PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

Desarrollo de modelo predictivo de desgaste basado en datos de ensayos según ASTM G-65 utilizando algoritmos de Machine Learning

Tesis para obtener el título profesional de Ingeniero Mecánico

INGENIERÍA MECÁNICA

AUTOR:
Renato Cabanillas Flores

ASESOR:
Dr. José Sakihama Uehara

Lima, julio, 2022

RESUMEN

Los ensayos de desgaste según la norma ASTM G-65 son realizados para determinar la resistencia al desgaste abrasivo de bajo esfuerzo de un material mediante su exposición al contacto con arena seca. Estos ensayos permiten la evaluación de recargues duros o hardfacing con la finalidad de encontrar los efectos de los elementos aleantes sobre la microestructura y la resistencia al desgaste. Por su parte, el aprendizaje automático, conocido como Machine Learning, es una técnica del campo de la inteligencia artificial que busca desarrollar modelos computacionales con la capacidad de realizar tareas de clasificación y regresión.

La metodología utilizada para realizar el entrenamiento, y posterior evaluación de los modelos obtenidos, consiste en la digitalización de los registros de ensayos de desgaste ejecutados por la American Welding Society, el análisis del comportamiento de la pérdida de masa en función del porcentaje de la concentración de los elementos presentes en el depósito del recargue duro y el desarrollo de los siguientes algoritmos de modelos de aprendizaje automático: k-vecinos cercanos (KNN), red neuronal artificial (ANN) y máquina de aprendizaje extremo (ELM). Posterior al entrenamiento, se emplearon los modelos ya entrenados para calcular la pérdida de masa en probetas previamente ensayadas en el Laboratorio de Materiales de la Pontificia Universidad Católica del Perú (PUCP) y así evaluar la efectividad de los modelos en la sección de resultados.

Para los modelos entrenados se identificaron las variantes con mejor efectividad en la predicción de pérdida de masa, las cuales fueron la red neuronal artificial de 3 capas entrenada en 1000 épocas, el modelo de k-vecinos cercanos con 6 vecinos y la máquina de aprendizaje extremo con 10,000 neuronas. Para la comparación con datos de ensayos realizado sen la PUCP se obtuvo un error medio absoluto de 0.086 g para ANN, 0.726 g para KNN 0.853 g para ELM; en contraste con los valores de 0.228 g, 0.321 g y 0.666 g obtenidos para los ensayos realizado por la AWS, respectivamente. De entre los 3 modelos entrenados, se identifica que la red neuronal artificial congrega la mayor cantidad de puntos cercanos a la igualdad entre el valor real de pérdida de masa y la predicción calculada mediante el modelo.

Se concluye que la ANN puede predecir con éxito la pérdida de masa en función de la composición química del depósito y su dureza, alcanzando una precisión del 85.75%. Por otro lado, la ELM requiere elevados recursos computacionales para entrenarse por encima de las 500,000 neuronas, así como un análisis más profundo para evitar el sobreajuste del modelo a los datos de entrenamiento. El algoritmo KNN no calcula exitosamente la masa perdida en un ensayo de desgaste debido a que entrega valores de promedios locales para datos que no se estructuran de forma ordenada. Finalmente, los resultados alcanzados brindan validez a la aplicación de técnicas de aprendizaje automático para encontrar la pérdida de masa en ensayos de desgaste.

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

TEMA DE TESIS PARA OPTAR EL TÍTULO DE INGENIERO MECÁNICO

TÍTULO : **DESARROLLO DE MODELO PREDICTIVO DE DESGASTE BASADO EN DATOS DE ENSAYOS SEGÚN ASTM G-65 UTILIZANDO ALGORITMOS DE MACHINE LEARNING**

ÁREA : Materiales

PROPUESTO POR : Renato Cabanillas Flores

ASESOR : Dr. José Sakihama Uehara

TESISTA : Renato Cabanillas Flores

CÓDIGO : 20150687

FECHA : 25/11/2021

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

TEMA DE TESIS PARA OPTAR EL TÍTULO DE INGENIERO MECÁNICO

DESARROLLO DE MODELO PREDICTIVO DE DESGASTE BASADO EN DATOS DE ENSAYOS SEGÚN ASTM G-65 UTILIZANDO ALGORITMOS DE MACHINE LEARNING

Introducción

1. Marco teórico
2. Metodología
3. Resultados
4. Conclusiones

Bibliografía

Anexos

Dr. José Sakihama

Asesor

Contenido

Tabla de ilustraciones

INTRODUCCIÓN

En nuestro país, la distribución del producto bruto interno (PBI) marca que el 49% de éste proviene de los sectores agricultura, pesquero e industria, donde este último representa el 83% del bloque [1]. Asimismo, el sector minero representa el 10% del PBI y es responsable del 16% de la inversión privada con proyectos en diferentes fases de inversión cuya cartera acumulada asciende a más de 15,000 millones de dólares [2]. En todas las actividades económicas mencionadas se requiere el uso de equipos y maquinaria que estarán expuestos a desgaste abrasivo.

En ese sentido, el uso de recargues duros como método de protección y aumento de la resistencia al desgaste de los componentes es una práctica ampliamente utilizada. Con el objetivo de encontrar la concentración idónea de elementos químicos que componen el material de aporte, muchas instituciones han realizado ensayos según la norma ASTM G-65 probando distintos porcentajes de carbono y elementos aleantes como titanio, vanadio, molibdeno, etc. Sin embargo, la realización de los ensayos mencionados implica una inversión de tiempo y dinero, pues demanda materiales, equipos, personal, entre otros.

En la actualidad, los avances en la ciencia de los datos, particularmente en aprendizaje automático (Machine Learning), han permitido desarrollar algoritmos con la capacidad de tomar los resultados de cientos de ensayos y generar modelos con capacidad de predecir los resultados para nuevas combinaciones o concentraciones. Estos algoritmos existen desde hace más de 15 años y recientemente han obtenido popularidad al ser utilizados para obtener modelos predictivos en diversas áreas como lo son las finanzas, salud, manufactura o el control automático, entre otras.

A partir de ello, se propone desarrollar un sistema que, utilizando algoritmos de *machine learning*, sea entrenado con los datos de los ensayos realizados por la American Welding Society (AWS). Luego, evaluar el sistema desarrollado con resultados obtenidos de ensayos realizados en la PUCP con la finalidad de encontrar el algoritmo que presente la mejor aproximación en la predicción de resistencia al desgaste. En adición, los resultados de este

trabajo no solo permitirán brindar mayor claridad sobre la utilidad de entrenar sistemas de aprendizaje automático para predecir el desgaste, sino que también podrían ser llevados al análisis de otros tipos de desgaste como lo es el adhesivo o por corrosión.

Objetivo general:

Predecir el desgaste abrasivo en recargues duros por soldadura según el ensayo ASTM G-65 utilizando algoritmos de aprendizaje automático.

Objetivos específicos:

1. Obtener, procesar y preparar datos de ensayo ASTM G-65 para que sirvan como entrenamiento de sistemas de aprendizaje automático.
2. Entrenar sistemas desarrollados con diferentes algoritmos para predecir la pérdida de masa esperada en un ensayo utilizando Python.
3. Evaluar la aproximación de los resultados obtenidos mediante los sistemas desarrollados en contraste con los datos reales de laboratorio.
4. Determinar el algoritmo con mejor rendimiento y validar su uso para la predicción del desgaste abrasivo.

ET LUX IN TENEBRIS LUCET
MCMXVII

MARCO TEÓRICO

Este capítulo desarrollará los conceptos que permitan entender el desgaste abrasivo, los recargues duros, el Machine Learning y las redes neuronales artificiales. En consistencia con la necesidad del uso de grandes cantidades de información experimental, se incluirá una explicación resumida del ensayo de desgaste abrasivo ASTM G-65, cuyos resultados serán estudiados en este trabajo.

1.1. Desgaste abrasivo

El desgaste es el proceso en el que la interacción de la superficie de un material con su entorno resulta en pérdida de esta. Este proceso es causado por el movimiento relativo entre dos elementos que, independientemente de su estado, se encuentren en contacto y bajo carga [3]. En la mayoría de los casos, el desgaste es causado a raíz del contacto de las asperezas de dos superficies y se produce de manera progresiva en el tiempo [4]. Estas condiciones se presentan de manera cotidiana en labores de generación de energía, manufactura, así como transporte y procesamiento de minerales. Los principales mecanismos de desgaste son el desgaste abrasivo, erosivo, por cavitación, adhesivo, corrosivo, por fricción y por fatiga. Cada uno de ellos responde a condiciones físicas, cinemática del movimiento relativo entre elementos y la composición química, que definirán cuál de los mecanismos mencionados primará. Asimismo, en condiciones de trabajo reales, se presentan varios tipos de desgaste en un mismo proceso.

El desgaste abrasivo se presenta cuando un sólido entra en contacto y movimiento relativo con un material de igual o superior dureza. Este mecanismo ocurre tanto en el contacto de dos superficies de diferente dureza como en la interacción de dos superficies con partículas de dureza superior. La presencia de rayones y cortes son consecuencias notorias del desgaste abrasivo [4].

Este sistema de desgaste se puede clasificar en función del tipo de contacto, dentro de lo cual existe la abrasión de bajo esfuerzo, abrasión de alto esfuerzo y la abrasión penetrante; así como se puede segmentar según el mecanismo físico que lo origina, lo cual puede ser corte, fatiga, fractura y extracción de grano [5].

1.1.1. Clasificación según el tipo contacto

1.1.1.1. Desgaste abrasivo de bajo esfuerzo

El desgaste abrasivo de bajo esfuerzo se produce en situaciones donde la fuerza de contacto entre las partículas o superficie abrasiva es baja. Usualmente este desgaste se presenta en el deslizamiento de

partículas en el cual las fuerzas externas son menores. Dada la baja fuerza presente entre las superficies, el desgaste se incrementará en función del paso continuo de las partículas sobre el material blando [6].

En concordancia con la poca incidencia de la fuerza de contacto en este tipo de desgaste, factores como la velocidad de abrasión, dureza, ángulo de ataque, filo y tamaño del grano son los que determinarán la severidad de este. Asimismo, el porcentaje de carbono presente en la superficie blanda será determinante para mejorar la resistencia al desgaste [5].

1.1.1.2. Desgaste abrasivo de alto esfuerzo

El desgaste abrasivo de alto esfuerzo se presenta cuando las superficies o partículas en contacto están sometidas a altas cargas. En el caso de las partículas duras, estás tienen un área de contacto reducida con la superficie blanda, por lo cual se pueden generar esfuerzos en el orden de los 2800 MPa [7].

En contraste con el desgaste abrasivo de bajo esfuerzo, la fuerza de contacto es primordial para determinar la severidad del desgaste. Los demás factores mencionados en el anterior tipo de desgaste pasan a una posición secundaria. Debido a los altos esfuerzos se generarán grietas y fracturas en los microconstituyentes [7].

1.1.1.3. Desgaste abrasivo penetrante

Este tipo de desgaste ocurre cuando la fuerza de contacto entre las superficies es de impacto. En este caso, los altos esfuerzos son causados por las altas fuerzas de impacto, lo que lleva a la dureza del material a un segundo plano. En el desgaste abrasivo penetrante se requieren superficies tenaces para absorber la energía del impacto, por lo cual los materiales frágiles no son recomendados frente a esta situación [6].

1.1.2. Clasificación según mecanismo de desgaste

A partir del análisis microscópico se pueden identificar 4 mecanismos en los que se genera el desgaste abrasivo, los cuales son: corte, fractura, fatiga y extracción de grano. El mecanismo de corte consiste en el rayado de la superficie por parte de granos duros a nivel microscópico; el uso de lubricantes resulta beneficioso para reducir el efecto del corte. El mecanismo de fractura se produce cuando un indentador filudo impacta con una superficie frágil y se presenta como grietas de ventilación que se propagan a la superficie, fragmentación localizada o acumulación sucesiva de grietas debido al movimiento de granos. La fatiga como mecanismo de desgaste es causado por el esfuerzo repetitivo producido por granos sobre

la superficie. Finalmente, la extracción de grano es un mecanismo raro que se presenta usualmente en elementos cerámicos con granos de gran tamaño y baja fuerza intergranular [4].

1.2. Recargues duros

La aplicación de recargues duros es una práctica utilizada a nivel industrial como técnica para evitar el desgaste excesivo de una pieza y orientar las actividades de mantenimiento hacia la recuperación de componentes, en lugar de incurrir en el reemplazo de éstos [8]. También conocido como "hardfacing", los recargues duros son aleaciones especiales aplicadas sobre superficies metálicas mediante los distintos procesos de soldadura. El objetivo de la aplicación es mejorar la resistencia al desgaste, sea de abrasión, impacto, corrosión u otros [9].

Las aplicaciones de hardfacing pueden ser divididas en dos grupos: fabricación y mantenimiento. En el primer caso, se aplica el recubrimiento en las zonas que estarán sujetas a mayor desgaste. De esta manera, no toda la pieza o componente se fabrica con el mismo material y así se reducen costos. En el segundo caso, los recargues servirán para recuperar tolerancias dimensionales iniciales y proteger zonas blandas, así como mejorar sus propiedades mecánicas con la finalidad de prolongar la vida útil del activo.

1.2.1. Selección del material de aporte

La selección del material de aporte depende de los siguientes tres factores: metal base, tipo de desgaste y proceso de soldeo. A continuación, se desarrollará cada uno de éstos.

1.2.1.1. Metal base

La compatibilidad entre en metal base y el material de aporte es un factor que tendrá impacto en cualquier aplicación de recargues duros. Entre las condiciones que afectan a la compatibilidad de los materiales a soldar se encuentra el espesor del metal a depositar, la cantidad de capas a utilizar, así como los agrietamientos generados por gradientes de temperatura, las tensiones residuales y deformaciones [6].

1.2.1.2. Tipo de desgaste

Este factor influenciará la elección del material para la última capa del recargue, la cual se encontrará en contacto con el material.

1.2.1.3. Proceso de soldadura

Existen una variedad de elementos que influyen en la elección del proceso de soldadura, entre ellos se destacan las dimensiones del elemento a soldar, los requerimientos de servicio, las propiedades mecánicas del material base y del material de aporte, el espacio donde se realizará el procedimiento, el costo de la operación y la habilidad del soldador [10].

1.2.2. Microestructuras resistentes al desgaste

Las principales microestructuras, las cuales presentan relaciones entre sus propiedades mecánicas y su resistencia al desgaste, serán presentadas a continuación.

1.2.2.1. Martensita

La martensita es un constituyente conocido por su alta dureza y fragilidad. Su dureza es proporcional al porcentaje de carbono presente, por otro lado, los elementos aleantes no afectan sus propiedades [11].

1.2.2.2. Austenita

Este microconstituyente es una solución sólida de carbono y hierro (y). Presenta la mayor solubilidad del carbono a 1145 °C. En condiciones ambientales presenta inestabilidad salvo que presente un elevado contenido de carbono o elementos aleantes. Se destacan aspectos como la forma de sus granos, la cual es poligonal y la ausencia de magnetismo [11].

1.2.2.3. Carburos

Los carburos son el producto de la unión entre el carbono y un elemento aleante, se caracterizan por su alta dureza y resistencia. En lo referente al desgaste, su capacidad para resistirlo depende de la relación de dureza entre los carburos y la matriz en la que se encuentran, pudiendo tener un comportamiento positivo de mejora de resistencia cuando la dureza del carburo es mayor que la de la matriz, o negativo al funcionar como concentrador de esfuerzos en el caso contrario. Los carburos pueden presentarse en un sistema ordenado en forma de red, así como de manera dispersa [12].

Los carburos en red presentan una estructura con elevada dureza y fragilidad, lo cual los hace susceptibles al impacto. De acuerdo con lo mencionado anteriormente sobre la austenita, ésta acompaña a los carburos, dada su contenido de elementos aleantes, fungiendo como matriz. Por otro lado, los

carburos dispersos, presentan el mejor rendimiento frente a desgastes combinados cuando se emplean adecuadamente los depósitos [13].

1.2.3. Influencia de la composición química de los electrodos en la resistencia al desgaste

Las propiedades mecánicas y de resistencia al desgaste serán determinadas por los elementos químicos presentes en el revestimiento del electrodo a utilizarse para el recargue duro. Elementos como el vanadio, titanio, molibdeno y carbono son conocidos por tener gran influencia en la dureza y la microestructura del hardfacing. A continuación, se presentan los efectos de la presencia de los elementos indicados en la mejora de la resistencia al desgaste.

En el caso del carbono, éste se agrega en forma de grafito, y se busca tener una concentración entre el 8 y 10%; en ese rango se presenta una relación directamente proporcional entre el porcentaje de grafito y el incremento de la dureza. Cuando se supera el 12% se presenta un cambio en la matriz debido al alto porcentaje de carbono, lo cual conduce a fisuras y reducción en la resistencia. El vanadio y titanio presentan consecuencias similares sobre la dureza del recargue en el que son depositados; la dureza incrementa proporcionalmente a la concentración de estos elementos. Los carburos de titanio y vanadio alcanzarán durezas de 3200 HV y 2800 HV respectivamente. La recomendación es que el contenido de Fe-V y Fe-Ti se encuentre en los rangos de 10-12% y 12-15% respectivamente, de lo contrario se acrecentará la presencia de escoria. Finalmente, el molibdeno presente un buen comportamiento en la formación de carburos hasta el 4% de concentración; sin embargo, al pasar ese contenido genera una microestructura susceptible a las fisuras por fragilidad.

1.2.4. Aplicación del recargue duro

Para la aplicación del recargue se debe realizar un procedimiento establecido que asegure su correcta deposición. Este proceso inicia con la limpieza, la cual tiene el objetivo de evitar defectos como porosidades y grietas; de existir presencia de óxidos o herrumbres, éstos serán esmerilados, también se retirarán las pinturas, grasas y suciedad presentes. Seguidamente se realizarán los procesos de mecanizado necesarios para preparar la superficie. En caso se conciba el uso de más de una capa se repetirá el proceso de limpieza antes de colocar la siguiente capa. Asimismo, en el caso de aceros con porcentajes de carbono mayores al 0.45% se deberá reducir el gradiente térmico mediante un precalentamiento con la finalidad de evitar la formación de microestructuras frágiles como la martensita

y la bainita inferior. Finalmente, se deben considerar las deformaciones producidas por temperaturas elevadas, las cuales propician las fisuras por tensiones residuales [10].

1.3. Machine Learning

La cuarta revolución industrial ha traído consigo el uso de nuevas tecnologías y términos que día a día cobran mayor relevancia; tan sólo en el 2020, Elsevier publicó más de 10,000 artículos de ingeniería que desarrollan el concepto de Machine Learning. Este proceso computacional, también referenciado como aprendizaje automático, consiste en la construcción de modelos predictivos en función de patrones que se observan en grandes conjuntos de datos [14]. A diferencia de la simulación numérica, la cual utiliza un método inductivo a partir de ecuaciones básicas aplicadas a pequeños elementos, este proceso consiste en la deducción del comportamiento basándose en estadística sin la necesidad de instrucciones específicas [15]. En el marco de la ciencia de los datos, el aprendizaje automático es el puente entre las redes neuronales y la inteligencia artificial. La computadora aprende mediante el paso de grandes cantidades de bloques de datos por redes neuronales, las cuales son modificadas en función de su éxito en la predicción de resultados. Asimismo, la toma de decisiones basada en el aprendizaje adquirido es el eje de la inteligencia artificial.

1.3.1. Modos de aprendizaje

Los modos de aprendizaje se pueden clasificar según el tipo de supervisión, la capacidad de ampliar el aprendizaje y cómo realizan el proceso deductivo. Según el problema que se desea resolver o la data que será tratada se elige el modo y posteriormente se construye el algoritmo en función de éste.

1.3.1.1. Nivel de supervisión

Ésta es la característica más importante del modelo de aprendizaje y definirá los alcances y limitaciones del sistema. A medida que se reduce el nivel de supervisión, se incrementa la automatización de los procesos.

En el aprendizaje supervisado, el sistema conoce que datos corresponden a la entrada y que datos son la salida o resultado esperado [16]. En estos modelos, el algoritmo buscará acercar su predicción al valor de la variable de interés, la cual puede ser valores numéricos, como de afirmación o negación. Las tareas principales para la cuales se considera este método de aprendizaje son la clasificación y la predicción de valor objetivo [17]. En la actualidad, las aplicaciones del aprendizaje supervisado son las responsables del reconocimiento facial y de huellas dactilares, el dictado de voz a texto, detección de

fraude basado en patrones históricos y la mejora de los planes de mantenimiento en equipos y máquinas. En el campo de la ingeniería mecánica, el aprendizaje supervisado está presente en la realidad aumentada para instalación y reparación de componentes, el control adaptativo para optimización de procesos y la comercialización de maquinaria [18]. A continuación, se presentan los algoritmos comúnmente utilizados para el desarrollo de este aprendizaje.

1.3.2. Algoritmos de aprendizaje automático

1.3.2.1. K-vecinos cercanos

Este método consiste en la creación de grupos, o vecindarios, en los cuales un conjunto de datos corresponde a un resultado único. La tarea de aprendizaje se basa en la formación de vecindarios, los cuales son identificados como las secciones de distintos colores en la figura 1, bajo una métrica de distancia elegida y, en función de ello, la generación de una función de predicción, la cual permite determinar a qué vecindario pertenece un nuevo valor.

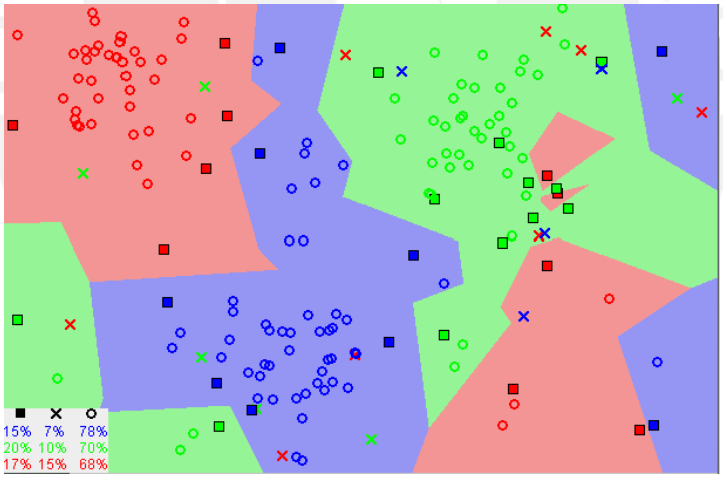


Figura 1. Esquema de clasificación K-nearest neighbors. [19]

Cada elemento del conjunto de datos es denotado por un vector $\boldsymbol{x}$ con un componente por cada variable de interés, siendo representado como $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$. Dentro del conjunto de datos, los bloques de similares características serán agrupados en vecindarios, de esta manera, un vecindario y contiene múltiples elementos x y se denota como $[\boldsymbol{x_a}, \boldsymbol{x_b}, \dots, \boldsymbol{x_z}] \in \boldsymbol{y}$. Para las tareas de clasificación, el algoritmo se apoya en los vecindarios designados por los datos que se proveen al sistema. Dado que es un modelo de aprendizaje supervisado, este es provisto de una base de datos $\boldsymbol{D} = \{z_1, z_2, \dots, z_n\}$ donde cada z representa una correspondencia $(\boldsymbol{x}, \boldsymbol{y})$, considerando lo indicado sobre x y y anteriormente [20].

Sobre la base del entrenamiento con los datos de $\boldsymbol{D}$, el algoritmo procederá a ubicar los nuevos valores en alguno de los vecindarios $\boldsymbol{y}$ definidos con el objetivo de encontrar el vecindario para el cual las características del nuevo valor encuentren mayor cercanía. La cercanía se define como la menor distancia entre el nuevo vector y el conjunto de vectores de un vecindario. Para el cálculo de la distancia existe una variedad de enfoques, los cuales son presentados a continuación:

- Distancia euclidiana

  Es la medición de la raíz cuadrada de la sumatoria de las diferencias entre coordenadas elevada al cuadrado. Esta medida es la más común para cálculo de distancia entre puntos en 2 dimensiones como se muestra en la ecuación 1.

$$Dist_{Euc} = \sqrt{\sum_{k=1}^{m} (x_{ik} - x_{jk})^2} \quad (1)$$

  Donde m es la cantidad de características o componentes de $\boldsymbol{x}$ que se estudian, x_{ik} representará a una componente del vector más cercano de un vecindario y x_{jk} a la componente de $\boldsymbol{x}$.

- Distancia de Manhattan:

  Esta medida considera la suma de las distancias entre los diferentes componentes como una sumatoria de valores absolutos. En la ecuación 2 se aprecia la expresión que toma los valores indicados.

$$Dist_{Mht} = \sum_{k=1}^{m} |x_{ik} - x_{jk}| \quad (2)$$

- Distancia de Minkowski:

  Esta medida es la forma más general de determinar la distancia dado que a partir del valor p. Dependiendo de si es p es 1 o 2, la ecuación de distancia de Minkowski será igual a la de Manhattan y Euclidiana, respectivamente. En la ecuación 3 se observe como dependiendo de p se toma la forma de la ecuación 1 o 2.

$$Dist_{Mik} = \sqrt[p]{\sum_{k=1}^{m} |x_{ik} - x_{jk}|^p} \quad (3)$$

Esta generalización de la distancia permite adecuar el parámetro a las necesidades del algoritmo de una manera más práctica.

Finalmente, el valor de k para el modelo de los k-vecinos cercanos determinará el número de vecindarios para el cual se desarrolla el modelo. Cuando el número de vecindarios es 1, es decir, un modelo de 1-NN se obtiene el desarrollo para un modelo de regresión lineal simple, el cual será desarrollado a continuación.

1.3.2.2. Regresión lineal

El método de regresión lineal consiste en la deducción de una relación lineal entre dos variables: X como el elemento independiente e Y como dependiente, las cuales, en la figura 2, representan el eje de las abscisas y las ordenadas, respectivamente. Estas variables serán referidas también como el vector predictor y el valor esperado, respectivamente. Al igual que el método anterior, puede ser aplicado tanto para la clasificación como predicción de valores objetivo; sin embargo, es mayormente utilizado para tareas predictivas [20].

