UNIVERSIDAD DEL BÍO-BÍO
FACULTAD DE CIENCIAS EMPRESARIALES

Facultad de Ciencias Empresariales

Departamento de Sistemas de Información

Análisis de datos a través de Machine Learning para predecir el comportamiento de máquinas de aserrío de alta productividad

Proyecto de tesis para optar al título de Ingeniero Civil en Informática

---

Diego Joaquín Reyes Hernández

Diego.reyes1401@alumnos.ubiobio.cl

Profesor Guía:

Patricio Alejandro Galdames Sepúlveda

Profesor Co Guía:

Mario Alejandro Ramos Maldonado

Septiembre 2020

Agradecimientos

***Agradezco a mis profesores**, Mario y Patricio por apoyarme con dedicación y compromiso, en la ultima etapa de mi carrera, gracias a sus enseñanzas, me encaminaron a este logro. Gracias a todas esas semanas de reuniones que, con cada tema aprendido y desarrollado, iban creciendo las preguntas y ganas de continuar aprendiendo más.*

***Agradezco a mis padres y hermana**, Jorge, Cecilia, Renato y Monserrat. Soy un afortunado de contar con las tres personas que me criaron y formaron para poder llegar a ser la persona que soy hoy, además de mi hermana que ha sido uno de mis pilares en la vida. Gracias a ustedes y a su amor he podido realizar todo lo que me he propuesto.*

***Agradezco a mis amigos y polola,** por todos estos años de cariño y amistad, en particular a Eduardo, Roberto, Oscar y Macarena, por su dedicación, ánimo y conocimiento brindado.*

***Un especial agradecimiento a mis Abuelos**, Graciela y Roberto. Que lamentablemente no podrán estar físicamente para la finalización de esta etapa, pero siempre han estado en corazón y mente. Gracias a ellos soy hoy una persona correcta y con valores.*

Tabla de contenido

Capítulo 1: Introducción

En este capítulo se presenta introducción tanto a los conceptos principales del proyecto, como también el problema específico a resolver y la metodología que se utilizara.

1. Introducción

La presente investigación tiene como propósito presentar y explicar la comparación de diferentes modelos Machine Learning, para la problemática que se nos presenta y es de suma importancia para el sector forestal. La predicción de la velocidad de alimentación en el proceso de aserrado, muy importante para la productividad de las empresas industriales del sector primario y secundario, lo que hace que, predecir este tipo de valores tenga un impacto en diferentes procesos, en particular en la industria forestal.

Gracias las nuevas tecnologías digitales es posible recopilar gran cantidad de datos sobre maquinarias claves en procesos transversales para este tipo de compañías manufactureras. La disponibilidad de sensores y técnicas de máquinas de aprendizaje automático (Machine Learning) permiten el despliegue y análisis inteligente capaz de asociarse a estrategias modernas de prescripción, esto es, recomendaciones operacionales para que las variables se mantengan en los límites de control y garanticen el cumplimiento de la función objetivo. El problema puede observarse como una caja negra desde el punto de vista del sistema, con muchas entradas (Características) y una sola salida (Velocidad de alimentación).

Para el desarrollo de este proyecto de título se utilizó la metodología CRIS-DM, que es una metodología de proceso estándar de la industria, probado y robusto, para proyectos de minería de datos y análisis. Esta metodología describe claramente los pasos, procesos y flujos necesarios para ejecutar cualquier proyecto analítico, desde la formalización de los requisitos comerciales hasta la prueba y el despliegue de la solución. La ciencia de datos, la minería de datos y el ML se caracterizan por

ejecutar múltiples procesos para extraer conocimientos e información de los datos. “Por lo tanto se puede decir que el análisis de datos es realmente tanto un arte como una ciencia, porque no siempre trata de ejecutar algoritmos sin razón; gran parte del esfuerzo principal implica la comprensión del negocio, siendo el valor real de los esfuerzos, y los métodos adecuados para articular los resultados finales y las percepciones” (Sarkar,2018).

“La metodología CRIS-DM se describe en términos de un modelo de procesos jerárquico, que consiste en conjunto de tareas descritas en cuatro niveles de abstracción: Fase, Tarea genérica, Tarea especializada e instancia de proceso” (Chapman et al., 2000).

Esta metodología nos permite construir una solución de extremo a extremo para cualquier proyecto. Se identifican 6 pasos claves (figura 1) algunos de ellos iterativos, al igual que un proyecto de software tenemos un ciclo de vida del desarrollo de software con varias fases o pasos a seguir para un proyecto de desarrollo de software:

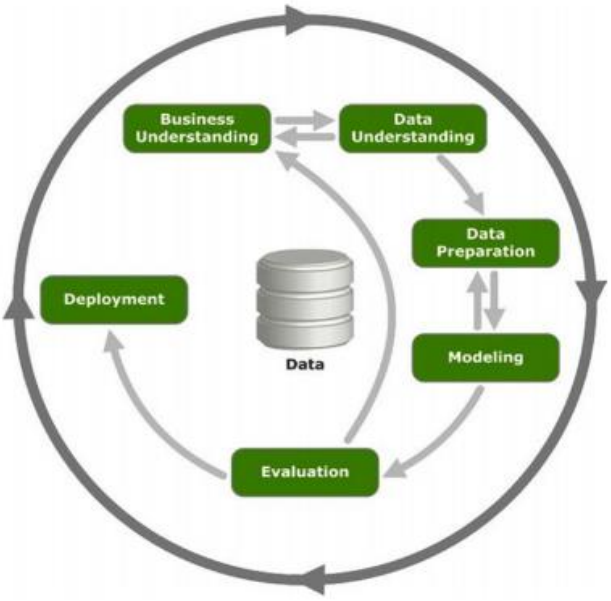


Figura 1: Ciclo de vida de un proyecto de minería de datos CRISP-DM. (Chapman et al.,2000)

1. Comprensión del negocio

   Esta fase inicial se centra en la comprensión de los objetivos y requisitos del proyecto desde una perspectiva empresarial, y luego convertir este conocimiento en una definición del problema de la minería de datos y un plan preliminar diseñado para lograr los objetivos.

2. Comprensión de los datos

   La segunda fase denominada comprensión de los datos comienza con la recopilación inicial de datos y continua con actividades que permitirán familiarizarse con los datos, identifica problemas de calidad de estos datos descubrir las primeras percepciones de los datos, y/o detectar subconjuntos interesantes para formar hipótesis sobre la información global y parcial.

3. Preparación de datos

   La tercera fase denominada preparación de los datos abarca todas las actividades necesarias para construir el conjunto de datos definitivo (datos que se introducirán en el modelo ML) a partir de los datos iniciales en bruto. Es probable que las tareas de preparación de datos se realicen vareas veces y no en cualquier orden prescrito. Las tareas abarcan la selección de tablas, registros y atributos, así como la transformación y limpieza de los datos para el modelo ML.

4. Modelo

   En esta cuarta fase se seleccionan y aplican diversas técnicas de modelización y se calibran sus parámetros para su optimo valor. En general existen varias técnicas para el mismo tipo de problema de minería de datos. Algunas de ellas tienen determinados requisitos sobre la forma de los datos, en algunas ocasiones se debe a preparar los datos nuevamente dependiendo del modelo que requiera el problema.

5. Evaluación

   En esta etapa del proyecto, se ha seleccionado y construido un modelo (o modelos) indicado a partir de la perspectiva de los datos. Pero antes de poder proceder al desarrollo final del modelo, es importante evaluar y revisar el desempeño del modelo, para así asegurar que el modelo alcance adecuadamente los objetivos establecidos. Al término de esta fase se decidirá la utilización de los resultados a partir de la extracción de datos.

6. Implementación

   La creación del modelo no suele ser el fin del proyecto. Incluso si el propósito del modelo es aumentar el conocimiento de los datos, los conocimientos adquiridos deben organizarse y presentarse de manera que el cliente pueda utilizarlos sin inconvenientes. Esto implica en algunas ocasiones la aplicación de modelos “Vivos” dentro de los procesos para la toma de decisiones de una organización, por ejemplo, en tiempo real la personalización de las páginas web o la repetición de la puntuación de las bases de datos de marketing.

   En relación a los requisitos, la última fase llamada de implementación, puede ser tan simple como la generación de un informe o tan compleja como la implementación de una extracción de datos de la empresa. Es relevante, que el cliente entienda por adelantado que acciones deben llevarse a cabo, para poder utilizar realmente y de forma correcta los modelos generados.

Durante la investigación y siguiendo el orden que nos plantea la metodología, se pudo llegar a entrenar con éxito modelos predictivos que se presentan en literatura (Multilayer Perceptron, Redes neuronales recurrentes, Maquinas de soporte vectorial de regresión) que pudieran predecir la velocidad de alimentación, comparándolos entre ellos, a través de métricas que nos permitiera conocer y entender los resultados.

2. Descripción del problema

La industria forestal es uno de los principales sectores económico del país 2,0% del PIB nacional y también uno de los principales subsectores económicos del centro sur del país. En este sector, la producción de madera aserrada resulta estratégica para la industria de la construcción, dada la sustentabilidad de la madera. En esta industria, la incorporación de automatización de procesos ha sido significativa, sin embargo, aun con un bajo nivel de aprovechamiento de los datos y diversas variables significativas que no son aún medidas en línea.

En la industria forestal existen muchos procesos importantes para el ciclo de producción de madera aserrada. El aserrado es el más común de los procesos denominados Machining Processes, que se refiere a los procesos en los que una pieza de materia prima se corta en una forma y tamaño final deseados, mediante un proceso de eliminación de material controlado. Es gracias a las máquinas de aserrío poder realizar este proceso que conlleva transportar por una cinta, troncos de madera hacia la sierra para su corte como se observa en la figura 2.
Existen diversos factores que intervienen en el proceso de aserrado, que impactan tanto en el desgaste de las herramientas, su fuerza de corte, el consumo de energía, etc., algunos de ellos que se pueden mencionar son, por ejemplo, los relacionados propiamente tal con las características mecánicas y anatómicas de la madera como la temperatura, densidad, calidad de la superficie, etc., los factores de alimentación como la velocidad de alimentación, la profundidad, potencia y dirección del corte (tangencial, longitudinal, etc.), etc. y los factores de la hoja de sierra como los ángulos de despeje, dirección de rotación, temperatura de los dientes, etc. (Nasir & Cool, 2018).

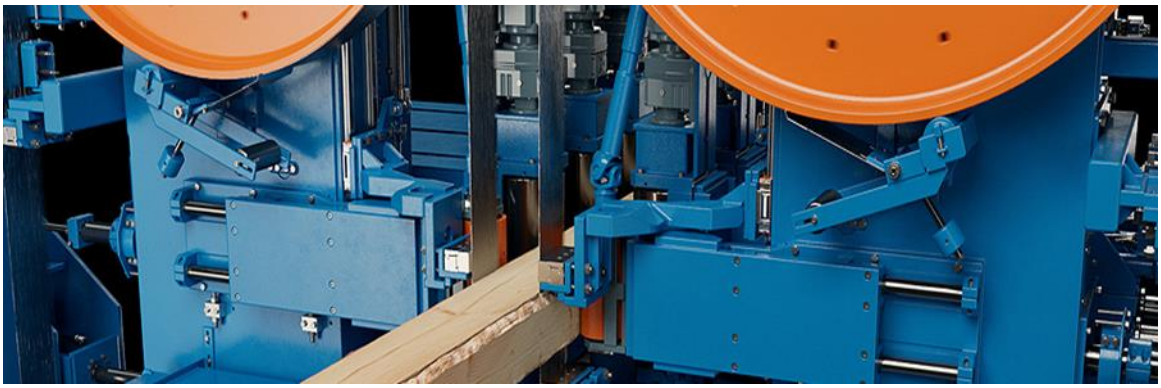


Figura 2: Proceso de corte en máquina de aserrío. Fuente: CMPC

En las fábricas de sectores tanto primarios como secundarios, uno de sus objetivos si no el más importante, es como poder aumentar la productividad, aumentando la velocidad de alimentación, pero el efecto adverso de este enfoque sobre los factores que mencionamos anteriormente y otros más, imponen limitaciones practicas al alcance real de la velocidad de alimentación que se está utilizando.

En un contexto de industria del futuro 4.0, la interpretación de los datos son claves para la toma de decisiones. La actual disponibilidad de capacidad de cómputo y de almacenamiento viabilizan el uso del internet industrial (Internet of Things, IoT), la captura de datos y su tratamiento en tiempo real en procesos industriales. Hoy, algoritmos de análisis basados en inteligencia artificial hacen posible la toma de decisiones en línea y la predicción de fenómenos difíciles de modelar con técnicas convencionales. La disponibilidad de sensores y técnicas de máquinas de aprendizaje automático (Machine Learning) permiten el despliegue y análisis inteligente capaz de asociarse a estrategias modernas de prescripción, esto es, recomendaciones operacionales para que las variables se mantengan en los límites de control y garanticen el cumplimiento de la función objetivo.

La problemática de este proyecto, se define como la predicción de la velocidad de alimentación, que permita la ayuda al aumento del factor de operación y la productividad en plantas de aserrío de alta producción, asegurando la calidad del producto, a través de técnicas de Machine Learning que permita a los tomadores de decisiones tener una herramienta de apoyo. Se debe evaluar diferentes escenarios

de funcionamientos de equipos que son manejadas por al menos 70 variables de control, a través de datos masivos obtenidos en tiempo real, disponibles y alojados en una base de datos. La problemática se puede observar como una caja negra desde el punto de vista de un sistema que posee una entrada y una salida, como se muestra a continuación en la figura 3:

Figura 3: Caja negra problemática. Fuente propia.

3. Objetivos del proyecto de titulo

A continuación, se describen los objetivos definidos para el desarrollo de este proyecto:

1. Objetivo General:

Estudiar modelos de Machine Learning para análisis de datos a través de modelos algorítmicos de Machine Learning, que permita predecir el comportamiento de una maquinaria industrial de aserrío que apoye la toma de decisiones.

2. Objetivos Específicos:

1. Evaluación y comparación de algoritmos de Machine Learning
2. Depuración de datos para crear data set de validación
3. Investigación y selección de métricas
4. Selección de modelos y métricas

Capítulo 2: Marco Teórico

El marco teórico, que se presenta a continuación, permite conocer los conceptos básicos necesarios para el entendimiento del proyecto.

Definiendo concepto que son claves tanto globales como generales, podremos tener una idea de lo que actualmente está sucediendo en el mercado de las compañías manufactureras que ya son parte de la cuarta revolución industrial o como se denomina “industria 4.0”, enfocándonos en las empresas de rubro de la madera, entendiendo procesos que actualmente cuentan automatizados y las herramientas tecnológicas que están

Posteriormente mencionaremos las técnicas de predicción como son Machine Learning, profundizando en sus aspectos claves, sus modelos y su importancia en las empresas hoy en día.

Para finalizar se explicará detalladamente acerca de la metodología que se utilizará para poder realizar este proyecto la cual esta.

1. Industria 4.0

La industria a través de los años ha avanzado a pasos agigantados con la ayuda de las distintas herramientas y tecnologías que al pasar de las décadas acrecientan la capacidad de cómputo y acción frente a distintos procesos industriales manufactureros. Como mencionan (Quin, Liu & Grosvenor, 2016) citados en (Vaidyaa, et al., 2018) “Desde la primera revolución industrial, las revoluciones posteriores han resultado en la fabricación, a partir de agua y máquinas de vapor para la producción eléctrica y digital automatizada que hace que el proceso de fabricación sea más complicado, automático y sostenible, para que las personas puedan operar máquinas de manera mucho más simple, eficiente y persistente”.

“El termino Industria 4.0 representa la cuarta revolución industrial que es definida como el nuevo nivel de organización y control, sobre toda la cadena de valor del ciclo de vida de los productos, la que está orientada a requisitos de clientes cada vez más individualizados” (Vaidyaa, et al.,2018).

Pero más simple que eso, ¿Que se entiende por Revolución Industrial?

