Universidad de Cienfuegos "Carlos Rafael Rodríguez"

Facultad de Ingeniería

Carrera de Ingeniería Química

TRABAJO DE DIPLOMA

"Metodología para la aplicación de algoritmos de aprendizaje automático en el control de la calidad de la empresa GydeMa de la provincia de Cienfuegos".

Autor: Ángel David Vázquez Romero

Tutor: MSc. Alejandro Valdés López

MSc. Elisa María Chou Rodríguez

Cienfuegos, 2020

Pensamiento

"No entiendes algo hasta que lo aprendes a hacer de más de una forma".
"Una cosa o idea parece significativa solo cuando tenemos varias formas diferentes de representarla: diferentes perspectivas y diferentes asociaciones".

Marvin Minsky

Dedicatoria

 A mi Madre porque sin su amor y apoyo no hubiera sido posible cumplir este sueño.

 A mi Padre por estar siempre a mi lado, por ser mi guía y el mejor amigo que se pudiera desear.

 A mi Abuelita por darme todo su amor.

Agradecimientos

✓ *...A mi profesor y tutor por sus estupendas recomendaciones, por su exigencia, dedicación, crítica y aliento ...*

✓ *...A mi profesora y tutora por su asistencia, por su supervisión y dirección.*

✓ *...A mi padre por su ayuda y dedicación ...*

✓ *...A los amigos que me dieron su apoyo incondicional...*

Muchas gracias

Resumen

El control estadístico de la calidad juega un papel imprescindible, pero las técnicas tradicionales de estadística y las herramientas de gestión clásicas no son suficientes, se requiere de nuevas herramientas de análisis como lo es el llamado Aprendizaje Automático (Machine Learning), el cual está revolucionando los sistemas de control de calidad de las empresas industriales. Con esta investigación se llega a una metodología que facilita el uso de los algoritmos de aprendizaje automático, para el control de la calidad en el proceso productivo de la empresa de la industria alimenticia GydeMa. Se hace una revisión de los diversos algoritmos de regresión y clasificación de aprendizaje automático; y su aplicabilidad al control de la calidad en los procesos. Se utiliza como herramienta computacional, la biblioteca de Python Scikit-learn para el aprendizaje automático y el análisis de datos, con lo cual se arriba a los resultados, donde se hace evidente que la integración de los modelos de aprendizaje automático, con otras herramientas de control de la calidad, resulta apropiado para la automatización y optimización de estos procesos favoreciendo la búsqueda de posibles mejoras y una gestión más eficiente, constituyendo la metodología propuesta un punto de apoyo para la implementación de los algoritmos citados en el contexto de la empresa GydeMa de la provincia de Cienfuegos.

Palabras claves: gestión de la calidad, aprendizaje automático (Machine Learning), regresión, máquinas de soporte vectorial, árboles de decisión, bosques aleatorios, redes bayesianas, redes neuronales, aprendizaje supervisado.

Abstract

Statistical quality control plays an essential role, but traditional statistical techniques and classic management tools are not enough. New analysis tools are required, such as Machine Learning, which is revolutionizing the quality control systems of industrial companies. This research leads to a methodology that facilitates the use of machine learning algorithms for quality control in the production process of the GydeMa food industry company. A review of the various machine learning regression and classification algorithms is made; and its applicability to quality control in processes. The Python Scikit-learn library for machine learning and data analysis is used as a computational tool, which leads to the results, where it becomes evident that the integration of machine learning models with other Quality control is appropriate for the automation and optimization of these processes, favoring the search for possible improvements and more efficient management, the proposed methodology constituting a point of support for the implementation of the aforementioned algorithms in the context of the company GydeMa de the province of Cienfuegos.

Key words: quality management, machine learning, regression, vector support machines, decision trees, random forests, Bayesian networks, neural networks, supervised learning.

Índice

Índice de Tablas

Índice de Figuras

INTRODUCCIÓN

Introducción

La calidad es el conjunto de características que tienen un producto o servicios que permiten satisfacer a los clientes. En términos prácticos ello implica cumplir con las especificaciones establecidas en el diseño para satisfacer las necesidades de los clientes.

Muchos expertos aseguran que históricamente, de los niveles de calidad de los productos y servicios depende en gran medida el desarrollo económico de las comunidades, ya que les permite obtener beneficios de manera sostenible.

Según afirma Mukherjee (2019) la calidad no es una meta, es una marcha hacia adelante incluso cero defectos no es lo último para ser alcanzado. Por lo cual, el control estadístico de la calidad juega un papel imprescindible ya que va más allá de la inspección para concentrarse en la identificación y eliminación de los problemas que causan defectos. Herramientas como las gráficas de control se volvieron un medio popular para identificar los problemas de calidad en los procesos de producción y asegurar la coherencia de la producción (Gutiérrez y de la Vara, 2013).

En la actualidad con el avance de las tecnologías existe una disponibilidad de gran cantidad de información en relación a diferentes procesos y sistemas (tantos industriales como logísticos), servicios (ventas, conexiones entre usuarios, consumo eléctrico, etc.) o tráfico de datos (registros de eventos en enrutadores, servidores y otros equipos). El análisis de estos datos puede proporcionar información muy valiosa acerca del comportamiento de los procesos ya que estos esconden información de gran valor para saber, no solo lo que sucede a nuestro alrededor, sino también lo que va a pasar en un futuro, en algunos casos con niveles de precisión muy altos.

Para poder tratar estas grandes cantidades de datos y darle valor añadido extrayendo el conocimiento que existen en ellos, las técnicas tradicionales de estadística y las herramientas de gestión clásicas no son suficientes, debido a que no están preparadas para trabajar con tantos datos ni tan variados (Aguilar et al., 2018). Por lo que se hace evidente la necesidad de nuevas herramientas de análisis como lo es el llamado Aprendizaje Automático (Machine Learning) el cual está revolucionando los sistemas de control de calidad de las empresas industriales, ofreciendo nuevas capacidades para optimizar sus procesos a través de los datos.

El aprendizaje automático es un método de análisis de datos que automatiza la construcción de modelos analíticos. Es una rama de la inteligencia artificial basada en la idea de que los sistemas pueden aprender de los datos, identificar patrones y tomar decisiones con una mínima intervención humana. Machine Learning contribuye a

optimizar la gestión de calidad en las operaciones y procesos, reduce costes, minimiza errores y ayuda a las empresas a ganar en agilidad.

Problema de investigación

No existe ninguna investigación que determine la mejor estrategia para la aplicación de los algoritmos de aprendizaje automático en el proceso de control de la calidad en la Empresa GydeMa, de la provincia de Cienfuegos.

Hipótesis

La conformación de una metodología facilitará el uso de los algoritmos de aprendizaje automático para el control de la calidad en el proceso productivo de la empresa de la industria alimenticia GydeMa.

Objetivo general

Elaborar una metodología que facilite el uso de los algoritmos de aprendizaje automático para su implementación en el control de la calidad de la empresa GydeMa, de la provincia de Cienfuegos.

Objetivos específicos

1. Establecer los fundamentos teórico-conceptuales necesarios para el entendimiento de los algoritmos de aprendizaje automático y su aplicabilidad al control de la calidad en los procesos.
2. Elaborar una metodología para la implementación de los algoritmos de aprendizaje automático en el control de la calidad de la empresa GydeMa.
3. Ejemplificar el uso de diversos algoritmos de regresión y clasificación para su aplicación en el caso de estudio.

Métodos de investigación

Método hipotético-deductivo: Fue empleado para a partir de la observación del fenómeno objeto de estudio construir la hipótesis de la investigación.

Método histórico-lógico: Se utiliza con el propósito de profundizar en la evolución de los algoritmos analizados y su aplicación en el control de la calidad.

Método analítico-sintético: Se emplea para el examinar y sintetizar la información en la literatura examinada con el propósito de extraer los elementos más relevantes para lograr una adecuada comprensión de los temas objeto de estudio.

Análisis documental: Se utiliza para la recopilación de información a partir de los libros, tesis y artículos consultados.

Estructura de la tesis

En el **capítulo 1** se presentan los aspectos más relevantes sobre el concepto de calidad, su evolución histórica y gestión. Es analizado el sistema de gestión de la calidad basado en la norma ISO 9001:2015, herramientas de control de la calidad y la metodología de mejora de procesos seis sigmas. Finalmente se abordan los algoritmos de aprendizaje automático, su relación con otras disciplinas, clasificación, principios de funcionamiento y vinculación al control de la calidad.

En el **capítulo 2** se describe la empresa GydeMa destacándose su flujo tecnológico y procedimiento de control de la calidad. Se presenta la metodología propuesta y se describen los fundamentos matemáticos de los algoritmos de aprendizaje automático, métricas para medir su desempeño y el procedimiento de optimización bayesiana para el ajuste de los hiper-parámetros.

En el **capítulo 3** se fundamenta la aplicabilidad de los algoritmos analizados, herramientas computacionales existentes para su uso. Luego mediante el uso de tablas y figuras se presentan los resultados de la aplicación de los algoritmos de regresión y clasificación.

CAPÍTULO I

Capítulo I: Marco teórico referencial

I.1 El significado de la calidad

La calidad es generalizada y reside en todas las formas y figuras tangibles y concretas, procesos y productos. La calidad ha sido un reflejo de la civilización humana, del progreso en la ciencia y tecnología, de una demanda cada vez mayor de la sociedad humana por mejores bienes y servicios, y un esfuerzo consecuente para satisfacer la demanda. La búsqueda, el deseo o la pasión por la calidad tuvieron una entrada bastante reciente en el ámbito de industria o, en general, el ámbito de las actividades humanas organizadas. La calidad está asociada con todo lo que nos rodea, juzgado en términos de características o parámetros de la entidad particular que consideramos (Mukherjee, 2019).

Según Vega et al. (2011) la calidad es la interacción entre el modo de pensar de la empresa y los procesos que en ella se gestionan en el día a día, y la búsqueda permanente de la perfección en todas y cada una de las personas que la integran con el objetivo de transformar la sociedad, atender las necesidades del entorno y satisfacer a las partes interesadas.

Krishnamoorthi et al. (2019) afirma que la definición de calidad más aceptada es aquella que plantea que la calidad tiene varias dimensiones:

1. Rendimiento: la capacidad del producto para hacer el trabajo que se supone que debe hacer.
2. Características: cosas que se suman a la comodidad y el confort.
3. Confiabilidad: la capacidad de desempeñarse sin fallas con el tiempo.
4. Conformidad: el grado en que el producto cumple con los códigos de un estado o una comunidad.
5. Durabilidad: el tiempo que durará el producto hasta que se descarte.
6. Capacidad de servicio: la capacidad de realizar reparaciones de manera fácil, rápida y razonable costo.
7. Estética: atractivo sensorial, como el color, el sonido, la sensación y la comodidad.
8. Calidad percibida: la impresión que el producto crea en la mente del cliente.

La definición anterior de calidad de Garvin solo refuerza la idea de que la calidad tiene muchos aspectos y no se pueden definir fácilmente en una frase u oración simple. Puede tener diferentes significados para diferentes productos, e incluso para el mismo producto, puede tener un significado diferente para diferentes usuarios.

Según Vega et al. (2011) la calidad "es una simple y no analizable propiedad, que aprendemos a reconocer solo a través de la experiencia"; es algo que no se toca, y difiere con el tiempo en relación con una misma cosa; esto se debe al mejoramiento continuo. También sustenta el enfoque de las organizaciones que aprenden, al afirmar que cuando una organización tiene distintas experiencias, siempre le queda un aprendizaje.

Una definición más trascendente de la calidad aparece en la norma ISO 9000: 2000. En ella, la calidad se define como el grado con el que un conjunto de características inherentes cumple los requisitos. Grado significa que se puede usar calidad con adjetivos como mala, buena y excelente. Inherente se define como que existe en algo, en especial como una característica permanente. Las características pueden ser cuantitativas o cualitativas. Un requisito es una necesidad o expectativa que se especifica; en general está implícita en la organización, sus clientes y otras partes interesantes, o bien es obligatoria (Besterfield, 2009).

Podemos definir la calidad de muchas maneras. La mayoría de las personas tiene una comprensión conceptual de la calidad en relación con una o más características deseables que un producto o servicio debe poseer. La calidad se ha convertido en uno de los factores de decisión del consumidor más importantes en la selección entre productos y servicios competidores. El fenómeno es generalizado, independientemente de si el consumidor es un individuo, una organización industrial, una tienda minorista, un banco o institución financiera o un programa de defensa militar. En consecuencia, comprender y mejorar la calidad son factores clave que conducen al éxito empresarial, el crecimiento y la mejora de la competitividad (Montgomery, 2013).

I.2 Evolución histórica de la Calidad

Esforzarse por la calidad, en el sentido de buscar la excelencia en las actividades, siempre ha sido parte del esfuerzo humano. La historia proporciona numerosos ejemplos de personas que alcanzaron los más altos niveles de excelencia, o calidad, en su individuo o colectivo. La música de Beethoven, las Grandes Pirámides de Egipto y Los templos del sur de la India son solo algunos ejemplos. La calidad atrae a los humanos y proporciona una sensación de satisfacción, razón por la cual la mayoría de las personas disfruta escuchar un buen concierto, ver una buena obra de teatro, observar una bella foto o incluso montar un auto bien construido (Krishnamoorthi et al., 2019).

La calidad es inherente a la especie humana; es decir, existe desde la propia existencia del hombre sobre la Tierra. Se originó con el hombre de las cavernas, quien buscaba en el alimento algo que saciara el hambre, en sus armas de defensa y en el abrigo para conseguir el calor, condiciones que le permitieran sobrevivir a las épocas y los ambientes que lo rodeaban. Con el paso del tiempo el ser humano fue mejorando la calidad de la respuesta a sus necesidades básicas, como alimento y seguridad, para lo cual fue apropiándose de prácticas encaminadas al mejoramiento continuo de sus armas, de sus métodos de cultivo, desarrollando su propia tecnología, primero para labrar la piedra (Edad de Piedra) y luego los metales (Edad del Metal), y otros desarrollos, hasta llegar a la Edad Media, en la cual los artesanos eran quienes se encargaban de manejar los bienes de producción y de consumo. Todo el proceso se realizaba en forma conjunta, es decir, el diseño, el desarrollo del producto y su control, de tal manera que el artesano se aseguraba de que el producto contara con la calidad requerida para el cliente (Vega et al., 2011).

Krishnamoorthi et al. (2019) plantea que durante los últimos 70 años, el término "calidad" se ha utilizado en el mercado para indicar qué tan libre de defectos tiene un producto comprado y qué tan bien cumple con las necesidades de su usuario. Después de la Revolución Industrial a principios de 1900, con la producción en masa se adoptaron técnicas para fabricar grandes cantidades de productos para satisfacer la creciente demanda de bienes. Se necesitaban esfuerzos especiales para lograr la calidad en los productos a partir de piezas producidas en masa. La variabilidad o la falta de uniformidad en las piezas producidas en masa crearon problemas de calidad durante el montaje de piezas. Por lo tanto, se necesitaban nuevos métodos para minimizar esta variabilidad y garantizar la uniformidad en las piezas que se producían en masa. El Dr. Walter A. Shewhart, trabajando para los antiguos Laboratorios Bell a principios de 1920, fue pionero en el uso de estadísticas para monitorear y controlar la variabilidad en la fabricación de partes y productos, y gráficos de control inventados para este propósito.

Durante la Segunda Guerra Mundial, el ejército estadounidense empezó a utilizar procedimientos de muestreo estadístico y a imponer estrictas normas a sus proveedores. El War Production Board ofreció cursos gratuitos de capacitación en los métodos estadísticos desarrollados dentro de Bell System. El impacto sobre la producción en tiempos de guerra fue mínimo, pero el esfuerzo dio lugar a especialistas en la calidad, quienes empezaron a utilizar y extender estas herramientas en sus organizaciones. Así,

el control estadístico de la calidad se extendió y adoptó en forma gradual en las industrias de manufactura. Se inventaron tablas de muestreo con la etiqueta MIL-STD para las normas militares, que aún se usan. La primera publicación profesional de la disciplina, Industrial Quality Control, se publicó en 1944, y poco tiempo después se fundaron sociedades profesionales (entre las que destaca la American Society for Quality Control, ahora conocida como la American Society for Quality o ASQ,) para desarrollar, promover y aplicar los conceptos de la calidad (Evans y Lindsay, 2008).

En la posguerra, comprendida entre las décadas de 1950 y 1960, los productos de Estados Unidos se consideraban de gran calidad a pesar de los defectos que pudieran presentar, mientras que los del Japón eran pésimos, de muy baja calidad. Aparecen entonces Edward Deming y Joseph Juran, quienes se dedican a enseñar a los japoneses sus principios para mejorar la calidad, en especial técnicas para el mejoramiento continuo, entre las que se encuentran las técnicas estadísticas, búsqueda de la satisfacción del cliente y formación para todas las personas. Se impone el control estadístico del proceso y durante la década de 1950 se dio un auge de hacer las cosas muy bien, en especial en los países más involucrados en la guerra, los cuales hoy representan las potencias del mundo (Vega et al., 2011).

La historia de la calidad se ha desarrollado extremadamente rápido en los últimos tiempos en consonancia con el desarrollo asombrosamente rápido de la ciencia y tecnología, la sociedad, la economía e incluso la política, así como herramientas cuantitativas (esencialmente estadísticas), a menudo bastante sofisticadas están siendo ampliamente utilizadas para la evaluación de la calidad. Ahora hay ingenieros, estadísticos, gerentes expertos y expertos de marca que son reconocidos como Profesionales de la Calidad (Mukherjee, 2019).

I.3 Gestión de la Calidad

La gestión de la calidad es el conjunto de actividades de la función general de dirección, que determina la política de calidad, sus objetivos, procesos e indicadores, con las responsabilidades correspondientes. Tiene fundamento en la satisfacción de las necesidades de los clientes externos, sin desconocer que los clientes internos son muy importantes para el cumplimiento de los objetivos de la calidad (Vega et al., 2011).

Según NC-ISO 9001 (2015)los principios de la gestión de la calidad son:

- enfoque al cliente;
- liderazgo;

- compromiso de las personas;
- enfoque a procesos;
- mejora;
- toma de decisiones basada en la evidencia;
- gestión de las relaciones.

Vega et al. (2011) afirma que de los principios antes mencionados se desprenden los siguientes beneficios: fidelidad de los clientes, velocidad de articulación y alineación de procesos, nuevas oportunidades de mercado, capacidad para desarrollar referenciaciones competitivas y comparativas y facilidad para establecer relaciones de beneficio recíproco con los proveedores.

I.3.1 Sistema de Gestión de Calidad basados en ISO 9001:2015

La gestión de la calidad de una empresa necesita de un sistema de apoyo para poder llevarse a cabo con efectividad, el cual debe estar integrado en los procesos, procedimientos, instrucciones de trabajo, mediciones y controles de las operaciones propias de la organización (Izaguirre, 2017).

La adopción de un sistema de gestión de la calidad es una decisión estratégica para una organización que le puede ayudar a mejorar su desempeño global y proporcionar una base sólida para las iniciativas de desarrollo sostenible (NC-ISO 9001, 2015).

La norma ISO 9001: 2015 responde a las últimas tendencias y es compatible con otros sistemas de gestión, tales como ISO 14001. Esta presenta una nueva estructura de nivel superior, para que sea más fácil de utilizar en conjunto con otras normas de sistemas de gestión, con una mayor importancia que se da al riesgo (Cambra, 2017).

Esta norma propone un sistema de gestión de calidad bien definido, basado en un marco de referencia que integra conceptos, principios, procesos y recursos fundamentales establecidos relativos a la calidad para ayudar a las organizaciones a hacer realidad sus objetivos. Su objetivo es incrementar la consciencia de la organización sobre sus tareas y compromiso para satisfacer las necesidades y las expectativas de sus clientes y partes interesadas, así como lograr la satisfacción con sus productos y servicios (NC-ISO 9001, 2015).

La norma NC ISO 9001:2015 explica que el pensamiento basado en riesgos permite a una organización determinar los factores que pueden causar que sus procesos y su sistema de gestión de la calidad se desvíen de los resultados planificados, para poner en

marcha controles preventivos que minimicen los efectos negativos y maximicen el uso de las oportunidades a medida que surjan (Izaguirre, 2017).

Los beneficios potenciales para una organización de implementar un sistema de gestión de la calidad basado en la norma NC ISO 9001:2015 son:

- Capacidad para proporcionar regularmente productos y servicios que satisfagan los requisitos del cliente y los legales y reglamentarios aplicables.
- Facilitar oportunidades de aumentar la satisfacción del cliente.
- Abordar los riesgos y oportunidades asociadas con su contexto y objetivos.
- La capacidad de demostrar la conformidad con requisitos del sistema de gestión de la calidad especificados.

I.4 Control Estadístico de la Calidad

El control estadístico juega un papel primordial en el desempeño de los procesos, ya que en estos siempre existen fuentes de variación bajo condiciones normales o comunes de trabajo. Todas las M´s (mano de obra, maquinaria, materiales, método, medio ambiente y medición) aportan variación a las variables de salida del proceso, en forma natural inherente; además pueden aportar variaciones especiales o fuera de lo común, por lo que resulta fundamental distinguir de forma eficiente entre ambos tipos de variación, para así tomar las medidas adecuadas en cada caso, haciéndose necesario trabajar en el control de la calidad de los procesos, apoyándose en el uso correcto de las herramientas estadísticas, las cuales son imprescindibles para su identificación, con la finalidad de mejorar el desempeño de estos y de las organizaciones como un todo, lo que se traduce, en el éxito empresarial (Cambra, 2017).

I.4.1 Herramientas para el control estadístico de la calidad: gráficos o cartas de control

El objetivo básico de una carta de control es observar y analizar el comportamiento de un proceso a través del tiempo. Así, es posible distinguir entre variaciones por causas comunes y especiales (atribuibles), lo que ayudará a caracterizar el funcionamiento del proceso y decidir las mejores acciones de control y de mejora. Cuando se habla de analizar el proceso nos referimos principalmente a las variables de salida (características de calidad), pero las cartas de control también pueden aplicarse para analizar la variabilidad de variables de entrada o de control del proceso mismo (Gutiérrez y de la Vara, 2013).

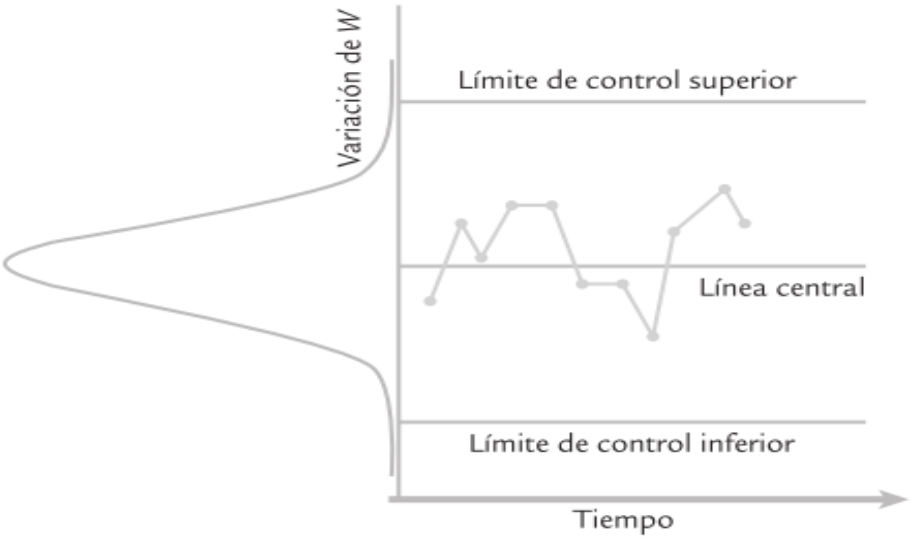


Figura 1.1: Idea y elementos de una carta de control Fuente:(Gutiérrez y de la Vara, 2013)

La línea central de una carta de control representa el promedio del estadístico que se está graficando, cuando el proceso se encuentra en control estadístico. Las otras dos líneas se llaman límites de control, superior e inferior, y están en una posición tal que, cuando el proceso está en control estadístico, hay una alta probabilidad de que prácticamente todos los valores del estadístico (puntos) caigan dentro de los límites. De esta manera, si todos los puntos están dentro de los límites, entonces se supone que el proceso está en control estadístico. Por el contrario, si al menos un punto está fuera de los límites de control, entonces esto es una señal de que pasó algo especial y es necesario investigar su causa. En general, los límites de control son estimaciones de la amplitud de la variación del estadístico (promedio, rangos, etc.) que se grafica en la carta.

Lo que se observa en una carta de control no sólo es que un punto caiga fuera de los límites de control, sino también cualquier formación o patrón de puntos que tenga muy poca probabilidad de ocurrir en condiciones "normales", lo cual será una señal de alerta de posibles cambios debidos a causas especiales (Gutiérrez, 2010).

Besterfield (2009) afirma que la gráfica de control sirve para tener un registro continuo de determinada característica de calidad. Es una fotografía del proceso a través del tiempo. Cuando se completa la gráfica, se sustituye por una nueva y la gráfica llena se guarda en un archivo en la oficina. La gráfica se usa para mejorar la calidad del proceso,

para determinar la capacidad del proceso, para ayudar a determinar especificaciones efectivas, para determinar cuándo dejar al proceso por sí solo, y cuándo hacer ajustes, y para investigar las causas de la calidad inaceptable o marginal.

