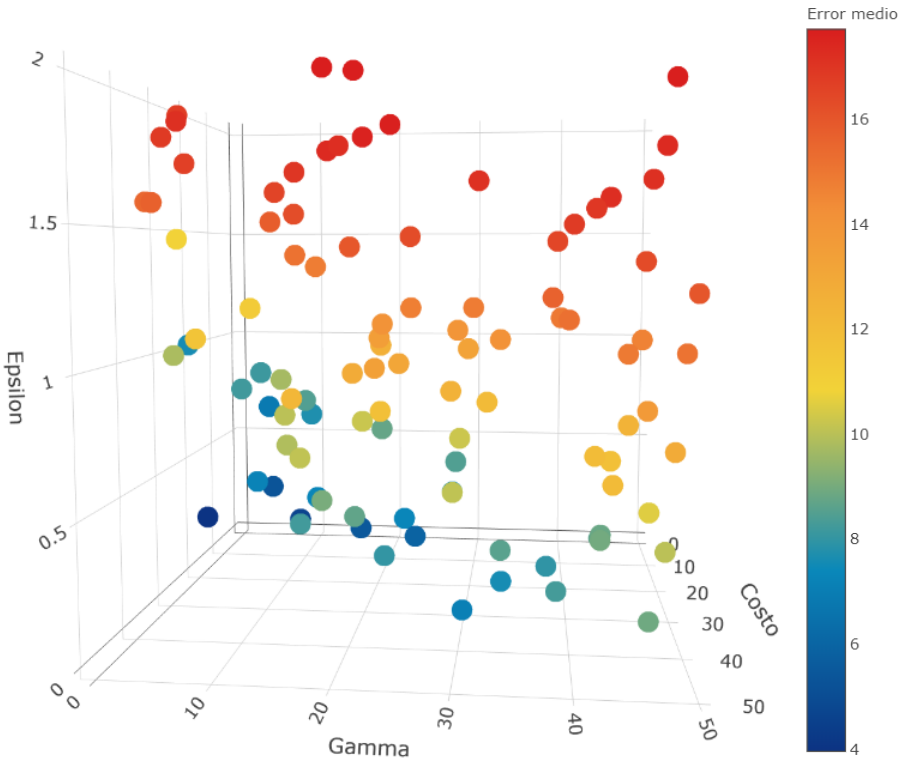
Elaboración propia.

Existen cinco columnas en las tablas, tres son los hiperparámetros de cada modelo, una es la medida de error y la primera columna es el número de la combinación aleatoria de los hiperparámetros. Como se puede observar, las cinco mejores combinaciones de hiperparámetros para la búsqueda aleatoria y para los modelos completos son las mismas (56, 62, 71, 81 y 84). Esto nos indica en primera instancia que a este nivel de detalle no existe *overfitting* ni hacia el conjunto de entrenamiento, ni hacia el de prueba. También nos deja ver un poco la tendencia del error dentro del espacio de los hiperparámetros. En primer lugar, se nota que el reducir el valor de gamma mejora el desempeño del modelo, y en segundo, que los valores más pequeños de épsilon también dan mejores resultados. Pero para tener un mejor panorama del comportamiento de error respecto a los hiperparámetros del modelo, se revisarán las gráficas de desempeño del modelo (Ilustraciones 29 y 30).

Como se puede observar en las gráficas, tanto para la búsqueda aleatoria como para los modelos completos, los puntos mostrados son los mismos. Esto es obvio porque representan el desempeño de los modelos con los mismos hiperparámetros. Las gráficas consisten de tres ejes que representan a los hiperparámetros y de un código de color que representa la medida de error de los modelos con dichos hiperparámetros. A un lado de la gráfica se encuentra el código de color que establece que los puntos con color rojo son aquellos con un mayor error y, así, el error va disminuyendo de forma cromática hasta llegar a un azul intenso que representa el menor error presentado.

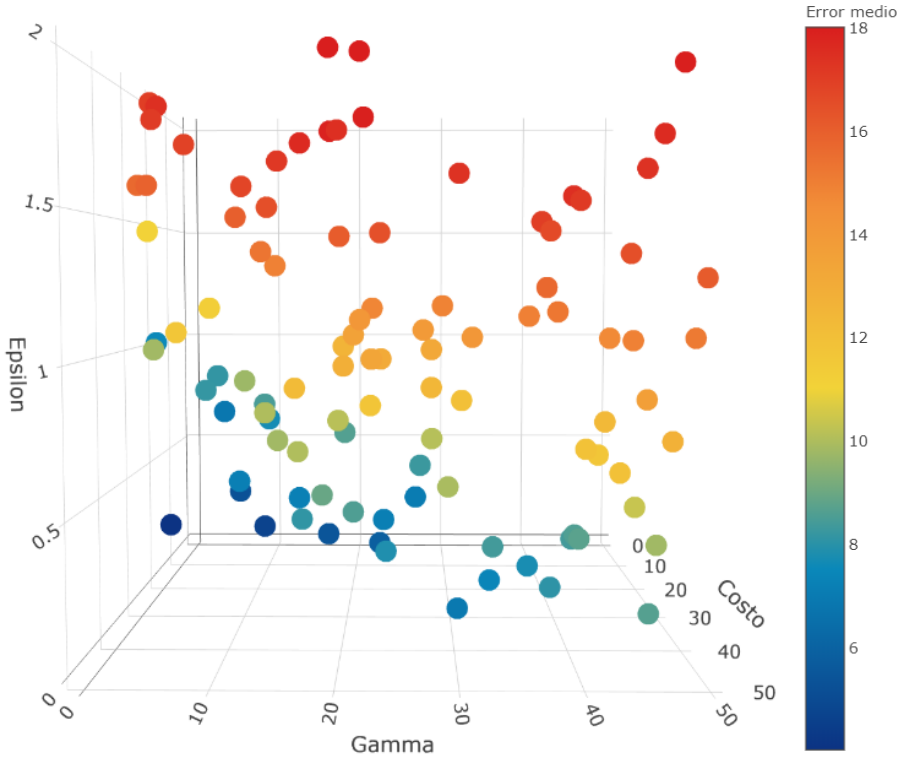
Comparando la distribución del error en estas dos gráficas, se puede notar un comportamiento muy similar entre los casos de la búsqueda aleatoria y de los modelos completos. Sin embargo, si se pone atención, se notará que existe una ligera diferencia de coloración en ciertos puntos, por lo que no son exactamente iguales. Lo más importante a notar a simple vista es que la escala de error para cada caso es diferente. Para la búsqueda aleatoria el RMSE tiende a ser menor que para los modelos completos, de modo que el error mínimo de la búsqueda está por debajo de un RMSE de cuatro, mientras que el mínimo de los modelos completos se encuentra por encima de cuatro. De la misma forma, el error máximo de los modelos completos llega a 18, mientras que el máximo de la búsqueda aleatoria no llega a este valor.

Ilustración 29: Gráfica de desempeño de la búsqueda aleatoria 1: Plano gamma-épsilon



Elaboración propia.

Ilustración 30: Gráfica de desempeño de los modelos completos fase 1: Plano gamma-épsilon

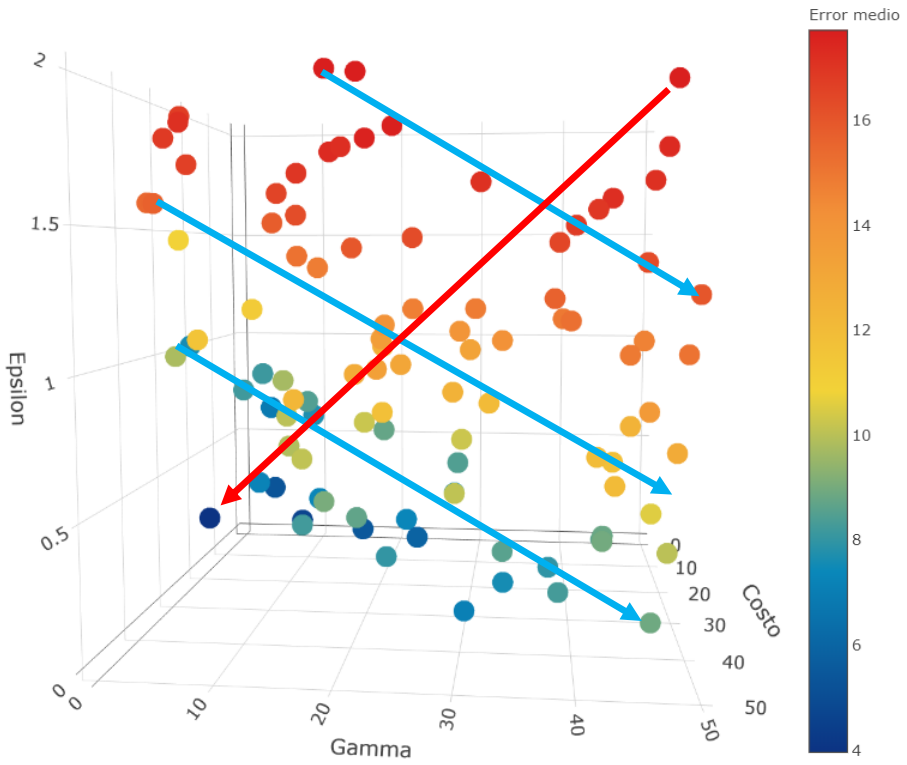


Elaboración propia.

Esta situación es de esperarse, debido a que los modelos están entrenados con el conjunto de entrenamiento. De forma que, cuando se hace la predicción en la *cross-validation*, se ocupan datos que forman parte de este conjunto, que de alguna forma el algoritmo conoce. Mientras que cuando se hace el pronóstico con los modelos completos, se emplean los datos del conjunto de prueba, los cuales son totalmente ajenos para el modelo. Esto significa que el desempeño de los modelos completos forzosamente será peor que los de la búsqueda aleatoria desde un punto de vista general. Sin embargo, como se aprecia en las gráficas, esta diferencia no es tan grande, por lo que hasta ahora los modelos mostrados tienen una generalización aceptable, aunque los RMSE's que se obtienen son altos.

Ahora, se hará un análisis de la distribución del error con estos hiperparámetros ocupando de nuevo la ilustración 29:

Ilustración 31: Análisis de la distribución del error en la Ilustración 29



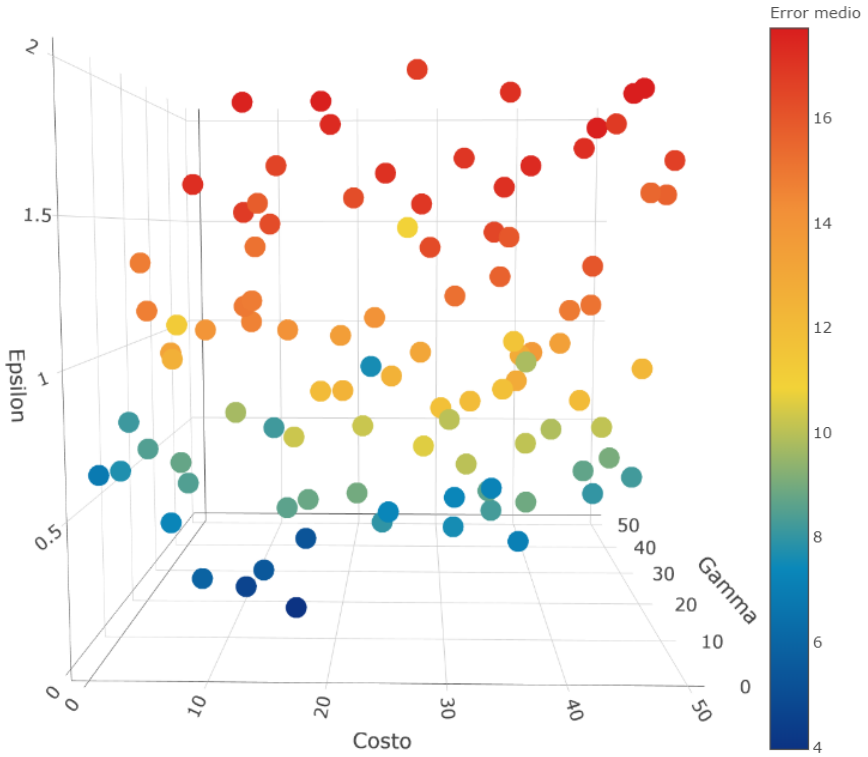
Elaboración propia.

En la Ilustración 31 se nota cómo se distribuye el error. La flecha roja indica la dirección en que el error va disminuyendo. De esta forma, nos podemos dar cuenta que para obtener modelos con mejores desempeños los valores de gamma y épsilon deben ser bajos, confirmando lo que se había descrito con los cinco mejores modelos. Por otro lado, las flechas azules indican planos de modelos cuyo error es igual o muy similar. Así, nos podemos imaginar familias de

planos paralelos cuyo error se mantiene constante con una dirección diagonal hacia abajo cuando la gráfica es vista desde el plano gamma-épsilon.

Hasta ahora, solo se ha hablado de dos hiperparámetros: gamma y épsilon. Mientras que el costo se ha dejado afuera. Para mostrar los motivos de esto, se muestra la siguiente gráfica:

Ilustración 32: Gráfica de desempeño de la búsqueda aleatoria 1: Plano costo-épsilon



Elaboración propia.

En este caso, solo ocupamos la gráfica de la búsqueda aleatoria, debido a que la de los modelos completos es prácticamente la misma. Como se puede notar, en un mismo nivel del costo existe mucha diferencia entre los errores. Por ejemplo, si se toma la región cercana a un costo de diez, se notará que a un nivel similar de épsilon (0.25), los puntos tienen muchos errores diferentes: hay puntos verdes, puntos azules claro y puntos azules oscuros. Estos efectos provienen más de gamma y épsilon que lo que genera el costo. Aunque sí ha de notarse que, con un menor costo, se tienen mejores desempeños. Cabe recordar que esta corrida de modelos se tomó únicamente alrededor de veinte minutos, por lo que, si las combinaciones aleatorias de hiperparámetros se ven sesgadas hacia un sector, se puede volver a hacer este proceso hasta tener una distribución los más uniforme posible.

Una vez que se analizan los mejores cinco desempeños de cada caso, así como las tendencias que marcan las gráficas, se reduce el espacio de hiperparámetros a la zona que mejores

desempeños mostró. En este caso, se inicia una nueva búsqueda aleatoria con las siguientes características:

- Costo: Distribución uniforme de 0 a 10.
- Gamma: Distribución uniforme de 0 a 10.
- Épsilon: Distribución uniforme de 0 a 0.25.

Para esta búsqueda, se tiene la misma cantidad de modelos que en la búsqueda anterior. Los resultados se muestran en seguida:

Tabla 11: Cinco mejores modelos de la búsqueda aleatoria 2

n	Costo.	Gamma.	Épsilon.	RMSE.
5	5.6533	1.8015	0.0778	3.8052
28	2.2314	1.9073	0.0580	3.8067
38	2.6973	1.0821	0.1217	3.8018
47	6.1287	1.8715	0.1279	3.7957
55	3.0249	1.8126	0.0642	3.8041

Elaboración propia.

Tabla 12: Cinco mejores modelos completos de la fase 2

n	Costo.	Gamma.	Épsilon.	RMSE.
5	5.6533	1.8015	0.0778	3.8251
28	2.2314	1.9073	0.0580	3.8228
45	9.5567	1.4729	0.0155	3.8282
55	3.0249	1.8126	0.0642	3.8185
78	7.0685	1.7774	0.0371	3.8199

Elaboración propia.

