UNIVERSIDAD NACIONAL JOSÉ MARIÁ ARGUEDAS

FACULTAD DE INGENIERIA

ESCUELA PROFESIONAL DE INGENIERIA DE SISTEMAS

Presentado por:

JOSE HUARACA FLORES

"USO DE REDES NEURONALES ARTIFICIALES PARA EL PRONOSTICO DE DEMANDA DE USO DE LIBROS EN LA BIBLIOTECA EPIS – UNAJMA 2018"

Asesor:

Ing. ROBERTO QUISPE QUISPE

TESIS PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO DE SISTEMAS

ANDAHUAYLAS – APURÍMAC – PERÚ

2019

DEDICATORIA

Mi tesis la dedico a Dios por permitirme llegar a este momento tan importante en mi vida, que con su infinita bondad y amor puedo llegar a alcanzar los triunfos a los que aspiro y vencer las adversidades que encontrare en mi camino.

A mis amados padres Elisa Flores Villegas y Anastasio Huaraca Olarte, por ser mi guía, por demostrarme su cariño y apoyo incondicional.

A mis hermanas Marina, Gloria y Lurdes por su cariño y apoyo incondicional, por compartir momentos significativos, por darme fuerza y por escucharme en todo momento.

AGRADECIMIENTO

Son muchas las personas que han contribuido al proceso y conclusión de este proyecto de tesis.

En primer lugar, a Dios y a mi familia por ser parte de mi vida y formación personal. Al ingeniero Roberto Quispe Quispe quien es el asesor de esta Tesis, quien me apoyo y me alentó para que concluyera esta investigación.

Agradezco a la universidad José María Arguedas de Andahuaylas, donde me he formado profesionalmente y he recibido apoyo en todo momento.

Por último, pero no menos importante quiero agradecer a todas las personas que estuvieron involucradas con este proyecto de tesis por brindarme parte de su conocimiento y experiencia para poder culminar esta meta.

TABLA DE CONTENIDO

INDICE DE GRAFICOS

INDICE DE FIGURAS

RESUMEN

El presente trabajo de Investigación tiene por objetivo simular predecir la demanda de uso de libros de la biblioteca especializada de la Escuela Profesional de Ingeniería de Sistemas de la UNAJMA, utilizando redes neuronales artificiales, la cual se encuentra dentro de la Inteligencia Artificial Subsimbólica. El diseño de la investigación es no experimental. Para la muestra se ha seleccionado mediciones de préstamos de libros mensuales (Und) agrupados por meses (enero 2015 – noviembre 2018) haciendo un total de cuarenta y siete meses.

El desarrollo de la investigación comienza con la construcción del corazón matemático de las redes neuronales artificiales, luego se procede a la creación de propio código basándose en las fórmulas obtenidas y los pasos que componen el proceso de aprendizaje del perceptrón multicapa y el algoritmo Back Propagation.

Los resultados de la investigación apoyan la utilización de las redes neuronales artificiales como técnica confiable en la predicción de series de tiempo. Resuelve este tipo de problemas de manera eficiente, encontrándose resultados satisfactorios, con el valor de test de validación y el valor del error cuadrático medio.

Palabras clave: red neuronal artificial, inteligencia artificial subsimbólica.

INTRODUCCION

La planeación en toda industria es una necesidad. Se puede decir incluso que el objetivo importante de la planeación es tratar de prever lo que sucederá en el futuro en base de una recopilación de hechos o sucesos acaecidos con anterioridad, esto repercute en la toma de decisiones y en la planificación de recursos para una mayor y eficiente producción. La predicción debe ser lo más realista posible y además ser confiable. Las redes neuronales artificiales son las que actualmente están causando un mayor impacto, debido a su extraordinaria aplicabilidad práctica.

La simulación por computadora es una herramienta interdisciplinaria y tiene aplicaciones en muchos campos de la ciencia y la tecnología, los resultados de los experimentos obtenidos a través de la simulación influyen cada vez más en las decisiones tomadas en todos los campos del quehacer humano.

En el presente trabajo de investigación se pretende predecir la demanda de uso de libros de la biblioteca especializada de la escuela profesional de ingeniería de Sistemas de la UNAJMA.

CAPITULO I

1 PROBLEMA DE INVESTIGACION

1.1 Descripción del problema

La biblioteca especializada de la escuela profesional de Ingeniería de Sistemas presento su preocupación en saber el volumen de préstamos periódicos para abastecer la demanda, que número de trabajadores debe tener para brindar un servicio rápido, entre otros factores. Luego de analizar, se determinó que el principal problema de debía a una falta de predicción, en ocasiones se encontraba casos en la escasa cantidad de ejemplares para el préstamo, falta de personal y otros. Detectado el problema, la coordinación necesita un modelo predictivo, para poder tomar decisiones a corto mediano plazo y programar sus actividades.

Si se mantienen el problema se generaría deficiencias en el área de biblioteca y el servicio que presta. Al estar en el sector educación cuya función principal es la formación de nuevos profesionales, esto provoca el incremento de la demanda de material bibliográfico.

Prever lo que sucederá en el futuro en base de una recopilación de un hecho o suceso acaecido con anterioridad, es importante para la planeación a fin de dar una solución al problema, es conveniente realizar un pronóstico de demanda de material bibliográfico. Por lo tanto, es necesario un modelo que nos permita

predecir en series de tiempo la demanda de material bibliográfico usaremos para el presente trabajo de investigación un modelo de red neuronal artificial; puesto que existe información cuantitativa el cual brinde una predicción de valores futuros del volumen de la demanda a través del tiempo, pretendiendo mejorar significativamente la toma de decisiones.

1.2 Delimitación del problema

1.2.1 Alcances

Para realizar el presente trabajo se dispuso de un conjunto de datos que corresponden al registro de préstamos efectuados en la biblioteca especializada de la Escuela Profesional de Ingeniería de Sistemas, para ello se construyó una red neuronal artificial con el fin de efectuar el pronóstico de la demanda de libros con el propósito de demostrar que dicha herramienta resuelve problemas de este tipo de manera eficiente, teniendo resultados altamente satisfactorios.

1.2.2 Limitaciones

No existió limitaciones para efectuar el entrenamiento de una Red Neuronal Artificial para efectuar el pronóstico de demanda de libros en la biblioteca especializada de la Escuela Profesional de Ingeniería de Sistemas.

1.3 Formulación del problema

1.3.1 Problema general

¿Es factible utilizar redes neuronales artificiales para predecir la demanda de material bibliográfico en la biblioteca de la EPIS - UNAJMA?

1.3.2 Problemas específicos

- ¿Es realizable diseñar el modelo de predicción de demanda de material bibliográfico en la EPIS - UNAJMA?
- ¿Es posible calcular valores futuros a partir del modelo de predicción de demanda de material bibliográfico de la EPIS - UNAJMA?

1.4 Justificación

La elección e implementación de un método adecuado de pronóstico siempre ha sido un tema de gran importancia para las empresas. Un error significante en el pronóstico de la demanda de material bibliográfico podría generar la insatisfacción de los usuarios y en un pronóstico erróneo aumentaría los gastos para la institución.

La estimación del comportamiento de algunas variables en el futuro puede realizarse utilizando diversos métodos de pronósticos. Cada método de proyección tiene una aplicación de carácter especial que hace de su selección un problema de decisión influenciado por diversos factores, como, por ejemplo, la validez y disponibilidad de los datos históricos, la precisión deseada del pronóstico, los beneficios de los resultados, los periodos futuros que se desee pronosticar y el tiempo disponible para hacer el estudio entre otros.

Existen diversos métodos para proyectar la demanda, dentro de los mismos se encuentra el método de las redes neuronales artificiales. Debido a su constitución y a sus fundamentos, presentan un gran número de características semejantes a las del cerebro humano. Por ejemplo, son capaces de aprender de la experiencia, de generalizar de casos anteriores a nuevos casos y de abstraer características esenciales. Las redes neuronales artificiales son metodologías novedosas que

permiten hacer pronósticos donde hay cierto comportamiento no lineal y prometen un buen desempeño en este campo de investigación.

La principal ventaja de las aplicaciones de las redes neuronales reside en el procesado paralelo, adaptativo y no lineal. En la actualidad las redes neuronales artificiales son un campo muy consolidado en el que continuamente se desarrollan nuevos métodos y aplicaciones.

Aplicando las redes neuronales artificiales, pretendemos predecir la demanda de material bibliográfico en distintos instantes, las cuales permitirán realizar cálculos complejos (cálculos empíricos), y así obtener valores de simulaciones de las cantidades estimadas con la que se contará en un futuro.

1.5 Objetivos de la Investigación

1.5.1 Objetivo general

Predecir la demanda de uso de material bibliográfico de la biblioteca especializada de la EPIS – UNAJMA 2018.

1.5.2 Objetivos específicos

- Diseñar el modelo de predicción de demanda de uso de material bibliográfico de la biblioteca especializada de la EPIS – UNAJMA 2018.
- Calcular los valores futuros a partir del modelo de predicción de demanda de uso de material bibliográfico de la biblioteca especializada de la EPIS – UNAJMA 2018.

CAPITULO II

2 ANTECEDENTES

2.1 Antecedentes de la investigación

2.1.1 Antecedentes a nivel internacional

La demanda se establece como un requisito primordial que debe soportar el plan financiero o plan de ventas con lo que, la mayor parte de las veces, el proceso de planificación de la demanda está fuertemente condicionado por elementos que, en un principio, tiene que ver con el mercado y los clientes.

El método usual de planeación al interior de las empresas se inicia con un proceso de pronóstico de la demanda, pronóstico bien sea elaborado con un simple crecimiento sobre resultados anteriores o mediante herramientas tecnológicas que permiten efectuar innumerables cálculos para tratar de que el pronóstico sea lo más cercano a la realidad. Muchas de estas herramientas son extremadamente costosas y, peor aún, no son muchas las personas que pueden efectuar un pronóstico seguro a partir de los resultados de las mismas.

Actualmente, las redes neuronales artificiales son una de las técnicas más usadas para la resolución de pronósticos de este tipo debido a sus óptimos resultados y su fácil metodología de resolución.

La idea de utilizar la red neuronal artificial en el pronóstico de series de tiempo que fue aplicada por primera vez en 1964, cuando se utilizó una red neuronal

artificial lineal para el pronóstico del clima (Sebastian, Verónica, Bibiana, & Ramon, 2003).

Se pueden encontrar muchos trabajos de investigación sobre este tema, a nivel internacional como el de Gómez (2010) denominado “Pronóstico de una serie temporal usando redes neuronales”, que tuvo como objetivo principal realizar la predicción y la comparación un conjunto de datos, los cuales corresponden a las demandas máximas mensuales de energía eléctrica proporcionados por la Comisión Federal de Electricidad, utilizando dos enfoques para lograr tal objetivo, la metodología de Box y Jenkins (1973) para series de tiempo y el uso de las redes neuronales artificiales.

A nivel nacional como el de Jiménez (2013) denominado pronóstico de demanda de llamadas en los call center, utilizando redes neuronales artificiales, que construyó una red neuronal artificial utilizada para el pronóstico de demanda de llamadas del centro de atención telefónica (call center) de clientes de la empresa ABC con el propósito de demostrar que dicha herramienta resuelve este tipo de problemas de manera eficiente, encontrando resultados altamente satisfactorios.