Según Gutiérrez y de la Vara (2013) existen dos tipos generales de cartas de control: para variables y para atributos. Las cartas de control para variables se aplican a características de calidad de tipo continuo, que intuitivamente son aquellas que requieren un instrumento de medición (pesos, volúmenes, voltajes, longitudes, resistencias, temperaturas, humedad, etcétera). Las cartas de control para variables tipo Shewhart más usuales son:

- $\bar{x}$ (de medias)
- *R* (de rangos)
- *S* (de desviaciones estándar)
- *X* (de medidas individuales)

Estas formas distintas de llamarle a una carta de control se deben al correspondiente estadístico que se representa en la carta, y por medio de la cual se busca analizar una característica importante de un producto o un proceso.

Existen características de calidad de un producto que no son evaluadas con un instrumento de medición en una escala continua o al menos en una escala numérica. En estos casos, el producto se juzga como conforme o no conforme, dependiendo de si posee ciertos atributos; o también al producto se le podrá contar el número de defectos o no conformidades que tiene. Este tipo de características de calidad son monitoreadas a través de las cartas de control para atributos:

- *p* (proporción o fracción de artículos defectuosos)
- *np* (número de unidades defectuosas)
- *c* (número de defectos)
- *u* (número promedio de defectos por unidad)

I.4.2 Six Sigma como metodología de mejora de procesos

Six sigma es un método organizado y sistemático para resolver problemas estratégicos, mejora del sistema y desarrollo de nuevos productos y servicios que se basan en métodos estadísticos y el método científico para hacer reducciones dramáticas en tasas de defectos definidas por el cliente y / o mejoras en las variables de salida clave (Allen, 2019).

Según Gutiérrez y de la Vara (2013), en su nivel más elemental la meta de Six Sigma, que le da el nombre, es lograr procesos con una calidad Six Sigma, es decir, que como máximo generen 3.4 defectos por millón de oportunidades de error. Esta meta se pretende alcanzar mediante un programa vigoroso de mejora, diseñado e impulsado por la alta dirección de una organización, en el que se desarrollan proyectos a lo largo y ancho de la organización con el objetivo de lograr mejoras, así como eliminar defectos y retrasos de productos, procesos y transacciones.

Allen (2019) afirma que es cierto que Six Sigma comparte la característica de muchas modas en que sus métodos y principios asociados no se derivan de ningún claro y riguroso fundamento o axiomas matemáticos, aun así, propiedades de Six Sigma sugieren que podría ser relevante durante mucho tiempo, incluye: el método es relativamente específico y por lo tanto, fácil de implementar e incorpora el principio de presupuesto justificación para cada proyecto. Por lo tanto, los participantes aprecian su falta de ambigüedad, y la gerencia aprecia el énfasis en el resultado final.

El proceso "Seis Sigma" es un método sistemático que utiliza datos, rigurosamente medidos y analizados, para identificar las fuentes de error (causas raíces de un problema) y las formas de eliminarlas, generando mayor satisfacción del cliente y ahorros económicos sustanciales. Tiene seis niveles, donde 1Sigma es cometer muchos errores y 6Sigma muy pocos. Si se quiere saber en qué nivel Sigma se encuentra una organización la lista siguiente puede ayudar:

- 1Sigma= 690.000 errores por millón de unidades.
- 2Sigma= 308.538 errores por millón de unidades.
- 3Sigma= 66.807 errores por millón de unidades.
- 4Sigma= 6.210 errores por millón de unidades.
- 5Sigma= 233 errores por millón de unidades.
- 6Sigma= 3,4 errores por millón de unidades.

Esta filosofía se basa en el ciclo iterativo: definir, medir, analizar, mejorar, controlar (DMAIC) (ver figura 1.2) empleada para optimizar los procesos existentes. El objetivo de DMAIC es elevar la calidad (Cambra, 2017).

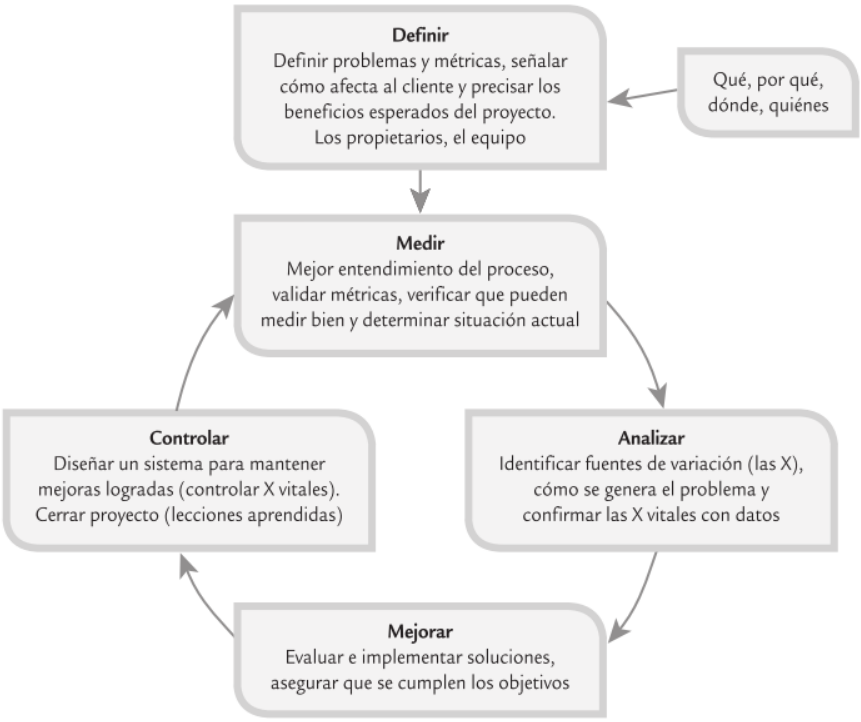


Figura 1.2: Las cinco etapas en la realización de un proyecto Six Sigma Fuente:(Gutiérrez y de la Vara, 2013)

Six Sigma cambia la forma en que las personas piensan y trabajan; cambia la cultura de las corporaciones. Se ha convertido tanto en un modelo de negocio como en una cultura empresarial. Hoy en día, muchas empresas de fabricación y las industrias farmacéuticas exigen el uso de herramientas Six Sigma en su operación comercial diaria. Ingenieros y gerentes de ingeniería necesitan tener un conocimiento más profundo de Six Sigma como más las empresas, agencias gubernamentales y organizaciones adoptan el método ology para mejorar sus productos, procesos y servicios. Six Sigma proporciona un enfoque sistemático para mejorar un proceso, identificando el causa raíz de un problema, encontrar un diseño robusto, realizar estadísticas análisis de datos y evaluación de un sistema de medición, todos los cuales son relevante a las tareas de los ingenieros (Zhan y Ding, 2015).

I.4.3 Aprendizaje Automático (Machine Learning)

El aprendizaje tiene varias definiciones basadas en el contexto en el que se usa y en diversas entidades involucradas en el proceso de aprendizaje. La necesidad de máquinas para aprender y así adaptarse a los cambios en su entorno condujo al surgimiento del

campo acertadamente llamado "aprendizaje automático." Se espera que una máquina aprenda y prediga los resultados futuros basado en los cambios que nota en la estructura externa, datos / entradas alimentados que tendría que ser respondido y el programa / función para el que fue creado. Esto forma la base de las diversas capacidades computacionales complejas que cualquier sistema moderno basado en inteligencia artificial (IA) requeriría e incluye cálculos relacionados con el reconocimiento de patrones, diagnóstico de datos, control de sistema o entorno, actividades de planificación, etc. (Shi y Iyengar, 2020).

Según Sarkar et al. (2018) los principales dominios o campos asociados con Machine Learning son:

• Inteligencia artificial
• Procesamiento natural del lenguaje
• Procesamiento de datos
• Matemáticas
• Estadísticas
• Ciencias de la Computación
• Aprendizaje profundo
• Ciencia de los datos

La Figura 1.3 brinda una buena idea con respecto a estos campos que se superponen con Machine Learning basado en conceptos, metodologías, ideas y técnicas. Un punto importante para recordar aquí es que esto definitivamente no es una lista exhaustiva de dominios o campos, sino que describe los principales campos asociados en tándem con Machine Learning.

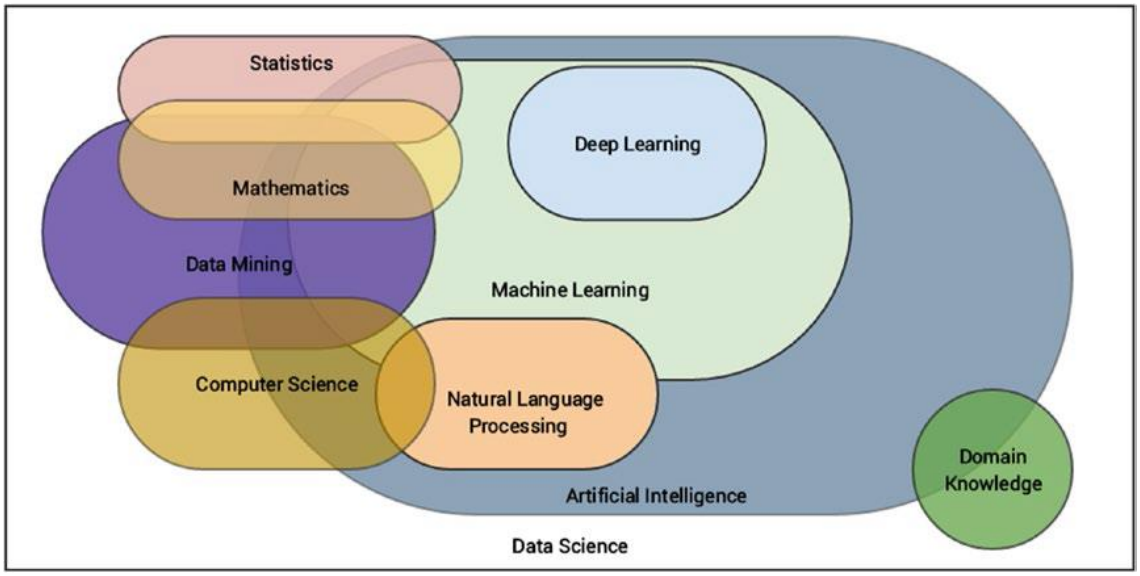


Figura 1.3: Machine Learning: un verdadero campo multidisciplinario Fuente:(Sarkar et al., 2018)

El interés en Machine Learning es desarrollar algoritmos eficientes para diseñar los modelos y también para análisis y predicción. La última parte está ganando importancia en los albores de lo que llamamos la era de los grandes datos, cuando uno tiene que lidiar con grandes cantidades de datos, que pueden estar representados en espacios de gran dimensionalidad. El análisis de datos para tales aplicaciones establece demandas de algoritmos para ser computacionalmente eficiente y al mismo tiempo robusto en su rendimiento, porque algunos de estos datos están contaminados con gran ruido y también, en algunos casos, los datos pueden tener valores faltantes. Dichos métodos y técnicas han estado en el centro de la investigación científica durante varias décadas en diversas disciplinas, como estadística y aprendizaje estadístico, reconocimiento de patrones, informática, minería de datos, visión artificial, bioinformática y diagnóstico médico asistido por computadora, por nombrar algunos (Theodoridis, 2015).

Hay varios escenarios en los que podría ser beneficioso hacer que las máquinas aprendan, según Sarkar et al. (2018) estos son:

• Falta de experiencia humana suficiente en un dominio (por ejemplo, simulando navegaciones en territorios desconocidos o incluso planetas espaciales).

• Los escenarios y el comportamiento pueden seguir cambiando con el tiempo (por ejemplo, disponibilidad de infraestructura en una organización, conectividad de red, etc.).

• Los humanos tienen suficiente experiencia en el dominio, pero es extremadamente difícil que explique o traduzca formalmente esta experiencia en tareas computacionales (por ejemplo, discurso de reconocimiento, traducción, reconocimiento de escenas, tareas cognitivas, etc.).

• Abordar problemas específicos de dominio a escala con grandes volúmenes de datos con muchas condiciones y limitaciones complejas.

Los escenarios mencionados anteriormente son solo varios ejemplos donde hacer que las máquinas aprendan sería más efectivo que invertir tiempo, esfuerzo y dinero en tratar de construir sistemas inteligentes por debajo del par que podrían estar limitado en alcance, cobertura, rendimiento e inteligencia. Nosotros como humanos y expertos en dominios ya es suficiente tener conocimiento sobre el mundo y nuestros respectivos dominios, que pueden ser objetivos, subjetivos, y a veces incluso intuitivo. Con la disponibilidad de grandes volúmenes de datos históricos, podemos aprovechar el Paradigma de Machine Learning para hacer que las máquinas realicen tareas específicas ganando suficiente experiencia, observar patrones en los datos durante un período de tiempo y luego usar esta experiencia para resolver tareas en el futuro con mínima intervención manual. La idea central sigue siendo hacer que las máquinas resuelvan tareas que pueden ser fácilmente definidas intuitivamente y casi involuntariamente pero extremadamente difícil de definir formalmente.

I.4.3.1 Tipos de sistemas de aprendizaje automático

Según Sarkar et al. (2018) hay tantos tipos diferentes de sistemas de Machine Learning que es útil clasificarlos en amplias categorías en función de:

• Si están o no capacitados con supervisión humana (supervisados, sin supervisión, semi-supervisado y aprendizaje de refuerzo).

• Si pueden o no aprender gradualmente sobre la marcha (aprendizaje en línea y aprendizaje por lotes).

• Si funcionan simplemente comparando nuevos puntos de datos con puntos de datos conocidos, o detectar patrones en los datos de entrenamiento y construir un modelo predictivo, mucho como hacen los científicos (aprendizaje basado en instancias y aprendizaje basado en modelos).

En el presente trabajo se abordará con más énfasis al aprendizaje automático supervisado y los distintos algoritmos que este incluye.

Aprendizaje supervisado

En el Aprendizaje supervisado un conjunto de ejemplos de capacitación con las respuestas correctas (objetivos) es proporcionado y, en base a este conjunto de entrenamiento, el algoritmo se generaliza para responder correctamente a todas las entradas posibles. Esto también se llama aprender de los ejemplos (Marsland, 2015).

De acuerdo con Sarkar et al. (2018) los métodos o algoritmos de aprendizaje supervisados incluyen algoritmos de aprendizaje que toman muestras de datos (conocidos como datos de entrenamiento) y resultados asociados (conocidos como etiquetas o respuestas) con cada muestra de datos durante el modelo de proceso de entrenamiento. El objetivo principal es aprender un mapeo o asociación entre muestras de datos de entrada *x* y sus salidas correspondientes y basadas en múltiples instancias de datos de entrenamiento. Este conocimiento aprendido se puede utilizar en el futuro para predecir una salida y para cualquier nueva muestra de datos de entrada *x* que haya sido previamente desconocido o invisible durante el proceso de capacitación del modelo. Estos métodos se denominan supervisados porque el modelo aprende sobre muestras de datos donde las respuestas / etiquetas de salida deseadas ya se conocen de antemano en la fase de entrenamiento.

El aprendizaje supervisado básicamente trata de modelar la relación entre las entradas y sus resultados correspondientes de los datos de entrenamiento para que podamos predecir respuestas de salida para nuevas entradas de datos basadas en el conocimiento que obtuvo anteriormente con respecto a las relaciones y mapeos entre entradas y sus salidas objetivo. Esta es precisamente la razón por la cual los métodos de aprendizaje supervisado se utilizan ampliamente en análisis predictivo donde el objetivo principal es predecir alguna respuesta para algunos datos de entrada que normalmente son alimentado a un modelo de Machine Learning supervisado y entrenado. Los métodos de aprendizaje supervisados son de dos clases principales basadas en el tipo de tareas de LD que pretenden resolver:

• Clasificación (es un enfoque supervisado en el que la máquina aprende de la entrada de datos que se le otorga y posteriormente usa este aprendizaje para clasificar una nueva observación).

• Regresión (se usa para asignar categorías a datos sin etiquetar. Para este enfoque contamos con varias variables predictoras y una variable de respuesta continua, y

tratamos de encontrar una relación entre todas estas variables que nos permita estimar un resultado continuo).

I.4.3.2 Algoritmos del Aprendizaje Supervisado

Support Vector Machines (Maquinas de Soporte Vectorial)

Support Vector Machines (SVM) es un muy potente y versátil modelo de aprendizaje automático, capaz de realizar clasificaciones lineales o no lineales, regresiones e incluso detección de valores atípicos. Las SVM son particularmente muy adecuado para la clasificación de conjuntos de datos complejos pero de tamaño pequeño o mediano (Géron, 2019).

Su rendimiento se basa en encontrar los vectores que definan los hiperplanos de separación entre los grupos, con mayor margen. El modelo SVM determinara una superficie para predecir los valores con el mayor margen posible a ambos lados del hiperplano. En el caso en el que haya que recurrir a una superficie no lineal para predecir, el SVM transforma el espacio de atributos en un espacio lineal de mayor dimensionalidad. Para ello se mapea cada punto del conjunto de datos, mediante una transformación no lineal $\phi(x) : R^2 \rightarrow F$, en el espacio denominado de características F de dimensión mayor, de forma que cada punto x_n es proyectado en un punto $\phi(x_n)$. Si la transformación es escogida adecuadamente, el espacio de características F sería un espacio de mayor dimensionalidad, donde la relación entre el conjunto de datos proyectado y los factores subyacentes a la variabilidad del conjunto de datos original sería lineal.

Este procedimiento de mapear los datos en un espacio de características mediante una transformación no lineal, se agrupan en lo que se denomina métodos basados en kernel. En principio sería muy complicado entrenar un clasificador que pudiera separar ambos conjuntos. Sin embargo, cada punto puede ser mapeado mediante una transformación no lineal $\phi(x)$ en un espacio de características F de dimensión M. Si la transformación no lineal se escoge adecuadamente, ambos conjuntos de datos serían fácilmente separables utilizando como frontera un hiperplano *h* en F (Aguilar et al., 2018).

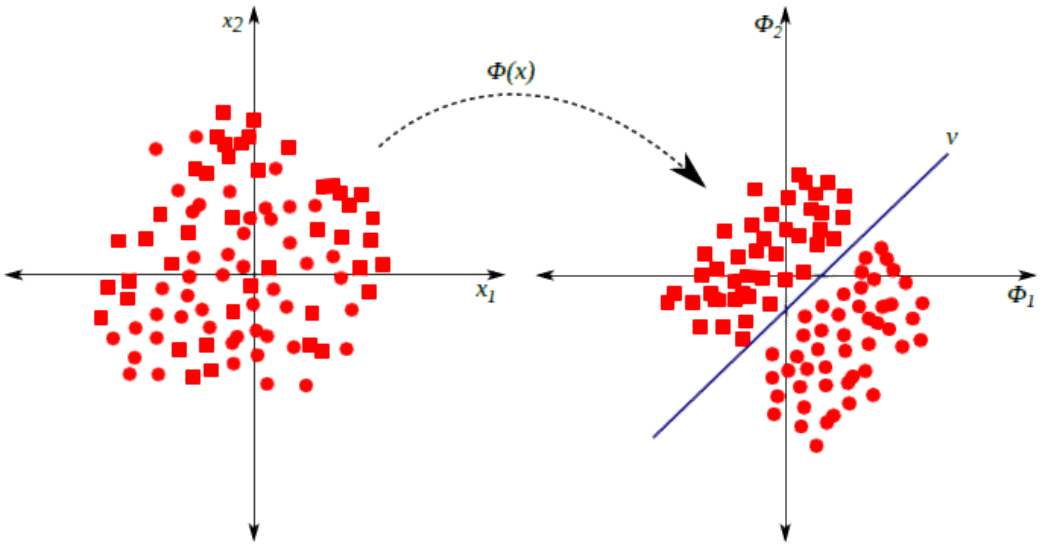


Figura 1.4: En las SVM dos conjuntos de datos difíciles de clasificar en R^2 (izquierda) son proyectado mediante ϕ (x) en el espacio de características (derecha), donde pueden ser clasificados empleando un hiperplano *h* como frontera Fuente:(Aguilar et al., 2018)

Según Sarkar et al. (2018) los SVM se pueden adaptar para su uso con casi cualquier tipo de tarea de aprendizaje, incluida la clasificación y la predicción numérica. Las aplicaciones notables incluyen:

- Clasificación de datos de expresión génica de microarrays en el campo de la bioinformática para identificar cáncer u otras enfermedades genéticas.
- Categorización de texto, como la identificación del idioma utilizado en un documento o la clasificación de documentos por tema.
- La detección de eventos raros pero importantes como fallas en el motor de combustión, brechas de seguridad o terremotos.

Decision Trees (Arboles de decisión)

Los árboles de decisión se ubican dentro de una rama del aprendizaje automático denominada aprendizaje simbólico, en la que también se encuentran los modelos de reglas de decisión, estrechamente relacionados con los árboles (Sampedro y Garcıa, 2012).

Los árboles de decisión son una clase de algoritmo de aprendizaje supervisado que crean un diagrama de flujo que consiste en una secuencia de nodos, donde los valores de una muestra se utilizan para hacer una decisión sobre el siguiente nodo al que ir. El

recorrido por el árbol nos dará la predicción del nuevo dato de entrada. En el proceso de entrenamiento se debe hacer crecer el árbol, decidiendo qué características elegir y qué condiciones usar para dividir, junto con saber cuándo parar. Como un árbol generalmente crece arbitrariamente, se tendrá que recortar para que sea útil. La técnica más utilizada consiste en hacer crecer el árbol con todas las características y se prueban diferentes puntos de división utilizando una función de costo. Se selecciona la división con el mejor costo. Este algoritmo es de naturaleza recursiva ya que los grupos formados se pueden subdividir utilizando la misma estrategia. Esto hace que el nodo raíz sea el mejor predictor (Aguilar et al., 2018).

El uso de árboles de decisión lo podemos encontrar en varios campos, tales como extracción y clasificación de texto, a nivel médico para detección de cáncer, problemas del corazón o detección de diferentes enfermedades como el parkinson o si un paciente debe ser hospitalizado al tener el dengue, o también lo podemos encontrar en los mercados de valores para saber cómo se va a comportar la bolsa (Ruiz, 2018).

Random Forest (Bosques aleatorios)

Otro método basado en conjuntos llamado bosques aleatorios (o bosques de árboles de decisión) se enfoca solo en conjuntos de árboles de decisión. Este método fue defendido por Leo Breiman y Adele Cutler, y combina los principios básicos de embolsado con selección aleatoria de características para agregar diversidad adicional a los modelos de árbol de decisión. Después de que se genera el conjunto de árboles (el bosque), el modelo usa un voto para combinar las predicciones de los árboles. Los bosques aleatorios combinan versatilidad y potencia en un único enfoque de aprendizaje automático. Debido a que el conjunto usa solo una pequeña porción aleatoria de la totalidad del conjunto de características, los bosques aleatorios pueden manejar conjuntos de datos extremadamente grandes, donde la llamada "maldición de la dimensionalidad" puede causar que otros modelos fallen. En él, sus tasas de error para la mayoría de las tareas de aprendizaje están a la par con casi cualquier otro método. Vale la pena señalar que, en relación con otros métodos basados en conjuntos, los bosques aleatorios son bastante competitivos y ofrecen ventajas clave. Por ejemplo, tienden a ser más fáciles de usar y menos propensos al sobreajuste (Lantz, 2019).

K-Nearest Neighbors (Vecinos más cercanos)

K-Nearest Neighbors (K-NN) es un modelo de aprendizaje automático no paramétrico en el que el modelo memoriza la observación de entrenamiento para clasificar los datos de prueba no vistos. También se le puede llamar aprendizaje basado en instancias. Este modelo a menudo se denomina aprendizaje lento, ya que no aprende cualquier cosa durante la fase de entrenamiento como regresión, Random Forest, etc. En cambio, comienza a funcionar solo durante la fase de prueba / evaluación para comparar la prueba dada observaciones con las observaciones de entrenamiento más cercanas, lo que llevará un tiempo significativo comparando cada punto de datos de prueba. Por lo tanto, esta técnica no es eficiente en big data; además, el rendimiento se deteriora cuando el número de variables es alto debido a la maldición de dimensionalidad (Géron, 2019).

Ruiz (2018) afirma que la clasificación para cada nuevo objeto observado se basa en la asignación a la clase que obtenga mayor votación de sus *k* vecinos más cercanos. El valor *k* será el número de vecinos más cercanos con características similares para realizar la comparación y llevar a cabo la elección.