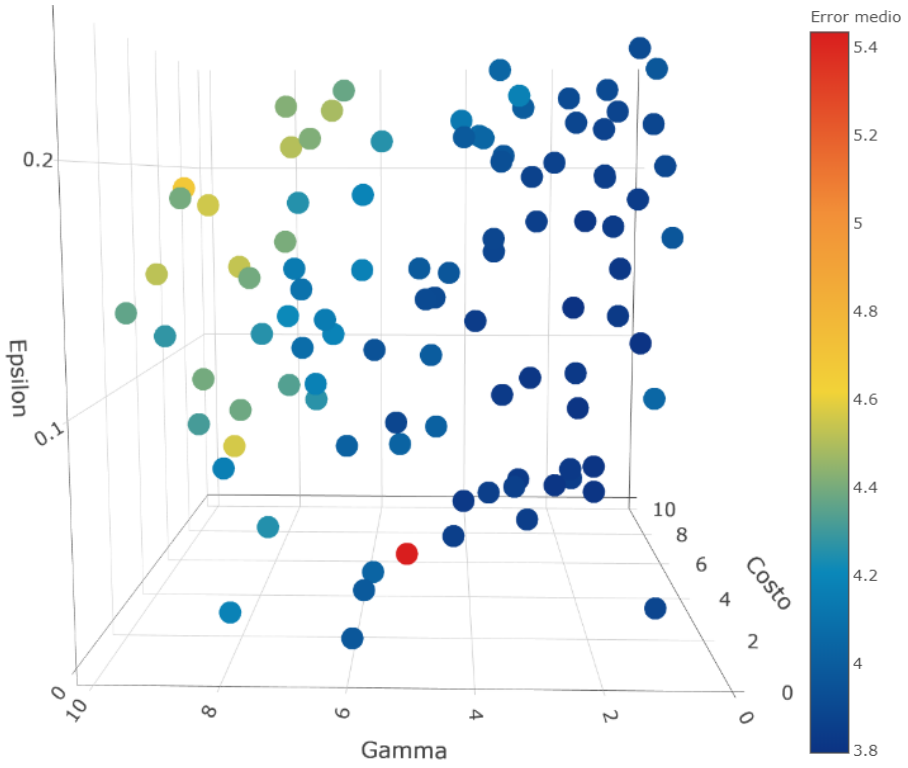
Como se puede observar en los resultados presentados en las tablas, al reducir el espacio de hiperparámetros, los mejores desempeños que se obtuvieron tienen un error mucho menor que los encontrados en la primera búsqueda. En este caso, sin embargo, las mejores combinaciones de hiperparámetros ya no se corresponden entre sí. Unas combinaciones se repiten, pero existen cuatro de ellas que no se encuentran ni en una ni en la otra tabla. Esto quiere decir que el nivel de detalle al que hemos llegado está comenzando por ser factor para que el modelo se sobreajuste hacia el conjunto de entrenamiento (búsqueda aleatoria) o al conjunto de prueba (modelos completos).

Desde estas tablas también se pueden ver algunos de los efectos de los hiperparámetros sobre el desempeño de los modelos. En primera instancia, los mejores valores de gamma están

entre uno y dos. En los modelos completos, épsilon se presenta por debajo de 0.1, mientras que en la búsqueda aleatoria está desde 0.05 hasta 0.13. Y, por otro lado, mientras que el costo se ubica en una zona definida de 2 a 6.5 para la búsqueda aleatoria, en los modelos completos se esparce a lo largo de todo el espacio de 0 a 10 que se le permitió.

Para tener una mejor referencia de los nuevos espacios de hiperparámetros, se analizarán las gráficas resultantes de esta segunda búsqueda:

Ilustración 33: Gráfica de desempeño de la búsqueda aleatoria 2: Plano gamma-épsilon



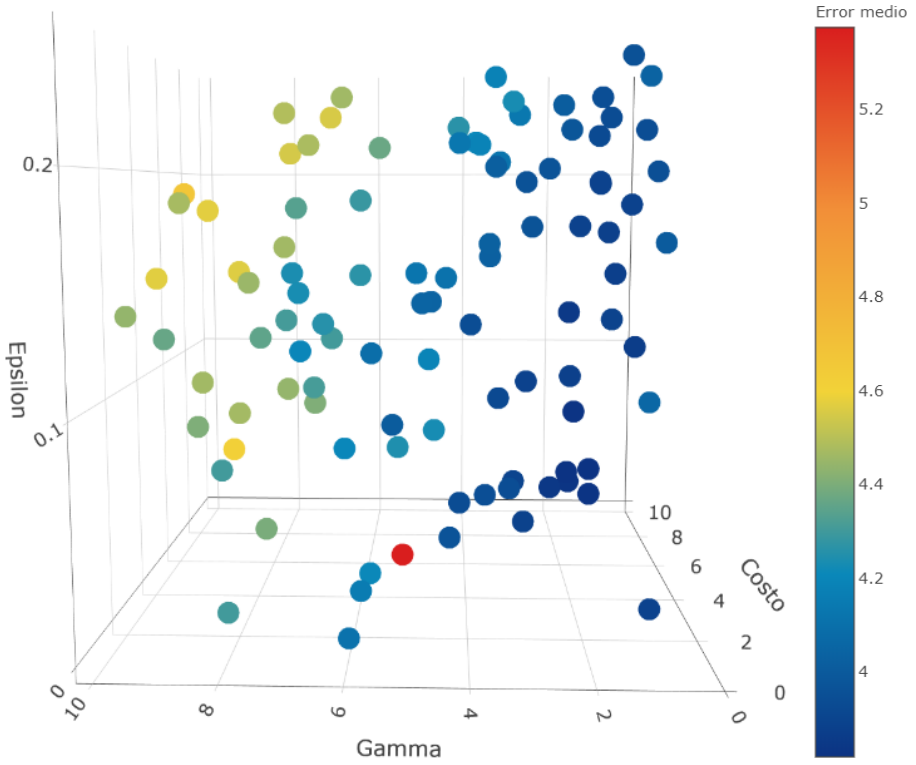
Elaboración propia.

Como se puede observar en las Ilustraciones 33 y 34, los espacios de hiperparámetros siguen compartiendo muchas similitudes. En general, a menor gamma, mejor es el desempeño de los modelos. En cuanto a épsilon no se muestra mucha variación de 0.15 hacia abajo. Y, por otro lado, el costo sigue siendo un parámetro que no muestra gran impacto sobre el desempeño del modelo. Se puede notar cómo es que, a iguales niveles del costo, los errores que se presentan son muy diversos, aunque se tiene una leve tendencia que. a menor costo, mejor desempeño.

A diferencia de las gráficas anteriores, se puede notar que éstas tienen una variación en la escala de color mucho menor. Esto se debe a un punto muy interesante ubicado en gamma 5.06, épsilon 0.047 y costo 0.097. La gran densidad de puntos cuenta con un error por debajo de 4.4, sin embargo, al existir este punto la escala de color se amplía. El valor del error de este punto es de

5.38 para los modelos completos y de 5.43 para la búsqueda aleatoria. Es una anomalía si se consideran el error de todos los puntos que están a su alrededor. Lo único especial que este punto tiene es que su costo es el mínimo de las combinaciones aleatorias de hiperparámetros. Esto nos indica que hay un límite en el valor del costo de modo que, si es muy pequeño, el desempeño del modelo empezará a verse afectado.

Ilustración 34: Gráfica de desempeño de los modelos completos fase 2: Plano gamma-épsilon



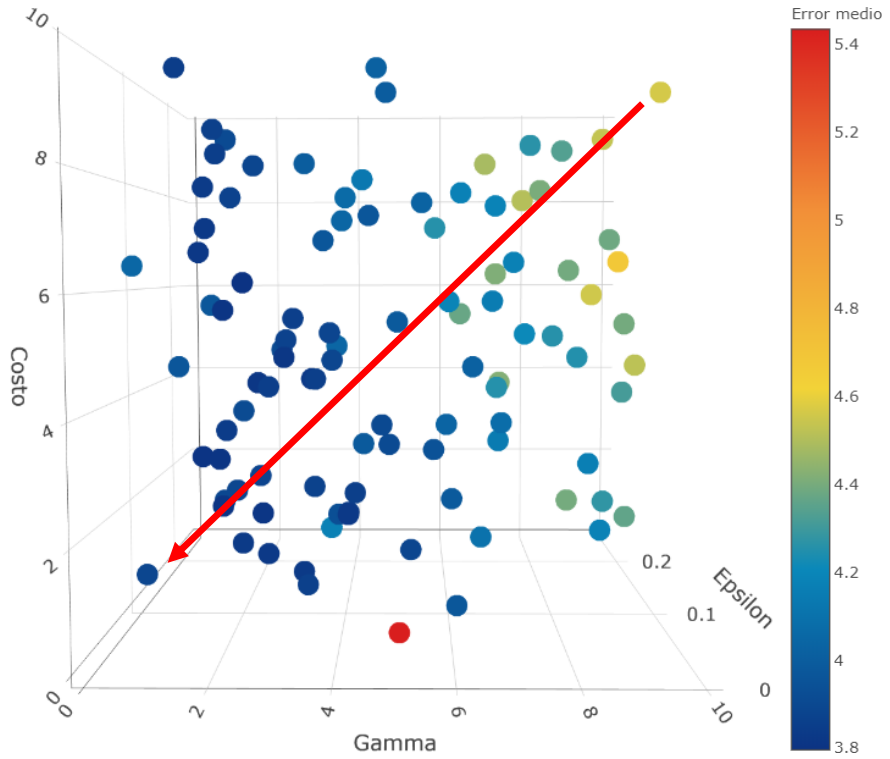
Elaboración propia

Ahora, como se puede observar en la Ilustración 35, al cambiar el plano desde el que se ve la gráfica se puede notar una clara tendencia hacia valores de gamma y costo bajos. Claro que hay que tener en cuenta respetar el límite de 0.1 en el costo que se discutió anteriormente. Como hasta este momento los resultados siguen mejorando y no existe una diferencia significativa entre los espacios de error resultantes de la búsqueda aleatoria y los modelos completos, se continuará con otra búsqueda aleatoria. De esta forma, se tendrá un espacio más reducido en el cual podremos encontrar mejores resultados de forma consistente.

Un aspecto a tomar en cuenta es que el tiempo de procesamiento de estos segundos modelos es mucho mayor que el de la primera búsqueda. En este caso, el tiempo de procesamiento está entre una hora y media y dos horas, por lo que hay que considerar si vale la pena hacer otra búsqueda. Lo interesante aquí es la pregunta de por qué aumenta tanto el tiempo de procesamiento

si se están ejecutando la misma cantidad de modelos. La respuesta yace en uno solo de los hiperparámetros. Como se ha de recordar, épsilon en un modelo SVM de regresión representa el ancho del margen que rodea a la curva de regresión. Los ejemplos que se encuentren en dicho margen son los llamados vectores de soporte. Éstos se encuentran cuando los multiplicadores de Lagrange son diferentes de cero al resolver el problema dual de programación cuadrática con restricciones. Así, en la primera búsqueda se tenía un margen desde cero hasta dos. Mientras que, en la segunda, épsilon tomó valores de cero a 0.25. Al reducir el margen, lo que se provoca es que existan muchos más ejemplos que se ubiquen exactamente sobre éste, por lo que los valores de los multiplicadores no serán de cero, sino que se tendrán que calcular. Con un margen más grande, la mayor parte de los ejemplos se ubican dentro del margen, por lo que el valor de sus multiplicadores es de cero. Esto se puede comprobar fácilmente al comparar el número de vectores que se ocuparon en las dos diferentes búsquedas, por ejemplo, en un modelo de la primera búsqueda se tienen 372 vectores de soporte, mientras que en uno de la segunda se tienen 3853 vectores. De esta forma, ni gamma ni el costo son factores, sino que épsilon es el que provoca que ciertos modelos sean más demandantes.

Ilustración 35: Análisis de la gráfica de desempeño de la búsqueda aleatoria 2: Plano gamma-costo



Elaboración propia.

Ya que se hicieron los análisis correspondientes a esta segunda búsqueda, los hiperparámetros que se seleccionaron para la siguiente búsqueda se muestran a continuación. Cabe

recalcar que, aunque se observó una ligera tendencia con el valor del costo, en realidad no fue tan pronunciada como con gamma o épsilon, por lo que mantuvo el rango de valores de la anterior búsqueda para este hiperparámetro:

- Costo: Distribución uniforme de 0 a 10.
- Gamma: Distribución uniforme de 0 a 2.
- Épsilon: Distribución uniforme de 0 a 0.15.

El tiempo que esta búsqueda se llevó fue prácticamente el mismo que se presentó en la búsqueda pasada. Sin embargo, el desempeño de los modelos debería de mejorar. Podemos ver los mejores desempeños a continuación:

Tabla 13: Cinco mejores modelos de la búsqueda aleatoria 3

n	Costo.	Gamma.	Épsilon.	RMSE.
2	4.3920	1.5449	0.1089	3.7850
18	2.9606	1.6931	0.0962	3.7830
61	3.1749	1.3571	0.1145	3.7857
64	3.4397	1.3367	0.1149	3.7846
71	2.8108	1.6525	0.1031	3.7852

Elaboración propia.

Tabla 14: Cinco mejores modelos completos de la fase 3

n	Costo.	Gamma.	Épsilon.	RMSE.
15	2.7304	1.8081	0.0449	3.8177
29	4.4701	1.8437	0.0598	3.8119
38	5.0056	1.8468	0.0662	3.8128
40	3.1859	1.7803	0.0447	3.8161
56	3.1962	1.6567	0.0508	3.8185

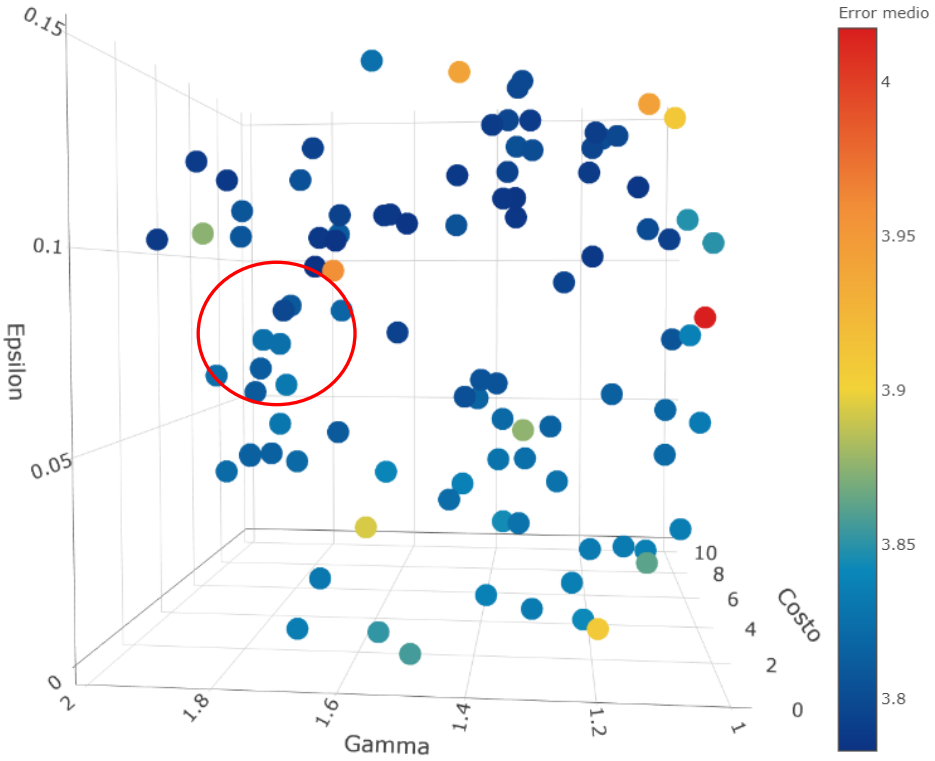
Elaboración propia.

Como se puede notar en las tablas, los mejores resultados de cada caso no coinciden en ninguna combinación de hiperparámetros. A la vez, mientras que los desempeños en la búsqueda aleatoria mejoran un poco respecto a los pasados, para los modelos completos la mejora en el desempeño es mínima. Todo esto nos indica que hemos llegado a un detalle en el espacio de hiperparámetros en donde el modelo se está sobreajustando al conjunto de entrenamiento. Podemos ver que, en cuanto a los valores de los hiperparámetros, el costo se mantiene muy parecido en los dos casos, pero tanto en gamma como en épsilon, los valores varían. En primer lugar, los valores de gamma para la búsqueda aleatoria van de 1.33 a 1.7, mientras que para los

modelos completos van de 1.65 a 1.85. Esta diferencia nos indica un desplazamiento hacia la derecha en el eje gamma. Segundo, los valores de épsilon para la búsqueda aleatoria van de 0.096 a 0.115; y para los modelos completos de 0.045 a 0.066. Lo que representa un desplazamiento hacia abajo.

Al observar las Ilustraciones 36 y 37, se puede notar la gran diferencia que existe en los nuevos espacios generados. Una cosa importante para señalar es que los valores del costo muy cercanos a cero siguen afectando drásticamente el desempeño del modelo.