2.1.2 Antecedentes a nivel nacional

Yanarico (2013) denominado desarrollo de un software utilizando redes neuronales artificiales para la simulación del ciclo hidrológico de la laguna Aricota, que tuvo como objetivo desarrollar un software que permitiera realizar análisis de cada variable hidrológica, el entrenamiento de una red neuronal artificial (para obtener valores de incremento del volumen y nivel de agua de la laguna Aricota), y la simulación del ciclo hidrológico de la laguna Aricota.

CAPITULO III

3 MARCO TEORICO

3.1 Pronostico

Pronosticar es el arte de especificar información significativa acerca del futuro. Se menciona que los pronósticos jamás son perfectos debido a que, básicamente, se utilizan métodos que generan pronósticos sobre la base de la información previa. Los pronósticos serán menos confiables mientras mayor sea el horizonte que se va pronosticar. (Mendoza, 2011)

3.2 Demanda

La demanda es el deseo que se tiene de un determinado producto pero que está respaldado por una capacidad de pago. Se refiere a las cantidades de un producto que los consumidores están dispuestos a comprar a los posibles precios del mercado. La demanda es la cantidad de bienes o servicios que el comprador o consumidor está dispuesto a adquirir a un precio dado y en un lugar establecido, con cuyo uso pueda satisfacer parcial o totalmente sus necesidades particulares o pueda tener acceso a su utilidad intrínseca. (Armstrong & Green, 2006).

3.3 Biblioteca especializada

Las bibliotecas especializadas son centros de información que aglutinan, tratan y difunden información relativa a un tema o a un grupo de temas afines. Normalmente se trata de organismos vinculados a centros de investigación, organizaciones industriales o culturales, laboratorios, asociaciones profesionales, departamentos gubernamentales y todo tipo de instituciones que desarrollan su trabajo en un ámbito determinado, acompañando los principales objetivos de la institución a la que pertenecen.

La ALA (Asociación Latinoamericana de Archivos) la define como: “la biblioteca establecida, mantenida y administrada por una firma comercial, una corporación privada, una asociación, un organismo estatal u otro grupo o entidad que tienen interés por una materia específica para atender las necesidades de información de sus miembros o personal y alcanzar los objetivos de la organización”. La UNESCO, por su parte, las define como “aquellas bibliotecas que dependen de una asociación, servicio oficial, departamento, centro de investigación, sociedad erudita, asociación profesional, museo, empresa o cualquier otro tipo de organismo, y cuyos acervos pertenezcan en su mayoría a una rama particular, por ejemplo: ciencias naturales, ciencias sociales, historia, etc”.

Su aparición es relativamente reciente, adquiriendo más fuerza a partir de la segunda mitad del siglo XX como resultado de la explosión de la información y de la especialización como valor social. Estados Unidos fue pionero en la creación de sociedades de bibliotecas especializadas. La SLA (Special Libraries Association) fue creada en el año 1909 y quince años más tarde inauguraba la división de Ciencia y Tecnología bajo el nombre de Grupo Tecnológico. Casi al mismo tiempo, en el año 1932, se fundaba la ARL (Association of Research Libraries).

La misión principal de estas bibliotecas es proporcionar información a las personas que enfocan sus actividades hacia estos temas, así como a las que trabajan en esos centros, para que puedan desarrollar adecuadamente sus tareas en base a estudios e investigaciones.

3.4 Redes neuronales

Las redes neuronales (también conocidas como sistemas conexionistas) son un modelo computacional basado en un gran conjunto de unidades neuronales simples (neuronas artificiales) de forma aproximadamente análoga al comportamiento observado en los axones de las neuronas en los cerebros biológicos1. La información de entrada atraviesa la red neuronal (donde se somete a diversas operaciones) produciendo unos valores de salida.

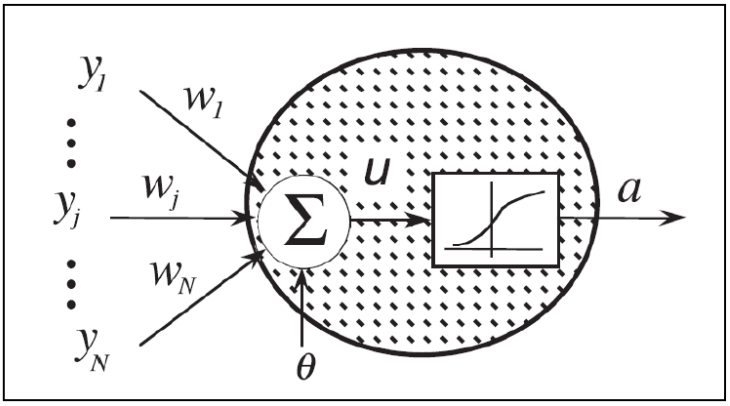

Figura 1. Modelo de neurona McCulloch y Pitts.
Fuente: Hu, Y. H., & Hwang, J.-N. (2002). Handbook of Neural Network Signal Processing. U.S.: CRC Press LLC.

Cada neurona está conectada con otras a través de unos enlaces. En estos enlaces el valor de salida de la neurona anterior es multiplicado por un valor de peso. Estos pesos en los enlaces pueden incrementar o inhibir el estado de activación de las neuronas adyacentes. Del mismo modo, a la salida de la neurona, puede existir una función limitadora o umbral, que modifica el valor resultado o impone un límite que se debe sobrepasar antes de propagarse a otra neurona. Esta función se conoce como función de activación.

Estos sistemas aprenden y se forman a sí mismos, en lugar de ser programados de forma explícita, y sobresalen en áreas donde la detección de soluciones o

características es difícil de expresar con la programación convencional. Para realizar este aprendizaje automático, normalmente, se intenta minimizar una función de pérdida que evalúa la red en su total. Los valores de los pesos de las neuronas se van actualizando buscando reducir el valor de la función de pérdida. Este proceso se realiza mediante la propagación hacia atrás.

El objetivo de la red neuronal es resolver los problemas de la misma manera que el cerebro humano, aunque las redes neuronales son más abstractas. Los proyectos de redes neuronales modernos suelen trabajar desde unos miles a unos pocos millones de unidades neuronales y millones de conexiones que, si bien son muchas órdenes, siguen siendo de una magnitud menos compleja que la del cerebro humano, más bien cercana a la potencia de cálculo de un gusano.

Nuevas investigaciones sobre el cerebro a menudo estimulan la creación de nuevos patrones en las redes neuronales. Un nuevo enfoque está utilizando conexiones que se extienden mucho más allá y capas de procesamiento de enlace en lugar de estar siempre localizado en las neuronas adyacentes. Otra investigación está estudiando los diferentes tipos de señal en el tiempo que los axones se propagan, como el aprendizaje profundo, interpola una mayor complejidad que un conjunto de variables booleanas que son simplemente encendido o apagado.

Las redes neuronales se han utilizado para resolver una amplia variedad de tareas, como la visión por computador y el reconocimiento de voz, que son difíciles de resolver usando la ordinaria programación basado en reglas. Históricamente, el uso de modelos de redes neuronales marcó un cambio de dirección a finales de los años ochenta de alto nivel, que se caracteriza por sistemas expertos con conocimiento incorporado en si-entonces las reglas, a bajo nivel de aprendizaje

automático, caracterizado por el conocimiento incorporado en los parámetros de un modelo cognitivo con algún sistema dinámico.

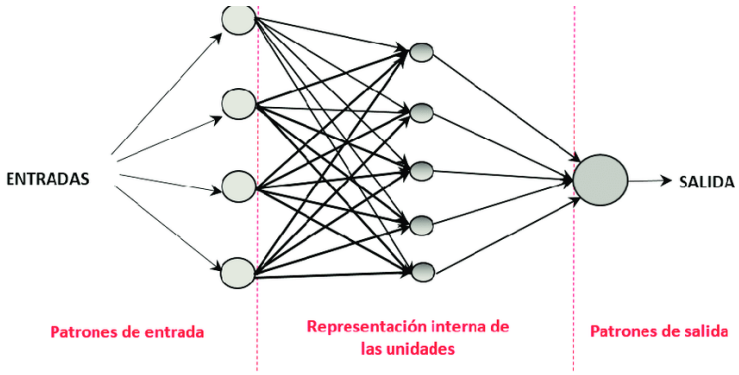

Figura 2. Modelo de red neurona McCulloch y Pitts.
Fuente: Hu, Y. H., & Hwang, J.-N. (2002). Handbook of Neural Network Signal Processing. U.S.: CRC Press LLC.

3.5 Inteligencia Artificial

La inteligencia artificial (Artificial Intelligence, o AI) es la simulación de procesos de inteligencia humana por parte de máquinas, especialmente sistemas informáticos. Estos procesos incluyen el aprendizaje (la adquisición de información y reglas para el uso de la información), el razonamiento (usando las reglas para llegar a conclusiones aproximadas o definitivas) y la autocorrección. Las aplicaciones particulares de la AI incluyen sistemas expertos, reconocimiento de voz y visión artificial.

El término AI fue acuñado por John McCarthy, un informático estadounidense, en 1956 durante la Conferencia de Dartmouth, donde nació la disciplina. Hoy en día, es un término general que abarca todo, desde la automatización de procesos robóticos hasta la robótica actual. Ha ganado prominencia recientemente debido, en parte, a los grandes volúmenes de datos, o al aumento de velocidad, tamaño y variedad de datos que las empresas están recopilando. AI puede realizar tareas

tales como identificar patrones en los datos de manera más eficiente que los seres humanos, lo que permite a las empresas obtener más información sobre sus datos.

3.6 Neurona

Una neurona es una unidad de procesamiento de información que es fundamental para el funcionamiento de una red neuronal artificial. (Haykin, 1999)

3.7 Sinapsis

Una sinapsis se utiliza para conectar una capa a otra. Las sinapsis contienen las matrices de peso utilizados por la red neuronal. Las matrices de peso tienen los valores de conexión entre cada una de las neuronas en las dos capas que son conectado por esta sinapsis. (Heaton, 2010)

3.8 Aprendizaje

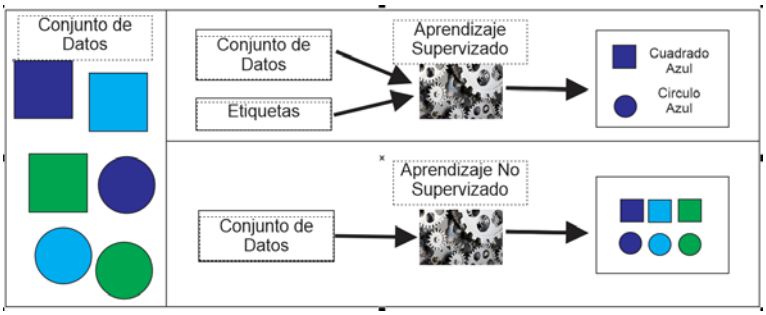

Figura 03. Tipos de Aprendizaje, Aprendizaje Supervisado y Aprendizaje no Supervisado.