La Revolución industrial suele referirse al complejo de innovaciones tecnológicas que, al sustituir la habilidad del ser humano por la de la maquinaria y la fuerza humana y animal por energía mecánica, provoca el paso desde la producción artesanal a la fabril, dando así lugar al nacimiento de la economía moderna (Chavez,2004).

De acuerdo con esto, las consecuencias que trajo consigo la industrialización fue la sucesión relacionada a cambios tecnológicos que pudieron sustituir la capacidad humana por instrumentos mecánicos y diferentes tipos de energía. Estos cambios de equipo y métodos trajeron consigo nuevas formas de organización industrial y la economía se vio al alza, debido al acelerado crecimiento de los ingresos (Lucas,2003).

Hablando un poco de la historia, “La primera revolución industrial tiene sus inicios a fines del ciclo 18 y se caracterizó por plantas de producción mecánicas. La industria 2.0 fue el periodo en que los productos industriales florecieron tanto como en volumen como en variedad. Desde la década de los 70’s a la actualidad la industria tubo una tercera revolución enfocándose en el uso de la electrónica y tecnologías de la información (TI), para la automatización de la producción” (Bigliardi, et al., 2019). Fue así como la industria mundial fue avanzando con los años y con nuevos tipos de herramientas o tecnologías destacadas en cada época (Figura 4). Lo anterior provocó una oleada de digitalización generalizada que creo un entorno adecuado para la industria 4.0 de hoy en día.

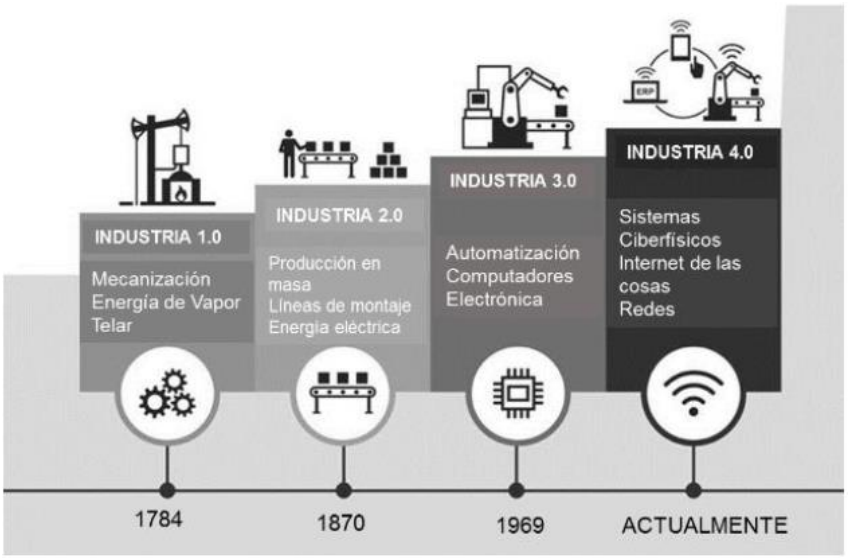


Figura 4. Evolución de la revolución industrial. Fuente: (Rozo,2020)

En la literatura existen definiciones acerca de lo que industria 4.0 significa, Vaidyaa, Ambadb & Bhosle postulan que “La industria 4.0 permite una producción inteligente, eficiente, eficaz, individualizada y personalizada a un costo razonable. Con la ayuda de computadoras más rápidas, maquinas más inteligentes, sensores más pequeños, almacenamiento y transmisión de datos más baratos que podrían hacer que las maquinas o los productos fueran cada día más inteligente para comunicarse

con cada uno y aprender uno de los otros". La Industria 4.0 describe la transición hacia una red de suministro fuertemente centrada en los datos con una amplia integración de las tecnologías de la información (TI) y una mayor automatización mientras se mantiene a los usuarios informados (Lizarralde, et al., 2020). Pero es en el año 2011, donde el concepto de industria 4.0 fue propuesto por primera vez en la Feria de Hannover, Alemania, para referirse a la "Industria Inteligente" (2011), Se describe como la digitalización de los sistemas y de los procesos industriales, en su interconexión mediante el Internet de la Cosas (IoT) e el internet de los servicios, para poder conseguir una mayor individualización y capacidad de flexibilidad de los procesos productivos. "Está compuesta de tecnologías avanzadas, por lo que las soluciones son mucho más flexibles, inteligentes y totalmente autónomas" (Rozo,2020).

Es así como los autores se refieren a la industria 4.0 a un concepto de cambio, que está apoyada en su mayoría con tecnologías para los procesos industriales que han ido evolucionando hasta llegar a ser automatizados, y debido a esto surge la posibilidad de analizar en más detalles el mundo industrial a través de los datos. Este tipo de industria 4.0 que la vemos cada día con más frecuencia, se puede encontrar en grandes compañías de diferentes rubros manufactureros.

¿Cuáles son los principales aspectos de la industria 4.0?

Existen distintos aspectos o tecnologías que conlleva el termino de industria 4.0. Son estas tecnologías (Figura 5) que han revolucionados la fabricas en los procesos claves y que han ayudado a los actuales directivos poder tomar decisiones de manera más precisa y exacta. Para efectos de este proyecto, en la industria maderera no se queda atrás con las tecnologías que posee actualmente, pero es de nuestro interés poder entender términos como IoT o Big data, que posee una total relación entre ellos.

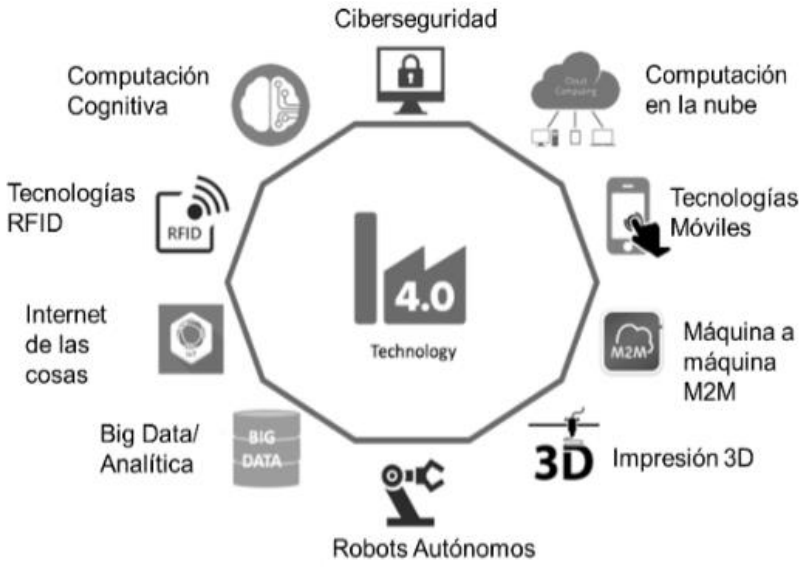


Figura 5. Tecnologías presentes en la Industria 4.0. Fuente: (Saturno, 2017)

2. Machine Learning

Machine Learning es una forma de la IA (Inteligencia Artificial), que permite a un sistema aprender de los datos mediante una programación explicita. Conforme el algoritmo ingiere datos de entrenamiento, es posible producir modelos más precisos basados en datos (Berzal, 2018).

El aprendizaje automático comenzó a tener una mayor relevancia en la década de los 90's, cuándo investigadores y científicos comenzaron a darle más importancia como un subcampo de la IA, probabilidad y la estadística, que funcionan mucho mejor en comparación con el uso de modelos basados en reglas fijas que requiere de mucho más tiempo y esfuerzo manual (Sarkar, Bali, Sharma, 2018).

Uno de los profesores más renombrados de ciencia de datos postulo una de las definiciones más discutidas para referirse a ML:

"Se dice que un programa computacional aprende de la experiencia **E** con respecto a alguna clase de tareas **T** y la medida de rendimiento **P**, si su rendimiento en las tareas **T**, según la medida de **P**, mejora con la experiencia **E**" (Mitchel,1997)

Actualmente estos tres conceptos **T**, **P** y **E** son los principales componentes de cualquier algoritmo de aprendizaje. Al simplificar la definición anterior podemos decir que, el aprendizaje automático consiste en aprender algoritmos que:

- ✓ Mejoran su rendimiento **P**
- ✓ Al ejecutar alguna tarea **T**
- ✓ Con el tiempo, aumenta gracias a la experiencia **E**

Refiriéndose a la capacidad del ordenador el aprendizaje automático proporciona mecanismos mediante los cuales el ordenador es capaz de aprender por sí mismo a resolver un problema. En estos casos el programador se encarga de diseñar un algoritmo de aprendizaje que resulta de cierta forma adecuado para el problema a solucionar pero que es el ordenador el que resuelve el problema aprovechando los datos que tenga acceso y la heurística de aprendizaje incorporadas por el programador (Berzal,2018).

Componentes Principales de Machine Learning

Para poder entender los modelos que se utilizaran para resolver el problema que nos presenta esta investigación definimos algunas clasificaciones más relevantes sobre el aprendizaje automático.

Tipos de aprendizaje automático

Una primera clasificación de las técnicas de aprendizaje automático puede realizarse atendiendo a la filosofía utilizada en el proceso de adquisición del conocimiento (Berzal,2018):

1. Aprendizaje Supervisado

Los métodos o algoritmos de aprendizaje supervisados incluyen algoritmos de aprendizaje que toman muestras de datos de entrada (datos de entrenamiento) y sus resultados correspondientes (datos de salida) con la muestra total de datos durante el proceso de entrenamiento del modelo. El objetivo principal de este tipo de aprendizaje es aprender una relación entre los datos de entrada $\boldsymbol{x}$ y sus correspondientes salidas $\boldsymbol{y}$, sobre la base de múltiples instancias de datos de

entrenamiento. Este aprendizaje es el que se utilizara luego para poder realizar las predicciones de las salidas $\boldsymbol{y}'$, para cualquier nueva muestra de datos de entrada $\boldsymbol{x}'$ que se haya sido parte de la porción de datos de entrenamiento. Este método se denomina supervisado, porque el modelo aprende con una cierta cantidad de datos donde las salidas deseadas se conocen de antemano en la fase de entrenamiento (Sarkar,2018).

Este aprendizaje básicamente trata de modelar la relación entre los datos y sus resultados correspondientes, de manera que pudiéramos predecir las respuestas de salida de los nuevos datos basado en los conocimientos que obtuvo el modelo con anterioridad. En consecuencia, de lo antes mencionado, este tipo de aprendizaje es utilizado ampliamente en análisis predictivo, donde el objetivo principal es predecir alguna respuesta dependiendo de los datos de entrada que le proporcionemos al modelo ML entrenado y supervisado.

Los dos principales modelos de aprendizaje supervisado son:

1. Modelo de Clasificación

Los modelos basados en clasificación tienen como objetivo clave poder predecir las etiquetas de salida o las respuestas de manera categórica, para los datos de entrada en base al modelo que ha pasado por una fase de entrenamiento previo. Los valores de salida son desordenados y discretos por los que, cada respuesta de salida pertenece a una clase o categoría específica y discreta.

Supongamos que, como ejemplo del mundo real, queremos poder saber la predicción del clima, mucho más simple, digamos que queremos saber si mañana estará soleado o lluvioso, basándose en múltiples muestra de datos de entrada de días anteriores que consiste en atributos o características. Como se observa tenemos dos tipos de salida, por tanto, podemos decir que se denomina como un problema de clasificación binaria, con sus respectivas etiquetas o clasificaciones, soleando o lluvioso.

Alguno de los algoritmos de clasificación más populares incluye: Regresión logística, Máquinas de Soporte de Vectores, arboles de decisión, Los K-NN, y

algunos más. No entraremos en mayores detalles sobre este modelo para efectos de este proyecto.

2. Modelo de Regresión

El modelo de aprendizaje automático que cuyo objetivo principal es la estimación de valores continuos, puede denominarse modelos de regresión. Los métodos basados en regresión se entrenan con muestras de datos de entrada que tienen respuestas de salidas que son continuas, valores numéricos a diferencia del modelo de clasificación, donde tenemos categorías o clases discretas como valores de salida. Al igual que el modelo de clasificación, el modelo de regresión también hace uso de los atributos o características de los datos entrada y sus correspondientes valores de salida la que es numérica y continua. El modelo puede predecir las salidas para nuevos datos (datos de prueba) no considerados en los datos de entrenamiento.

Uno de los ejemplos más famosos y simples acerca del modelo de regresión simple es el de predecir el valor de las casas basado en datos relativas a las áreas del terreno en pies cuadrados.

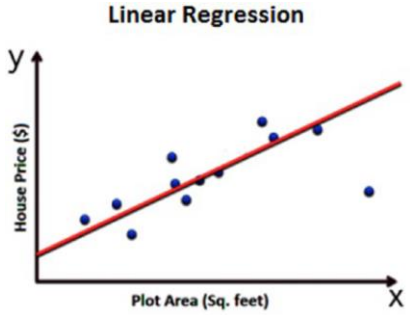


Figura 6: Aprendizaje supervisado: Grafico ejemplo modelo de regresión para la predicción del valor de la casa. Fuente: (Sarkar,2018).

Se intenta determinar si existe alguna relación entre los datos del área de la casa y la variable de salida es el valor de la casa, esta última es la variable que se desea predecir. Por tanto, una vez que aprendamos esta tendencia o la relación representada por la Figura 6, podremos predecir los precios de las casas en el futuro para cualquier área.

La regresión multivariable, intenta modelar datos donde tienen una variable de salida y múltiples variables en forma de un único vector $\boldsymbol{x}$ en lugar de una única variable explicativa. Se desea predecir $\boldsymbol{y}$ basándose en las diferentes características presentes en el vector $\boldsymbol{x}$. Volviendo al ejemplo anterior sobre la predicción del valor de la casa, ahora nos gustaría predecir su valor dependiendo de los valores de entrada del vector $\boldsymbol{x}$ que pueden corresponder a características como el área, el número de dormitorios, número de baños, total de pisos, etc. Basándose en lo anterior el modelo intentara aprender la relación entre cada vector de características y su correspondiente precio de la casa, para que en un futuro el modelo sea capaz de predecir su precio.

2. Aprendizaje No Supervisado

"El aprendizaje no supervisado o aprendizaje por observación o sin profesor, construye descripciones hipótesis o teorías a partir de un conjunto de hechos u observaciones, sin que exista información de ejemplos adicionales sobre la clasificación de la salida esperada del conjunto, ejemplos del conjunto de entrenamiento. Sera el modelo de aprendizaje no supervisado el que decida como han de agruparse los datos del conjunto de entrenamiento dependiendo de los diferentes métodos de agrupamiento o Clustering, decidiendo que tipo de patrones son más interesantes del conjunto." (Berzal,2018)

Los métodos de aprendizaje supervisado requieren datos de entrenamiento donde los resultados que queremos predecir, están disponibles en forma de salidas discretas o valores continuos. Sin embargo, a menudo no tenemos la libertad o la ventaja de tener datos de entrenamiento predefinidos, aun así, necesitamos encontrar patrones de nuestros datos para poder entrenar nuestros modelos. En este escenario, los métodos de aprendizaje no supervisados, son extremadamente efectivos (Sarkar,2018).

Se denominan métodos no supervisados a los modelo o algoritmo que aprenden las estructuras, patrones y relaciones de los datos sin ninguna ayuda o supervisión, proporcionando resultados. El aprendizaje no supervisado está preocupado por

tratar extraer conocimientos o información significativa a partir de datos en ligar de tratar de predecir algún resultado basado en entrenamiento.