Ilustración 36: Gráfica de desempeño de la búsqueda aleatoria 3: Plano gamma-épsilon

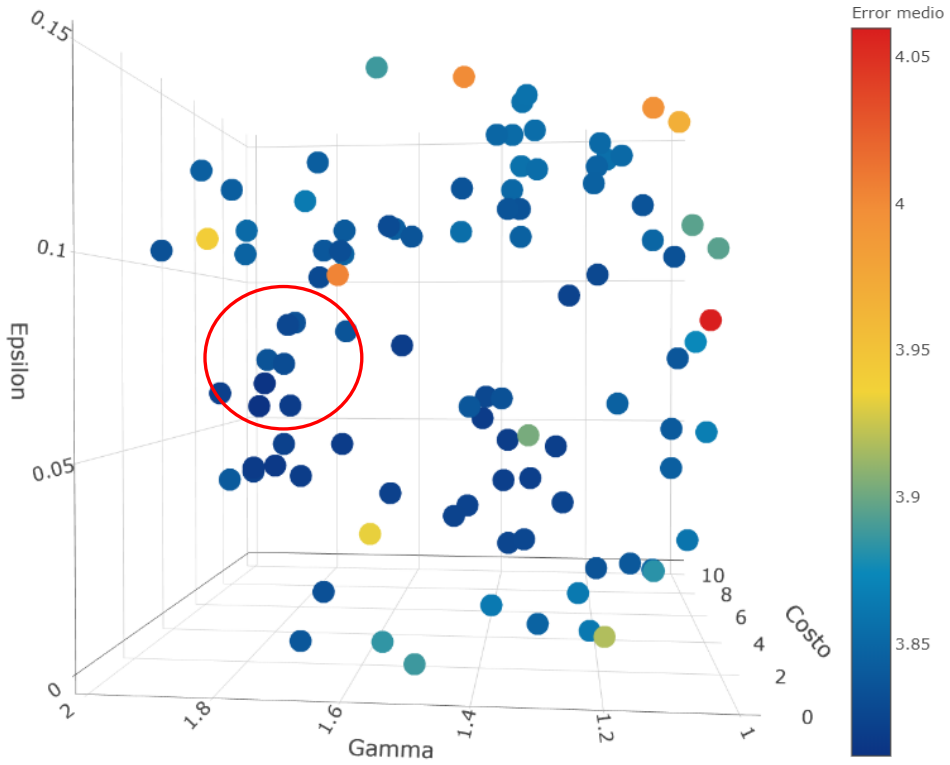


Elaboración propia.

En estas gráficas, el parámetro que mayor tendencia muestra es épsilon. Mientras que, para la búsqueda aleatoria un valor de épsilon mayor a 0.1 es mejor, para los modelos completos uno que se encuentre cerca de 0.05 resulta más benéfico. Por su parte, gamma no resulta tan importante como lo había sido hasta ahora.

Al revisar las gráficas, se notará una zona marcada. Analizando las gráficas y los resultados de los RMSE's de los modelos, se determinó que esa zona era aquella en donde mejor desempeño tuvieron los modelos completos, pero, a la vez, en donde el valor del RMSE de los modelos provenientes de la búsqueda aleatoria se parecía mucho al de los modelos completos. De esta forma, no se sobreajusta el modelo hacia el conjunto de prueba.

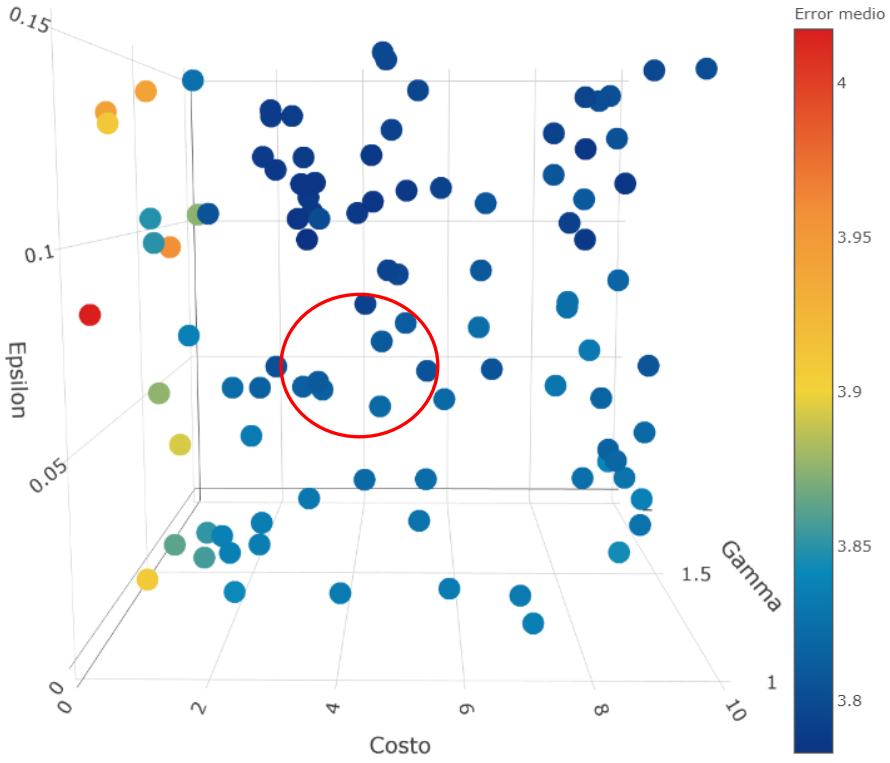
Ilustración 37: Gráfica de desempeño de los modelos completos fase 3: Plano gamma-épsilon



Elaboración propia.

Por último, se muestra una imagen de la zona seleccionada desde el eje del costo:

Ilustración 38: Gráfica de desempeño de la búsqueda aleatoria 3: Plano costo-épsilon



Elaboración propia.

En esta nueva gráfica se puede notar que las combinaciones con valores de costo muy cercanos a cero tienen una caída en el desempeño muy pronunciada. Mientras que el resto de los valores se mantienen constantes, influidos sobre todo por épsilon más que por el costo.

Ya que definimos la zona en donde aún se encuentran similitudes entre la búsqueda aleatoria y los modelos completos, se procede a hacer una búsqueda de malla. Esta búsqueda es más exhaustiva que la aleatoria, de modo que prueba combinaciones de valores más cercanos entre sí y así encontrar un mejor desempeño final. Se debe recordar que, en la metodología, para la búsqueda aleatoria se propusieron cinco valores distintos para cada hiperparámetro. En la búsqueda aleatoria se emplearon los siguientes valores:

- Costo: 3, 3.5, 4, 4.5 y 5.
- Gamma: 1.81, 1.82, 1.83, 1.84 y 1.85.
- Épsilon: 0.06, 0.065, 0.07, 0.075 y 0.08.

Cuando se tienen definidos estos valores, se hacen todas las combinaciones disponibles entre ellos. En este caso, se tienen 5^3 combinaciones, es decir, 125 combinaciones distintas. Esto va a generar 1250 modelos distintos en la búsqueda de malla y 125 más en los modelos completos, para un total de 1375 modelos. En realidad, el tiempo que se tardó este procedimiento fue muy similar al de las búsquedas aleatorias, no modificaron mucho el tiempo los 275 modelos adicionales. Los mejores cinco resultados mostrados en cada caso fueron los siguientes:

Tabla 15: Cinco mejores modelos de la búsqueda de malla

n	Costo.	Gamma.	Épsilon.	RMSE.
101	3	1.81	0.08	3.791263
106	3	1.82	0.08	3.791239
111	3	1.83	0.08	3.791332
116	3	1.84	0.08	3.791249
121	3	1.85	0.08	3.791300

Elaboración propia.

Tabla 16: Cinco mejores modelos completos de la fase 4

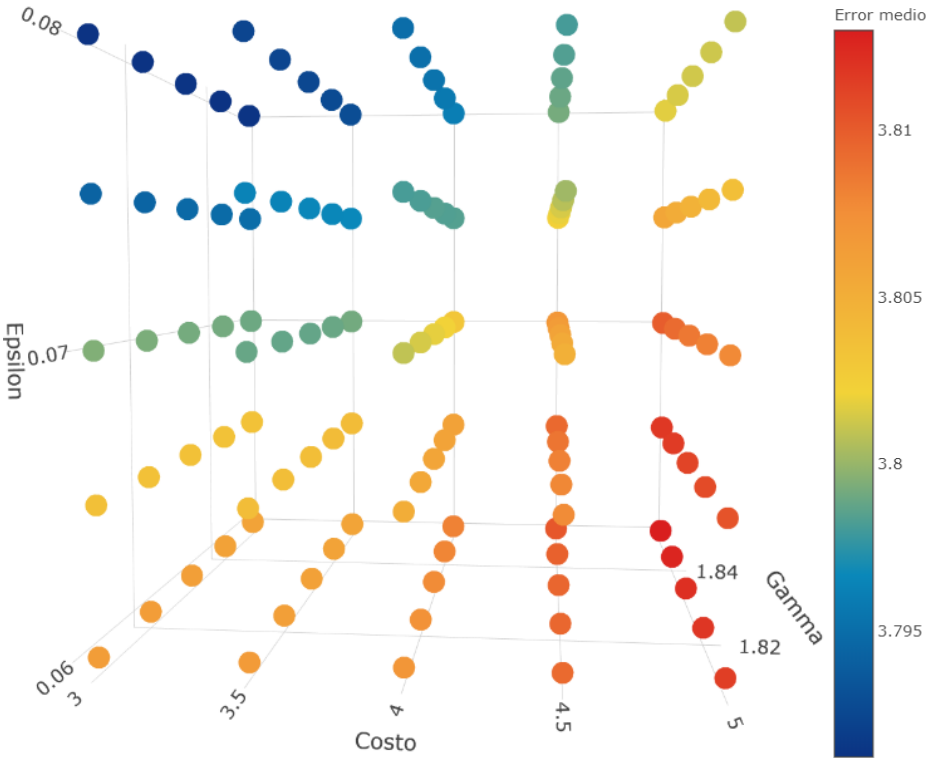
n	Costo.	Gamma.	Épsilon.	RMSE.
15	5	1.83	0.060	3.810341
20	5	1.84	0.060	3.810263
25	5	1.85	0.060	3.810042
35	5	1.82	0.065	3.810239
40	5	1.83	0.065	3.810318

Elaboración propia.

Como era de esperarse, los mejores modelos no se corresponden para nada en cuanto a la combinación de hiperparámetros. Sin embargo, aunque la tendencia con épsilon se mantiene, es decir, que para los modelos completos es mejor una épsilon menor y para la búsqueda de malla es mejor una épsilon mayor; en el costo sí se marcó una diferencia que no se había notado: el desempeño de los modelos completos mejora con un mayor costo, y el desempeño de la búsqueda de malla mejora con un menor costo. Además, como ya se venía observando, gamma no tiene una influencia muy notoria a este nivel de detalle.

Ahora, para tener una visión más representativa de los espacios que se crearon, se revisarán sus gráficas correspondientes:

Ilustración 39: Gráfica de desempeño de la búsqueda de malla: Plano costo-épsilon

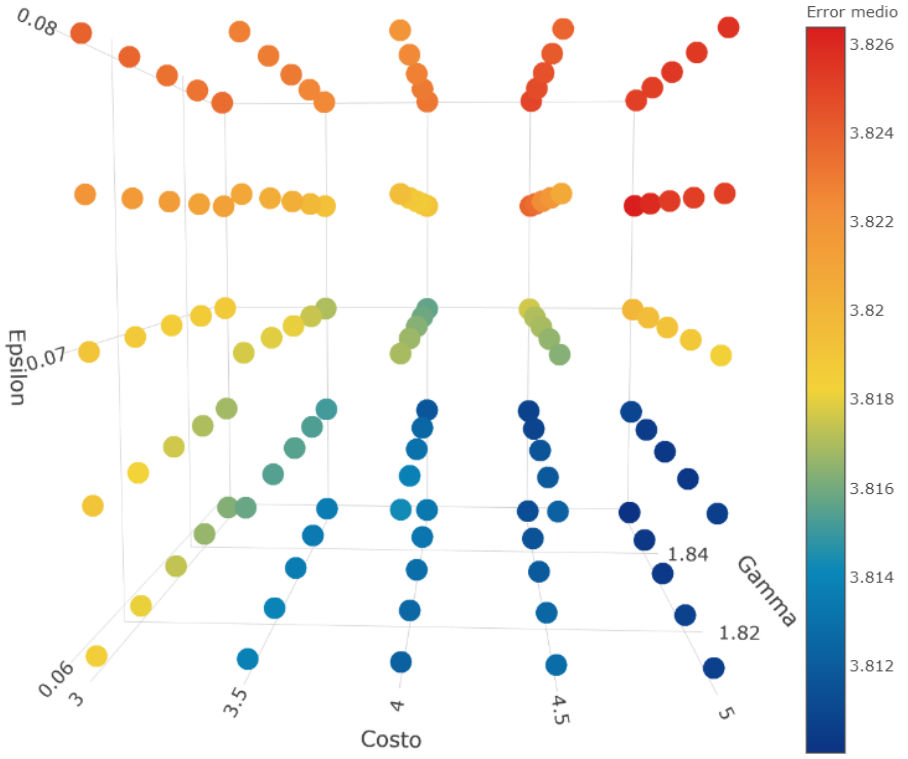


Elaboración propia.

La disociación que existe entre los espacios es muy notoria en este momento que no se cree necesario señalar las tendencias con flechas como en las anteriores ocasiones. Para la búsqueda de malla, los mejores resultados se hallan en un costo de tres y una épsilon de 0.08; y sus peores resultados se encuentran en un costo de cinco y una épsilon de 0.06. Por otro lado, para los modelos completos, sus mejores desempeños se encuentran en un costo de cinco y una épsilon de 0.06; y sus peores desempeños en un costo de cinco y una épsilon de 0.08. Esta diferencia nos hace ver que los espacios de error de este modelo con los conjuntos de entrenamiento y prueba son

mayormente iguales (mostrado en las primeras búsquedas aleatorias), pero comparados a detalle, existen diferencias significativas entre ellos.

Ilustración 40: Gráfica de desempeño de los modelos completos fase 4: Plano costo-épsilon



Elaboración propia.

El caso de los modelos completos es muy interesante, puesto que, en lugar de comportarse como la búsqueda de malla, en donde la esquina superior izquierda se antepone a la esquina inferior derecha; los mejores y peores resultados que muestra están sobre el mismo valor del costo: cinco. Esto puede significar que en esta región el error no se comporte como un hiperplano, sino como una curva diferente, pero serían necesarios más puntos para poder determinar esto.

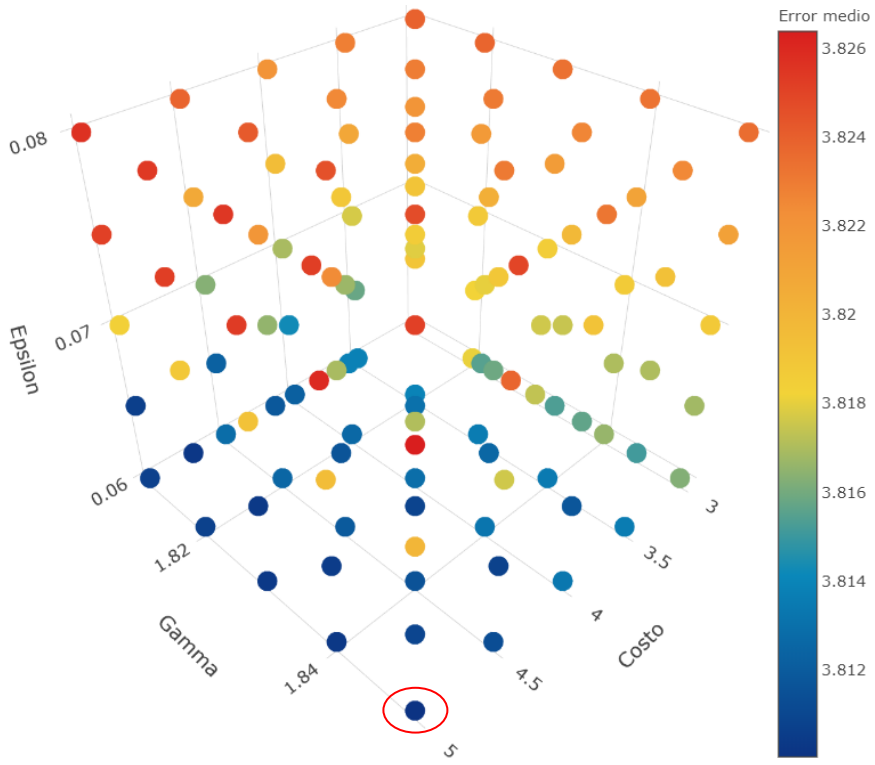
Como se puede notar en las tablas, los modelos con mejor RMSE son los de la búsqueda de malla. No podemos tomar estas combinaciones de hiperparámetros como nuestro resultado final porque esas mismas combinaciones tienen un error significativamente diferente en los modelos completos. Así, evitamos el *overfitting* del modelo. Cabe aclarar, que el RMSE basado en *cross-validation* puede llegar tan bajo como 3.72 promedio, sin embargo, cuando se compara con un modelo completo, el RMSE de éste es de 4.02. Esto refleja que el modelo no tiene la generalización deseada, sino que se sobreajustó hacia el conjunto de entrenamiento.