La importancia principal de una red neuronal es su capacidad para aprender de su entorno. En el contexto de las redes neuronales, el aprendizaje se define como un proceso por el cual los parámetros libres de una red neuronal se adaptan a través de un proceso continuo de la estimulación por el medio ambiente. El tipo de aprendizaje es determinado por la manera en que los cambios de parámetros

se llevan a cabo (ver figura 3). La red es estimulada por el medio ambiente, los cambios en la red como resultado de la estimulación y la red responde al ambiente en una nueva forma después de la ocurrencia del cambio. (Yeung, Cloete, Shi, & Ng, 2010)

Hemos encontrado que después de un número finito de pasos la regla de aprendizaje por refuerzo proporcionará una solución correcta si se cumplen ciertas condiciones. Sin embargo, hay otras reglas de aprendizaje que le darán soluciones correctas en determinadas condiciones. Estas reglas también se basan en los procedimientos de gradiente para minimizar ciertas funciones de criterio, como funciones de costo (por errores de clasificación), a veces llamadas funciones de error o funciones de energía. (Veelenturf, 1995)

Claramente una de las diferencias entre el aprendizaje automático y la estadística es la terminología, que por supuesto es una función de la historia de las disciplinas. Esto se ve agravado por el estándar en matemáticas. Otra diferencia fácilmente descubierta es que el aprendizaje de la máquina tiene mejores nombres para sus actividades. Las diferencias más importantes son que aprendizaje automático tiende a tener un énfasis en heurísticas simples y rápidas, mientras que las estadísticas tienden a comenzar con un modelo para los datos, a menudo no existe un modelo real de los datos o solo un trivial en el aprendizaje de la máquina. (Dunne, 2007)

3.9 Entrenamiento

La formación es el proceso por el cual estos pesos de conexión están asignados. Los pesos utilizados en las conexiones entre las diferentes capas tienen mucha importancia en el funcionamiento de la red neuronal y la caracterización de una red. (Rao & Rao, 1995)

La mayoría de algoritmos de entrenamiento comienzan mediante la asignación de números aleatorios a una matriz de ponderaciones. Entonces, se examina la validez de la red neuronal. Las ponderaciones se basan a juntado de lo bien que la red neuronal realizó y la validez de los resultados. Este proceso se repite hasta que el error de validación está dentro de un límite aceptable. Hay muchas maneras de entrenar redes neuronales. Métodos de entrenamiento generalmente se dividen en las categorías de enfoques híbridos supervisadas, sin supervisión, y varios. (Heaton, 2008)

Multicapa estaba contenida en la tesis de Paul Werbos en 1974. Esta tesis presenta el algoritmo en el contexto de las redes generales, con las redes neuronales como caso especial, y no se difundió en la comunidad de redes neuronales. No fue sino hasta mediados de la década de 1980 que el algoritmo Back Propagation fue redescubierto y ampliamente publicitado. Fue redescubierto independientemente por David Rumelhart, Geoffrey Hinton y Ronald Williams, David Parker y Yann Le Cun. El algoritmo fue popularizado por su inclusión en el grupo de procesamiento distribuido en paralelo libro dirigido por psicólogos David Rumelhart y James McClelland, la publicación de este libro impulsado un torrente de investigación en redes neuronales. El perceptrón multicapa, entrenado por el algoritmo Back Propagation, es actualmente la red neuronal más utilizada. (Hagan, Demuth, & Beale, 1996)

3.10 Simulación

La simulación es una técnica para estimar las medidas de desempeño del sistema modelado. Se consideran varias técnicas para la simulación de modelos de procesamiento, utilizando metodologías de programación convencionales. (Freeman & Skapura, 1991)

3.11 Características

Las redes neuronales artificiales se pueden caracterizar más adecuadamente como modelos computacionales con propiedades particulares, tales como la capacidad de adaptarse o aprender, generalizar, agrupar u organizar los datos y cuya operación se basa en el procesamiento en paralelo. (Krose & van der Smagt, 1996)

3.12 Logistic Regression

Gracias a la regresión lineal se pueden predecir cantidades continuas como el precio de las casas como una función lineal que depende de los datos de entrada. Sin embargo, en algunas ocasiones resulta de interés predecir variables discretas, es decir, para predecir por ejemplo si un conjunto de píxeles representa un “0” o un “1”. Los problemas de este tipo son problemas de clasificación y la regresión logística es un algoritmo de clasificación bastante simple para aprender a tomar dichas decisiones.

En la regresión lineal, se probó a predecir los valores de $^{(i)}$ usando una función lineal $\boldsymbol{y} = h_\theta(\boldsymbol{x}) = \boldsymbol{\theta}^T\boldsymbol{x}$. Este tipo de funciones no son las adecuadas para predecir valores binarios ($^{(i)} \in \{0,1\}$), por lo que se usa una hipótesis diferente para predecir la probabilidad de pertenecer a una clase frente a la contraria. Concretamente, se usará una función de la forma:

$$P(y = 1|x) = h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} \equiv \sigma(\theta^T x)$$

$$P(y = 0|x) = 1 - P(y = 1|x) = h_\theta(x)$$

La función $\sigma(z) \equiv \frac{1}{1+e^{-z}}$, también llamada función “logística”, contiene todos los valores de $\boldsymbol{\theta^T x}$ dentro del rango [0,1] para poder ser interpretado como probabilidad. El objetivo es encontrar un valor de θ tal que la probabilidad de

$\boldsymbol{P}(\boldsymbol{y} = 1|x)$ sea alta cuando $\boldsymbol{x}$ pertenece a la clase “1” y pequeña cuando pertenezca a la clase “0”. La función de coste que se usara con un set de entrenamiento cuyos targets son binarios es:

$$J(\theta) = \sum_i (y^i \log\left(h_\theta\left(x^{(i)}\right)\right) + (1 - y^{(i)})\log(1 - h_\theta(x^{(i)})))$$

En la función anterior, solo uno de los dos términos de la sumatoria es distinto de cero para cada ejemplo del set de entrenamiento (dependiendo de la clase de $\boldsymbol{y^{(i)}}$ es “1” o “0”). Una vez que se tiene la función coste, es necesario aprender a clasificar los datos como “1” o “0” comprobando las probabilidades de pertenecer a cada una de las clases. Si $\boldsymbol{P}(\boldsymbol{y} = 1|x) > \boldsymbol{P}(\boldsymbol{y} = 0|x)$, entonces se clasifica como “1”, “0” en caso contrario. Esto último es lo mismo que comprobar $h(\boldsymbol{x})$ > 0.5.

Para minimizar la función coste se pueden utilizar las mismas herramientas que con la regresión lineal. Hay que proveer a la función optimizadora el cálculo del coste y del gradiente para cualquier valor de $\boldsymbol{\theta}$. En este caso, las componentes del gradiente se pueden calcular como:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \sum_i x_j^{(i)}(h_\theta\left(x^{(i)}\right) - y^{(i)})$$

Escrito en forma vectorial, el gradiente se puede expresar como:

$$\nabla_\theta J(\theta) = \sum_i x_j^{(i)}(h_\theta\left(x^{(i)}\right) - y^{(i)})$$

Esta última ecuación es esencialmente la misma que la del gradiente para la regresión lineal, solo que ahora $h(\boldsymbol{x}) = \sigma(\boldsymbol{\theta T x})$, por lo que, haciendo $\hat{y} = \sigma(\boldsymbol{\theta T x}) - \boldsymbol{y}$ la equivalencia de la ecuación (3-5) sigue siendo válida.

3.13 Función de Activación

Si se considera un problema de aprendizaje supervisado en el que se tiene acceso a determinados ejemplos para el entrenamiento ((ᵢ), (ᵢ)), las redes neuronales permiten definir la forma de la función $h_{W,b}(\boldsymbol{x})$, la cual puede ser compleja o incluso no lineal.

Antes de describir redes neuronales, se empezará describiendo la red neuronal más sencilla posible, es decir, aquella que se corresponde con una sola "neurona". El diagrama de una neurona es el siguiente:

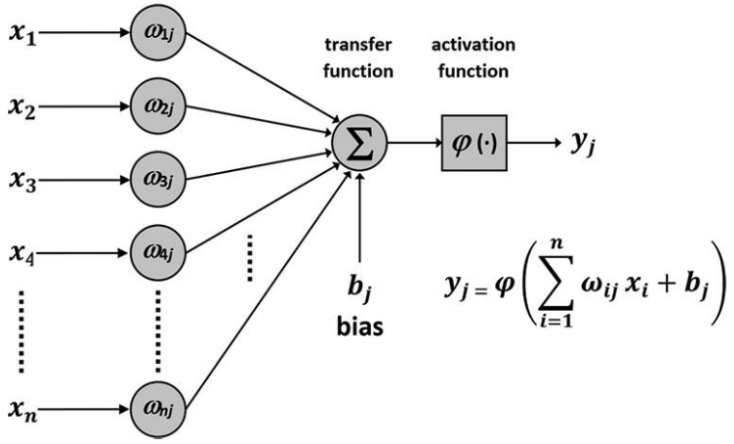

Figura 04. diagrama de una red neuronal artificial

Esta neurona es la unidad computacional que toma como entradas x1, x2, x3, x4, … , xn y produce una salida de acuerdo con la ecuación $y = h_{W,b}(x) = F(W^T x + b) = f(\sum_{i=1}^{n} W_i x_i + b)$, donde f (φ en la Figura 01) es la función de activación y su dominio e imagen son $f: \mathfrak{R} \to \mathfrak{R}$. se tomara $f(*)$ como la función sigmoide:

$$f(z) = \frac{1}{1 + e^{-z}}$$

Por tanto, nuestra neurona simple se corresponde exactamente con el algoritmo de regresión logística.

Aunque se va a utilizar la función sigmoide, cabe destacar que otra elección bastante común de f es la tangente hiberbólica o tanh:

$$f(z) = \tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

Además, investigaciones recientes han encontrado una función de activación diferente: la función lineal rectificada (rectified linear function o ReLU en inglés). Esta función trabaja bien en redes neuronales con Deep Learning y es diferente de la sigmoide y la tanh pues no está limitada y no es continuamente diferenciable. Esta función viene dada por:

$$f(z) = \max(0, z)$$

A continuación, se muestran en una sola gráfica las tres funciones mencionadas anteriormente para su mejor visualización y comparación:

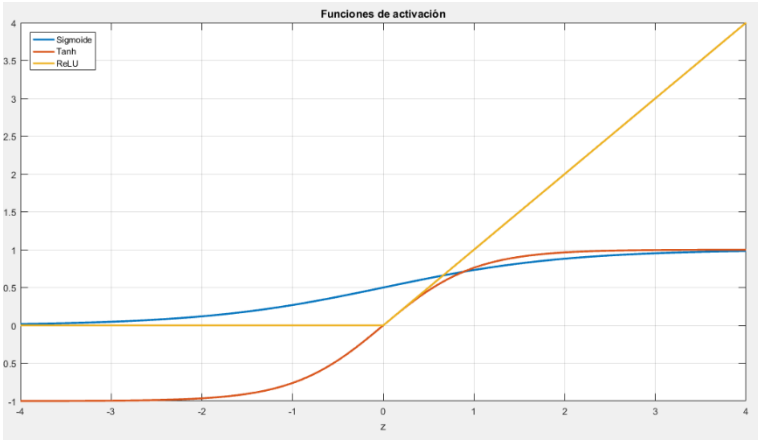

Figura 05. Comparación de funciones de activación

La función tanh (z) es una versión reescalada de la sigmoide, pues el rango de salida es de [−1,1] en vez de ser de [0,1]. La función lineal rectificada es una función lineal definida a trozos que satura el valor a 0 cuando la entrada z es menor que dicho valor.