Los métodos de aprendizaje no supervisados pueden clasificarse en las siguientes categorías:

1. Clustering
2. Reducción de Dimensionalidad
3. Detección de anomalías
4. Association rule-mining

3. Análisis de Componentes Principales (PCA)

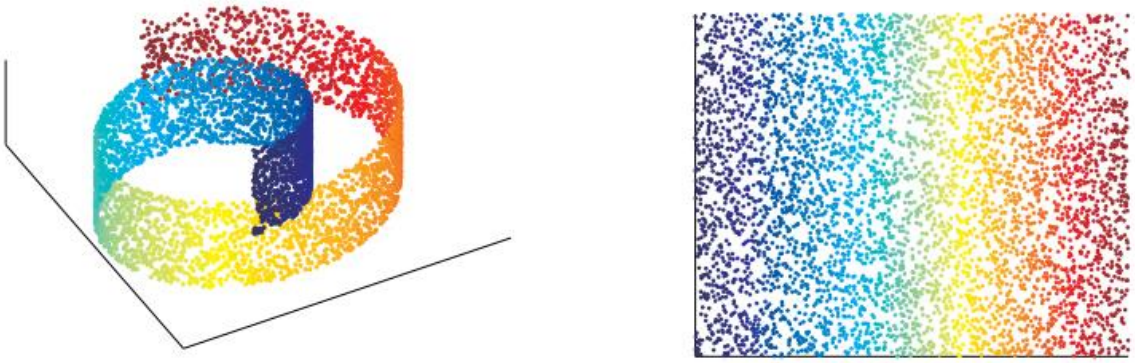
La complejidad de todo modelo tanto de clasificación como de regresión, depende del número de entradas. Esto determina tanto su complejidad temporal y espacial, así como también el número necesario de datos para el entrenamiento de dichos modelos.

El análisis de componentes principales, es método estadístico capaz de seleccionar un subconjunto de las características que de cierta forma representen la gran mayoría del conjunto total de datos. En los entornos en que los datos poseen un gran número de características, a menudo es ventajoso realizar una reducción de dimensión, o encontrar una representación de menor dimensión que preserve alguna de sus principales propiedades (Mohri, et al.,2018).

Del Hecho que existe una muestra con diversas dimensiones a la vez que es capaz de conservar su información. Suponga que existe una muestra con $\boldsymbol{n}$ individuos cada uno con $\boldsymbol{p}$ variables $\boldsymbol{(X_1, X_2, \dots, X_p)}$, es decir, el espacio muestral tiene $\boldsymbol{p}$ dimensiones. PCA permite encontrar un número de factores subyacentes ($\boldsymbol{\mathcal{Z} < p}$) que es capaz de explicar un gran porcentaje que las $\boldsymbol{p}$ variables originales. Donde antes se necesitaban $\boldsymbol{p}$ valores para poder caracterizar a cada individuo, ahora solo

bastan $\mathcal{Z}$ valores. Cada una de esta $\mathcal{Z}$ nuevas variables reciben el nombre de componente principal.

Los beneficios de la reducción de la dimensionalidad suelen ilustrarse mediante datos simulados, a manera de ejemplo se puede ver en la figura 7ª, el conjunto de datos denominado “Rollo Suizo”. En este ejemplo los datos estudiados como entrada son tridimensionales, pero se encuentran en una múltiple bidimensionalidad al “desdoblarlo” como se observa en la figura 7b. Recalcar que, sin embargo, lo múltiplos exactos de baja dimensión se encuentran muy rara vez en la práctica. El ejemplo solo se mostró para ilustrar de maneras más sencilla verificar la eficacia de la reducción de dimensionalidad.



(a) (b)

Figura 7: Data Set “Rollo Suizo”. (a) Representación tridimensional. (b) Representación de bidimensional. (Mohri,2018)

Cálculo de las componentes principales

Para calcular los componentes principales utilizados en PCA, primero se calculará la matriz de Covarianza $\sum$ de los datos (1). Si poseemos d-dimensiones, la matriz será de tamaño ($d \times d$):

$$\sum = X\,X^T \tag{1}$$

Una vez que tenemos esa matriz, obtenemos los k mayores eigenvectores, auto vectores o vectores propios de la matriz de Covarianza, con $k \ll d$, algo que podemos hacer utilizando el algoritmo numérico de tipo iterativo. Esos vectores propios serán nuestros componentes principales. Dado que la matriz de covarianza es simétrica, es diagonalizable y sus vectores propios pueden normalizarse para que sean ortonormales (2):

$$\sum = XX^T = WDW^T \tag{2}$$

Sin embargo, el cálculo de los vectores propios W de la matriz de covarianza puede ser demasiado costoso o puede que nos enfrentemos a un problema que posee una dimensionalidad tan elevada que no sea posible calcular la matriz de Covarianza (Problemas con miles o millones de característica distintas). En casos esos tipos de casos se puede recurrir a otras técnicas estadísticas como son la descomposición en valores singulares pero que para efecto de este proyecto no es necesario profundizar debido al número reducido de características.

Dada una matriz con los de los que disponemos X, de tamaño ($n \times d$), podemos descomponer esa matriz (3) como:

$$X = UDV^T \tag{3}$$

Donde U es una matriz ortogonal del mismo tamaño que nuestra matriz de datos $(n \times d), D$ es una matriz diagonal $(d \times d)$ y V es otra matriz ortogonal $(d \times d)$. Los vectores propios de la matriz de Covarianza $\sum$ serán las columnas de la matriz U, dado que $\sum = XX^T, X = UDV^T \; y \; V^T V = I$, donde se puede deducir que (4):

$$\sum = UD^2U^T \tag{4}$$

Para reducir la dimensionalidad de los datos, basta con que nos quedemos únicamente con los k primeros componentes principales (5), usando $k \ll d$:

$$\check{X} \approx \check{U}\check{D}\check{V}^T \tag{5}$$

Donde es la aproximación de nuestra matriz de datos X $(n \times d)$ como el producto de tres matrices más pequeñas, que significa la reducción de la dimensionalidad: $\check{U}$ $(n \times k)$, $\check{D}$ $(k \times k)$ y $\check{V}$ $(k \times d)$, (Berzal, 2018).

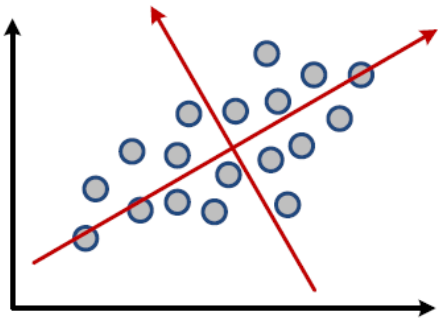


Figura 8: Ejemplo de un análisis de componentes principales (PCA): gráficamente se muestra los dos primeros componentes principales del conjunto de datos. Fuente: (Berzal,2018).

4. Redes Neuronales

Una Red Neuronal Artificial (RNA) es un modelo computacional y una arquitectura de nodos interconectados que simulan neuronas y de cierta forma cómo funcionan en nuestro cerebro. Los nodos e interconexiones son análogos a la red neuronal humana. Una RNA tiene una capa de entrada, una de salida y al menos una capa oculta entre las capas de entrada y salida con interconexiones entre ellas como se puede observar en la figura 9.

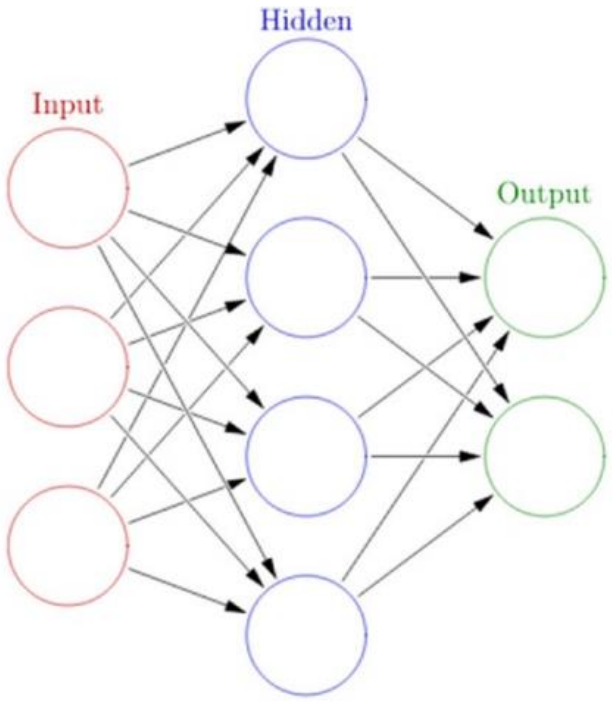


Figura 9: Modelo general de una red neuronal artificial. Fuente: (Sarkar, 2018).

Pero al hablar de modelos y de redes neuronales, debemos entender en primer lugar como se define una neurona.

Neurona

Se puede representar una Neurona desde el punto de vista matemático, el cual la define como un modelo de integración y disparo que se representa como la función (6):

$$V_j = \sum_{i=0}^{n} x_i \; w_{ij} \tag{6}$$

Donde V_j representa el potencial de la 'Membrana' para la j-ésima neurona. x_i la salida actual de la i-ésima neurona y w_{ij} el peso de la conexión desde la neurona i hasta la neurona j.

Este tipo de modelo de neuronal artificial consta de dos etapas (Figura 10). En una primera etapa se combinan las entradas provenientes de otras neuronas teniendo en cuenta los pesos de las sinapsis. El resultado de esta primera etapa es la entrada neta o excitación de la neurona. En la segunda etapa de la neurona, la entrada neta se utilizará directamente para determinar el valor de salida de la neurona, que se propagará a otras neuronas. Nos interesara que el comportamiento de la neurona este sesgado como en una neurona biológica. Matemáticamente hablando podemos añadir ese sesgo como parámetro interno que depende del estado de la neurona. Este sesgo es un parámetro adicional de la neurona al igual que los pesos. De hecho, a menudo se asume que el sesgo no es más que un peso adicional vinculando a una entrada fija con valor 1.

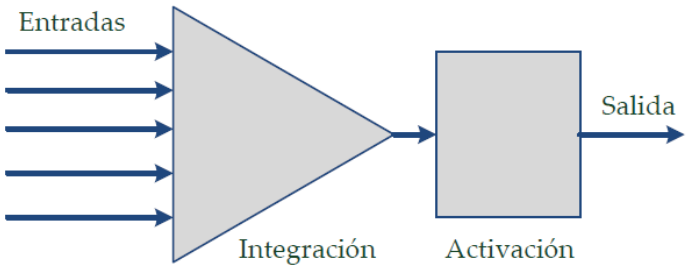


Figura 10: Modelo simplificado Neurona Artificial con integración y activación. Fuente: (Berzal,2018).

Funciones de Activación

Aun que se supone que la combinación de entradas y pesos genera directamente la salida, en el modelo general de neurona artificial, la generación de la salida de la neurona es responsabilidad de una fase de activación independiente de la fase de integración de entradas. Generalmente, la activación de una neurona artificial no entrega entrada neta, sino que, aplica algún tipo de transformación no lineal a esta entrada.

Es una transformación no lineal debido que, al construir redes con múltiples capas, si las neuronas fuesen siempre lineales el resultado final del cálculo realizado por la red neuronal seguiría siendo la función lineal de las entradas. Dicho de otro modo, cualquiera sea la red neuronal compuesta solamente por elementos lineales, se podría sustituir por una red formada únicamente por una capa de neuronas lineales.

De esta forma, llegamos al modelo estándar de una neurona artificial utilizada en todos los modelos de redes neuronales (Figura 11).

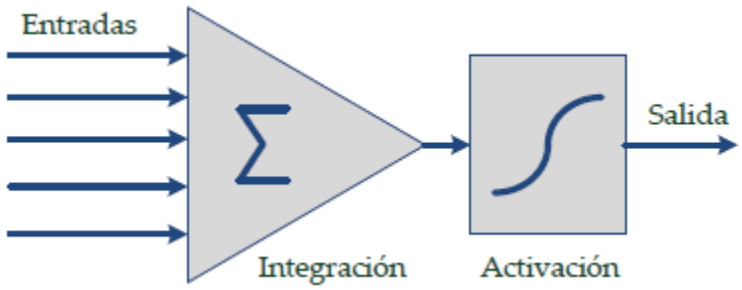


Figura 11: Modelo estándar Neurona Artificial con una función de activación no lineal. Fuente: (Berzal,2018).

Los modelos de neuronas utilizados en las redes neuronales artificiales combinan sus entradas usando pesos que modelan sus conexiones sinápticas y a continuación, le aplican a la entrada neta de la neurona una función de activación o transferencia. Para efectos de este proyecto de título se realizaron pruebas a los distintos modelos con 3 funciones de activación diferentes.

1. Función de activación Lineal Rectificada (ReLU)

Se define la función de activación lineal rectificada, como una función que no requiere realizar una operación aritmética para poder entregar una salida no lineal. Se define de la siguiente forma (7):

$$y = \begin{cases} z, & si\ z \leq 0 \\ 0, & si\ z < 0 \end{cases} \tag{7}$$

La función de activación lineal rectificada utilizada por unidades ReLU definida como:

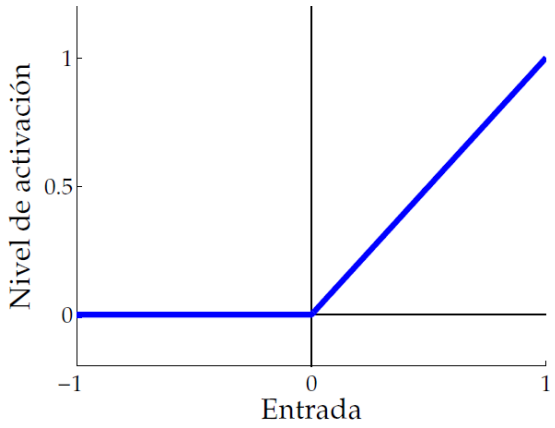


Figura 12: Funcion de activación ReLU. Fuente elaboración propia

2. Función de activación Sigmoidal (Sigmoid)

Nos interesara que la función de activación de una neurona sea, además de no lineal, estrictamente creciente, continua y derivable. La función sigmoidal satisface los requisitos mencionados anteriormente, lo que hace especialmente interesante de utilizarlo en redes neuronales que se entrenan usando Back Propagation.

La función de activación sigmoidal en su expresión aritmética es (8):

$$y = \frac{1}{1 + e^{-z}} \tag{8}$$

La función de activación sigmoidal como función logística tiene un rango de activación [0,1] se expresa:

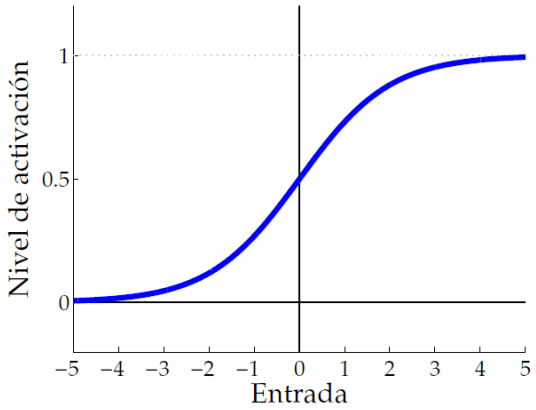


Figura 13: Funcion de activación sigmoidal. Fuente: Elaboración propia.

3. Funcion de activación Tangente Hiperbólica (TanH)

La tangente hiperbólica se utiliza en trigonometría esférica y se define de la siguiente forma (9):

$$y = \frac{1 - e^{-2z}}{1 + e^{-2z}} \tag{9}$$

La función de activación tangente hiperbólica se expresa:

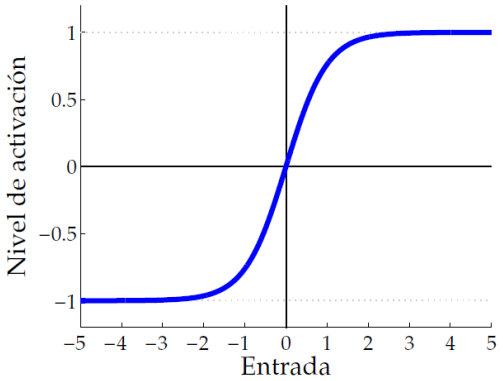


Figura 14: Funcion de activación TanH. Fuente: Elaboración propia.

5. Deep Learning

El Deep Learning y sus técnicas, pertenecen a al subconjunto de técnicas de Machine Learning, que han tomado mayor fuerza o protagonismo en esta década debido al incremento de empresas 4.0, debido a su capacidad de procesar conjunto de datos de gran escala, reducción del costo del hardware y la conectividad de red mejorada. El progreso actual de la investigación en Machine Learning y el procesamiento de la información también han sido factores que contribuyen en gran medida a la prominencia del aprendizaje profundo. A diferencia del Machine Learning tradicional, donde se necesita un experto en el dominio del tema para ayudar a la extracción de características, el Deep Learning puede aprender características automáticamente de un conjunto de datos (Jauro, et al., 2020).