Al final, lo que nos interesa es conseguir el modelo completo con mejor RMSE que no se separe tanto del desempeño que mostró la misma combinación de hiperparámetros en la búsqueda

de malla. Así, primero tomaremos al modelo completo con mejor desempeño. Si el RMSE promedio de la búsqueda aleatoria es muy similar al RMSE del modelo, podremos asegurar que esta combinación de hiperparámetros tiene un grado de generalización suficiente que refute la idea de *overfitting* hacia cualquiera de los dos conjuntos.

En primer lugar, el mejor desempeño en los modelos completos es un RMSE de 3.810042 correspondiente a la combinación número 25. Dicha combinación se compone de un costo de cinco, una gamma de 1.85 y una épsilon de 0.06. Si buscamos esa combinación en la Ilustración 39, observaremos que su valor es cercano al 3.81. Pero, para estar seguros, buscamos en el registro de los RMSE promedios de la búsqueda de malla, en el lugar 25 y se tiene: 3.812983. Como se observa, la diferencia entre estas dos cantidades es de solamente un 0.08%, lo cual no es significativo. Así que se toma este modelo como modelo final del procedimiento. Para ubicar dónde se encuentra este modelo se muestra la siguiente gráfica:

Ilustración 41: Gráfica de desempeño de los modelos completos fase 4



Elaboración propia.

Así, los hiperparámetros del modelo final son los siguientes:

- Costo = 5.
- Gamma = 1.85.
- Épsilon = 0.06.

Ahora bien, tenemos un valor de RMSE final de 3.810042, pero qué tan cercano está esto de lo que obtendríamos en un escenario real con nuevos datos. En primer lugar, desglosaremos los RMSE's del proceso de *cross-validation* para ver qué se obtuvo ahí:

Tabla 17: Resultados de la cross-validation de la búsqueda de malla en la combinación 25

Fold	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10
RMSE	4.1179	3.7389	3.7487	3.8707	3.8991	3.9617	3.5795	3.7786	3.42182	4.0124

Elaboración propia.

Como se puede notar, las medidas de error que se obtuvieron fueron muy variadas, pero se promediaron en 3.813. Sin embargo, existen valores de error tan bajos como 3.4182, o tan altos como 4.1179. El hecho de que el RMSE del modelo completo se ubique tan cerca del promedio resultante de la búsqueda de malla nos da una seguridad mayor de que la generalización del problema es alta. Sin embargo, a lo largo de nuevas predicciones se pueden esperar errores tan altos o tan bajos como los mostrados en la *cross-validation*, lo importante es que el promedio de esos errores se mantendrá siempre cerca del 3.81. Ahora es un momento adecuado para mencionar que la combinación de hiperparámetros que se eligió no es una optimización del modelo. Los métodos que empleamos (búsqueda aleatoria y de malla) no aseguran encontrar la mejor combinación de hiperparámetros posibles, sin embargo, encontrar estos valores requeriría de un tiempo y poder de cómputo muy grande. Dado lo que logramos observar en los espacios de error, no estamos muy alejados de la solución más balanceada que existe. Claro que se podría mejorar el RMSE, pero muy posiblemente sería en una medida mínima y con un uso de recursos que no justificaría la mejora. Así, una situación importante al momento de buscar hiperparámetros es saber cuándo parar.

Una vez que tenemos el modelo completo, es hora de profundizar en el análisis que se realizó. Nuestro enfoque inicial al ir modificando el valor de los hiperparámetros es obtener un mejor desempeño para nuestro modelo. No obstante, al momento de modificarlos, se está cambiando la curva de regresión de una forma específica. Ya hablamos de los efectos de épsilon en la curva, ahora es tiempo de los otros dos hiperparámetros. Si observamos las primeras búsquedas, notamos que los factores más influyentes en los modelos fueron épsilon y gamma. En este caso, cuando el valor de gamma es elevado, las curvas en forma de campana de la función de base radial gaussiana se vuelven más estrechas. Al presentarse este efecto, la curva de decisión se vuelve más propensa al *overfitting*, pues estas curvas en forma de campana se ajustan cada vez más a los ejemplos que se le presentan. Es por ello que fue necesario reducir su magnitud, de forma que la regulación del modelo fuera mayor y, por tanto, también su generalización. Por tanto, nuestro modelo se compone de curvas de campana más amplias que toman en cuenta la posición general de los puntos, en lugar de dejarse influir por cada uno de ellos. Al final de las búsquedas,

gamma fue uno de los hiperparámetros que menos influencia tuvo, debido a que los valores que se le otorgaron no modifican mucho la curvatura de las campanas.

En el caso del costo, éste es un hiperparámetro de regulación que decide el balance entre permitir o no violaciones de margen. Cuando su valor es elevado, su regulación es baja, por lo que permite muchas violaciones de margen. Por otro lado, si su valor es reducido, se encuentra muy regulado y no permite violaciones de margen. En el primer caso, es causa de *underfitting* y en el segundo de *overfitting*. Por ello es que hay que encontrar un balance en su valor. En nuestro caso, se mostró cómo al inicio los valores muy elevados del costo generaban pésimos desempeños, por lo que se redujo su valor (esto era un *underfitting*). Por otro lado, cuando se restringió su valor aún más, al estar muy cercano a cero, los resultados también decayeron (ahora era un *overfitting*). Al final nos quedamos con un valor de cinco, el cual no es ni tan alto ni tan bajo: es un balance que nos permite tener una generalización alta.

Y bien, por último, en este capítulo, se decidirá si los resultados obtenidos valieron o no la pena. Ya validamos el modelo en cuanto a su uso general, pero ahora se debe recordar que antes de este trabajo existía el Tüfekci. Hay que empezar por lo más obvio, ¿el RMSE que se obtuvo supera a los que obtuvo Tüfekci? La repuesta es un claro no. El algoritmo BREP supera el RMSE que se obtuvo en este trabajo. Sin embargo, no existió otro algoritmo que lo hiciera. En cuanto a la comparación del algoritmo directa con el SVM de Tüfekci, en este trabajo se obtuvo un RMSE muy superior al que él tuvo que fue de 4.563. De hecho, ese valor se superó con la primera búsqueda aleatoria. Lo que demuestra que con un poco de trabajo más de parte de Tüfekci, pudo obtener desempeños mejores en cada uno de los algoritmos.

Como mencionamos antes, las diferencias metodológicas entre uno y otro estudio hacen difícil su comparación directa. La diferencia entre el RMSE que obtuvo Tüfekci y el de este estudio es muy baja, lo que no es tan comparable es el grado de generalización que se obtuvo aquí comparado con el que él obtuvo. Él reutilizó los mismos datos en repetidas ocasiones para el entrenamiento del modelo, mientras que nosotros dejamos el conjunto de prueba totalmente fuera del entrenamiento del modelo. De esta forma, nuestra simulación de nuevos datos es más confiable que la que él realizó. Nuestra medida de error puede ser más alta, pero sabemos que se mantendrá siempre que los datos que se le inserten pertenezcan a la misma distribución subyacente con la que se le entrenó. Mientras que Tüfekci no tiene esa misma seguridad, puede ser que su modelo tenga un alto grado de generalización o puede que no lo tenga, la cuestión es que no tiene certeza de ello.

Por todo lo que se mencionó anteriormente, se concluye que la validez del modelo que se obtuvo es suficiente para justificar su desarrollo, aun cuando no se logró superar el error que Tüfekci obtuvo en su estudio.

3.4 Consideraciones Futuras

Hasta este momento tenemos una metodología para recolectar datos, limpiarlos y encontrar un modelo lo más balanceado posible. Sin embargo, para quien quiera usar el modelo la pregunta obligada es: ¿cómo lo uso? Bueno, pues ésta sería una pregunta muy sencilla. Lo que se hace es tomar las condiciones ambientales que se esperan a determinada hora del día y se le ordena al modelo que haga una predicción. Cabe recordar que este modelo está pensado para usarse con una semana de anticipación al momento en que se requiere la energía. Entonces, simplemente se le ingresan esos datos al modelo y se encuentra el pronóstico. En este momento, un lector ávido se preguntaría: ¿cómo puedo saber qué condiciones ambientales se presentarán al momento que deseo generar la energía? Y esa sería una excelente pregunta.

La solución que propuso Tüfekci fue emplear los pronósticos del servicio meteorológico local para estimar las condiciones que se presentarán en el momento que se generará energía. Esta solución parece que le funcionó en la planta de donde obtuvo los datos. Pero no parece ser una forma de resolución suficientemente adecuada. Al emplear un pronóstico como entrada al modelo, el espacio de variables se modifica, por lo que el algoritmo puede no estar entrenado para arrojar estimaciones con la misma precisión. En primer lugar, el modelo se entrena con datos que se midieron dentro de la planta, por más que el ciclo Brayton tome aire ambiental, las condiciones dentro de una planta y fuera de ella no son las misma. El servicio meteorológico no va a tener un pronóstico de las condiciones dentro de la planta, sino en la zona en que se ubica. Ahora, el pronóstico del servicio meteorológico y las condiciones reales en la planta pueden tener una correspondencia lineal o no lineal entre sí, de modo que el algoritmo podría acercarse a una estimación suficientemente precisa para la planta usando estas entradas.

Otra situación que se debe tener en cuenta es que de la forma que Tüfekci propone usar el modelo se está haciendo un pronóstico a partir de un pronóstico. Los valores que se midieron en la planta tienen un ruido natural en ellos por el simple hecho de que no hay sensor que pueda medir en una escala absoluta todo el tiempo. Siempre habrá pequeñas variaciones en sus mediciones que dependan de eventos aleatorios desde niveles a gran escala como a nivel atómico. Este ruido en un inicio es absorbido por el modelo. Pero al depender de otro pronóstico, el nivel de ruido se incrementa. Ahora no sólo se tiene el error de los sensores al entrenar el algoritmo, sino que se tiene el error del pronóstico.

De esta forma, se tienen tres formas de poder usar el modelo:

- Pronóstico del servicio meteorológico: Esta solución es de la que se ha estado hablando y consiste en usar el modelo directamente con el pronóstico de las condiciones ambientales que proporciona el servicio meteorológico local.
- Entrenamiento con pronóstico: Hasta este momento ya pudimos observar que, usando como variables de entrada a las condiciones ambientales medidas en el lugar, es posible predecir la carga eléctrica total de una planta de CC. Sin embargo, si estas mediciones

tienen una relación con los pronósticos del servicio meteorológico, entonces se podrían reemplazar las mediciones dentro de la planta por las predicciones ambientales del servicio como variables de entrada. De esta forma, el algoritmo se entrenaría para adecuarse al error de las estimaciones de las condiciones ambientales, lo absorbería. Para esta solución se propondría tener un periodo en donde se hagan las mediciones dentro de la planta y se tuviera un registro de los pronósticos del servicio meteorológico de modo que se observe qué tan cercanos se encuentran dichos valores: si el error del pronóstico no es muy grande y si los cambios en uno se reflejan en el otro.

- Modelo de pronóstico para condiciones ambientales: La última solución que se tiene es muy parecida a la primera. En lugar de depender de un pronóstico de la zona, se debería de realizar un pronóstico propio para cada variable de entrada. Así, cuando se elabora la base de datos, se tienen elementos para entrenar a todos los distintos modelos. Esto es conveniente porque se estarían prediciendo las medidas que los sensores arrojarían dentro de la planta. Sin embargo, es demasiado trabajo hacer un modelo confiable para cada una de las variables de entrada, independientemente del método que se elija ellas.

  Todo esto queda por ser una incógnita y, para este trabajo, es una consideración futura.

Otra pregunta que se hará el lector o quien esté interesado en esta metodología es una pregunta recurrente para cualquier problema que se quiera resolver con ML. ¿Cuántos datos son necesarios para que el modelo tenga una alta precisión? Bueno, pues la respuesta en el medio de la ciencia de datos es que no hay un número de datos definidos, que la cantidad de datos depende de cada caso. Por ejemplo, para reconocimiento de imágenes se requieren de centenas de miles a millones de ejemplos para tener una base adecuada para el entrenamiento. Para regresiones o clasificaciones con bases de datos tradicionales el requerimiento puede ir desde los cientos hasta las centenas de miles, pero por lo general se ubica en los miles.

Para nuestro caso, no se tiene ni me atreveré a proponer un número mínimo, porque todo depende de cómo se recabe la base de datos. En los escritos de Tüfekci nunca se justifica el por qué del tamaño de la base de datos: si el tamaño actual es el mínimo de datos necesarios, si tiene definido un límite mínimo menor, o si la base es de ese tamaño porque fue cuando terminó su permiso dentro de la planta. La pregunta es muy difícil, pero a continuación se otorgará una guía de cómo se podría determinar dicho número.

En primer lugar, la variación que se pueda encontrar en una base de datos es un factor muy importante. Puede que se tenga miles de datos, pero si los estados que presentan los datos son muy diferentes entre sí (muy dispersos), no se tendrán los suficientes ejemplos para entrenar al algoritmo. Por lo que, para una planta de CC, podrían ser suficientes cien mediciones de datos siempre y cuando la variación entre ellos no sea tan alta, por ejemplo, que se tomen a la misma hora del día en una zona en donde las condiciones no se modifiquen muchos. Sin embargo, este modelo sólo funcionaría correctamente al intentar predecir la carga eléctrica en esa misma hora, a diferentes horas con diferentes condiciones muy seguramente no podrá hacer un pronóstico exacto.

La variación es uno de los problemas que hacen necesarias bases de datos enormes, pero también es necesaria para poder entrenar algoritmos flexibles que sean capaces de dar una respuesta adecuada a escenarios muy distintos entre sí. Por ello, más allá de proponer un número fijo de ejemplos necesarios, se recomienda tener un periodo de un mes o quince días en el que se hagan la mayor cantidad de mediciones posibles durante el día con una variación alta en las condiciones ambientales. Cabe recordar que las plantas de CC son despachables y firmes, por lo que se puede controlar hasta cierto punto su salida de energía; para las situaciones que esta metodología es válida es cuando se trabaja al 100% o muy cercano a éste. Por lo que las mediciones sólo se deben realizar mientras la planta funciona a su carga total.

Una vez que termina el periodo de prueba, entrenar un algoritmo con los hiperparámetros por defecto que tiene el modelo. Si se observa que el error cumple con las expectativas, se tendrá la respuesta del número mínimo de datos. Lo que se recomienda seguir haciendo es continuar con la medición y ampliando la base de datos. Así, cada mes se puede hacer un nuevo modelo con los hiperparámetros por defecto e ir colocando los resultados en una gráfica cuya variable independiente sea el número de datos de entrenamiento y su variable dependiente sea la medida de error que se presentó. Llegará el momento en que el desempeño del modelo ya no mejorará, o peor, que vaya decrementando. Para el primer caso, cuando el desempeño se mantiene estático o su mejora es insignificante, la primera cantidad de datos que tuvo dicho rendimiento será el mínimo de datos. Para el segundo caso, cuando el desempeño del modelo decrece, significa que el modelo está sufriendo de *overfitting* y, de igual forma, la cantidad de datos mínima será el último punto en donde aumentó el desempeño. De cualquier forma, cuando se tiene el mínimo de datos es adecuado iniciar con la búsqueda de hiperparámetros, lo más seguro es que si se hace con los modelos anteriores, todos terminarán con un sobreajuste.

Una vez que se tiene una solución a esa pregunta que atormenta a todos los científicos de datos al iniciar un nuevo proyecto, se continuará con otra que lo hace constantemente. ¿Una vez que se tiene el modelo balanceado, se terminó el trabajo? La verdad es que la mayor parte de los sistemas en que se emplea ML son dinámicos y nunca conservan las mismas propiedades a través de largos periodos de tiempo. Por lo que es necesario actualizar el modelo constantemente. Primero, ¿cuándo es necesario actualizar el modelo? Bueno, esta pregunta depende del nivel de error que se espera que tenga el algoritmo. Si en repetidas ocasiones en un periodo de tiempo corto se tiene un error más alto que el esperado, es una señal de que es necesario actualizar el algoritmo. Para determinar esto sería conveniente usar las técnicas de confiabilidad que emplean muchos sistemas de calidad, en este caso, contaríamos como defectos cuando nuestro error se encuentra por encima de un límite establecido.