Por último, comentar una igualdad que será de utilidad en el futuro: si $f(z) = 1/(1 + e^z)$ es la función sigmoide, entonces su derivada viene dada por $f'(z) = f(z)(1 - f(z)))$. Si, por el contrario, $f(z)$ es la función de tanh, entonces su derivada resulta ser $f'(z) = 1 - (f(z))^2$. La función lineal rectificada tiene un gradiente de 0 cuando $z \leq 0$ y 1 en cualquier otro caso.

3.14 Funcionalidad

La neurona realiza una suma de las señales eléctricas al llegar a sus dendritas. Esta suma se compara con un umbral para determinar si la neurona se excita, dando como resultado una generación de indicio a la dendrita de otra neurona. A finales del siglo diecinueve al comienzo no se encontraron señales en una neurona a estar sujeto a la atenuación en las sinapsis, es decir, la sinapsis ayudó a controlar la fuerza de la señal eléctrica pasado a la neurona. (Taylor, 2006)

3.15 Estructura

Las neuronas procesan la información en la membrana. La membrana regula tanto la transmisión y el procesamiento de la información. La suma de las señales y la comparación con un umbral es un combinado efecto de la membrana y el citoplasma. Si se genera un impulso se transmiten y las sinapsis establecen algunos transmisores de moléculas libres. La neurona tiene dendritas, un cuerpo celular y un axón. Los mismos tres elementos estarán presentes en nuestras unidades de computación artificiales. (Rojas, 1996)

3.16 Topología

En una red neuronal artificial, múltiples neuronas se interconectan para formar una red para facilitar la distribución informática. La configuración de las interconexiones se puede describir de manera eficiente con un grafo dirigido. Un grafo dirigido consta de nodos (en el caso de una red neuronal artificial, las neuronas, así como insumos externos) y arcos dirigidos (en el caso de una red neuronal artificial, enlaces sinápticos). La topología de la gráfica puede ser categorizado (ver figura 5) como sea acíclico o cíclico. (Hu & Hwang, 2002)

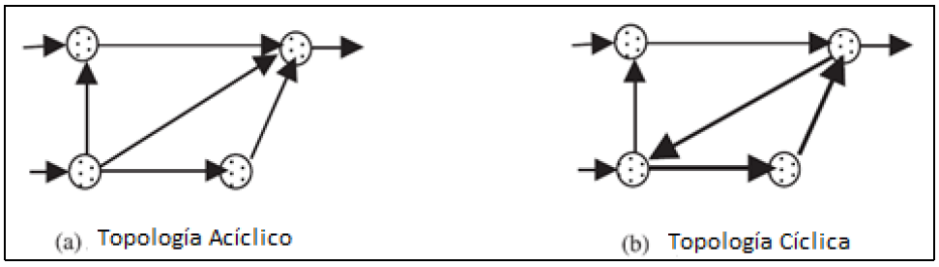

Figura 06. Topología de la red (a) Un grafo acíclico y (b) un gráfico cíclico. El ciclo en (b) se enfatiza con líneas gruesas.

Fuente: Hu, Y. H., & Hwang, J.-N. (2002). Handbook of Neural Network Signal Processing. U.S.: CRC Press LLC.

3.17 Arquitectura

Incluye definir el número de capas, el número de neuronas en cada capa, y el esquema de interconexión entre las neuronas. La selección del número de capas es controlada por el algoritmo de entrenamiento. Algunos algoritmos de entrenamiento pueden requerir solo una capa, mientras que otros pueden requerir un mínimo de tres capas. (Galushkin, 2007)

De manera que esencialmente el desarrollo podría ser caracterizado como un proceso de esculpir una red neuronal artificial de un número fijo de unidades y su azar interconexiones. (Quinlan, 2003)

Como ejemplo se tiene la arquitectura de una red de capa individual con una salida y dos entradas (ver figura 7) y la arquitectura del perceptrón multicapa (ver figura 8)

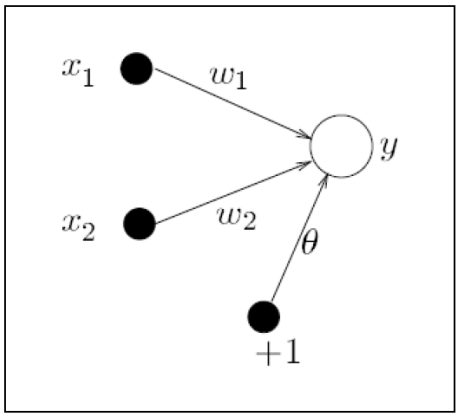

Figura 07. Red de capa individual con una salida y dos entradas.

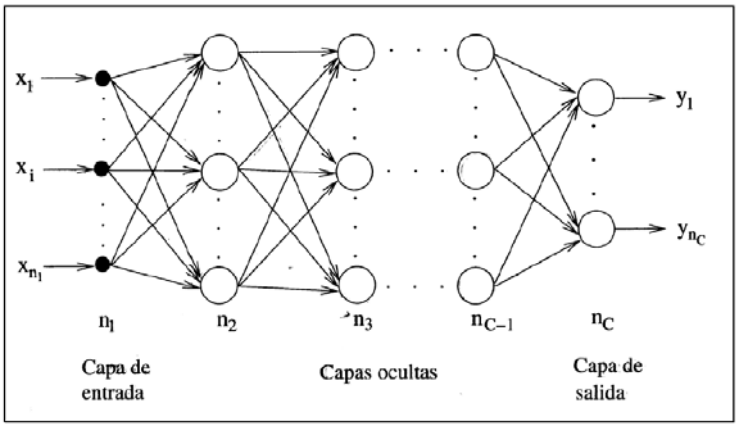

Figura 08. Arquitectura del perceptrón multicapa.

Fuente: Isasi Viñuela, P., & Galván León, I. M. (2004). Redes de neuronas artificiales: Un enfoque práctico. España: Pearson.

3.18 Back Propagation

En el algoritmo de Back Propagation se requiere de una capa de entrada, una capa de salida y una capa oculta. Se selecciona el número de capas ocultas en base a la complejidad del problema. El número de neuronas de entrada y salida de cada capa es específico para un problema. Las interconexiones entre neuronas son controladas por el algoritmo de entrenamiento y la naturaleza del problema. (Khare & Nagendra Shiva, 2007)

Si ahora se supone un set de entrenamiento $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$ de m ejemplos. Se entrenará a la red neuronal usando el algoritmo de gradient descen (descenso por gradiente). Es más, para un ejemplo concreto (x, y), se define la función coste con respecto a ese ejemplo como:

$$J(W, b; x, y) = \frac{1}{2} ||\boldsymbol{h}_{\boldsymbol{W},\boldsymbol{b}}(\boldsymbol{x}) - \boldsymbol{y}||^2$$

Dado un conjunto de m ejemplos de entrenamiento, se puede definir la función coste total como:

$$J(W, b) = \left[\frac{1}{m}\sum_{i=1}^{m} J(W, b; x^{(i)}, y^{(i)})\right] + \frac{\lambda}{2}\sum_{l=1}^{nl-1}\sum_{i=1}^{sl}\sum_{j=1}^{sl+1}\left(W_{ji}^{(l)}\right)^2 = \left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\left|\left|\boldsymbol{h}_{\boldsymbol{W},\boldsymbol{b}}(\boldsymbol{x}) - \boldsymbol{y}^{(\boldsymbol{i})}\right|\right|^2\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{nl-1}\sum_{i=1}^{sl}\sum_{j=1}^{sl+1}\left(W_{ji}^{l}\right)^2$$

Las dos ecuaciones anteriores difieren ligeramente pues los pesos tienen un weigth decay aplicado y, por el contrario, el termino de bias no.

Dado un ejemplo de set de entrenamiento (x, y), lo primero a realizar es calcular todas las actividades a lo largo de la red, incluido el valor de salida de la función

$h_{W,b}(x)$. Seguidamente, para cada nodo i de la capa l, seria conveniente calcular un "termino de error" $\delta_i^{(l)}$ que mida cuan responsable es dicho nodo del error cometido a la salida de la red. Para los nodos en la capa de salida, se puede medir directamente la diferencia ente el resultado y el valor real o target y definir dicha diferencia como $\delta_i^{(nl)}$, donde n_l es la capa de salida. Para las neuronas de las capas ocultas se calculará $\delta_i^{(l)}$ según una medida ponderada de los términos de error de los nodos que se toman $a_i^{(l)}$ como entrada.

A continuación, se muestra en detalle el algoritmo de retro programación:

1. Calcular las actividades de las n_l capas de la red utilizando el método de feedforward visto en el apartado 4.3, concretamente en la ecuación (4-4)
2. Para la i-ésima unidad en la capa de salida, establecer:

$$\delta_i^{nl} = \frac{\partial}{\partial z_i^{(nl)}}\left(\frac{1}{2}\left\|\boldsymbol{y} - \boldsymbol{h_{W,b}(x)}\right\|^2\right) = -\left(y_i - a_i^{(n_l)}\right).f'(z_i^{(n_l)})$$

3. Para $l = n_l - 1,\ n_1 - 2,\ n_1 - 3, \dots, 2$ establecer para cada nodo i de la capa l:

$$\delta_i^{(l)} = \left(\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)}\delta_j^{(l+1)}\right).f'(z_i^{(l)})$$

4. Calcular las derivadas parciales deseadas, que vienen dadas por:

$$\frac{\partial}{\partial W_{ij}^{(l)}}J(W,b;x,y) = a_j^{(l)}\delta_i^{(l+1)}$$

$$\frac{\partial}{\partial b_i^{(l)}}J(W,b;x,y) = \delta_i^{(l+1)}$$

Además, el algoritmo antes descrito se puede reescribir en forma matricial con el fin de acelerar el tiempo en el que se realiza los cálculos. Con este fin, se utilizará el operador producto elemento a elemento (también llamado producto Hadamard) y se denotará como "⊙". El algoritmo queda ahora como:

1. Calcular las actividades de las n_l capas de la red utilizando el método de feedforward visto en el apartado 4.3, concretamente en la ecuación (4-6)
2. Para la capa de salida, establecer:

$$\delta^{n_l} = -(y - a^{n_l}) \odot f'(z^{n_l})$$

3. Para $l = n_l - 1,\ n_1 - 2,\ n_1 - 3, \ldots, 2$ establecer:

$$\delta^{(l)} = \left((W^{(l)})^T \delta^{l+1}\right) \odot f'(z^l)$$

4. Calcular las derivadas parciales deseadas, que vienen dadas por:

$$\nabla_{W^{(l)}} J(W, b; x, y) = \delta^{l+1} (a^{(l)})^T$$

$$\nabla_{b^{(l)}} J(W, b; x, y) = \delta^{l+1}$$

Por último, el algoritmo completo para calcular el gradient descent expresado en pseudocodigo es el siguiente:

1. Establecer $\Delta W^{(l)} \coloneqq 0, \Delta b^{(l)} \coloneqq 0$ (matriz/vector de ceros) para todo l
2. Desde $i = 1\ hasta\ m$,
   a) Usar la retropropagación para calcular $\nabla_{W^{(l)}} J(W, b; x, y)$ y $\nabla_{b^{(l)}} J(W, b; x, y)$
   b) Establecer $\Delta W^{(l)} \coloneqq \Delta W^l + \nabla_{W^{(l)}} J(W, b; x, y)$
   c) Establecer $\Delta b^{(l)} \coloneqq \Delta b^{(l)} + \nabla_{b^{(l)}} J(W, b; x, y)$
3. Actualizar parámetros:

$$W^{(l)} = W^{(l)} - \alpha \left[\left(\frac{1}{m} \Delta W^{(l)}\right) + \lambda W^{(l)}\right]$$

$$b^{(l)} = b^{(l)} - \alpha \left[\left(\frac{1}{m} \Delta b^{(l)}\right)\right]$$

Para entrenar a la red neuronal, se puede realizar el método del gradient descent repetidamente para reducir la función coste $J(W, b)$

3.19 Enfoques

Galushkin (2007) considera que son posibles cuatro enfoques para la investigación de redes neuronales:

- Enfoque psicológico, cuando es necesario modelar algunos paradigmas psicológicos que requieran un desarrollo e investigación de la red neuronal con un poco de estructura definida.
- Enfoque neurofisiológico, cuando la red neuronal es desarrollada e investigada sobre la base de los conocimientos de la estructura de una parte del cerebro.
- Enfoque algorítmico, cuando se formula algún problema matemático y una red neuronal adecuada con el algoritmo correspondiente ajustado a esta solución está diseñado sobre la base de esta formulación.
- Enfoque sistemático que combina todos los enfoques mencionados anteriormente.