Las técnicas de Deep Learning son capaces de descubrir características de los datos, como las técnicas de aprendizaje de representaciones, pero también son capaces de crear nuevas características a partir de otras. De esta forma, son capaces de no solo de poder identificar características que nos ayuden a discriminar mejor, sino de aumentar el nivel de abstracción al que se trabaja mediante de jerarquías con varios niveles de características. En otras palabras, estas técnicas realizan ingeniería de características de forma completamente automática (Berzal, 2018).

Como menciona Berzal en su libro, las características claves del Deep Learning tienen relación con las redes neuronales artificiales que están especialmente indiciadas para problemas complejos en los que existen multitud de casos particulares. Lo realmente interesante de las redes neuronales artificiales es que son capaces de resolver problemas que ningún programador humano había sido capaz de resolver con la ayuda de un ordenador o computador.

Las principales diferencias que se expresan en literatura sobre Machine Learning y Deep Learning tienen relación con, los nuevos modelos que se agregan a Machine Learning bajo el subconjunto llamado Deep Learning, con sus algoritmos basados en redes neuronales y gracias a las capacidades tecnológicas que existen hoy en

día, son capaces de poder utilizar, los algoritmos Deep Learning basados en redes neuronales que tienen la capacidad de poseer más capas ocultas las que pueden o no estar totalmente conectadas entre ellas y no teniendo esta posibilidad con redes neuronales de Machine Learning que solo pueden tener una capa oculta totalmente conectada(Kim & Jeong, 2020).

Gracias a las capacidades tecnológicas, surgieron con estos modelos Deep Learning, funciones de activación y optimizadores que mejoraran las redes neuronales clásicas y fueron adoptadas a estas nuevas tecnologías (Djellai & Adda, 2020).

Casi todos los algoritmos de Deep Learning se pueden describir como instancias particulares de una receta, que aplicamos para resolver un problema concreto de Machine Learning:

- Recopilar conjunto de datos asociados al problema (un gran número facilita en gran medida).
- Funcion de perdida [Loss Function] apropiada al problema.
- Selección modelo de red neuronal y establecer hiper parámetros.
- Aplicación de algoritmo de optimización para minimizar la función de coste, ajustando los parámetros de la red.

Modelos

Para efecto de este proyecto de título, se explicarán los siguientes modelos de Machine Learning que son los que serán comparados entre ellos para la elección del modelo más significativo y preciso para la resolución del problema, esto corresponde a:

1. Multi Layer Perceptron (MLP).

Es la más clásica de las RN. Un perceptrón multicapa, también conocido como MLP, es una red neuronal artificial totalmente conectada y alimentada con al menos tres capas (entrada, salida y al menos una capa oculta) donde cada capa es completamente conecta a la capa adyacente (figura 15). Cada neurona suele ser una unidad de procesamiento no lineal (Sarkar, 2018).

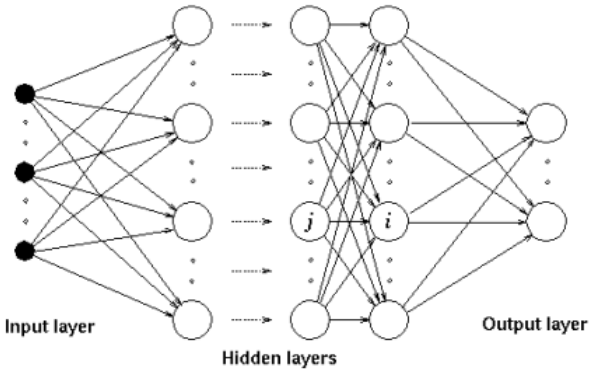


Figura 15: Red Neuronal Artificial. Fuente: (March, 2020)

El modelo de MLP usado tiene como variables x_i las entradas de la red, y_j la salida de la capa oculta y z_k las salidas de la capa final de la red; t_j las salidas objetivo, w_{ij} son los pesos de la capa oculta y θ_j sus umbrales o bias, w'_{kj}los pesos de la capa de salida y θ'_k sus umbrales o bias. La operación del MLP se expresa matemáticamente con la siguiente función (10): (Montecinos,2020):

$$z_k = \sum_j w'_{kj}\, y_j - \theta'_i = \sum_j w'_{kj}\ f\left(\sum_i w_{ij}\, x_i - \theta_j\right) - \theta'_i \tag{10}$$

2. Red Neuronal Recurrente

La Red Neuronal Recurrente (RNN) es un tipo especial de red neuronal artificial que permite información persistente basada en el conocimiento del pasado mediante el uso de un tipo de arquitectura en bucle (figura 16). Estas redes en bucles se llaman recurrentes debido que realiza las mismas operaciones y cálculos para todos los elementos en una misma secuencia de datos de entradas. Los RNN tiene una memoria que es capaz de capturar información de secuencias de datos pasada e inyectarlas a la neurona.

Suponer que s_t es la salida la capa oculta en el momento t , x_t es la entrada del modelo RNN en el momento t, o_t es la salida del modelo en el momento t, Wes el parámetro de peso entre la capa de entrada y la oculta. V es el parámetro entre la capa oculta y la salida, U es el parámetro de peso entre la capa de entrada y la oculta, el sesgo de la capa oculta es b , con una función de activación de la capa oculta σ, el sesgo de la capa de salida c y la función de activación de la capa de salida μ (Ren, et al.,2020).

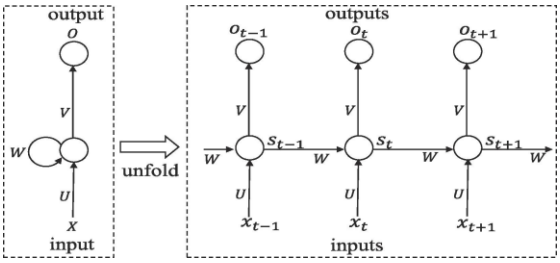


Figura 16: Red Neuronal RNN capa oculta. Fuente: (Ren, et al.,2020)

$$s_t = \sigma\,(U_{x_t} + W_{s_{t-1}} + b) \tag{11}$$

$$o_t = \mu\,(V_{s_t} + c) \tag{12}$$

Luego al sustituir la ecuación (11) por (12) un y otra vez, se puede obtener la siguiente fórmula:

$$o_t = \mu\,(V_{s_t} + c) \tag{13}$$

$$= \mu\, V\left(\sigma\left(U_{x_t} + W_\sigma\left(U_{x_{t-1}} + W_\sigma\left(U_{x_{t-2}} + W_\sigma\left(U_{x_{t-2}} + \ldots\right)\right)\right)\right)\right) \tag{14}$$

De la formula (14) se puede concluir que la salida del modelo se ve afectada por los valores de entrada anteriores, es por esto, que el modelo es capaz de recordar valores de entradas anteriores.

3. Soporte de Regresión Vectorial

El Soporte de Regresión Vectorial (SVR) es un algoritmo de Machine Learning supervisado para problemas de regresión óptimos, con maximización del margen nominal del conjunto de entrenamiento. El uso de este modelo es aplicable cuando los tipos de datos son multivariables o complejos. La ecuación (15) define el modelo de la siguiente manera (Panahi, et al., 2020):

$$f(x) = \langle \omega , \phi(x) \rangle + b \tag{15}$$

Donde $f(x)$ es la forma general de la función de regresión no lineal y $\langle ., . \rangle$ representa el producto interno de dos vectores. $\phi(x)$ denota un mapeo no lineal con el espacio dado x, y b y ω son pesos escalares y vectoriales, respectivamente.

Su función es la de interpretar en un hiper plano de dimensión reducida a la real, una regresión, para interpretar esta función se debe:

$$Min: \left[\frac{1}{2} \|\omega\|^2 + C \sum_{i=1}^{n} (\xi_i + \xi_i^*) \right] \tag{16}$$

$$Sujeto\ a: \begin{cases} y_i - (\omega\varphi(x_i) + b_i) \leq \varepsilon + \xi_i \\ (\omega\varphi(x_i) + b_i) - y_i \leq \varepsilon + \xi_i^* \\ \xi_i , \xi_i^* \geq 0 \end{cases} \tag{17}$$

Donde ξ_i , ξ_i^* son variables de holgura y, C denota la variable de penalización ε representa la función de perdida insensible que muestra la calidad de

aproximación. La ecuación (17) se resuelve utilizando la función Lagrangiana por lo tanto la función de regresión final (18) se puede expresar como:

$$f(x) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*)\, k(x, x_i) + b \tag{18}$$

Donde $\alpha_i\,, \alpha_i^*$ son multiplicadores Lagrange y $k(x, x_i) = \langle \phi(x), \phi(x_i) \rangle$ es la función Kernel.

Entrenamiento de modelos basados en redes neuronales

Back Propagation

La retro propagación es un algoritmo de propagación de errores que, dado los errores observados en la capa de salida, propaga esos errores hace atrás en la red para ir ajustando sus parámetros internos. La forma en que se ajusta estos parámetros depende de la derivada de la función de activación de las neuronas (Gamage & Samarabandu, 2020).

El algoritmo de retro propagación es una técnica para entrenar a las redes neuronales artificiales y llevo un resurgimiento de la popularidad de las redes neuronales en los años 80's. El algoritmo posee dos etapas principalmente: Propagación y actualización de peso, se describe brevemente (Sarkar,2018):

1. Propagación:
    a. Los vectores de muestreo de los datos de entrada se propagan hacia adelante a través de la red neuronal para generar los valores de la capa de salida.
    b. Comparar el vector de salida generado o valores de predicción con el vector de salida real o deseado.
    c. Calcular la diferencia de error en las unidades de salida (cualquier técnica).
    d. Reproducir los valores de error para generar deltas en casa neurona.

2. Actualización del peso:
    a. Calcular los gradientes de peso multiplicado el delta de la salida (error) y la entrada de la activación.
    b. Utiliza la tasa de aprendizaje para determinar el porcentaje del gradiente que debe restarse del peso original y actualizar el peso de las neuronas.

Estas dos etapas se repiten muchas veces con múltiples iteraciones/épocas hasta que conseguimos cada vez mejores resultados con respecto a la predicción de los datos. La retro propagación se utiliza junto a algoritmos de optimización o funciones estocásticas.

La función para poder calcular el Back Propagation depende del Error de una red neuronal. Esta función (19) se define como gradiente de error y se expresa de la siguiente manera: (Kim, et al.,2020):

$$E_k = \frac{1}{2}\sum_k (\, F_k - f(\sum_{i=1}^{l} w_{ij}x_i - \, d_j) - \, d_k))^2 \tag{19}$$

Algoritmos de optimización

Muchas técnicas de aprendizaje automático se basan en convertir el proceso de aprendizaje en un problema de optimización que podemos resolver mediante herramientas de cálculo numérico. En el caso particular de las redes neuronales, sabemos cómo calcular el gradiente de error de forma eficiente utilizando back propagation (Berzal, 2018).

Para convertir un problema de aprendizaje en un problema de optimización, se define una función de error, coste o perdida de la función (20):

$$f(x): \mathbb{R}^n \rightarrow \mathbb{R} \tag{20}$$

Como pretendemos reducir el error, que depende del conjunto x de parámetros de nuestro modelo, nuestro problema de aprendizaje se reduce al siguiente problema de minimización (21):

$$x^* = \arg\min f(x) \tag{21}$$

Tal como mencionamos, muchas técnicas de optimización se basan en seleccione primero una dirección en la que optimizar una función y a continuación, determinar cuál es el cambio idóneo en la dirección que nos permite minimizar la función de error. Para efectos de este proyecto de título nos enfocamos en solo dos con entradas multivariables, para los modelos seleccionados antes mencionados.

1. Gradiente descendente estocástico (SGD)

El gradiente es utilizado para actualizar los valores de los parámetros en una red neuronal, es calculado a partir del conjunto de datos de entrenamiento completo, a partir de una muestra del conjunto de entrenamiento. Cuanto menor sea la muestra utilizada para el cálculo del gradiente, mayor será el error cometido por la estimación. La expresión de para el cálculo del SDG se expresa en la siguiente función (22) (Ge, et al.,2015):

$$\Delta x = -\eta(\nabla_x f(x) + \varepsilon) \tag{22}$$

Donde ε representa el error en nuestra estimación del gradiente, η es la taza de aprendizaje del sistema.

2. Adaptive Moment Estimation (Adam)

Es un algoritmo de optimización basado en el grado de funciones objetivas estocásticas y estimaciones adaptativas de momentos de menor orden. Este algoritmo es adecuado para problemas donde existen grandes cantidades de datos y/o parámetros (Kingma & Lei Ba, 2017). Adam se define como dos medias móviles de los gradientes (23) y (24):

$$m(t) = \beta_1\, m(t-1) + (1-\beta_1)\, g(t) \tag{23}$$

$$v(t) = \beta_2\, v(t-1) + (1-\beta_2)\, g^2(t) \tag{24}$$

Donde $m(t)$ es una estimación del primero momento del gradiente (su media). Y $v(t)$ es una estimación del segundo momento del gradiente (su varianza). Los parámetros que controlan las medias móviles como lo son, β_1 y β_2 con valores cercanos a 0. Adam al corregir el sesgo hacia 0, a partir de las fórmulas (25) y (26) se puede encontrar la fórmula de la función (27) de optimización Adam:

$$\widehat{m}(t) = \frac{m(t)}{1-\beta_1^t} \tag{25}$$

$$\hat{v}(t) = \frac{v(t)}{1-\beta_2^t} \tag{26}$$

$$\Delta x = -\frac{\eta}{\sqrt{\hat{v}(t)} + \varepsilon}\, \widehat{m}(t) \tag{27}$$

Donde η es una taza de aprendizaje inicial (hiper parámetro del algoritmo), ε se utiliza para evitar que la función se indetermine con valores del orden 10^{-8}.

Tasa de aprendizaje

El valor de la taza de aprendizaje se expresa en literatura con la letra η controla el tamaño del cambio de los pesos en cada iteración de las funciones de optimización en el modelo. Un ritmo de aprendizaje demasiado pequeño puede ocasionar una disminución importante en la velocidad de convergencia del modelo y la posibilidad de acabar atrapado en un mínimo local. A su vez un ritmo de aprendizaje grande, puede conducir a inestabilidades en la función de error, lo cual se evitará que se produzca la convergencia esperada debido que nunca dará saltos muy grandes en torno del mínimo y nunca poder llegar a alcanzarlo (Marín, 2012).

Se recomienda elegir un ritmo o tasa de aprendizaje lo más grande posible sin que provoque oscilaciones demasiado grandes. En general, la taza suele estar entre $0{,}5$ y $1e^{-4}$

Evaluación de resultados

Independiente del tipo de técnica de aprendizaje que se desee utilizar para la resolución de problemas a los que nos enfrentemos, demos ser capaces de poder medir en cierta forma el resultado del proceso de aprendizaje del modelo. Debemos entender que un modelo no es más que una simplificación de la realidad y por ende estará siempre sujeto a errores. Por tanto, se debe evaluar la calidad del modelo obtenido para poder observar si realmente es el que nos sirva en la práctica (Barzal, 2018).

Para la evaluación de los modelos que se utilizan en este proyecto de título, se utilizaron métricas asociadas a Machine Learning y sobre modelos de entrenamiento supervisado.

Métricas

Una vez construido el modelo a partir de un conjunto de datos, es imperioso poder evaluar su calidad, para decidir en la práctica si utilizar el modelo y poder diferenciarlo de otros.

Las métricas se pueden diferenciar por modelos de regresión y modelos de clasificación, cada uno posee métricas distintas que mide aspectos diferentes para cada uno de ellos. Para efectos de este proyecto nos enfocaremos en las métricas de regresión.

1. Métricas de Clasificación

Cuando se aplica un modelo de clasificación a un conjunto de datos previamente etiquetado, podemos ver fácilmente como etiqueta los diferentes ejemplos nuestro clasificador, Solo existen cuatro posibilidades, que podemos representar en una matriz de contingencia o matriz de confusión (figura 17).