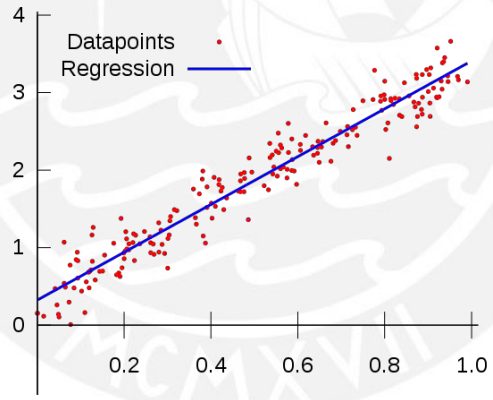


Figura 2. Diagrama de dispersión para regresión lineal mediante Machine Learning. [21]

El modelo matemático de regresión lineal para el aprendizaje automático considera parámetros desconocidos que serán determinados en el entrenamiento, los cuales son introducidos a la ecuación en conjunto con una variable que represente el error, de manera que la relación toma la forma vista en la ecuación 4.

$$\boldsymbol{Y} = \beta_0 + \beta_1 \boldsymbol{X} + \varepsilon \qquad (4)$$

Donde β_0 y β_1 son los parámetros desconocidos y ε representa los errores o valores no considerados en el modelo [16]. Se observa que, de no existir los parámetros por determinar, la relación tomaría la forma

$Y = b + mX$ para la cual no habría proceso de aprendizaje y se reduciría el modelo a una recta definida [20].

Este modelo es entrenado con el objetivo de reducir el valor ε al mínimo, para ello es importante considerar que, al tratarse de un método lineal, la suma de errores positivos y negativos resultaría un error absoluto que no aporte una referencia apropiada de la efectividad del modelo. Por ello, el error será elevado al cuadrado y de esta forma, al acumular los errores para cada dato, se podrá tener una medida representativa [16]. De esta manera, se buscará reducir el error cuadrado al mínimo, el cual es representado en la ecuación 5.

$$SCE = \sum_{i=1}^{n} {\varepsilon_i}^2 \tag{5}$$

Luego de haber revisado las relaciones planteadas para los valores que definirán la recta de regresión, se procede a detallar el proceso de aprendizaje bajo este modelo. El sistema basa su entrenamiento en un resultado binario, en el cual un valor de 0 representará el fracaso en predecir un valor y el valor de 1, el éxito. De esta forma, para cada valor x de la data, se realizará la siguiente evaluación comparativa mediante la ecuación 6.

$$h(x) \begin{cases} 1, & \sum_{i=0}^{n} x_i \cdot \beta_1 > 0 \\ 0, & \sum_{i=0}^{n} x_i \cdot \beta_1 < 0 \end{cases} \tag{6}$$

Donde $h(x)$ es el valor obtenido a partir del modelo actual y que será comparado con $c(x)$ es el valor real que se recoge de la data y en base a si es positivo o negativo se le asigna el valor de 1 o cero, respectivamente. Posteriormente se modifica el valor de β_1 según la ecuación 7.

$$\boldsymbol{\beta_1} \leftarrow \boldsymbol{\beta_1} + \eta \cdot [c(x) - h(x)] \cdot \boldsymbol{x} \tag{7}$$

Además, la variable η se conoce como ritmo de aprendizaje y tomará un valor entre 0 y 1, definido arbitrariamente por el usuario [16].

1.3.2.3. Regresión logística

Una alternativa a la regresión lineal es el modelo de regresión logística, el cual parte del modelo lineal y agrega una función de probabilidad para convertirse en una manera especializada de clasificar valores. Este modelo trabajará mediante la composición de la función detallada en la ecuación 4 dentro de la función logística (Ec.8).

$$h(z) = \frac{1}{1 + e^{-z}} \tag{8}$$

Donde z será reemplazado por la función detallada en la ecuación 8, obteniendo así el siguiente modelo de la ecuación logística para clasificación se presenta en la ecuación 9.

$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}} \tag{9}$$

En este modelo se utiliza el mismo criterio y método de implementación revisado para la regresión lineal, con la variante de que, en lugar de obtener un modelo de regresión con el objetivo de predecir valores, se apunta a determinar el corte que podrá segmentar dos grupos de datos con el mayor éxito [22].

1.3.2.4. Máquinas de vectores de soporte (SVM)

Uno de los principales problemas de los algoritmos de aprendizaje automática es el sobreajuste de los datos. Los modelos revisados hasta el momento trabajan con matemática relativamente sencilla, en la cual se realizan los cálculos y evaluaciones para todos los elementos que componen la data de entrenamiento. Las máquinas de vectores de soporte (*Support Vector Machine* en inglés) se caracterizan por reducir la influencia de valores extremos presentes y reducir el grupo de datos considerados para el aprendizaje. De esta manera, reducen el efecto de variables que no son de interés mediante la aplicación de un modelo matemático más sofisticado [16].

El objetivo con el que se desarrollaron estas máquinas fue clasificar y separar elementos mediante la definición de un hiperplano que genere una división en el espacio vectorial donde los datos existen, como se observa en la figura 3, se genera un plano entre los puntos más cercanos de dos bloques de datos distintos. De manera similar con los modelos anteriores, los datos tienen la forma (x_i, y_i) en la cual x_i es a su vez un conjunto de variables $[x_{i1}, x_{i2}, x_{i3}, \dots x_{in}]$ que pertenecen al espacio R^n. Particularmente para el caso de las SVM, y_i es un elemento binario, es decir que solo podrá tomar uno

de dos posibles valores, ello con la finalidad justamente de clasificar los datos en dos categorías distintas [16].

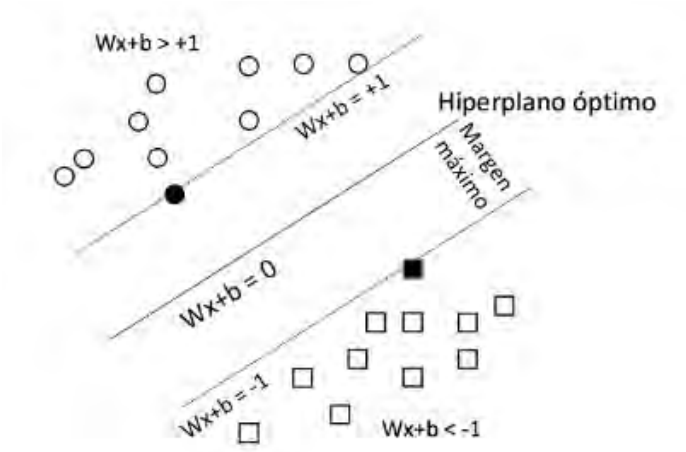


Figura 3. Ejemplo de hiperplano para R2 mediante SVM [16]

1.4. Redes neuronales artificiales

Una red neuronal artificial es un modelo computacional conformado por capas, las cuales a su vez se constituyen de arreglos de neuronas. El funcionamiento de las neuronas, la unidad básica de procesamiento de información en este modelo, está inspirado en el comportamiento biológico de las neuronas del cerebro humano. Las neuronas artificiales se encuentran presentes en tres tipos de capas: de entrada, de salida y ocultas [22]. El flujo de la información sigue una trayectoria similar a los impulsos eléctricos que cruzan se mueven en nuestro cerebro, partiendo de la señal eléctrica que proviene de sensores como los ojos o las papilas gustativas, la información es recibida del exterior por una capa de entrada. Posteriormente, la información recibida y procesada en la entrada ingresa a un complejo sistema de arreglos de neuronas, el cual es conocido como capa oculta; en esa zona, las neuronas interactúan entre ellas intercambiando información para finalmente converger en la capa de salida [16].

1.4.1. Perceptrón multicapa

El perceptrón multicapa es el modelo de red neuronal más simple, compuesto por una capa de entrada y una de salida, ambas formadas por una neurona respectivamente. En este modelo se puede encontrar el funcionamiento básico de las neuronas y su interacción con otras unidades de procesamiento básicas [23].

1.4.1.1. Neuronas

Las neuronas cumplen la función básica de procesamiento de la información. Es en las neuronas, donde de acuerdo con la función de transferencia que utilicen, la variable $\boldsymbol{X}$ es operada por la función $\boldsymbol{f_{transferencia}}$ para obtener el resultado $\boldsymbol{Y}$. En la figura 4 se aprecia una ilustración de una neurona artificial, para la cual los puntos marcados con una equis y fondo rojo representan los valores de ingreso o *"inputs"* que tiene este elemento.

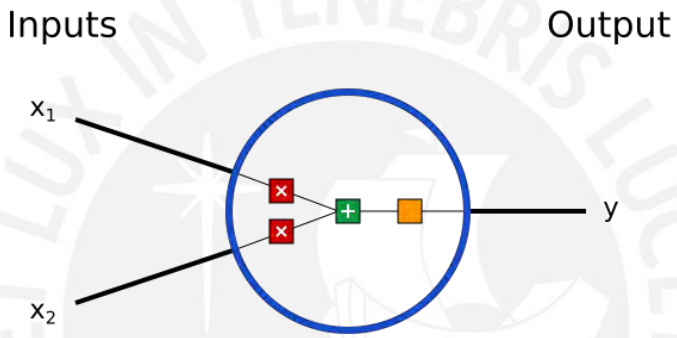


Figura 4. Neurona artificial. [24]

La cruz de fondo verde representa la función de transferencia y el cuadro amarillo corresponde al resultado o *"output"*. Además, se puede notar que la neurona recibe múltiples ingresos, lo que implica que el valor $\boldsymbol{X}$ se calcule como el promedio ponderado de estos valores.

$$\boldsymbol{Y} = f(\boldsymbol{\Sigma}) \tag{10}$$

En la ecuación 10 se presenta la función de transferencia, donde $\boldsymbol{Y}$ representa el output y $\boldsymbol{\Sigma}$, el promedio ponderado de los ingresos, cuya ponderación se realiza en función del peso asignado a las neuronas predecesoras, el cual puede ser un valor entre 0 y 1 [23]. En adición, para el universo de relaciones posibles para utilizarse como función de transferencia, la más comúnmente utilizada es la función *Sigmoid*, cuya estructura se presentó para algoritmos anteriores (Ec.8, 9) y se representa en la ecuación 11 [25].

$$f(\boldsymbol{\Sigma}) = \frac{1}{1 + e^{-\Sigma}} \tag{11}$$

El uso de la función *Sigmoid* se justifica en buscar reducir la linealidad del procesamiento de la información, lo cual también se puede lograr utilizando otras funciones populares como la tangente hiperbólica, ReLu y sus variaciones, así como Softmax y similares.

1.4.1.2. Propagación de la información

La comunicación de la información entre las neuronas se produce solo entre neuronas de diferentes capas y puede ser tanto desde el input hacia el output como en dirección opuesta. En la figura 5, cada círculo representa una neurona y se aprecia que las neuronas se comunican individualmente con cada uno de sus pares en la siguiente capa.

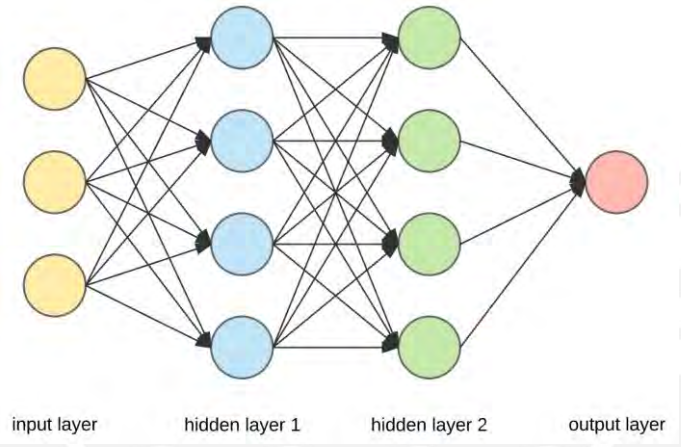


Figura 5. Red neuronal artificial formada por 2 capas ocultas. [26]

En el proceso de entrenamiento de una red neuronal la comunicación se dirige en ambos sentidos; en primer lugar, se avanza desde el ingreso a la salida con la finalidad de obtener un valor y posteriormente calcular el error [23]. Una vez determinado el error, la comunicación del error para la modificación de los pesos asignados a cada neurona se produce en el sentido inverso. Por otro lado, una vez entrenado un modelo, el flujo de la información se daría solo desde el ingreso hacia la salida con el motivo de determinar los valores objetivo.

1.4.2. Proceso de aprendizaje

El proceso de aprendizaje de una red neuronal artificial consiste en una secuencia de ensayo y error repetitiva por una cantidad de veces definida o hasta encontrar un error determinado. El proceso repetitivo buscará minimizar la diferencia entre el resultado de la red con los valores o esperados de un proceso o conjunto de datos. Consecuentemente, se puede interpretar de manera matemática como un

conjunto de vectores y matrices, donde ***i*** y ***o*** representan el "input" y "output" de la red neuronal (Ec.12) [25].

$$i = \begin{bmatrix} i_1 \\ i_2 \\ \vdots \\ i_j \end{bmatrix}, \qquad o = \begin{bmatrix} o_1 \\ o_2 \\ \vdots \\ o_j \end{bmatrix} \tag{12}$$

Además, el vector **W** (Ec.13) [25] representa el conjunto de pesos que relacionan los datos al ingreso con los resultados a la salida, y que es el objetivo a modificar durante el proceso de entrenamiento.

$$W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1J} \\ w_{21} & w_{22} & & w_{2J} \\ \vdots & \vdots & \ddots & \vdots \\ w_{K1} & w_{K2} & \cdots & w_{KJ} \end{bmatrix} \tag{13}$$

Asimismo, se presenta el operador Γ (Ec.14) [25] el cual es parte de la ecuación que determina el resultado de la red (Ec.15).

$$\Gamma = \begin{bmatrix} f(\cdot) & 0 & \cdots & 0 \\ 0 & f(\cdot) & & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & f(\cdot) \end{bmatrix} \tag{14}$$

El operador es una matriz diagonal cuyo único valor repetido a lo largo de la diagonal es la función de activación que se considera para la red en cuestión. Seguidamente, se observa cómo se articulan los vectores y matrices presentados en la ecuación 15 [25].

$$o = \Gamma[Wi] \tag{15}$$

Por otro lado, se denomina como ***d*** al vector que representa a los valores deseados para el vector ***o*** (Eq. 16) [25]. En general, se espera una diferencia entre ambos vectores; sin embargo, en el caso perfecto se tendría una red cuyos pesos sean los precisos para obtener un *output* equivalente a lo esperado. Este concepto se desarrolla con mayor profundidad en la sección 1.5 donde se revisan las máquinas de aprendizaje extremo.

$$d = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_j \end{bmatrix} \tag{16}$$

Seguidamente, se presenta la fórmula para determinar el error en la ecuación 17 [25]. La función del error servirá posteriormente para encontrar la variación individual de cada peso.

$$E = \frac{1}{2}\sum_{k=1}^{K}(d_k - o_k)^2 = \frac{1}{2}\|d - o\| \tag{17}$$

A partir del error $\boldsymbol{E}$ y una razón de aprendizaje (*learning rate*) $\boldsymbol{\eta}$, se determina el efecto individual que tendrá el error sobre cada peso de la matriz $\boldsymbol{W}$. En la ecuación 18.1 [25] se presenta la variación del error total en la dirección de los pesos. Dado que el error es una matriz, se optimización se realiza mediante una función gradiente, es por ello que se observa la presencia de derivadas parciales que reducen el error en la dirección de cada vector peso.

$$\Delta w_{kj} = -\eta \frac{\partial E}{\partial w_{kj}} \tag{18.1}$$

$$\Delta w_{kj} = w'_{kj} - w_{kj} \tag{18.2}$$

La variación de los pesos individuales se puede expresar también de acuerdo a la ecuación 18.2 [25] y consecuentemente se despeja el nuevo valor del peso w'_{kj} en función de los términos de la expresión 18.1. Finalmente, el nuevo valor de un peso se determinará mediante la ecuación 19 [25].

$$w'_{kj} = w_{kj} + \frac{1}{2}\eta(d_k - o_k)\left(1 - o_k^2\right)i_j \tag{19}$$

1.5. Máquina de aprendizaje extremo

El concepto de máquina de aprendizaje extremo (ELM, por sus siglas en inglés) es relativamente reciente en comparación con las redes neuronales artificiales. Guang-Bin Huang y su equipo de la escuela de Ingeniería Eléctrica y Electrónica de la Universidad Tecnológica de Nanyang lo presentaron en 2004 como un nuevo esquema de aprendizaje para las redes neuronales de propagación hacia adelante. La máquina de aprendizaje extremo consiste en una red neuronal conformada por una sola capa, la cual

cuenta con una gran cantidad de neuronas. En contraste con el proceso iterativo de prueba y error para el cálculo de pesos de las neuronas en una red neuronal artificial multicapa tradicional, para este sistema se utilizan ecuaciones de matemática matricial [27].

1.5.1. Concepto matemático de la ELM

La propuesta de Huang se basa en que se puede resolver una ecuación matricial que minimice el error del resultado de una red neuronal de una sola capa y cuyo resultado sean los pesos de las neuronas en función de la matriz de datos de entrada y de resultado esperado. Para la resolución de la ecuación se utiliza la matriz conocida como inversa generalizada de Moore-Penrose y para minimizar el error se aplica el concepto del módulo mínimo de los mínimos cuadrados.

1.5.1.1. Inversa generalizada de Moore-Penrose

A una matriz que cumpla con las condiciones propuestas por Penrose y Moore y que además sea la inversa generalizada de otra matriz se le conoce como inversa generalizada de Moore-Penrose. Las condiciones indicadas fueron descritas en la publicación de Penrose [28] y se presentan a continuación. Para una matriz **A** finita, sea rectangular o cuadrada, de elementos reales o complejos existe otra matriz única **X** que satisface cuatro criterios.

- AXA = A
- XAX = X
- (AX)* = AX
- (XA)*= XA

Esta matriz **X** también es referida como pseudo-inversa dado que emula a la inversa de una matriz cuadrada para las condiciones de una rectangular. La formulación de la matriz X se determina como el producto de dos matrices, la primera siendo la inversa del producto de la traspuesta de A con A y la segunda siendo la traspuesta de A y se calcula con la siguiente ecuación [29]:

$$X = pinv(A) = (A^t A)^{-1} A^t \tag{20}$$

El concepto de pseudo-inversa es utilizado en la formulación de algoritmos de aprendizaje automático en los que se busca una solución rápida, usualmente en procesos no estacionarios, que ofrezca simpleza y precisión [30].

1.5.1.2. Solución de mínimos cuadrados de sistemas lineales

La solución de mínimos cuadrados de mínima norma de sistemas lineales es un método de aproximación para resolver una ecuación de múltiples soluciones [29]. Para un sistema $\boldsymbol{Ax} = \boldsymbol{y}$ se define una función de error $\boldsymbol{\varepsilon}=\boldsymbol{Ax} - \boldsymbol{y}$, la cual puede ser minimizada buscando el valor mínimo de su módulo o norma como se puede ver en la ecuación 21 [27], donde x_o es el valor de interés a determinar y tanto $\boldsymbol{A}$ como $\boldsymbol{y}$ son matrices rectangulares.

$$\varepsilon_{min} = \|Ax - y\|_{min} = \|Ax_0 - y\| \tag{21}$$

Además, para resolver el valor del $\boldsymbol{x_0}$ considerando que la ecuación de mínimos cuadrados siempre admite al menos una solución y que esa solución es también la solución de las ecuaciones normales [29].

$$A^t A x_0 = A^t y \rightarrow x_0 = (A^t A)^{-1} A^t y \tag{22}$$

De manera que el problema se puede resolver mediante la aplicación de la inversa generalizada de Moore-Penrose [27], cuya expresión se muestra en la ecuación 22. Consecuentemente, los pesos que se modificaban de manera iterativa en la ecuación 19 son calculados mediante la resolución de la expresión 22.

1.6. Aspectos de programación

La programación de una red neuronal artificial se puede realizar desde cero en cualquier lenguaje; no obstante, existen librerías en varios lenguajes que simplifican el proceso. Particularmente, el lenguaje de programación Python, en su versión 3.6.9, incluye tres librerías que brindan soporte al desarrollo de algoritmos de aprendizaje automático, y particularmente la implementación de modelos basados en redes neuronales artificiales.

a. En primer lugar, la librería *Keras,* que es una interfaz de programación de aplicaciones (API, por sus siglas en inglés) orientada al aprendizaje profundo que trabaja en conjunto con la librería *TensorFlow*. Keras incluye bloques de capas, así como modelos completos para el aprendizaje automático, de manera que el usuario puede importar estos elementos de acuerdo con las necesidades u objetivos a desarrollar. Dentro de los parámetros a elegir se encuentra la función de activación, el número de capas, el tipo de modelo, el optimizador y las métricas a presentar [31].

b. En segundo lugar, *TensorFlow* es una librería que facilita la computación numérica y el procesamiento de grandes bloques de información para el aprendizaje automático. La participación de *TensorFlow* en los algoritmos de redes neuronales artificiales toma protagonismo en la etapa de entrenamiento del modelo, para la cual esta librería dispone la plataforma en la cual ocurrirá el aprendizaje. Asimismo, la librería incluye funciones que disponen el procesamiento de audio, video, y grandes cantidades de datos y se agiliza su funcionamiento al operarse en un entorno acelerado por una unidad de procesamiento tensorial [32].
c. En tercer lugar, *Scikit-Learn* es una librería con funciones similares a las desarrolladas por *TensorFlow* y *Keras* que está orientada a funciones a aplicaciones específicas del aprendizaje automático como la clasificación, regresión y agrupación de datos [33]. La funcionalidad de *Scikit-Learn* se encuentra en ofrecer una variedad de modelos específicos ya desarrollados, lo que brinda la facilidad al usuario de implementar algoritmos rápidamente, pero reduce la libertad para modificarlos de acuerdo con sus requerimientos particulares.

ET LUX IN TENEBRIS LUCET
MCMXVII

METODOLOGÍA

En este capítulo se presentará el proceso de entrenamiento de la red neuronal que posteriormente operará como un sistema de predicción de la resistencia al desgaste abrasivo de un cupón en función de la concentración de los elementos aleantes en un recargue duro. En primer lugar, se indicará el proceso de obtención de datos, su limpieza y ordenamiento. Seguidamente, se desarrollará la concepción de la red neuronal y sus características para poder aprender y encontrar las relaciones presentes entre la concentración de aleantes y la dureza con la resistencia al desgaste. Finalmente, se presentará el proceso de entrenamiento para esta red.

2.1. Obtención de datos

Un requerimiento importante para el desarrollo de modelos de aprendizaje automático es contar con una basta cantidad de datos, que a su vez sean de calidad. Para el desarrollo de la presente investigación se ha optado por el uso de los registros de ensayos de desgaste abrasivo según la norma ASTM G-65 de la American Welding Society (Tabla 1). Estos ensayos fueron realizados en cupones revestidos con recargues duros de elementos aleantes como cromo, níquel, zinc, silicio, vanadio, niobio, tungsteno, manganeso y aluminio. En los registros de la prueba se puede encontrar información del proceso de soldadura, la dureza en Rockwell C, la capa analizada y la pérdida de peso, el cual es el indicador que permitirá determinar la resistencia al desgaste.

Tabla 1. Ensayos de desgaste según norma ASTM G-65. [34]

Table 1 — ASTM G-65 Practice A Test Results: Dry Sand, Rubber Wheel, 6000 Revolutions, 30-lb Force

Test Code	Welding Process	Layer	Hardness, R_C	C	Mn	Si	Cr	Mo	V	Nb	W	Ni	Al	Average Weight Loss, g	Microstructure Type[a]
1	SAW	1		2.880			19.60	0.47	0.04					0.70	PA
2	SAW	2		3.170			20.90	0.51	0.04					0.55	NE
C	SAW	1		2.150	2.51	0.53	17.60	0.36						0.81	PA
W02	SAW	1	23	0.062	1.02	0.36	0.81	0.19				1.40		2.35	FB
W02	SAW	2	21	0.045	0.94	0.40	1.22	0.29				2.08		2.57	FB
W02	SAW	4	24	0.035	0.88	0.43	1.60	0.37				2.69		2.64	FB
W03	SAW	1	16	0.097	1.96	0.38	0.08	0.28				0.07		2.50	FB
W03	SAW	2	20	0.107	2.28	0.43	0.07	0.40				0.06		2.31	FB
W03	SAW	4	23	0.121	2.59	0.46	0.05	0.51				0.04		2.21	FB
W04	SAW	1	22	0.171	1.23	0.38	1.06	0.27				0.08		2.04	FB
W04	SAW	2	28	0.150	1.44	0.47	1.51	0.37				0.08		2.26	FB
W04	SAW	4	33	0.134	1.69	0.57	1.98	0.49				0.08		2.22	FB
W05	SAW	1	29	0.096	2.26	0.44	1.61	0.43						2.54	FB
W05	SAW	2	35	0.098	2.69	0.55	2.45	0.65						2.26	FB
W05	SAW	4	38	0.101	2.86	0.68	3.12	0.85						2.10	FB
W06	SAW	1	36	0.083	0.99	0.25	5.52					0.15		2.43	MS
W06	SAW	2	36	0.078	0.94	0.27	8.58					0.23		2.35	MS
W06	SAW	4	37	0.075	0.92	0.30	11.25					0.31		2.18	MS
W07	SAW	1	13	0.155	0.98	0.26	0.50	0.10				0.07		1.95	FB
W07	SAW	2	15	0.139	1.11	0.30	0.65	0.14				0.07		2.04	FB
W07	SAW	4	20	0.110	1.22	0.35	0.84	0.19				0.07		2.32	FB
W08	SAW	1	35	0.062	0.97	0.27	7.03	0.48						2.50	MS
W08	SAW	2	36	0.051	0.95	0.33	10.90	0.84						2.56	MS
W08	SAW	4	35	0.044	0.89	0.29	13.97	0.91						2.38	MS
W09	SAW	1	43	0.201	0.85	0.30	6.67							1.66	MS
[illegible]														1.78	MS

Para efectos de analizar la estandarización de la data, dado que ésta será comparada con ensayos realizados en otras instituciones, se realizó una limpieza y ordenamiento de la información. A través de tablas de conversión se calculó la dureza Vickers y de esta manera se obtiene la tabla 2, cuyos datos del ensayo serán la información sobre la cual se trabajará el sistema.