3.20 Aplicaciones

Las redes neuronales pueden utilizarse en un gran número y variedad de aplicaciones, tanto comerciales como militares.

Se pueden desarrollar redes neuronales en un periodo de tiempo razonable, con la capacidad de realizar tareas concretas mejor que otras tecnologías. Cuando se implementan mediante hardware (redes neuronales en chips VLSI), presentan una alta tolerancia a fallos del sistema y proporcionan un alto grado de paralelismo en el procesamiento de datos. Esto posibilita la inserción de redes neuronales de bajo coste en sistemas existentes y recientemente desarrollados.

Hay muchos tipos diferentes de redes neuronales; cada uno de los cuales tiene una aplicación particular más apropiada. Algunas aplicaciones comerciales son:

- Biología:
    - Aprender más acerca del cerebro y otros sistemas.
    - Obtención de modelos de la retina.
- Empresa:
    - Evaluación de probabilidad de formaciones geológicas y petrolíferas.
    - Identificación de candidatos para posiciones específicas.
    - Explotación de bases de datos.
    - Optimización de plazas y horarios en líneas de vuelo.
    - Optimización del flujo del tránsito controlando convenientemente la temporización de los semáforos.
    - Reconocimiento de caracteres escritos.
    - Modelado de sistemas para automatización y control.
- Medio ambiente:
    - Analizar tendencias y patrones.
    - Previsión del tiempo.
- Finanzas:
    - Previsión de la evolución de los precios.
    - Valoración del riesgo de los créditos.
    - Identificación de falsificaciones.
    - Interpretación de firmas.
- Manufacturación:
    - Robots automatizados y sistemas de control (visión artificial y sensores de presión, temperatura, gas, etc.).
    - Control de producción en líneas de procesos.

    - Inspección de la calidad.

- Medicina:
    - Analizadores del habla para ayudar en la audición de sordos profundos.
    - Diagnóstico y tratamiento a partir de síntomas y/o de datos analíticos (electrocardiograma, encefalogramas, análisis sanguíneo, etc.).
    - Monitorización en cirugías.
    - Predicción de reacciones adversas en los medicamentos.
    - Entendimiento de la causa de los ataques cardíacos.
- Militares:
    - Clasificación de las señales de radar.
    - Creación de armas inteligentes.
    - Optimización del uso de recursos escasos.
    - Reconocimiento y seguimiento en el tiro al blanco.

La mayoría de estas aplicaciones consisten en realizar un reconocimiento de patrones, como ser: buscar un patrón en una serie de ejemplos, clasificar patrones, completar una señal a partir de valores parciales o reconstruir el patrón correcto partiendo de uno distorsionado. Sin embargo, está creciendo el uso de redes neuronales en distintos tipos de sistemas de control.

Desde el punto de vista de los casos de aplicación, la ventaja de las redes neuronales reside en el procesado paralelo, adaptativo y no lineal.

El dominio de aplicación de las redes neuronales también se lo puede clasificar de la siguiente forma: asociación y clasificación, regeneración de patrones, regresión y generalización, y optimización.

3.21 Hipótesis de investigación

3.21.1 Hipótesis general

H0: Las redes neuronales artificiales no predicen la demanda de préstamo de libros de la biblioteca especializada de la EPIS - UNAJMA.

H1: Las redes neuronales artificiales predicen la demanda de préstamo de libros de la biblioteca especializada de la EPIS - UNAJMA

3.21.2 Hipótesis especifica

Primera Hipótesis Especifica

H0: La validación de la predicción de demanda de uso de libros de la biblioteca especializada EPIS - UNAJMA con el uso de redes neuronales, no será superior a 55%.

H1: La validación de la predicción de demanda de uso de libros de la biblioteca especializada EPIS - UNAJMA con el uso de redes neuronales, será superior a 55%.

Segunda Hipótesis Especifica

H0: El error de entrenamiento del modelo de predicción de demanda de uso de libros de la biblioteca especializada de la EPIS - UNAJMA con el uso de redes neuronales, no será inferior a 15%.

H1: El error de entrenamiento del modelo de predicción de demanda de uso de libros de la biblioteca especializada de la EPIS - UNAJMA con el uso de redes neuronales, será inferior a 15%.

3.22 Variables e indicadores

3.22.1 Identificación de variables

en el presente proyecto de tesis

Variable 1:

Redes neuronales artificiales.

Variable 2:

Demanda de uso de libros en la biblioteca especializada de la EPIS - UNAJMA.

3.22.2 Clasificador de Variables.

Para este proyecto, se utilizará dos variables:

Variables independientes : Redes neuronales artificiales.

Variables dependientes : Demanda de uso de libros en la biblioteca especializada de la EPIS - UNAJMA.

3.22.3 Definición de variables

- **Redes neuronales artificiales**

  Por la función que cumplen en la hipótesis: Variable independiente.

  Por su naturaleza: Desactiva.

  Por el método de estudio: Cualitativo.

  Por la posesión de la característica: Discreta.

  Por los valores que adquiere: Politómica.

- **Demanda de uso de libros en la biblioteca especializada de la EPIS - UNAJMA**

Por la función que cumplen en la hipótesis: Variable dependiente.

Por su naturaleza: Activa.

Por la posesión de la característica: Continuo.

Por los valores que adquiere: Politómica.

3.22.4 Operacionalización de variables

Variable independiente

- Variable independiente: Redes neuronales artificiales.
- Definición conceptual: Las redes neuronales artificiales son un paradigma de aprendizaje y procesamiento automático inspirado en la forma en que funciona el sistema nervioso.
- Definición operacional: Para tener una red neuronal la cual se ajuste de manera adecuada a los datos, se tiene que realizar varias configuraciones como el número de capas ocultas, el número de neuronas por cada capa y las funciones de transferencias.
- Indicadores: Valor de validación (aceptación de resultado obtenido por la red), valor del error cuadrático medio (error de entrenamiento de la red).
- Valores: Numéricos (expresado en porcentaje).

Variable dependiente.

- Variable dependiente: Demanda de uso de Libros de la biblioteca Especializada EPIS
- Definición conceptual: La demanda es la cantidad de libros que son solicitados por los estudiantes de la Escuela Profesional de Ingeniería de Sistemas de la Universidad Nacional José María Arguedas durante un periodo de tiempo.

- Definición operacional: Para realizar la predicción usando series de tiempo el cual es necesario que el conjunto de datos cumpla cierta hipótesis para que, en primer lugar, se busque un modelo estadístico que se ajuste a estos de manera adecuada y posteriormente realizar la predicción.
- Indicadores: Cantidad (el número total de los datos), valor máximo (el mayor número de los datos), valor mínimo (el menor número de los datos).
- Valores: Numéricos.

3.23 Diseño de investigación

Diseño no experimental

La presente investigación está enfocada en un diseño no experimental, de investigación transaccional descriptivo. Tiene como objetivo indagar la incidencia de las modalidades o niveles de una o más variables en una población. Es un estudio puramente descriptivo y cuando establecen hipótesis, éstas son también descriptivas de pronóstico de valores. (Hernández Sampieri, Fernández Collado, & Baptista Lucio, Metodología de la Investigación, 2010)

Población y muestra

Población: Se ha seleccionado mediciones de volumen de préstamos en unidades (und) agrupados por meses (enero 2017 - abril 2019), haciendo un total de veintiocho meses, datos que fueron proporcionados por la biblioteca especializada de la EPIS.

Muestra: Debido a que los datos están almacenados y son manipulados electrónicamente, no fue necesario de establecer una muestra, fue conveniente trabajar con toda la población, por lo tanto, hablamos de una población muestral.

3.24 Método de investigación

Se usará para el presente trabajo de investigación el método descriptivo puesto que se basa en observaciones y además se evidencia los cuatro factores psicológicos como son atención, sensación, percepción y reflexión.

3.25 Técnicas de instrumentos de acopio de datos

La recopilación de datos es importante para establecer las bases de parámetros para el entrenamiento de la red neuronal artificial y la búsqueda de diferente información documentaria que guarde relación con la investigación.

Métodos Pronóstico: Método de diseño de pronóstico con redes neuronales artificiales.

Técnicas: Algoritmo Back Propagation – Perceptrón multicapa.

Herramientas: Lenguaje de programación Python

Fuente: Los datos empleados fueron obtenidos de la biblioteca especializada de la Escuela Profesional de Ingeniería de Sistemas de la Universidad Nacional José María Arguedas.

3.26 Técnicas de análisis de datos

Para la implementación de las redes neuronales artificiales para el pronóstico de demanda de hielo industrial, no fue necesario realizar el pre-procesamiento de datos del volumen de préstamos de libros (und), dado que los datos se encontraron libres de ruido (no contiene valores nulos ni valores con formato inconsistente). La confiabilidad del resultado estuvo dada por el valor del test validación y del error cuadrático medio.

Selección de Pruebas Estadísticas

Por el diseño de la investigación utilizado no hubo pruebas estadísticas. Los resultados obtenidos de la red neuronal artificial sirvieron para verificar el porcentaje del test de validación y del error cuadrático medio.

CAPITULO IV

4 DISEÑO METODOLOGICO

4.1 Descripción

La Universidad Nacional José María Arguedas, tiene como actividad principal la formación académica de futuros profesionales entre las cuales se encuentra la Escuela Profesional de Ingeniería de Sistemas (EPIS), en merito a ello la Escuela profesional cuenta con una biblioteca especializada para el apoyo y consulta de las materias impartidas en la EPIS (Anexo 01. Malla Curricular de la EPIS).

Su misión es ser la universidad líder en la formación de profesionales de Ingeniería de sistemas a nivel regional, nacional e internacional. Por cuanto su visión es satisfacer integralmente a sus lectores aplicando políticas de mejora continua y un buen servicio a sus usuarios, siendo en este caso de estudio los estudiantes de Ingeniería de Sistemas de la UNAJMA.

Para el desarrollo de la red neuronal artificial, se utilizó la hoja de cálculo Excel y el lenguaje de programación Python versión 3.6.3 + Anaconda Navigator 1.9.7 y como interfaz Spyder 3.2.4. de igual manera se utilizaron las siguientes librerías scipy: 0.19.1, numpy: 1.13.3, matplotlib: 2.1.0, pandas: 0.20.3, statsmodels: 0.8.0 y sklearn: 0.19.1.