Ahora, ya que decidimos que se requiere actualizar el modelo, ¿cómo se hace? En el capítulo de “Fundamentos de ML” observamos dos formas: en línea y por lotes. La forma más sencilla sería por lotes, de forma que cada vez que empiece a fallar el algoritmo, lo volvemos a entrenar desde el inicio, tal vez descartando algunos de los datos que ya no sean compatibles con

los más recientes. La actualización en línea es más complicada porque nunca se crea un nuevo modelo, sino que se va actualizando el inicial con los nuevos ejemplos que se tienen. Para esto es necesario especificar un nuevo parámetro que decide cuánta importancia darles a los nuevos ejemplos. Se pueden hacer modelos con mucha inercia, de modo que los nuevos ejemplos no influyan demasiado; o con poca inercia, de modo que los nuevos datos moldeen el modelo. Todo esto depende de la aplicación. Ahora, al hablar de una planta de CC es necesario actualizar al modelo debido al desgaste que sufre la maquinaria, así, un análisis muy complejo de desgate se simplifica con sólo actualizar la base y el modelo.

Por último, una consideración futura es dejar de emplear paralelismo con el CPU, a menos que se tenga uno con una enorme cantidad de núcleos (como 28 o más); y empezar a usar paralelismo con un GPU. Está demostrado que los núcleos especializados del GPU son mucho más eficientes que los núcleos del procesador. De esta forma, se pueden hacer los análisis más rápidamente y hay una mayor flexibilidad para hacer búsquedas más extensivas.

Por ahora, se termina la presentación del caso. Más adelante se encontrarán las conclusiones a las que se llegaron, las referencias en las que se basó este trabajo y algunos anexos que proporcionan información que no se detalló en el escrito.

Conclusiones

Antes de iniciar las conclusiones, se comentará que el trabajo realizado es muy extenso. A lo largo de éste se tocan temas muy diversos que resulta imposible englobarlos todos en esta sección. En la "Presentación del Caso" se hizo un intento por abarcar a detalle cada aspecto de esta investigación, por lo que se recomienda que si existe una duda en las conclusiones se consulte la sección pasada, muy seguramente se encontrarán las repuestas ahí. De esta forma, todo lo que se mencione aquí será lo más conciso y general posible.

Comenzaremos por donde todo el mundo inicia: ¿se cumplió el objetivo? A lo que se tiene que responder que se cumplió el objetivo exitosamente. En este trabajo se logró desarrollar un modelo altamente exacto para pronosticar la carga eléctrica total de una planta de ciclo combinado basado en las condiciones ambientales del lugar de operación. Pero lo que tiene un mayor valor fue conformar una metodología para consolidar dicho modelo y validar su generalización cuando se le presenten nuevos datos. De modo que el modelo se comporta de la forma en que se espera que se comporte cuando se le ingresen datos totalmente nuevos (que se compongan de las mismas variables y tengan la misma distribución de probabilidad subyacente).

Ahora bien, ¿cuál es el motivo de hacer este trabajo? Bueno, la motivación proviene de los cambios recientes que ha sufrido México en la estructura de su industria eléctrica. En 2013 se aprobó la Reforma energética que conllevó la creación de la LIE, la cual modifica todo el sector eléctrico del país, pero, sobre todo, dos de sus grandes conformantes: la generación y la comercialización de la energía. Para estos dos sectores, los cambios radican en que ya no forman parte de la cadena de valor de CFE exclusivamente, sino que el mercado se abre a productores independientes de energía y a prestadores de servicios y comercialización en un nuevo mercado. Dicho mercado es controlado por dos entidades: el CENACE y la CRE. Estas entidades son responsables de mediar el MEM y de administrar el SEN, por lo que una de sus responsabilidades es asignar a cada productor independiente la cantidad de energía que se le comprará para alimentar el SEN. Estas asignaciones están basadas en los pronósticos de demanda que el CENACE realiza para cada hora del día en toda la extensión de país, y en un modelo de programación lineal que le permite analizar a cuál productor le conviene comprar la energía.

Antes de comprar energía a cualquier generador, el CENACE le pide a cada uno de ellos un pronóstico de la energía máxima que le pueden ofrecer para un día y hora determinados. En el caso de las plantas de CC, al considerarse como firmes y despachables, lo que se entrega es la capacidad instalada de la planta. Así, el CENACE muy seguramente les asignará a estas plantas una cantidad por debajo de su capacidad instalada (con un factor de seguridad), de forma que la planta sea capaz de producir dicha cantidad de energía.

El problema que presenta este enfoque (que está establecido en la regulación) es que, cuando las plantas de CC trabajan a su máxima capacidad o muy cerca de ella, existen diferentes

variables que afectan su rendimiento, de modo que no alcanzan el valor de su capacidad instalada. Es aquí donde entra la segunda hipótesis (H2) de este trabajo, en donde se especula si las condiciones ambientales del lugar de operación son factor para la cantidad de energía que genera una planta de CC. Cabe mencionar que esta hipótesis no es totalmente ciega, sino que estudios anteriores han demostrado que ciertos factores ambientales afectan de manera significativa el accionar de las turbinas de gas. Por lo que, si se conocen estos efectos, ¿por qué la regulación no los tiene en cuenta? Esa es una pregunta que no soy capaz de responder con certeza, pero que más adelante se propondrán razones.

Así, llegamos a nuestro estudio, el cual está basado en uno anterior realizado por Tüfekci. En este último, se realizó un meta-análisis de algoritmos de Machine Learning para predecir la carga eléctrica total de una planta de CC ubicada en Turquía. De este estudio se extrajo la base de datos que recolectó Tüfekci durante seis años y que contempla 9568 datos distintos con cuatro variables de entrada y una variable objetivo. Al analizar la base de datos se notó que el promedio de la carga eléctrica total de la planta era de 454.4 MW, pero su capacidad instalada es de 480 MW, lo que hace notorio que existen factores que afectan el desempeño de la planta y no se puede confiar únicamente en la capacidad instalada. Se observó que las condiciones ambientales que afectan más a la generación de energía es la temperatura ambiente y la presión de salida del vapor de la turbina de vapor.

Con el conocimiento de que los factores ambientales afectan la generación de energía, lo siguiente fue elegir cómo pronosticar la energía generada a partir de las condiciones ambientales. La forma tradicional en que este procedimiento se lleva a cabo es mediante un análisis de ecuaciones diferenciales parciales para sistemas termodinámicos complejos. Este enfoque es muy demandante en cuanto a tiempo y recursos computacionales, además de que a veces no se encuentran soluciones al sistema de ecuaciones. Lo que nos llevó al camino de ML. ML permite hacer análisis de sistemas complejos siempre y cuando se cuente con una cantidad enorme de datos. Estos análisis son mucho menos demandantes que el enfoque tradicional y permiten adaptarse al desgate de la maquinaria. El algoritmo específico que se eligió fueron las Máquinas de Vectores de Soporte (SVM’s), ya que están basadas en modelos y presentan una alta flexibilidad. Por otra parte, al observar las complicaciones que representa el obtener un pronóstico exacto de la carga total de una planta CC, se puede comprender por qué no se incluye en la regulación.

Ahora, daremos paso a los beneficios que conlleva realizar un modelo de pronóstico en lugar de emplear la capacidad instalada para los generadores. En primera instancia, la aplicación más inmediata del modelo se encuentra en el mercado de corto plazo. En el MDA se requiere un pronóstico de hasta una semana en adelanto al momento en que se requiere la energía, en el MTR de quince minutos antes; por lo que si se conocen las condiciones ambientales dentro de la planta en el momento en que se requiere la energía, se puede hacer un pronóstico altamente exacto con la SVM. Así, se evitará tener excesos de energía que, a largo plazo, se desperdiciarán; o faltantes de ella, que pueden significar multas u horas de trabajo extra como sanción impuesta por el

CENACE. En los mercados de largo y mediano plazo, los contratos se hacen con entes privados o con el CENACE para un suministro de energía en un contrato con duraciones de tres a quince años. Normalmente se establece una cantidad de energía base a surtir, pero si las condiciones ambientales no son adecuadas y se predice que en cierto momento no se llegará a cumplir con dicho acuerdo, se pueden tomar medidas con anterioridad para abastecer la energía prometida y no incumplir con el contrato. Por otro lado, si se observa con anterioridad que se tendrá un excedente de energía, se puede buscar colocar dicha energía en el mercado de corto plazo. En general, tener un pronóstico más exacto, permite disminuir pérdidas y maximizar los ingresos.

Ahora, la forma en que se comprobó la primera hipótesis fue mediante el desarrollo de modelos SVM's y un proceso de búsqueda de hiperparámetros para dichos modelos. Esta búsqueda tuvo dos objetivos: obtener un modelo con una medida de error aceptable y que éste contara con una generalización robusta y validada. El tipo de problema que se presenta en este trabajo es el adecuado para realizar un modelo de regresión: con la carga eléctrica total como variable objetivo y las condiciones ambientales medidas en la planta como variables de entrada. El modelo que se ocupó fue una regresión SVM con parámetro épsilon y *kernel* de función de base radial gaussiana, por lo que los hiperparámetros correspondientes son: costo, gamma y épsilon. El procedimiento se inició separando la base de datos en un conjunto de entrenamiento y uno de prueba con una partición 80-20; y se emplearon las técnicas de búsqueda aleatoria y búsqueda de malla conjuntadas con *cross-validation* de diez *folds*.

De esta forma, se corrieron 4250 modelos distintos con 425 combinaciones de hiperparámetros diferentes para monitorear el desempeño de los modelos respecto a los hiperparámetros. Sin embargo, a la vez que se hicieron estas búsquedas se propuso una medida de contrapeso con el fin de evitar que el modelo se sobreajustara al conjunto de entrenamiento. La medida fue crear modelos completos cuyo entrenamiento se diera con el conjunto de entrenamiento y cuya validación se diera con el conjunto de prueba para cada combinación de hiperparámetros. Así que en total se tuvieron 4675 modelos distintos. La forma en que se pudieron evaluar tantos modelos distintos, sin emplear una cantidad enorme de tiempo, fue asignando las corridas de los modelos en paralelo, empleando al máximo los recursos del CPU. De todos los modelos que se evaluaron, se escogió el más balanceado posible, cuyos hiperparámetros fueron:

Costo = 5.

Gamma = 1.85.

Épsilon = 0.06.

Un objetivo implícito en el trabajo fue la meta de superar la medida de error que Tüfekci consiguió en su estudio. Sin embargo, el RMSE de 3.81 al que se llegó no lo supera (el de Tüfekci fue de 3.787). Lo que se logró comprobar fueron las dos hipótesis planteadas: primero, las SVMS's construyeron modelos de alta exactitud para nuestro problema y a un bajo costo computacional; segundo, las condiciones ambientales del lugar de operación de la planta

resultaron muy buenos predictores de la energía generada por ésta. Cabe señalar que el RMSE no tiene un sentido de escala en cuanto a la magnitud de los datos que compara, por lo que, si calculamos el porcentaje de desviación esperado con el modelo de mejor desempeño, se tiene una desviación de alrededor de ±0.6%, la cual sin duda es una alta exactitud. No obstante, la mayor ventaja que tiene nuestro modelo respecto al BREP de Tüfekci es la generalización que logra por las razones expresadas en el capítulo de "Resultados y Análisis de Resultados". Por lo que se espera que el error de las predicciones que haga el modelo seleccionado promedie en un RMSE=3.81 con el uso de nuevos datos.

Ya que se tiene el modelo se puede referenciar al capítulo de "Consideraciones Futuras" para observar qué cambios se le pueden hacer. Por lo pronto, la metodología desarrollada se considera aplicable a cualquier planta de CC de México, lo que nos indica que el objetivo se cumplió. Es clave señalar que el modelo que se obtuvo no tiene como tal una aplicación práctica, sino que se empleó como ejemplo para el uso de la metodología. La metodología, por otra parte, es aplicable a cualquier problema en el que intervenga un conjunto masivo de datos y en el que se desee emplear SVM's. Simplemente se requerirían de unas pocas modificaciones específicas para cada problema. Se puede abordar un problema de clasificación, en lugar de uno de regresión, y hasta se puede emplear otro algoritmo de ML, considerando sus respectivos parámetros. Todo el tiempo que se invirtió en el desarrollo metodológico fue con el propósito de que los modelos resultantes tuvieran un grado de generalización alto y, por tanto, la calidad de las predicciones que hagan durante su entrenamiento sea muy similar a la que realicen cuando se le presenten nuevos datos.

Aquí se acaba todo el trabajo, se han expuesto las intenciones que se tenían al iniciar esta investigación y, en realidad, se considera que los resultados obtenidos han cumplido con las expectativas y hasta se han superado. Existen varios campos en los que se profundizó el análisis mucho más de lo que se había planeado, ya sea por necesidad o por el gran interés que despertó en mí la investigación. Por lo pronto, agradezco a quien se haya tomado el tiempo de leer este trabajo y, sobre todo, a quienes tuvieron la paciencia para permitirme terminarlo.

Referencias

Arrieta, F. R., & Lora, E. E. (2005). Influence of ambient temperature on combined-cycle. *Appl Energy*(80), 261-272.

Brownlee, J. (11 de Diciembre de 2017). *Difference Between Classification and Regression in Machine Learning*. Obtenido de Machine Learning Mastery: https://machinelearningmastery.com/classification-versus-regression-in-machine-learning/. Consultado: 15 de Julio de 2019.

CENACE. (17 de Junio de 2016). *Manual de Mercado de Energía de Corto Plazo.* Obtenido de https://www.cenace.gob.mx/Docs/MarcoRegulatorio/Manuales/Manual%20de%20Mercado%20de%20Energ%C3%ADa%20de%20Corto%20Plazo%20DOF%202016%2006%2017.pdf. Consultado: 12 de Junio 2019.

CENACE. (2017). *Mercado y Operaciones.* Obtenido de https://www.cenace.gob.mx/MercadoOperacion.aspx. Consultado: 12 de Junio 2019.

CENACE. (s.f.). *Subastas.* Obtenido de Mercado y Operaciones: https://www.cenace.gob.mx/Paginas/Publicas/MercadoOperacion/Subastas.aspx. Consultado: 12 de Junio 2019.

Çengel, Y. A., & Boles, M. A. (2015). *Termodinámica.* Ciudad de México: McGraw-Hill Education.

CIEP. (6 de Diciembre de 2017). *La Reorganización de la Industria Eléctrica en México: Las Consecuencias de la Reforma Energética.* Obtenido de https://es.scribd.com/document/366505214/La-Reorganizacion-de-la-Industria-Electrica-en-Mexico#fullscreen&from_embed. Consultado: 21 de Junio 2019.

CONAPO. (14 de Septiembre de 2018). *Población a inicio de año.* Obtenido de Proyecciones de la Población de México y de las Entidades Federativas, 2016-2050: https://datos.gob.mx/busca/dataset/proyecciones-de-la-poblacion-de-mexico-y-de-las-entidades-federativas-2016-2050/resource/b39477ce-b625-488d-875b-bd65ed120b6b. Consultado: 25 de Junio 2019.

Congreso de la Unión. (11 de Agosto de 2014). *Ley de la Industria Eléctrica.* Obtenido de Diario Oficial de la Federación: http://www.diputados.gob.mx/LeyesBiblio/pdf/LIElec_110814.pdf. Consultado: 23 de Junio 2019.

CRE. (2018). *Reporte anual del Mercado Eléctrico Mayorista.* Obtenido de CENACE: https://www.cenace.gob.mx/Docs/MercadoOperacion/ReporteAnual/2017/Reporte%20Anual%20del%20MEM%202017%20-%20Monitor%20Independiente%20del%20Mercado.pdf. Consultado: 23 de Junio 2019.

Garbade, M. J. (11 de Agosto de 2018). *Regression Versus Classification Machine Learning: What's the Difference?* Obtenido de Quick Code: https://medium.com/quick-code/regression-versus-classification-machine-learning-whats-the-difference-345c56dd15f7. Consultado: 15 de Julio de 2019.

Géron, A. (2017). *Hands-On Machine Learning with Scikit-Learn and Tensor Flow.* Sebastopol: O'Reilly Media.

Leisch, D. M. (2019). *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*. Obtenido de R package version 1.7-2: https://CRAN.R-project.org/package=e1071

MathWorks. (s.f.). *Understanding Support Vector Machine Regression*. Obtenido de MathWorks: https://www.mathworks.com/help/stats/understanding-support-vector-machine-regression.html. Consultado: 31 de Julio de 2019.

Microsoft Corporation and Steve Weston. (2019). *doParallel: Foreach Parallel Adaptor for the 'parallel' Package*. Obtenido de R package version 1.0.15: https://CRAN.R-project.org/package=doParallel

Mueller, J. P., & Massaron, L. (2016). *Machine Learning for Dummies.* Nueva Jersey: John Wiley & Sons.