Tabla 2. Digitalización en Python con librería Pandas de datos de ensayo ASTM G-65 hechos por la AWS.

Process	Layer	C	Si	Mn	Ni	Cr	Mo	V	W	Nb	Al	Ti	Zr	Co	HV prom	MassLoss
SAW	1	0.062	0.36	1.02	1.40	0.81	0.19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	247	2.35
SAW	2	0.045	0.40	0.94	2.08	1.22	0.29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	235	2.57
SAW	4	0.035	0.43	0.88	2.69	1.60	0.37	0.0	0.0	0.0	0.0	0.0	0.0	0.0	252	2.64
SAW	1	0.097	0.38	1.96	0.07	0.08	0.28	0.0	0.0	0.0	0.0	0.0	0.0	0.0	201	2.50
SAW	2	0.107	0.43	2.28	0.06	0.07	0.40	0.0	0.0	0.0	0.0	0.0	0.0	0.0	227	2.31

Una vez que se tienen los datos ordenados, se procede a tomar una muestra equivalente al 80% de los datos totales. Esta muestra será utilizada para entrenar al sistema y el 20% restante servirá como datos para evaluación, con los cuales se aplicará el modelo predictivo y se comparará el resultado obtenido con el esperado. En la tabla 3 se presentan las medidas estadísticas de las variables a utilizarse.

Tabla 3. Descripción estadística de datos de ensayo según norma ASTM G-65 desarrollado por la AWS.

	count	mean	std	min	25%	50%	75%	max
C	96.0	1.017490	1.334464	0.035	0.15675	0.3805	1.5600	5.45
Si	96.0	0.525937	0.428838	0.000	0.27000	0.4200	0.6800	2.07
Mn	96.0	2.546604	3.730034	0.000	0.85750	1.0300	1.6225	15.58
Ni	96.0	0.365521	0.744572	0.000	0.00000	0.0000	0.2875	4.22
Cr	96.0	9.075833	8.066872	0.000	2.74250	6.7250	12.6125	31.80
Mo	96.0	0.947187	2.822857	0.000	0.13500	0.4750	0.8425	27.00
V	96.0	0.047708	0.139966	0.000	0.00000	0.0000	0.0125	0.80
W	96.0	0.048125	0.191472	0.000	0.00000	0.0000	0.0000	1.02
Nb	96.0	0.004792	0.023169	0.000	0.00000	0.0000	0.0000	0.18
Al	96.0	0.068125	0.252496	0.000	0.00000	0.0000	0.0000	1.43
Ti	96.0	0.019688	0.192897	0.000	0.00000	0.0000	0.0000	1.89
Zr	96.0	0.013229	0.129619	0.000	0.00000	0.0000	0.0000	1.27
Co	96.0	0.005729	0.056134	0.000	0.00000	0.0000	0.0000	0.55
HV prom	96.0	429.145833	149.025959	178.000	275.50000	438.0000	549.0000	746.00

2.1.1. Análisis de datos de entrenamiento

A continuación, se presentan los gráficos que relacionan el porcentaje del elemento con el desgaste producido en el ensayo. Es importante notar que estos gráficos permiten visibilizar la distribución de los datos recabados; sin embargo, no se pueden realizar conclusiones a partir de la influencia del aumento o decrecimiento de la presencia de uno de los elementos sobre la resistencia al desgaste dado que muchas variables cambian en cada prueba.

2.1.1.1. Carbono

En el caso del carbono se encuentra dos aspectos a destacar, los cuales se observan en la figura 6, en primer lugar, se identifica que la mayor dispersión de datos se presenta a partir del 1% de concentración, mientras que para concentraciones menores se puede presenciar un bloque de datos de baja dispersión. Se encuentra que para el bloque de probetas entre 0 y 1% de carbono, se presenta una pérdida de masa entre 1.25 y 2.75 gramos, mientras que a partir del 1% C se reduce la pérdida de masa en las probetas.

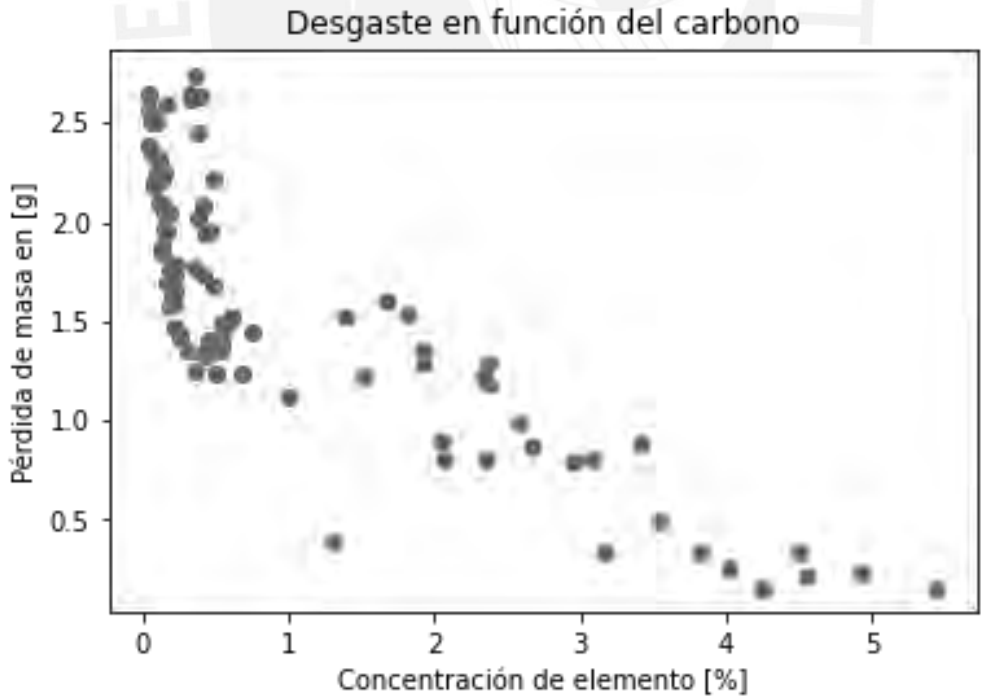


Figura 6. Distribución de la pérdida de masa en función de la concentración de carbono para ensayo ASTM G-65 realizado por AWS.

2.1.1.2. Cromo

En la figura 7 se presenta un gráfico con valores presentes con mucha variación y manera muy dispersa. Esto se valida con el cálculo de desviación estándar que para el cromo presentó el valor más elevado siendo este de 8.06 g de pérdida de masa en contraste con los valores menores a la unidad para los otros elementos. Se pueden visualizar valores de pérdida de masa entre en 1 y 2.5 g. Alrededor del 5% de cromo se aprecia un bloque de datos bastante sólido que obtuvo una pérdida de material localizada entre 1.0 y 1.5 gramos, por otro lado, para concentraciones por debajo de este valor, se aprecia un incremento del desgaste que aumenta a medida que el cromo disminuye. Sin embargo, a simple vista es difícil establecer una relación entre la concentración de este elemento y el desgaste abrasivo.

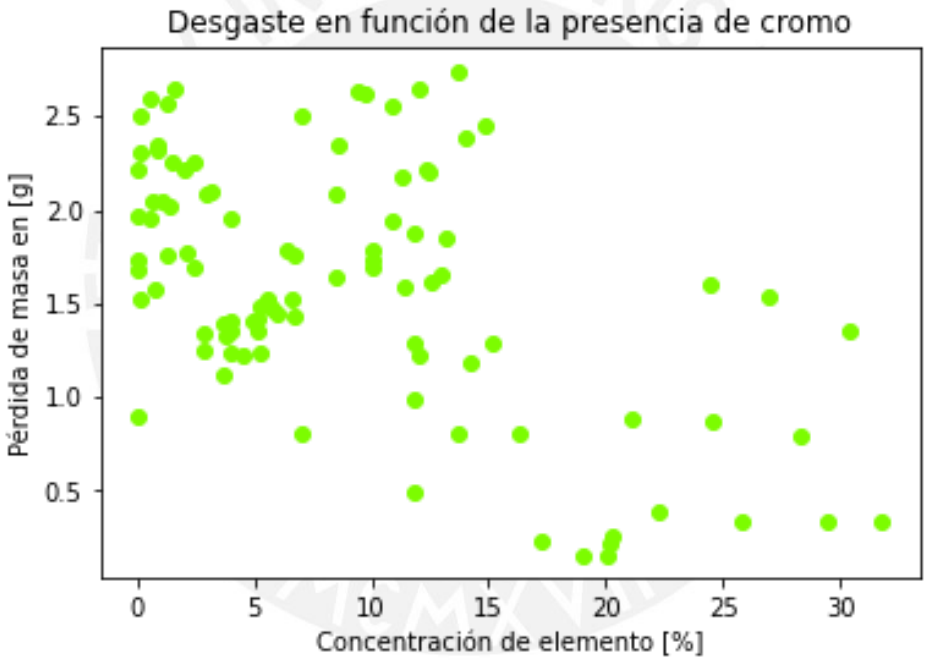


Figura 7. Distribución de la pérdida de masa en función de la concentración de cromo para ensayo ASTM G-65 realizado por AWS.

2.1.1.3. Silicio

En el caso del silicio (Figura 8) se encuentra que el rango de concentración que se trabaja es significativamente menor que la que se encuentra para los casos anteriores. Se puede notar la mayor presencia de datos entre el 0.25 y 0.75% obteniendo valores de pérdida de masa de hasta 2.6 gramos en los puntos intermedios. Se puede apreciar un decrecimiento de la pérdida de masa a medida que incrementa la concentración de este elemento aleante, por lo cual su presencia resulta beneficiosa para mejorar su resistencia al desgaste.

Además, se encuentra una relación bastante fuerte entre 0.25 y 0.50% de concentración, donde a pesar de algunos valores fuera del grupo, se observa una pérdida de masa mínima alrededor de 1.3 gramos para el ensayo realizado.

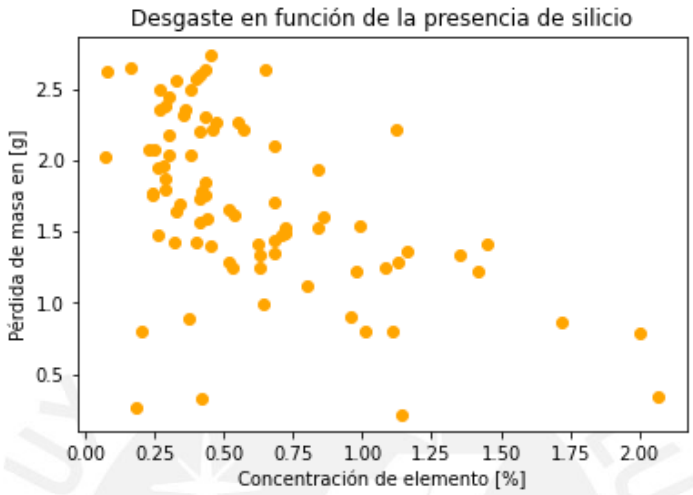


Figura 8. Distribución de la pérdida de masa en función de la concentración de silicio para ensayo ASTM G-65 realizado por AWS.

2.1.1.4. Manganeso

En el caso del manganeso se aprecia que se ha realizado una gran cantidad de pruebas entre 0 y 4% de concentración, cuyos resultados se presentan en la figura 9. Se identifica que a altos niveles de manganeso se obtienen altos valores de pérdida de masa lo cual una consecuencia negativa.

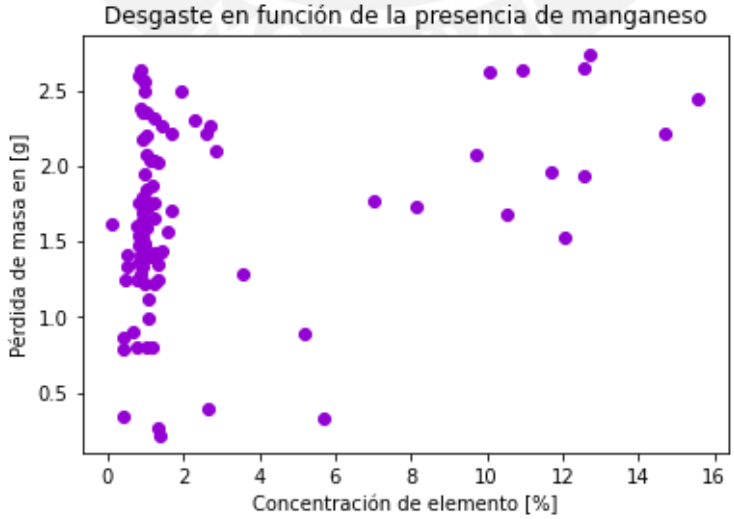


Figura 9. Distribución de la pérdida de masa en función de la concentración de carbono para ensayo ASTM G-65 realizado por AWS.

Además, en la figura 10 se muestra que alrededor del 0.5% de manganeso se presentaron valores bajos de pérdida de masa, lo cual representa un punto importante a considerar. Dado que gráficos de dispersión para los elementos aleantes que se han observado presentan valores centrados entre 1.5 y 2.5 g, es de interés encontrar que se pueden obtener valores más bajos de desgaste en ciertos niveles de concentración.

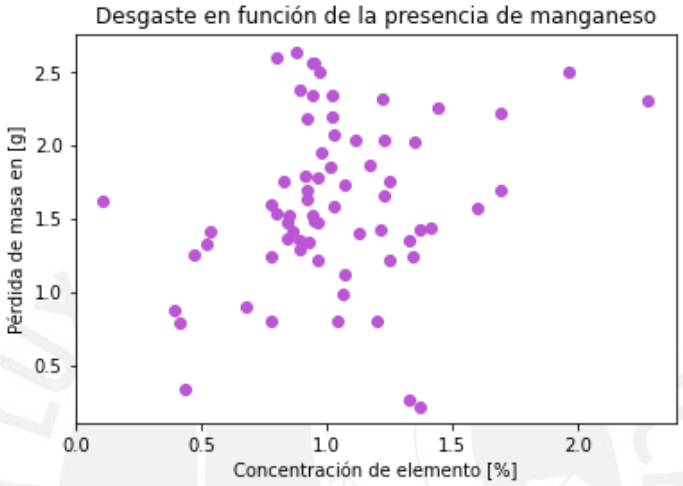


Figura 10. Distribución de la pérdida de masa en función de la concentración de manganeso para ensayo ASTM G-65 realizado por AWS.

2.1.1.5. Molibdeno

En la figura 11 se aprecia una variedad de valores dispersos entre 0 y 2% de concentración de molibdeno, obteniendo pérdidas de masa de hasta 2.7 gramos. Este conjunto de datos presenta una desviación estándar alta con un valor de 2.82 y en efecto se nota como los valores ocupan el espacio de manera uniforme.

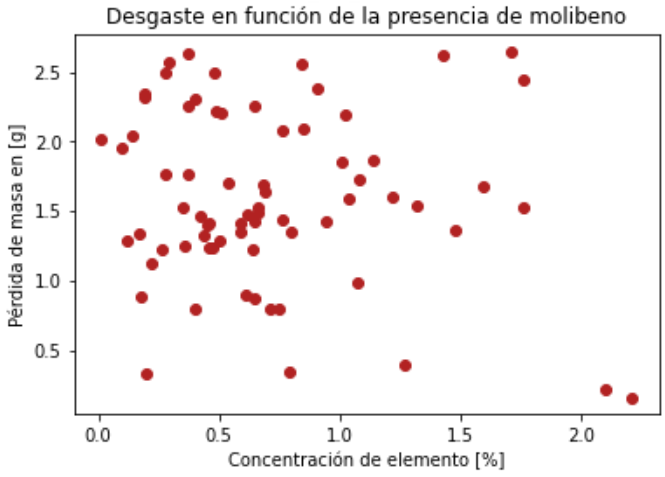


Figura 11. Distribución de la pérdida de masa en función de la concentración de molibdeno para ensayo ASTM G-65 realizado por AWS.

2.1.1.6. Níquel

En el caso del níquel (Figura 12) existe una menor cantidad de datos presentes en el gráfico, esto se debe a que en varios casos utilizaron una concentración del 0.0% y se optó por retirar esos valores del gráfico para apreciar la distribución para el níquel de mejor forma. Se observa una dispersión ubicada hasta valores de concentración del 2% y en muy pocos casos, valores superiores a esa cifra.

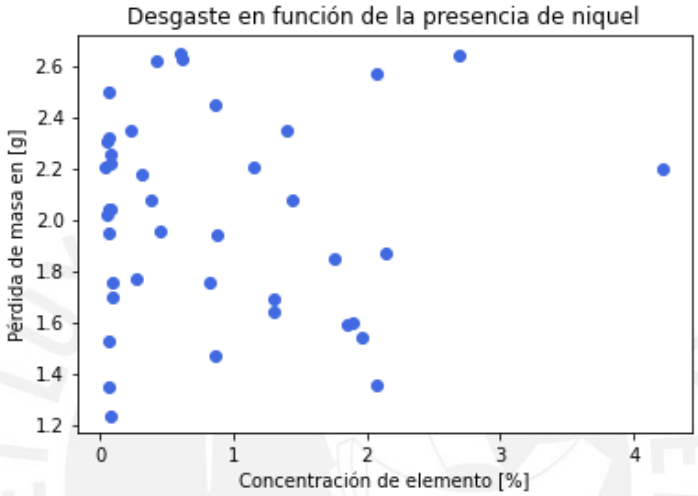


Figura 12. Distribución de la pérdida de masa en función de la concentración de níquel para ensayo ASTM G-65 realizado por AWS.

2.1.1.7. Vanadio

En la figura 13, el vanadio se presenta una muestra de menos datos con una agrupación de valores para concentración entre 0.0 y 1.0% con una pérdida de masa de 1.3 g en promedio.

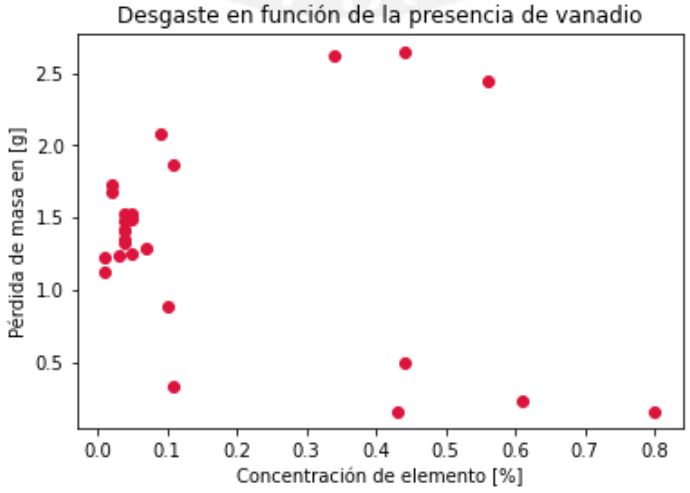


Figura 13. Distribución de la pérdida de masa en función de la concentración de vanadio para ensayo ASTM G-65 realizado por AWS.

2.1.1.8. Tungsteno

En el caso del tungsteno también se presenta una muestra reducida y dispersa para la cual no se aprecia relación aparente entre la concentración y la resistencia al desgaste (Figura 14).

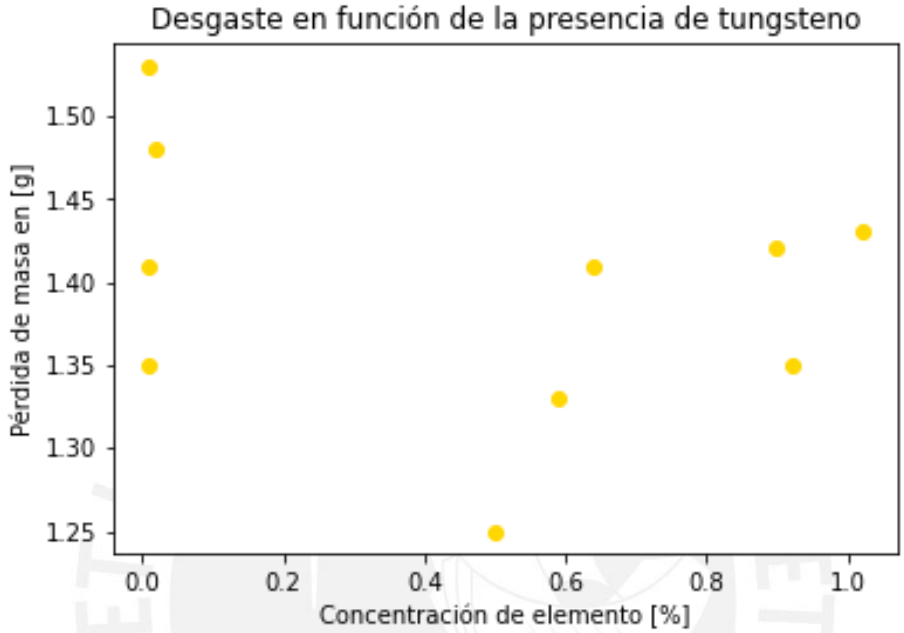


Figura 14. Distribución de la pérdida de masa en función de la concentración de tungsteno para ensayo ASTM G-65 realizado por AWS.

2.1.1.9. Otros elementos

Además de los elementos presentados en los gráficos, también se tienen datos de niobio, aluminio, cobalto, titanio y zirconio, para los cuales no se presentan gráficos. El motivo principal es que los datos recabados representan una muestra de pocos datos no nulos y se considera que la tabla de resumen de indicadores estadísticos brinda suficiente información sobre estos.

2.1.1.10. Dureza

En la figura 15 se presenta el gráfico para la dureza, en el cual se aprecia un comportamiento que se puede aproximar a una recta. La forma que toman los datos es la esperada, dado que, como se indicó en el marco teórico, la dureza tiene una relación con la resistencia al desgaste.

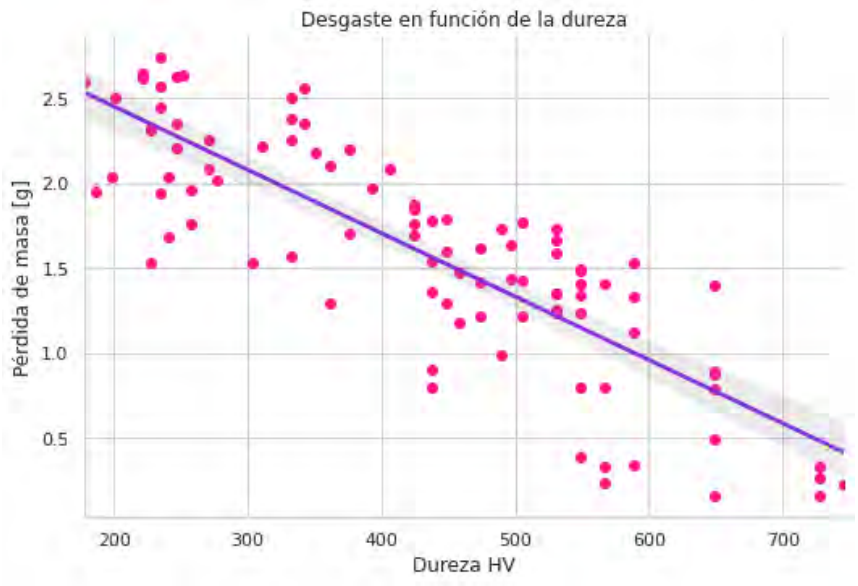


Figura 15. Distribución de la pérdida de masa en función de la dureza HV para ensayo ASTM G-65 realizado por AWS.

Para los datos considerados, la dureza promedio es de 429.15 HV con una desviación estándar de 149.02. Además, el coeficiente de correlación lineal es R= -0.837, lo cual confirma la tendencia lineal de los valores presentados.

2.1.1.11. Pérdida de masa

Finalmente, en la figura 16 se presenta la distribución de la pérdida de masa en los ensayos realizados. La mayoría de los valores se agrupa a partir de 1.25 gramos, con un valor máximo de 2.74 gramos de pérdida de material. Asimismo, se encuentra una reducción de masa promedio de 1.59 g con una desviación estándar de 0.664 g.

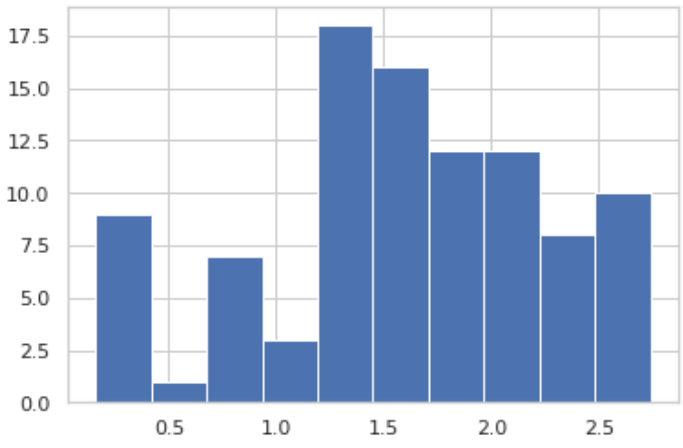


Figura 16. Histograma de la pérdida de masa para ensayo ASTM G-65 realizado por AWS.

2.2. Desarrollo de algoritmos de aprendizaje automático

2.2.1. K-vecinos cercanos (KNN)

El modelo de regresión mediante el algoritmo de k-vecinos cercanos (KNN) es presentado en la figura 17. Este algoritmo se diferencia por su simplicidad, debido a ello es que se realiza un proceso iterativo en el cual se propone una cantidad de vecinos a considerarse y en función de los resultados se redefine la cantidad a utilizar.