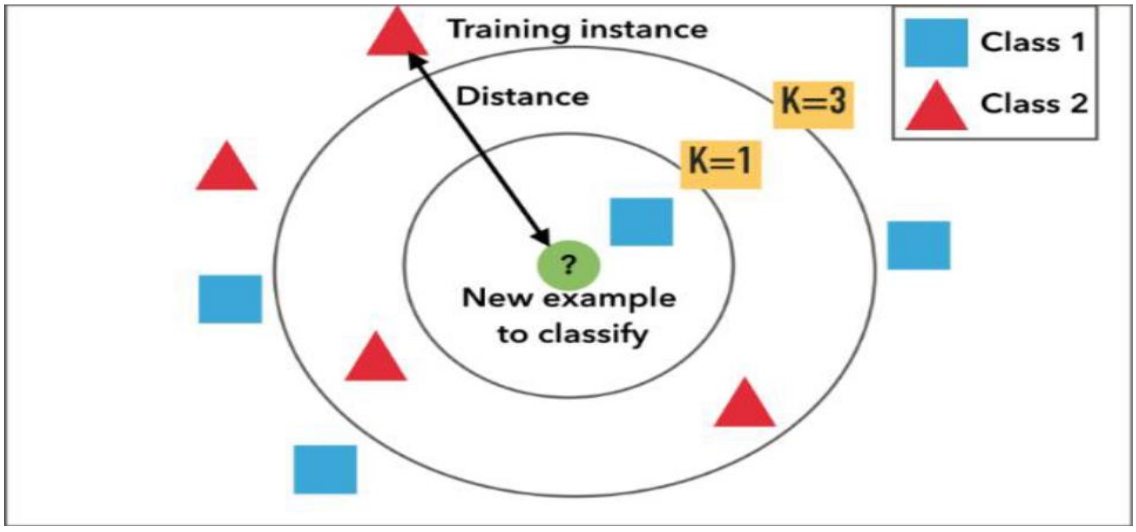


Figura 1.5: Ejemplo de clasificación K-NN Fuente:(Ruiz, 2018)

Como ventajas podemos destacar que K-Nearest Neighbors no necesita una adaptación para clasificación de más de dos clases y que es sencillo de implementar, por otro lado, como inconvenientes encontramos que es muy sensible a datos irrelevantes y la dimensionalidad de los mismos, es sensible al ruido y lento en el caso que tengamos muchos datos de entrenamiento. Es adecuado indicar también la importancia de la

elección de la función distancia para la elección de los vecinos al ejemplo en observación, en la que normalmente se utiliza Euclídea. Aun así, existen otros métodos mejorados para la elección del vecino en este algoritmo usando la indexación del vector distancia.

Naive Bayes (Redes Bayesianas)

El concepto de algoritmo de Bayes es bastante antiguo y existe desde el siglo XVIII desde Thomas Bayes. Thomas desarrolló los principios matemáticos fundamentales para determinar la probabilidad de eventos desconocidos de los eventos conocidos (Dangeti, 2017).

Según Sarkar et al. (2018) el algoritmo Naive Bayes define un método simple para aplicar el teorema de Bayes a los problemas de clasificación. Aunque no es el único método de aprendizaje automático que utiliza métodos bayesianos, es el más común. Creció en popularidad debido a sus éxitos en la clasificación de textos, donde alguna vez fue el factor estándar. Las fortalezas y debilidades de este algoritmo son las siguientes:

Fortalezas:

- Simple, rápido y muy efectivo.
- Funciona bien con datos ruidosos y faltantes.
- Requiere relativamente pocos ejemplos para el entrenamiento, pero también funciona bien con gran cantidad de ejemplos.
- Fácil de obtener la probabilidad estimada para una predicción.

Debilidades:

- Se basa en una suposición a menudo defectuosa de igual importancia y características independientes.
- No es ideal para conjuntos de datos con muchas funciones numéricas.
- Las probabilidades estimadas son menos confiables que las predichas clases.

El algoritmo Naive Bayes se nombra como tal porque hace algunas suposiciones "ingenuas" sobre los datos. En particular, Naive Bayes supone que todas las características del conjunto de datos son igualmente importantes e independientes. Estas suposiciones rara vez son ciertas en la mayoría de las aplicaciones del mundo real.

Neural Networks (Redes Neuronales)

Una red neuronal artificial (ANN) modela la relación entre un conjunto de señales de entrada y una señal de salida utilizando un modelo derivado de nuestra comprensión de cómo un cerebro biológico responde a los estímulos de las entradas sensoriales. Al igual que un cerebro usa una red de células interconectadas llamadas neuronas para proporcionar una gran capacidad de aprendizaje, el ANN utiliza una red de neuronas o nodos artificiales para resolver problemas de aprendizaje desafiantes (Lantz, 2019).

En las neuronas reales, la dendrita recibe señales eléctricas de los axones de otras neuronas, pero en un perceptron, estas señales eléctricas se representan como valores numéricos. En las sinapsis entre la dendrita y los axones, las señales eléctricas se modulan en diversas cantidades. Esto también se modela en el perceptron, multiplicando cada valor de entrada por un valor llamado peso. Una neurona real dispara una señal de salida solo cuando la potencia total de las señales de entrada excede un cierto umbral. Modelamos este fenómeno en un perceptron calculando la suma ponderada de las entradas para representar la intensidad total de las señales de entrada y luego aplicando una función de activación a la suma para determinar su salida. Al igual que en las redes neuronales biológicas, esta salida se alimenta a otros perceptrones. El entrenamiento de un RNA consiste en presentar unos vectores a la entrada e intentar que la salida reproduzca los patrones deseados, mediante la modificación de los pesos según el algoritmo escogido (Aguilar et al., 2018).

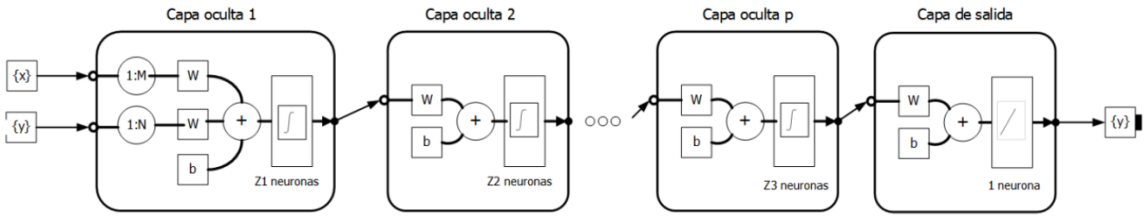


Figura 1.6: Esquema de una red neuronal Fuente:(Aguilar et al., 2018)

Según Du y Swamy (2014) las redes neuronales pueden clasificarse en redes neuronales de avance, redes neuronales recurrentes y sus híbridos. Las topologías de red populares son redes de alimentación directa en capas totalmente conectadas, redes recurrentes, red de redes, redes de alimentación en capas con conexiones laterales y redes celulares.

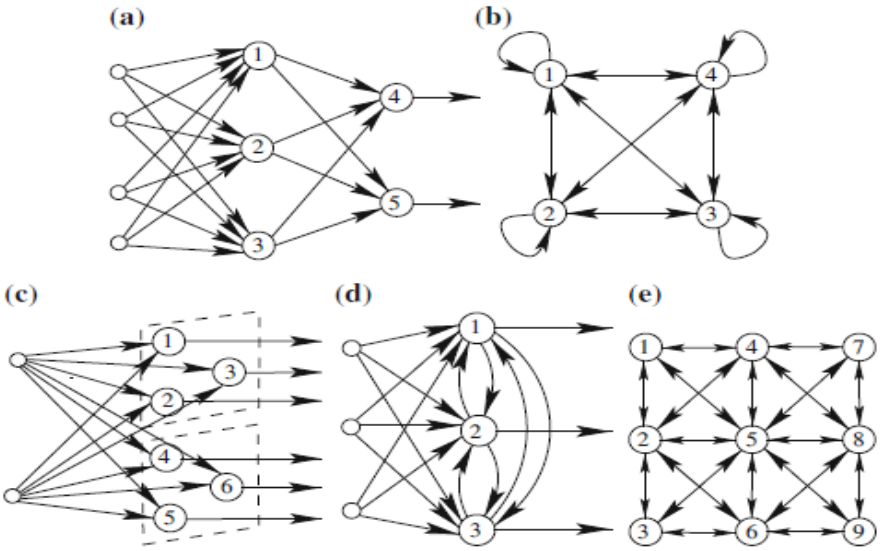


Figura 1.7: Arquitectura de las redes neuronales. a) Redes de alimentación directa. b) Redes recurrentes. c) Red de redes. d) redes de alimentación en capas con conexiones laterales. e) Redes celulares Fuente:(K.-L. Du y Swamy, 2014)

Lantz (2019) afirma que las redes neuronales están biológicamente motivadas. Cada neurona es un nodo computacional, que representa una función no lineal. Las redes neuronales poseen las siguientes ventajas:

• Aprendizaje adaptativo: pueden adaptarse cambiando los parámetros de la red en un entorno circundante.

• Generalización: una red neuronal entrenada tiene una capacidad de generalización superior.

• Naturaleza no lineal de uso general: funcionan como una caja negra.

• Auto organización: Las redes neuronales basadas en el aprendizaje tienen una propiedad de auto organización.

• Paralelismo masivo e implementaciones simples de VLSI: La unidad generalmente tiene una propiedad uniforme. Esta estructura paralela permite un alto paralelismo implementaciones de software y hardware.

• Robustez y tolerancia a fallas: una red neuronal puede manejar fácilmente imprecisos, información difusa, ruidosa y probabilística. Es un sistema de información distribuido, donde la información se almacena en toda la red de manera distribuida por la estructura de red. Por lo tanto, el rendimiento general no se degrada significativamente cuando la información en algunos nodos se pierde o algunas conexiones en la red están dañadas. La red se repara a sí misma y, por lo tanto, posee una capacidad de tolerancia a fallas.

Según Du y Swamy (2014) las redes neuronales pueden tratarse como una herramienta estadística general para casi todas las disciplinas de ciencia e ingeniería. Las aplicaciones pueden estar en modelado e identificación del sistema, clasificación, reconocimiento de patrones, optimización, control, aplicación industrial, comunicaciones, procesamiento de señales, análisis de imágenes, bioinformática y minería de datos. El reconocimiento de patrones es fundamental para la inteligencia biológica y artificial; es un completo proceso que reúne las observaciones, extrae características de las observaciones y clasifica o describe las observaciones. El reconocimiento de patrones es una de las más fundamentales aplicaciones de redes neuronales.

I.5 Vinculación del Aprendizaje Automático con la Calidad

En el mercado global altamente competitivo de hoy, ganar requiere una calidad casi perfecta. Por lo tanto, es necesario cada vez mejores herramientas que garanticen la mejor calidad para los productos y servicios al menor costo posible. En los últimos años con el avance exponencial de las tecnologías y las grandes bases de datos se ha logrado un progreso impresionante del control de la calidad gracias a los algoritmos de aprendizaje automático.

En un estudio realizado por Choi et al. (2020) sobre los aditivos de concreto se utilizaron diversos algoritmos del aprendizaje automático entre ellos las redes neuronales los cuales demostraron ser una herramienta muy útil para determinar la calidad de los aditivos de forma precisa, rápida y garantizar su rendimiento previsto al alterar las propiedades del concreto. En otra investigación desarrollada por Stoyanov y Bailey (2017) sobre la calidad de los productos electrónicos fabricados con técnicas de fabricación aditiva (AM), como la impresión de inyección de tinta 3D, se demostró que los modelos y algoritmos de aprendizaje automático pueden ser utilizados para lograr y mantener una calidad óptima del producto durante las ejecuciones de producción y para realizar el control predictivo del proceso (MPC). Los resultados obtenidos por estos muestran que, para una dinámica moderadamente no lineal del proceso de impresión 3D, los modelos de espacio de estado pueden informar sobre las tendencias (estados) del proceso esperadas y las características de calidad del producto relacionadas, incluso en grandes horizontes de predicción.

Mohammadi y Wang (2016) utilizaron el algoritmo de clasificación Support Vector Machine basándose en los datos recopilados a lo largo de un proceso de fabricación de

material resistente a la abrasión para la predicción de la calidad del producto de las bolas quemadas.

En una investigación desarrollada por Yaseen et al. (2018) sobre el hormigón celular ligero el cual se produce mediante la creación de una estructura celular en una matriz cementosa durante el proceso de mezcla se utilizaron modelos de aprendizaje automático para predecir la resistencia a la compresión del hormigón celular, ahorrando laboriosos lotes de prueba necesarios para lograr la calidad deseada del producto.

Peres et al. (2019) capacitaron y evaluaron varios clasificadores de aprendizaje automático en diversas métricas para predecir defectos dimensionales en una línea de ensamblaje multietapa automotriz real. Los resultados mostraron que los modelos no lineales como XGBoost y Random Forest son capaces de modelar la complejidad de dicho entorno, logrando una alta tasa positiva real y prometen la mejora de los enfoques de control de calidad existentes, lo que permite abordar los defectos y las desviaciones antes y así ayudar a reducir la chatarra y los costos de reparación.

CAPÍTULO II

Capítulo II: Materiales y Métodos

II.1 Descripción del caso de estudio (Empresa GydeMa)

La actual Empresa Productora y Comercializadora de Glucosas y Derivados del Maíz, es conocida comercialmente con el nombre de GydeMa, fue puesta en marcha en enero de 1981 después de un proceso inversionista que duró 6 años y se subordinó por entonces a la rama Confitera del Ministerio de la Industria Alimentaria (MINAL). Es la única de su tipo en el país y fue creada en ese entonces para producir Glucosa ácida como materia prima para la producción de caramelos y la exportación de estos a países del Consejo de Ayuda Mutua Económica.

Se encuentra localizada en la Zona Industrial # 2, en el Reparto Pueblo Griffo de la Ciudad de Cienfuegos, provincia de Cienfuegos, exactamente ubicada en la periferia noreste de la ciudad cabecera de este territorio. Limita al norte con la Empresa DIVEP, al este con la Fábrica de Hielo, Almacenes de Productos Frescos y con la Línea de Ron HRL, por el oeste con la Carpintería en Blanco y el Taller de Ómnibus Escolares, y limitando al sur con el asentamiento poblacional de Pueblo Griffo.

II.1.1 Descripción general del flujo productivo (Producción de Almidón)

Recepción del maíz

El maíz llega a la fábrica por medio de camiones los cuales descargan el maíz en el embudo o pipa de relleno (capacidad del embudo 40t). Mediante un sistema de transportadores horizontales y verticales el maíz es elevado y descargado en los silos. Existen dos silos con un volumen de 446m³ (aproximadamente 600t) lo cual equivale a una producción de 90h, estos silos no están diseñados para almacenar el maíz por un gran espacio de tiempo, su objetivo es compensar fluctuaciones de los camiones.

Limpieza

La limpieza comienza cuando el maíz es transportado de los silos al tamiz vibratorio donde se separan principalmente pajas, pedazos de mazorca, piedras, etc. El tamiz esta combinado con un aspirador el cual retira arena y polvo. Este equipo también separa granos muertos, granos con desperfectos y partículas de menor tamaño que el maíz. Posteriormente el maíz pasa a un tamiz rotatorio donde se realiza una clasificación de control removiendo, granos con desperfectos, piedras y otros materiales. El polvo se envía a un filtro de manga para posteriormente ser usado en la elaboración de pienso al igual que los granos defectuosos. Después el maíz se envía a una balanza (100 kg

capacidad) donde se pesa para después ser elevado (mediante un transportador de carrilones o de cubo) hasta un distribuidor de cadena, el cual distribuye el maíz a los tanques de remojo.

Maceración

El proceso de maceración es la parte fundamental de la producción de almidón de maíz, porque una buena maceración facilita posteriormente la separación de todos los componentes del grano (germen, fibra gruesa, fibra fina, gluten y almidón). El proceso de maceración o remojo se realiza con agua sulfurosa a una concentración aproximada de 1500 ppm la cual se calienta de 50 a 53 ºC para de esta forma eliminar una indeseable actividad microbiana y una solubilización eficiente de las sustancias solubles del maíz. La maceración se realiza con agua sulfurosa porque el azufre rompe los enlaces del almidón, siendo más fácil su separación y la de los demás componentes. En esta sección existen 10 tanques de maceración con un volumen de 45m^3 (aproximadamente 30t) los cuales tienen recirculación propia (de agua) y entre ellos mismos. El agua se alimenta continuamente al tanque que contiene el maíz más viejo (o el de mayor tiempo en esta área) se bombea a través de todos los tanques y se extrae por el tanque donde se encuentra el grano más nuevo. La alimentación y extracción de agua de remojo cambia a nuevos tanques cíclicamente en la medida que los tanques cumplen el ciclo de remojo, esta operación varía entre 40 y 60 h en dependencia del tipo de maíz. Es importante tener en cuenta que primeramente se alimentará el grano a los tanques y después el agua para evitar una explosión de los mismos. Cuando un tanque se esté vaciando de agua, no habrá recirculación a este tanque ni entrada de agua a la sección. Una vez completada la operación el maíz se retira del tanque por el fondo y se envía a un transportador deshidratador (tornillo sin fin) hasta la sección de molienda.

Molienda y Degerminación

El grano llega al primer molino (Molino Previo 1) donde se rasga o se parte el maíz para de esta forma liberar el germen y facilitar después la separación de los demás constituyentes. Posteriormente se pasa a los degerminadores 1 y 2, donde se separa el germen por diferencia de densidad Como el germen es rico en grasa se separa de la mezcla por reboso, mientras que el resto de los componentes van al fondo. La solución acuosa libre de gran parte del germen es transportada por un tornillo sin fin deshidratador hacia el segundo molino (Molino Previo 2) donde se terminan de

desgarrar los granos y se libera el germen que haya quedado. Posteriormente se bombea la solución acuosa hacia el degerminador 3 donde se retira todo el germen restante. Por último, la solución libre de germen se envía al molino fino donde se tritura el grano como tal.

Lavado de Fibra gruesa y Fibra fina

Una vez triturado el grano se bombea al área de lavado de fibra, porque el maíz posee una parte celulósica (la cual es necesario retirar) compuesta por dos tipos de fibra, la gruesa y la fina. Estas fibras se lavan por separado para así obtener mejores resultados a la hora de extraer el almidón que contienen. Primeramente, se procede al lavado de la fibra gruesa en un equipo compuesto de una serie de tornillos sin fin dentro de cilindros de malla, aquí se separa la fibra gruesa y se envía la solución al lavado de fibra fina donde se separa esta última. Los dos tipos de fibra se unen para formar el forraje y la lechada de almidón resultante del proceso de lavado se recepciona en el tanque de almidón crudo y en el tanque misceláneo.

Separación Primaria

La lechada de almidón proveniente de los tanques se envía hacia un sistema de pequeños filtros los cuales retiran impurezas que pudieran dificultar el funcionamiento de la centrifuga encargada de separar el almidón del gluten (proteína insoluble del maíz). Por la fuerza centrífuga el gluten se separa por reboso y el almidón por el fondo se envía a la siguiente etapa.

Refinación

El refinado consiste en terminar de separar el almidón de la proteína, para esto se pasa la lechada por una serie de centrífugas (1er, 2do y 3er refino). En esta sección se lava el almidón con agua a contracorriente y en el 3er refino se envía esta agua a una centrífuga denominada concentrador de middlings (principalmente proteínas y granos de almidón), aquí se separa el agua de la lechada, esta última se envía al tanque de almidón crudo y el agua se reincorpora al proceso. En el 3er refino se recircula lechada al separador primario y el almidón ya refinado se envía al tanque de refino donde parte del almidón será enviado a la producción de glucosa y la restante pasará a la próxima sección.

Desaguado

El almidón refinado tiene una concentración de 18 ºBe y se acondiciona en un rango de 11 a 13 ºBe antes de pasarla al filtro rotatorio al vacío con descarga de cuchilla. Mediante la acción del filtro se obtiene una pasta que se envía mediante un tornillo dosificador a la próxima área. (Capacidad máxima del desagüe 1260 kg de almidón seco/h).

Secado

El almidón proveniente de la sección anterior posee una humedad aproximada de un 43% y se transporta a un secador neumático calentado por vapor, el cual a través de un ventilador con aire caliente eleva el almidón. Una vez en la parte superior del secador se alimenta a dos torres por el costado logrando un efecto de ciclón, de esta manera se seca aún más el almidón y se separa gran parte del aire caliente del mismo. Cuando el almidón llega a la base de las torres una parte se recircula a la etapa del desaguado, de esta forma se facilita el posterior secado de la pasta que retiran las cuchillas. El resto del almidón se envía a un tamiz para separar posibles impurezas y partículas de almidón que no se secaron completamente.

Empaquetamiento

El almidón proveniente del tamiz se almacena temporalmente para después empaquetarlo en sacos de 20 kg.

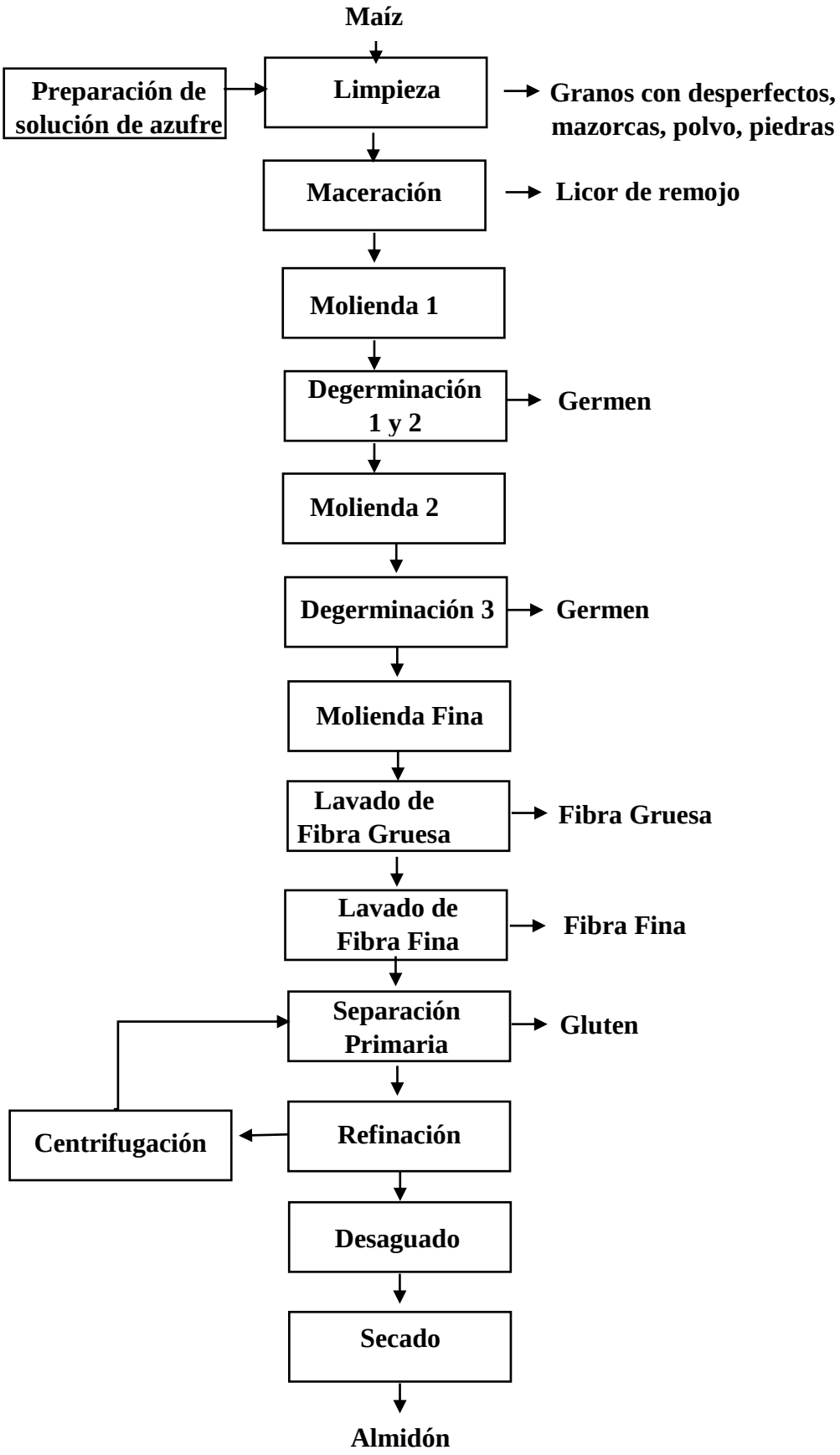


Figura 2.1: Diagrama de Bloques de la producción de Almidón en GydeMa Fuente: Elaboración Propia

II.1.2 Control de la calidad dentro de la empresa

La empresa cuenta con un Área de Normalización, Metrología y Control de Calidad (NMCC) la cual está conformada del siguiente modo:

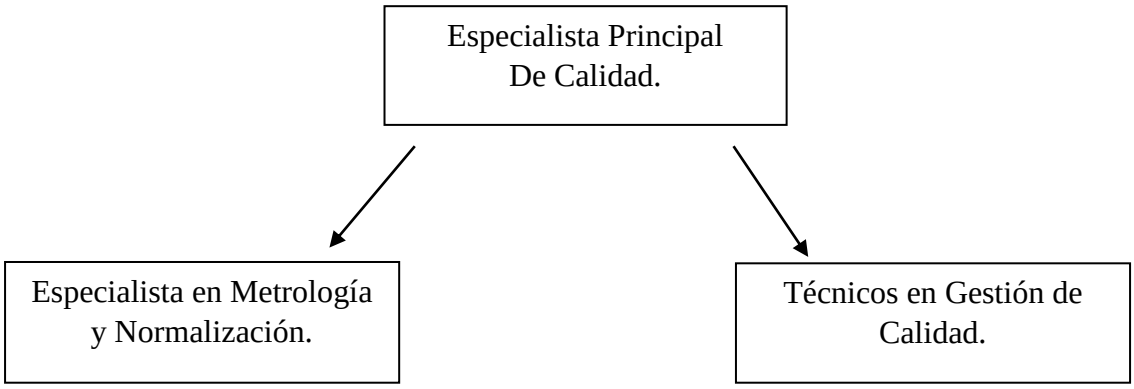


Figura 2.2: Estructura organizacional del Área de NMCC en GydeMa Fuente: Elaboración Propia

El Especialista Principal de Calidad es el encargado de todo el Sistema de Gestión de la Calidad, el cual se rige por normas cubanas y procedimientos que se desprenden de las normas ISO 9001 2008 de Calidad, al cual se subordinan el Especialista en Metrología y Normalización y los Técnicos en Gestión de la Calidad. Estos a su vez se subordinan al Área Técnica Productiva.