Tabla 01. Datos de la empresa

RUC	20527760314
Razón Social	Universidad Nacional José María Arguedas
Teléfono	(083) 421992
Dirección Legal	Jr. Juan Francisco Ramos 380
Provincia	Andahuaylas
Departamento	Apurímac
Estado	Activo

Fuente: Universidad Nacional José María Arguedas

4.2 Definición de Variables

Las variables de entrada de la red neuronal constituyen los valores de la serie temporal, la misma que se encuentra estructurada de la siguiente manera:

Tabla 02. Datos de entrenamiento

	Mes	Volumen de Demanda (Unidad/libros)		
		2017	2018	2019
Datos para la fase de Entrenamiento	Enero	79	88	100
	Febrero	77	76	82
	Marzo	96	72	95
	Abril	88	77	72
	Mayo	97	97	
	Junio	91	91	
	Julio	91	78	
	Agosto	99	70	
	Setiembre	97	92	
	Octubre	72	93	
	Noviembre	79	88	
	Diciembre	74	75	

A continuación, se grafica la serie de tiempo de los datos para la fase de entrenamiento (ver figura 09)

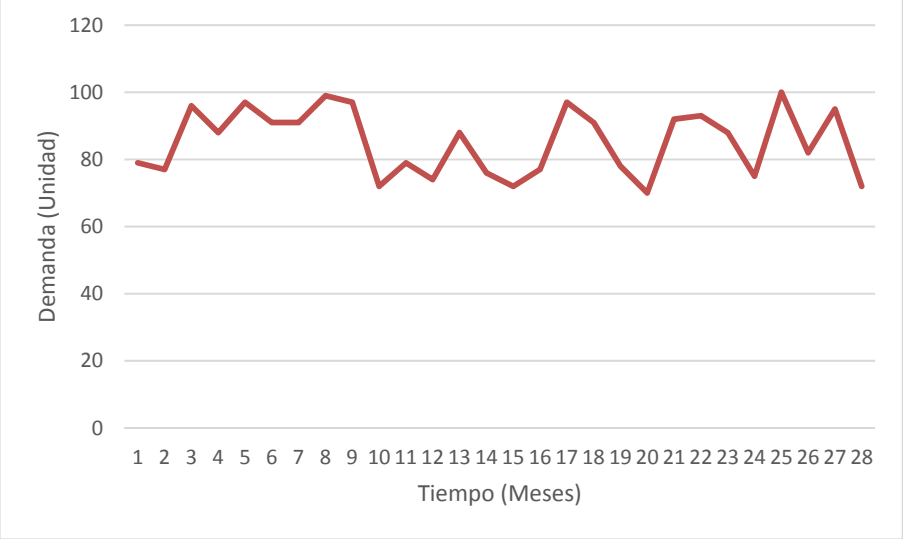

Figura 09. Serie de tiempo de los datos de entrenamiento.

Fuente: Elaboración propia.

4.2.1 Conjuntos de entrenamiento, validación y prueba

Python es un lenguaje de programación interpretado, orientado a objetos de alto nivel y con semántica dinámica. Su sintaxis hace énfasis en la legibilidad del código, lo que facilita su depuración y, por tanto, favorece la productividad. Ofrece la potencia y la flexibilidad de los lenguajes compilados con una curva de aprendizaje suave.

Aunque Python fue creado como lenguaje de programación de uso general, cuenta con una serie de librerías y entornos de desarrollo para cada una de las fases del proceso de Data Science. Esto, sumado a su potencia, su carácter open source y su facilidad de aprendizaje le ha llevado a tomar la delantera a otros lenguajes propios de la analítica de datos por medio de Data Science como pueden ser SAS (software comercial líder hasta el momento) y R (también open source, pero más propio de entornos académicos o de investigación).

Scikit-Learn es una de estas librerías gratuitas para Python. **Cuenta con algoritmos de clasificación, regresión, clustering y reducción de dimensionalidad.** Además, presenta la compatibilidad con otras librerías de Python como NumPy, SciPy y matplotlib.

La gran variedad de algoritmos y utilidades de Scikit-learn la convierten en la herramienta básica para empezar a programar y estructurar los sistemas de análisis datos y modelado estadístico. Los algoritmos de Scikit-Learn se combinan y depuran con otras estructuras de datos y aplicaciones externas como Pandas o PyBrain.

La ventaja de la programación en Python, y Scikit-Learn en concreto, es la variedad de módulos y algoritmos que facilitan el aprendizaje y trabajo del científico de datos en las primeras fases de su desarrollo. La formación de un

Máster en Data Science hace hincapié en estas ventajas, pero también prepara a sus alumnos para trabajar en otros lenguajes. La versatilidad y formación es la clave en el campo tecnológico.

4.2.1.1 Dividir el conjunto de datos

- Conjunto de entrenamiento: Conjunto de datos que se utilizará para aprender un modelo
- Conjunto de validación: Conjunto de datos que se utilizará para ajustar los hiper-parámetros (o parámetros, por simplificar) del modelo
- Conjunto de prueba: Conjunto de datos que se utilizará para evaluar el rendimiento del modelo finalmente obtenido

- Hay veces que no nos podemos permitir el lujo de no usar todos los datos para el entrenamiento
  - Validación cruzada
- **Separación de datos en scikit-learn: librería cross_validation**

  from sklearn.model_selection **import** train_test_split

  X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.25)

4.2.2 Selección de Topología

Se analizó el comportamiento de las topologías teniendo como factor de variación el número de neuronas en la capa de entrada. La red fue de tipo perceptrón multicapa.

4.2.2.1 Modelos de redes neuronales (supervisadas)

Perceptron multicapa (MLP) es un algoritmo de aprendizaje supervisado que aprende una función $f(.): R^m \longrightarrow R^0$ entrenando en un conjunto de datos, donde ***m*** es el número de dimensiones para la entrada y ***0*** es el número de

dimensiones para la salida. Dado un conjunto de características $X = x_1, x_2, \dots, x_m$ y un objetivo $\boldsymbol{y}$, puede aprender un aproximador de función no lineal para la clasificación o la regresión. Es diferente de la regresión logística, ya que, entre la entrada y la capa de salida, puede haber una o más capas no lineales, llamadas capas ocultas. La Figura 10 muestra un MLP de una capa oculta con salida escalar.

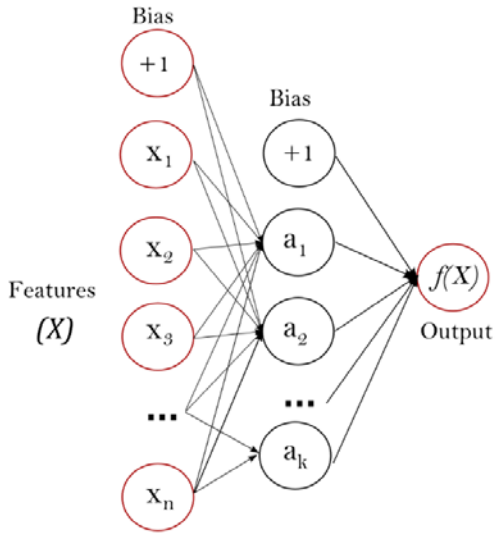

Figura 10. Una capa oculta de MLP.

La capa de la izquierda, conocida como la capa de entrada, consiste en un conjunto de neuronas $\{x_i / x_1, x_2, \dots, x_m\}$ representando las entidades de entrada. Cada neurona en la capa oculta transforma los valores de la capa anterior con una suma lineal ponderada $w_1x_1 + w_2x_2 + \cdots + w_mx_m$, seguido de una función de activación no lineal $g(.): R \longrightarrow R$- Me gusta la función del bronceado hiperbólico. La capa de salida recibe los valores de la última capa oculta y los transforma en valores de salida.

El módulo contiene los atributos públicos **coefs_** y **intercepts_. coefs_** es una lista de matrices de peso, donde la matriz de peso en el índice ***i*** representa los

pesos entre capas ***i*** y capa ***i+1*** . **intercepts_** es una lista de vectores de polarización, donde el vector en el índice ***i*** representa los valores de sesgo agregados a la capa ***i+1***.

Las ventajas de Perceptron multicapa son:

- Capacidad para aprender modelos no lineales.
- Capacidad para aprender modelos en tiempo real (aprendizaje en línea) utilizando partial_fit.

Las desventajas de Perceptron multicapa (MLP) incluyen:

- MLP con capas ocultas tiene una función de pérdida no convexa donde existe más de un mínimo local. Por lo tanto, diferentes inicializaciones de peso aleatorio pueden llevar a diferentes validaciones de precisión
- MLP requiere ajustar una serie de hiperparámetros como el número de neuronas ocultas, capas e iteraciones.
- MLP es sensible a la escala de características.

4.2.2.2 Regresión

La clase **MLPRegressor** implementa un perceptrón multicapa (MLP) que entrena usando la propagación hacia atrás sin función de activación en la capa de salida, que también puede verse como la función de identidad como función de activación. Por lo tanto, utiliza el error cuadrado como la función de pérdida, y la salida es un conjunto de valores continuos.

MLPRegressor también admite la regresión de múltiples salidas, en la que una muestra puede tener más de un objetivo.

4.2.2.3 Regularización

Ambos **MLPRegressor** y **MLPClassifier** utilizan el parámetro alpha para el término de regularización (L2 regularización) que ayuda a evitar el sobreajuste adaptando pesos con grandes magnitudes. La siguiente gráfica muestra la función de decisión variable con el valor de alfa.

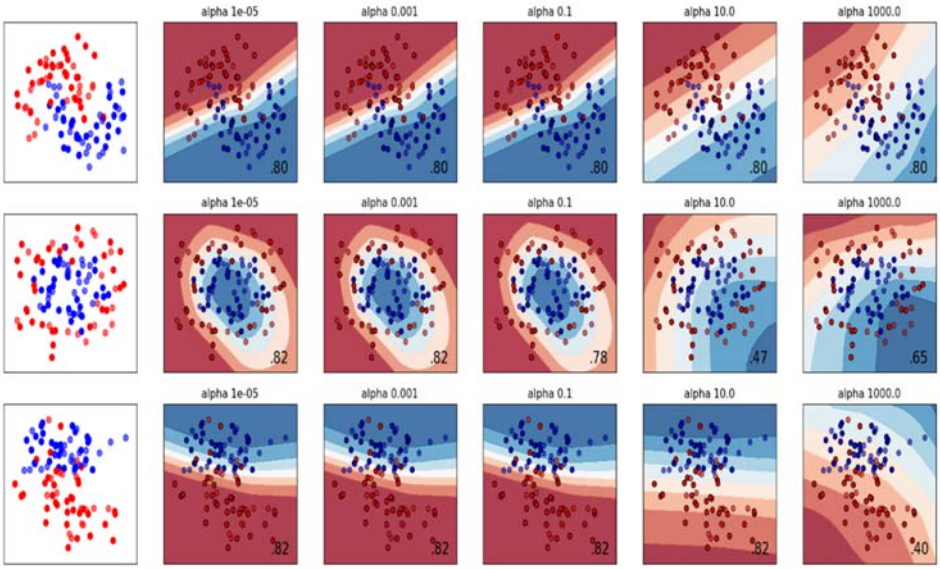

Figura 11. MLPRegressor y MLPClassifier utilizan el parámetro alpha

Varía la regularización en el Perceptrón multicapa

Una comparación de diferentes valores para el parámetro de regularización 'alfa' en conjuntos de datos sintéticos. La gráfica muestra que diferentes alfas producen diferentes funciones de decisión.