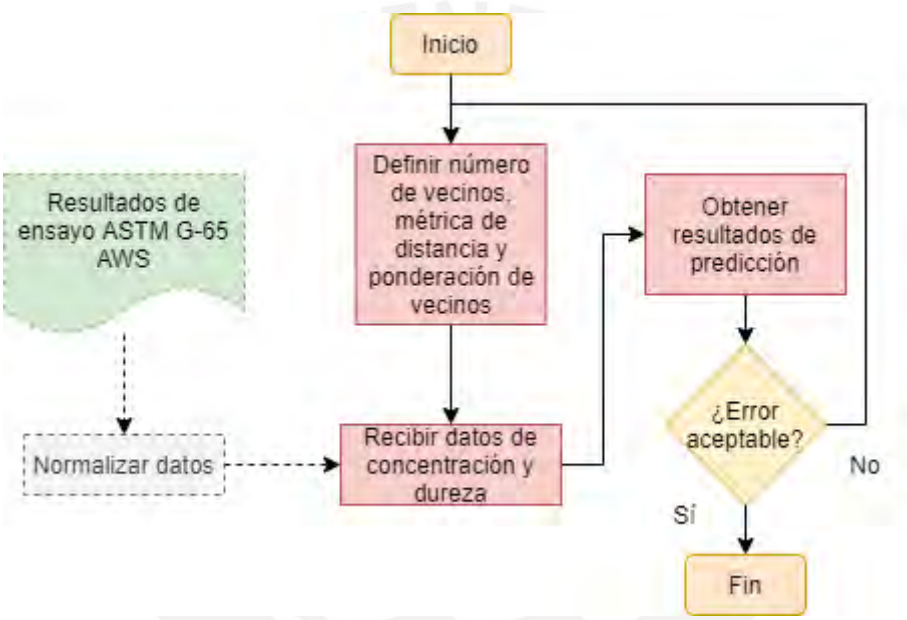


Figura 17. Diagrama de flujo de elaboración de algoritmo de k-vecinos cercanos.

Con respecto a la métrica de distancia se optó por la distancia de Minkowski (Ec.3) con valor de p=2, por lo cual se tendría como métrica la distancia Euclidiana (Ec.2). Para los k vecinos a considerarse se optará por una ponderación uniforme de las distancias, es decir, que se tomará la influencia de cada vecino elegido como equivalente en valor independientemente de la cercanía al punto de interés. Asimismo, se utilizará 80% de los datos para el entrenamiento y el 20% restante será utilizado para la validación.

Como parte final del desarrollo del modelo KNN se evaluará el comportamiento gráfico de las predicciones realizadas. Dado que la tendencia de las predicciones de los algoritmos de vecinos cercanos es acercar los resultados al valor promedio de entrenamiento a medida que incrementa el número de vecinos, no bastará con considerar los valores de las métricas de error, también se analizará cómo se comporta de manera gráfica.

2.2.2. Red neuronal artificial (ANN)

Para la implementación del algoritmo de red neuronal artificial se iniciará con definir el flujo que seguirán los datos analizados en el punto 2.1 (Figura 18). Seguidamente se desarrollará cada paso y cómo los datos son procesados y relacionados entre sí. Asimismo, se detallarán las características que tomará la red y demás aspectos importantes a considerar.

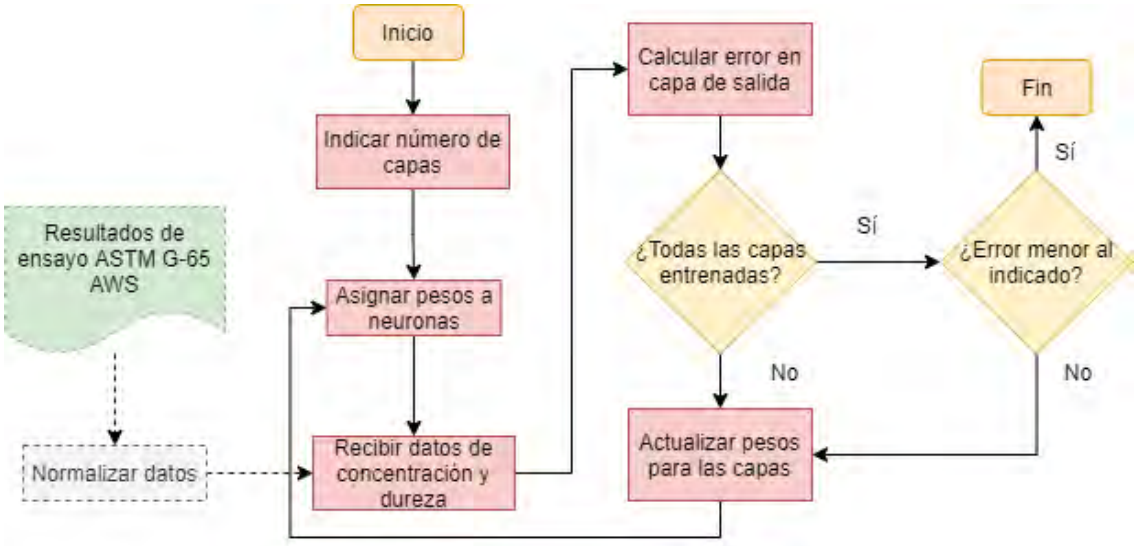


Figura 18. Diagrama de flujo de elaboración de algoritmo de red neuronal artificial.

2.2.2.1. Determinar número de capas

Para la implementación de la ANN se requiere una capa para el ingreso de datos, otra para la salida y como mínimo una capa oculta. En este caso, se realizarán un sistema de 3 capas, el cual se esquematiza en la figura 19.

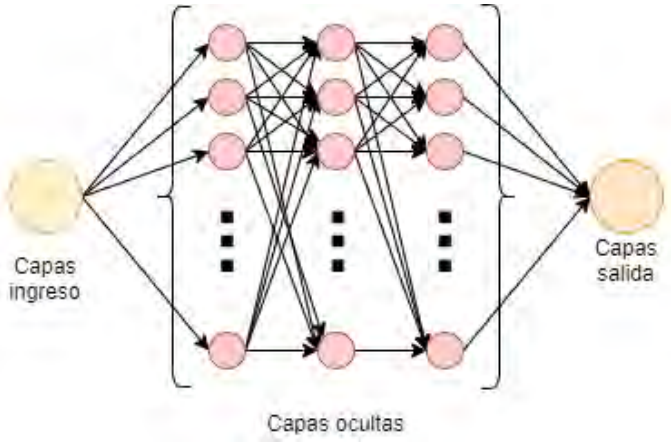


Figura 19. Red neuronal artificial a implementarse.

En este caso se está buscando un modelo que se adecue rápidamente a los datos recibidos, por lo cual se trabajará con 64 neuronas en cada una de las capas ocultas. De esta manera, se reduce la linealidad del modelo, lo cual responde al comportamiento visto en los gráficos de la sección anterior. En la tabla 4 se observa que, como resultado de la red propuesta, se obtienen 13,569 parámetros entrenables, los cuales serán modificados dependiendo de la función de activación que modifique sus valores y el resultado de error que reciban en cada iteración.

```
Model: "entrenar_desgaste"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
ingreso (Dense)              (None, 64)                1024
_________________________________________________________________
oculta_1 (Dense)             (None, 64)                4160
_________________________________________________________________
oculta_2 (Dense)             (None, 64)                4160
_________________________________________________________________
oculta_3 (Dense)             (None, 64)                4160
_________________________________________________________________
salida (Dense)               (None, 1)                 65
=================================================================
Total params: 13,569
Trainable params: 13,569
Non-trainable params: 0
```

Tabla 4. Parámetros a entrenarse de la red neuronal artificial.

2.2.2.2. Normalización de los datos

El proceso de normalización tiene la finalidad de igualar los efectos que tendrá cada aspecto a evaluarse (las concentraciones y dureza). Anteriormente se pudo apreciar que la dureza se ubica en valores entre 178 y 746, mientras que algunas concentraciones van por debajo de 0.5%. Normalizar los valores permite que independientemente de la magnitud, sean llevadas a una escala entre -1 y 1, de lo contario, la dureza sería considerada en mayor nivel que las concentraciones y el análisis multivariable perdería el efecto de la mayoría de los valores. En la tabla 5 se presentan parte de los datos normalizados.

C	Si	Mn	Ni	Cr	Mo	V	W	Nb	Al	Ti	Zr	Co	HV prom
0.271652	0.732356	-0.454850	-0.490914	-0.306914	-0.211554	-0.340857	-0.251342	-0.206815	0.957935	-0.102062	-0.102062	-0.102062	-0.846469
0.683803	1.478559	-0.457530	2.302638	2.643424	0.188749	-0.340857	-0.251342	-0.206815	-0.269806	-0.102062	-0.102062	-0.102062	0.059414
1.238333	2.784412	-0.578173	-0.490914	1.924434	-0.105279	-0.340857	-0.251342	-0.206815	-0.269806	-0.102062	-0.102062	-0.102062	1.475274
-0.662056	0.102749	-0.229650	-0.383470	-0.879626	-0.161959	-0.340857	-0.251342	-0.206815	-0.269806	-0.102062	-0.102062	-0.102062	-0.792787
1.605521	3.600571	-0.567449	-0.490914	2.531857	-0.055684	-0.340857	-0.251342	-0.206815	-0.269806	-0.102062	-0.102062	-0.102062	1.072660

Tabla 5.Datos AWS normalizados para entrenamiento de ANN.

La asignación de pesos la realiza el sistema de manera aleatoria, esta etapa sirve para confirmar que se está creando un modelo de red neuronal artificial, independientemente de su capacidad para efectuar una regresión con éxito.

2.2.2.3. Entrenamiento de red neuronal

Tras realizar los pasos anteriores se inicia el entrenamiento en sí, el cual depende de las variables de épocas, denominadas "epochs" en inglés, y la razón de validación. Las épocas determinan la cantidad de veces que los datos pasan a través de la red neuronal desde el punto de input hasta la salida y la razón de validación indica el porcentaje de datos de entrenamiento utilizados para calcular el error obtenido por época.

En este caso, se realiza un entrenamiento en función del número de épocas que minimice el error en el proceso de regresión, por lo cual, este proceso es iterativo y se guarda un registro histórico de la evolución del modelo en función del pasar de las épocas. Se toma un valor arbitrario de épocas y se evalúa el comportamiento del error, no solo considerando su valor sino también su crecimiento o decrecimiento con el pasar de las épocas. El error absoluto al que se apuntó fue un valor cercano a 0.01, dado que, para el rango de valores de desgaste, este representase un error porcentual bajo.

2.2.2.4. Evaluación con nuevos datos

Posterior al entrenamiento de la red, se procede a ingresar datos que el modelo no ha visto antes y obtener un valor de pérdida de masa para éstos. Es importante notar que estos datos también deben estar normalizados, dado que es en base a valores normalizados que se ha entrenado al sistema. Los datos que se utilizarán son el 20% separado de los datos iniciales para este procedimiento. Esta fase es el final de la elaboración del modelo, la cual permite encontrar la diferencia entre los valores reales y los obtenidos por el modelo; sin embargo, es a partir de la satisfacción con el rendimiento del modelo obtenido que se podrá proponer el cambio de algunos indicadores o valores que construyen al modelo para obtener uno que sí cumpla con las expectativas. Por ello, el procedimiento, como se describió anteriormente, es iterativo y es el usuario final quien define el cumplimiento del objetivo planteado.

2.2.3. Máquina de aprendizaje extremo (ELM)

En el caso del algoritmo para la implementación de una máquina de aprendizaje extremo se procederá de manera análoga al algoritmo de red neuronal artificial (ANN) en lo referente al ordenamiento y normalización de los datos a utilizarse en el entrenamiento. Este algoritmo requiere del uso de sólo una capa oculta, cuya construcción también se desarrollará de acuerdo con lo presentado en el diagrama de flujo de la figura 20.

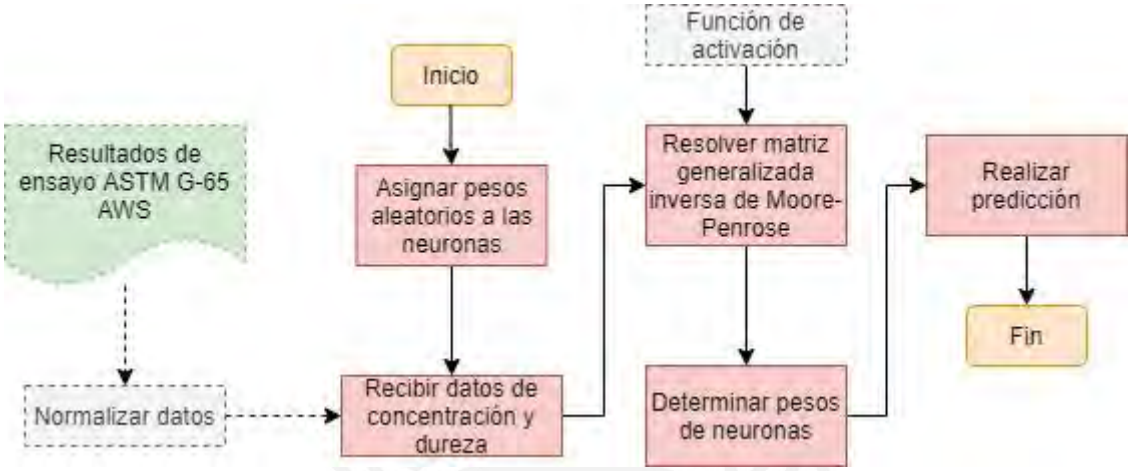


Figura 20.Diagrama de flujo de elaboración de algoritmo de máquina de aprendizaje extremo.

En este sistema, la variable de control para reducir el error es el número de neuronas que conformarán la capa oculta de esta red. En consistencia con la definición, que establece que para un número infinito de neuronas se obtendrá un modelo que se ajuste perfectamente a los datos de ingreso, se busca obtener un modelo con la cantidad de neuronas suficientes para predecir el desgaste con un error aceptable y que no tenga un elevado consumo de recursos computacionales. La figura 21 presenta un esquema simplificado de la máquina de aprendizaje extremo.

Figura 21. Red neuronal para entrenamiento mediante aprendizaje extremo.

En contraste con el algoritmo de ANN, en el cual la historia del entrenamiento permitía visualizar la evolución de los valores de error, en este caso se contará con un registro de la reducción del error en función del número de neuronas presentes en la capa oculta. Para el ambiente virtual en el cual se

desarrolla el modelo se cuenta con 107 GB de memoria de disco y 12.69 GB de memoria de acceso aleatorio (RAM), los cuales serán limitantes para la cantidad de neuronas a incluir.

2.2.3.1. Normalización de datos

El proceso de normalización de los datos se realiza de igual manera que para el entrenamiento de la red neuronal artificial, por lo cual se define la misma función que toma el valor de cada columna para ser restado con su correspondiente media y dividido por su respectiva desviación estándar.

2.2.3.2. Estructuración de capa oculta

Se realizarán varios modelos donde se probarán capas ocultas de 10, 100, 1000, 10000, 100000, 500000 y 1000000 neuronas. Para la asignación de valores aleatorios iniciales se utilizará la función "*random*" de la librería *numpy* de Python. De esta manera, se tendrán modelos de "n" neuronas en su capa oculta, todas con un peso a la entrada asignado aleatoriamente. Los valores de los sesgos también serán asignados siguiendo el sistema de asignación aleatoria de los pesos.

Se inicia con la obtención de tres matrices, la primera es el arreglo de los datos de entrenamiento normalizados, la cual es independiente al número de neuronas o la estructura de la red. La segunda y tercera corresponden a los pesos y sesgos de las neuronas, respectivamente, y sus valores serán asignados de manera aleatorio. Además, la matriz de pesos $\boldsymbol{w}_{\boldsymbol{aleatorio}}$ tendrá dimensiones variables, dependiendo de la cantidad *N* de neuronas que se consideren.

$$X_{entrenamiento} = \begin{bmatrix} 0.271395 & \cdots & -0.811872 \\ \vdots & \ddots & \vdots \\ 1.882991 & \cdots & 1.488269 \end{bmatrix}_{11\times 95}$$

$$w_{aleatorio} = \begin{bmatrix} w_{1-1} & \cdots & w_{1-N} \\ \vdots & \ddots & \vdots \\ w_{1-11} & \cdots & w_{11-N} \end{bmatrix}_{11\times N}$$

$$b_{aleatorio} = \begin{bmatrix} b_1 \\ \cdots \\ b_N \end{bmatrix}_{11\times 1}$$

Se define la matriz $\boldsymbol{G}$ como el resultado de las operaciones entre las tres matrices definidas anteriormente. $\boldsymbol{G}$ es equivalente al producto de las matrices de valores de entrenamiento y pesos más el sesgo.

$$G = X_{entrenamiento} \cdot w_{aleatorio} + b_{aleatorio}$$

$$G = \begin{bmatrix} 0.271395 & \cdots & -0.811872 \\ \vdots & \ddots & \vdots \\ 1.882991 & \cdots & 1.488269 \end{bmatrix}_{11\times 95} \cdot \begin{bmatrix} w_{1-1} & \cdots & w_{1-N} \\ \vdots & \ddots & \vdots \\ w_{1-11} & \cdots & w_{11-N} \end{bmatrix}_{11\times N} + \begin{bmatrix} b_1 \\ \dots \\ b_N \end{bmatrix}_{11\times 1}$$

Seguidamente, se calcula la matriz ***H***, la cual será operada para obtener su inversa generalizada posteriormente. Previo a ello, los valores de ***G*** serán argumento de la función *ReLU* y de esa manera se generará la matriz ***H***.

$$H = ReLU(G)$$

Finalmente, se realiza el producto de la inversa de Moore-Penrose de ***H*** y la matriz que agrupa a los valores de desgaste obtenidos mediante ensayo por AWS.

$$Y_{entrenamiento} = \begin{bmatrix} 2.35 \\ \dots \\ 1.35 \end{bmatrix}_{95\times 1}$$

$$w_{salida} = H^{+} \cdot Y_{entrenamiento}$$

Con ello, se podrá contar con una capa oculta de neuronas con los valores para sus pesos al ingreso y a la salida definidos, a partir de la cual se procederá a la aplicación de la misma y la predicción de valores.

2.2.3.3. Predicción de valores de desgaste

Se ingresan los datos del ensayo de AWS que no fueron utilizados para la formulación de la matriz y se evalúa la cercanía entre los valores reales y los predecidos. Asimismo, se realiza el proceso de comparar si el modelo predice con eficacia los resultados del ensayo realizado en el Laboratorio de Materiales PUCP. La comparación de las predicciones para ambas muestras, así como los valores de error son presentados en la sección de resultados.

$$Predicción = \boldsymbol{H} \cdot w_{salida}$$

RESULTADOS

En esta sección se presentarán los tres modelos obtenidos para determinar la pérdida de masa en función de la composición y dureza obtenidos en un ensayo. En función de los resultados del entrenamiento se seleccionará al modelo con mejor rendimiento de cada algoritmo y posteriormente se compararán los modelos para determinar que algoritmo estima con mayor exactitud los valores de desgaste. Esta evaluación consiste en medir la capacidad del modelo para determinar el resultado de desgaste de ensayos realizados por la AWS, pero que no formaron parte del bloque de datos de entrenamiento, lo que corresponde a un 20% de los datos tomados (24 probetas).

A partir de los modelos, se evaluará su rendimiento, el cual se mide en función del error absoluto (MAE) y la raíz cuadrada del error cuadrado medio (RMSE). Asimismo, se considerará la gráfica de dispersión que contrasta valores reales con predecidos y su evolución en relación con la variación de sus parámetros como lo son neuronas o número de vecinos, dependiendo del caso.

Posteriormente, se utilizarán los modelos exitosos para estimar las mediciones ensayadas en la PUCP y validar así la capacidad de predecir tanto valores pertenecientes al mismo bloque de ensayo como realizados en un laboratorio adicional al de la AWS. De esta forma, se obtiene la seguridad de que el modelo podrá encontrar valores de desgaste para futuros ensayos ASTM G-65 independientemente del laboratorio en el cual estos fuesen realizados.

3.1. Modelos obtenidos

Se han realizado 3 tipos de algoritmos para la predicción del desgaste, para los cuales se medirá el éxito que tuvieron sus resultados en función de la cercanía a las cifras reales. Para los algoritmos de vecinos cercanos y máquina de aprendizaje extremo se propusieron múltiples modelos en los cuales se variaba el número de vecinos y la cantidad de neuronas, respectivamente. Por otro lado, en el algoritmo de red neuronal artificial se presentará el registro de la evolución de las métricas para el modelo desarrollado.

3.1.1. Modelo de k-vecinos cercanos (KNN)

En función de los números de vecinos k elegidos, se obtuvieron la misma cantidad de modelos de manera que se pueda comparar cómo el número de vecinos influencia la capacidad del modelo para predecir el desgaste. El análisis consistirá tanto en una evaluación cualitativa de los gráficos de dispersión que miden la relación entre los valores predecidos y los reales, así como una evaluación cualitativa de las medidas de error en la predicción realizada.

3.1.1.1. Comparativo entre valores predecidos y reales por modelo

Se presentan 19 modelos de regresión mediante k-vecinos cercanos en los cuales se varió el número de vecinos a considerarse partiendo desde 1 hasta 95. Se cubrieron los números de vecinos desde 1 hasta 10 y posteriormente se plantearon los modelos variando el número de vecinos de 10 en 10. De acuerdo con la estructura del algoritmo, a medida que aumente el número de vecinos se espera que todos los valores predecidos se alinien alrededor del valor promedio de los mismos.

En los modelos de 1 a 5 vecinos (Figura 22) se observa la dispersión de los valores estimados en el rango de 0.0 a 3.0 gramos de pérdida de material. A partir de ello, se valida que el algoritmo ubica a todos los valores dentro del rango propio de los resultados del ensayo real, por lo cual se puede afirmar que el modelo brinda una primera aproximación inicial exitosa.

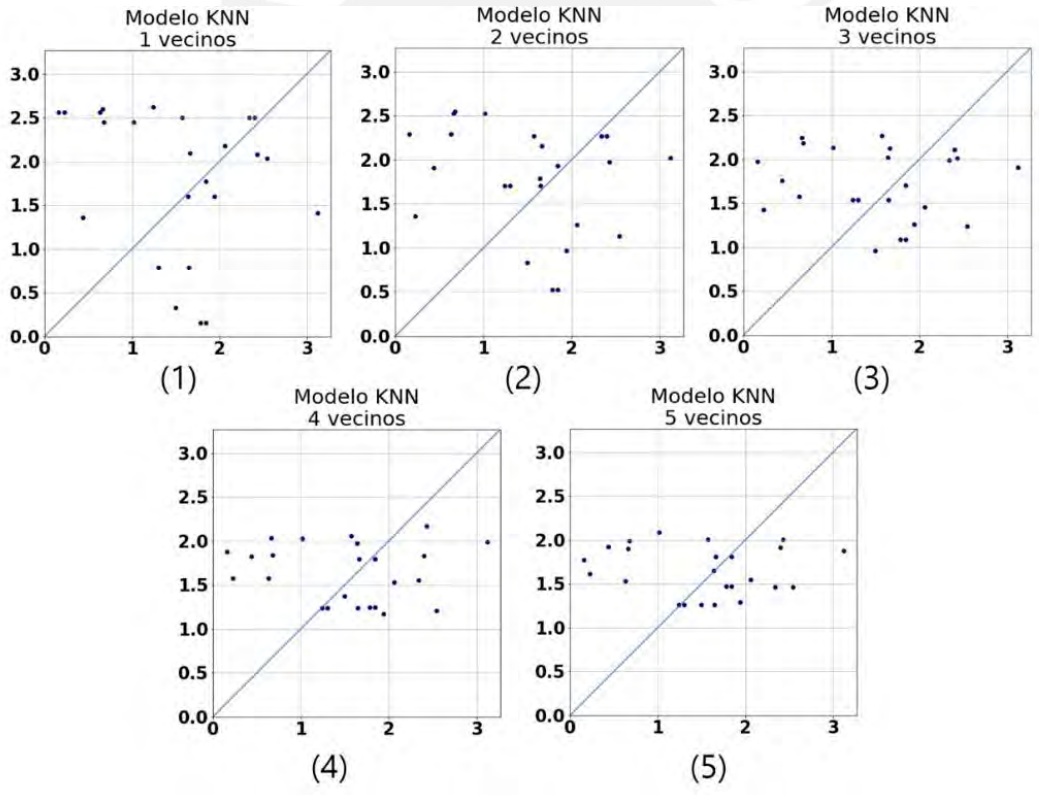


Figura 22. Gráficos de dispersión entre valor real y predicho para k-vecinos cercanos. (1) 1 vecino, (2) 2 vecinos, (3) 3 vecinos, (4) 4 vecinos, (5) 5 vecinos.

En el segundo grupo de modelos KNN analizado (Figura 23), se observa que los resultados se ubican entre 1.0 y 2.5 gramos para el modelo de 6 vecinos. En adición, el hecho de que, a medida que la cantidad de vecinos se incrementa, los valores predecidos converjan hacia un rango de 1.5 a 2.0 gramos muestra la tendencia a acercarse al valor promedio.