Los Técnicos en Gestión de la Calidad se encargan del control de las materias primas, así como de los productos terminados, mientras que el Especialista en Metrología y Normalización es el encargado de llevar un registro de las mediciones y control donde se resumen la cantidad de equipos, distribuidos por magnitud y por rangos así como las mediciones realizadas a diferentes etapas del proceso, además de mantener actualizadas las normas que se cumplen para cada uno de los procesos de la fábrica.

Tabla 2.1: Especificaciones fundamentales de calidad de la materia prima principal

ÍNDICE	MAÍZ YELLOW	MAÍZ PLATA
Humedad (%)	15.0	15.0
Almidón (%)	70.0 min	68.0 min
Acidez (%)	35.0 máx	35.0 máx
Viabilidad (%)	55 min	65 min
Germinación (%)	55 min	65 min

Especificaciones de las normas de calidad del producto terminado.

- Aspecto: Polvo fino de color blanco, amarillento y libre de materias extrañas.
- Olor: Característico, libre de olores extraños.

Especificaciones Físico – Químicas:

- Humedad en % máx (B.S) _____ 9.0 -13.00.
- Cenizas en % máx (B.S) _____ 0.30
- Proteínas en % máx (B.S) _____ 1.00 (Solo para uso en mezcla seca.).
- Proteínas en % máx (B.S) _____0.80 (Niños enfermos.) y comercialización.
- Grasas en % máx (B.S) _____ 0.50
- Dióxido de azufre mg/L máx _____ 70.00
- pH _____ 5.0-7.0

II.2 Metodología propuesta

Mediante un análisis de la literatura se comprobó que el desarrollo y despliegue de modelos de aprendizaje automático implica una serie de pasos que son casi similares al proceso de modelado estadístico, para desarrollar, validar e implementar los modelos. Los pasos según (Dangeti, 2017) son:

1. Recopilación de datos: los datos para el aprendizaje automático se recopilan directamente de datos fuente estructurados, desguace web, API, interacción de chat, etc., el aprendizaje automático puede funcionar tanto en datos estructurados como no estructurados (voz, imagen y texto).

2. Preparación de datos y tratamiento perdido / atípico: los datos deben formatearse según el algoritmo de aprendizaje automático elegido; además, el tratamiento de valor perdido debe ser realizado mediante la sustitución de valores perdidos y atípicos con la media / mediana.

3. Análisis de datos e ingeniería de características: los datos deben analizarse para encontrar patrones ocultos y relaciones entre variables, etc. La ingeniería de características con el conocimiento comercial apropiado resolverá el 70 por ciento de los problemas.

4. Algoritmo de entrenamiento sobre datos de entrenamiento y validación: Los datos se dividirán en tres fragmentos (datos de entrenamiento, validación y prueba) en lugar de dos (entrenar y probar) en modelado estadístico. El aprendizaje automático se aplica en

la formación. Los datos y los hiperparámetros del modelo se ajustan según los datos de validación para evitar el sobreajuste.

5. Prueba del algoritmo en los datos de la prueba: una vez que el modelo ha demostrado ser lo suficientemente bueno, su rendimiento será verificado con datos de prueba no vistos. Si el rendimiento sigue siendo lo suficientemente bueno, podemos proceder al siguiente y último paso.

6. Implementar el algoritmo: se implementarán algoritmos de aprendizaje automático entrenados en transmisión de datos en vivo para clasificar los resultados.

Tomando lo anterior como referencia se propone la metodología de la figura 2.3 donde mediante un diagrama heurístico se destacan paso a paso las fases para la utilización de estos algoritmos en la empresa GydeMa.

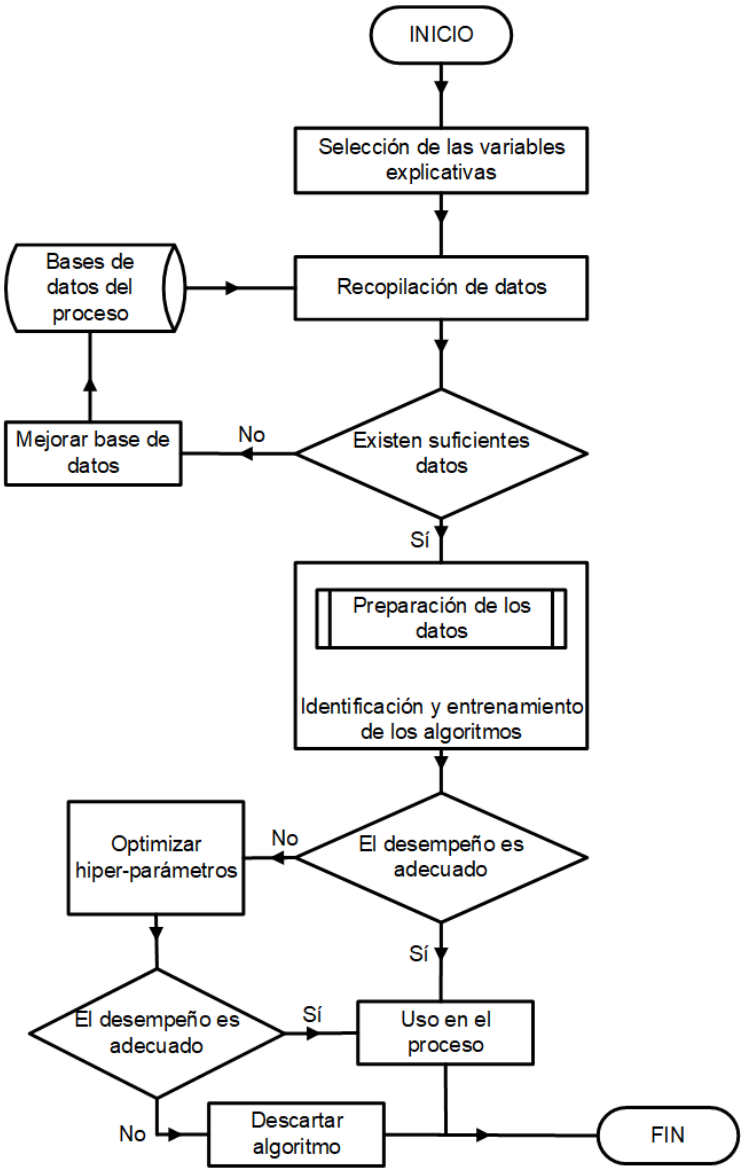


Figura 2.3: Metodología propuesta

Hay que destacar que el proceso de selección de las variables explicativas puede ser un proceso complejo. En el caso de la empresa estas pueden ser los parámetros de especificación de la calidad de materias primas y producto final, pero en dependencia de los objetivos del estudio puede ser necesario la inclusión de variables de control del proceso como presión, temperatura, etc.

II.3 Descripción de las herramientas de aprendizaje automático

II.3.1 Regresión lineal

La regresión lineal es una técnica paramétrica utilizada para predecir variables continuas, dependientes, dado un conjunto de variables independientes. Es de naturaleza paramétrica porque hace ciertas suposiciones basadas en el conjunto de datos.

Según Sarkar et al. (2018) la regresión utiliza una función lineal para aproximar o predecir la variable dependiente dada como:

$$y = a * x + b$$

Donde *y* es la variable dependiente, *x es* la variable independiente. El símbolo *a* denota la intersección de la línea de regresión y *b* es la pendiente de la misma.

El objetivo con regresión lineal es minimizar la distancia vertical entre todos los datos y nuestra línea, por lo tanto, para determinar la mejor línea debemos minimizar la distancia entre todos los puntos y la distancia de nuestra línea.

Una de las formas en que el modelo de regresión encuentre la mejor línea de ajuste es utilizando el criterio de mínimos cuadrados para reducir el error. El algoritmo intenta minimizar el error con respecto a la pendiente y la intercepción. Utiliza la forma de error al cuadrado, se muestra de la siguiente manera:

$$q = \sum (y_{observed} - y_{predicted})^2$$

donde, *q* es el error cuadrado total. Minimizamos el error total para obtener la pendiente e interceptar la mejor línea de montaje.

En la actualidad la regresión lineal simple es utilizada en muy pocos escenarios debido a que por lo general una variable dependiente depende de múltiples factores.

II.3.2 Regresión lineal múltiple

La regresión múltiple también se conoce como regresión multivariable. Estos métodos intentan modelar datos donde tener una variable de salida de respuesta *y* en cada

observación, pero múltiples variables explicativas en forma de vector *x* en lugar de una sola variable explicativa. La idea es predecir *y* basándose en las diferentes características presentes en *x* (Aguilar et al., 2018).

La ecuación de la regresión lineal múltiple es:

$$y = a_1 * x_1 + a_2 * x_2 + a_n * x_n + b$$

Donde *y* es la variable dependiente, b es la intercepción, $x_{1\ldots n}$ son las variables independientes y $a_{1\ldots n}$ sus respectivos coeficientes.

Generalmente cuando se trata de regresión lineal múltiple lo primero que se debe hacer es enfocarse en seleccionar las mejores variables independientes que puedan contribuir a la variable dependiente, para esto debemos construir una matriz de correlación para todas las variables independientes e incluimos la variable dependiente. El valor de correlación nos da una idea de que variable es significativa, a partir de esta matriz elegimos las variables independientes en orden decreciente de valor de correlación y ejecutamos el modelo de regresión para estimar los coeficientes minimizando la función de error. Nos detenemos cuando no hay mejora destacada en la función de estimación mediante la inclusión de la siguiente característica independiente.

Algo importante a tener en cuenta al utilizar regresión lineal múltiple es que agregar más variables independientes no significa que la regresión sea mejor u ofrece mejores predicciones, en algunos casos agregar variables independientes puede empeorar las cosas, a esto se le conoce como ajuste excesivo. Por otra parte, cuando se agrega más variables independientes se crean relaciones entre ellas, es decir, no solo las variables independientes están relacionas con la variable dependiente, sino que también están potencialmente relacionadas entre sí, a esto se conoce como multicolinealidad. El escenario óptimo es que todas las variables independientes se correlacionen con la variable dependiente pero no entre sí.

II.3.3 Regresión Polinomial

La regresión polinómica es un caso especial de regresión múltiple donde se modela la variable de respuesta *y* como un polinomio de décimo grado de la característica de entrada *x*. Básicamente es una regresión múltiple, donde cada característica en el vector de características de entrada es un múltiplo de *x* (Géron, 2019).

La ecuación para la regresión Polinomial es:

$$y = a_1 * x_1 + a_2 * x_1^2 + b$$

Donde *y* es la variable dependiente, *b* es la intercepción, $x_{1\ldots n}$ son las variables independientes y $a_{1\ldots n}$ sus respectivos coeficientes.

Al utilizar la regresión polinomial hay que tener en cuenta:

- A medida que aumentamos la complejidad de la formula, el número de características también aumenta, lo que a veces es difícil de manejar.
- Tiene una tendencia a ajustarse drásticamente incluso en un simple conjunto de datos unidimensional.
- Los Polinomios superiores pueden terminar produciendo resultados extraños en la extrapolación.
- Cambiar el valor de *y* en un punto del conjunto de entrenamiento puede afectar el ajuste del polinomio para puntos de datos que están muy lejos.

La regresión polinomial es muy similar a la regresión lineal, con una ligera desviación en que tratamos nuestro espacio de características.

II.3.4 Support Vector Machine

La Máquina de Vectores de Soporte (SVM) puede ser usada tanto para métodos de clasificación como de regresión. SVM busca optimizar el proceso de clasificación y/o regresión, estableciendo una línea de decisión que separe las clases maximizando el margen entre esta línea y los puntos más cercanos a este hiperplano; y así clasificar claramente los puntos en el plano. La SVM tiene la capacidad de modelar problemas complejos lineales y no lineales y generar resultados altamente precisos, incluso en datos ruidosos. Sin embargo, su proceso de entrenamiento puede llevar mucho tiempo en un gran volumen de datos (Ríos et al., 2019)

Según Gala (2013) Support Vector Machine (SVM) es originalmente creado para problemas de clasificación binaria que, dado un set de entrenamiento *[(x1; y1);…; (xn; yn)] con yi = ±1* en un espacio de *features* de dimensión *d*, busca construir hiper-planos, es decir hiper-superficies de dimensión d - 1, que separen las clases. Matemáticamente, podemos definir un hiper-plano como:

$$x: f_{(x)} = x^T\beta + \beta_0 = 0$$

Donde es un vector unidad, *x* son los *features* y x^T es simplemente el vector transpuesto. Luego, podemos definir una regla de clasificación teniendo en cuenta $sgn(f(x) = x^T\beta + \beta_0)$ (se puede demostrar que *f(x)* es geométricamente la distancia

del punto *x* al hiper-plano). Suponiendo que las clases son perfectamente separables (Ver figura 2.3 (a) y (b)) podemos encontrar un tal que $y_i f(x_i) > 0 \forall_i$.

Para lidiar con problemas en donde las clases no sean perfectamente separables introducimos variables de costos ε_i que cuantifican el error cometido en dicha clasificación (ver figura 2.3 (c)) para luego imponer un máximo en la suma de los costos $\sum_i \varepsilon_i < C$.

Esta técnica es conocida como *Soft Margin* y permite que los modelos *SVM* no sobre ajusten los datos y, por lo tanto, tengan una mejor generalización.

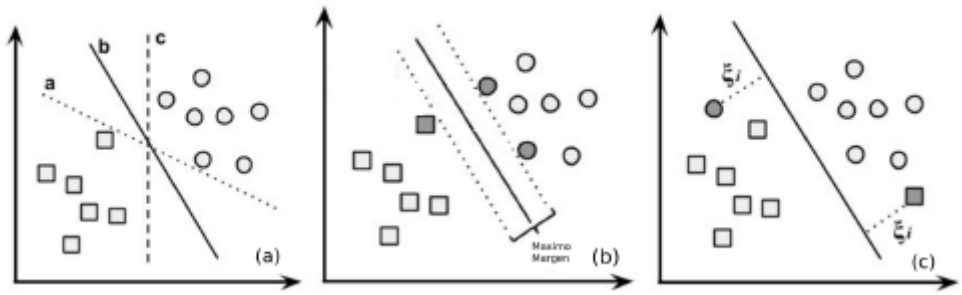


Figura 2.4: Esquema de una clasificación mediante Support Vector Machine Fuente: (Gala, 2013)

Como puede verse en la figura 2.3 (a), hay casos en los que existen varios hiper-planos que separan las clases y que pueden ser utilizados como referencia para la clasificación. Finalmente nos quedamos con aquel hiper-plano que maximice la separación o margen entre las clases (Ver figura 2.3 (b)). Matemáticamente esto se traduce en:

$$y_i \left(x_i^T \beta^{\beta, \beta_0}{}^{máx\ M} + \beta_0 \right) \geq M(1 - \varepsilon_i)$$

$$\sum_i \varepsilon_i \leq C$$

donde M es el margen entre las clases. Se puede demostrar que el hiper-plano va a depender solo de un subconjunto de observaciones (llamadas *support vectors*) que geométricamente caen adentro, o sobre el margen que separa ambas clases.

Es importante notar que este algoritmo busca hiper-plano lineales en el espacio de *features*. Una forma de generalizar dicho procedimiento es agrandar el espacio de f*eatures* a través de un conjunto de funciones bases *h(x)*. Así los hiper-planos se buscarán en un nuevo espacio de *features* definido por $h(x_i) = (h_1(x_i), h_2(x_i), \ldots)$ traduciéndose en superficies no lineales en el espacio de *features* original.

Como se dijo previamente, los métodos *SVM* fueron creados para resolver problemas de clasificación binaria, sin embargo, existen generalizaciones para resolver problemas de clasificación múltiple y problemas de regresión.

Suponiendo que queremos clasificar entre *K* clases diferentes, existen 2 tipos de generalizaciones. La primera opción es generar un clasificador binario $f_k(x)$ por cada clase a estudiar. Cada clasificador f_k tratará como una clase a las observaciones que pertenecen a la clase *k* y como otra clase a las observaciones que pertenecen a todas las otras clases originales.

Finalmente tendremos *K* clasificaciones para cada observación, lo cual puede dar lugar a ambigüedades en dicha clasificación. Otro problema que puede aparecer en dicha generalización es el desbalance entre las clases. Esto quiere decir que puede suceder que para una dada clase *k* existan pocas observaciones que pertenezcan a dicha clase comparado a la cantidad de observaciones que no pertenecen a dicha clase.

La segunda opción para generalizar las técnicas *SVM* para clasificación múltiple, es generar un clasificador $f_{k;k'}$ que estudie límites entre cada par de clases. De esta manera tendremos $K(K-1)/2$ clasificadores, lo que, de vuelta, puede resultar en ambigüedades a la hora de clasificar una nueva observación.

Las Máquinas de Soporte Vectorial adaptadas para resolver problemas de regresión se conocen por el acrónimo SVR (del inglés Support Vector Regression). Los vectores de soporte de regresión utilizan los mismos principios que de clasificación con solo algunas diferencias menores. Dado que la salida es un número real, se vuelve muy difícil predecir la información disponible, que tiene infinitas posibilidades, sin embargo, la idea principal es la misma, minimizar el error, individualizar el hiperplano que maximiza el margen teniendo en cuenta que se tolera parte del error. Este algoritmo funciona tanto para datos linealmente separables como datos no linealmente separables (Gala, 2013).

En los datos linealmente separables se utiliza el margen (γ) como el hiperplano para separar los datos, y lo definimos a partir del supuesto que para el conjunto de entrenamiento (x_i, y_i), *i=1,…,m,* con $x_i \epsilon \mathbb{R}$ e $y_i \epsilon \{-1,1\}$, existe un hiperplano que separa los datos, de la forma:

$$f_{(x)} = xw + b$$

El margen sería entonces, las distancias de los puntos más cercanos al hiperplano y el principal objetivo a conseguir es maximizar el margen que se define de la siguiente manera:

$$\gamma = \frac{1}{2}\left(\frac{\omega}{\|\omega\|_2}.x^+ - \frac{\omega}{\|\omega\|_2}.x^-\right)$$

Para los datos no linealmente separables, que son la mayoría de casos en los que trabajamos con datos reales, encontrar un hiperplano óptimo que separa de forma adecuada los datos no es tarea fácil. En este problema se introduce un condicionante que permite tener un modelo menos rígido, permitiendo que exista un error aceptable. Este error lo denotamos con $\epsilon \geq 0$, siendo ahora el supuesto de la forma :

$$y_i(\omega \cdot x_i + b) - 1 + \varepsilon_i \geq 0, \forall_i$$

Y el problema de optimización de la forma:

$$min\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(\xi + \xi^*)$$

Donde *w* es la magnitud del vector o hiperplano, *C* es una constante y debe ser mayor a 0, determina el equilibrio entre la regularidad de la función y la cuantía hasta la cual toleramos desviaciones mayores que las bandas de soporte, ξ y ξ^* son variables que controlan el error cometido por la función de regresión al aproximar a las bandas. Si el valor de la constante *C* es muy grande consideramos que el conjunto está perfectamente representado por nuestro hiperplano predictor y si el valor de *C* es muy pequeño estaríamos admitiendo un número muy elevado de ejemplos mal representados.

En espacios de no separables, podemos necesitar hiperplanos de mayor dimensión que permitan convertir al problema de datos no separables mediante una proyección a datos linealmente separables. Esto es posible utilizando las funciones de Kernel, que permiten proyectar la información a un espacio de características de mayor dimensión. Según Anzola (2016) las funciones kernel frecuentemente utilizadas por medio de SVM son:

Kernel lineal: este se recomienda cuando hay vectores de datos dispersos.

$$K(x_i, x_j) = x_i * x_j$$

Kernel polinomial: este es un Kernel muy utilizado para modelar relaciones no lineales. Sin embargo, a medida que aumenta el parámetro *d* (grado del polinomio) la superficie de clasificación se hace más compleja.

$$K(x_i, x_j) = (1 + x_i * x_j)^d$$

Kernel gaussiano: Kernel de base radial o gaussiano es uno de los más utilizados. El parámetro σ controla la forma del hiperplano de separación y este puede optimizarse a partir de métodos de validación cruzada:

$$K(x_i, x_j) = exp\left(-\frac{\left||x_i - x_j|\right|^2}{2\sigma^2}\right)$$

Otras funciones Kernel:

Neural (Sigmoid, Tanh) Kernel:

$$K(x_i, x_j) = tanh(ax_i * x_j + b)$$

Anova Kernel:

$$K(x_i, x_j) = \left(\sum_i exp\left(-\gamma(x_i - x_j)\right)\right)^d$$

Fourier Series Kernel:

$$K(x_i, x_j) = \frac{\sin(N + \frac{1}{2})(x_i + x_j)}{\sin(\frac{1}{2}(x_i - x_j))}$$

Spline Kernel:

$$K(x_i, x_j) = \sum_{r=0}^{k} x_i^r x_j^r + \sum_{s=1}^{N} (x_i - t_s)^k + (x_j - t_s)^k$$

El rendimiento del algoritmo de Vectores de Soporte Regresion depende de una buena configuracion de los parametros de *C* y de los del Kernel.

II.3.5 Decision Trees

Los árboles de decisión son algoritmos de aprendizaje automático supervisado que pueden ser utilizados tanto para problemas de clasificación como de regresión. La idea

de estos métodos es subdividir el espacio de datos de entrada (*features*) en subregiones y luego ajustar un modelo local a cada uno de estos subconjuntos. Esta idea puede ser representada a través de un árbol en donde el espacio de entrada es subdividido en 2 regiones en cada nodo del árbol, llegando finalmente a subdividir el espacio de *features* en 5 regiones en las cuales se ajusta un modelo local teniendo en cuenta los datos del set de entrenamiento que caen en cada región (Friedman et al., 2001).

Según Carranza et al. (2018)dado una base de datos de tamaño N, con p *inputs* (x) y un *output* (y), por cada observación (n) del total N. Esto es, $(xi,i)\ \forall\ i$=1,2,…,N con X=(xi1,xi2,…,xip). El proceso determina cuál es el input divisor (la variable independiente por la cual se genera la subdivisión) y el criterio de división (el valor por el cual el árbol se rige para generar nuevos nodos).

$$f(x) = \sum_{j=1}^{M} C_j I \qquad \forall\ x \in R_j$$

Si se adopta como criterio de minimización la suma de los cuadrados Σ $(yi-f(x))$, es fácil observar que el mejor valor para Cmj es el promedio de yi en la región Rm:

$$\widehat{C_j} = \frac{\sum_{i=1}^{n}(y_i|x_i)}{n} \forall\ i\ \in\ R_j$$

Para encontrar la mejor partición del árbol en términos de la suma mínima de cuadrados, considerando toda la muestra, siendo k la variable divisora y s el criterio, se define el par de semi-planos:

$$R_1(k,s) = \{X|X_k\ \leq s\}\ y\ R_2(k,s) = \{X|X_k\ > s\}$$

El objetivo consiste en buscar la variable divisora k y el criterio de partición s que resuelva la siguiente ecuación:

$$min_{k,s}\left[min_{c_1}\sum_{x_{i\in R_1\,(k,s)}}(y_{i-C_1})^2 + min_{c_2}\sum_{x_{i\in R_2\,(k,s)}}(y_{i-C_2})^2\right]$$

Al elegir y minimizar k y s, se obtienen los valores mínimos para Cj correspondientes para cada nodo y así se genera una nueva partición en dos subnodos. Luego de encontrar la mejor partición, se dividen los datos en los dos subconjuntos resultantes y se repite el proceso en cada uno de ellos hasta que no exista suficiente heterogeneidad para continuar el algoritmo o hasta que el tamaño de los nodos sea inferior a un mínimo establecido a priori.

De esta manera, cada hoja de un árbol de regresión contiene aquellas observaciones lo suficientemente similares como para descartar la necesidad de generar nuevos nodos, representando un subconjunto homogéneo en función de los parámetros iniciales fijados en el algoritmo. El promedio del output para los datos de cada nodo se puede tomar como una predicción adecuada de aquellos valores ajenos a la muestra, para los cuales se desconoce el output, pero que tienen valores de inputs similares a los datos del nodo.

La varianza de los datos de cada nodo se puede tomar como una medida de impureza en la estimación. La razón para usar la varianza como medida de impureza se justifica en el hecho que el mejor estimador del *output* en un nodo es el promedio del valor de la variable predicha en las observaciones que corresponden a dicho nodo. Por lo tanto, la varianza es el error cuadrático medio del promedio utilizado como estimador.