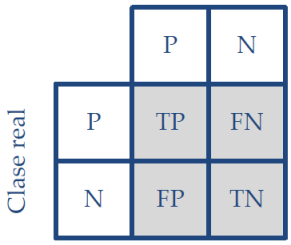


Figura 17: Matriz de Confusión. Fuente:(Berzal,2018)

- Verdaderos positivos (TP)
- Falsos Positivos (FP)
- Falsos Negativos (FN)
- Verdaderos Negativos (TN)

Los cuatro casos posibles se recogen de la matriz de confusión (Figura 14) de tamaño 2 x 2 en el caso de los problemas de clasificación binaria. De la matriz se desglosan diferentes métricas que no estudiaremos con profundidad, pero mencionaremos:

- Precisión o Accuracy
- Recall
- F-Score
- Main Reciprocal Rank

2. Métricas de Regresión

Al construir modelos que sean capaces de predecir el valor de un variable numérica, que toma valores dentro de un rango continuo, diremos que se utilizaran modelos de regresión para solucionar este tipo de problemas. Al igual que el modelo de clasificación este tipo de modelo posee métricas referentes a la predicción de valores continuos, algunos de los más relevantes para este proyecto son:

1. Mean Squared Error (MSE)

La medida de error más utilizada para estimar la calidad de un modelo de predicción cuantitativa es el error cuadrático medio (MSE), que aplicamos sobre las N muestras del conjunto de datos sobre el que estamos estimando la calidad de nuestro modelo cuantitativo, se expresa de la siguiente forma la función de error cuadrático medio (28):

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(f(x) - y)^2 \tag{28}$$

2. Root Mean Squared Error ($RMSE$)

Representa la desviación de las diferencias entre los valores de las predicciones $f(x)$ y los valores observados y. Esas diferencias se denominan residuos cuando se está calculando sobre el conjunto de datos. La función (29) que define RMSE se expresa como:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(f(x) - y)^2} \tag{29}$$

3. Coeficiente de correlación lineal de Pearson (r)

Es un estimador muestral utilizado para evaluar asociación lineal entre dos conjuntos de datos o variables $X\ e\ Y$. Se define como un índice que mide si los puntos tienen tendencia a disponerse en una línea recta. Puede tomar valores entre -1 y 1.

Este método estadístico utiliza la media y varianza de los puntos tomados en el conjunto de datos. Función (30) define la covarianza muestral entre $X\ e\ Y$ dividida por el producto de sus desviaciones (Laguna, 2019):

$$Pearson\ (r) = \frac{S_{XY}}{S_X\ S_Y} \tag{30}$$

4.Coeficiente de correlación (r^2)

Evalúa el grado en el que el modelo de regresión lineal explica las variaciones que se producen entre el conjunto de observaciones encontradas producidas en la variable dependiente de estas.

Este coeficiente nos indica el grado de ajuste de la recta de regresión a los valores de la muestra, y se definen como el porcentaje de la variabilidad total de la variable dependiente Y que es explicada por la recta de regresión (31). La cantidad adimensional que solo puede tomar valores entre 0 y 1, entre más cercano a uno mayor será la fuerza de asociación entre ambas variables (Laguna, 2019).

$$Correlacion\ (r^2) = \ (\frac{S_{XY}}{S_X\ S_Y})^2 \tag{31}$$

Capítulo 3: Metodología

En este capítulo, se describe los términos, datos, instrumentos y procedimientos utilizados para entender cómo lograr los objetivos de la investigación para la resolución de la problemática.

1. Definiciones, Siglas y Abreviaciones

Definiciones generales orientas al proyecto se describen a continuación:

- **RNN:** Redes Neuronales Recurrentes
- **MLP:** Multi Layer Perceptron
- **SVR:** Maquinas de Soporte de Regresión Vectorial
- **VnN:** Valor no numérico
- **Python:** Lenguaje de programación multiparadigma, ya que soporta orientación a objetos
- **Anaconda:** Es una distribución gratuita y de código abierto de los lenguajes de programación, Python y R para computación científica.
- **Spyder:** Es un entorno de desarrollo integrado (IDE) multiplataforma de código abierto para programación científica enfocado en Python. Spyder integra varios paquetes destacadas en la pila científica de Python.
- **Librería Estándar:** Es un conjunto de módulos y paquetes que se distribuyen junto con Python.
- **Paquete:** Carpeta que contiene varios módulos.
- **Modulo:** Fichero que contiene Scripts y es ejecutable.
- **Script:** Definición de funciones y variables en lenguaje en Python
- **Entorno virtual Anaconda:** Es un espacio independiente a la instalación normal del Software, el objetivo es asilar recursos y librerías. Permite tener entornos virtuales con diferentes versiones de Python y librerías.

- **Outlier:** En estadística, este concepto significa un valor atípico en un grupo de datos, que difiere significativamente de otras observaciones.

2. Desarrollo

Para el desarrollo de este proyecto de título, se utilizó la metodología CRISP-DM mencionado en el capítulo 1, que menciona pasos a seguir para un proyecto de minería de datos como muestra la figura 18, adoptado para este proyecto con modelos Machine Learning a estudiar y comparar.

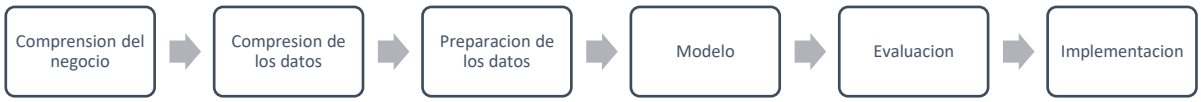


Figura 18: Fases CRISP-DM. Fuente: Elaboración propia.

Para resolver la problemática, que es descrita como menciona la Figura (1) en una caja negra con una entrada al sistema, descrita como variables, características o factores que posee el proceso de aserrado en las forestales y una salida que se desea predecir, la velocidad de alimentación descrita en el capítulo 1.

Este Proyecto contemplo la implementación de tres modelos Machine Learning en lenguaje de programación Python, para su entrenamiento y comparación en el estudio de predicción de valores continuos. Permitiendo al usuario poder ingresar distintos data set, entrenar modelos modificando sus híper parámetros vistos en el capítulo anterior, predecir valores y la visualización de resultados.

Lo que se propuso en este proyecto para el desarrollo de la solución de la problemática planteada por el cliente, busco la elección del modelo que obtuviera un mayor porcentaje de correlación entre los valores reales entregados por los datos de salida de prueba y los valores de predicción. Para poder obtener un modelo de predicción que sea capaz de predecir con una baja taza de correlación, velocidades de alimentación en el proceso de aserrado, para las decisiones sobre las funciones que desempeña una máquina de aserrío de alta productividad.

Este proyecto, si bien no contemplo un sistema que agrupe todos los modelos, la implementación de cada modelo fue explicada, para que el usuario sea capaz de utilizarla y probar los modelos con distintos datos. Para el desarrollo de la solución se requieren distintas herramientas descritas a continuación:

Herramientas para el desarrollo

Para el Desarrollo e implementación de los modelos de Machine Learning, se utilizaron distintas herramientas que aportaron durante el proyecto, las que se mencionan a continuación:

1. Hardware

Servidor de pruebas:

- Equipo (1):
  - Procesador: Intel Core i5 2.40 GHz
  - Tipo de sistema: 64 bits
  - RAM: 8 GB
  - Disco Duro: 1 TB
  - Disco Solido: 256 GB
  - Sistema Operativo: Windows 10 Pro

- Equipo (2):
  - Procesador: Intel Core i5 2.40 GHz
  - Tipo de sistema: 64 bits
  - RAM: 4 GB
  - Disco Duro: 750 GB
  - Sistema Operativo: Windows 10 pro

2. Hardware

El Servidor de pruebas se realizó en un Entorno Anaconda.

- Software Anaconda versión 5.2.0
- Lenguaje de programación Python versión 3.5.5
- Editor de código e IDE Spyder versión 3.2.8

- Librerías Python:
  - Numpy versión 1.14.0
  - Pandas versión 0.23.0
  - Scikit-Learn versión 0.19.1
  - Tensor Flow versión 1.10
  - Keras versión 2.2.2
  - Matplotlib versión 2.2.2
  - Nltk versión 3.3.0
  - Scipy: versión 1.1.0

- Administrador de planilla de datos Excel versión 2019 plus

Procedimiento

Como mencionamos anteriormente, para este desarrollo de este proyecto de título, nos guiamos por una metodología CRISP-DM, donde seguimos un orden planteado en la figura (18). Como primer punto, sobre la comprensión del negocio, se menciona que la problemática descrita por el cliente para este proyecto, en el capítulo 1, es el tema central a desglosar, que se obtuvieron los objetivos y planteo soluciones al problema. Algunas variables de los datos se muestran a continuación (tabla 1):

id	Time Stamp	Tracción Montaje DER	Tracción Montaje IZQ	Rango Pmont IZQ	Rango Pmont DER	Rango Vib IZQ	Rango Vib DER	. . .
831 734	12/6/2020 10:42	155.67	162.964	Tracción Montaje IZQ Normal	Tracción Montaje DER Normal	Vibración SH IZQ Normal	Vibración SH DER Normal	
831 735	12/6/2020 10:42	155.67	178.964	Tracción Montaje IZQ Normal	Tracción Montaje DER Normal	Vibración SH IZQ Normal	Vibración SH DER Normal	
. . .								

Tabla 1: Pequeño fragmento de los datos de entrada.

Se debe tener encuentra que el problema posee más de 70 características de entrada para el sistema, entre variables numéricas, caracteres y registros de tiempo, con más de 60.000 datos aproximadamente.

Preparación de datos

Una vez que el cliente hizo la entrega de los datos, se firmó una carta de confidencialidad por motivos de divulgación, estos datos fueron entregados desde una base de datos empresarial y exportadas a una planilla Excel.

Para este proyecto de título se le realizo a los datos un pre procesamiento para mejorar en el entrenamiento de los modelos. Este pre procesamiento se realizó de dos formas:

- Manual

De forma manual se inspecciono los datos, el primer cambio en los datos fue la modificación de las variables no numéricas a variables numéricas en forma representativa de los datos, como por ejemplo las variables de la tabla (1):

Rango Pmont IZQ	Rango Pmont DER	Rango Vib IZQ	Rango Vib DER
Tracción Montaje IZQ Normal	Tracción Montaje DER Normal	Vibración SH IZQ Normal	Vibración SH DER Normal
Tracción Montaje IZQ Normal	Tracción Montaje DER Normal	Vibración SH IZQ Anormal	Vibración SH DER Normal

Tabla 2: Fragmento de variables no numéricas de la tabla (1).

Donde se reemplazaron los datos que se observaban a través de caracteres, poseían una cierta representación binaria, por lo tanto, los valores que poseen una salida “Normal” se les asigno el valor numérico de 1, y a los valores “Anormales” se les asigno el valor de 0.

Luego se eliminaron las variables que tuvieran columnas de valores iguales, o que su desviación estándar tuviera un valor de 0, debido que no tenía una relevancia para el estudio, si no sufrían cambios en el amplio número de datos por columna. A su vez se buscaron outliers, se eliminaron los índices y ordenaron los valores del tiempo. Se conservaron los datos en una planilla Excel para su importación. Para poder ratificar las columnas con valores numéricos igual 0 en su desviación estándar se realizó una inspección visual a través de una matriz de correlación eliminando alrededor de 20 variables. Entendiendo que el data set es limitado en comparación al gran número de datos que posee la empresa, donde para este estudio existían

variables constantes, pero con un número más grande de datos, existe probabilidades que se encuentren datos diferentes.

- Automática

Con el paquete de datos de Python Pandas se pudo importar un archivo con extensión “xlsx” correspondiendo a un archivo Excel, donde se almacenan los datos pre tratados manualmente, para poder utilizar este fichero y traspasar los datos a una variable local de tipo DataFrame.

Se implementa el método estadístico de Análisis de Componentes Principales en Python, para la reducción de dimensionalidad de las variables conservando sus principales propiedades como es descrito en el capítulo 3. Gracias a la función Decomposition() del paquete de datos de Python Sklearn, creamos la función PCA y reducimos la dimensionalidad con un 85% de representatividad del conjunto de datos, reduciendo a un más las variables de los datos originales, disminuyendo de 50 aproximadamente a 14 variables representativas para el total de datos. Para al finalizar realizamos una estandarización de datos con la función StandardScaler() que tiene un rango de estandarización entre [0,1], el cual se aplicó a los datos una vez realizado el PCA y al vector de salida de los datos.

Para finalizar se creó una función que pudiera dividir el conjunto total de datos en dos, los datos de prueba y los datos de entrenamiento, cada uno con sus respectivas salidas y con un porcentaje 20 y 80 respectivamente que se puede observar los datos aproximadamente (Tabla 3). Para esta función se utilizó las funciones Iloc() de Pandas, que permite dividir un DataFrame en listas de arreglos.

Datos de entrenamiento	Datos de Prueba
58.000	**12.000**

Tabla 3: Datos de entrenamiento y prueba.

Modelo

Se escogieron 3 modelos Machine Learning para la comparación y posterior elección del mejor modelo para solucionar el problema de regresión de valores continuos planteado. De esta forma se crearon funciones en ficheros distintos, pero

con las funciones de pre tratamiento de datos igual para cada uno de ellos, de esta forma la implementación tanto del modelo RNN y MLP se utilizó con la función Sequential() de los modelos que están en el paquete de datos Keras de Tensor Flow para Python. Y se diferencian uno de otro por las capas que posee cada red neuronal distinta en cada modelo, los hiper parámetros se muestran en las tablas a continuación:

Numero de capas	Tipo de capa	Neuronas por capa	Funcion de Activación	Optimizador	Tasa de aprendizaje	Épocas de entrenamiento	Función de perdida

Tabla 4: Hiper parámetros elegidos para MLP.

Numero de capas	Tipo de capa	Neuronas por capa	Funcion de activación	Time Step	Optimizador	Funcion de perdida	Épocas	Tasa de aprendizaje

Tabla 5: Hiper parámetros elegidos para RNN.

El modelo SVR se implementó al igual como los dos modelos anteriores, en una función dentro del fichero, gracias a la función svr() del modelo SVM del paquete de datos Sklearn de Python. Para la definición de los valores que toma los hiper parámetros descrito en la tabla (6), se utilizaron los visto en literatura y los por defecto de las funciones, además dentro de la inicialización de la capa de inicio los pesos de cada neurona se inicializaron con valores aleatorios.

Constante C	Kernel	Épsilon

Tabla 6: Hiper parámetros elegidos para SVR.

Evaluación

Los resultados de los modelos serán presentados en el siguiente capítulo, los que serán representados por métricas para modelos de regresión no paramétricos y el coeficiente de correlación descrito en el capítulo 2. Una vez creados y entrenados los modelos, se definirá la correlación que existe entre la salida de los datos de prueba, descritos como “datos reales” y los valores de predicción que, del modelo en ambos casos los datos están formados por un vector o lista, gracias a esto se

pueden aplicar la correlación que existe entre la lista de datos. Los resultados obtenidos en el aparto siguiente permitirán entregar un resultado al cliente.

Capítulo 4: Resultados

A continuación, se presentan los resultados obtenidos que se exponen en dos categorías, correspondiendo a los resultados del preprocesamiento de datos y los modelos utilizados. Tanto las métricas como los híper parámetros asociados a los modelos, están descritos en el capítulo anterior.

1. Preprocesamiento de datos

En este punto se exponen los resultados de la experimentación realizada al utilizar una matriz de correlación para poder obtener visualmente las variables que tenían una desviación estándar de valor numérico 0, se puede ver gráficamente una correlación de Pearson entre el total de las variables de los datos, en un mapa de calor que se observa a continuación (Figura 19). Se debe tener en cuenta como se mencionó en el capítulo 2 sobre el coeficiente de Pearson, que posee un rango entre -1 y 1, descritos por los colores azul y amarillo propiamente tal, y se observa en el mapa de calor que existen zonas donde el color no corresponde a ninguno de los descritos, lo que demuestra debido que no está dentro de la escala, poder ser evaluado y lo que indica que no existe linealidad entre las variables de este tipo.