Alfa es un parámetro para el término de regularización, también conocido como término de penalización, que combate el sobreajuste ajustando el tamaño de

las ponderaciones. El aumento de alfa puede arreglar una alta varianza (un signo de sobrealimentación) al fomentar pesos más pequeños, lo que da como resultado una gráfica de límite de decisión que aparece con menos curvaturas. De manera similar, la disminución del valor alfa puede corregir un alto sesgo (un signo de falta de adaptación) al fomentar un mayor peso, lo que podría generar un límite de decisión más complicado.

4.2.2.4 Algoritmos

Trenes MLP que utilizan el **descenso de gradiente estocástico**, **Adam** o **L-BFGS**. Gradiente de gradiente estocástico (SGD) actualiza los parámetros utilizando el gradiente de la función de pérdida con respecto a un parámetro que necesita adaptación, es decir

$$\omega \leftarrow \omega - \eta \left(\alpha \frac{\partial R(\omega)}{\partial \omega} + \frac{\partial Loss}{\partial \omega} \right)$$

η está la velocidad de aprendizaje que controla el tamaño de paso en la búsqueda de espacio de parámetros. Es la función de pérdida utilizada para la red. $Loss$

Más detalles se pueden encontrar en la documentación de **SGD**.

Adam es similar a SGD en el sentido de que es un optimizador estocástico, pero puede ajustar automáticamente la cantidad para actualizar los parámetros basándose en estimaciones adaptativas de momentos de orden inferior.

Con SGD o Adam, la capacitación es compatible con el aprendizaje en línea y en mini lotes.

L-BFGS es un solucionador que se aproxima a la matriz de Hess que representa la derivada parcial de segundo orden de una función. Además, se

aproxima a la inversa de la matriz de Hesse para realizar actualizaciones de parámetros. La implementación utiliza la versión Scipy de **L-BFGS**.

Si el solucionador seleccionado es 'L-BFGS', la capacitación no admite el aprendizaje en línea ni en mini lotes.

4.2.2.5 Complejidad

Supongamos que hay muestras de entrenamiento, características, capas ocultas, cada una con neuronas, por simplicidad, y neuronas de salida. La complejidad del tiempo de la propagación hacia atrás es, donde está el número de iteraciones. Dado que la propagación hacia atrás tiene una complejidad de tiempo alta, es recomendable comenzar con un número menor de neuronas ocultas y pocas capas ocultas para el entrenamiento. $nmkhoO(n.m.h^k.o.i)$

4.2.2.6 Modelo Matemático

Dado un conjunto de ejemplos de entrenamiento donde y , una capa oculta, una neurona oculta MLP aprende la función donde y son parámetros del modelo. representan los pesos de la capa de entrada y la capa oculta, respectivamente; y representan el sesgo agregado a la capa oculta y la capa de salida, respectivamente. Es la función de activación, configurada por defecto como el bronceado hiperbólico. Se da como $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n) x_i \epsilon \boldsymbol{R}^n$ $y_i \epsilon \{0,1\} f(x) = W_2 g(W_1^T x + b_1) + b_2 W_1 \epsilon \boldsymbol{R}^m W_{2,} b_1, b_2 \epsilon \boldsymbol{R} W_1, W_2 b_1, b_2 g(.): R \longrightarrow R$

$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

Para la clasificación binaria, pasa a través de la función logística para obtener valores de salida entre cero y uno. Un umbral, establecido en 0.5, asignaría muestras de salidas mayores o iguales a 0.5 a la clase positiva, y el resto a la clase negativa. $f(x)g(z)\} = {}^{1}/_{(1 + e^{-z})}$

Si hay más de dos clases, sería un vector de tamaño (n_classes,). En lugar de pasar a través de la función logística, pasa a través de la función softmax, que se escribe como,$f(x)$

$$sofmax(z)_i = \frac{\exp(z_i)}{\sum_{l=1}^{k} \exp(z_l)}$$

donde representa el elemento th de la entrada a softmax, que corresponde a la clase, y es el número de clases. El resultado es un vector que contiene las probabilidades de que la muestra pertenezca a cada clase. La salida es la clase con mayor probabilidad. $z_i iiKx$

En regresión, la salida permanece como; por lo tanto, la función de activación de salida es solo la identidad. ,$f(x)$

MLP usa diferentes funciones de perdida dependiendo del tipo de problema. La función de perdida para la clasificación es Cross-Entropy, que en el caso binario se da como.

$$Loss(\hat{y}, y, W) = -yln\hat{y} - (1-y)\ln(1-\hat{y}) + \alpha\|W\|_2^2$$

dónde $\alpha\|W\|_2^2$ es un término de regularización L2 (también conocido como penalización) que penaliza los modelos complejos; y $\alpha > 0$ Es un hiperparámetro no negativo que controla la magnitud de la penalización.

Para la regresión, MLP usa la función de pérdida de error cuadrada; Escrito como,

$$Loss(\hat{y}, y, W) = \frac{1}{2}\|\hat{y} - y\|_2^2 \quad + \alpha\|W\|_2^2$$

A partir de los pesos aleatorios iniciales, el perceptrón multicapa (MLP) minimiza la función de pérdida al actualizar repetidamente estos pesos. Después de calcular la pérdida, una pasada hacia atrás la propaga desde la capa de salida a las capas anteriores, proporcionando a cada parámetro de peso un valor de actualización destinado a disminuir la pérdida.

En el descenso del gradiente, el gradiente. ∇LossW de la pérdida con respecto a los pesos se calcula y se deduce de W. Más formalmente, esto se expresa como,

$$W^{i+1} = W^i - \epsilon \nabla Loss_W^i$$

Donde i es el paso de iteración, y ϵ es la tasa de aprendizaje con un valor mayor 0.

El algoritmo se detiene cuando alcanza un número máximo preestablecido de iteraciones; o cuando la mejora en la pérdida está por debajo de un cierto número pequeño.

4.2.3 Normalización de Datos

El proceso de normalización de datos fue necesario para la investigación puesto que la respuesta de las neuronas viene dada por la función de activación sigmoide. Esta función está diseñada para devolver valores entre 0 y 1. La normalización de los datos no supone ninguna pérdida de información. Con este proceso se obtiene la serie temporal multiplicada por un factor y desplazada de tal forma que los valores están dentro del rango [0,1].

En normalización los datos **z** son reescalados de manera tal que cualquier específico **z** será ahora 0 ≤ z ≤ 1, y se realiza a través de esta fórmula:

$$z = \frac{x - \min(x)}{\max(x) - \min(x)}$$

a) La normalización hace que el entrenamiento sea menos sensible a la escala de características, por lo que podemos resolver mejor los coeficientes.

Ejemplo consideremos el conjunto de datos de precios de vivienda en california, que tiene características como **number of bedrooms** y la **median household income.** Cada uno tiene diferentes unidades y escalas, asi que considere estos atributos de características

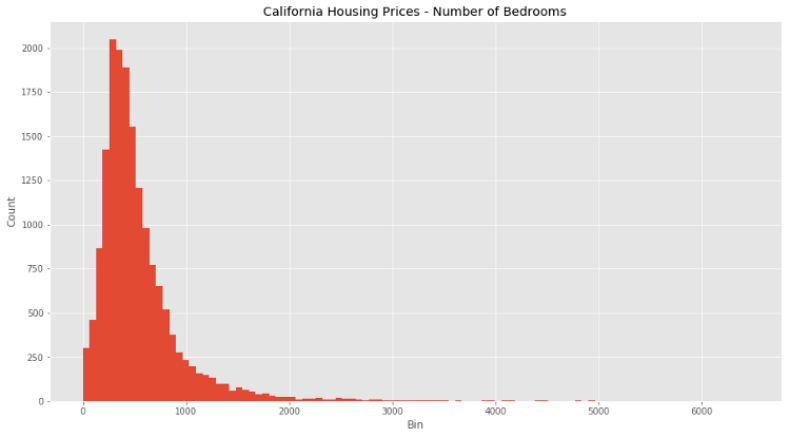

Figura 12. Datos Originales de Number of Bedrooms

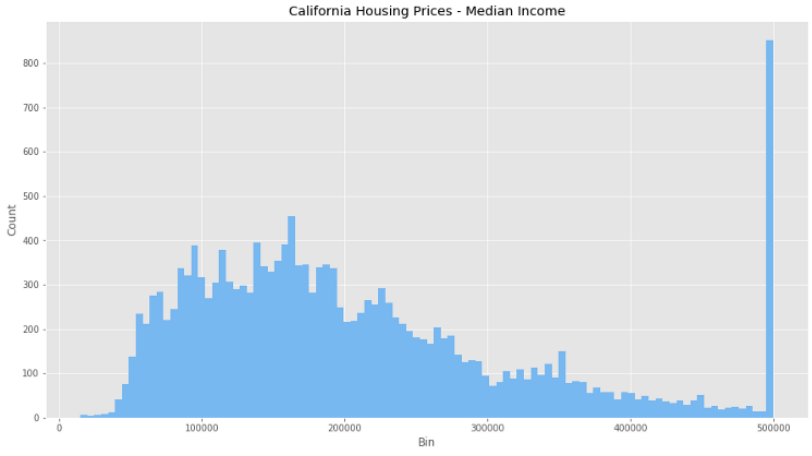

Figura 13. Datos Originales de Median Income

Podemos ver que hay algunos comportamientos extraños con ambas características (¿cómo podemos tener el Number bedrooms más de 1000?), Así como problemas atípicos y de agrupamiento masivos. También tenemos una agrupación de Median income $ 500,000, por lo que el conjunto de datos probablemente coloque a alguien sobre ese soporte en esa bandeja. Va a ser difícil equiparar ambas características como están ahora.

Veamos lo que la Normalizacion hace

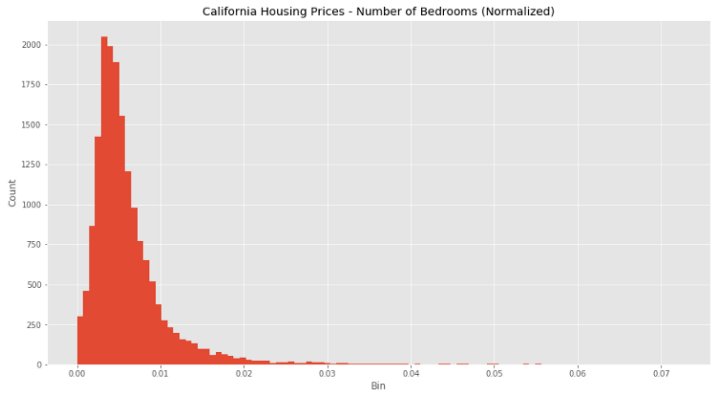

Figura 14. Datos Normalizados de Number of Bedrooms

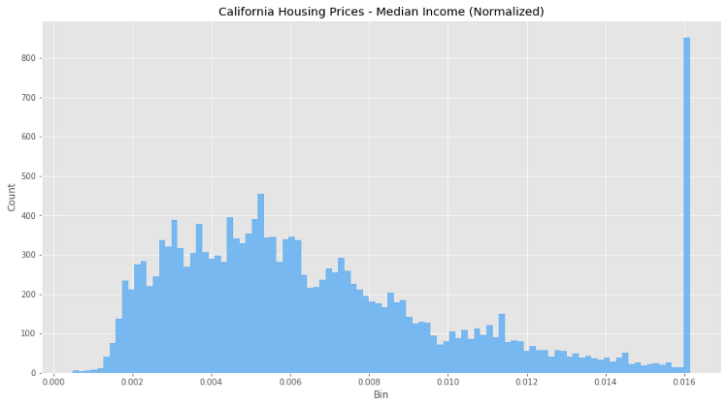

Figura 15. Datos Normalizados de Median Income

Todos los valores están ahora entre 0 y 1, y los valores atípicos se han ido, pero aún permanecen visibles dentro de nuestros datos normalizados. Sin embargo, nuestras características ahora son más coherentes entre sí, lo que nos permitirá evaluar mejor los resultados de nuestros futuros modelos.

b) El uso de un método de normalización mejorará el análisis de múltiples modelos.