Si bien el modelo matemático presentando solo sirve para problemas de regresión, se puede generalizar fácilmente a problemas de clasificación teniendo en cuenta la distribución de observaciones de diferentes clases que hay en cada subregión y definiendo la probabilidad de que una nueva observación pertenezca a la clase *K* en dicha región de manera frecuentista:

$$P(x \epsilon K | x \epsilon R_m) = \frac{N_{K,R_m}}{N_{R_m}}$$

donde N_{K,R_m} es la cantidad de observaciones de clase *K* en R_m y N_{R_m} es la cantidad de observaciones en R_m. Luego podemos clasificar una nueva observación teniendo en cuenta cual es la clase más probable.

Uno de los principales problemas de los árboles de decisión es su inestabilidad, esto quiere decir que pequeños cambios en el set de entrenamiento pueden llevar a predicciones muy diferentes. Es por esto que en general conviene entrenar muchos árboles de decisión y luego promediar el resultado de cada uno para obtener un resultado final. El algoritmo conocido como *Random Forest* es una implementación de esta técnica (Breiman, 2001).

II.3.6 Random Forest

Este método consiste en entrenar **N** árboles de decisión, pero seleccionando en cada nodo $m < p$ *features* de manera aleatoria que serán estudiados para sub-dividir el espacio de entrada como fue explicado anteriormente. De esta manera se reduce la correlación entre los diferentes árboles. Si bien el valor de *m* es un parámetro que puede ser ajustado a cada problema para obtener un mejor rendimiento, es común usar

$m = \sqrt{p}$ o $m = {}^{p}/_{3}$ para problemas de clasificación y regresión respectivamente (Brownlee, 2016).

Es un gran algoritmo para entrenar temprano en el proceso de desarrollo del modelo, para ver cómo se desempeña y es difícil construir un mal modelo con este algoritmo debido a su simplicidad.

II.3.7 Redes Bayesianas

Formalmente, la estructura de una red bayesiana sobre un conjunto de variables aleatorias $x_1, \ldots, x_n, C$ es un grafo acíclico dirigido cuyos vértices corresponden a las variables, cuyos arcos codifican las dependencias e independencias probabilísticas entre tripletas de variables y, en cada uno de los vértices, se representa una distribución categórica local $p(x_i|pa(x_i))$ o $p(c|pa(c))$ donde $pa(x_i)$ es un conjunto de valores para el conjunto de variables Pa(*Xi*), que son los padres de la variable *Xi* en el modelo gráfico. Lo mismo se aplica para $pa(c)$ (Bielza y Larrañaga, 2014). Por lo tanto, la factorización que permite llevar a cabo la red bayesiana de la probabilidad conjunta de todas las variables aleatorias y que trata de evitar estimar un número exponencial de parámetros es la siguiente:

$$p(x, c) = p(c|\text{pa}(c)) \prod_{i=1}^{n} p(x_i|\text{pa}(x_i))$$

Las redes bayesianas, cuando se utilizan con propósitos de realizar tareas de clasificación, reciben el nombre de clasificadores bayesianos. En los clasificadores bayesianos, el objetivo es asignar la clase más probable a una instancia determinada, definida por un conjunto de valores de las variables predictoras. En términos probabilísticos, se asigna a una instancia de prueba la etiqueta de clase con la mayor *probabilidad a posteriori (MA P)*. Es decir:

$$arg_c \, max \, p(c|\mathbf{x})$$

Utilizando la regla de Bayes, podemos relacionar los términos de las ecuaciones anteriores y, además, puesto que el objetivo es calcular el valor de *C* con mayor *probabilidad a posteriori*, no es necesario tener en cuenta el denominador en la regla de Bayes (el factor de normalización). De esta manera, obtenemos la siguiente expresión:

$$arg_c \, max \, p(c|\mathbf{x}) = arg_c \, max \, p(\mathbf{x}|c)$$

Así, podemos utilizar esta ecuación para hallar la clase con la mayor *probabilidad a posteriori*. Esta ecuación establece el caso general de los clasificadores bayesianos, en

el que *p(x, c)* se puede factorizar de diferentes maneras. Para llevar a cabo esta factorización tenemos que buscar lo que se denomina el manto de Markov (*Markov blanket*) de la variable *C* para encontrar la solución de la ecuación anterior. El manto de Markov se define como el conjunto de variables MBc que hacen que, dado dichas variables, la variable *C* sea condicionalmente independiente de las demás variables de la red bayesiana. El manto de Markov está formado, tomando a la variable *C* de referencia, por los padres, los hijos y los padres de los hijos de dicha variable. De esta forma:

$$p(c|\mathbf{x}) = p\left(c\middle|\mathbf{x}_{MB_c}\right)$$

Para el caso específico en el que la variable *C* no tenga padres y, utilizando la regla de la cadena, la probabilidad conjunta de las variables predictoras y de la variable clase se puede expresar de la siguiente manera:

$$p(\mathbf{x}|c) = p(c)p(\mathbf{x}|c)$$

de tal forma que el objetivo es maximizar en *c*.

Los distintos clasificadores bayesianos que se diferencian en la forma en la que factorizan $(p(x)|c)$. Los clasificadores más conocidos son naive Bayes, tree agumented naive Bayes (TAN), k-dependence Bayesian classifier (k-DB), semi-naive Bayes y Bayesian multinet.

II.3.8 Redes Neuronales

Para entender en mayor detalle este método es necesario estudiar el funcionamiento de cada neurona por separado. Cada neurona recibe un conjunto de datos de entrada x_i y a partir de estos genera una combinación lineal $x_i w_i$ de los mismos, donde w_i es un vector de pesos que depende de cada neurona. Además cada neurona consta de una función de activación $f(x.w)$ que recibe los datos de entrada pesados por *w* y produce un valor entre 0 y 1. Dependiendo de la forma de esta función es el nombre que recibe la neurona. Las neuronas más simples son llamadas perceptrones y están caracterizadas por una función de activación donde $f(x.w) = sgn(x.w)$(Labajo y Labajo, 2020).

Generalmente, según Anzola (2016) la función de activación puede ser de tipo escalón, lineal, sigmoidal y gaussiana. La figura 2.5 ilustra las principales funciones de activación, sus intervalos y las gráficas correspondientes.

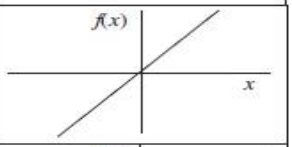
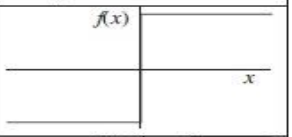
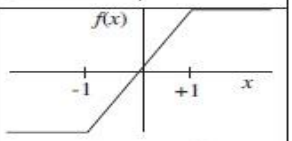
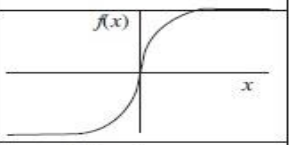
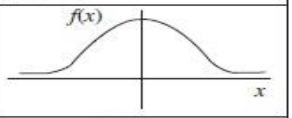
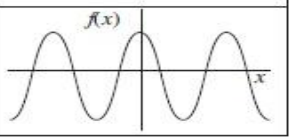
	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty,+\infty]$	
Escalón	$y = sign(x)$ $y = H(x)$	$\{-1,+1\}$ $\{0,+1\}$	
Lineal a tramos	$y = \begin{cases} -1, si\ x < -l \\ x, si\ +l \le x \le -l \\ +1, si\ x > +l \end{cases}$	$[-1,+1]$	
Sigmoidea	$y = \frac{1}{1+e^{-x}}$ $y = tgh(x)$	$[0,+1]$ $[-1,+1]$	
Gaussiana	$y = Ae^{-Bx^2}$	$[0,+1]$	
Sinusidal	$y = A\ sen(\omega x + \varphi)$	$[-1,+1]$	

Figura 2.5: Principales funciones de activación Fuente: (Anzola, 2016)

Habiendo definido la arquitectura y el tipo de neuronas que se van a utilizar, faltan definir los pesos *w*, correspondientes a cada neurona, que producen el mejor resultado. Para esto el algoritmo más utilizado es el llamado *backpropagation* que consiste en ir determinando los errores cometidos capa a capa, empezando por la capa de salida.

El algoritmo *backpropagation* funciona de la siguiente manera: el algoritmo tiene dos fases, una hacia adelante y una hacia atrás. En la primera fase, el patrón de entrada se presenta a la red y se propaga a través de las capas hasta llegar a la capa de salida, obteniéndose los valores de salida de la red. La segunda fase inicia cuando se comparan los valores obtenidos con los valores de salida esperados para así obtener el error. La segunda fase transmite hacia atrás el error, a partir de la capa de salida, hacia todas las neuronas de la capa intermedia que contribuyan directamente a la salida, recibiendo el porcentaje de error aproximado a la participación de la neurona intermedia en la salida original. Este proceso se repite, capa por capa, hasta que todas las neuronas de la red hayan recibido un error que describa su aportación relativa al error total. Basándose en el valor del error recibido, se reajustan los pesos de conexión de cada neurona, de

manera que la siguiente vez que se presente el mismo patrón, la salida esté más cercana a la deseada y, por tanto, disminuya el error (López Sotelo y Caicedo Bravo, 2009).

El algoritmo finaliza cuando se verifica su condición de parada, ya sea porque el error calculado de la salida es inferior al permitido, o porque se ha superado el número de iteraciones, por lo cual se considera que se deberán hacer ajustes al diseño de la red, pues la solución no converge, o se debe ampliar el número de iteraciones. Generalmente, la función que se utiliza es sigmoidal (Anzola, 2016).

A continuación se define la notación utilizada para revisar la formulación matemática del algoritmo *backpropagation* (López Sotelo y Caicedo Bravo, 2009); (Brío y Molina, 2006); (Hilera González y Martínez Hernando, 2000) siendo,

x_p	patron o vector de entrada
x_{pi}	entrada i-ésima del vector de entrada x_p
N	dimension del vector de entrada
P	número de ejemplos, vectores de entrada y salida diferentes
L	número de neuronas de la capa oculta: h
M	número de neuronas de la capa de salida, dimensión del vector de salida
w_{ji}^h	peso de interconexión entre la neurona i-ésima de la entrada y la j-ésima de la capa oculta
θ_j^h	término de tendencia de la neurona j-ésima de la capa oculta
$Neta_{pj}^h$	entrada neta de la j-ésima neurona de la capa oculta
i_{pj}	salida de la j-ésima neurona de la capa oculta
f_j^h	función de activación de la j-ésima unidad oculta
w_{kj}^o	peso de interconexión entre la j-ésima neurona de la capa oculta y la k-ésima neurona de la capa de salida
θ_k^o	término de tendencia de la k-ésima neurona de la capa de salida
$Neta_{pk}^o$	entrada neta de la k-ésima neurona de la capa de salida
y_{pk}	salida de la k-ésima unidad de salida
f_k^o	función de activación de la k-ésima unidad de salida
d_{pk}	valor de salida deseado para la k-ésima neurona de la capa de salida
e_p	valor de error para el p-ésimo patrón de aprendizaje
α	tasa o velocidad de aprendizaje

δ_{pk}^{o} término de error para la k-ésima neurona de la capa de salida

δ_{pj}^{h} término de error para la j-ésima neurona de la capa de oculta h

f'^{h}_{j} derivada de la función de activación de la j-ésima neurona de la capa oculta

f'^{o}_{k} derivada de la función de activación de la k-ésima neurona de la capa salida

Los pasos que sigue el algoritmo *backpropagation* son los siguientes:

1. Se inicializan los pesos del MLP.
2. Mientras la condición de parada sea falsa se ejecutan los pasos 3 al 12.
3. Se aplica el vector de entrada a la red $x_p = [x_{p1}, x_{p2}, \ldots, x_{pi}, \ldots x_{pN}]^T$.
4. Se calculan los valores de las entradas netas para la capa oculta:

$$Neta_{pj}^{h} = \sum_{i}^{N} w_{ji}^{h} x_{pi} + \theta_{j}^{h}$$

5. Se calcula la salida de la capa oculta:

$$i_{pj}^{h} = f_{j}^{h}(Neta_{pj}^{h})$$

6. Se calculan los valores netos de entrada para la capa de salida:

$$Neta_{pk}^{o} = \sum_{j=1}^{L} w_{kj}^{o} i_{pj}^{o} + \theta_{k}^{o}$$

7. Su calculan las salidas de la red:

$$y_{pk} = f_{k}^{o}(Net_{pk}^{o})$$

8. Se calculan los términos de error para las unidades de salida:

$$\delta_{pk}^{o} = (d_{pk} - y_{pk}^{o}) f_{k}^{o}(Neta_{pk}^{o})$$

9. Se estiman los términos de error para las unidades ocultas:

$$\delta_{pj}^{h} = f_{j}^{h'}(Neta_{pj}^{h}) \sum_{k=1}^{M} \delta_{pk}^{o} w_{kj}^{o}$$

10. Se actualizan los pesos en la capa de salida:

$$w_{kj}^{o}(t+1) = w_{kj}^{o}(t) + \alpha \delta_{pk}^{o} i_{pj}^{h}$$

11. Se actualicen los pesos de la capa oculta:

$$w_{ji}^{h}(t+1) = w_{ji}^{h}(t) + \alpha\delta_{pj}^{h}x_{pi}$$

12. Se verifica si el error global cumple con la condición de finalizar:

$$E_p = \frac{1}{2}\sum_{p=1}^{P}\sum_{k=1}^{M}(d_{pk} - y_{pk})^2$$

El algoritmo *backpropagation* encuentra su valor mínimo de error local o global mediante la aplicación de pesos descendentes (algoritmo del gradiente descendente). Con el gradiente descendente, cada vez que se realizan cambios a los pesos de la red, se asegura el descenso por la superficie del error hasta encontrar el valle más cercano, lo que puede hacer que el proceso de aprendizaje se detenga en un mínimo local de error. Uno de los problemas que presente este algoritmo de entrenamiento es que busca minimizar la función de error, pudiendo caer en un mínimo local o en algún punto estacionario, con lo cual no se llega a encontrar el mínimo global de la función de error. Dado el inconveniente que presenta el algoritmo de gradiente descendente en la red *backpropagation* de tener el parámetro de aprendizaje α fijo, se revisa el mismo algoritmo, pero con un α variable. Este valor también podría variar en el proceso de aprendizaje con el fin de modificar el tamaño de la variación de los pesos Δw_i y acelerar la convergencia del algoritmo de aprendizaje (Hilera González y Martínez Hernando, 2000).

En la misma vía, el algoritmo de aprendizaje del gradiente conjugado es otro de los más famosos algoritmos de optimización multivariable para el aprendizaje de redes tipo MLP. Este algoritmo usualmente funciona mejor que el *backpropagation*, y puede ser utilizado en las mismas aplicaciones. En el algoritmo del gradiente conjugado se busca minimizar el error sobre una dirección conjugada (el cambio en el gradiente de la función es perpendicular), lo que produce una convergencia más rápida que si la búsqueda se hiciese en la dirección del gradiente descendente (Anzola, 2016).

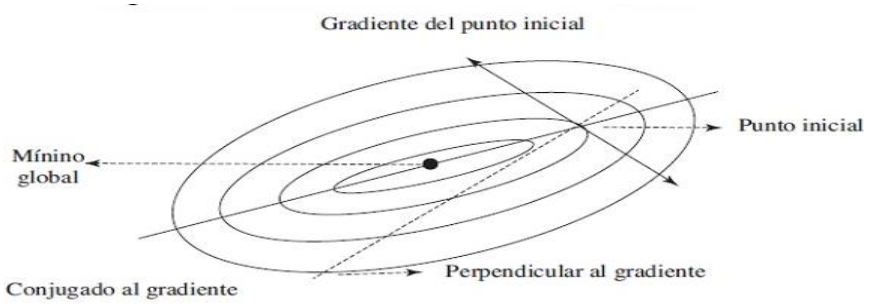


Figura 2.6: Calculo del error – Gradiente Conjugado Fuente:(López Sotelo y Caicedo Bravo, 2009)

Como se muestra en la gráfica, si en vez de calcular una dirección perpendicular a la dirección previa, se calcula una dirección conjugada y se minimiza a lo largo de esta, se llega más rápido al mínimo global de la función (López Sotelo y Caicedo Bravo, 2009).

II.4 Métricas para medir el desempeño de los modelos

II.4.1 Modelos de regresión

En problemas de regresión las métricas más utilizadas según Aguilar et al. (2018) son las siguientes:

Media del error absoluto (MAE – Mean Absolute Error): Es la suma en valor absoluto de la media entre las predicciones y el valor real. Da una idea de la magnitud del error, pero no de su dirección (por encima o por debajo del valor real).

$$MAE = \frac{\sum_{i=1}^{n} SR_p(i) - SR_0(i)}{n}$$

Error cuadrático medio (MSE- Mean Squared Error): Ofrece una idea de la magnitud del error. Si realizamos la raíz cuadrada de MSE (RMSE - Root Mean Squared Error) podemos convertir esta medida a las unidades reales de la variable de salida y nos ofrece una descripción más significativa del resultado.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(SR_p(i) - SR_0(i))^2}$$

Coeficiente de determinación (R^2): Nos da una indicación de la bondad del ajuste de un conjunto de predicciones a los valores reales. Ofrece un valor entre 0 y 1, indicando 0 el peor ajuste y 1 un ajuste perfecto. Por lo tanto, valores de R^2 igual o menores a 0.5 corresponden a malos ajustes

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(SR_0(i) - SR_p(i))^2}{\sum_{i=1}^{n}(SR_0(i) - \frac{1}{n}\sum_{i=1}^{n}(SR_0(i))^2}$$

SR_o y SR_p son los datos reales y las predicciones respectivamente.

II.4.2 Modelos de clasificación

En problemas de clasificación para comprender las distintas métricas es necesario entender previamente de donde se extraen las mismas. Para ello vamos a proceder a explicar lo que es la matriz de confusión, una matriz que nos ayuda a saber cuál ha sido el rendimiento de un algoritmo. Esta matriz simplemente nos muestra en un cuadro los siguientes valores, una vez realizado el entrenamiento y posterior validación con aquellos datos que tenemos para testear usando la validación cruzada: falsos positivos, falsos negativos, verdaderos positivos y verdaderos negativos. Para entender mejor este concepto procedamos a mostrar cómo sería esta matriz para una clasificación binaria.

Tabla 2.2. Matriz de confusión para clasificación binaria Fuente: (Zamorano, 2018)

		Predicción	
		Positivos	Negativos
Observación	Positivos	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Como podemos observar, las etiquetas conocidas corresponden a las filas, y las que el algoritmo predice a las columnas. Para definir correctamente estos valores podemos indicar que:

Verdaderos Positivos (VP): Tenemos que es la cantidad de aquellas observaciones que fueron clasificados como pertenecientes a una clase y acertó.

Falsos Positivos (FP): Esas observaciones que no pertenecían a una clase, pero se llegaron a considerar perteneciente a ellas, son considerados positivos, pero como sabemos la salida también sabemos que se ha equivocado.

Falsos Negativos (FN): Aquellas observaciones que pertenecían a una clase, pero no se consideraron en ella, considerados como negativos.

Verdaderos Negativos (VN): Es la cantidad de aquellas observaciones que no pertenecían a una clase y las clasificó correctamente.

A partir de estos 4 valores previos podemos sacar nuevas métricas de rendimiento que nos ayudarán a extraer más información. Estas métricas sirven para medir clasificaciones binarias. Para clasificación multiclase es necesario sacar el valor promedio para cada una, así para cada métrica tenemos *VPi* que son los Verdaderos Positivos para la clase i, *VNi* = Verdaderos Negativos para la clase i, *FPi*= Falsos Positivos para la clase i y por último *FNi* = Falsos Negativos para la clase i. Las métricas más utilizadas según Zamorano (2018) son:

Exactitud

Nos sirve para sacar el porcentaje de observaciones que ha clasificado correctamente, cuanto más cercano sea su valor a 1 mejor será.

$$Exactitud = \frac{VP + VN}{VP + VN + FP + FN}$$

$$Exactitud\ promedio = \frac{1}{C}\sum_{i=1}^{C}\frac{VP_i + VN_i}{VP_i + FP_i + VN_i + FN_i}$$

Donde C es el número de clases cuando realizamos clasificación multiclase.

Sensibilidad

Mide la habilidad del clasificador de encontrar todas las observaciones positivas, buscamos que este valor sea lo mayor en una escala del 0 al 100. Su fórmula es como sigue:

$$Sensibilidad = \frac{VP}{(VP + FN)}$$

$$Sensibilidad\ promedio = \frac{1}{C}\sum_{i=1}^{C}\frac{VP_i}{VP_i + FN_i}$$

Precisión

La precisión mide la habilidad de un clasificador de no etiquetar como positivo una observación que se debe considerar como negativa. Cuanto mayor sea este valor mejor será. Su fórmula es:

$$Precisión = \frac{VP}{VP + FP}$$

$$Precisión\ promedio = \frac{1}{C}\sum_{i=1}^{C}\frac{VP_i}{VP_i + FP_i}$$

Log Loss

Esta métrica mide la incertidumbre que ha tenido nuestro clasificador con respecto a la etiqueta real que correspondía, es decir, si la clasificación ha variado o desviado mucho o poco con respecto a lo que debía ser. Lo que se pretende es que este valor sea lo más cercano a 0. Lo que hace es cuantificar la precisión del clasificador penalizando las clasificaciones falsas. Minimizando este valor quiere decir que maximizamos la precisión del clasificador.

Para calcular este valor, el clasificador asigna una probabilidad de pertenecer a cada clase para cada ejemplo. Matemáticamente la fórmula para calcular este valor es:

$$\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} log p_{ij}$$

donde N es el número de instancias en nuestro conjunto de datos, M es el número de etiquetas posibles, *y*ij es un valor binario que nos dice si para la instancia i su etiqueta es j, y *p*ij es la probabilidad con el que el clasificador le ha asignado la etiqueta j a la instancia i.

II.5 Ajuste de hiper-parámetros. Optimización bayesiana

Según (Fernández Sánchez, 2019) la optimización Bayesiana, como su nombre indica, es una técnica que se utiliza para optimizar una *función objetivo f*(**x**) (también llamada *latente* o *subyacente*). Su aplicación se hace en escenarios donde las observaciones tienen un coste elevado, pueden tener ruido y no se tiene una expresión para *f*(**x**). Dadas estas tres características, el objetivo es obtener los valores que además de dar la mayor cantidad de información, minimicen la *función objetivo* en el menor número posible de observaciones. La expresión matemática que recoge esto sería la siguiente:

$$x_* = mín_{x \in \chi \subset \mathbb{R}^d} f(x)$$

donde x_* es el valor de entrada, el cual minimiza la *función subyacente f*(**x**), y X es el espacio de características donde se está optimizando.

Nótese que si se elige mal X no se optimizará correctamente *f*(**x**). Con el propósito de efectuar esta minimización, se llevará a cabo un *trade − off* entre explotar soluciones prometedoras y explorar zonas desconocidas del espacio de entradas. Para ello se

utilizará una función de adquisición la cual es de gran importancia dentro de la optimización Bayesiana, puesto que regula el *trade − off* entre explotación y exploración. Existen múltiples métodos utilizados, como pueden ser *probability of improvement* (PI), *expected improvement* (EI), *lower confidence bound* (LCB), *entropy search* (ES), y un portafolios de varias estrategias, esta última suele dar mejores resultados.

La optimización Bayesiana es una técnica que realiza sus predicciones en base a la creencia que tiene sobre el modelo. Este tipo de aproximación a los problemas obtiene mejores resultados que realizar una simple búsqueda en rejilla o una búsqueda aleatoria, puesto que ninguna de estas dos estrategias utiliza el modelo para guiar el proceso de minimización.

Resumiendo, la optimización Bayesiana puede dividirse en 2 etapas:

● Etapa de aprendizaje: Aprender el modelo y los híper-parámetros a partir de la información disponible, muestras evaluadas e información a priori, lo que ayuda a generar un modelo predictivo.

●Etapa de decisión: Decidir la siguiente muestra a partir del modelo predictivo mediante la maximización de la función de adquisición.

Ambas etapas son ejecutadas en bucle durante un número determinado de iteraciones. Al finalizar el bucle, el mejor valor encontrado será devuelto como el óptimo.

CAPÍTULO III

Capítulo III: Análisis de resultados

III.1 Aplicabilidad del aprendizaje automático a la calidad en la industria alimenticia

La calidad es un factor clave para la industria alimentaria moderna porque la alta calidad del producto es la base del éxito en el mercado altamente competitivo de hoy. En la industria alimentaria, la evaluación de la calidad todavía depende en gran medida de la inspección manual, que es tediosa, laboriosa y costosa, y se ve fácilmente influenciada por factores fisiológicos, lo que induce resultados de evaluación subjetivos e inconsistentes. Para satisfacer la mayor conciencia, sofisticación y mayor expectativa de los consumidores, es necesario mejorar la evaluación de calidad de los productos alimenticios. Si la evaluación de la calidad se logra automáticamente, la velocidad y la eficiencia de la producción se pueden mejorar además de la mayor precisión de la evaluación, con una reducción correspondiente en los costos de producción (C.-J. Du y Sun, 2006).

La utilización de modelos de aprendizaje automático en la industria alimenticia es cada vez más frecuente y se está convirtiendo poco a poco en una prioridad. En un estudio realizado por Gazeli et al. (2020) utilizaron la espectroscopia de descomposición inducida por láser (LIBS) asistida por algoritmos de aprendizaje automático para recuperar la información contenida en los espectros LIBS para proporcionar una metodología simple, confiable y ultrarrápida en la clasificación de aceites de oliva en términos del grado de acidez y origen geográfico. La combinación de la técnica LIBS con los enfoques de análisis estadístico de aprendizaje automático constituyeron una herramienta muy poderosa para el control de calidad rápido y remoto del aceite de oliva.