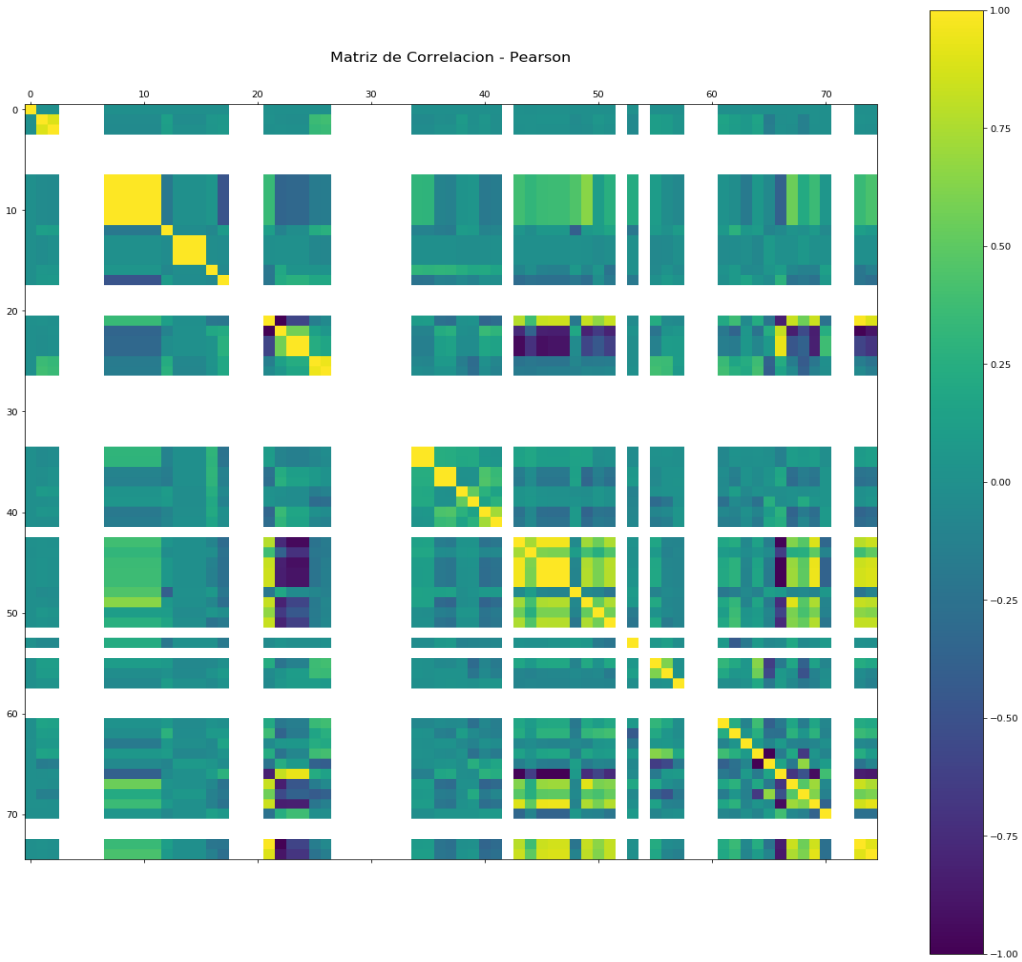


Figura 19. Matriz de correlación Pearson para el total de los datos. Fuente: elaboración propia.

Una vez que se realizó una reducción de dimensionalidad manualmente y se eliminaron los datos que no estaban en la escala de linealidad de la matriz de correlación, se pasó a la etapa automática del preprocesamiento de datos. Los datos que se ingresaron al sistema, se les realizo un PCA para reducir aún más su dimensionalidad eliminando los valores que son linealmente dependientes entre ellos, a continuación se muestra un mapa de calor, que muestra gráficamente la matriz de correlación de Pearson entre los datos que ingresaron al sistema (Figura 20) y a continuación uno mapa de calor (Figura 21) que explica los que quedaron al reducir la dimensionalidad y quedar solo con los datos linealmente independiente entre ellos obteniendo 15 dimensiones o valores que entraran a los modelos :

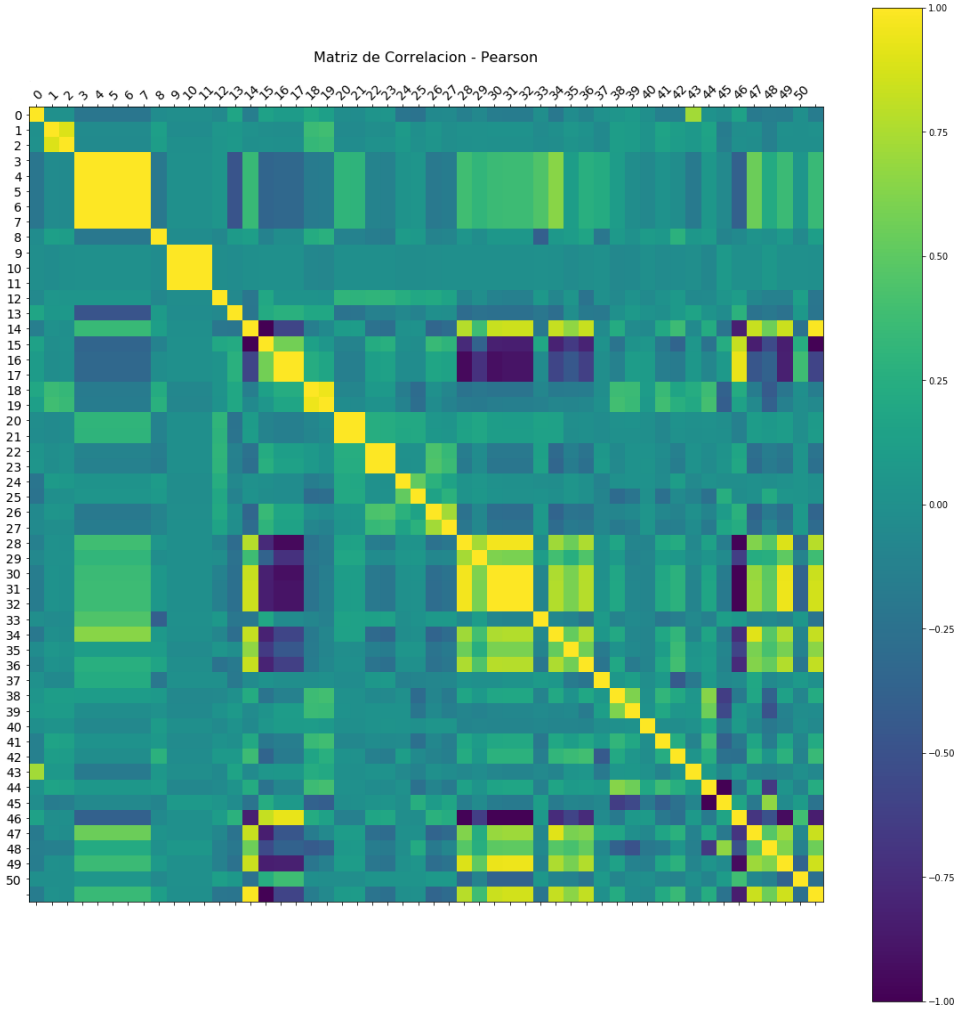


Figura 20: Matriz de correlación Pearson para los datos de entrada al sistema. Fuente: elaboración propia

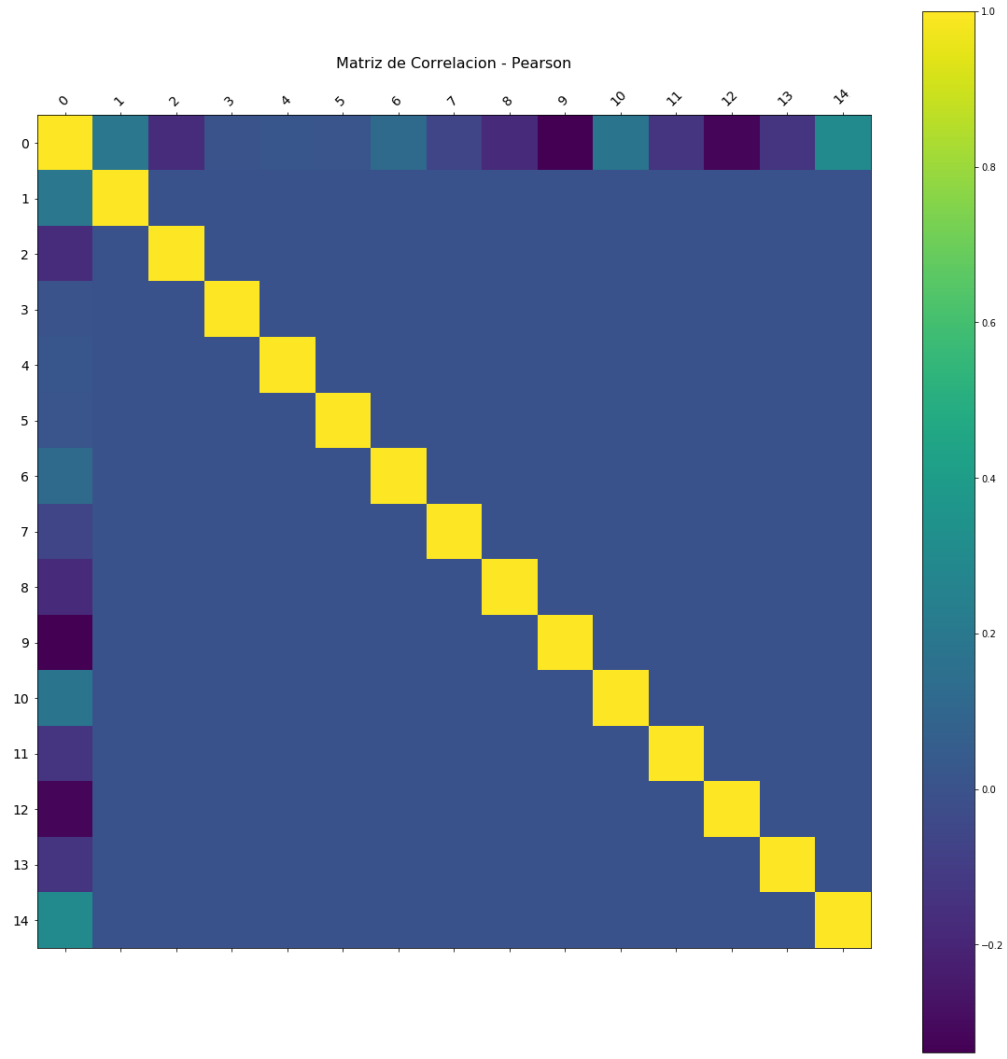


Figura 21: Matriz de correlación Pearson para los datos luego de realizar PCA. Fuente: Elaboración propia.

2. Modelos

A continuación, se presentan los resultados para los experimentos realizados con los modelos que en el capítulo 2 se describieron. Para efecto de la comparación de modelos se utilizaron para todos los experimentos el mismo Data Set, con el mismo número de datos.

Los hiper parámetros utilizados en cada modelo se describieron en el capítulo anterior.

Como métrica de éxito del entrenamiento del modelo se utilizará la métrica de correlación (r^2) que esta definida por la correlación entre los valores de salida de

predicción hecha por el modelo y los resultados reales correspondientes por los datos de prueba. El siguiente grafico de dispersión muestra, una comparación entre los valores antes mencionados. Este tipo de grafico se aplica a todos los modelos que se analizaran, y se muestra a modo de ejemplo (Figura 22).

El grafico de dispersión (Figura 22) se expresa como un 92% de correlación entre ambas salidas (color verde para los valores de predicción hechos por el modelo y color azul para las salidas reales) en un modelo MLP realizado a un universo de 12.000 datos con salidas que varían entre 0 y 1.

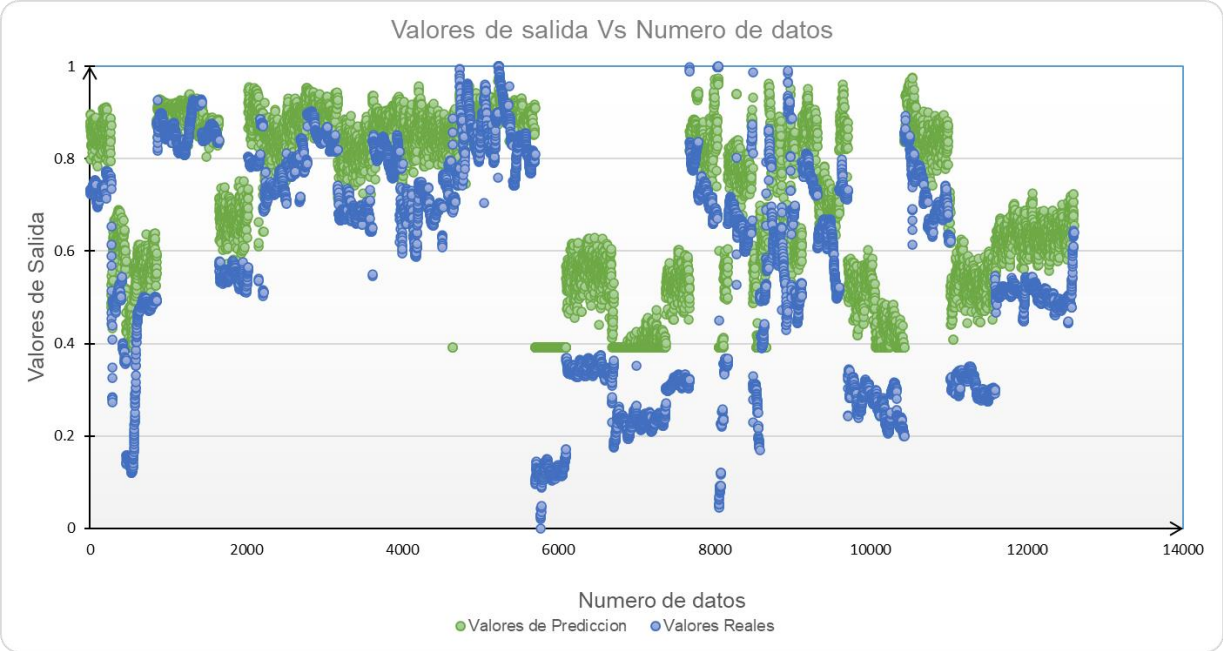


Figura 22: Grafico de dispersión entre los valores de predicción y reales sobre un modelo MLP. Fuente: Elaboración propia.

1. Multi Layer Perceptron

En este punto se exponen los resultados de la experimentación realizada con el modelo MLP, con diferentes hiper parámetros. Además, por cada cambio de hiper parámetro se presentarán 5 repeticiones, cambiando a través de su semilla de inicialización los pesos de las neuronas artificiales de valores aleatorios, para calcular del total de experimentos por caso su nivel de dispersión para los resultados a través de la desviación estándar y promedio de correlación de las compilaciones totales.

Para los experimentos realizados, solo se modificaron los hiper parámetros denominados “parámetros de cambio”, que son lo que se modificaran para este proyecto de título:

- Funcion de Activación
- Tasa de aprendizaje
- Épocas de entrenamiento

Los demás hiper parámetros se mantuvieron sin cambios para todos los experimentos de este modelo, y se denominan “parámetros constantes”, que se muestran a continuación (Tabla 7):

Número de capas	Tipo de capa	Neuronas por capa	Optimizador	Función de perdida	Tiempo de ejecución (Segundos/época)	CPU
3	Densa	10	ADAM	MSE	2	1

Tabla 7: Tabla de parámetros constantes.

MLP #1

Para el experimento número 1, tiene como parámetros de cambio:

Funcion de Activación	Tasa de aprendizaje	Épocas de entrenamiento
RELU	0,001	5

Tabla 8: Tabla parámetros de cambio MLP número 1.