Payan, T., Díaz, S. P., & Cossío, y. J. (2016). *Estado de Derecho y Reforma Energética en México.* Ciudad de México: Tirant Lo Blanch.

PwC. (8 de Septiembre de 2015). *Resumen de las Bases del Mercado Eléctrico.* Obtenido de https://es.scribd.com/document/306308897/PwC-Resumen-de-Las-Bases-Del-Mercado-Electrico. Consultado: 21 de Junio de 2019.

R Core Team. (2019). *R: A Language and Environment for Statistical Computing*. Obtenido de R Foundation for Statistical Computing: https://www.R-project.org/

Schölkopf, B. (1998). SVMs- a practical consequence of learning theory. *IEEE Intelligent Systems*, 18-21.

SENER. (12 de Enero de 2018). *Capitulo 6. Revisión de Ofertas.* Obtenido de Manual de Vigilancia del Mercado:

https://www.cenace.gob.mx/Docs/MarcoRegulatorio/Manuales/Manual%20de%20Vigilancia%20del%20Mercado%20DOF%202018%2001%2012.pdf. Consultado: 12 de Junio 2019.

SENER. (2018). *Pronóstico de consumo por región de control, escenario de planeación 2018-2032.* Obtenido de Prospectiva del Sector Eléctrico 2018-2032: http://base.energia.gob.mx/Prospectivas18-32/PSE_18_32_F.pdf. Consultado: 25 de Junio 2019.

Sievert, C. (2018). *plotly for R*. Obtenido de https://plotly-r.com

statcompute. (19 de Marzo de 2016). *Improve SVM Tuning through Parallelism*. Obtenido de R-bloggers: https://www.r-bloggers.com/improve-svm-tuning-through-parallelism/. Consultado: 1 de Octubre 2018.

Tüfekci, P. (2014). Prediction of full load electrical power output of a base load operated. *Electrical Power and Energy Systems*(60), 126-140.

University of California, Irvine. (septiembre de 2014). *Combined Cycle Power Plant Data Set.* Obtenido de Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/combined+cycle+power+plant#

USAID. (2013). *Greenhouse Gases in México.* Obtenido de https://www.climatelinks.org/sites/default/files/asset/document/2017_USAID_GHG%20Emissions%20Factsheet_Mexico_0.pdf. Consultado: 23 de Junio 2019.

Anexos

Anexo 1. Mercado del Día en Adelanto.

(Segunda Sección) DIARIO OFICIAL Viernes 17 de junio de 2016

Mercado del Día en Adelanto

4.1 Secuencia de Eventos en el Mercado del Día en Adelanto

4.1.1 A las 00:00 horas del séptimo día anterior al Día de Operación, el Sistema de Recepción de Ofertas se habilitará para recibir Ofertas para el Día de Operación.

4.1.2 Posteriormente, a las 10:00 del día anterior al Día de Operación se cerrará la ventana de recepción de Ofertas para el Mercado del Día en Adelanto y se iniciará el proceso de asignación y despacho de Unidades de Central Eléctrica en el Mercado del Día en Adelanto. Las ofertas recibidas en forma posterior a la hora de cierre, serán consideradas en los procesos de Asignación Suplementaria de Unidades de Central Eléctrica para Confiabilidad y en los procesos del Mercado de Tiempo Real.

4.1.3 A las 17:00 se publicarán los resultados de la asignación y despacho de Unidades de Central Eléctrica en el Mercado del Día en Adelanto, así como los Precios Marginales Locales del Mercado del Día en Adelanto y los precios para los Servicios Conexos asignados en el Mercado del Día en Adelanto.

4.1.4 La secuencia de ejecución de los eventos que conciernen al Mercado del Día en Adelanto se muestra en la Figura 2.

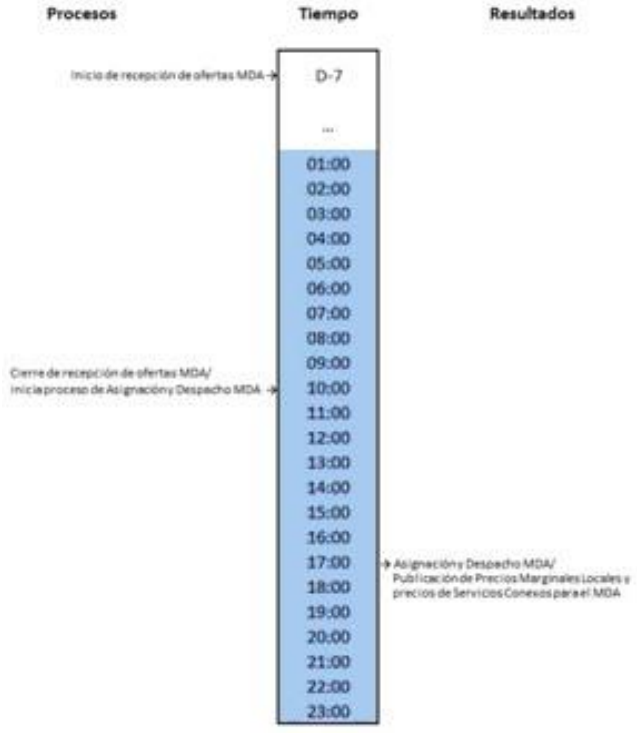


Figura 2. Secuencia de eventos en el Mercado del Día en Adelanto.

4.2 Recepción de Ofertas

4.2.1 Todos los días, los Participantes del Mercado presentarán sus Ofertas correspondientes al Mercado del Día en Adelanto al CENACE, durante el periodo de recepción de Ofertas, el cual estará abierto desde 7 días naturales previos al Día de Operación hasta las 10:00 horas del día anterior al Día de Operación.

4.2.2 El Sistema de Recepción de Ofertas dentro del periodo de recepción de Ofertas del Mercado del Día en Adelanto, enviará automáticamente una notificación de rechazo al Participante del Mercado, en caso de que su Oferta no cumpla con los requisitos establecidos en el presente Manual.

4.2.3 El CENACE notificará inmediatamente al Participante del Mercado a través del Sistema de Información del Mercado que su Oferta ha sido rechazada, identificando claramente la causa del rechazo.

4.2.4 En caso de que la Oferta cumpla con todos los criterios establecidos en este Manual, el CENACE notificará al Participante del Mercado que su Oferta ha sido validada y que es consistente con los Precios de Referencia, y por lo tanto el CENACE la aceptará para su consideración en el Mercado de Energía de Corto Plazo.

4.2.5 El Participante del Mercado podrá sustituir la Oferta enviada siempre y cuando ocurra dentro del periodo de recepción de Ofertas para el Mercado del Día en Adelanto.

4.2.6 Para la asignación y despacho de Unidades de Central Eléctrica en el Mercado del Día en Adelanto se tomará en cuenta solamente la última Oferta Validada y Consistente.

4.2.7 En caso de que se entreguen Ofertas no válidas o no se entreguen Ofertas antes de la conclusión del periodo de recepción de Ofertas del Mercado del Día en Adelanto, el CENACE utilizará la oferta por omisión basada en los Parámetros de Referencia registrados de la Unidad de Central Eléctrica, o en su defecto, los Parámetros de Referencia calculados por el CENACE. En el caso de las Entidades Responsables de Carga, el CENACE utilizará ofertas por omisión basadas en los Pronósticos de Demanda correspondientes, elaborados por el CENACE. El CENACE sólo usará ofertas por omisión en caso de que una Entidad Responsable de Carga elija la opción correspondiente. La oferta por omisión será vinculante y por lo tanto se considerará exigible para todos los efectos legales correspondientes.

4.2.8 En caso de que se entreguen Ofertas que no sean consistentes con los Precios de Referencia o que rebasen la oferta tope o la oferta piso establecidas por la Autoridad de Vigilancia del Mercado, el CENACE aplicará los Precios de Referencia correspondientes a la Unidad de Central Eléctrica para su consideración en el Mercado de Energía de Corto Plazo.

4.3 Asignación y despacho de Unidades de Central Eléctrica en el Mercado del Día en Adelanto

4.3.1 El CENACE iniciará diariamente el proceso de asignación y despacho de Unidades de Central Eléctrica en el Mercado del Día en Adelanto al cerrar el periodo de recepción de Ofertas para el Mercado del Día en Adelanto según el huso horario predominante en el sistema interconectado de que se trate. Los resultados de dicho proceso serán publicados por el CENACE en el Sistema de Información del Mercado a las 17:00 horas.

4.3.2 El CENACE llevará a cabo el proceso de asignación y despacho de Unidades de Central Eléctrica en el Mercado del Día en Adelanto para cada sistema interconectado en forma independiente.

4.3.3 El CENACE preparará con antelación los siguientes insumos para un día natural dividido en intervalos horarios, a fin de utilizarlos en el modelo AU-MDA.

Anexo 2. Mercado en Tiempo Real.

DIARIO OFICIAL
(Segunda Sección) Viernes 17 de junio de 2016

Capítulo 6

Mercado de Tiempo Real

6.1 Secuencia de Eventos en el Mercado de Tiempo Real

6.1.1 A las 22:00 horas del día anterior al Día de Operación, se conocerá el último programa horario de arranques, paros y cambios de configuración de las Unidades de Central Eléctrica para el Día de Operación preparado en el proceso de Asignación Suplementaria de Unidades de Central Eléctrica para Confiabilidad.

6.1.2 Posteriormente, cada cinco minutos se toma una muestra de los resultados del Estimador de Estado para identificar la topología de la red eléctrica que está operando, y conocer los valores estimados de las potencias de generación de las Unidades de Central Eléctrica, la demanda nodal y el flujo de potencia en las interconexiones internacionales.

6.1.3 Asimismo, cada quince minutos se utiliza la función de análisis de seguridad para determinar los límites de transmisión para las condiciones observadas en la muestra más reciente del Estimador de Estado.

6.1.4 Cada quince minutos el CENACE utilizará las funciones de Pronóstico de Demanda y pronóstico de generación intermitente para determinar sus valores esperados para los siguientes diez intervalos de quince minutos, tomando como punto de partida la demanda y la generación intermitente observada en la muestra más reciente de los resultados del Estimador de Estado.

6.1.5 Las Ofertas de Venta de energía y Servicios Conexos para el Mercado de Tiempo Real serán tomadas de la última Oferta Validada y Consistente con los Precios de Referencia de la Unidad de Central Eléctrica. Los ajustes a las Ofertas deberán entregarse 2 horas antes del inicio de la Hora de Operación.

6.1.6 Quince minutos antes del inicio de la Hora de Operación, el CENACE publicará la actualización del programa de arranques, paros y cambios de configuración de las Unidades de Central Eléctrica para la Hora de Operación, con detalle de quince minutos, preparado con base en los resultados del modelo AU-TR. Las decisiones que implican una obligación física se asientan en el Registro de Instrucciones de Despacho.

6.1.7 Cinco minutos antes de cada intervalo de quince minutos, dentro de la Hora de Operación, el CENACE publicará la asignación de reservas para suministrar los Servicios Conexos comerciados en el mercado y los puntos base del nivel de generación para las unidades que no operarán bajo el CAG centralizado en el intervalo de quince minutos. Las decisiones se determinan mediante el modelo DERS-MI. Las decisiones que implican una obligación física se asientan en el Registro de Instrucciones de Despacho.

6.1.8 Inmediatamente antes del inicio de cada intervalo de cinco minutos, dentro de la Hora de Operación, mediante el modelo DERS-I, el CENACE calculará los puntos base del nivel de generación de las Unidades de Central Eléctrica que estarán bajo el CAG centralizado, y los factores de participación económicos para aquellas unidades que participan en el despacho económico y en el servicio de Regulación Secundaria de Frecuencia. Esta información será suministrada al CAG para ser utilizada durante el intervalo de cinco minutos.

6.1.9 La secuencia de ejecución de los modelos utilizados para tomar las decisiones rutinarias que conciernen al Mercado de Tiempo Real, se muestran en la Figura 3. Los nombres de los modelos tienen como sufijo la hora y el minuto en el que inicia el periodo para el que toman decisiones. Cuando el periodo tiene divisiones, éstas son señaladas. Se indica el tipo de decisiones que toman, por ejemplo: arranque, paro, asignación de reservas, despacho, entre otras. Con colores distintos se identifica si se trata de decisiones que se convierten en obligaciones de entrega física o sólo son decisiones que serán informadas de forma preventiva a los Participantes del Mercado. Por último, se muestran en forma aproximada los momentos en que inicia la recolección de datos y en el que termina el proceso con la emisión de las instrucciones correspondientes a las decisiones de cada modelo.

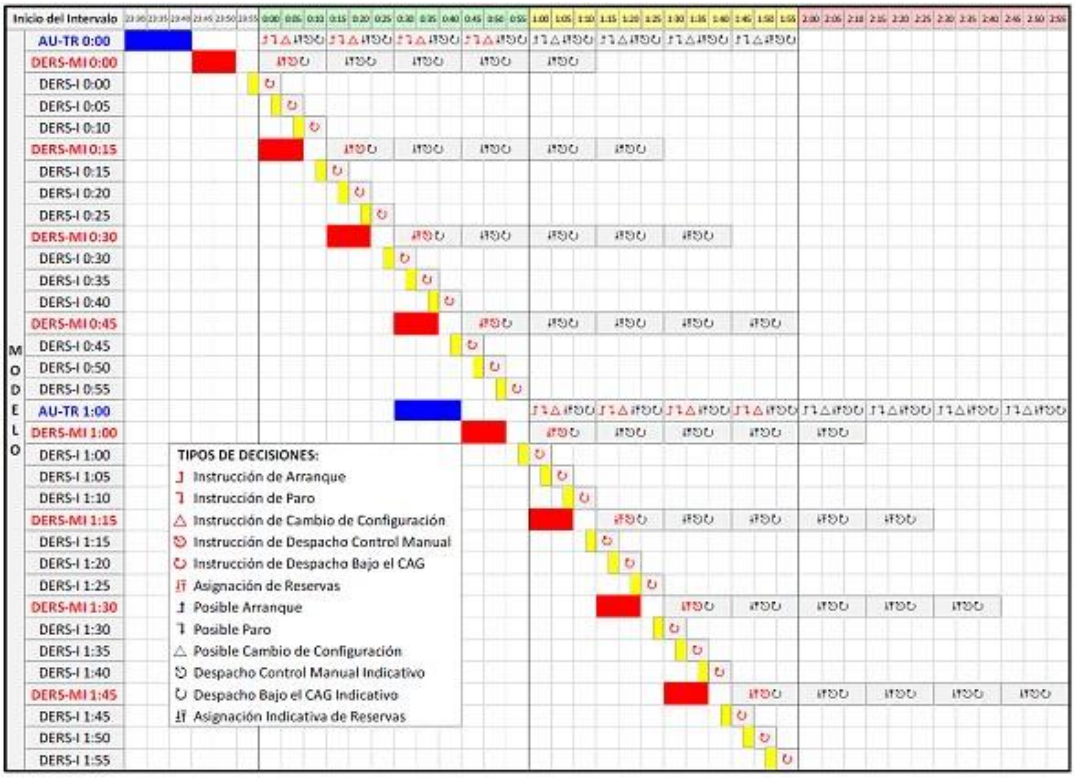


Figura 3. Secuencia de ejecución de modelos para el Mercado de Tiempo Real.

6.2 Recepción de ofertas

6.2.1 Todos los días, los Participantes del Mercado presentarán sus Ofertas de Venta correspondientes al Mercado de Tiempo Real al CENACE, durante el periodo de recepción de ofertas, el cual estará abierto desde 7 días antes del día de operación y hasta 2 horas antes de la Hora de Operación.

6.2.2 Los Participantes del Mercado podrán especificar que una misma Oferta de Venta puede ser utilizada para todos los Intervalos de Despacho del Día de Operación.

6.2.3 Los parámetros de la Oferta de Venta del Mercado de Tiempo Real que podrán ser distintos de la Oferta de Venta presentada en el Mercado del Día en Adelanto son los siguientes:

(a) Estatus de asignación;

(b) Límite de Despacho Económico Máximo y Límite de Despacho Económico Mínimo; y,

(c) Oferta de disponibilidad de reservas.

6.2.4 Los Participantes del Mercado podrán especificar que una modificación de la Oferta de Venta surtirá efectos a partir de cierto Intervalo de Despacho del Día de Operación, el cual deberá establecer cuando presente su Oferta de Venta.

6.2.5 El Sistema de Recepción de Ofertas enviará automáticamente una notificación de validación y asignará el estatus de Oferta Validada a las Ofertas que cumplan los requisitos establecidos en el presente Manual. Una vez validada, la Oferta de Venta será exigible para todos los efectos legales correspondientes.