En otra investigación desarrollada por Gupta (2018) sobre la calidad del vino se utilizaron técnicas de aprendizaje automático como la regresión lineal para determinar la dependencia de la variable objetivo en variables independientes y las redes neuronales y Support Vector Machine para predecir los valores de la variable dependiente.

Gunaratne et al. (2019) utilizo varios modelos de aprendizaje automático para predecir la calidad del chocolate basándose en datos fisicoquímicos y propiedades sensoriales.

Vidyarthi et al. (2019) en un estudio sobre el tamaño y predicción de los granos de almendra implementaron varios modelos populares de aprendizaje automático (ML) para construir un modelo de conjunto apilado (SEM), prediciendo la masa de los

núcleos de almendras individuales en función de las características derivadas de los píxeles de los núcleos individuales en la imagen. La precisión de predicción y la robustez del procesamiento de imágenes y SEM se analizaron utilizando la cuantificación de la incertidumbre. El error promedio al estimar la longitud promedio de 1000 almendras fue de 3.12%. Del mismo modo, los errores medios asociados con la predicción de la masa de 1000 granos fueron del 0,63%. El algoritmo desarrollado en imágenes de almendras en este estudio se puede utilizar para facilitar un rendimiento rápido de almendras y evaluaciones de calidad. De esta manera se fundamenta la aplicabilidad de estos algoritmos para tareas de control de la calidad, destacándose que de acuerdo a la literatura consultada se prevé un aumento mayor de estas herramientas en el futuro.

En la Figura 1.2 se mostraron las 5 etapas en la realización de un proyecto Six Sigma. El autor considera que la implementación de estas técnicas puede tener un impacto positivo en el control de la calidad de la Empresa objeto de estudio. Con la construcción de un modelo de aprendizaje automático se pueden explorar en un ambiente controlado como varía la calidad de un producto vía la simulación de las variables de entrada al proceso permitiendo identificar aquellas de mayor influencia. Esto permitiría buscar mejoras en los procesos y automatizar el control dentro de la entidad integrándose con el resto de las herramientas de control estadístico de la calidad expuestas en el capítulo 1.

III.2 Herramientas computacionales existentes

Existen varios lenguajes de programación que son útiles para su uso en el aprendizaje automático, entre ellos tenemos Python, R, Matlab y Julia. Todo ellos ofrecen un abanico numeroso de posibilidades y de bibliotecas para la aplicación de métodos o algoritmos en inteligencia artificial (Ruiz, 2018).

Una de las formas más fáciles de comenzar con el aprendizaje automático y el análisis de datos es a través de la biblioteca de Python Scikit-learn. Python es un gran lenguaje para ideas prototipo que eventualmente se convierten en implementaciones estándar de la industria. Algunas de las bibliotecas Python más duraderas y exitosas como NumPy, SciPy, Pandas y Matplotlib forman la columna vertebral de Scikit-learn. Las herramientas son simples, por lo que son fáciles de usar. Sin embargo, Scikit-learn se siente como lenguaje ensamblador porque la biblioteca tiene un nivel relativamente bajo lo que permite un control adecuado de los parámetros en los algoritmos.

Otras bibliotecas de nivel superior como TensorFlow, Theano, o Caffe ofrecen una configuración más robusta. Hadoop o Apache Spark son algunos marcos estándar de la industria de más alto nivel para tratar Big Data donde el énfasis es el paralelismo y la informática distribuida. Comúnmente, el Apache la biblioteca de nivel inferior de Mahout se utiliza para interactuar con estas arquitecturas paralelas, proporcionando un Interfaz completa de Java. Además de TensorFlow, algunas de las bibliotecas de aprendizaje automático más comunes incluyen Theano, Caffe, Torch y Computational Graph Toolkit (Shukla, 2017).

Por lo anterior se propone en un nivel inicial el uso de Scikit-learn fundamentado en que no encarece los costos de la empresa por el concepto de adquirir licencia debido a que la biblioteca es libre. En caso de que el problema a tratar sea de mayor complejidad las otras bibliotecas referidas anteriormente son buenas opciones, pero presentan una curva de aprendizaje mayor dificultando su uso por no expertos en la industria.

III.3 Descripción del análisis de datos

Durante el transcurso de la investigación resultó imposible la adquisición de los datos en la empresa GydeMa debido a las medidas de restricción impuestas necesariamente por la pandemia del coronavirus. Por ello se analizaron los diversos algoritmos disponibles en la biblioteca Scikit-learn para el enfrentamiento de problemas de regresión y clasificación utilizando como base los datos el precio de las casas de Boston para la regresión y un estudio de la calidad de vinos para la clasificación.

III.3.1 Descripción de los datos de regresión

El dataset de los precios de casas de Boston fue creado por Harrison, D. y Rubinfeld, D.L. constituyendo un estándar para el análisis de problemas de regresión. Consiste en 506 casos en el que se relacionan 13 atributos (variables independientes) para predecir el precio de casas (variable dependiente). Los atributos son:

Información de atributo (en orden):

- CRIM: Tasa de criminalidad per cápita por ciudad
- ZN: Proporción de terreno residencial dividido en zonas para lotes de más de 25,000 pies cuadrados.
- INDUS: Proporción de acres de negocios no minoristas por ciudad
- CHAS: Variable ficticia Charles River (= 1 si el tramo limita con el río; 0 en caso contrario)

- NOX: Concentración de óxidos nítricos (partes por 10 millones)
- RM: Promedio de habitaciones por vivienda
- AGE: Proporción de edad de las unidades ocupadas por el propietario construidas antes de 1940
- DIS: Distancias ponderadas a cinco centros de empleo de Boston
- RAD: Índice de accesibilidad a carreteras radiales
- TAX: Tasa de impuesto a la propiedad de valor total por $ 10,000
- PTRATIO: Relación alumno-profesor por localidad
- B 1000 (Bk - 0.63) ^ 2 donde Bk es la proporción de negros por ciudad
- LSTAT: % menor estado de la población
- MEDV: Valor medio de viviendas ocupadas por sus propietarios en $ 1000

En el juego de datos no existen valores perdidos y una copia de estos puede ser encontrada en la dirección electrónica:
https://archive.ics.uci.edu/ml/machine-learning-databases/housing/

III.3.2 Descripción de los datos de clasificación

Los datos son el resultado de un análisis químico de vinos cultivados en la misma región en Italia por tres cultivadores diferentes. Hay trece diferentes medidas tomadas para diferentes componentes encontrados en los tres tipos de vino los que son:

- Alcohol
- Ácido málico
- Ceniza
- Alcalinidad de cenizas
- Magnesio
- Fenoles totales
- Flavonoides
- Fenoles no flavonoides
- Proantocianinas
- Intensidad de color
- Hue
- OD280 / OD315 de vinos diluidos
- Proline

El objetivo del modelo de clasificación es predecir, dado un nuevo vector a cuál de las tres clases de vino pertenece. El juego de datos cuenta con 178 instancias, donde no

existen valores perdidos. Una copia de estos puede ser encontrada en la dirección electrónica: https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data

III.4. Desempeño de algoritmos de regresión

Como estrategia de regresión inicialmente se comprobó el desempeño de cada algoritmo con los parámetros de defecto que presenta la librería. Luego, como no necesariamente la inclusión de más variables en el modelo condiciona un mejor ajuste fueron seleccionadas mediante la optimización bayesiana aquellas mejores variables, así como también los hiper-parámetros que ofrecían un mejor desempeño. Sin embargo, como la forma en las que se seleccionan los datos para construir el modelo puede influir en el rendimiento de este se realizó una validación cruzada mediante 10 permutaciones aleatorias y como conjunto de prueba un 25% del total de datos iniciales.

Inicialmente se probaron los modelos de regresión lineal y polinomial de grado 2 (Anexo 1). Mayores grados no fueron analizados en los modelos polinomiales pues en experimentos previos se comprobó que el rendimiento de estos decrecía con el aumento del grado, además que de esta forma son más susceptibles al sobreajuste. En la Figura 3.1 se observan los resultados de este proceso.

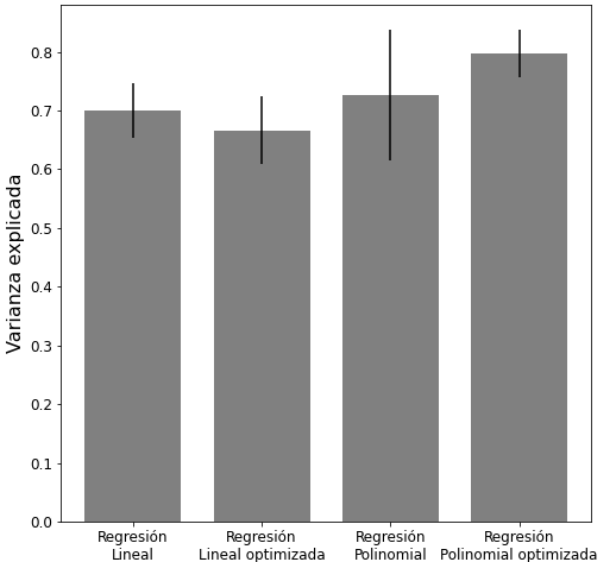


Figura 3.1: Comparación de los modelos de regresión lineal y polinomial

Se comprobó que en el caso de la regresión lineal con 12 variables se lograba una explicación de la varianza del 70% mientras que en el caso del modelo polinomial con 5 variables la varianza explicada es del 79% aproximadamente, ligeramente superior a los parámetros de defecto. De estos resultados es notable el alto ajuste de la regresión

polinomial con hiper-parámetros optimizados llegando en promedio casi al 80% y con una baja dispersión.

En la Figura 3.2 aparece representado el desempeño de diferentes kernel en las máquinas de soporte vectorial. En esta se observa que los mejores resultados se corresponden al kernel lineal con casi un 70% de la varianza explicada y que con un kernel polinomial no necesariamente un aumento del grado mejorará el desempeño. Sin embargo, esta situación fue mejorada (Anexo 2) mediante el establecimiento de mejores hiper-parámetros, que para las máquinas de soporte vectorial son la cantidad de variables explicativas, el parámetro de regularización C y épsilon. En la Figura 3.3 aparece representado este proceso. Como se puede observar el desempeño del kernel lineal se mantuvo aproximadamente constante mientras que el del resto aumentó hasta aproximadamente un 80.68 % de la varianza explicada, con una desviación estándar del 4.3 %, cuando se utilizan en el entrenamiento las mejores nueve variables, un parámetro C=10 y épsilon=1. Esta mejora es significativa pues en el caso de la máquina de soporte vectorial con kernel rbf el aumento en desempeño fue casi del 15%.

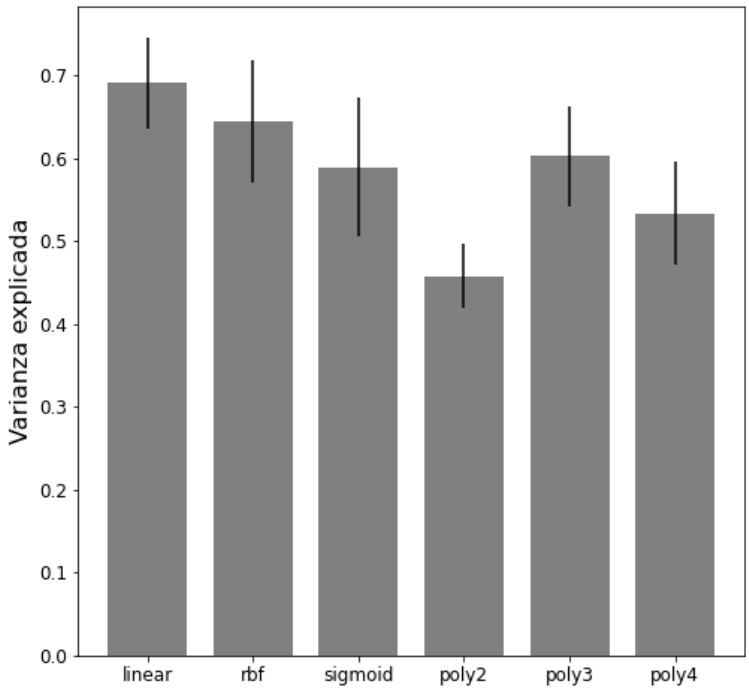


Figura 3.2: Desempeño de diferentes kernels en las máquinas de soporte vectorial para la regresión

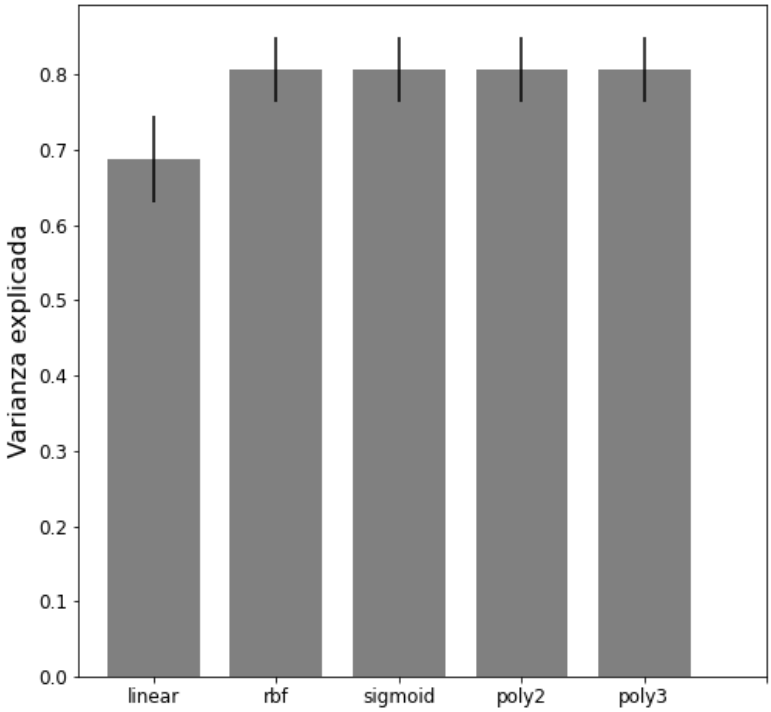


Figura 3.3: Desempeño de diferentes kernels optimizados en las máquinas de soporte vectorial para la regresión

Posteriormente se llevó a cabo el mismo proceso con los algoritmos árbol de decisión, bosques aleatorios y vecinos más cercanos. En la Figura 3.4 (Anexo 3) se muestran los resultados obtenidos. Se puede comprobar que hubo una mejora tanto en los algoritmos de árbol de decisión y vecinos más cercanos, pero en el caso de los bosques aleatorios no fue así. Esto refleja que hubo un mejor desempeño del algoritmo con los parámetros iniciales que no mediante la búsqueda con la optimización bayesiana. Hay que destacar que esto sucede debido a que la optimización bayesiana no garantiza un óptimo global, sino que explora una región multidimensional mediante un proceso aleatorio donde cada nuevo punto de búsqueda es seleccionado mediante una función de adquisición. Para el lector interesado en este tema se le sugiere que consulte el trabajo de Archetti y Candelieri (2019) con gran relación con los algoritmos de aprendizaje automático tratados en esta investigación.

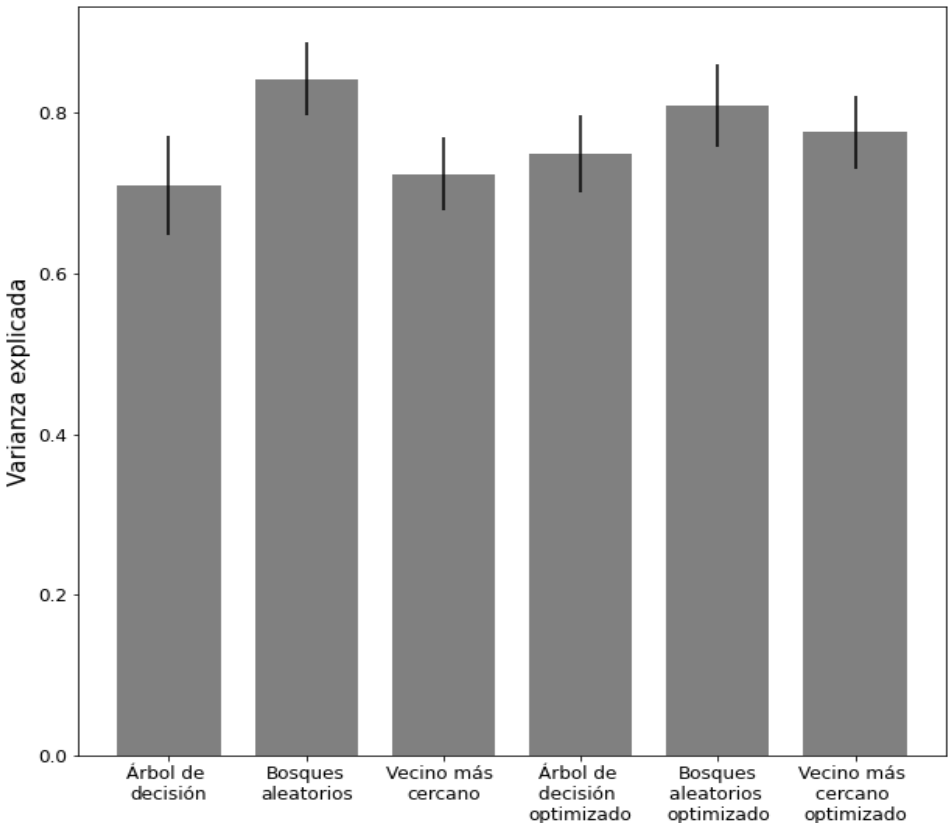


Figura 3.4: Desempeño de otros algoritmos de regresión

Finalmente se construyó una red neuronal de tipo perceptrón multicapa con 1 capa oculta con 100 neuronas y una capa de salida para la predicción. Fue necesario establecer un máximo de iteraciones igual a 500, pues el implantado por defecto no era suficiente para converger en el ajuste de la red. Bajo estas condiciones se obtuvo una varianza explicada del 84,45 % con una desviación estándar de 3,12% (Anexo 4). Mejor desempeño aún puede obtenerse si se modifica la arquitectura de la red añadiéndose otra capa quedando la primera capa oculta con 122 neuronas y la segunda con 73 para una varianza explicada del 88.4% con una desviación estándar de 2,29%. Esto fue determinado optimizando mediante una red neuronal con dos capas ocultas cuyos hiperparámetros fueron la cantidad de neuronas en cada capa (Anexo 5).

El la Figura 3.5 se muestra el resultado de esta optimización bayesiana. Se puede ver que no se presenta una convergencia lo que reafirma que, aunque los modelos de redes neuronales son muy poderosos, tienen como problema la dificultad del ajuste de sus parámetros. Además, este proceso puede tomar un tiempo excesivo. El código presentado fue corrido en una computadora personal Lenovo, con procesador Intel Core i3 de 2.3 GHz y 4 GB de RAM tomando un tiempo de 12 minutos con 18 segundos.

Aunque esto no parece significativo hay que considerar que en el caso analizado fueron optimizados solo la cantidad de neuronas en la arquitectura preestablecida, cuando en la práctica pueden considerarse la cantidad de capas, la cantidad de neuronas por capas y diferentes arquitecturas lo que en muchos contextos resulta computacionalmente imposible.

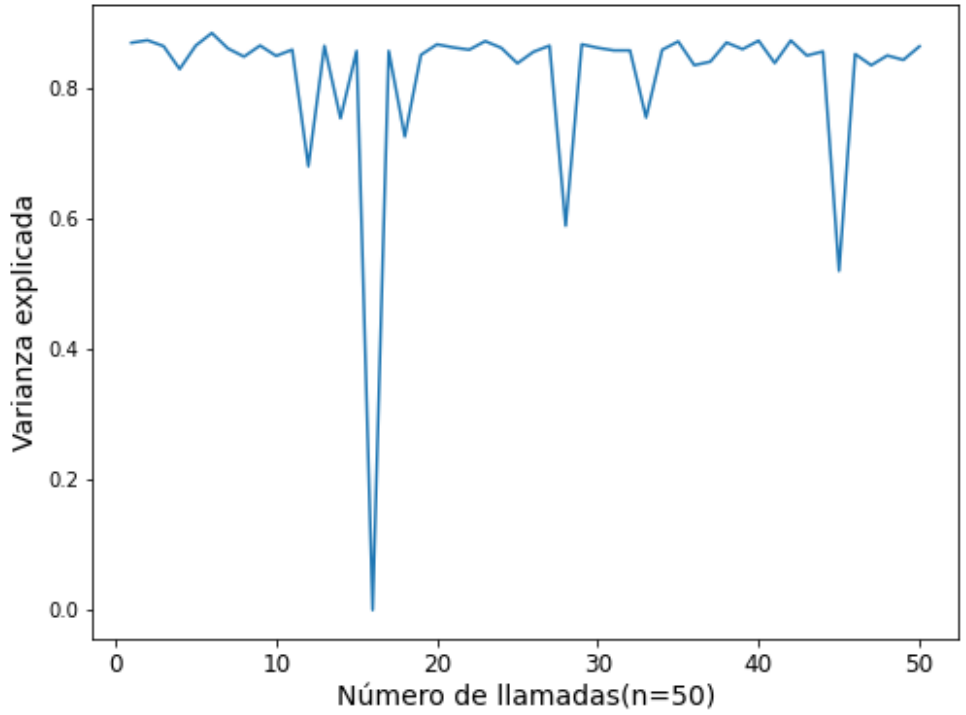


Figura 3.5: Convergencia de la optimización bayesiana para la red neuronal de regresión

En la Tabla 3.1 aparecen resumido el desempeño de los mejores algoritmos tratados hasta el momento mientras que en la Tabla 3.2 se presenta el análisis ANOVA para la comparación de estos. Puesto que el valor-P de la prueba-F es menor que 0,05, existe una diferencia estadísticamente significativa entre las medias de las 4 variables con un nivel del 95,0% de confianza.

Tabla 3.1: Resumen estadístico de la varianza explicada para los mejores algoritmos de regresión

Algoritmo	**Hiperparámetros**	**Promedio**	**Desviación Estándar**	**Coeficiente de Variación**
Regresión polinomial	Grado=2 Variables explicativas =5	0,797365	0,0418522	5,24881%
Máquina de soporte vectorial	kernel=rbf Variables explicativas=9	0,824175	0,0440283	5,3421%

	C=10 Épsilon=1			
Bosques aleatorios	Variables explicativas=13 Número de estimadores=100	0,843675	0,050697	6,00907%
Red neuronal	Número de capas ocultas=2 Cantidad de neuronas=(122,73)	0,883995	0,0241276	2,72938%

Tabla 3.2: Tabla ANOVA para los algoritmos de regresión con mejor desempeño

Fuente	*Suma de Cuadrados*	*Gl*	*Cuadrado Medio*	*Razón-F*	*Valor-P*
Entre grupos	0,039881	3	0,0132937	7,77	0,0004
Intra grupos	0,0615818	36	0,00171061		
Total (Corr.)	0,101463	39			

En la Tabla 3.3 se muestran los resultados de la prueba de múltiples rangos de acuerdo al criterio de Duncan destacándose la formación de tres grupos homogéneos. De esta manera se puede comprobar que con un nivel de confianza del 95 % existe una diferencia significativa en el desempeño de la red neuronal respecto a los bosques aleatorios. Por ello este sería el algoritmo propuesto para realizar predicciones en esta situación.

Tabla 3.3: Prueba de múltiples rangos para algoritmos de regresión

	Casos	*Media*	*Grupos Homogéneos*
Regresión Polinomial	10	0,797365	X
Máquina de soporte vectorial	10	0,824175	XX
Bosques aleatorios	10	0,843675	X
Red Neuronal	10	0,883995	X

Contraste	*Sig.*	*Diferencia*	*+/- Límites*
Bosques aleatorios - Máquina de soporte vectorial		0,0194993	0,0375128
Bosques aleatorios - Red Neuronal	*	-0,0403203	0,0375128
Bosques aleatorios - Regresión Polinomial	*	0,0463094	0,0375128
Máquina de soporte vectorial - Red Neuronal	*	-0,0598196	0,0375128
Máquina de soporte vectorial - Regresión Polinomial		0,0268101	0,0375128
Red Neuronal - Regresión Polinomial	*	0,0866297	0,0375128

III.5 Desempeño de los algoritmos de clasificación

El primer algoritmo probado para la predicción de las clases fueron las máquinas de soporte vectorial, considerando los parámetros de defecto para los diferentes kernels disponibles en la librería. En la Figura 3.6 se muestra el desempeño de estos destacándose una alta precisión en todos. Aunque habría que realizar un análisis ANOVA y prueba de rangos múltiples para determinar si existen diferencias significativas entre los kernels por inspección visual, el autor considera que el rbf es el más adecuado en este contexto fundamentado no solo en el alto desempeño sino también en la poca variabilidad dada por la pequeña desviación estándar de la precisión del modelo. En el Anexo 6 se muestra el código utilizado.