Los resultados del experimento se observan a continuación (Tabla 9):

Número de repetición	Error Cuadrático Medio	Correlación
1	0,028	93,52%
2	0,030	92,94%
3	0,031	87,34%
4	0,033	85,15%
5	0,035	80,19%
Promedio	**0,031**	**87,83%**
Desviación estándar	**0,003**	**0,055**

Tabla 9: Experimentación modelo MLP número 1.

MLP #2

Para el experimento número 2, tiene como parámetros de cambio:

Funcion de Activación	Tasa de aprendizaje	Épocas de entrenamiento
SIGMOID	0,001	5

Tabla 10: Tabla parámetros de cambio MLP número 2.

Los resultados del experimento se observan a continuación (Tabla 11):

Número de repetición	Error Cuadrático Medio	Correlación
1	0,024	92,81%
2	0,024	92,56%
3	0,021	92,52%
4	0,025	92,50%
5	0,023	92,44%
Promedio	**0,023**	**92,52%**
Desviación estándar	**0,002**	**0,001**

Tabla 11: Experimentación modelo MLP número 2.

MLP #3

Para el experimento número 3, tiene como parámetros de cambio:

Funcion de Activación	Tasa de aprendizaje	Épocas de entrenamiento
TANH	0,001	5

- Tabla 12: Tabla parámetros de cambio MLP número 3.

Los resultados del experimento se observan a continuación (Tabla 13):

Número de repetición	Error Cuadrático Medio	Correlación
1	0,023	91,97%
2	0,026	92,10%
3	0,027	92,44%
4	0,032	93,23%
5	0,024	93,01%
Promedio	**0,026**	**92,51%**
Desviación estándar	**0,004**	**0,005**

Tabla 13: Experimentación modelo MLP número 3.

MLP #4

Para el experimento número 4, tiene como parámetros de cambio:

Funcion de Activación	Tasa de aprendizaje	Épocas de entrenamiento
RELU	0,001	100

- Tabla 14: Tabla parámetros de cambio MLP número 4.

Los resultados del experimento se observan a continuación (Tabla 15):

Número de repetición	Error Cuadrático Medio	Correlación
1	0,023	91,97%
2	0,026	92,10%
3	0,027	92,44%
4	0,032	93,23%
5	0,024	93,01%
Promedio	**0,026**	**92,55%**
Desviación estándar	**0,004**	**0,006**

Tabla 15: Experimentación modelo MLP número 4.

MLP #5

Para el experimento número 5, tiene como parámetros de cambio:

Funcion de Activación	Tasa de aprendizaje	Épocas de entrenamiento
SIGMOID	0,001	100

- Tabla 16: Tabla parámetros de cambio MLP número 5.

Los resultados del experimento se observan a continuación (Tabla 17):

Número de repetición	Error Cuadrático Medio	Correlación
1	0,027	93,91%
2	0,030	92,35%
3	0,030	87,32%
4	0,037	61,32%
5	0,047	48,64%
Promedio	**0,034**	**76,71%**
Desviación estándar	**0,008**	**0,204**

Tabla 17: Experimentación modelo MLP número 5.

MLP #6

Para el experimento número 6, tiene como parámetros de cambio:

Funcion de Activación	Tasa de aprendizaje	Épocas de entrenamiento
TANH	0,001	100

Tabla 18: Tabla parámetros de cambio MLP número 6.

Los resultados del experimento se observan a continuación (Tabla 19):

Número de repetición	Error Cuadrático Medio	Correlación
1	0,031	78,49%
2	0,033	74,89%
3	0,035	67,07%
4	0,038	63,94%
5	0,034	64,42%
Promedio	**0,038**	**69,77%**
Desviación estándar	**0,002**	**0,065**

Tabla 19: Experimentación modelo MLP número 6.

MLP #7

Para el experimento número 7, tiene como parámetros de cambio:

Funcion de Activación	Tasa de aprendizaje	Épocas de entrenamiento
RELU	1	100

- Tabla 20: Tabla parámetros de cambio MLP número 7.

Los resultados del experimento se observan a continuación (Tabla 21):

Número de repetición	Error Cuadrático Medio	Correlación
1	0,055	3,73E-32
2	0,116	4,79E-32
3	0,061	0
4	0,095	3,73E-32
5	0,142	4,79E-32
Promedio	**-**	**-**

Tabla 21: Experimentación modelo MLP número 7.

Experimentación MLP

De los experimentos MLP# 1 al 3 se modificaron las funciones de activación, y los demás hiper parámetros se mantuvieron constantes. Teniendo en consideración el número de épocas de entrenamiento, que para estos tres experimentos fue 5. De las tablas (9), (11) y (13) se puede expresar la siguiente tabla comparativa (tabla 22):

Experimento	Funcion de activación	Error Cuadrático medio promedio	Correlación Promedio
MLP #1	RELU	0,031	**87,83%**
MLP #2	SIGMOID	0,023	**92,52%**
MLP #3	TANH	0,026	**92,51%**

Tabla 22: Tabla comparativa de experimentación de los modelos MLP # 1,2 y 3.

De los experimentos MLP# 4 al 6 se modificaron nuevamente las funciones de activación y los demás hiper parámetros se mantuvieron constantes a excepción de las épocas de entrenamiento de modelo, para estos experimentos el valor de las épocas de entrenamientos se modificó a 100 épocas. De las tablas (15), (17) y (19) se puede expresar la siguiente tabla comparativa (tabla 23):

Experimento	Funcion de activación	Error Cuadrático medio promedio	Correlación Promedio
MLP #4	RELU	0.032	**92,55%**
MLP #5	SIGMOID	0.034	**76,71%**
MLP #6	TANH	0.034	**69,77%**

Tabla 23: Tabla comparativa de la experimentación de los modelos MLP # 4,5 y 6.

Conclusión experimentación MLP

De las tablas (22) y (23) se puede concluir, que al mantener constante las funciones de activación SIGMOID y TANH, funciones que son completamente no lineales y se aumenta las épocas de entrenamiento, el modelo sufre de sobre entrenamiento y sus salidas no poseen una correlación precisa entre los valores reales y la predicción. En el caso de la función de activación RELU que es una función con una parte lineal y otro no, un mayor número de épocas produce que aumente su correlación promedio, llegando a tener la mayor correlación en un experimento realizado. Por lo anterior se escogerá el experimento número 2 con los hiper parámetros más adecuados para el modelo MLP y con correlación más alta entre los valores reales y los predichos, expresado en la siguiente tabla (24):

Modelo	Correlación
MLP	**92,51%**

Tabla 24: Modelo MLP de mayor correlación.

En experimento número 7 (tabla 21), se modificó su taza de aprendizaje a un valor alto para observar los límites de la búsqueda de mínimos locales, en consecuencia, se observa que posee una correlación negativa, de esta forma el experimento demuestra que no es relevante para el estudio modificarlo de tal forma y no se tomara en cuenta.

2. Máquinas de soporte vectoriales de regresión

En tabla (25) se expresan la experimentación realizada sobre el modelo SVR donde se realizaron 14 experimentos, modificando sus hiper parámetros principales mencionados en el capítulo anterior. Al igual que los demás modelos, se utilizaron métricas para modelos de regresión. Los experimentos se realizaron bajo la CPU 2 y se muestran a continuación:

Número de Experimento	Constante C	Kernel	Épsilon	Error Cuadrático Medio	Correlación	CPU
SVR #1	1	RBF	0,1	0,022	**91,87%**	2
SVR #2	1	POLY	0,1	0,023	**91,29%**	2
SVR #3	1	LINEAR	0,1	0,019	**90,28%**	2
SVR #4	1	SIGMOID	0,1	0,019	**89,69%**	2
SVR #5	1	RBF	0,5	0,060	**VnN**	2
SVR #6	1	LINEAR	0,5	0,060	**VnN**	2
SVR #7	10	RBF	0,1	0,022	**91,27%**	2
SVR #8	10	SIGMOID	0,1	0,019	**90,53%**	2
SVR #9	10	LINEAR	0,1	1,425	**49,49%**	2
SVR #10	10	RBF	0,2	0,014	**86,19%**	2
SVR #11	10	LINEAR	0,2	0,019	**82,59%**	2
SVR #12	100	RBF	0,1	0,022	**91,27%**	2
SVR #13	100	LINEAR	0,1	0,027	**88,98%**	2
SVR #14	1000	LINEAR	0,1	0,022	**91,24%**	2

Tabla 25: Experimentación de modelo SVR.

Experimentación SVR

La experimentación en base al modelo SVR sobre los resultados obtenidos en la tabla 25, se explican de la siguiente manera:

Para los primeros 4 experimentos se modificó el Kernel utilizado en un modelo SVR, manteniendo constantes y por defectos de la librería los valores de los hiper parámetros. La comparación de los valores de los experimentos se puede observar en la siguiente tabla (26):

Número de Experimento	Constante C	Kernel	Épsilon	Error Cuadrático Medio	Correlación
1	1	RBF	0,1	0,022	**91,87%**
2	1	POLY	0,1	0,023	**91,29%**
3	1	LINEAR	0,1	0,019	**90,28%**
4	1	SIGMOID	0,1	0,019	**89,69%**

Tabla 26: Experimentación número SVR # 1, 2 ,3 y 4

Los experimentos 5 y 6, se realizaron cambiando el valor de Épsilon y se probó solo en dos Kernel diferentes y manteniendo el valor de la constante C. La comparación se expresa en la siguiente tabla (27):

Número de Experimento	Constante C	Kernel	Épsilon	Error Cuadrático Medio	Correlación
SVR #5	1	RBF	0,5	0,060	**VnN**
SVR #6	1	LINEAR	0,5	0,060	**VnN**

Tabla 27: Experimentación número SVR # 5 y 6.

Los experimentos 7, 8 y 9, se aumenta el valor de la constante C, para algunos valores de los Kernel de la tabla (26). Los valores se expresan en la siguiente tabla (28):

Número de Experimento	Constante C	Kernel	Épsilon	Error Cuadrático Medio	Correlación
SVR #7	10	RBF	0,1	0,022	**91,27%**
SVR #8	10	SIGMOID	0,1	0,019	**90,53%**
SVR #9	10	LINEAR	0,1	1,425	**49,49%**

Tabla 28: Experimentación número SVR # 7, 8 y 9.

Para los experimentos 10 y 11, se escogieron de la tabla (28) los dos valores de correlación más alta y más baja a los Kernel, que se observan anteriormente. Se aumentó el valor de épsilon y se mantuvo el valor de la constante C. Esta comparación se expresa en la siguiente tabla (29):

Número de Experimento	Constante C	Kernel	Épsilon	Error Cuadrático Medio	Correlación
SVR #10	10	RBF	0,2	0,014	**86,19%**
SVR #11	10	LINEAR	0,2	0,019	**82,59%**

Tabla 29: Experimentación número SVR # 10 y 11.

Los experimentos 12 y 13, se aumenta el valor de la constante C y se reduce el valor de Épsilon, dejando lo en su valor por defecto. Se realizará este experimento para los valores de Kernel de la tabla anterior (29).

El Experimento 14 se exagera el valor de la constante C, mantenido el valor de Épsilon para un Kernel que a medida de las otras experimentaciones anteriores se ha mantenido con un alto nivel de correlación al cambio de hiper parámetros.

Número de Experimento	Constante C	Kernel	Épsilon	Error Cuadrático Medio	Correlación
SVR #12	100	RBF	0,1	0,022	**91,27%**
SVR #13	100	LINEAR	0,1	0,027	**88,98%**
SVR #14	1000	LINEAR	0,1	0,022	**91,24%**

Tabla 30: Experimentación número SVR # 12, 13 y 14.

Conclusión experimentación SVR

De las tablas (26) y (28) se puede concluir que, al aumentar el valor de la constante C, los valores de correlación de los Kernel tanto como RBF y SiGMOID se ven afectado en menor medida, no así el valor de correlación con respecto al Kernel LINEAR que se ve gravemente afectado.

De tabla (30) se puede concluir que al hacer un aumento de 10% en el valor de la contante C para la tabla (29) pero descartando los resultados relacionados al Kernel SIGMOID, Existe un aumento de los valores de correlación para los experimentos mencionados. Para el ultimo experimento de la tabla (30), se aumentó el valor de C de forma exagerada (un 100% con referencia de la tabla (29), se puede observar que existe un aumento en la correlación. Por lo tanto, se al aumentar la Constante C, existe aumento no exagerada del valor de correlación, no así al modificar el valor de Épsilon, donde se observa que al acercar este valor a 1 la correlación baja hasta indeterminar su salida, como lo muestra la tabla (27).

Por lo anterior se escogerá el experimento número 1 de la tabla (25) con los hiper parámetros más adecuados para el modelo SVR y con correlación más alta entre los valores reales y los predichos, expresado en la siguiente tabla (31):

Modelo	Correlación
SVR	**91,87%**

Tabla 31: Modelo SVR de mayor correlación.

3. Redes Neuronales Recurrentes

En este punto se exponen los resultados de la experimentación realizada con el modelo RNN, con diferentes hiper parámetros. Además, por cada cambio de hiper parámetro se presentarán 5 repeticiones, cambiando a través de su semilla de inicialización los pesos de las neuronas artificiales de valores aleatorios, para calcular del total de experimentos por caso su nivel de dispersión para los resultados a través de la desviación estándar y promedio de correlación de las compilaciones totales.

Para los experimentos realizados, solo se modificaron los hiper parámetros denominados “parámetros de cambio”, que son lo que se modificaran para este proyecto de título:

- Funcion de Activación
- Tasa de aprendizaje
- Épocas de entrenamiento

Los demás hiper parámetros se mantuvieron sin cambios para todos los experimentos de este modelo, y se denominan “parámetros constantes”, que se muestran a continuación (Tabla 32):

Numero de capas	Tipo de capa	Neurona por capa	Optimizador	Time Step	Funcion de perdida	Tiempo de ejecución (Segundos/Épocas)	CPU
5	LSTM	64	ADAM	1	MSE	13	1

Tabla 32: Tabla de parámetros constantes.

RNN #1

Para el experimento número 1, tiene como parámetros de cambio:

Funcion de Activación	Tasa de aprendizaje	Épocas de entrenamiento
RELU	0,001	5

- Tabla 33: Tabla parámetros de cambio RNN número 1.

Los resultados del experimento se observan a continuación (Tabla 34):

Número de repeticiones	Error Cuadrático Medio	Correlación
1	0,044	92.01%
2	0,046	93,69%
3	0,042	94,17%
4	0,044	93,94%
5	0,039	92,23%
Promedio	**0,043**	**93,21%**
Desviación estándar	**0,002**	**0,010**

Tabla 34: Experimentación modelo RNN número 1.

RNN #2

Para el experimento número 2, tiene como parámetros de cambio:

Funcion de Activación	Tasa de aprendizaje	Épocas de entrenamiento
SIGMOID	0,001	5

Tabla 35: Tabla parámetros de cambio RNN número 2.

Los resultados del experimento se observan a continuación (Tabla 36):

Número de repeticiones	Error Cuadrático Medio	Correlación
1	0,037	87,35%
2	0,046	91,13%
3	0,049	91,19%
4	0,042	91,14%
5	0,042	91,22%
Promedio	**0,044**	**90,41%**
Desviación estándar	**0,004**	**0,017**

Tabla 36: Experimentación modelo RNN número 2.

RNN #3

Para el experimento número 3, tiene como parámetros de cambio:

Funcion de Activación	Tasa de aprendizaje	Épocas de entrenamiento
TANH	0,001	5

- Tabla 37: Tabla parámetros de cambio RNN número 3.

Los resultados del experimento se observan a continuación (Tabla 38):

Número de repeticiones	Error Cuadrático Medio	Correlación
1	0,039	91,75%
2	0,038	91,60%
3	0,038	90,61%
4	0,042	91,99%
5	0,042	92,38%
Promedio	**0,041**	**91,67%**
Desviación estándar	**0,002**	**0,006**

Tabla 38: Experimentación modelo RNN número 3.

RNN #4

Para el experimento número 4, tiene como parámetros de cambio:

Funcion de Activación	Tasa de aprendizaje	Épocas de entrenamiento
RELU	0,001	100

- Tabla 39: Tabla parámetros de cambio RNN número 4.