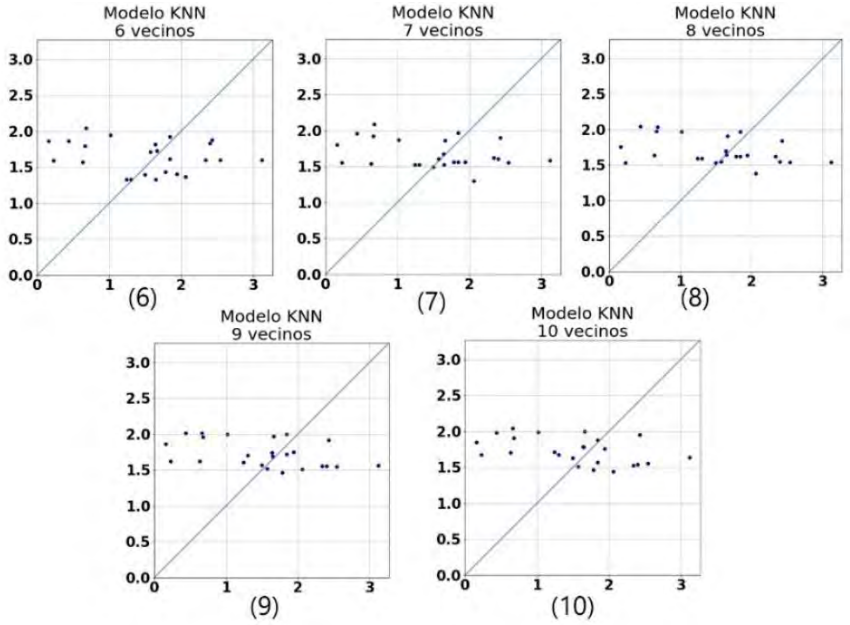


Figura 23. Gráficos de dispersión entre valor real y predicho para k-vecinos cercanos. (6) 6 vecinos, (7) 7 vecinos, (8) 8 vecinos, (9) 9 vecinos, (10) 10 vecinos.

Posteriormente, al analizar el comportamiento de los modelos propuestos a partir de los 20 vecinos e incrementando de 10 en 10 (Figura 24) se hace evidente la tendencia a una recta horizontal alrededor del valor promedio de desgaste de los datos de entrenamiento. A partir de este punto se observa que el modelo pierde la capacidad para estimar el desgaste de manera eficaz.

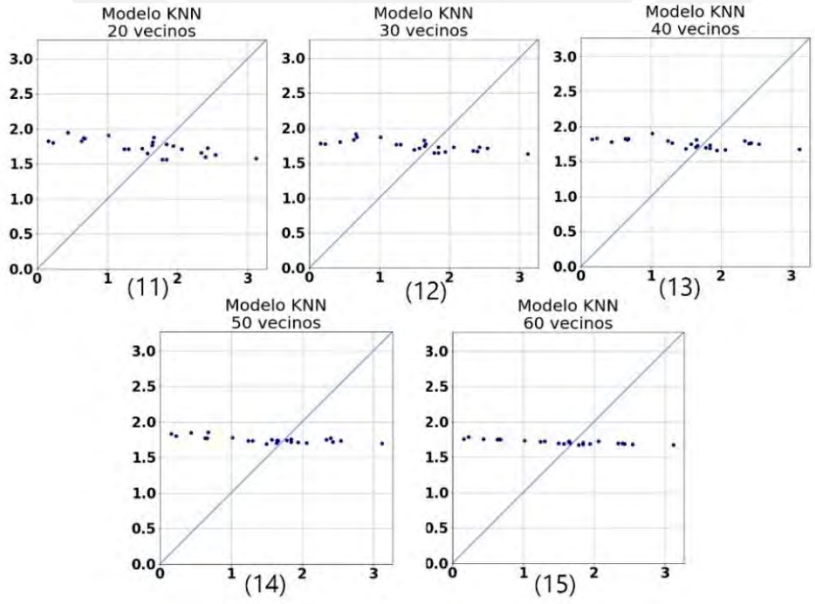


Figura 24. Gráficos de dispersión entre valor real y predicho para k-vecinos cercanos. (11) 20 vecinos, (12) 30 vecinos, (13) 40 vecinos, (14) 50 vecinos, (15) 60 vecinos.

Finalmente, en la figura 25 se observa que a partir de los 70 vecinos no se presenta una variación significativa entre las medidas obtenidas y se evidencia que, independientemente de las concentraciones de los elementos químicos medidos y la dureza, el modelo brindará como respuesta una cifra constante.

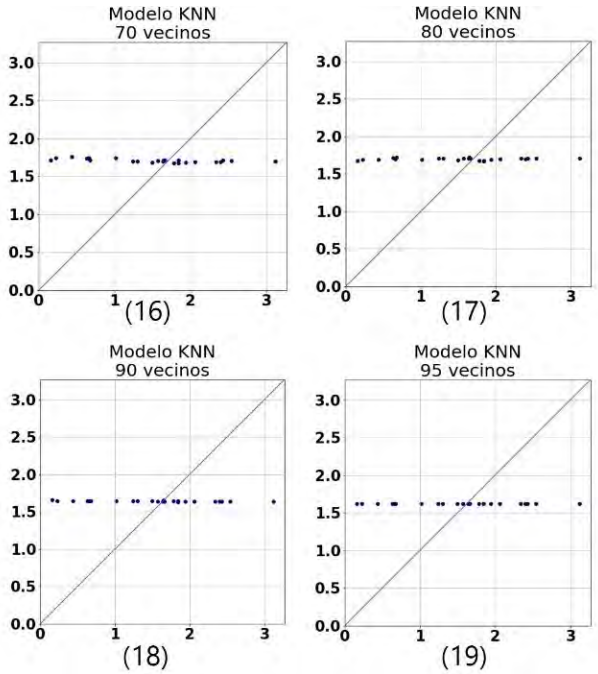


Figura 25. Gráficos de dispersión entre valor real y predicho para k-vecinos cercanos. (16) 70 vecinos, (17) 80 vecinos, (18) 90 vecinos, (19) 95 vecinos.

3.1.3.2. Evolución de modelos en función de la cantidad de vecinos

En la tabla 6 se presentan las medidas de error medio absoluto (MAE) y la raíz del error cuadrado medio (RMSE), ambas medidas en gramos.

Tabla 6. Variación de MAE y RMSE en función de cantidad de vecinos de modelo KNN.

N	MAE (g)	RMSE (g)
1	1.029	1.274
2	0.923	1.115
3	0.778	0.917
4	0.71	0.865
5	0.678	0.845
6	0.666	0.846
7	0.673	0.859
8	0.682	0.873
9	0.694	0.881
10	0.71	0.885

N	MAE (g)	RMSE (g)
20	0.693	0.871
30	0.676	0.849
40	0.657	0.835
50	0.64	0.827
60	0.636	0.811
70	0.626	0.798
80	0.616	0.78
90	0.609	0.775
95	0.607	0.768

Las estadísticas presentadas en la tabla 6 resumen el error de las predicciones en función de la cantidad de vecinos considerados en la regresión. Los resultados obtenidos son coherentes con lo presentado en las figuras de la 22 a la 25, en las cuales las predicciones tienen a agruparse alrededor del valor promedio de desgaste para la muestra.

Además, en la figura 26 se observa un decrecimiento en el error a medida que se aumenta la cantidad de vecinos desde 1 hasta 6. A partir de los 7 vecinos, las métricas de error aumentan y a partir de los 10 vuelven a reducirse. En adición, es importante notar que, si bien en los modelos por encima de 10 vecinos se obtuvieron menores valores de error, el modelo presentó valores alrededor del promedio a medida que aumentaron los vecinos y se perdió la cercanía a la recta de igualdad.

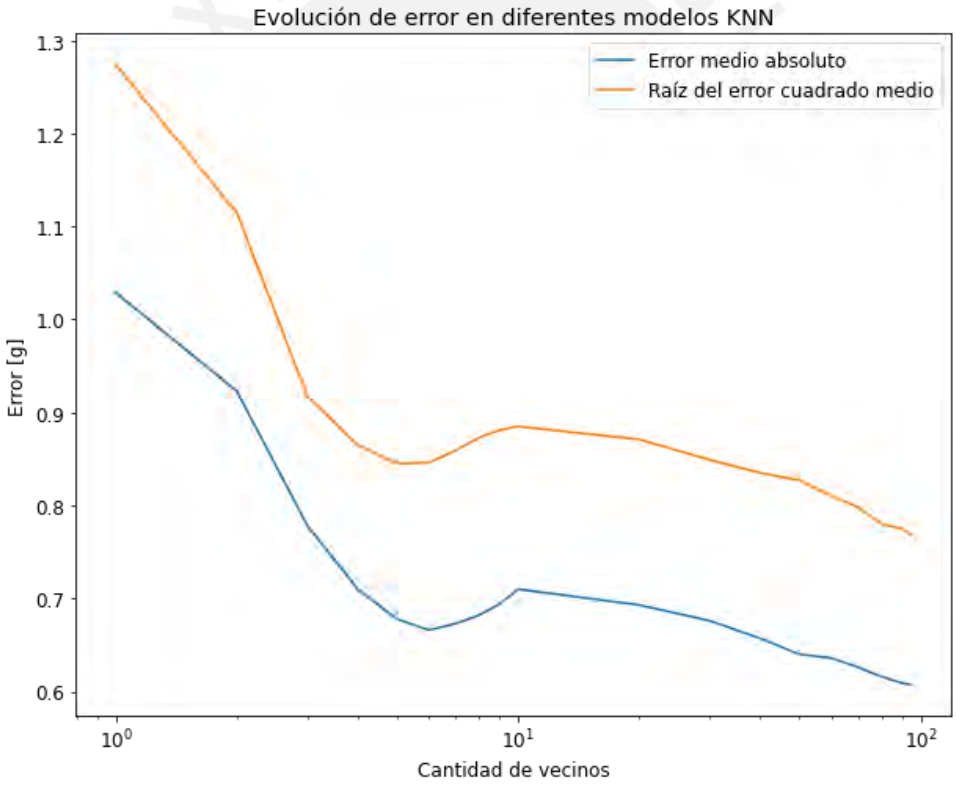


Figura 26. Evolución del MAE y el RMSE en función de la cantidad de vecinos en el algoritmo KNN. Escala logarítmica.

En base a lo analizado, se opta por considerar al modelo de 6 vecinos como el representativo del algoritmo de K-vecinos cercanos, dado que ese modelo muestra una dispersión de datos y las métricas de error más bajas en lo referente a MAE y RMSE.

3.1.2. Modelo de red neuronal artificial (ANN)

Para las 1000 épocas que pasó el modelo, se puede encontrar una reducción en los indicadores de error absoluto y error medio cuadrado, los cuales evolucionan de manera errática y con el paso de las épocas convergen hacia el valor objetivo de cero. En la tabla 7 se presentan tanto las métricas para los datos entrenados las cuales son '*loss'*,*'mae'* y '*mse'* así como para el bloque separado para la validación, cuyos indicadores llevan el mismo nombre con un '*val'* antes.

Tabla 7. Histórico de métricas del modelo en función de las épocas

	loss	mae	mse	rmse	val_loss	val_mae	val_mse	val_rmse	epoch
0	500.932556	14.787119	500.932556	22.381523	14.492801	3.222538	14.492801	3.806941	0
1	14.745877	3.453383	14.745877	3.840036	2.056140	1.169726	2.056140	1.433925	1
10	32.778976	5.383137	32.778976	5.725293	20.732586	3.922115	20.732586	4.553305	10
100	0.979703	0.810226	0.979703	0.989799	1.391435	1.012544	1.391435	1.179591	100
200	0.768842	0.748071	0.768842	0.876837	0.710696	0.756380	0.710696	0.843028	200
500	0.067323	0.203202	0.067323	0.259466	0.108684	0.246912	0.108684	0.329673	500
700	0.127553	0.294645	0.127553	0.357145	0.084822	0.228693	0.084822	0.291242	700
900	0.057643	0.186987	0.057643	0.240090	0.251293	0.455625	0.251293	0.501291	900
999	0.074190	0.210594	0.074190	0.272379	0.145337	0.271345	0.145337	0.381231	999

En la tabla se observa la historia del modelo con el pasar de las épocas, partiendo de un error cuadrado medio ('*mse'*, por sus siglas en inglés) de 500 gramos, lo cual indica una variación en la predicción de medio kilogramo. Sin embargo, en función de las pasadas se reduce este valor de manera notoria al llegar a la época número 100 y logra una estabilidad entre la época 900 y 1000. Es importante notar que el modelo presentó una mejor capacidad para predecir los valores con los que fue entrenado antes de llegar al final de la etapa.

3.1.2.1. Evolución del error absoluto

En la figura 27 se muestra que al inicio del proceso se presentan altos valores para el error de entrenamiento y validación, los cuales encuentran un decrecimiento notorio a partir de la época 200. Posteriormente a ese punto se ve un asentamiento entre 2 y 0 con un comportamiento errático, el cual marca valores de error de validación mayores a los de entrenamiento.

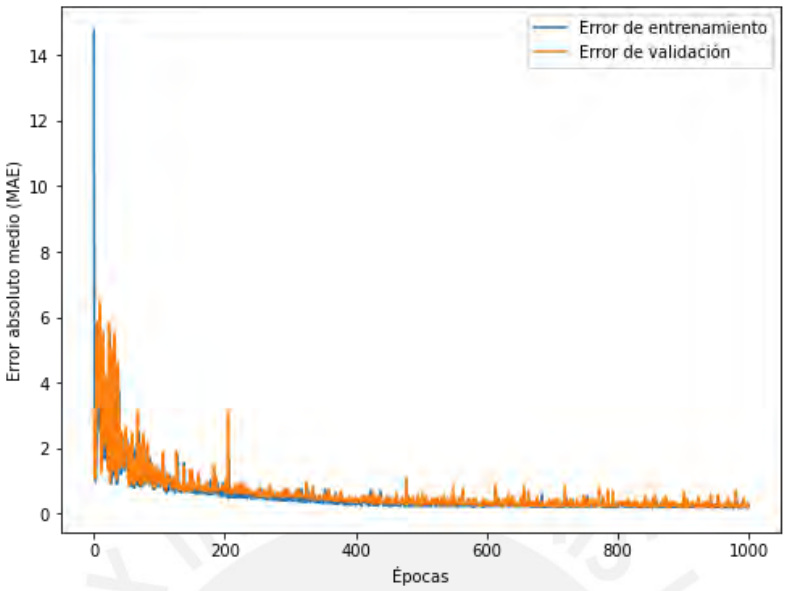


Figura 27. Evolución completa del error absoluto medio en función del pasar de las épocas

A medida que avanzan las épocas se estabiliza el valor de error y se encuentra que en las 10 últimas pasadas se cruzan las medidas de error en varias ocasiones como se puede observar en la figura 28. Asimismo, los valores de error de entrenamiento oscilan entre los 0.29 y 0.16 gramos, con menos picos que los resultados obtenidos para los datos de validación.

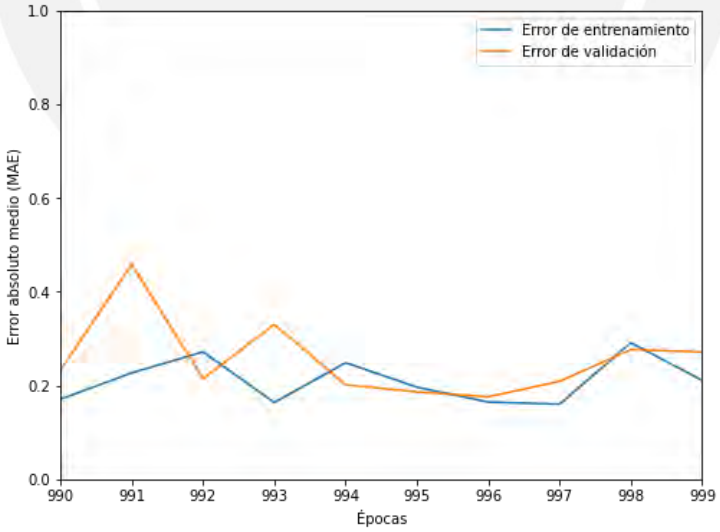


Figura 28. Evolución del error medio absoluto en función del pasar de las épocas. Solo últimas 50 épocas.

3.1.2.2. Evolución de la raíz del error cuadrado medio

En el caso de la raíz del error cuadrado medio, en la figura 29 se observa como parte de valores elevados en un inicio y converge de manera más rápida a los valores cercanos a cero. Por un lado, se podría haber optado por menos épocas para evitar el riesgo de sobreentrenar al modelo; sin embargo, el valor de épocas es el mismo para ambos errores, al ser un mismo modelo.

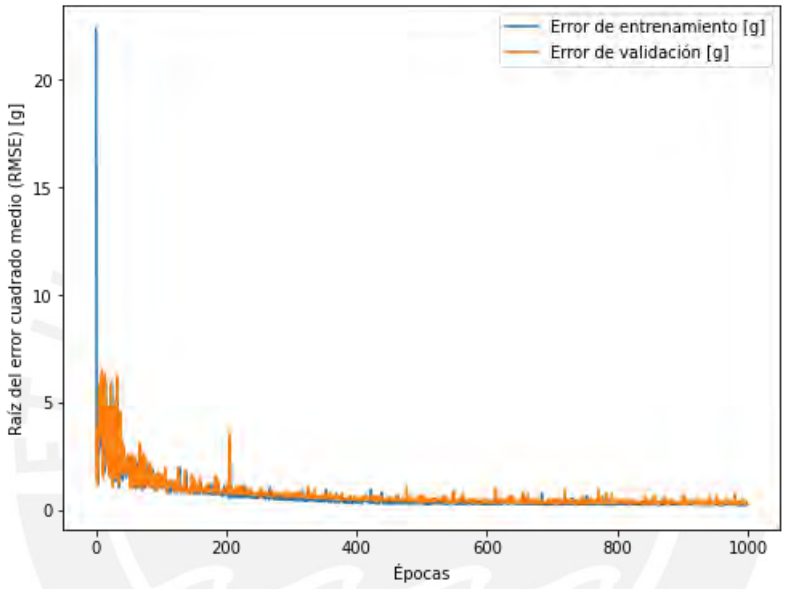


Figura 29. Evolución completa del error cuadrado medio en función del pasar de las épocas.

En la figura 30 se puede apreciar que la variación del error de entrenamiento es bastante reducida y oscila alrededor de valores entre 0.29 y 0.13 gramos. Si bien se observa un pico en el error de validación que llega a 0.52 y un desfase entre los valores de error de entrenamiento y validación, al encontrarse valores tan bajos para el rango que se tiene para pérdida de masa, se considera que el modelo funcionará sin problemas.

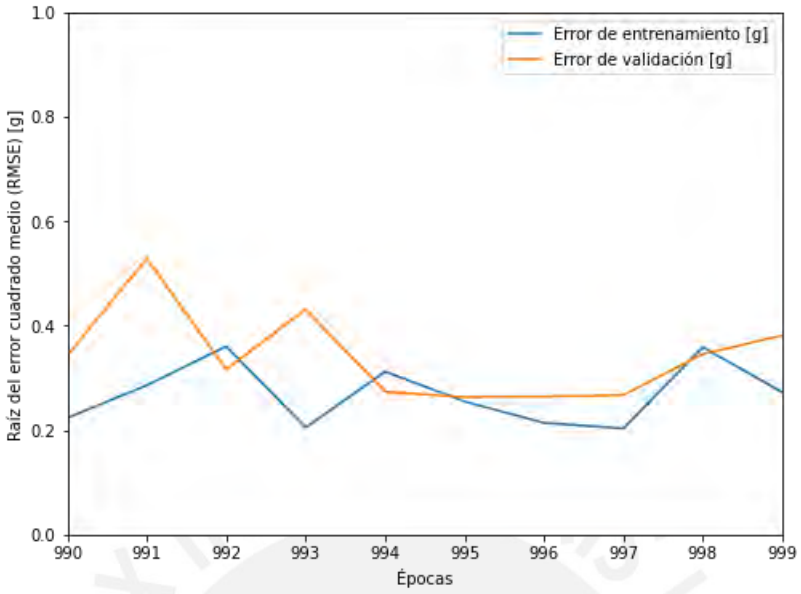


Figura 30. Evolución del error cuadrado medio en función del pasar de las épocas. Solo últimas 50 épocas.

3.1.3. Modelo de máquina de aprendizaje extremo (ELM)

El modelo de la máquina de aprendizaje extremo consiste en los valores de los pesos de las neuronas obtenidos a partir de la minimización del error utilizando la inversa generalizada de Moore-Penrose. A diferencia del modelo anterior, este no cuenta con una historia y una evolución de la reducción del valor de los errores dado que el concepto de máquina de aprendizaje extremo implica evitar este proceso mediante otro enfoque de cálculo. Por ello, se presentará la diferencia entre los valores reales y predecidos para los modelos obtenidos variando la cantidad de neuronas.

3.1.3.1. Comparativo entre valores predecidos y reales por modelo

Para cada uno de los números de neuronas propuestos se generó un modelo y se realizó la comparación entre sus valores predecidos y reales. A partir de ello se agruparon los resultados de acuerdo con la dispersión de los valores y los rangos en los que se encontraron. Asimismo, se tomaron los valores de error absoluto medio y la raíz del error cuadrado medio.

En el caso de los tres primeros modelos, se observa que para el modelo de una sola neurona se obtienen valores de desgaste predecidos en un rango cercano a cero, mientras que para 10 y 15 neuronas se tiene una forma similar. Particularmente, entre los gráficos b y c de la figura 31 se presenta similitud en la forma de dispersión de los valores. De acuerdo con la definición, se espera que, a mayor cantidad de neuronas, los puntos se ubiquen a menor distancia de la recta que cruza el gráfico.

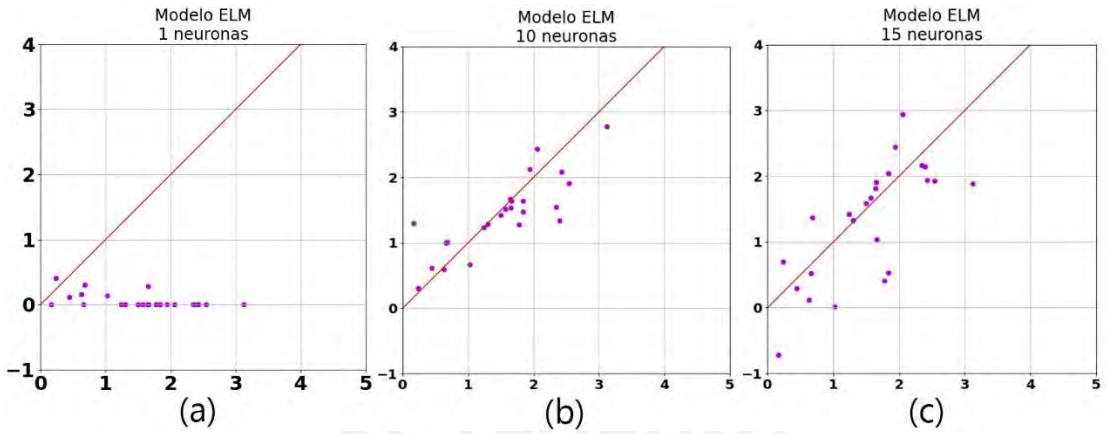


Figura 31. Gráficos de dispersión entre valor real y predicho para máquina de aprendizaje extremo. (a) 1 neurona, (b) 10 neuronas, (c) 15 neuronas.

Para el siguiente bloque de modelos, se presenta en la figura 32 como a medida que incrementa el recuento de neuronas se acercan más los puntos hacia la recta de la función identidad. Partiendo de una gran dispersión en el modelo de 20 neuronas, se observa una marcada reducción en el paso de 25 a 50 unidades. Un aspecto a destacar es como los valores se agrupan a medida que incrementan las neuronas e incluso se observa una medición cercana a los 2 gramos que cae precisamente en la recta de la función identidad.

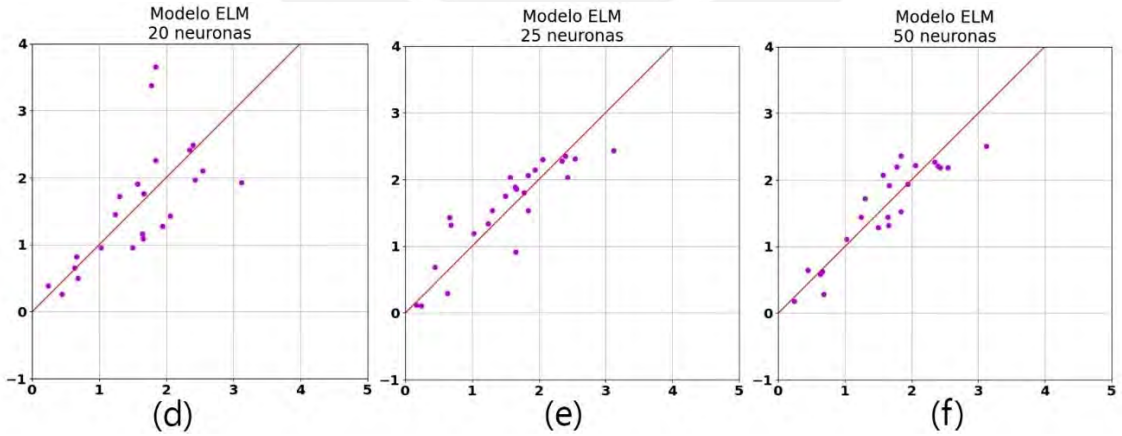


Figura 32. Gráficos de dispersión entre valor real y predicho para máquina de aprendizaje extremo. (d) 20 neuronas, (e) 25 neuronas, (f) 50 neuronas.

Posteriormente, en el modelo de 100 neuronas (g), el cual se aprecia en la figura 33, se encuentra una marcada diferencia con la tendencia a agruparse que se vio en los modelos desde (a) hasta (f). Asimismo, se presencia un resultado de medición de desgaste negativo. En concepto, es imposible que exista una

medición como tal puesto que indicaría que la probeta adquirió masa durante el ensayo de abrasión. La ocurrencia de un resultado así se entiende debido al componente matemático de los modelos, los cuales no entienden los hechos físicos que rigen los datos que reciben. Al ser un único punto ubicado en esta sección y encontrar que la mayoría de mediciones se ordena alrededor de la función identidad se le considera una falla aceptable.

El hecho de pasar de 100 neuronas marca un punto de quiebre puesto que los modelos de red neuronal artificial no trabajan con capas con tal cantidad de neuronas. A partir del modelo (g) se puede presenciar el funcionamiento propio de la máquina de aprendizaje extremo (ELM), en contraste con las mediciones de (a) a (f), cuyas cantidades de neuronas no difieren de las usuales en redes neuronales artificiales (ANN).