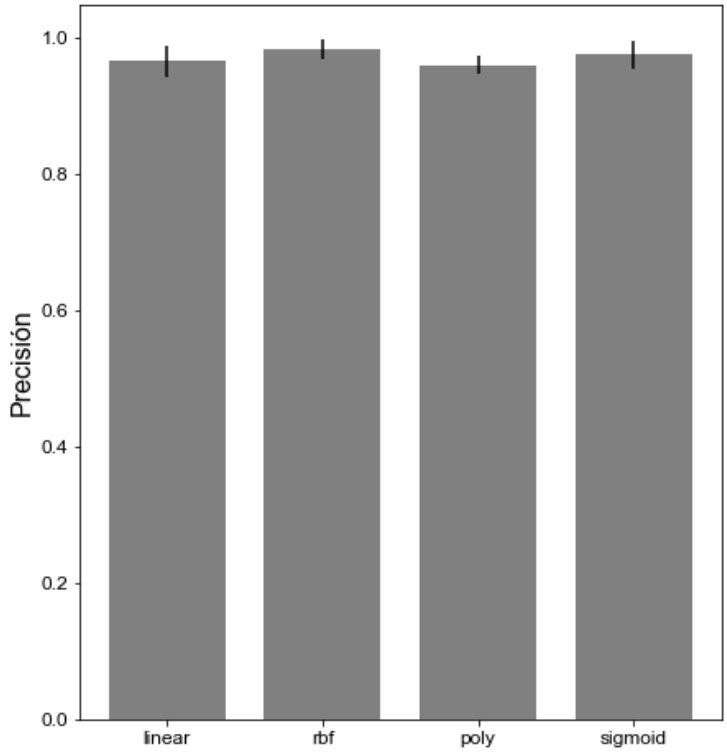


Figura 3.6: Desempeño de diferentes kernels en las máquinas de soporte vectorial para la clasificación

Posteriormente fueron analizados los árboles de decisión, los bosques aleatorios y las redes bayesianas. Los bosques aleatorios y las redes bayesianas presentaron un desempeño significativamente mejor que el árbol de decisión por lo que se optimizó este último algoritmo. En el Anexo 7 se presenta el código para llevar a cabo este proceso mientras que en la Figuras 3.7 y 3.8 se observan los resultados. Según la optimización bayesiana con 3 variables explicativas, una máxima profundidad del árbol

igual a ocho y un valor igual a la unidad de muestras requeridas para dividir un nodo interno se alcanza una mejoría en el desempeño del algoritmo. Sin embargo, en la Figura 3.7 se observa que para 100 llamadas no se alcanza una convergencia por lo que en teoría este pudiera ser mejorado aún más, pero a expensa de mayor tiempo de búsqueda. De acuerdo a los resultados aquí presentados el árbol de decisión no es comparable en desempeño a los bosques aleatorios o redes bayesianas por lo que se proponen estos últimos.

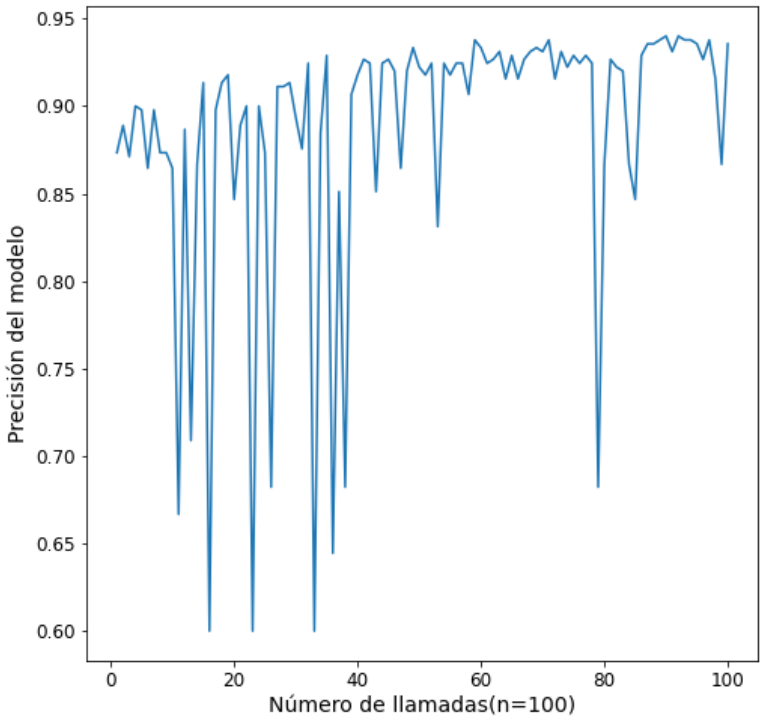


Figura 3.7: Optimización bayesiana de los hiper-parámetros del árbol de decisión

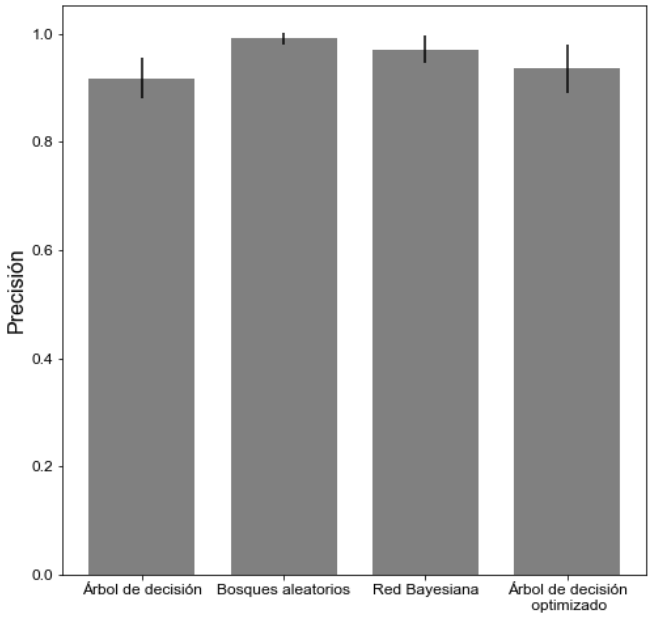


Figura 3.8: Desempeño de los árboles de decisión, bosques aleatorios y red bayesiana

En el caso de los vecinos más cercanos se obtuvo una precisión considerablemente elevada considerando cinco vecinos más cercanos (parámetro de defecto de la librería) pero si se consideran 12 variables explicativas y un vecino más cercano este puede ser mejorado. A pesar de la menor variabilidad por inspección visual de la Figura 3.9 no parece que la diferencia sea significativa, aunque sí se logra disminuir la variabilidad de los resultados. En el Anexo 8 se muestra el código utilizado.

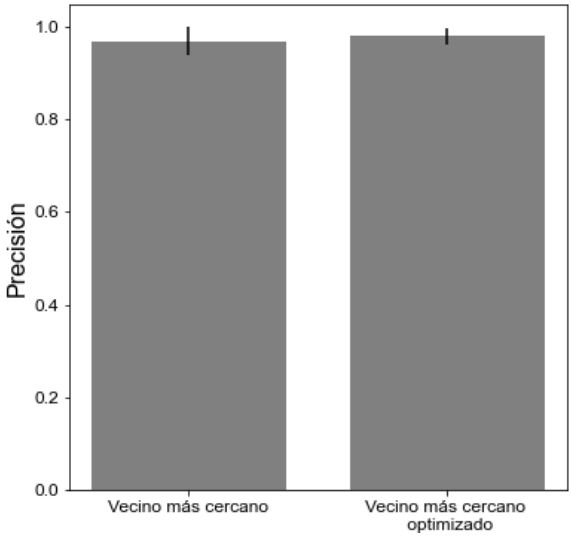


Figura 3.9: Desempeño del algoritmo de vecinos más cercanos

Finalmente, un resultado similar al algoritmo de vecinos más cercanos fue obtenido en las redes neuronales (Anexo 9). En la Figura 3.10 se observan estos resultados mientras que en la Figura 11 el proceso de búsqueda mediante la optimización bayesiana. Se puede ver que utilizando los parámetros de defecto del algoritmo se obtiene un buen resultado, pero si se cambia la arquitectura de la red a un perceptrón multicapa de dos capas ocultas con 45 y 150 neuronas por capa respectivamente la variabilidad de la precisión disminuye a la vez que se mejora ligeramente el rendimiento. Lo más interesante de este proceso se manifiesta en la Figura 3.11 donde se observa cierta convergencia del proceso de optimización a los hiper-parámetros señalados.

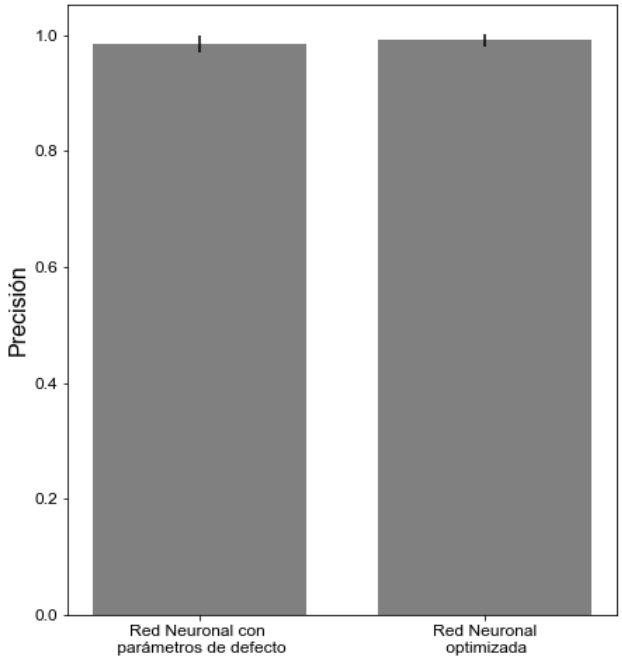


Figura 3.10: Desempeño de las redes neuronales

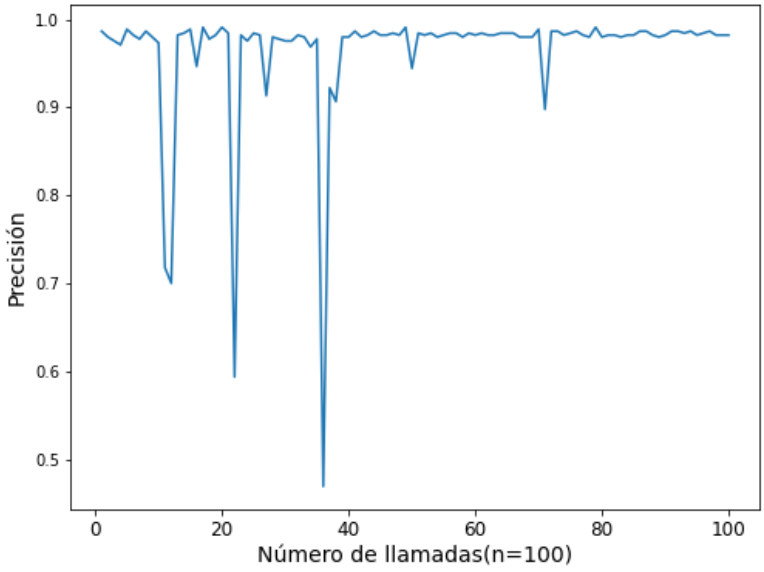


Figura 3.11: Optimización bayesiana de la red neuronal de clasificación

Luego al igual que en los algoritmos de clasificación se realizó un análisis ANOVA para determinar si existían diferencias significativas entre los algoritmos analizados. En la Tabla 4 se muestran los estadígrafos más importantes para los mejores algoritmos de clasificación mientras que en la Tabla 3.5 el análisis ANOVA. Luego, puesto que el valor-P de la razón-F es mayor que 0,05, no existe una diferencia estadísticamente significativa entre las medias de las 5 variables con un nivel del 95,0% de confianza.

Tabla 3.4: Resumen estadístico de la varianza explicada para los mejores algoritmos de clasificación

Algoritmo	**Hiperparámetros**	**Promedio**	**Desviación Estándar**	**Coeficiente de Variación**
Máquina de soporte vectorial	kernel=rbf Variables explicativas=13 C=1	0,984444	0,0149989	1,52359%
Bosques aleatorios	Variables explicativas=13 Número de estimadores=100	0,991111	0,0114755	1,15784%
Red Bayesiana	Parámetros de defecto	0,971111	0,0257667	2,65332%
Vecino más cercano optimizado	Variables explicativas=12 Número de vecinos=1	0,98	0,0194577	1,98548%
Red neuronal optimizada	Número de capas ocultas=2 Cantidad de neuronas=(45,150)	0,991111	0,0114755	1,15784%

Tabla 3.5: Tabla ANOVA para los algoritmos de mejor desempeño

Fuente	*Suma de Cuadrados*	*Gl*	*Cuadrado Medio*	*Razón-F*	*Valor-P*
Entre grupos	0,00282469	4	0,000706173	2,31	0,0727
Intra grupos	0,0137778	45	0,000306173		
Total (Corr.)	0,0166025	49			

En esta situación cualquiera de los algoritmos puede ser utilizado para la tarea de clasificación. Sin embargo, es preciso señalar que mientras los datos se mantengan en un rango similar no debe disminuir la capacidad de generalización de los modelos presentados, pero ante cambios en el proceso como por ejemplo de materias primas es conveniente que los algoritmos sean ajustados nuevamente, por lo que considerar en la elección otros factores como la facilidad de mantenimiento es adecuado para su uso en la industria.

III.6 Consideraciones finales

Hasta el momento se han analizado diferentes algoritmos para problemas de regresión o clasificación sin embargo la cantidad de estos no permite la inclusión de todos en este documento. En los casos analizados de regresión solo se consideró una variable dependiente, en la situación de tener que analizar múltiples variables de salida los códigos deben ser modificados. La misma librería Scikit-learn presenta facilidades para tratar con sistemas de esta naturaleza.

Al analizar el desempeño de cada algoritmo, se utilizó el análisis ANOVA para ver si existían diferencias significativas entre estos y de ser así la prueba de rangos múltiples para la selección del mejor modelo. En caso de que no se pueda determinar un mejor modelo como en el caso de la clasificación dado que no existen diferencias significativas entre estos, resulta interesante usar los n modelos y la predicción de la variable de salida sería la media aritmética de cada uno de ellos como en el caso de los modelos de ensamble. Esto favorecería la robustez de la predicción considerando más de un predictor en la toma de decisión.

CONCLUSIONES

Conclusiones:

1. La integración de los modelos de aprendizaje automático, con otras herramientas de control de la calidad, resulta apropiado para la automatización y optimización de estos procesos favoreciendo la búsqueda de posibles mejoras y una gestión más eficiente.
2. La metodología propuesta constituye un punto de apoyo para la implementación de los algoritmos citados en el contexto de la empresa GydeMa de la provincia de Cienfuegos. Esta permite un ahorro de tiempo por parte del personal de la industria, no capacitado en el uso de estas herramientas.
3. En los datos analizados se comprobó que, entre los modelos analizados, el de mejor desempeño fueron las redes neuronales, aunque en la clasificación no tuvo diferencias significativas con otros algoritmos. Por ello en esta situación se propone la utilización de todos los modelos como analogía a los métodos de ensamblaje.

RECOMENDACIONES

Recomendaciones:

- Aplicar los algoritmos con ayuda de los códigos presentados en esta investigación a los datos de la empresa GydeMa.
- Extender estudios de esta naturaleza a otros procesos productivos del territorio.

REFERENCIAS BIBLIOGRÁFICAS

Referencias Bibliográficas

Aguilar, R., Torres, J., & Martín, C. (2018). Aprendizaje Automático en la Identificación de Sistemas. Un Caso de Estudio en la Predicción de la Generación Eléctrica de un Parque Eólico. *Revista Iberoamericana de Automática e Informática., 16*(1), 114-127.

Allen, T. T. (2019). *Introduction to Engineering Statistics and Lean Six Sigma* (3 ed.): Springer.

Anzola, N. S. (2016). Máquinas de soporte vectorial y redes neuronales artificiales en la predicción del movimiento USD/COP spot intradiario. *ODEON 0*, 113-172.

Archetti, F., & Candelieri, A. (2019). *Bayesian Optimization and Data Science*: Springer.

Besterfield, D. H. (2009). *Control de calidad* (8 ed.). México: Pearson Educación.

Bielza, C., & Larrañaga, P. (2014). Discrete Bayesian network classifiers: a survey. *ACM Computing Surveys (CSUR), 47*(1), 1-43.

Breiman, L. (2001). Random forests. *Machine learning, 45*(1), 5-32.

Brío, B., & Molina, A. (2006). Redes neuronales y sistemas borrosos. In *Textos Universitarios*: Ra-Ma.

Brownlee, J. (2016). Master Machine Learning Algorithms. Discover How They Work and Implement Them From Scratch.

Cambra, A. (2017). *Mejora en la calidad del huevo de ponedora en el proceso de producción en la UEB Yaguaramas de la Empresa Avícola Cienfuegos.* (Tesis de Mastría), Universidad de Cienfuegos.

Carranza, J. P., Salomón, M. J., Piumetto, M. A., Monzani, F., Montenegro, M. G., & Córdoba, M. A. (2018). Random forest como técnica de valuación masiva del valor del suelo urbano: Una aplicación para la ciudad de Río Cuarto. COBRAC.

Cuba, Oficina Nacional de Normalización.(2015) NC-ISO 9001 *Sistemas de gestión de la calidad-Requisitos*. O N N .

Choi, H., Venkiteela, G., Gregori, A., & Najm, H. (2020). Advanced Quality Control Models for Concrete Admixtures. *Journal of Materials in Civil Engineering, 32*(2), 04019349.

Dangeti, P. (2017). *Statistics for Machine Learning. Build supervised, unsupervised, and reinforcement learning models using both Python and R*: Packt Publishing.

Du, C.-J., & Sun, D.-W. (2006). Learning techniques used in computer vision for food quality evaluation: a review. *Journal of Food Engineering, 72*(1), 39-55.

Du, K.-L., & Swamy, M. N. S. (2014). *Neural Networks and Statistical Learning*: Springer.

Evans, J. R., & Lindsay, W. M. (2008). *Administración y control de la calidad* (S. R. Cervantes Ed. 7 ed.). Cengage Learning.

Fernández Sánchez, D. (2019). *Creación de una herramienta de optimización Bayesiana en Python.*

Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The elements of statistical learning* (Vol. 1): Springer series in statistics New York.

Gala, Y. (2013). *Algoritmos SVM para problemas sobre big data.*

Gazeli, O., Bellou, E., Stefas, D., & Couris, S. (2020). Laser-based classification of olive oils assisted by machine learning. *Food chemistry, 302*, 125329.

Géron, A. (2019). *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow. Concepts, Tools, and Techniques to Build Intelligent Systems* (2 ed.): O'Reilly

Gunaratne, T. M., Gonzalez Viejo, C., Gunaratne, N. M., Torrico, D. D., Dunshea, F. R., & Fuentes, S. (2019). Chocolate Quality Assessment Based on Chemical Fingerprinting Using Near Infra-red and Machine Learning Modeling. *Foods, 8*(10), 426.

Gupta, Y. (2018). Selection of important features and predicting wine quality using machine learning techniques. *Procedia Computer Science, 125*, 305-312.

Gutiérrez, H. (2010). *Calidad total y productividad* (3 ed.) McGraw-Hill

Gutiérrez, H., & de la Vara, R. (2013). *Control estadístico de calidad y seis sigma* (3 ed.). México: McGraw-Hill Educación.

Hilera González, J. R., & Martínez Hernando, V. J. (2000). *Redes neuronales artificiales: fundamentos, modelos y aplicaciones*.

Izaguirre, L. D. (2017). *Diseño del sistema integrado de gestión calidad-medio ambiente y seguridad y salud en el trabajo en la Empresa Oleohidráulica Cienfuegos.* (Tesis de Maestría), Universidad de Cienfuegos.

Krishnamoorthi, K. S., Krishnamoorthi, V. R., & Pennathur, A. (2019). *A First Course in Quality Engineering. Integrating Statistical and Management Methods of Quality* (3 ed.). United States of America: CRC Press.

Labajo, A. L., & Labajo, J. L. (2020). Una red neuronal como herramienta de predicción de variables climáticas: Aplicación a la temperatura mínima media mensual en Castilla y León. *Acta de las Jornadas Científicas de la Asociación Meteorológica Española*,(31).

Lantz, B. (2019). *Machine Learning with R* (3 ed.): Packt Publishing.

López Sotelo, J. A., & Caicedo Bravo, E. F. (2009). Una aproximación práctica a las redes neuronales artificiales.

Marsland, S. (2015). *Machine Learning: An Algorithmic Perspective* (2 ed.). Chapman & Hall/CRC.

Mohammadi, P., & Wang, Z. J. (2016). *Machine learning for quality prediction in abrasion-resistant material manufacturing process.* Paper presented at the 2016 IEEE. Canadian Conference on Electrical and Computer Engineering (CCECE).

Montgomery, D. C. (2013). *Introduction to statistical quality control* (7 ed.). John Wiley & Sons (New York).

Mukherjee, S. P. (2019). *Quality. Domains and Dimensions*. Springer.

Peres, R. S., Barata, J., Leitao, P., & Garcia, G. (2019). Multistage quality control using machine learning in the automotive industry. *IEEE Access, 7*, 79908-79916.

Ríos, J., Ulla, G., & Borello Gianni, A. (2019). *Aplicación de regresión con vectores de soporte en un sistema recomendador de actividades sociales.* Paper presented at the (Ponencia). XXV Congreso Argentino de Ciencias de la Computación (CACIC), Universidad Nacional de Río Cuarto, Córdoba.

Sampedro, J. D. D., & Garcıa, J. D. (2012). *Estudio y aplicación de técnicas de aprendizaje automático orientadas al ámbito médico: estimación y explicación de predicciones individuales.* (Tesis de Maestría), *EPS-UAM*.

Sarkar, D., Bali, R., & Sharma, T. (2018). *Practical Machine Learning with Python*. Apress.

Shi, B., & Iyengar, S. S. (2020). *Mathematical Theories of Machine Learning - Theory and Applications*. Springer.

Shukla, N. (2017). *Machine Learning with TensorFlow*. Manning Publications.

Stoyanov, S., & Bailey, C. (2017). *Machine learning for additive manufacturing of electronics.* International Spring Seminar on Electronics Technology (ISSE).

Theodoridis, S. (2015). *Machine Learning. A Bayesian and Optimization Perspective*. Elsevier.

Vega, L. A. d., Builes, M. P. Á., Torres, C. A. B., Soler, C. E. G., & Cortés, A. V. (2011). *Administración por calidad*. Alfaomega.

Vidyarthi, S. K., Tiwari, R., & Singh, S. K. (2019). Size and mass prediction of almond kernels using machine learning image processing. *bioRxiv*.

Yaseen, Z. M., Deo, R. C., Hilal, A., Abd, A. M., Bueno, L. C., Salcedo-Sanz, S., & Nehdi, M. L. (2018). Predicting compressive strength of lightweight foamed concrete using extreme learning machine model. *Advances in Engineering Software, 115*, 112-125.

Zamorano, J. (2018). *Comparativa y análisis de algoritmos de aprendizaje automático para la predicción del tipo predominante de cubierta arbórea.* (Tesis de Maestría), Universidad Compultense de Madrid.

Zhan, W., & Ding, X. (2015). *Lean Six Sigma and Statistical Tools for Engineers and Engineering Managers*. Momentum Press.