6.2.6 El Sistema de Recepción de Ofertas dentro del periodo de recepción de ofertas del Mercado de Tiempo Real, enviará automáticamente una notificación de rechazo al Participante del Mercado, en caso de que su Oferta no cumpla con los requisitos establecidos en el presente Manual.

6.2.7 El CENACE notificará inmediatamente al Participante del Mercado a través del Sistema de Información del Mercado que su Oferta ha sido rechazada, identificando claramente la causa del rechazo.

6.2.8 En caso de que la Oferta cumpla con todos los criterios establecidos en este Manual, el CENACE notificará al Participante del Mercado que su Oferta ha sido validada y que es consistente con los Precios de Referencia, y por lo tanto, el CENACE la aceptará para su consideración en el Mercado de Tiempo Real.

6.2.9 El Participante del Mercado podrá sustituir la Oferta enviada siempre y cuando ocurra dentro del periodo de recepción de ofertas para el Mercado de Tiempo Real.

6.2.10 El CENACE utilizará la última Oferta Validada y Consistente con los Precios de Referencia para la Asignación de Unidades de Central Eléctrica en Tiempo Real y para el Despacho Económico con Restricciones de Seguridad.

6.2.11 En caso de que se entreguen Ofertas de Venta no válidas o no se entreguen Ofertas de Venta antes de la conclusión del periodo de recepción de ofertas del Mercado de Tiempo Real, el CENACE utilizará la oferta por omisión basada en los Parámetros de Referencia registrados de la Unidad de Central Eléctrica, o en su defecto, los Parámetros de Referencia calculados por el CENACE. La oferta por omisión será vinculante y por lo tanto se considerará exigible para todos los efectos legales correspondientes.

6.2.12 En caso de que se entreguen Ofertas que no sean consistentes con los Precios de Referencia o que rebasen la oferta tope o la oferta piso establecidas por la Autoridad de Vigilancia del Mercado, el CENACE aplicará los Precios de Referencia correspondientes a la Unidad de Central Eléctrica para su consideración en el Mercado de Tiempo Real.

6.2.13 Los operadores de las Unidades de Central Eléctrica notificarán de inmediato cualquier cambio en la disponibilidad o en los planes operativos de sus recursos para el Día de Operación. Los cambios en los parámetros que deberán informar son los siguientes:

(a) Disponibilidad de la unidad (disponible o no disponible);

(b) Límite de Despacho Económico Máximo y Límite de Despacho Económico Mínimo;

(c) Límite de Regulación Máximo y Límite de Regulación Mínimo;

(d) Disponibilidad para proporcionar el Servicio Conexo de Reserva de Regulación Secundaria de Frecuencia (disponible o no disponible).

6.2.14 Los operadores de las Unidades de Central Eléctrica utilizarán el Registro de Instrucciones de Despacho para notificar los cambios al CENACE. Los cambios notificados permanecerán vigentes hasta que los operadores de las Unidades de Central Eléctrica notifiquen el siguiente cambio.

6.3 Muestreo de los Resultados del Estimador de Estado

6.3.1 El CENACE tomará cada cinco minutos el caso más reciente resuelto por el Estimador de Estado, con la finalidad de extraer la información que caracteriza la red eléctrica que está en servicio, los valores de las potencias de generación de las Unidades de Central Eléctrica conectadas a la red, la demanda nodal y los flujos de potencia activa en elementos de la red específicos.

6.3.2 Estos datos servirán como insumo para Pronósticos de Demanda y pronósticos de generación intermitente, para determinar las restricciones de seguridad del despacho o para conocer el nivel de generación de las unidades y los flujos de potencia activa en las líneas de interconexión con sistemas vecinos.

6.4 Pronósticos de Demanda y de generación intermitente

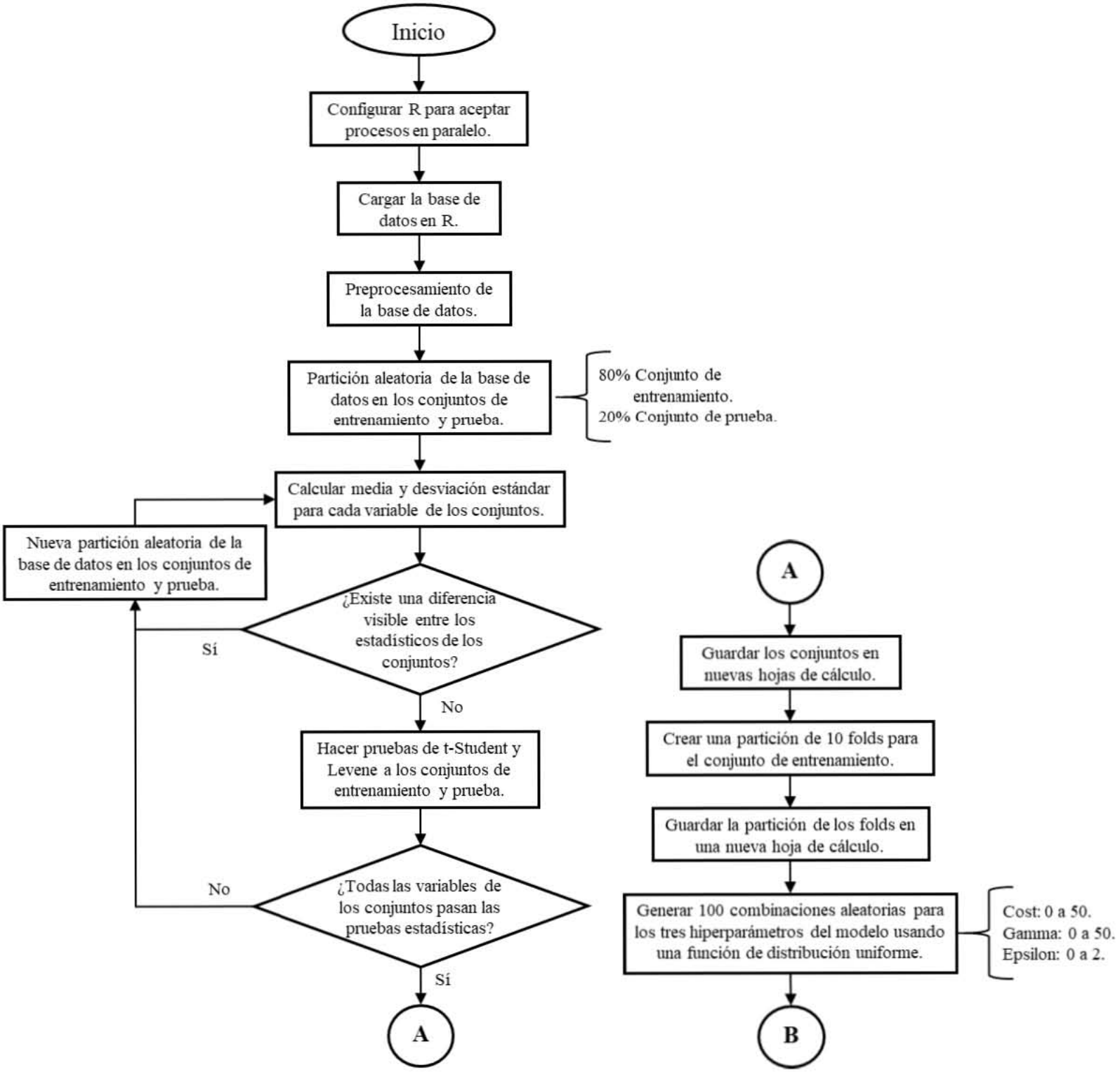
6.4.1 El CENACE ejecutará antes del inicio de cada Intervalo de Despacho las funciones para el Pronóstico de Demanda y el pronóstico de generación intermitente, para los próximos diez Intervalos de Despacho.

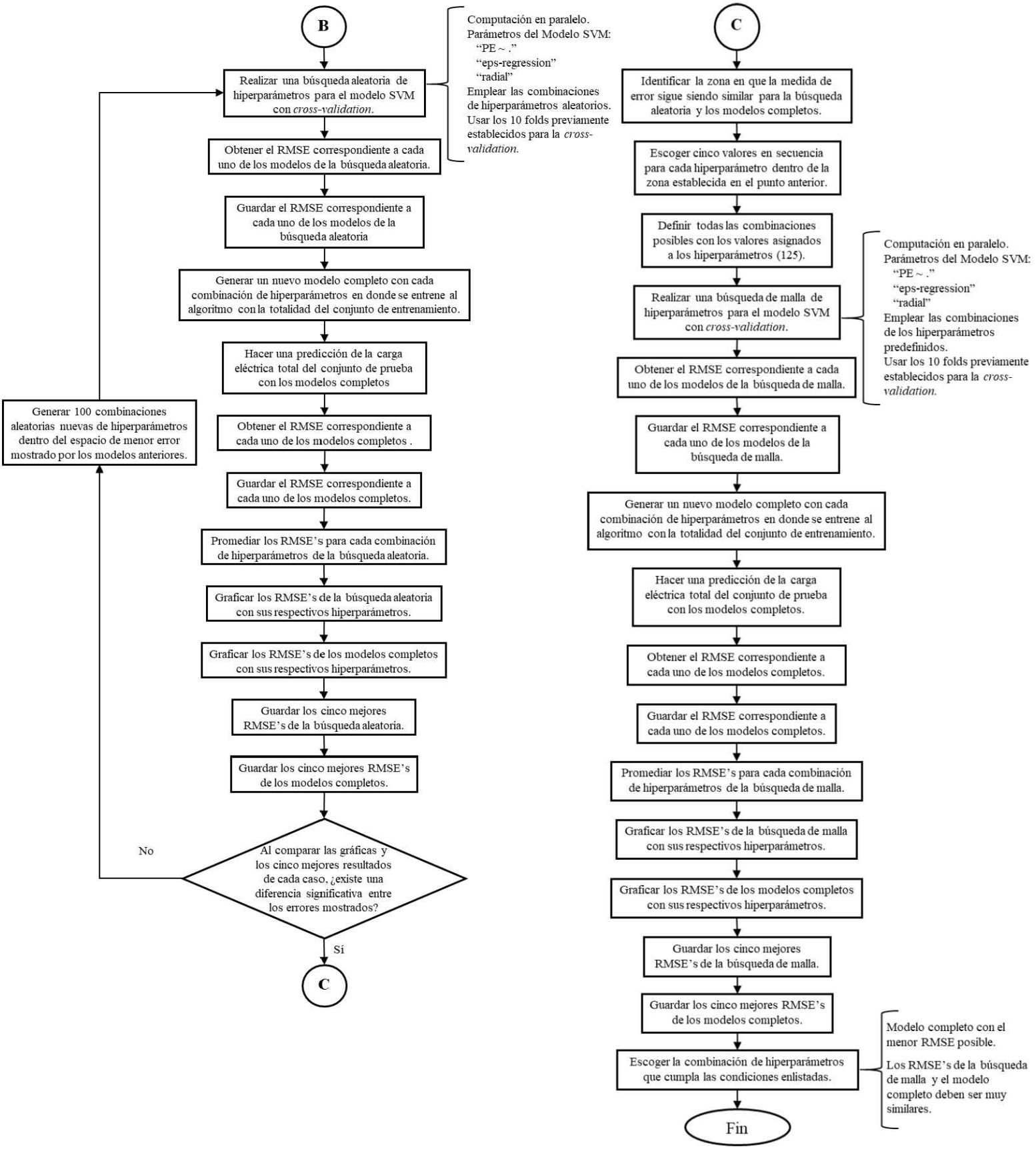
6.4.2 La ejecución de un nuevo pronóstico antes del inicio del siguiente Intervalo de Despacho actualizará los valores determinados en la ejecución anterior.

6.5 Identificación de Restricciones de Seguridad en el Despacho

6.5.1 El CENACE, para cada Intervalo de Despacho, deberá identificar las restricciones de transmisión que garanticen la Seguridad de Despacho en condiciones de funcionamiento normal o ante la ocurrencia de contingencias sencillas o contingencias múltiples.

Anexo 3. Diagrama de la Metodología Completo.




B
Realizar una búsqueda aleatoria de hiperparámetros para el modelo SVM con *cross-validation*.
Computación en paralelo.
Parámetros del Modelo SVM:
"PE ~ ."
"eps-regression"
"radial"
Emplear las combinaciones de hiperparámetros aleatorios.
Usar los 10 folds previamente establecidos para la *cross-validation*.
Obtener el RMSE correspondiente a cada uno de los modelos de la búsqueda aleatoria.
Guardar el RMSE correspondiente a cada uno de los modelos de la búsqueda aleatoria
Generar un nuevo modelo completo con cada combinación de hiperparámetros en donde se entrene al algoritmo con la totalidad del conjunto de entrenamiento.
Hacer una predicción de la carga eléctrica total del conjunto de prueba con los modelos completos
Obtener el RMSE correspondiente a cada uno de los modelos completos .
Guardar el RMSE correspondiente a cada uno de los modelos completos.
Promediar los RMSE's para cada combinación de hiperparámetros de la búsqueda aleatoria.
Graficar los RMSE's de la búsqueda aleatoria con sus respectivos hiperparámetros.
Graficar los RMSE's de los modelos completos con sus respectivos hiperparámetros.
Guardar los cinco mejores RMSE's de la búsqueda aleatoria.
Guardar los cinco mejores RMSE's de los modelos completos.
Al comparar las gráficas y los cinco mejores resultados de cada caso, ¿existe una diferencia significativa entre los errores mostrados?
No
Generar 100 combinaciones aleatorias nuevas de hiperparámetros dentro del espacio de menor error mostrado por los modelos anteriores.
Sí
C
C
Identificar la zona en que la medida de error sigue siendo similar para la búsqueda aleatoria y los modelos completos.
Escoger cinco valores en secuencia para cada hiperparámetro dentro de la zona establecida en el punto anterior.
Definir todas las combinaciones posibles con los valores asignados a los hiperparámetros (125).
Realizar una búsqueda de malla de hiperparámetros para el modelo SVM con *cross-validation*.
Computación en paralelo.
Parámetros del Modelo SVM:
"PE ~ ."
"eps-regression"
"radial"
Emplear las combinaciones de los hiperparámetros predefinidos.
Usar los 10 folds previamente establecidos para la *cross-validation*.
Obtener el RMSE correspondiente a cada uno de los modelos de la búsqueda de malla.
Guardar el RMSE correspondiente a cada uno de los modelos de la búsqueda de malla.
Generar un nuevo modelo completo con cada combinación de hiperparámetros en donde se entrene al algoritmo con la totalidad del conjunto de entrenamiento.
Hacer una predicción de la carga eléctrica total del conjunto de prueba con los modelos completos.
Obtener el RMSE correspondiente a cada uno de los modelos completos.
Guardar el RMSE correspondiente a cada uno de los modelos completos.
Promediar los RMSE's para cada combinación de hiperparámetros de la búsqueda de malla.
Graficar los RMSE's de la búsqueda de malla con sus respectivos hiperparámetros.
Graficar los RMSE's de los modelos completos con sus respectivos hiperparámetros.
Guardar los cinco mejores RMSE's de la búsqueda de malla.
Guardar los cinco mejores RMSE's de los modelos completos.
Escoger la combinación de hiperparámetros que cumpla las condiciones enlistadas.
Modelo completo con el menor RMSE posible.
Los RMSE's de la búsqueda de malla y el modelo completo deben ser muy similares.
Fin