En adelante, desde las 200 neuronas a más, la dispersión vuelve a ordenarse con tendencia a concurrir en un punto de la línea de la recta identidad, pero sin presencia de predicciones negativas. Además, al llegar al modelo de 500 neuronas (k) se puede identificar la tendencia que se observa en las mediciones por debajo de 100 neuronas, la cual consiste en el gradual acercamiento de la mayoría de ensayos a la función identidad, con la presencia de algunos puntos alternantes en la zona superior.

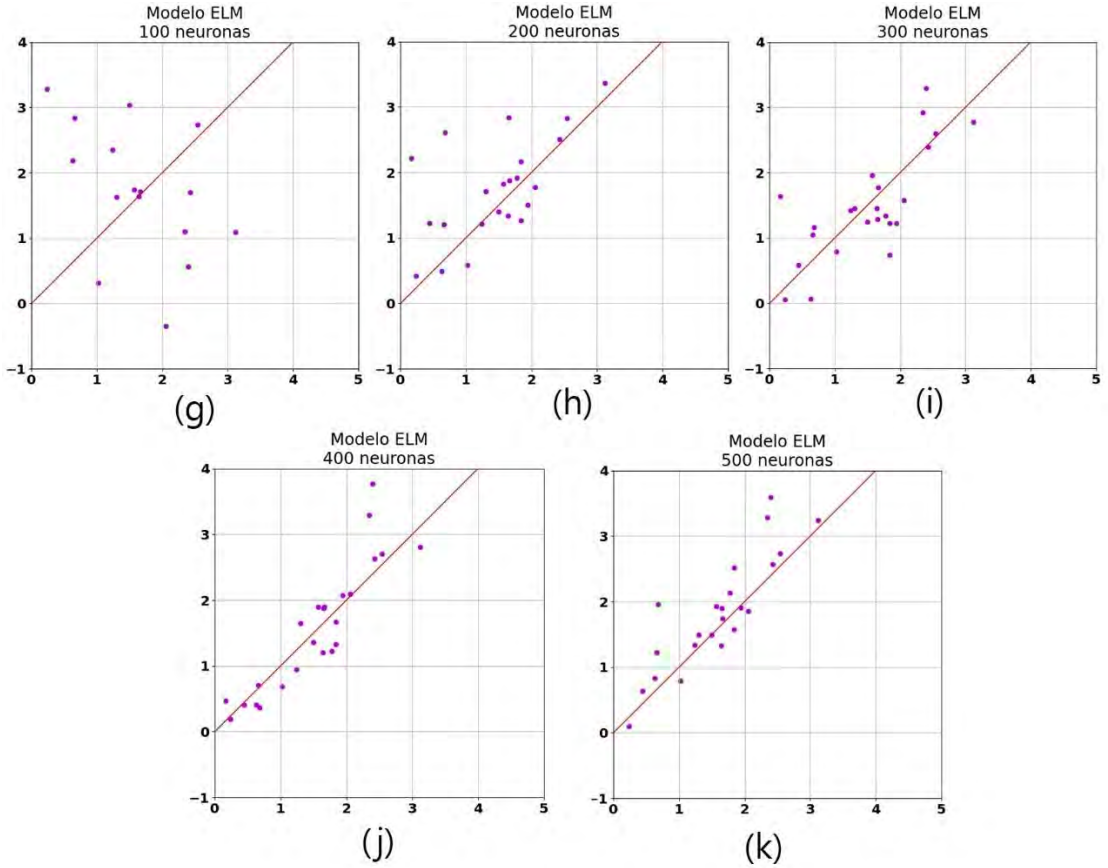


Figura 33. Gráficos de dispersión entre valor real y predicho para máquina de aprendizaje extremo. (g) 100 neuronas, (h) 200 neuronas, (i) 300 neuronas, (j) 400 neuronas, (k) 500 neuronas.

En la figura 34 se presenta que al pasar las 1000 neuronas, los modelos obtenidos toman un comportamiento bastante similar, con 3 puntos por debajo de cero, presentando la anomalía de desgaste negativo y un valor ubicado en el extremo superior que se encuentra notoriamente alejado del bloque de datos. En adición, se observa como los puntos se acercan a medida que incrementa el número de neuronas, presentando un modelo cada vez más cercano al valor de la igualdad entre los resultados reales del ensayo y los valores predecidos.

Para el modelo de 10,000 neuronas se observa una cercanía de datos en la dispersión con una tendencia similar a la presentada para los modelos de 10 y 15 neuronas, con la ventaja de que no solo se tienen ensayos ubicados en un menor rango, sino que también presentan menor distancia con respecto a la recta de igualdad.

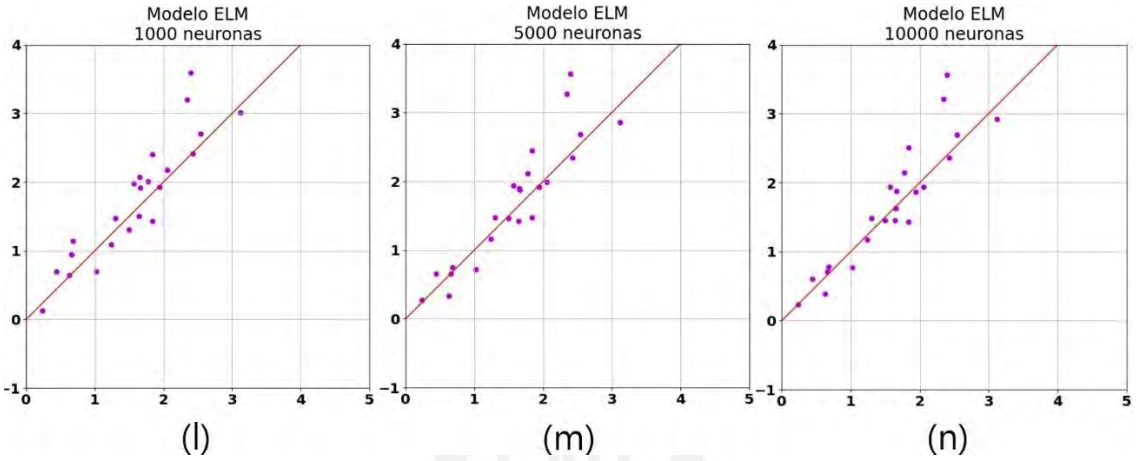


Figura 34. Gráficos de dispersión entre valor real y predicho para máquina de aprendizaje extremo. (l) 1000 neuronas, (m) 5000 neuronas, (n) 10000 neuronas.

En lo referente a los modelos con una cantidad de neuronas en el orden de los cientos de miles, cuyas predicciones se aprecian en la figura 35, se obtuvo un comportamiento semejante. Los cinco gráficos que muestran que la relación entre el valor real y predicho se replica tanto para los puntos cercanos a la recta de igualdad como para los resultados alejados y con valores anómalos. También se nota que el incremento en cientos de miles de neuronas en el modelo no generó una disminución considerable en la dispersión de los valores.

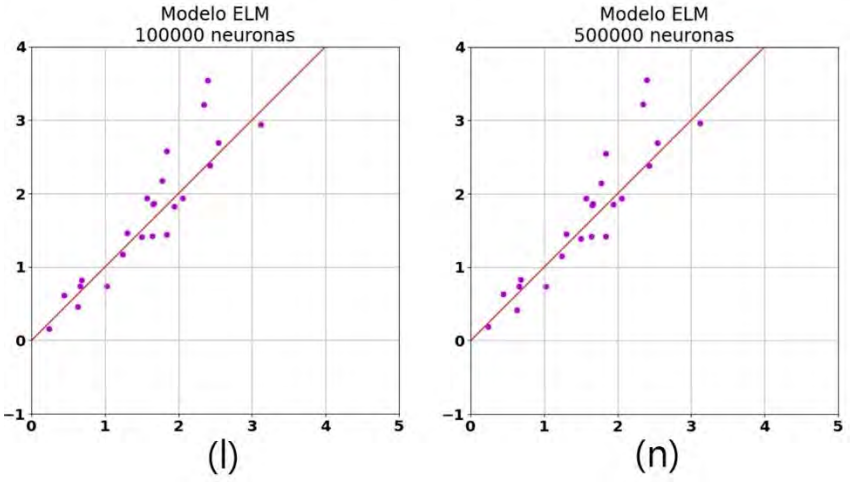


Figura 35. Gráficos de dispersión entre valor real y predicho para máquina de aprendizaje extremo. (o) 100000 neuronas, (p) 500000 neuronas.

Tras analizar la distribución de los datos en los diferentes modelos, y como estos cambian a medida que se incrementa el número de neuronas, se percibe que, en concordancia con los conceptos matemáticos, a medida que aumentan las neuronas se reduce la dispersión de los puntos. Sin embargo, se denotan

predicciones que fallaron repetidamente para los distintos modelos, tanto por resultar en medidas negativas de desgaste como por mantener un resultado notoriamente alejado del resto de datos.

3.1.3.2. Evolución de modelos en función de la cantidad de neuronas

En la tabla 8 se presentan las medidas de error medio absoluto (MAE) y la raíz del error cuadrado medio (RMSE), ambas medidas en gramos.

Tabla 8. Métricas de error en función de la cantidad de neuronas para modelo de máquina de aprendizaje extremo.

N	MAE	RMSE
1	1.484	1.687
10	0.316	0.443
15	0.514	0.654
20	0.505	0.700
25	0.290	0.361
50	0.293	0.385
100	3.326	5.973
200	0.632	0.933
300	0.432	0.549
400	0.321	0.435
500	0.435	0.689
1000	0.351	0.508
5000	0.312	0.457
10000	0.304	0.460
100000	0.329	0.489
200000	0.329	0.472
300000	0.320	0.460
400000	0.324	0.461
500000	0.319	0.464

Contrario a lo esperado, el incremento de la cantidad de neuronas no trajo consigo la reducción marcada en el MAE y la RMSE. Al contrastar este resultado con las observaciones a los gráficos del 31 al 35 se valida que el leve cambio aparente en el ordenamiento de los resultados genera leves variaciones en la media y desviación estándar del error.

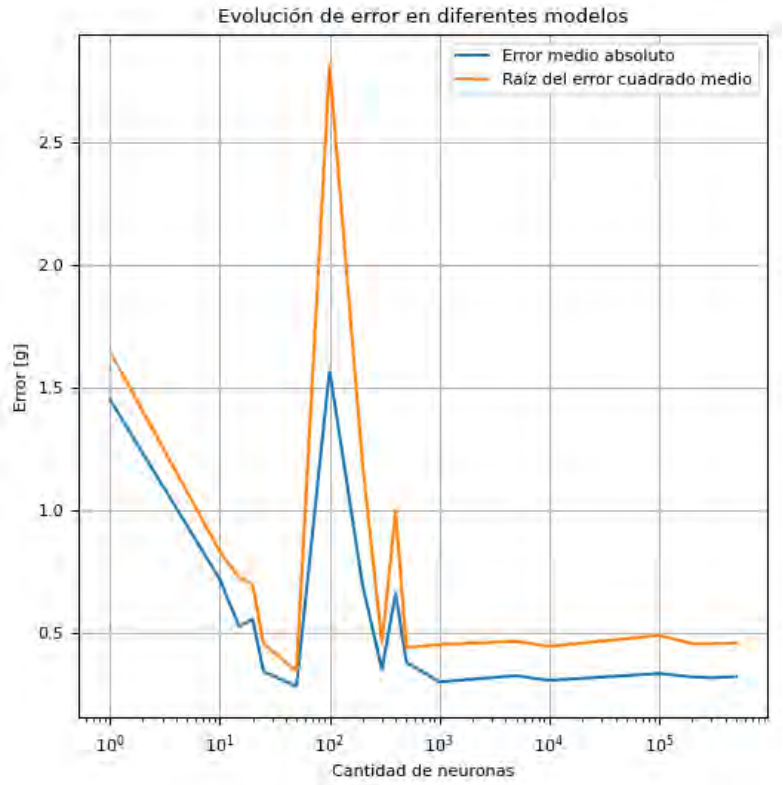


Figura 36. Métricas de error en función de la cantidad de neuronas para modelo de máquina de aprendizaje extremo. Escala logarítmica.

Finalmente, tras analizar los resultados de los diferentes modelos de Máquina de Aprendizaje Extremo (ELM), lo cual se puede realizar a partir de la observación de la figura 36, se puede afirmar que ninguno de los modelos, en las diferentes cantidades de neuronas, tiene la capacidad de predecir el desgaste de manera efectiva. Los resultados obtenidos presentan cercanía a los valores reales y se ubican en un espectro cercano a los resultados de los ensayos; sin embargo, los valores de error son considerablemente elevados en comparación con el rango en el que se encuentran las pérdidas de masa medidas experimentalmente. Además, para efectos de la comparación con los modelos de KNN y ANN se utilizará el modelo de ELM de 10,000 neuronas, dado que a partir de esa cantidad de neuronas se estabilizaron los valores de error.

3.2. Evaluación de valores predecidos

En base al entrenamiento realizado con los datos obtenidos en los ensayos de desgaste realizados por la AWS se ejecutará la predicción de valores de desgaste tanto para los datos pertenecientes al mismo ensayo que no formaron parte del entrenamiento de los modelos como para los provenientes de ensayos realizados en el laboratorio de la PUCP. La evaluación de los valores predecidos consistirá en medir su

cercanía a la recta de la función identidad, la cual representará la igualdad entre los valores predecidos y los reales.

3.2.1. Comparación entre predicciones por modelo para ensayos AWS

En esta sección se revisarán los gráficos de dispersión para cada uno de los modelos de forma independiente y posteriormente se realizará la comparación entre cada uno de los modelos para determinar cuál predice el desgaste abrasivo con mayor éxito.

3.2.1.1. Modelo KNN

De acuerdo con lo explicado en el punto 3.1.3.2. se determinó que el modelo representativo para el algoritmo de KNN sería el de 6 vecinos. En la figura 37 se muestran tres bloques de datos presentes en el gráfico de dispersión. En primer lugar, se ubican 7 puntos a la izquierda de la recta de igualdad los cuales se marcan en el rango de 1.5 a 2.5 gramos de valor predecidos para valores reales entre 0.0 y 1.0 g, etiquetados como "Lejanos – menor" al ser lejanos a la predicción que debió ser un valor menor. En el centro del gráfico se ubican la mayoría de los resultados los cuales cumplen con presentarse muy cerca al espacio de igualdad entre lo real y lo estimado, etiquetados como cercanos. Finalmente, a la derecha de la función identidad se observan 5 puntos que deberían encontrarse entre 2.2 y 3.5 gramos pero que se marcan en el mismo rango que el primer bloque, etiquetados como "Lejanos – mayor".

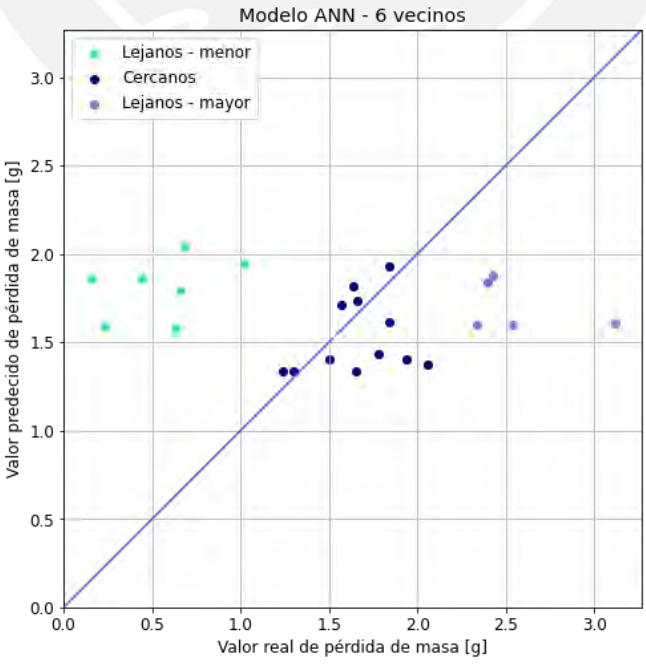


Figura 37. Gráfico de dispersión para valores reales y predichos para resultados del ensayo por AWS según modelo KNN (6 vecinos)

3.2.1.2. Modelo ANN

El modelo de red neuronal artificial, mostrado en la figura 38, presentó un ordenamiento de las predicciones con una clara tendencia cercana a la recta de igualdad entre los valores reales y los predecidos. El gráfico de dispersión muestra que para la mayoría de los resultados se logró una estimación del desgaste esperado.

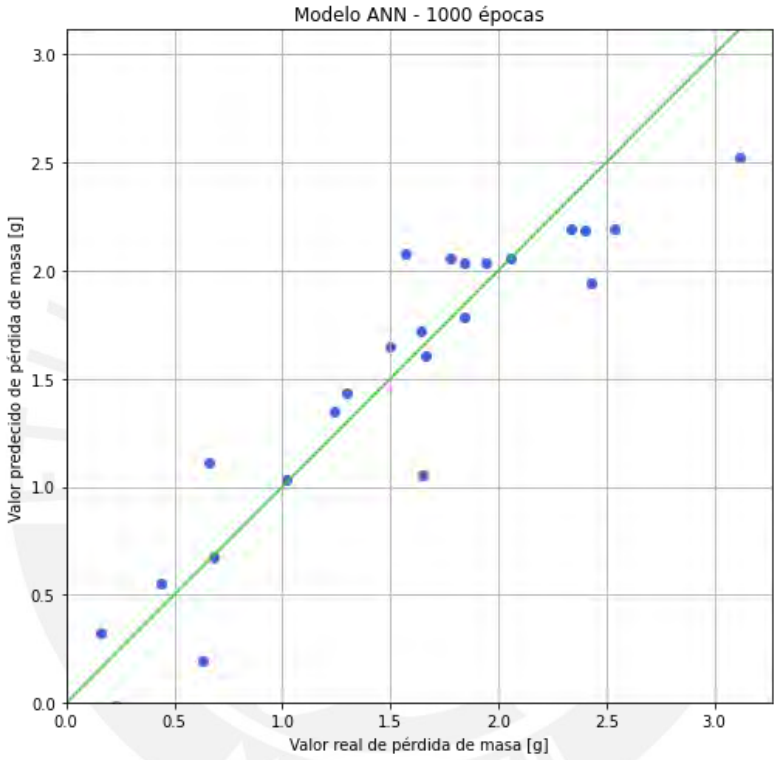


Figura 38. Gráfico de dispersión para valores reales y predichos para resultados del ensayo por AWS según modelo ANN (1000 épocas)

En líneas generales, se puede afirmar que el modelo predice el desgaste de manera efectiva la pérdida de masa en un ensayo de desgaste en función de la dureza y composición química.

3.2.1.3. Modelo ELM

Para el modelo de máquina de aprendizaje extremo se eligió el modelo de 10,000 neuronas (Figura 39). En este modelo se encontró que el grupo mayoritario de valores predecidos se ordena alrededor de la recta de la función identidad. A partir de la apreciación del gráfico de dispersión se visibiliza que, a excepción de 1 resultado anómalo, las predicciones del modelo su ubican entre 0 y 4, correspondiendo a los valores reales que se encuentran en el mismo rango.

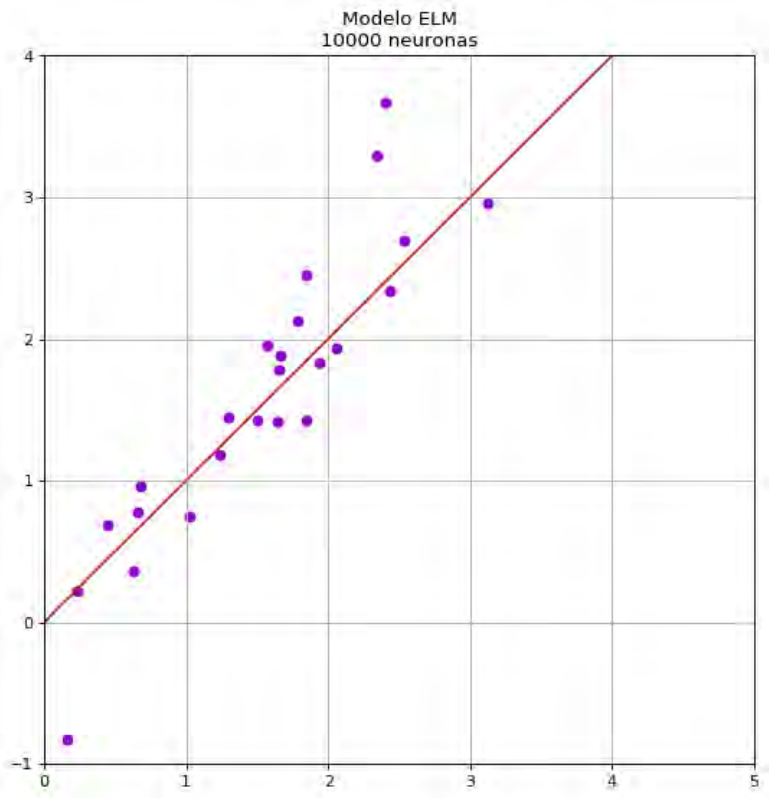


Figura 39. Gráfico de dispersión para valores reales y predichos para resultados del ensayo por AWS según modelo ELM (10000 neuronas)

3.2.1.4. Análisis conjunto de modelos

Al analizar los resultados de forma conjunta se puede comparar el comportamiento de cada modelo y la cercanía a la recta de la función identidad. Luego de haber revisado cada modelo con detenimiento y haber realizado el análisis correspondiente, se encontró que los modelos de red neuronal artificial (ANN) y máquina de aprendizaje extremo (ELM) son los que presentan mayor eficiencia para la predicción del desgaste abrasivo en función de la composición química y la dureza en el depósito cuando se utilizan los ensayos de AWS.

En primer lugar, a partir del gráfico de dispersión en la figura 40 se puede observar que tanto los resultados del modelo ANN como ELM se ordenan cercanos a la recta de la función identidad, en contraste con las predicciones para el algoritmo de KNN, el cual muestra mayor dispersión y una forma diferente. Adicionalmente, se observa que el cuadrante de 1.0 a 2.0 gramos de desgaste real es el que tiene la mayor concentración de resultados predecidos, a partir de ello se entiende que, en general, los tres modelos presentaron buen rendimiento al predecir los resultados ubicados al centro.

En la figura 16 se presentó un histograma de las mediciones de valores reales de desgaste, en el cual se observa que la mayor cantidad de resultados reales se encontraron entre 1.5 y 2.5 gramos, lo cual indica que las predicciones más cercanas al valor real se ubicaron entre 1.5 y 2.0 gramos de pérdida de masa.

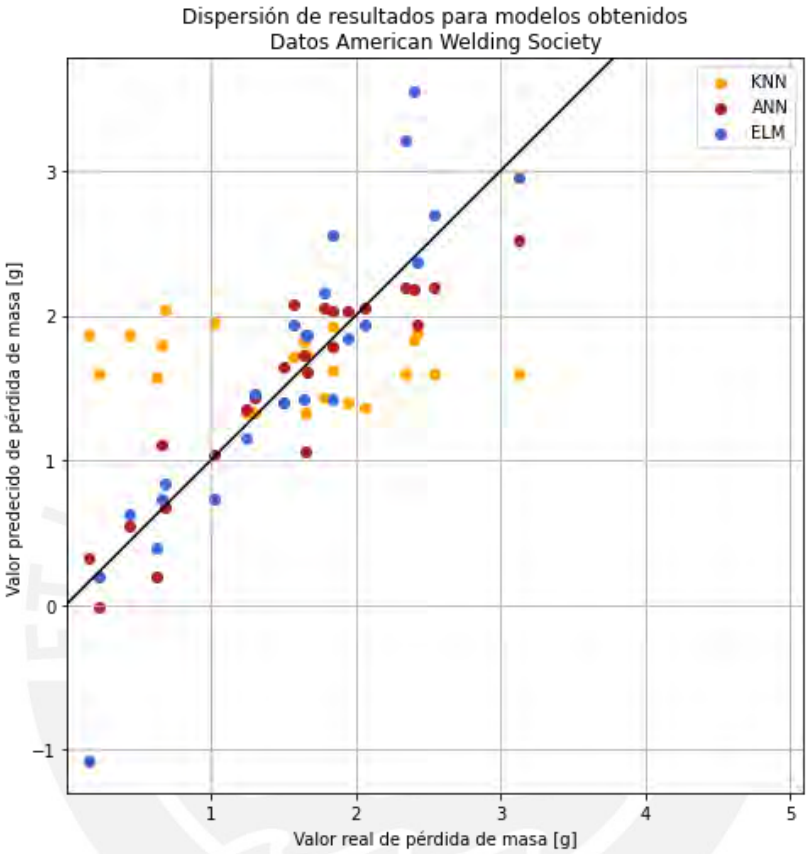


Figura 40. Gráfico de dispersión para los modelos KNN, ANN y ELM entre los resultados reales y las estimaciones realizadas con datos AWS.

En segundo lugar, se encuentra que el resultado negativo encontrado para la máquina de aprendizaje extremo (ELM) representa solo 1 error en contraste con los 23 valores restantes ubicados cerca a la diagonal de igualdad. Asimismo, en comparación con las 71 predicciones realizadas que obtuvieron valores físicamente posibles, independientemente de su precisión, la presencia de un resultado anómalo es considerable como aceptable pues representa sólo el 1.38% de valores calculados.

Finalmente, se puede determinar que los tres modelos predicen valores de desgaste físicamente válidos; sin embargo, solo los modelos de ANN y ELM presentan una tendencia cercana a los valores reales, por lo cual son estos dos los modelos que se consideran para evaluarse frente a los datos provenientes de los ensayos realizados en el laboratorio de Materiales PUCP.

3.3.2. Comparación entre predicciones por modelo para ensayos PUCP

En esta sección se revisarán los gráficos de dispersión para cada uno de los modelos de forma independiente y posteriormente se realizará la comparación entre cada uno de los modelos para determinar cuál predice el desgaste abrasivo con mayor éxito.