Los resultados del experimento se observan a continuación (Tabla 40):

Número de repeticiones	Error Cuadrático Medio	Correlación
1	0,049	91,89%
2	0,049	91,37%
3	0,048	90,94%
4	0,050	79,39%
5	0,051	72,51%
Promedio	**0,051**	**85,20%**
Desviación estándar	**0,001**	**0,088**

Tabla 40: Experimentación modelo RNN número 4.

RNN #5

Para el experimento número 5, tiene como parámetros de cambio:

Funcion de Activación	Tasa de aprendizaje	Épocas de entrenamiento
SIGMOID	0,001	100

- Tabla 41: Tabla parámetros de cambio RNN número 5.

Los resultados del experimento se observan a continuación (Tabla 42):

Número de repeticiones	Error Cuadrático Medio	Correlación
1	0,043	92,06%
2	0,044	92,05%
3	0,043	91,26%
4	0,042	91,33%
5	0,044	90,74%
Promedio	**0,043**	**91,49%**
Desviación estándar	**0,001**	**0,006**

Tabla 42: Experimentación modelo RNN número 5.

RNN #6

Para el experimento número 6, tiene como parámetros de cambio:

Funcion de Activación	Tasa de aprendizaje	Épocas de entrenamiento
TANH	0,001	100

- Tabla 43: Tabla parámetros de cambio RNN número 6.

Los resultados del experimento se observan a continuación (Tabla 44):

Número de repeticiones	Error Cuadrático Medio	Correlación
1	0,042	91,51%
2	0,040	89,83%
3	0,048	64,56%
4	0,055	42,68%
5	0,058	37,65%
Promedio	**0,049**	**65,19%**
Desviación estándar	**0,007**	**0,253**

Tabla 44: Experimentación modelo RNN número 6.

Experimentación RNN

De los experimentos 1 al 3 se modificaron las funciones de activación, y los demás hiper parámetros se mantuvieron constantes. Teniendo en consideración el número de épocas de entrenamiento, que para estos tres experimentos fue 5. De las tablas (34), (36) y (38) se puede expresar la siguiente tabla (45) comparativa:

Experimento	Funcion de activación	Error Cuadrático medio promedio	Correlación promedio
RNN #1	RELU	0,043	**93,21%**
RNN #2	SIGMOID	0,044	**90,41%**
RNN #3	TANH	0,041	**91,67%**

Tabla 45: Tabla comparativa de experimentación de los modelos RNN # 1,2 y 3.

De los experimentos 4 al 6 se modificaron nuevamente las funciones de activación y los demás hiper parámetros se mantuvieron constantes a excepción de las épocas de entrenamiento de modelo, para estos experimentos el valor de las épocas de entrenamientos se modificó a 100 épocas. De las tablas (40), (42) y (44) se puede expresar la siguiente tabla comparativa (tabla 46):

Número de Experimento	Funcion de activación	Error Cuadrático medio promedio	Correlación promedio
RNN # **4**	RELU	0,051	**85,20%**
RNN # **5**	SIGMOID	0,043	**91,49%**
RNN # **6**	TANH	0,049	**65,19%**

Tabla 46: Tabla comparativa de la experimentación de los modelos RNN número 4,5 y 6.

Conclusión experimentación RNN

De las tablas (45) y (46) se puede concluir, que al mantener constante las funciones de activación RELU y TANH, y se aumenta las épocas de entrenamiento, el modelo sufre de sobre entrenamiento y sus salidas no poseen una correlación tan precisa entre los valores reales y la predicción. En el caso de la función de activación

SIGMOID, un mayor número de épocas produce que aumente su correlación promedio.

Por lo anterior se escogerá el experimento RNN número 1 con los hiper parámetros más adecuados para el modelo MLP y con correlación más alta entre los valores reales y los predichos, expresado en la siguiente tabla (47):

Modelo	Correlación
RNN	**93,21%**

Tabla 47: Modelo RNN de mayor correlación.

Conclusión resultados

Bajo los hiper parámetros presentados y métricas correspondientes utilizadas para los tres modelos descritos y probados anteriormente, se puede agrupar en una tabla las tres salidas de mejor correlación promedio para poder compáralas entre sí, a su vez es relevante recordar que los datos que se utilizaron para el estudio de los modelos fue exactamente el mismo para todos los experimentos realizados en este estudio.

De las tablas (24), (31) y (47) se presenta la siguiente tabla (48) comparativa de los modelos que resuelven el problema que se presenta en este proyecto de título:

Modelo	Correlación
MLP	**92,51%**
SVR	**91,87%**
RNN	**93,21%**

Tabla 48: Tabla comparativa de Modelos Machine Learning.

De la tabla (39) se puede concluir que los tres modelos tienen una correlación distinta obtenidos gracias a las métricas que matemáticamente son capaces de entregarnos una correlación para ecuaciones lineales. Destacando sobre ellos el modelo de RNN, con una correlación promedio de salida del 93.21% lo que nos indica que al ser una red neuronal artificial o un modelo más robusto posee un mejor entrenamiento con los datos entregados, debido que posee una capa de retro alimentación de los datos que permite como estructura poder guardar valores en

series de tiempo comparables con valores nuevos ingresados. Este tipo de estructura es mucho más potente que los dos modelos anteriormente comparados, pese esto existen más pruebas que se le pueden seguir realizando a este modelo, para su entrenamiento y así mejoramiento de los resultados obtenidos, como la implementación de más capas, otros valores de time step para la predicción de datos con series de tiempo, en fin, los modelos Deep Learning entregan más versatilidad para el mejoramiento de los resultados de predicción debido que poseen características que años atrás por la falta de recursos tecnológicos no eran capaces de implementar ni evaluar. Se debe tomar este modelo para la predicción de los valores acerca de la velocidad de alimentación por lo anteriormente dicho, y hacer más estudios para así afinar el modelo para la entrega de resultados cada vez más exactos.

Capítulo 5: Conclusión

1. Conclusión

Según los resultados obtenidos en el capítulo anterior, los modelos de predicción que se escogieron para resolver el problema de obtener el valor de la velocidad de alimentación en el proceso de aserrado, fueron los correctos, debido que el porcentaje de correlación de los modelos supera en los tres casos el 90%, valor que es desde el punto de vista de la literatura, es más que aceptable, en adicción de lo anterior, la reducción de dimensionalidad con un método estadístico PCA permito que el conjunto de datos se pudiera expresar de mejor forma y así los modelos tuvieran un mejor desempeño para el proceso de entrenamiento. El modelo que se destacó del resto fue el RNN, que obtuvo una correlación promedio del 93,21%, lo que refleja que los modelos de gran robustez como lo son, los modelos basados redes neuronales artificiales, permitiendo una predicción de datos más precisa y no solo entrega una tendencia a seguir.

El Modelo RNN entrenado y probado, con los datos que proporción la empresa, permite la predicción de la velocidad de alimentación, de una máquina de aserrío, lo que garantiza resultados correctos de salida con un 93,21% promedio de éxito. Lo que permitirá apoyar a la empresa en la toma de decisiones futuras. Gracias este tipo de modelos también se pueden hacer predicciones que no solo, tengan que ver con la predicción de la velocidad de alimentación, sino que también se puede utilizar para poder predecir fallas en la máquina, vida útil de la sierra, etc., para la toma de decisión de mantenimiento o arreglos. Sin duda la adopción de este tipo de tecnología y gracias a lo que nos ofrece la cuarta revolución industrial nos permite una infinidad de beneficios para las empresas, como los expuesto en este proyecto.

Gracias a la metodología CRISP-DM pudimos tener pasos claros y detallados para el desarrollo de este proyecto, metodología correcta, que ayudo a la contante iteración de pasos anteriores para el mejoramiento de todos los puntos, sobre todo el de los modelos. Pese que se siguió un orden para el desarrollo del proyecto, la metodología nos propone un proceso de salida al sistema, pero este proyecto no

realizo una implementación o desarrollo de software, para dicha salida, sino que se itero constantemente para la solución del problema y cumpliendo los objetivos planteados.

2. Trabajos futuros

A futuro, se recomienda desde el punto de vista de la investigación y comparación de modelos, se podría hacer hincapié en el modelo de redes neuronales recurrentes y enfocarse la predicción de datos a través de su método de series de tiempo, con una ventana de time step mayor, así a su vez entrenar el modelo con una entrada masiva de estos y hacer una comparación con los resultados que se muestran en este proyecto de título con otros bajo el mismo contexto.

Desde el punto de vista del Software, se podría implementar un sisma que sea capaz de entregar los resultados de la predicción, para la toma de decisiones, alimentando al modelo con datos en tiempo real, desde la nube o vía streaming, además de entrecruzarlo con algún software capaz encontrar outliers en los data set para evitar salidas erróneas del sistema y sea visualmente potente.

3. Bibliografía

Brownlee, J. (2019). Develop Deep Learning Models On Theano And TensorFlow Using Keras. *Journal of Chemical Information and Modeling*, *53*(9), 1689–1699. https://doi.org/10.1017/CBO9781107415324.004

Berzal, F. (2018). Redes Neuronales & Deep Learning. *Departamento de Ciencias de La Computacion e IA*, 753.

Grosan, C., & Abraham, A. (2011). Machine Learning. In *Intelligent Systems Reference Library* (Vol. 17). https://doi.org/10.1007/978-3-642-21004-4_10

Baştanlar, Y., & Ozuysal, M. (2014). Introduction to Machine Learning Second Edition. In *Methods in molecular biology (Clifton, N.J.)* (Vol. 1107). https://doi.org/10.1007/978-1-62703-748-8_7

Wang, B., Lei, Y., Yan, T., Li, N., & Guo, L. (2020). Recurrent convolutional neural network: A new framework for remaining useful life prediction of machinery. *Neurocomputing*, *379*, 117–129. https://doi.org/10.1016/j.neucom.2019.10.064

Sherstinsky, A. (2020). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. *Physica D: Nonlinear Phenomena*, *404*, 132306. https://doi.org/10.1016/j.physd.2019.132306

Marín Diazaraque, J. M. (2007). Introducción a las redes neuronales aplicadas. *Manual Data Mining*, 1–31. Retrieved from halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/DM/tema3dm.pdf

Panahi, M., Gayen, A., Pourghasemi, H. R., Rezaie, F., & Lee, S. (2020). Spatial prediction of landslide susceptibility using hybrid support vector regression (SVR) and the adaptive neuro-fuzzy inference system (ANFIS) with various metaheuristic algorithms. *Science of the Total Environment*, *741*, 139937. https://doi.org/10.1016/j.scitotenv.2020.139937

Ge, R., Huang, F., Jin, C., & Yuan, Y. (2015). Escaping from saddle points: Online stochastic gradient for tensor decomposition. *Journal of Machine Learning Research*, *40*(2015).

Ren, T., Liu, X., Niu, J., Lei, X., & Zhang, Z. (2020). Real-time water level prediction of cascaded channels based on multilayer perception and recurrent neural network. *Journal of Hydrology*, *585*(December 2019), 124783. https://doi.org/10.1016/j.jhydrol.2020.124783

Bigliardi, B., Bottani, E., & Casella, G. (2020). Enabling technologies, application areas and impact of industry 4.0: A bibliographic analysis. *Procedia Manufacturing*, *42*(2019), 322–326. https://doi.org/10.1016/j.promfg.2020.02.086

Rafael, L. D., Jaione, G. E., Cristina, L., & Ibon, S. L. (2020). An Industry 4.0 maturity model for machine tool companies. *Technological Forecasting and Social Change*, *159*(June), 120203. https://doi.org/10.1016/j.techfore.2020.120203

Neugebauer, R., Hippmann, S., Leis, M., & Landherr, M. (2016). Industrie 4.0 - From the Perspective of Applied Research. *Procedia CIRP*, *57*, 2–7. https://doi.org/10.1016/j.procir.2016.11.002

Chaves Palacios, J. (2004). Desarrollo tecnológico en la Primera Revolución Industrial. *Norba. Revista de Historia*, *17*(17), 93–109.

Jauro, F., Chiroma, H., Gital, A. Y., Almutairi, M., Abdulhamid, S. M., & Abawajy, J. H. (2020). Deep learning architectures in emerging cloud computing architectures: Recent development, challenges and next research trend. *Applied Soft Computing Journal*, *96*, 106582. https://doi.org/10.1016/j.asoc.2020.106582

Kingma, D. P., & Ba, J. L. (2015). Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 1–15.

Saturno, M., Moura Pertel, V., Deschamps, F., & De Freitas Rocha Loures, E. (2018). Proposal of an Automation Solutions Architecture for Industry 4.0. *DEStech Transactions on Engineering and Technology Research*, (icpr). https://doi.org/10.12783/dtetr/icpr2017/17675

Kim, M. K., Kim, Y. S., & Srebric, J. (2020). Impact of correlation of plug load data, occupancy rates and local weather conditions on electricity consumption in a building using four back-propagation neural network models. *Sustainable Cities and Society*, *62*(June), 102321. https://doi.org/10.1016/j.scs.2020.102321

Gamage, S., & Samarabandu, J. (2020). Deep learning methods in network intrusion detection: A survey and an objective comparison. *Journal of Network and Computer Applications*, 102767. https://doi.org/10.1016/j.jnca.2020.102767

Sarkar, D., Raghav, B., & Sharma, T. (2020). Practical Machine Learning with Python. In *Practical Machine Learning with Rust*. https://doi.org/10.1007/978-1-4842-5121-8

Torres, J. (2020). Redes de Neuronas. *Universidad de Granada*, 1–39.

Antonio Alejandro Barría Lazo, Nibaldo Rodríguez Agurto, W. P. (2016). *RED NEURONAL SIGMOIDAL CON VALORES SINGULARES PARA LA SEVERIDAD DE FALLOS EN EQUIPOS DE ROTACIÓN*. 31.

Betancourt, E. R., Chacón, P. S., & Murillo, E. C. (2016). *Comparación de modelos de redes neuronales profundas para clasificar la polaridad de Tweets en español*. Retrieved from https://colab.research.google.com

Vaidya, S., Ambad, P., & Bhosle, S. (2018). Industry 4.0 - A Glimpse. *Procedia Manufacturing*, *20*, 233–238. https://doi.org/10.1016/j.promfg.2018.02.034

Mohri, M., Rostamizadeh, A., & Ameet, T. (2018). *Foundations of Machine Learning* (Second; F. B. Bash, ed.). The Adaptive Computations and Machine Learning series appears.

Rozo-García, F. (2020). Revisión de las tecnologías presentes en la industria 4.0. *Revista UIS Ingenierías*, *19*(2), 177–191. https://doi.org/10.18273/revuin.v19n2-2020019

Pete, C., Julian, C., Randy, K., Thomas, K., Thomas, R., Colin, S., & Wirth, R. (2000). Crisp-Dm 1.0. *CRISP-DM Consortium*, 76.

Natanael Esteban Rain Montecinos. (2020). MODELACIÓN DE LA PERFORMANCE DE UNA MÁQUINA CEPILLADORA USANDO UN ENFOQUE EN MACHINE LEARNING. *MODELACIÓN DE LA PERFORMANCE DE UNA MÁQUINA CEPILLADORA USANDO UN ENFOQUE EN MACHINE LEARNING*. https://doi.org/10.1017/CBO9781107415324.004

Nasir, V., & Cool, J. (2020). A review on wood machining: characterization, optimization, and monitoring of the sawing process. *Wood Material Science and Engineering*, *15*(1), 1–16. https://doi.org/10.1080/17480272.2018.1465465

Laguna, C. (2019). Correlación y regresión. *Análisis Exploratoria de Datos*, 143–178. https://doi.org/10.2307/j.ctvc5pc9g.6

Kim, K., & Jeong, J. (2020). Deep Learning-based Data Augmentation for Hydraulic Condition Monitoring System. *Procedia Computer Science*, *175*(2019), 20–27. https://doi.org/10.1016/j.procs.2020.07.007

Djellali, C., & adda, M. (2020). A New Hybrid Deep Learning Model based-Recommender System using Artificial Neural Network and Hidden Markov Model. *Procedia Computer Science*, *175*(2019), 214–220. https://doi.org/10.1016/j.procs.2020.07.032