ANEXOS

Anexos

Anexo [1] Prueba de los modelos de regresión lineal y polinomial de grado 2.

```
import sklearn
import numpy as np
import matplotlib.pyplot as plt

#Importando datos y utilidades
from sklearn.datasets import load_boston
from sklearn.model_selection import cross_val_score,ShuffleSplit
from sklearn.feature_selection import SelectKBest,f_regression

#Modelos
from sklearn.linear_model import LinearRegression as lr      #Regresión Lineal
from sklearn.preprocessing import PolynomialFeatures         #Para regresión polinomial

data=load_boston()
X=data.data
y=data.target

Regressor=lr()

Promedio=[]
D_estandar=[]
B_var=[]

cv=ShuffleSplit(n_splits=10, test_size=0.25, random_state=0)

Rl=cross_val_score(Regressor, X,y,scoring='explained_variance', cv=cv)
Promedio.append(Rl.mean())
D_estandar.append(Rl.std())

from skopt import gp_minimize

espacio=[(1,13)]  #Número de variables
n=50

def objective(params):
    n_var=params[0]
    X_new=SelectKBest(f_regression, k=n_var).fit_transform(X,y)
    R=cross_val_score(Regressor, X_new,y,scoring='explained_variance', cv=cv)
    return -R.mean()

Res_RL=gp_minimize(objective,espacio,n_calls=n,random_state=0)
B_var.append(Res_RL.x[0])

X_new=SelectKBest(f_regression, k=Res_RL.x[0]).fit_transform(X,y)
```

```
Rlo=cross_val_score(Regressor, X_new,y,scoring='explained_variance', cv=cv)
Promedio.append(Rlo.mean())
D_estandar.append(Rlo.std())

poly=PolynomialFeatures(degree=2)
Xn=poly.fit_transform(X)
Rl=cross_val_score(Regressor, Xn,y,scoring='explained_variance', cv=cv)
Promedio.append(Rl.mean())
D_estandar.append(Rl.std())

def objective1(params):
    n_var=params[0]
    X_new=SelectKBest(f_regression, k=n_var).fit_transform(X,y)
    Xn=poly.fit_transform(X_new)
    R=cross_val_score(Regressor, Xn,y,scoring='explained_variance', cv=cv)
    return -R.mean()

Res_RP=gp_minimize(objective1,espacio,n_calls=n,random_state=0)
B_var.append(Res_RP.x[0])
X_new=SelectKBest(f_regression, k=Res_RP.x[0]).fit_transform(X,y)
Xn=poly.fit_transform(X_new)
Rp=cross_val_score(Regressor, Xn,y,scoring='explained_variance', cv=cv)
Promedio.append(Rp.mean())
D_estandar.append(Rp.std())


Modelos=['Regresión\nLineal','Regresión\nLineal
optimizada','Regresión\nPolinomial','Regresión\nPolinomial optimizada']
plt.style.use('seaborn-bright')
plt.ioff()

plt.figure(figsize=(8,8))
plt.bar(x=np.arange(0,np.shape(Promedio)[0]),height=Promedio,width=0.75,yerr=D_est
andar,color='grey',tick_label=Modelos)
plt.xlabel('Comparación de Modelos',fontsize=16)
plt.ylabel('Varianza explicada',fontsize=16,family='TimesNewRoman')
plt.xticks(ticks=np.arange(0,4),fontsize=12,family='TimesNewRoman')
plt.yticks(fontsize=12,family='TimesNewRoman')
#plt.savefig("Fig 1.png")
plt.show()
```

Anexo [2] Prueba mejorada mediante el establecimiento de mejores hiper-parámetros

```
import sklearn
import numpy as np
import matplotlib.pyplot as plt

#Importando datos y utilidades
from sklearn.datasets import load_boston
from sklearn.model_selection import cross_val_score,ShuffleSplit
from sklearn.feature_selection import SelectKBest,f_regression
from sklearn.preprocessing import StandardScaler
#Modelos
from sklearn.svm import SVR       #Máquina de soporte vectorial de regresión

from skopt import gp_minimize     #Para optimización

data=load_boston()
X=data.data
y=data.target

kernels=['linear','rbf']

Promedio=[]
D_estandar=[]

cv=ShuffleSplit(n_splits=10, test_size=0.25, random_state=0)
scaler=StandardScaler()
X_scale=scaler.fit_transform(X)

espacio=[(1,13),        #Número de variables
        (0.01,10),      #C
        (0.01,1)]       #epsilon

n=50

B_var=[]

def objective(params):
    n_var=params[0]
    C=params[1]
    epsilon=params[2]

    Regressor=SVR(C=C,epsilon=epsilon)
    X_new=SelectKBest(f_regression, k=n_var).fit_transform(X_scale,y)
    R=cross_val_score(Regressor, X_new,y,scoring='explained_variance', cv=cv)
    return -R.mean()

for i in kernels:
    Regressor=SVR(kernel=i)
    Res_SVR=gp_minimize(objective,espacio,n_calls=n,random_state=0)
```

```
    X_new=SelectKBest(f_regression, k=Res_SVR.x[0]).fit_transform(X_scale,y)
    Regressor=SVR(kernel=i,C=Res_SVR.x[1],epsilon=Res_SVR.x[2])

    R_SVR=cross_val_score(Regressor, X_scale,y,scoring='explained_variance', cv=cv)
    Promedio.append(R_SVR.mean())
    D_estandar.append(R_SVR.std())
    B_var.append(Res_SVR.x)

Regressor=SVR(kernel='sigmoid')
X_new=SelectKBest(f_regression, k=Res_SVR.x[0]).fit_transform(X_scale,y)
Regressor=SVR(C=Res_SVR.x[1],epsilon=Res_SVR.x[2])
R_SVR=cross_val_score(Regressor, X_scale,y,scoring='explained_variance', cv=cv)
Promedio.append(R_SVR.mean())
D_estandar.append(R_SVR.std())
B_var.append(Res_SVR.x)

degree=np.array([2,3],dtype='int')
for j in degree:
    Regressor=SVR(kernel='poly',degree=j)
    Res_SVR=gp_minimize(objective,espacio,n_calls=n,random_state=0)

    X_new=SelectKBest(f_regression, k=Res_SVR.x[0]).fit_transform(X_scale,y)
    Regressor=SVR(C=Res_SVR.x[1],epsilon=Res_SVR.x[2])

    R_SVR=cross_val_score(Regressor, X_scale,y,scoring='explained_variance', cv=cv)
    Promedio.append(R_SVR.mean())
    D_estandar.append(R_SVR.std())
    B_var.append(Res_SVR.x)

Modelos=['linear','rbf','sigmoid','poly2','poly3']
plt.style.use('seaborn-bright')
plt.ioff()

plt.figure(figsize=(8,8))
plt.bar(x=np.arange(0,np.shape(Promedio)[0]),height=Promedio,width=0.75,yerr=D_est
andar,color='grey',tick_label=Modelos)
plt.xlabel('Desempeño de los kernel en las \n máquinas de soporte
vectorial',fontsize=16)
plt.ylabel('Varianza explicada',fontsize=16,family='TimesNewRoman')
plt.xticks(ticks=np.arange(0,6),fontsize=12,family='TimesNewRoman')
plt.yticks(fontsize=12,family='TimesNewRoman')
plt.savefig("Fig 3.png")
plt.show()
```

Anexo [3] Prueba con los algoritmos árbol de decisión, bosques aleatorios y vecinos más cercanos.

```
import sklearn
import numpy as np
import matplotlib.pyplot as plt

#Importando datos y utilidades
from sklearn.datasets import load_boston
from sklearn.model_selection import cross_val_score,ShuffleSplit
from sklearn.feature_selection import SelectKBest,f_regression
from sklearn.preprocessing import StandardScaler

#Modelos
from sklearn.tree import DecisionTreeRegressor as DTR       #Árbol de decisión
from sklearn.ensemble import RandomForestRegressor as RFR   #Bosques aleatorios
from sklearn.neighbors import KNeighborsRegressor as knn    #Vecinos más cercanos

data=load_boston()
X=data.data
y=data.target

Promedio=[]
D_estandar=[]
B_var=[]

cv=ShuffleSplit(n_splits=10, test_size=0.25, random_state=0)

Regressor=[DTR(),RFR()]

for i in Regressor:
    R=cross_val_score(i, X,y,scoring='explained_variance', cv=cv)
    Promedio.append(R.mean())
    D_estandar.append(R.std())

scaler=StandardScaler()
X_scale=scaler.fit_transform(X)

R=cross_val_score(knn(), X_scale,y,scoring='explained_variance', cv=cv)
Promedio.append(R.mean())
D_estandar.append(R.std())

from skopt import gp_minimize     #Para optimización
n=10

#Optimización del árbol de decisión
def objective_DTR(params):
    n_var=params[0]
    max_depth=params[1]
    min_samples_leaf=params[2]
```

```
    X_new=SelectKBest(f_regression, k=n_var).fit_transform(X,y)

Regressor=DTR(max_depth=max_depth,min_samples_leaf=min_samples_leaf,random
_state=0)
    R=cross_val_score(Regressor, X_new,y,scoring='explained_variance', cv=cv)
    return -R.mean()

espacio_DTR=[(1,13),        #Número de variables
             (1,10),        #Max_depth
             (1,10)]        #Min_samples_leaf

Res_DTR=gp_minimize(objective_DTR,espacio_DTR,n_calls=n,random_state=0)
X_new=SelectKBest(f_regression, k=Res_DTR.x[0]).fit_transform(X,y)
Regressor=DTR(max_depth=Res_DTR.x[1],min_samples_leaf=Res_DTR.x[2],random
_state=0)
R=cross_val_score(Regressor, X_new,y,scoring='explained_variance', cv=cv)
Promedio.append(R.mean())
D_estandar.append(R.std())
B_var.append(Res_DTR.x)

#Optimización de Bosques aleatorios
def objective_RFR(params):
    n_var=params[0]
    max_depth=params[1]
    min_samples_leaf=params[2]
    n_estimators=params[3]

    X_new=SelectKBest(f_regression, k=n_var).fit_transform(X,y)

Regressor=RFR(n_estimators=n_estimators,max_depth=max_depth,min_samples_leaf=
min_samples_leaf,random_state=0)
    R=cross_val_score(Regressor, X_new,y,scoring='explained_variance', cv=cv)
    return -R.mean()

espacio_RFR=[(1,13),        #Número de variables
             (1,10),        #Max_depth
             (1,10),        #Min_samples_leaf
             (1,150)]       #N_estimators

Res_RFR=gp_minimize(objective_RFR,espacio_RFR,n_calls=n,random_state=0)
X_new=SelectKBest(f_regression, k=Res_RFR.x[0]).fit_transform(X,y)

Regressor=RFR(n_estimators=Res_RFR.x[3],max_depth=Res_RFR.x[1],min_samples_
leaf=Res_RFR.x[2],random_state=0)
R=cross_val_score(Regressor, X_new,y,scoring='explained_variance', cv=cv)
Promedio.append(R.mean())
D_estandar.append(R.std())
B_var.append(Res_RFR.x)

#Optimización de Vecino más cercano
```

```
scaler=StandardScaler()
X_scale=scaler.fit_transform(X)

def objective_knn(params):
   n_var=params[0]
   n_neighbors=params[1]
   leaf_size=params[2]

   X_new=SelectKBest(f_regression, k=n_var).fit_transform(X_scale,y)

   Regressor=knn(n_neighbors=n_neighbors,leaf_size=leaf_size)
   R=cross_val_score(Regressor, X_new,y,scoring='explained_variance', cv=cv)
   return -R.mean()

espacio_knn=[(1,13),        #Número de variables
          (1,100),       #Número de vecinos
          (1,100)]       #Leaf_size

Res_knn=gp_minimize(objective_knn,espacio_knn,n_calls=n,random_state=0)
X_new=SelectKBest(f_regression, k=Res_knn.x[0]).fit_transform(X_scale,y)

Regressor=knn(n_neighbors=Res_knn.x[1],leaf_size=Res_knn.x[2])
R=cross_val_score(Regressor, X_new,y,scoring='explained_variance', cv=cv)

Promedio.append(R.mean())
D_estandar.append(R.std())
B_var.append(Res_knn.x)

Modelos=['Árbol de \ndecisión','Bosques\naleatorios','Vecino más \ncercano','Árbol de
\ndecisión \noptimizado','Bosques\naleatorios \noptimizado','Vecino más \ncercano
\noptimizado']
plt.style.use('seaborn-bright')
plt.ioff()

plt.figure(figsize=(10,10))
plt.bar(x=np.arange(0,np.shape(Promedio)[0]),height=Promedio,width=0.75,yerr=D_est
andar,color='grey',tick_label=Modelos)
plt.xlabel('Desempeño de otros algoritmos de regresión',fontsize=16)
plt.ylabel('Varianza explicada',fontsize=16,family='TimesNewRoman')
plt.xticks(ticks=np.arange(0,6),fontsize=12,family='TimesNewRoman')
plt.yticks(fontsize=12,family='TimesNewRoman')
plt.savefig("Fig 4.png")
plt.show()

print(Promedio)
print(D_estandar)
print(B_var)
```

Anexo [4] Prueba con una red neuronal de tipo perceptrón multicapa con 1 capa oculta con 100 neuronas y una capa de salida para la predicción.

```
import sklearn
import numpy as np
import matplotlib.pyplot as plt

#Importando datos y utilidades
from sklearn.datasets import load_boston
from sklearn.model_selection import cross_val_score,ShuffleSplit
from sklearn.feature_selection import SelectKBest,f_regression
from sklearn.preprocessing import StandardScaler

#Modelos
from sklearn.neural_network import MLPRegressor as MLPR

data=load_boston()
X=data.data
y=data.target

Promedio=[]
D_estandar=[]

cv=ShuffleSplit(n_splits=10, test_size=0.25, random_state=0)

def normalize(A):
   m=np.mean(A,axis=0)
   s=np.std(A,axis=0)
   return((A-m)/s)

X_scale=normalize(X)
y_scale=normalize(y)

Regressor=MLPR(hidden_layer_sizes=(100,),random_state=0,max_iter=500)

R=cross_val_score(Regressor, X_scale,y_scale,scoring='explained_variance', cv=cv)

Promedio.append(R.mean())
D_estandar.append(R.std())

print(Promedio)
print(D_estandar)
```

Anexo [5] Optimizando mediante una red neuronal con dos capas ocultas cuyos hiperparámetros fueron la cantidad de neuronas en cada capa.

```
from datetime import datetime

start=datetime.now()

import sklearn
import numpy as np
import matplotlib.pyplot as plt

#Importando datos y utilidades
from sklearn.datasets import load_boston
from sklearn.model_selection import cross_val_score,ShuffleSplit

#Modelos
from sklearn.neural_network import MLPRegressor as MLPR

data=load_boston()
X=data.data
y=data.target

Promedio=[]
D_estandar=[]

cv=ShuffleSplit(n_splits=10, test_size=0.25, random_state=0)

def normalize(A):
   m=np.mean(A,axis=0)
   s=np.std(A,axis=0)
   return((A-m)/s)

X_scale=normalize(X)
y_scale=normalize(y)


from skopt import gp_minimize     #Para optimización
n=50

#Optimización del árbol de decisión
def objective_MLPR(params):
   n1=params[0]
   n2=params[1]

   Regressor=MLPR(hidden_layer_sizes=(n1,n2),random_state=0,max_iter=500)
   R=cross_val_score(Regressor, X_scale,y_scale,scoring='explained_variance', cv=cv)
   return -R.mean()
```

```
espacio_MLPR=[(1,150),        #Neuronas primera capa
              (1,150)]        #Neuronas segunda capa

Res_MLPR=gp_minimize(objective_MLPR,espacio_MLPR,n_calls=n,random_state=0
)


Regressor=MLPR(hidden_layer_sizes=(Res_MLPR.x[0],Res_MLPR.x[1]),random_stat
e=0,max_iter=500)
R=cross_val_score(Regressor, X_scale,y_scale,scoring='explained_variance', cv=cv)
Promedio.append(R.mean())
D_estandar.append(R.std())

print(Promedio)
print(D_estandar)

plt.figure(figsize=(8,6))
plt.xlabel("Número de llamadas(n={})".format(n),fontsize=14)
plt.ylabel("Varianza explicada",fontsize=14)
plt.plot(np.linspace(1,n,n),-Res_MLPR.func_vals)
plt.xticks(fontsize=12)
plt.savefig('Optimización de la Red Neuronal.png')
plt.show()

delta=datetime.now() - start
print("El tiempo de cálculo fue {} ".format(delta))
```

Anexo [6] Código para determiner el desempeño de diferentes kernels en las máquinas de soporte vectorial para la clasificación.

```
import sklearn
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.svm import SVC

from sklearn.metrics import confusion_matrix,classification_report
from sklearn.feature_selection import SelectKBest,chi2
from sklearn.model_selection import cross_val_score,ShuffleSplit
from sklearn.preprocessing import StandardScaler


#Importando los datos
from sklearn.datasets import load_wine

data=load_wine()

X=data.data
y=data.target

Promedio=[]
Promedio_res=[]
D_estandar=[]
B_var=[]

kernels=['linear','rbf','poly','sigmoid']

cv=ShuffleSplit(n_splits=10, test_size=0.25, random_state=0)

scaler=StandardScaler()
X_scale=scaler.fit_transform(X)

for i in kernels:
   Classifier=SVC(kernel=i)
   R_SVC=cross_val_score(Classifier, X_scale,y,scoring='accuracy', cv=cv)
   Promedio_res.append(R_SVC.mean())
   D_estandar.append(R_SVC.std())
   Promedio.append(R_SVC)

P=pd.DataFrame(np.array(Promedio).T,columns=kernels).to_excel('SVC.xlsx')

plt.figure(figsize=(7,8))

plt.bar(x=np.arange(0,np.shape(Promedio_res)[0]),height=Promedio_res,width=0.75,ye
rr=D_estandar,color='grey',tick_label=kernels)
plt.xlabel('Desempeño de los kernel en las \n máquinas de soporte vectorial\n para
clasificación',fontsize=16)
```

```
plt.ylabel('Precisión',fontsize=16,family='Arial')
plt.xticks(ticks=np.arange(0,4),fontsize=12,family='Arial')
plt.yticks(fontsize=12,family='Arial')
plt.savefig("Fig 6.png")
plt.show()
```

Anexo [7] Código para determiner el desempeño de los árboles de decisión, bosques aleatorios y red bayesiana para la clasificación.

```
import sklearn
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from skopt import gp_minimize
from skopt.plots import plot_convergence

from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier as DTC
from sklearn.ensemble import RandomForestClassifier as RFC
from sklearn.naive_bayes import GaussianNB as GNB

from sklearn.feature_selection import SelectKBest,chi2
from sklearn.model_selection import cross_val_score,ShuffleSplit
from sklearn.preprocessing import StandardScaler

#Importando los datos
from sklearn.datasets import load_wine

data=load_wine()

X=data.data
y=data.target

Promedio=[]
Promedio_res=[]
D_estandar=[]

cv=ShuffleSplit(n_splits=10, test_size=0.25, random_state=0)

scaler=StandardScaler()
X_scale=scaler.fit_transform(X)

Classifier=[DTC(),RFC(),GNB()]

for i in Classifier:
    R=cross_val_score(i, X,y,scoring='accuracy', cv=cv)
    Promedio_res.append(R.mean())
    D_estandar.append(R.std())
    Promedio.append(R)


Algorithms=['Árbol de decisión','Bosques aleatorios','Red Bayesiana','Árbol de decisión\n optimizado']  #

espacio=[(1,13),
```

```
        (1,10),
        (1,10)]

n=100

def objective(params):
    n_var=params[0]
    max_depth=params[1]
    min_samples_leaf=params[2]

    Classifier=DTC(max_depth=max_depth,min_samples_leaf=min_samples_leaf)
    X_new=SelectKBest(chi2, k=n_var).fit_transform(X,y)
    R=cross_val_score(Classifier, X_new,y,scoring='accuracy', cv=cv)
    return -R.mean()

Res_DTop=gp_minimize(objective,espacio,n_calls=n,random_state=0)

Classifier=DTC(max_depth=Res_DTop.x[1],min_samples_leaf=Res_DTop.x[2])
X_new=SelectKBest(chi2, k=Res_DTop.x[0]).fit_transform(X,y)
R=cross_val_score(Classifier, X_new,y,scoring='accuracy', cv=cv)

Promedio.append(R)
Promedio_res.append(R.mean())
D_estandar.append(R.std())

plt.figure(figsize=(8,8))

plt.xlabel("Número de llamadas(n={})".format(n),fontsize=14)
plt.ylabel("Precisión del modelo",fontsize=14)
plt.plot(np.linspace(1,n,n),-Res_DTop.func_vals)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.savefig('Fig 7.png')
plt.show()

plt.figure(figsize=(8,8))
plt.bar(x=np.arange(0,np.shape(Promedio_res)[0]),height=Promedio_res,width=0.75,ye
rr=D_estandar,color='grey',tick_label=Algorithms)
plt.xlabel('Desempeño de los árboles de decisión\n bosques aleatorios y red
bayesiana',fontsize=16)
plt.ylabel('Precisión',fontsize=16,family='Arial')
plt.xticks(ticks=np.arange(0,4),fontsize=12,family='Arial')
plt.yticks(fontsize=12,family='Arial')
plt.savefig("Fig 8.png")
plt.show()

P=pd.DataFrame(np.array(Promedio).T,columns=Algorithms).to_excel('Trees.xlsx')
```

Anexo [8] Código para determiner el desempeño del algoritmo de vecinos más cercanos en clasificación.

```
import sklearn
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from skopt import gp_minimize

from sklearn.neighbors import KNeighborsClassifier as knn

from sklearn.feature_selection import SelectKBest,chi2
from sklearn.model_selection import cross_val_score,ShuffleSplit
from sklearn.preprocessing import StandardScaler

#Importando los datos
from sklearn.datasets import load_wine

data=load_wine()

X=data.data
y=data.target

Promedio=[]
Promedio_res=[]
D_estandar=[]

cv=ShuffleSplit(n_splits=10, test_size=0.25, random_state=0)

scaler=StandardScaler()
X_scale=scaler.fit_transform(X)

Classifier=knn()
R=cross_val_score(Classifier, X_scale,y,scoring='accuracy', cv=cv)
Promedio.append(R)
Promedio_res.append(R.mean())
D_estandar.append(R.std())

Algorithms=['Vecino más cercano','Vecino más cercano \n optimizado']

espacio=[(1,13),
        (1,50)]

n=100

def objective(params):
    n_var=params[0]
    n_neighbors=params[1]
```

```
    Classifier=knn(n_neighbors=n_neighbors)
    X_new=SelectKBest(chi2, k=n_var).fit_transform(X,y)
    X_scale=scaler.fit_transform(X_new)
    R=cross_val_score(Classifier, X_scale,y,scoring='accuracy', cv=cv)
    return -R.mean()

Res_knn=gp_minimize(objective,espacio,n_calls=n,random_state=0)

Classifier=knn(n_neighbors=Res_knn.x[1])
X_new=SelectKBest(chi2, k=Res_knn.x[0]).fit_transform(X,y)
X_scale=scaler.fit_transform(X_new)
R=cross_val_score(Classifier, X_scale,y,scoring='accuracy', cv=cv)

Promedio.append(R)
Promedio_res.append(R.mean())
D_estandar.append(R.std())

plt.figure(figsize=(8,8))

plt.xlabel("Número de llamadas(n={})".format(n),fontsize=14)
plt.ylabel("Precisión del modelo",fontsize=14)
plt.plot(np.linspace(1,n,n),-Res_knn.func_vals)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.savefig('Fig 9.png')
plt.show()

plt.figure(figsize=(6,6))
plt.bar(x=np.arange(0,np.shape(Promedio_res)[0]),height=Promedio_res,width=0.75,ye
rr=D_estandar,color='grey',tick_label=Algorithms)
plt.xlabel('Desempeño del algoritmo de vecinos más cercanos',fontsize=16)
plt.ylabel('Precisión',fontsize=16,family='Arial')
plt.xticks(ticks=np.arange(0,2),fontsize=12,family='Arial')
plt.yticks(fontsize=12,family='Arial')
plt.savefig("Fig 10.png")
plt.show()

P=pd.DataFrame(np.array(Promedio).T,columns=Algorithms).to_excel('knn.xlsx')
```

Anexo [9] Código para determiner el desempeño de las redes neuronales en clasificación.

```
import sklearn
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from skopt import gp_minimize

from sklearn.neural_network import MLPClassifier as MLPC

from sklearn.model_selection import cross_val_score,ShuffleSplit


#Importando los datos
from sklearn.datasets import load_wine

data=load_wine()

X=data.data
y=data.target

Promedio=[]
Promedio_res=[]
D_estandar=[]

cv=ShuffleSplit(n_splits=10, test_size=0.25, random_state=0)

def normalize(A):
    m=np.mean(A,axis=0)
    s=np.std(A,axis=0)
    return((A-m)/s)

X_scale=normalize(X)

Classifier=MLPC(hidden_layer_sizes=(100,),max_iter=500)
R=cross_val_score(Classifier, X_scale,y,scoring='accuracy', cv=cv)
Promedio.append(R)
Promedio_res.append(R.mean())
D_estandar.append(R.std())

n=100

#Optimización del árbol de decisión
def objective_MLPR(params):
    n1=params[0]
    n2=params[1]
```

```
    Classifier=MLPC(hidden_layer_sizes=(n1,n2),random_state=0,max_iter=500)
    R=cross_val_score(Classifier, X_scale,y,scoring='accuracy', cv=cv)
    return -R.mean()

espacio_MLPR=[(1,150),        #Neuronas primera capa
              (1,150)]        #Neuronas segunda capa

Res_MLPC=gp_minimize(objective_MLPR,espacio_MLPR,n_calls=n,random_state=0
)

Classifier=MLPC(hidden_layer_sizes=(Res_MLPC.x[0],Res_MLPC.x[1]),random_stat
e=0,max_iter=500)
R=cross_val_score(Classifier, X_scale,y,scoring='accuracy', cv=cv)
Promedio_res.append(R.mean())
D_estandar.append(R.std())
Promedio.append(R)

plt.figure(figsize=(8,6))
plt.xlabel("Número de llamadas(n={})".format(n),fontsize=14)
plt.ylabel("Precisión",fontsize=14)
plt.plot(np.linspace(1,n,n),-Res_MLPC.func_vals)
plt.xticks(fontsize=12)
plt.savefig('Optimización de la Red Neuronal de Clasificación.png')
plt.show()

Algoritmos=['Red Neuronal con \n parámetros de defecto','Red Neuronal\n optimizada']
P=pd.DataFrame(np.array(Promedio).T,columns=['Red Neuronal','Red Neuronal
optimizada']).to_excel('MLPC.xlsx')

plt.figure(figsize=(7,8))

plt.bar(x=np.arange(0,np.shape(Promedio_res)[0]),height=Promedio_res,width=0.75,ye
rr=D_estandar,color='grey',tick_label=Algoritmos)
plt.xlabel('Desempeño de las redes neuronales',fontsize=16)
plt.ylabel('Precisión',fontsize=16,family='Arial')
plt.xticks(ticks=np.arange(0,2),fontsize=12,family='Arial')
plt.yticks(fontsize=12,family='Arial')
plt.savefig("Fig 11.png")
plt.show()
```