3.3.2.1. Modelo ANN

En la figura 41 se marcan 4 puntos notoriamente cercanos a la línea de igualdad, lo cual, sumado al previo éxito en la predicción de los resultados para AWS, agrega validez al modelo de red neuronal artificial. Por otro lado, se observan dos puntos relativamente alejados de la función identidad; sin embargo, la distancia es corta en comparación con la magnitud de los valores de pérdida de masa.

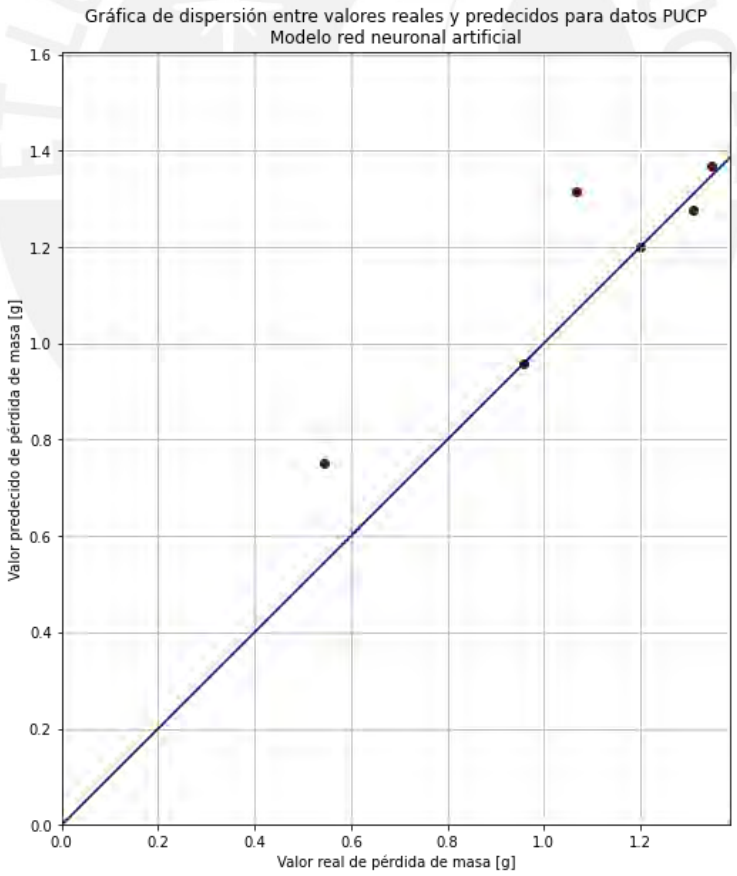


Figura 41. Gráfico de dispersión para valores reales y predichos para resultados del ensayo por PUCP según modelo ANN (1000 épocas)

3.3.2.2. Modelo ELM

En la figura 42 se puede apreciar que las predicciones realizadas para los datos PUCP se alejan de un comportamiento cercano a la recta identidad. Adicionalmente, se encuentra que una medición presenta un valor negativo de desgaste estimado. Asimismo, 3 estimaciones para los valores reales entre 1.0 y 1.5 g se ubicaron por encima de los 1.9 g.

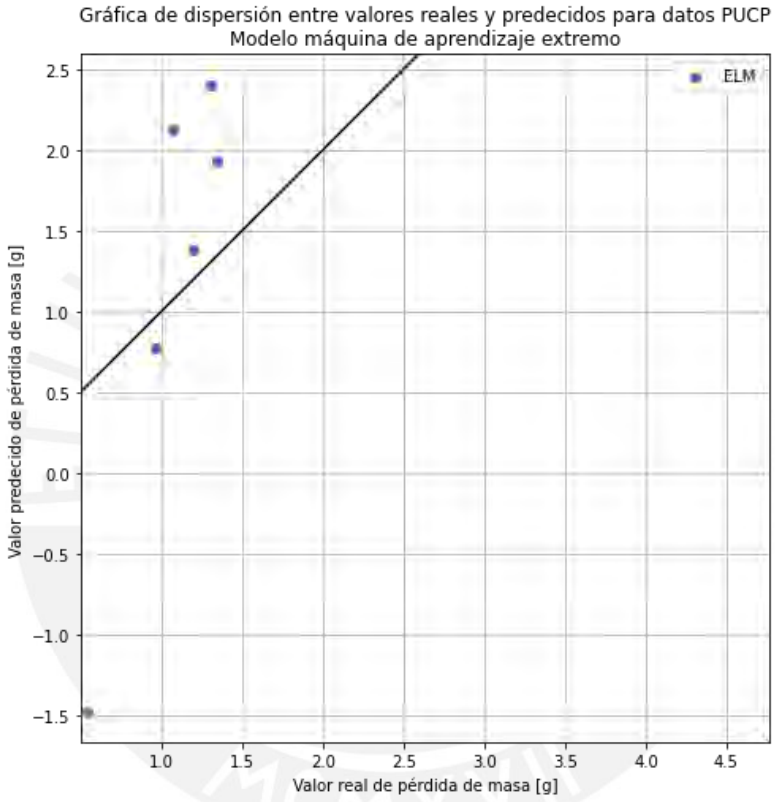


Figura 42. Gráfico de dispersión para valores reales y predichos para resultados del ensayo por PUCP según modelo ELM (10000 neuronas)

El hecho de que un modelo que presentó buen rendimiento en su entrenamiento tenga éxito en predecir datos de su mismo bloque de origen, pero no para estimar resultados en un bloque diferente es una situación conocida como sobreajuste. En este caso, el modelo puede predecir exitosamente los ensayos que tuvieron concentraciones de carbono, cromo, silicio, manganeso, y demás elementos, cercanas a las presentes en su entrenamiento; sin embargo, los datos PUCP presentan valores de concentraciones diferentes, por lo cual el modelo no logra predecirlos satisfactoriamente.

3.3. Evaluación de rendimiento de modelos

En el punto 3.1 se desarrollaron los modelos y para cada uno de ellos se obtuvieron métricas de error. En esta sección se realizará la comparación de los resultados de error para cada uno de los modelos, tanto para los ensayos realizados por la AWS como los realizados en PUCP. Se presentarán gráficos donde se verá el valor que alcanzó cada error para los distintos modelos entrenados.

3.3.1 Error absoluto medio (MAE)

A continuación, se presentan los resultados graficados en la figura 43. El error medio presenta sus menores valores para el algoritmo de red neuronal artificial (ANN), tanto para los datos de AWS como para los de PUCP, alcanzando valores de 0.228 g y 0.086, respectivamente. En el segundo lugar se encuentra el modelo de k-vecinos cercanos (KNN), el cual llegó a valores de error para ambos bloques de datos en el rango de 0.6 a 0.8 g, siendo estos 0.666 g 0.726 g para AWS y PUCP, correspondientemente. Por otro lado, el algoritmo de máquina de aprendizaje extremo (ELM) presentó un valor de rendimiento para el bloque de AWS que asciende a 0.321 g y 0.853 g para los datos PUCP. Según esta métrica, el modelo de ANN presenta el mejor rendimiento en la predicción de datos.

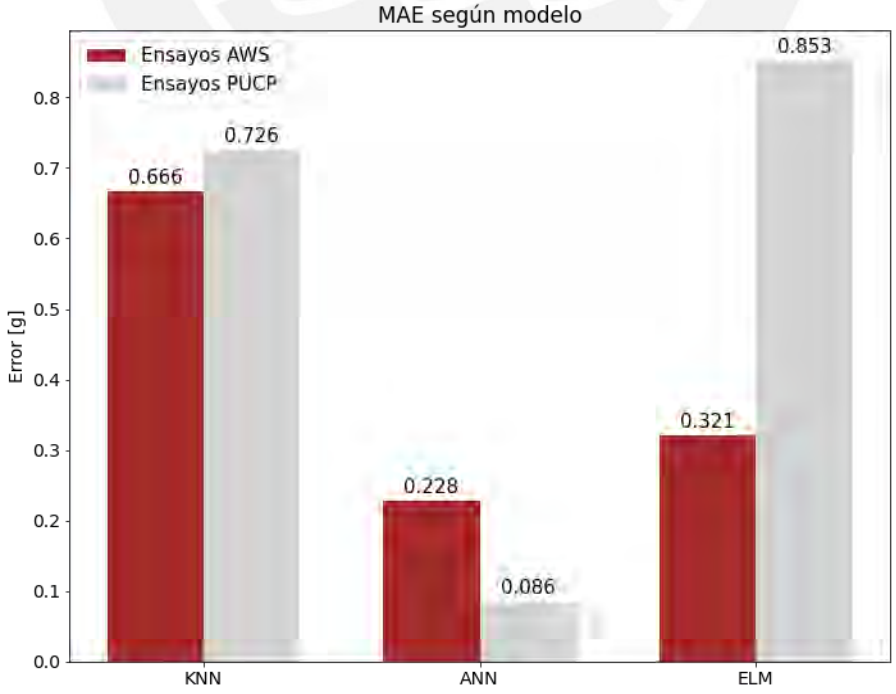


Figura 43. Comparativo de MAE para los modelos realizados.

3.3.2. Raíz del error cuadrado medio (RMSE)

La raíz del error cuadrado muestra dos aspectos diferentes con el MAE, los cuales son que se obtiene un menor error para las predicciones de ensayos PUCP en el algoritmo de KNN y un error en los datos AWS significativamente más elevado para el algoritmo ELM.

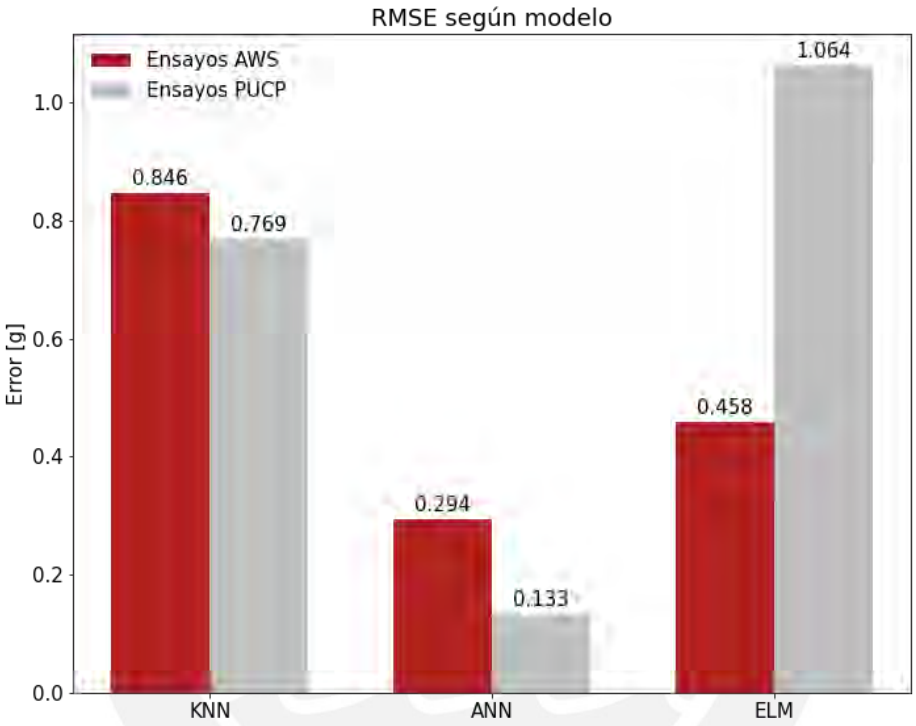


Figura 44. Comparativo de RMSE para los modelos realizados.

En esta métrica se encontraron valores de error en 0.846, 0.294 y 0.458, como se aprecia en la figura 44, para los resultados de AWS en los algoritmos de KNN, ANN y ELM, respectivamente. De forma análoga, se obtienen 0.769, 0.133 y 1.064 para los ensayos PUCP.

ET LUX IN TENEBRIS LUCET
MCMXVII

CONCLUSIONES

- El modelo de regresión desarrollado mediante el algoritmo de red neuronal artificial (ANN) presentó los mejores resultados al realizar predicciones cercanas tanto para el bloque de datos de la AWS como la PUCP, presentando un error medio absoluto de 0.228 gramos y una raíz del error cuadrado medio de 0.294 gramos. En adición, la evolución del error a medida que se prolonga su entrenamiento es decreciente, en contraste con modelos de KNN y ELM. Considerando que la media del desgaste de los ensayos realizados por la AWS y la PUCP fue de 1.60 g y 1.07 g, respectivamente, se determina que el modelo ANN puede calcular el desgaste de una probeta en función de su composición química y dureza de manera exitosa con una precisión del 85.75%.

- La máquina de aprendizaje extremo (ELM) presentó un rendimiento similar a la red neuronal artificial (ANN) en la estimación de la pérdida de masa para el bloque de datos ensayado por la American Welding Society. Además, no requirió de un entrenamiento extenso y sujeto a varios parámetros en contraste con la ANN, y a partir de las 200 neuronas redujo el error en función del incremento de éstas. Por otro lado, requirió elevados recursos computacionales a partir de las 500,000 neuronas. En ese sentido, se considera que el modelo requiere de una mayor cantidad de datos para su entrenamiento y así no limitarse a sólo predecir las muestras provenientes de un ensayo ni depender de una elevada cantidad de neuronas.

- El algoritmo de k-vecinos cercanos (KNN) fue el más sencillo de aplicar debido a que al aumentar la cantidad de vecinos k se observó la convergencia a 1.60 gramos de pérdida de masa, el cual fue el valor promedio de desgaste de los datos de entrenamiento. Los datos de entrenamiento contaron con más de 10 variables correspondientes a la concentración de elementos como carbono, cromo, silicio, manganeso, tungsteno y otros, así como la dureza. Considerando lo antes mencionado y el hecho de que los datos no forman una secuencia, se concluye que este modelo no ofrece una representación significativa de la pérdida de masa en el ensayo ASTM G-65.

- El presente trabajo brinda validación a la aplicación de modelos de aprendizaje automático en la predicción del desgaste abrasivo en base a la composición química del depósito y la dureza, lo cual puede fomentar su aplicación y profundización en su investigación en la industria de la recuperación de componentes mediante recargues duros. En un futuro, con el acceso a conjuntos

y bases de datos de mayor dimensión, se podrían desarrollar modelos para otros tipos de desgaste y en las diferentes condiciones que se presenten fuera de un laboratorio. Adicionalmente, el entrenamiento de modelos con menores valores de error brindará confiabilidad en esta tecnología a la industria para aplicarlos y seleccionar con seguridad los mejores recargues para su requerimiento de resistencia al desgaste.

- Finalmente, se concluye que la aplicación de algoritmos de aprendizaje automático permite realizar regresiones y modelos complejos de manera sistemática. Se realizaron un total de 37 modelos entre KNN, ANN y ELM, a partir de los cuales se tuvieron alternativas variadas para encontrar relacionar el desgaste con la composición química y la dureza del recargue ensayado.

Referencias

[1] IEDEB, «La Cámara,» 2019. [En línea]. Available: https://lacamara.pe/pbi-de-principales-sectores-de-la-economia-peruana-caerian-durante-2020. [Último acceso: 2021].

[2] Agencia AFP, «Gestión,» 20 Agosto 2020. [En línea]. Available: https://gestion.pe/economia/mineria-puede-ser-el-salvavidas-para-economia-peruana-en-recesion-noticia/. [Último acceso: 4 Abril 2021].

[3] R. Chattopadhyay, Surface Wear: Analysis, Treatment, and Prevention, Cleveland: ASM International, 2001.

[4] G. Stachowiak, Wear: Materials, Mechanisms and Practice, New York, United States: John Wiley & Sons Inc, 2006.

[5] American Society for Materials, ASM Handbook Vol. 18 - Friction, Lubrication, and Wear Technology., Cleveland: ASM, 1992.

[6] J. L. Sakihama Uehara, Estudio de la resistencia a la abrasión de bajo esfuerzo, Lima: Pontificia Universidad Católica del Perú, 2006.

[7] C. Lipson, Importancia del desgaste en el diseño, New Jersey: Prentice Hall, 1970.

[8] E. T. Matsumoto, «Biblioteca virtual del Instituto de Investigación de la Facultad de Geología, Minas, Metalurgia y Ciencias Geográficas.,» Diciembre 1999. [En línea]. Available: https://sisbib.unmsm.edu.pe/Bibvirtual/publicaciones/geologia/v02_n4/recubrimientos_pindustriales.htm. [Último acceso: 30 Abril 2021].

[9] H. Horwitz, Soldadura: Aplicaciones y Práctica, México D.F-: Alfaomega, 1989.

[10] L. Taipe, Influencia del contenido de vanadio en la resistencia al desgaste de recubrimientos duros con 5% de ferro-titanio., Lima: Pontificia Universidad Católica del Perú, 2013.

[11] W. D. Callister, Materials Science and Engineering, Massachusetts, Estados Unidos: John Wiley & Sons, Inc, 2009.

[12] D. R. Askeland, Ciencia e Ingeniería de los Materiales, Rolla, Missouri: International Thompson Editores, 1998.

[13] W. F. Smith, Fundamentos de la Ciencia e Ingeniería de los Materiales, Madrid, España: McGraw-Hill Interamericana de España, 1999.

[14] B. M. N. A. D. Jhon D. Keller, Fundamentals of Machine Learning for Predictive Data Analytics, Massachusetts: MIT Press, 2015.

[15] Oxford Lenguages, Oxford Dictionary, Oxford: Oxford University, 2021.

[16] C. Veliz, Aprendizaje automático. Introducción al aprendizaje profundo, Lima: Fondo Editorial de la Pontificia Universidad Católica del Perú, 2020.

[17] A. Géron, Hands-On Machine Learning with Scikit-Learn and TensorFlow, California: O'Reilly Media, Inc., 2017.

[18] VDMA Software and Digitalization, «Machine Learning in Mechanical and Plant Engineering,» VDMA, Frankfurt, Alemania, 2018.

[19] E. M. Mirkes, «The 1NN classification map based on the CNN extracted prototypes.,» 2013. [En línea]. Available: https://commons.wikimedia.org/wiki/File:Map1NNReducedDataSet.png. [Último acceso: 04 Abril 2022].

[20] B. Steele, Algorithms for Data Science, Cham, Switzerland: Springer, 2016.

[21] D. Fumo, «Linear Regression — Intro To Machine Learning #6,» 5 Marzo 2017. [En línea]. Available: https://medium.com/simple-ai/linear-regression-intro-to-machine-learning-6-6e320dbdaf06. [Último acceso: 20 Abril 2022].

[22] E. Alpaydin, Introduction to Machine Learning, Cambridge, Massachusetts: The MIT Press, 2014.

[23] M. Kubat, An Introduction to Machine Learning, Coral Gables, FL, USA: Springer, 2015.

[24] V. Zhou, «Towards Data Science,» 2019 Marzo 5. [En línea]. Available: https://towardsdatascience.com/machine-learning-for-beginners-an-introduction-to-neural-networks-d49f22d238f9. [Último acceso: 7 Abril 2022].

[25] J. M. Zurada, Introduction to Artificial Neural Systems, New York: West Publishing Company, 1992.

[26] S. Gong, «Medium,» 29 Agosto 2019. [En línea]. Available: https://gongster.medium.com/how-does-a-neural-network-work-intuitively-in-code-f51f7b2c1e3f. [Último acceso: 7 Abril 2022].

[27] Q. Z. C. S. G.B. Huang, «Extreme learning machine: a new learning scheme of feedforward neural networks,» de *Proceedings 2004 IEEE International Joint Conference on Neural*, 2004, p. 985–990.

[28] R. Penrose, «A generalized inverse for matrices,» de *Proceedings of the Cambridge Philosophical Society*, 1955, p. 406–13.

[29] T. N. G. Adi Ben-Israel, Generalized Inverses: Theory and Applications, New York: Springer-Verlag, 2003.

[30] J. T. a. A. v. Schaik, «Learning the pseudoinverse solution to network weights,» *Neural Networks,* vol. 1, nº 45, pp. 94-100, 2013.

[31] Keras, «Keras,» [En línea]. Available: https://keras.io/about/. [Último acceso: 10 07 2021].

[32] TensorFlow, «TensorFlow,» [En línea]. Available: https://www.tensorflow.org/. [Último acceso: 10 Julio 2021].

[33] Scikit-Learn, [En línea]. Available: https://scikit-learn.org/. [Último acceso: 10 Julio 2021].

[34] D. J. K. A. J. S. OGBORN, «Abrasion Resistance of Iron-Based Hardfacing Alloys,» American Welding Society, Miami, Florida, 1995.

[35] TensorFlow, «Get started with using TensorFlow to solve for regression problems (Coding TensorFlow),» 1 Marzo 2019. [En línea]. Available: https://youtu.be/-vHQub0NXI4. [Último acceso: 3 Enero 2022].

[36] G. P. Gara, «Towards Data Science,» 29 Mayo 2020. [En línea]. Available: https://towardsdatascience.com/build-an-extreme-learning-machine-in-python-91d1e8958599. [Último acceso: 3 Enero 2022].

[37] A. Noriega, *Resistencia al desgaste abrasivo de recargues duros fabricados con electrodos con 5% de FeTi y contenido variable de FeV,* Lima: Pontificia Universidad Católica del Perú, 2013.

Anexos

Anexo 1: Código de modelo de k-nearest neighbors (KNN)

Basado en código desarrollado por el equipo de TensorFlow (Google) [35]

General

Importar librerías: Scikit-learn para el aprendizaje automático, Pandas y Numpy para el procesamiento de dataframes.

```
[ ] import numpy as np
    from sklearn import neighbors
    import pandas as pd

    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import OneHotEncoder
    from sklearn.preprocessing import MinMaxScaler
    from sklearn.metrics import mean_squared_error,mean_absolute_error
```

Lectura datos AWS

Leer archivo con datos AWS: Se extrae la hoja de Excel que incluye los datos para el entrenamiento y se retiran los valores que son propios del ensayo ASTM G-65 para los cuales no hay variación.

Fuente: http://files.aws.org/wj/supplement/WJ_1995_08_s269.pdf

```
[ ] #Extraer información
    dataframe='/content/drive/MyDrive/TESIS/Data/Dataframe.xlsx'
    df=pd.read_excel(dataframe,sheet_name='Train')
    df=df.drop(columns=['Cu','Fuente','Force','Distance'])

    #Llenar celdas vacías con cero y eliminar celdas que queden vacías
    df=df.fillna(0)
    df=df.dropna()

    #Revisar dataframe
    df.head()
```

	Process	Layer	C	Si	Mn	Ni	Cr	Mo	V	W	Nb	Al	Ti	Zr	Co	HV prom	MassLoss
0	SAW	1	0.062	0.36	1.02	1.40	0.81	0.19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	247	2.35
1	SAW	2	0.045	0.40	0.94	2.08	1.22	0.29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	235	2.57
2	SAW	4	0.035	0.43	0.88	2.69	1.60	0.37	0.0	0.0	0.0	0.0	0.0	0.0	0.0	252	2.64
3	SAW	1	0.097	0.38	1.96	0.07	0.08	0.28	0.0	0.0	0.0	0.0	0.0	0.0	0.0	201	2.50
4	SAW	2	0.107	0.43	2.28	0.06	0.07	0.40	0.0	0.0	0.0	0.0	0.0	0.0	0.0	227	2.31

Revisar indicadores estadísticos del dataframe: Se obtienen medidas de media, desviación estándar, máximos, mínimos y cuartiles, así como cantidad de valores.

```
[ ] df.describe()
```

	Layer	C	Si	Mn	Ni	Cr	Mo	V	W	Nb	Al	Ti	Zr	Co	HV prom	MassLoss
count	119.000000	119.000000	119.000000	119.000000	119.000000	119.000000	119.000000	119.000000	119.000000	119.000000	119.000000	119.0	119.0	119.0	119.000000	119.000000
mean	2.294118	1.057361	0.553866	2.461210	0.335378	8.612941	0.818067	0.039664	0.060084	0.003866	0.082017	0.0	0.0	0.0	430.134454	1.601849
std	1.244329	1.409314	0.446045	3.661699	0.729522	7.775562	2.550526	0.126859	0.203339	0.020875	0.304738	0.0	0.0	0.0	152.495842	0.687301
min	1.000000	0.035000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	178.000000	0.160000
25%	1.000000	0.155000	0.270000	0.845000	0.000000	2.235000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	274.000000	1.240000
50%	2.000000	0.371000	0.430000	1.060000	0.000000	6.430000	0.440000	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	438.000000	1.640000
75%	4.000000	1.745000	0.720000	1.570000	0.100000	12.405000	0.730000	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	540.000000	2.090000
max	4.000000	5.450000	2.070000	15.580000	4.220000	31.800000	27.000000	0.800000	1.020000	0.180000	1.800000	0.0	0.0	0.0	746.000000	3.120000

Dar formato para entrenamiento: Se procesan los datos para introducirlos a una red neuronal posteriormente e iniciar el entrenamiento.