Anexo 4. Código en R de la metodología

```
#Definicion de librerias.

library(e1071)
library(readxl)
library(ggplot2)
library(plotly)
library(foreach)
library(lattice)
library(caret)
library(parallel)
library(iterators)
library(doParallel)
library(lawstat)

#Carga de la base de datos.

lapply(c('foreach','doParallel'), require,
character.only=T)
registerDoParallel(cores=6)


base<-read_xlsx("Folds5x2_pp.xlsx", col_name
= TRUE, sheet = 1)
base

#Estadisticos de la base de datos.

est<-summary(base)

for(i in 1:nrow(est)){
  for(j in 1:ncol(est)){
    est[i,j]<-unlist(strsplit(est[i,j],
split=':', fixed=TRUE))[2]
  }
}
est<-as.data.frame.matrix(est)
row.names(est)<-c("Minimo","1er
Cuartil","Mediana","Media","3er Cuartil",
"Maximo")

vr<-
c(var(base$AT),var(base$V),var(base$AP),var(b
ase$RH),var(base$PE))
vr<-as.character(round(vr,digits = 2))
vr<-as.data.frame(t(vr))
row.names(vr)<-c("Varianza")
names(vr)<-names(est)

est<-rbind(est,vr)

boxplot(base$PE, main="Diagrama de caja: PE",
xlab="PE", ylab="Carga eléctrica total [MW]",
col="blue")

write.csv(est, file = "Estadisticos
base.csv")

#Conjuntos de entrenamiento y prueba.

index <- 1:nrow(base)
testindex <- sample(index,
trunc(length(index)*0.2))
testset <- na.omit(base[testindex,])
trainset<- na.omit(base[-testindex,])

#Comparacion estadistica de conjuntos.

cd<-matrix(nrow=ncol(base),ncol=4)

for(i in 1:ncol(base)){
  cd[i,1]<-mean(t(testset[,i]))
  cd[i,3]<-mean(t(trainset[,i]))
  cd[i,2]<-sd(t(testset[,i]))
  cd[i,4]<-sd(t(trainset[,i]))
}

write.csv(cd, file = "Comparacion
conjuntos.csv")

write.csv(testset, file = "Conjunto
prueba.csv")

write.csv(trainset, file = "Conjunto
entrenamiento.csv")

trainset<-read.csv("Conjunto
entrenamiento.csv", header=T)
trainset<-trainset[,-1]

testset<-read.csv("Conjunto prueba.csv",
header=T)
testset<-testset[,-1]

#Comparación de medias y varianzas

grupo<-as.factor(c(rep(1,nrow(trainset)),
rep(2,nrow(testset))))
an<-rbind(trainset,testset)
an[,6]<-grupo
colnames(an)<-c(names(trainset),"Grupo")
```

```
aAT<-t.test(AT~Grupo, data= an)
aAT

aV<-t.test(V~Grupo, data= an)
aV

aAP<-t.test(AP~Grupo, data= an)
aAP

aRH<-t.test(RH~Grupo, data= an)
aRH

aPE<-t.test(PE~Grupo, data= an)
aPE

vAT<-levene.test(an$AT,an$Grupo)
vAT

vV<-levene.test(an$V,an$Grupo)
vV

vAP<-levene.test(an$AP,an$Grupo)
vAP

vRH<-levene.test(an$RH,an$Grupo)
vRH

vPE<-levene.test(an$PE,an$Grupo)
vPE

#Separacion en folds.

set.seed(2019)
k<-10
folds<-createFolds(1:nrow(trainset), k=k,
list=F)

write.csv(folds, "Cross V.csv")

folds<-as.vector(read.csv("Cross V.csv",
header = T)[,2])

#Busqueda aleatoria 1.

q<-100
cost1<-runif(q,min=0,max=50)
epsilon1<-runif(q,min=0,max=2)
gamma1<- runif(q,min=0,max=50)
rmse1<-matrix(rep(0,q*k), nrow=q, ncol=k)
rmsef<-matrix(rep(0,q), nrow=q, ncol=1)
amse<-c()
armse<-c()

result<-foreach(i = 1:q, .combine = rbind)
%do% {
  c <- cost1[i]
  g <- gamma1[i]
  ep<- epsilon1[i]
  out<-foreach(j = 1:max(folds), .combine =
rbind, .inorder = T) %dopar% {
    deve <- trainset[folds != j, ]
    test <- trainset[folds == j, ]
    mdl<-e1071::svm(PE ~ ., data = deve, type
= "eps-regression", kernel = "radial", cost =
c, gamma= g, epsilon = ep)
    pred <- predict(mdl, test[,])
    amse[j]<-sum((pred - test[,5])^2) /
length(pred)
    armse<-sqrt(amse[j])
  }
  rmse1[i,]<-as.vector(out)
  model<-e1071::svm(PE ~ ., data = trainset,
type = "eps-regression", kernel = "radial",
cost = c, gamma= g, epsilon = ep)
  predi<-predict(model, testset[,])
  rmsef[i]<-sqrt(sum((predi - testset[,5])^2)
/ length(predi))
}
rmse1
rmsef

rmsep<-c()

for(i in 1:q){
  rmsep[i]<-mean(rmse1[i,])
}
rmsep

g1<-plot_ly(x=cost1,y=gamma1,z=epsilon1,
type= 'scatter3d', mode= 'markers', marker=
list(color=rmsep, colorscale= "Portland",
showscale=TRUE, colorbar=list(title='Error
medio'))) %>%
  layout(
    title = "Búsqueda aleatoria de
hiperparámetros 1.",
    scene = list(
      xaxis = list(title = "Costo"),
```

```
      yaxis = list(title = "Gamma"),
      zaxis = list(title = "Epsilon")
    ))

gp1<-plot_ly(x=cost1,y=gamma1,z=epsilon1,
type= 'scatter3d', mode= 'markers', marker=
list(color=as.vector(rmsef), colorscale=
"Portland", showscale=TRUE,
colorbar=list(title='Error medio'))) %>%
  layout(
    title = "Búsqueda aleatoria de
hiperparámetros 1 con conjunto de prueba.",
    scene = list(
      xaxis = list(title = "Costo"),
      yaxis = list(title = "Gamma"),
      zaxis = list(title = "Epsilon")
    ))


m1<-matrix(c(1:q,rmsep),nrow=q,ncol=2)
p1<-sort(m1[,2])[5]
h1<-m1[m1[,2]<=p1,][,1]

hp1<-data.frame(n=h1, c=cost1[h1],
g=gamma1[h1], ep=epsilon1[h1],
error=m1[h1,2])

mp1<-matrix(c(1:q,rmsef), nrow=q,ncol=2)
pp1<-sort(mp1[,2])[5]
ho1<-mp1[mp1[,2]<=pp1,][,1]
hpp1<-data.frame(n=ho1, c=cost1[ho1],
g=gamma1[ho1], ep=epsilon1[ho1],
error=mp1[ho1,2])

write.csv(hpp1, file = "Errores minimos 1
prueba.csv")
write.csv(hp1, file = "Errores minimos
1.csv")

hp1$error-hpp1$error

#Busqueda aleatoria 2.

q<-100
cost2<-runif(q,min=0,max=10)
epsilon2<-runif(q,min=0,max=0.25)
gamma2<- runif(q,min=0,max=10)

rmse2<-matrix(rep(0,q*k), nrow=q, ncol=k)
rmsef2<-matrix(rep(0,q), nrow=q, ncol=1)
amse<-c()
armse<-c()

result<-foreach(i = 1:q, .combine = rbind)
%do% {
  c <- cost2[i]
  g <- gamma2[i]
  ep<- epsilon2[i]
  out<-foreach(j = 1:max(folds), .combine =
rbind, .inorder = T) %dopar% {
    deve <- trainset[folds != j, ]
    test <- trainset[folds == j, ]
    mdl<-e1071::svm(PE ~ ., data = deve, type
= "eps-regression", kernel = "radial", cost =
c, gamma= g, epsilon = ep)
    pred <- predict(mdl, test[,])
    amse[j]<-sum((pred - test[,5])^2) /
length(pred)
    armse<-sqrt(amse[j])
  }
  rmse2[i,]<-as.vector(out)
  model<-e1071::svm(PE ~ ., data = trainset,
type = "eps-regression", kernel = "radial",
cost = c, gamma= g, epsilon = ep)
  predi<-predict(model, testset[,])
  rmsef2[i]<-sqrt(sum((predi -
testset[,5])^2) / length(predi))
}
rmse2
rmsef2

rmsep2<-c()

for(i in 1:q){
  rmsep2[i]<-mean(rmse2[i,])
}
rmsep2

g2<-plot_ly(x=cost2,y=gamma2,z=epsilon2,
type= 'scatter3d', mode= 'markers', marker=
list(color=rmsep2, colorscale= "Portland",
showscale=TRUE, colorbar=list(title='Error
medio'))) %>%
  layout(
    title = "Búsqueda aleatoria de
hiperparámetros 2.",
    scene = list(
      xaxis = list(title = "Costo"),
      yaxis = list(title = "Gamma"),
      zaxis = list(title = "Epsilon")
```

```
    ))

gp2<-plot_ly(x=cost2,y=gamma2,z=epsilon2,
type= 'scatter3d', mode= 'markers', marker=
list(color=as.vector(rmsef2), colorscale=
"Portland", showscale=TRUE,
colorbar=list(title='Error medio'))) %>%
  layout(
    title = "Búsqueda aleatoria de
hiperparámetros 2 con conjunto de prueba.",
    scene = list(
      xaxis = list(title = "Costo"),
      yaxis = list(title = "Gamma"),
      zaxis = list(title = "Epsilon")
    ))


m2<-matrix(c(1:q,rmsep2),nrow=q,ncol=2)
p2<-sort(m2[,2])[5]
h2<-m2[m2[,2]<=p2,][,1]

hp2<-data.frame(n=h2, c=cost2[h2],
g=gamma2[h2], ep=epsilon2[h2],
error=m2[h2,2])

mp2<-matrix(c(1:q,rmsef2), nrow=q,ncol=2)
pp2<-sort(mp2[,2])[5]
ho2<-mp2[mp2[,2]<=pp2,][,1]
hpp2<-data.frame(n=ho2, c=cost2[ho2],
g=gamma2[ho2], ep=epsilon2[ho2],
error=mp2[ho2,2])

write.csv(hpp2, file = "Errores minimos 2
prueba.csv")

write.csv(hp2, file = "Errores minimos
2.csv")

#Busqueda aleatoria 3.

q<-100
cost3<-runif(q,min=0,max=10)
epsilon3<-runif(q,min=0,max=0.15)
gamma3<- runif(q,min=1,max=2)

rmse3<-matrix(rep(0,q*k), nrow=q, ncol=k)
rmsef3<-matrix(rep(0,q), nrow=q, ncol=1)
amse<-c()
armse<-c()

result<-foreach(i = 1:q, .combine = rbind)
%do% {
  c <- cost3[i]
  g <- gamma3[i]
  ep<- epsilon3[i]
  out<-foreach(j = 1:max(folds), .combine =
rbind, .inorder = T) %dopar% {
    deve <- trainset[folds != j, ]
    test <- trainset[folds == j, ]
    mdl<-e1071::svm(PE ~ ., data = deve, type
= "eps-regression", kernel = "radial", cost =
c, gamma= g, epsilon = ep)
    pred <- predict(mdl, test[,])
    amse[j]<-sum((pred - test[,5])^2) /
length(pred)
    armse<-sqrt(amse[j])
  }
  rmse3[i,]<-as.vector(out)
  model<-e1071::svm(PE ~ ., data = trainset,
type = "eps-regression", kernel = "radial",
cost = c, gamma= g, epsilon = ep)
  predi<-predict(model, testset[,])
  rmsef3[i]<-sqrt(sum((predi -
testset[,5])^2) / length(predi))
}
rmse3
rmsef3

rmsep3<-c()

for(i in 1:q){
  rmsep3[i]<-mean(rmse3[i,])
}
rmsep3

g3<-plot_ly(x=cost3,y=gamma3,z=epsilon3,
type= 'scatter3d', mode= 'markers', marker=
list(color=rmsep3, colorscale= "Portland",
showscale=TRUE, colorbar=list(title='Error
medio'))) %>%
  layout(
    title = "Búsqueda aleatoria de
hiperparámetros 3.",
    scene = list(
      xaxis = list(title = "Costo"),
      yaxis = list(title = "Gamma"),
      zaxis = list(title = "Epsilon")
    ))

gp3<-plot_ly(x=cost3,y=gamma3,z=epsilon3,
```

```
type= 'scatter3d', mode= 'markers', marker=
list(color=as.vector(rmsef3), colorscale=
"Portland", showscale=TRUE,
colorbar=list(title='Error medio'))) %>%
  layout(
    title = "Búsqueda aleatoria de
hiperparámetros 2 con conjunto de prueba.",
    scene = list(
      xaxis = list(title = "Costo"),
      yaxis = list(title = "Gamma"),
      zaxis = list(title = "Epsilon")
    ))


m3<-matrix(c(1:q,rmsep3),nrow=q,ncol=2)
p3<-sort(m3[,2])[5]
h3<-m3[m3[,2]<=p3,][,1]

hp3<-data.frame(n=h3, c=cost3[h3],
g=gamma3[h3], ep=epsilon3[h3],
error=m3[h3,2])

mp3<-matrix(c(1:q,rmsef3), nrow=q,ncol=2)
pp3<-sort(mp3[,2])[5]
ho3<-mp3[mp3[,2]<=pp3,][,1]
hpp3<-data.frame(n=ho3, c=cost3[ho3],
g=gamma3[ho3], ep=epsilon3[ho3],
error=mp3[ho3,2])

write.csv(hpp3, file = "Errores minimos 3
prueba.csv")

write.csv(hp3, file = "Errores minimos
3.csv")

#Busqueda de malla.

cost<- seq(3,5, by=0.5)
epsilon<-seq(0.06, 0.08, by = 0.005)
gamma<- seq(1.81,1.85, by= 0.01)
parms<-
expand.grid(cost=cost,gamma=gamma,epsilon=eps
ilon)

rmse<-matrix(rep(0,nrow(parms)*k),
nrow=nrow(parms), ncol=k)
rmsef4<-matrix(rep(0,nrow(parms)),
nrow=nrow(parms), ncol=1)
amse<-c()
armse<-c()

result<-foreach(i = 1:nrow(parms), .combine =
rbind) %do% {
  c <- parms[i, ]$cost
  g <- parms[i, ]$gamma
  ep<- parms[i, ]$epsilon
  out<-foreach(j = 1:max(folds), .combine =
rbind, .inorder = T) %dopar% {
    deve <- trainset[folds != j, ]
    test <- trainset[folds == j, ]
    mdl<-e1071::svm(PE ~ ., data = deve, type
= "eps-regression", kernel = "radial", cost =
c, gamma= g, epsilon = ep)
    pred <- predict(mdl, test[,])
    amse[j]<-sum((pred - test[,5])^2) /
length(pred)
    armse<-sqrt(amse[j])
  }
  rmse[i,]<-as.vector(out)
  model<-e1071::svm(PE ~ ., data = trainset,
type = "eps-regression", kernel = "radial",
cost = c, gamma= g, epsilon = ep)
  predi<-predict(model, testset[,])
  rmsef4[i]<-sqrt(sum((predi -
testset[,5])^2) / length(predi))
}
rmse
rmsef4

rmsep4<-c()

for(i in 1:nrow(parms)){
  rmsep4[i]<-mean(rmse[i,])
}
rmsep4

g4<-
plot_ly(x=parms$cost,y=parms$gamma,z=parms$ep
silon, type= 'scatter3d', mode= 'markers',
marker= list(color=rmsep4, colorscale=
"Portland", showscale=TRUE,
colorbar=list(title='Error medio'))) %>%
  layout(
    title = "Búsqueda de malla de
hiperparámetros.",
    scene = list(
      xaxis = list(title = "Costo"),
      yaxis = list(title = "Gamma"),
      zaxis = list(title = "Epsilon")
    ))
```

```
gp4<-
plot_ly(x=parms$cost,y=parms$gamma,z=parms$epsilon, type= 'scatter3d', mode= 'markers', marker= list(color=as.vector(rmsef4), colorscale= "Portland", showscale=TRUE, colorbar=list(title='Error medio'))) %>%
  layout(
    title = "Búsqueda de malla de hiperparámetros con conjunto de prueba.",
    scene = list(
      xaxis = list(title = "Costo"),
      yaxis = list(title = "Gamma"),
      zaxis = list(title = "Epsilon")
    ))


m4<-
matrix(c(1:nrow(parms),rmsep4),nrow=nrow(parms),ncol=2)
p4<-sort(m4[,2])[5]
h4<-m4[m4[,2]<=p4,][,1]

hp4<-data.frame(n=h4, c=parms$cost[h4], g=parms$gamma[h4], ep=parms$epsilon[h4], error=m4[h4,2])

mp4<-matrix(c(1:nrow(parms),rmsef4), nrow=nrow(parms),ncol=2)
pp4<-sort(mp4[,2])[5]
ho4<-mp4[mp4[,2]<=pp4,][,1]
hpp4<-data.frame(n=ho4, c=parms$cost[ho4], g=parms$gamma[ho4], ep=parms$epsilon[ho4], error=mp4[ho4,2])

write.csv(hpp4, file = "Errores minimos 4 prueba.csv")

write.csv(hp4, file = "Errores minimos 4.csv")

Mdl<-svm(PE~., data= trainset, type= "eps-regression", kernel= "radial", cost= parms$cost[25], gamma= parms$gamma[25], epsilon= parms$epsilon[25])
mdl.pred<-predict(Mdl,testset[,])
mape<-sum((abs(mdl.pred - testset[,5])/testset[,5])*100)/length(testset$PE)
mape
```

```
vmse<-sum((mdl.pred - testset[,5])^2) / length(testset$PE)
vrmse<-sqrt(vmse)
vrmse
```