1. A los datos leídos del ensayo AWS se les retira las columnas para las cuales no se indicaron datos en el documento del ensayo.
2. Separar 80% de datos para el entrenamiento y se crean dos columnas, una para introducir los valores a la ANN y otra solo con el dato de pérdida de masa para evaluar el rendimiento.
3. Se obtienen realiza la descripción estadística de los valores separados para el entrenamiento.

```
[ ] #Eliminar columnas sin datos y llenar vacíos con cero
    df=df.drop(labels=['Process','Layer','Ti','Zr','Co'],axis=1)
    df=df.fillna(0)
    df=df.dropna()
```

```
[ ] #Se separa el 80% de los datos
    train_dataset = df.sample(frac=0.8,random_state=0)
    test_dataset = df.drop(train_dataset.index)
```

```
[ ] #Se obtienen estadísticas de dataset de entrenamiento
    train_stats = train_dataset.describe()
    train_stats.pop("MassLoss")
    train_stats = train_stats.transpose()

    #Se extrae columna de pérdida de masa para ser contrastada al final
    train_labels = train_dataset.pop('MassLoss')
    test_labels = test_dataset.pop('MassLoss')
```

Lectura datos PUCP

Leer datos de ensayo PUCP: Se revisan los datos de los ensayos realizados en el Laboratorio de Materiales de la PUCP. Estos datos se pueden encontrar en el repositorio de tesis:

- Ainsworth Noriega: http://tesis.pucp.edu.pe/repositorio/handle/20.500.12404/4938
- Luiggi Taipe: http://tesis.pucp.edu.pe/repositorio/handle/20.500.12404/4897

Estos datos serán utilizados al final del procedimiento para evaluar el rendimiento del modelo frente a datos no vistos de manera complementaria al bloque que prueba propio de los datos AWS.

```
[ ] #Importar archivo de Excel
    ruta_p='/content/drive/MyDrive/TESIS/Data/pucp_df.xlsx'
    df_pucp=pd.read_excel(ruta_p)

    #Presentar dataframe pucp
    df_pucp
```

[ ]

	C	Si	Mn	Ni	Cr	Mo	V	W	Nb	Al	HV prom	MassLoss
0	0.619	0.964	2.212	0.046	6.862	0.435	0.031	0.013	0	0.0120	550.2	1.3471
1	0.579	0.882	2.083	0.043	6.237	0.389	0.305	0.010	0	0.0118	576.2	1.3101
2	0.616	0.743	2.162	0.046	6.877	0.428	0.707	0.006	0	0.0076	536.6	1.2006
3	0.596	0.876	2.044	0.043	6.183	0.404	2.116	0.000	0	0.0000	684.0	0.5435
4	0.632	0.953	2.150	0.045	6.721	0.430	1.080	0.001	0	0.0000	651.0	0.9582
5	0.578	0.859	2.099	0.043	6.165	0.396	0.115	0.005	0	0.0000	596.0	1.0665

Dar formato para evaluación: Se separa la columna de pérdida de masa (gramos) y crea otra columna con esos datos. De esta manera se obtienen dos dataframes, uno con los datos para evaluar y otro con el valor real, que será contrastado con el obtenido por el modelo.

```
[ ] test_pucp=df_pucp.drop(columns='MassLoss',axis=1)
    pucp_labels = df_pucp.pop('MassLoss')
```

Entrenamiento

Normalizacion de datos: Para los datos de entrenamiento, realiza la normalización de valores.

$$Z = \frac{x-\mu}{\sigma}$$

De esta manera, el hecho de que valores como la dureza y la composición, que se encuentran en distintos rangos, son evaluados inicialmente como inputs con el mismo peso.

```
[ ] #Se define función de normalización
    def norm(x):
      return (x - train_stats['mean']) / train_stats['std']

    #Se aplica normalización a datos de entrenamiento y de prueba
    normed_train_data = norm(train_dataset)
    normed_test_data = norm(test_dataset)
```

```
[ ] X_train=normed_train_data
    y_train=df.MassLoss[df.index.isin(train_dataset.index)]
    from sklearn.neighbors import KNeighborsRegressor
    neigh = KNeighborsRegressor(n_neighbors=6)
    neigh.fit(X_train, y_train)
```

```
    KNeighborsRegressor(n_neighbors=6)
```

```
[ ] X_train=normed_train_data
    y_test=np.array(df.MassLoss[~df.index.isin(train_dataset.index)].tolist())
```

```
[ ] prediction_test=neigh.predict(normed_test_data)
    prediction_pucp=neigh.predict(norm(test_pucp))
```

Anexo 2: Código de modelo de red neuronal artificial (ANN)

Basado en código desarrollado por el equipo de TensorFlow (Google) [35]

General

Importar librerías: TensorFlow y Keras para el aprendizaje automático, Pandas para el procesamiento de dataframes.

```
[ ] import tensorflow as tf
    from tensorflow import keras
    from tensorflow.keras import layers
    import pandas as pd
```

Lectura datos AWS

Leer archivo con datos AWS: Se extrae la hoja de Excel que incluye los datos para el entrenamiento y se retiran los valores que son propios del ensayo ASTM G-65 para los cuales no hay variación.

Fuente: http://files.aws.org/wj/supplement/WJ_1995_08_s269.pdf

```
[ ] #Extraer información
    dataframe='/content/drive/MyDrive/TESIS/Data/Dataframe.xlsx'
    df=pd.read_excel(dataframe,sheet_name='Train')
    df=df.drop(columns=['Cu','Fuente','Force','Distance'])

    #Llenar celdas vacías con cero y eliminar celdas que queden vacías
    df=df.fillna(0)
    df=df.dropna()

    #Revisar dataframe
    df.head()
```

	Process	Layer	C	Si	Mn	Ni	Cr	Mo	V	W	Nb	Al	Ti	Zr	Co	HV prom	MassLoss
0	SAW	1	0.062	0.36	1.02	1.40	0.81	0.19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	247	2.35
1	SAW	2	0.045	0.40	0.94	2.08	1.22	0.29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	235	2.57
2	SAW	4	0.035	0.43	0.88	2.69	1.60	0.37	0.0	0.0	0.0	0.0	0.0	0.0	0.0	252	2.64
3	SAW	1	0.097	0.38	1.96	0.07	0.08	0.28	0.0	0.0	0.0	0.0	0.0	0.0	0.0	201	2.50
4	SAW	2	0.107	0.43	2.28	0.06	0.07	0.40	0.0	0.0	0.0	0.0	0.0	0.0	0.0	227	2.31

Revisar indicadores estadísticos del dataframe: Se obtienen medidas de media, desviación estándar, máximos, mínimos y cuartiles, así como cantidad de valores.

```
[ ] df.describe()
```

	Layer	C	Si	Mn	Ni	Cr	Mo	V	W	Nb	Al	Ti	Zr	Co	HV prom	MassLoss
count	119.000000	119.000000	119.000000	119.000000	119.000000	119.000000	119.000000	119.000000	119.000000	119.000000	119.000000	119.0	119.0	119.0	119.000000	119.000000
mean	2.294118	1.057361	0.553866	2.461210	0.335378	8.612941	0.818067	0.039664	0.060084	0.003866	0.082017	0.0	0.0	0.0	430.134454	1.601849
std	1.244329	1.409314	0.446045	3.661699	0.729522	7.775562	2.550526	0.126859	0.203339	0.020875	0.304738	0.0	0.0	0.0	152.495842	0.687301
min	1.000000	0.035000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	178.000000	0.160000
25%	1.000000	0.155000	0.270000	0.845000	0.000000	2.235000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	274.000000	1.240000
50%	2.000000	0.371000	0.430000	1.060000	0.000000	6.430000	0.440000	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	438.000000	1.640000
75%	4.000000	1.745000	0.720000	1.570000	0.100000	12.405000	0.730000	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	540.000000	2.090000
max	4.000000	5.450000	2.070000	15.580000	4.220000	31.800000	27.000000	0.800000	1.020000	0.180000	1.800000	0.0	0.0	0.0	746.000000	3.120000

Dar formato para entrenamiento: Se procesan los datos para introducirlos a una red neuronal posteriormente e iniciar el entrenamiento.

1. A los datos leídos del ensayo AWS se les retira las columnas para las cuales no se indicaron datos en el documento del ensayo.
2. Separar 80% de datos para el entrenamiento y se crean dos columnas, una para introducir los valores a la ANN y otra solo con el dato de pérdida de masa para evaluar el rendimiento.
3. Se obtienen realiza la descripción estadística de los valores separados para el entrenamiento.

```
#Eliminar columnas sin datos y llenar vacíos con cero
df=df.drop(labels=['Process','Layer','Ti','Zr','Co'],axis=1)
df=df.fillna(0)
df=df.dropna()
```

```
#Se separa el 80% de los datos
train_dataset = df.sample(frac=0.8,random_state=0)
test_dataset = df.drop(train_dataset.index)
```

```
#Se obtienen estadísticas de dataset de entrenamiento
train_stats = train_dataset.describe()
train_stats.pop("MassLoss")
train_stats = train_stats.transpose()

#Se extrae columna de pérdida de masa para ser contrastada al final
train_labels = train_dataset.pop('MassLoss')
test_labels = test_dataset.pop('MassLoss')
```

Lectura datos PUCP

Leer datos de ensayo PUCP: Se revisan los datos de los ensayos realizados en el Laboratorio de Materiales de la PUCP. Estos datos se pueden encontrar en el repositorio de tesis:

- Ainsworth Noriega: http://tesis.pucp.edu.pe/repositorio/handle/20.500.12404/4938
- Luiggi Taipe: http://tesis.pucp.edu.pe/repositorio/handle/20.500.12404/4897

Estos datos serán utilizados al final del procedimiento para evaluar el rendimiento del modelo frente a datos no vistos de manera complementaria al bloque que prueba propio de los datos AWS.

```
[ ] #Importar archivo de Excel
    ruta_p='/content/drive/MyDrive/TESIS/Data/pucp_df.xlsx'
    df_pucp=pd.read_excel(ruta_p)

    #Presentar dataframe pucp
    df_pucp
```

	C	Si	Mn	Ni	Cr	Mo	V	W	Nb	Al	HV prom	MassLoss
0	0.619	0.964	2.212	0.046	6.862	0.435	0.031	0.013	0	0.0120	550.2	1.3471
1	0.579	0.882	2.083	0.043	6.237	0.389	0.305	0.010	0	0.0118	576.2	1.3101
2	0.616	0.743	2.162	0.046	6.877	0.428	0.707	0.006	0	0.0076	536.6	1.2006
3	0.596	0.876	2.044	0.043	6.183	0.404	2.116	0.000	0	0.0000	684.0	0.5435
4	0.632	0.953	2.150	0.045	6.721	0.430	1.080	0.001	0	0.0000	651.0	0.9582

Dar formato para evaluación: Se separa la columna de pérdida de masa (gramos) y crea otra columna con esos datos. De esta manera se obtienen dos dataframes, uno con los datos para evaluar y otro con el valor real, que será contrastado con el obtenido por el modelo.

```
[ ]  test_pucp=df_pucp.drop(columns='MassLoss',axis=1)
     pucp_labels = df_pucp.pop('MassLoss')
```

Entrenamiento

Normalizacion de datos: Para los datos de entrenamiento, realiza la normalización de valores.

$$Z = \frac{x-\mu}{\sigma}$$

De esta manera, el hecho de que valores como la dureza y la composición, que se encuentran en distintos rangos, son evaluados inicialmente como inputs con el mismo peso.

```
[ ]  #Se define función de normalización
     def norm(x):
       return (x - train_stats['mean']) / train_stats['std']

     #Se aplica normalización a datos de entrenamiento y de prueba
     normed_train_data = norm(train_dataset)
     normed_test_data = norm(test_dataset)
```

Construcción del modelo: Al analizar los gráficos provenientes de los datos de AWS (presentados en documento de trabajo de investigación), se encontró que solo la dureza y el carbono forman una distribución que se asemejo a exponencial o cuadrática. Por ello se opta por más capas de neuronas, las cuales reducen la linealidad del modelo.

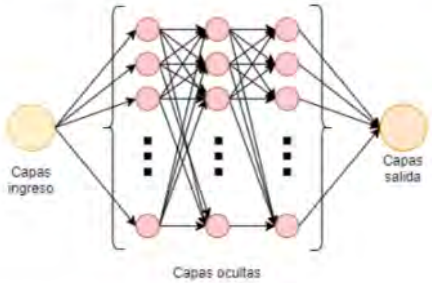
Se define una función de optimización RMSprop (Root Mean Square) en función de una razón de aprendizaje (*learning rate*), valor entre 0 y 1, para el cual se optó por 0.001.

Para la compilación se considera la función de pérdida y además se consideran las métricas como *mae*: mean absolute error y *mse*: mean square error.

```
[ ]  #Definir función de creación de modelo
     def build_model():
       model = keras.Sequential([
         #Capa de ingreso
         layers.Dense(64, activation='relu', input_shape=[len(train_dataset.keys())],name='ingreso'),

         #Capas ocultas
         layers.Dense(64, activation='relu',name='oculta_1'),
         layers.Dense(64, activation='relu',name='oculta_2'),
         layers.Dense(64, activation='relu',name='oculta_3'),

         #Capa de salida
         layers.Dense(1,name='salida')
       ],name='entrenar_desgaste')

       #Optimizar por RMS, Learning rate=0.001
       optimizer = tf.keras.optimizers.RMSprop(0.001)

       #Compilar modelo a partir de definición
       model.compile(loss='mse',
                     optimizer=optimizer,
                     metrics=['mae', 'mse',tf.keras.metrics.RootMeanSquaredError()])
       return model
```

Se genera un modelo según lo indicado en la función de creación de modelo

```
[ ]  #Construir modelo
     model = build_model()
```



Se presenta el modelo creado en la función de creación titulado: **Entrenar desgaste** y se indica un total de 13,313 parametros entrenables.

Se presenta el modelo creado en la función de creación titulado: **Entrenar desgaste** y se indica un total de 13,313 parametros entrenables.

```
[ ]  model.summary()
```

```
Model: "entrenar_desgaste"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
ingreso (Dense)              (None, 64)                768
_________________________________________________________________
oculta_1 (Dense)             (None, 64)                4160
_________________________________________________________________
oculta_2 (Dense)             (None, 64)                4160
_________________________________________________________________
oculta_3 (Dense)             (None, 64)                4160
_________________________________________________________________
salida (Dense)               (None, 1)                 65
=================================================================
Total params: 13,313
Trainable params: 13,313
Non-trainable params: 0
_________________________________________________________________
```

Prueba de modelo sin entrenar: Se toma el modelo realizado y le pide calcular el desgaste. Dado que el modelo no ha sido entrenado, se espera que genere valores sin sentido alguno; sin embargo, el objetivo de este paso es saber si tiene la capacidad de generar valores, independientemente de si tienen sentido o no.

```
[ ]  #Tomar 10 valores de los datos para prueba
     example_batch = normed_train_data[:10]

     #Aplicar modelo
     example_result = model.predict(example_batch)

     #Presentar datos tomados
     print(example_batch)

     #Imprimir desgaste calculado
     example_result
```

```
           C        Si        Mn  ...        Nb        Al   HV prom
48  0.271395  0.716102 -0.465487  ... -0.207936  0.950423 -0.811872
94  0.681755  1.465224 -0.468007  ... -0.207936 -0.271312  0.085582
95  1.233876  2.776188 -0.581373  ... -0.207936 -0.271312  1.488269
8  -0.658257  0.084030 -0.253871  ... -0.207936 -0.271312 -0.758690
97  1.599470  3.595541 -0.571296  ... -0.207936 -0.271312  1.089400
22 -0.611252 -0.267121 -0.437776  ... -0.207936 -0.271312  0.085582
7  -0.646319 -0.150071 -0.316852  ... -0.207936 -0.271312 -1.024602
10 -0.685117  0.037210 -0.001946  ... -0.207936 -0.271312 -0.619086
45 -0.627666 -0.290531 -0.276544  ... -0.207936  3.236250 -0.619086
89 -0.467253  0.271310  2.076430  ... -0.207936 -0.271312 -1.184149

[10 rows x 11 columns]
```

```
[ ]  [10 rows x 11 columns]
     array([[-0.11899695],
            [-0.38886088],
            [-0.6729592 ],
            [-0.17068735],
            [-0.74440295],
            [-0.12320748],
            [-0.1650123 ],
            [-0.14994086],
            [-0.01297727],
            [-0.04791642]], dtype=float32)
```

Entrenamiento de ANN: Se utiliza la librería Keras para presentar la evolución del modelo que se va a entrenar. Además, se indican los siguientes parámetros importantes:

- **EPOCHS**: Número de épocas, es decir la cantidad de veces que los datos pasarán desde la neurona de input a través de la red hacia el output y de regreso. En este caso, se optó por 2890 a partir de un proceso iterativo.
- **Validation split**: Es el porcentaje de datos del bloque para entrenamiento que se separan y no serán entrenados, pero que participaran del proceso de entrenamiento. Para ellos, a medida que se actualiza el modelo con cada época, se les procesa en el modelo y se determinan un error (parametro a reducir).

```
[ ]  # Display training progress by printing a single dot for each completed epoch
     class PrintDot(keras.callbacks.Callback):
       #Se define la función de paso de épocas
       def on_epoch_end(self, epoch, logs):
         if epoch % 100 == 0: print('')
         print('.', end='')

     #Se indica la cantidad de épocas
     EPOCHS = 1000

     #Se define como evoluciona la historia del modelo, con un validation split de 0.2
     history = model.fit(
       train_dataset, train_labels,
       epochs=EPOCHS, validation_split = 0.2,
       verbose=0,
       callbacks=[PrintDot()])
```

Anexo 3: Código de modelo de máquina de aprendizaje extremo (ELM)

Basado en códigos desarrollados por el equipo de TensorFlow (Google) [35] y por Gleen Paul Gara PhD [36].

General

Importar librerías: scikit-learn para el aprendizaje automático, Pandas y Numpy para el procesamiento de dataframes y Scipy para el álgebra líneal

```
[ ] import numpy as np
    import pandas as pd
    import scipy
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import OneHotEncoder
    from sklearn.preprocessing import MinMaxScaler
```

Lectura datos AWS

Leer archivo con datos AWS: Se extrae la hoja de Excel que incluye los datos para el entrenamiento y se retiran los valores que son propios del ensayo ASTM G-65 para los cuales no hay variación.

Fuente: http://files.aws.org/wj/supplement/WJ_1995_08_s269.pdf

```
[ ] #Extraer información
    dataframe='/content/drive/MyDrive/TESIS/Data/Dataframe.xlsx'
    df=pd.read_excel(dataframe,sheet_name='Train')
    df=df.drop(columns=['Cu','Fuente','Force','Distance'])

    #Llenar celdas vacías con cero y eliminar celdas que queden vacías
    df=df.fillna(0)
    df=df.dropna()

    #Revisar dataframe
    df.head()
```

	Process	Layer	C	Si	Mn	Ni	Cr	Mo	V	W	Nb	Al	Ti	Zr	Co	HV prom	MassLoss
0	SAW	1	0.062	0.36	1.02	1.40	0.81	0.19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	247	2.35
1	SAW	2	0.045	0.40	0.94	2.08	1.22	0.29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	235	2.57
2	SAW	4	0.035	0.43	0.88	2.69	1.60	0.37	0.0	0.0	0.0	0.0	0.0	0.0	0.0	252	2.64
3	SAW	1	0.097	0.38	1.96	0.07	0.08	0.28	0.0	0.0	0.0	0.0	0.0	0.0	0.0	201	2.50
4	SAW	2	0.107	0.43	2.28	0.06	0.07	0.40	0.0	0.0	0.0	0.0	0.0	0.0	0.0	227	2.31

Revisar indicadores estadísticos del dataframe: Se obtienen medidas de media, desviación estándar, máximos, mínimos y cuartiles, así como cantidad de valores.

```
[ ] df.describe()
```

	Layer	C	Si	Mn	Ni	Cr	Mo	V	W	Nb	Al	Ti	Zr	Co	HV prom	MassLoss
count	119.000000	119.000000	119.000000	119.000000	119.000000	119.000000	119.000000	119.000000	119.000000	119.000000	119.000000	119.0	119.0	119.0	119.000000	119.000000
mean	2.294118	1.057361	0.553866	2.461210	0.335378	8.612941	0.818067	0.039664	0.060084	0.003866	0.082017	0.0	0.0	0.0	430.134454	1.601849
std	1.244329	1.409314	0.446045	3.661699	0.729522	7.775562	2.550526	0.126859	0.203339	0.020875	0.304738	0.0	0.0	0.0	152.495842	0.687301
min	1.000000	0.035000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	178.000000	0.160000
25%	1.000000	0.155000	0.270000	0.845000	0.000000	2.235000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	274.000000	1.240000
50%	2.000000	0.371000	0.430000	1.060000	0.000000	6.430000	0.440000	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	438.000000	1.640000
75%	4.000000	1.745000	0.720000	1.570000	0.100000	12.405000	0.730000	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	540.000000	2.090000
max	4.000000	5.450000	2.070000	15.580000	4.220000	31.800000	27.000000	0.800000	1.020000	0.180000	1.800000	0.0	0.0	0.0	746.000000	3.120000

Dar formato para entrenamiento: Se procesan los datos para introducirlos a una red neuronal posteriormente e iniciar el entrenamiento.

1. A los datos leídos del ensayo AWS se les retira las columnas para las cuales no se indicaron datos en el documento del ensayo.
2. Separar 80% de datos para el entrenamiento y se crean dos columnas, una para introducir los valores a la ANN y otra solo con el dato de pérdida de masa para evaluar el rendimiento.
3. Se obtienen realiza la descripción estadística de los valores separados para el entrenamiento.

```
[ ] #Eliminar columnas sin datos y llenar vacíos con cero
    df=df.drop(labels=['Process','Layer','Ti','Zr','Co'],axis=1)
    df=df.fillna(0)
    df=df.dropna()
```

```
[ ] #Se separa el 80% de los datos
    train_dataset = df.sample(frac=0.8,random_state=0)
    test_dataset = df.drop(train_dataset.index)
```

```
[ ] #Se obtienen estadísticas de dataset de entrenamiento
    train_stats = train_dataset.describe()
    train_stats.pop("MassLoss")
    train_stats = train_stats.transpose()

    #Se extrae columna de pérdida de masa para ser contrastada al final
    train_labels = train_dataset.pop('MassLoss')
    test_labels = test_dataset.pop('MassLoss')
```

Lectura datos PUCP

Leer datos de ensayo PUCP: Se revisan los datos de los ensayos realizados en el Laboratorio de Materiales de la PUCP. Estos datos se pueden encontrar en el repositorio de tesis:

- Ainsworth Noriega: http://tesis.pucp.edu.pe/repositorio/handle/20.500.12404/4938
- Luiggi Taipe: http://tesis.pucp.edu.pe/repositorio/handle/20.500.12404/4897

Estos datos serán utilizados al final del procedimiento para evaluar el rendimiento del modelo frente a datos no vistos de manera complementaria al bloque que prueba propio de los datos AWS.

```
[ ] #Importar archivo de Excel
    ruta_p='/content/drive/MyDrive/TESIS/Data/pucp_df.xlsx'
    df_pucp=pd.read_excel(ruta_p)

    #Presentar dataframe pucp
    df_pucp
```

	C	Si	Mn	Ni	Cr	Mo	V	W	Nb	Al	HV prom	MassLoss
0	0.619	0.964	2.212	0.046	6.862	0.435	0.031	0.013	0	0.0120	550.2	1.3471
1	0.579	0.882	2.083	0.043	6.237	0.389	0.305	0.010	0	0.0118	576.2	1.3101
2	0.616	0.743	2.162	0.046	6.877	0.428	0.707	0.006	0	0.0076	536.6	1.2006
3	0.596	0.876	2.044	0.043	6.183	0.404	2.116	0.000	0	0.0000	684.0	0.5435
4	0.632	0.953	2.150	0.045	6.721	0.430	1.080	0.001	0	0.0000	651.0	0.9582

Dar formato para evaluación: Se separa la columna de pérdida de masa (gramos) y crea otra columna con esos datos. De esta manera se obtienen dos dataframes, uno con los datos para evaluar y otro con el valor real, que será contrastado con el obtenido por el modelo.

```
[ ] test_pucp=df_pucp.drop(columns='MassLoss',axis=1)
    pucp_labels = df_pucp.pop('MassLoss')
```

Entrenamiento

Normalizacion de datos: Para los datos de entrenamiento, realiza la normalización de valores.

$$Z = \frac{x-\mu}{\sigma}$$

De esta manera, el hecho de que valores como la dureza y la composición, que se encuentran en distintos rangos, son evaluados inicialmente como inputs con el mismo peso.

```
[ ] #Se define función de normalización
    def norm(x):
      return (x - train_stats['mean']) / train_stats['std']

    #Se aplica normalización a datos de entrenamiento y de prueba
    normed_train_data = norm(train_dataset)
    normed_test_data = norm(test_dataset)
```

```
[ ] np.random.normal(size=[2,3])
```

```
array([[ 2.18264586,  0.88051891, -0.98890515],
       [-0.56391096, -1.44630265, -0.03458984]])
```

```
[ ] #Número de parámetros entrenables
    input_size = normed_train_data.shape[1]
    #número de neuronas
    hidden_size = 10000

    #Pesos (weights) de las neuronas y sesgos (biases)
    input_weights = np.random.normal(size=[input_size,hidden_size])
    biases = np.random.normal(size=[hidden_size])

    #Función de activación ReLU
    def relu(x):
       return np.maximum(x, 0, x)
```

```
[ ] len(input_weights[0])
```

```
10000
```

```
[ ] #Función de creación de matriz inversa generalizda
    #Moore-Penrose
    def hidden_nodes(X,input_weights,biases):
        G = np.dot(X, input_weights)
        G = G + biases
        H = relu(G)
        return H
```

```
[ ] X_train=normed_train_data
    y_train=df.MassLoss[df.index.isin(train_dataset.index)]
    output_weights = np.dot(scipy.linalg.pinv2(hidden_nodes(X_train,input_weights,biases)), y_train)
```

```
[ ] y_train
```

```
0      2.35
1      2.57
2      2.64
3      2.50
4      2.31
       ...
113    1.35
114    1.53
116    2.74
117    2.69
118    1.35
Name: MassLoss, Length: 95, dtype: float64
```

```
[ ] X_train
```

	C	Si	Mn	Ni	Cr	Mo	V	W	Nb	Al	HV prom
48	0.271395	0.716102	-0.465487	-0.479214	-0.301199	-0.206583	-0.342839	-0.252732	-0.207936	0.950423	-0.811872
94	0.681755	1.465224	-0.468007	2.323173	2.663714	0.191420	-0.342839	-0.252732	-0.207936	-0.271312	0.085582
95	1.233876	2.776188	-0.581373	-0.479214	1.941172	-0.100919	-0.342839	-0.252732	-0.207936	-0.271312	1.488269
8	-0.658257	0.084030	-0.253871	-0.371430	-0.876741	-0.157273	-0.342839	-0.252732	-0.207936	-0.271312	-0.758690

```
[ ] output_weights
```

```
array([ 0.00174365, -0.01878439, -0.00549543, ...,  0.00148073,
        0.00027431,  0.00070777])
```

```
[ ] def predict(X,input_weights,biases,output_weights):
        out = hidden_nodes(X,input_weights,biases)
        out = np.dot(out, output_weights)
        return out
```