   Además, si tuviéramos que usar algún algoritmo en este conjunto de datos antes de normalizarlo, sería difícil (posiblemente no posible) converger los vectores debido a los problemas de escalado. La normalización condiciona mejor los datos para la convergencia.

c) La normalización asegurará que un problema de convergencia no tenga una variación masiva, haciendo que la optimización sea factible.

   Pero a veces es posible que no desees **normalize** tus datos.

   Los datos proporcionados son proporcionales, por lo que normalizing podrían no proporcionar estimadores correctos. O bien, la escala entre las características de sus datos es importante, por lo que desea mantenerlos en su conjunto de datos. Debe pensar en sus datos y comprender si las transformaciones que está aplicando están en línea con los resultados que está buscando.

Tenga en cuenta que existe un debate que indica que es mejor tener los valores de entrada centrados en 0: estandarización, en lugar de entre 0 y 1. Por lo tanto, hacer su investigación también es importante, para que comprenda qué tipo de datos necesita su modelo.

4.2.3.1 La Estandarización de los Datos

Aquí los datos z se reajustarán tal que $\mu = 0$ y $\sigma = 1$, y se realiza a través de esta fórmula:

$$z = \frac{x_i - \mu}{\sigma}$$

a) Compara las características que tienen diferentes unidades o escalas.

Considere nuestros datos anteriores con ***housing*** y ***income***, ambos tienen diferentes escalas y unidades. Podemos comenzar a comparar estas características y usarlas en nuestros modelos una vez que las tengamos ***standardized***.

Más adelante, cuando esté ejecutando modelos (regresión logística, SVM, perceptrones, redes neuronales, etc.), los pesos estimados se actualizarán de manera similar en lugar de a velocidades diferentes durante el proceso de creación. Esto le dará resultados más precisos cuando los datos hayan sido los primeros ***standardized***.

Vamos a verlo en python:

```
de preprocesamiento de importación sklearn
# Obtener nombres de columna primeros
nombres = df.columns
# Crear el
escalador de objetos de escalador = preprocessing.StandardScaler ()
# Ajuste sus datos en el objeto del escalador
scaled_df = scaler.fit_transform (df)
scaled_df = pd.DataFrame (scaled_df, columns = names)
```

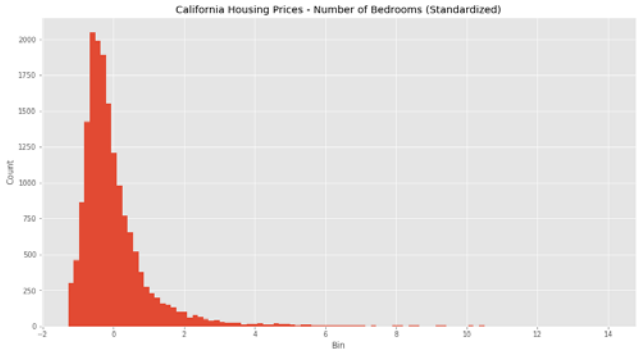

Figura 16. Datos estandarizados de Number of Bedrooms

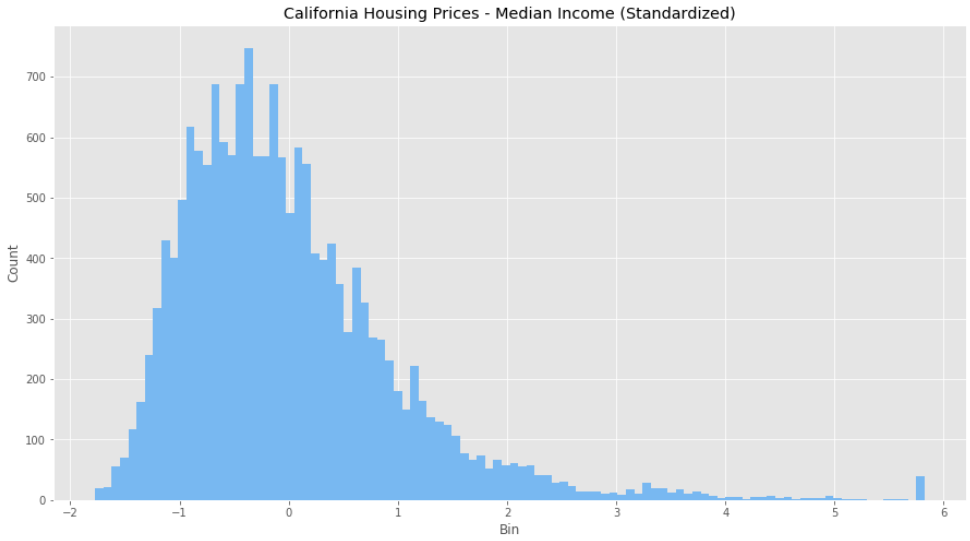

Figura 17. Datos Esctandarizados de Median Income

Parece que hemos ajustado todos los valores atípicos en ***bedrooms*** y ***income***, y tenemos una distribución mucho más normal para cada característica. No es perfecto, pero los datos están en mucho mejor estado de lo que era cuando ejecutamos el nuestro ***normalization***. Parece que, debido a la gran diferencia en escalas y unidades, standardizing es una mejor transformación para este conjunto de datos.

b) La estandarización tiende a hacer que el proceso de entrenamiento se comporte bien porque se mejora la condición numérica de los problemas de optimización.

   Considere si está haciendo PCA, la salida solo se puede interpretar correctamente cuando las características se han centrado en sus medios. Una vez más, comprender lo que quiere lograr y el modelo que utilizará, son condiciones necesarias para comprender las diferentes decisiones de transformación.

   Sin embargo, si estandariza sus datos, tenga en cuenta que podría estar descartando cierta información. Si esa información no es necesaria, el proceso puede ser útil, de lo contrario, impedirá sus resultados.

4.2.4 Fase de Entrenamiento

La fase de entrenamiento de una red neuronal sirvió para encontrar los pesos y umbrales óptimos, inicialmente aleatorios. En cada etapa se calculó el error cometido por la red cuando intenta predecir el resultado. La clave del proceso de entrenamiento fue modificar los pesos y umbrales en la dirección que minimiza el error. Para ello se usaron solo los patrones de entrenamiento y una reiteración de, a veces, cientos de etapas en las que se repiten los mismos pasos hasta llegar a la estabilización de los pesos y umbrales.

Al final del proceso, cuando los valores de los pesos y umbrales se estabilizaron asintóticamente, la red neuronal se dice que ha aprendido a pronosticar de forma aproximada la cantidad demandada de libros en la biblioteca.

El proceso de entrenamiento de las redes neuronales se hiso con el lenguaje de programación Python en donde se utilizó la siguiente librería:

```
from sklearn.neural_network import MLPRegressor
```

4.2.5 Implementación

Se llevó a cabo la implementación del modelo haciendo uso de la técnica de Back Propagation para el perceptrón multicapa. Para proceder con la implementación de este modelo, antes presentamos un resumen de los datos (ver tabla) más relevantes del mismo que nos ayudará a internarnos en el proceso de su implementación del código en python.

Tabla Resumen de datos	
Data pre-procesa	28 registros
Tipo de variable	Cuantitativa (unidades de libros)
Frecuencia de data	Mensual
Tamaño del Conjunto de Entrenamiento	28
Numero de Neuronas de Entrada	Aleatorio (sklearn)
Numero de capas ocultas	1
Numero de Neuronas Ocultas	(3x3)
Numero de Neuronas de Salida	1
Función de transferencia	Funcion sigmoide
Función de error	Error cuadrático medio

Fuente: Elaboración propia.

Se procedió a explicar las instrucciones necesarias para implementar el modelo, si desea estudiar el código completo (ver anexo).

Se comenzó a desarrollar el modelo de redes neuronales artificiales.

Las redes neuronales tipo perceptrón usaron el método del descenso del gradiente para minimizar la función error. Antes de explicarlo es conveniente ver porqué el método estándar de optimización del cálculo no es el adecuado.

Supongamos que tenemos una función (w). El objetivo es calcular $w*$, que es el valor de w que minimiza (w). Por ejemplo, la función $(w)=(w-1)2$. El valor que anula la derivada es el mínimo $f'(w)=2(w-1)=0 \rightarrow w*=1$.

En redes neuronales este método no es efectivo porque la función a minimizar depende de demasiadas variables y tiene una forma demasiado compleja como para trabajar con ella de forma general.

Para esto debemos tener en claro que un ajuste de curva lineal se define como una función lineal que modela el conjunto de datos dado como

$$f(x) = w_0 x + w_1$$

de tal forma que el sistema debe estimar los parámetros w_0 y w_1. Este modelo no se ajusta al ruido, porque de ser así se podría tener un sobreajuste de curva.

El método de descenso de gradiente nos permite estimar cada nuevo parámetro a partir del anterior, teniendo en cuenta la derivada de la función de coste, definida como el error cuadrático entre los datos conocidos y los datos encontrados con el modelo. De esta manera, el parámetro actual se calcula como:

$$w[n+1] = w[n] - K\frac{\partial L(w[n])}{\partial w[n]}$$

Con

$$K = \begin{pmatrix} k_0 \\ k_1 \end{pmatrix}$$

Y $L(w[n])$, $L(w[n])$, correspondiente a la función de coste. Los parámetros se estiman utilizando las ecuaciones recursivas:

$$w_0[n+1] = w_0[n] + k\sum_{i+1}^{m}(y_i - f(x;w))$$

$$w_1[n+1] = w_1[n] + k\sum_{i+1}^{m}\big(y_i - f(x;w)\big) * x_i$$

Donde **k** es la constante de paso, y_i es el valor real del dato, y $f(x;w)$ es el valor al remplazar los parámetros en la función.

Método descenso de gradiente con tasa de aprendizaje variable: es una variante del método de descenso de gradiente, en la cual la constante de paso (tasa de aprendizaje), que en el algoritmo original es constante (*k*), se calcula en línea, para hacer el ajuste más fi no y disminuir las oscilaciones. Dicha constante se determina a través de un problema de optimización unidimensional, utilizando

ajuste cuadrático, ajuste cúbico o búsqueda dorada.

Método descenso de gradiente modificado: nuestra propuesta de modificación de este método tiene tres características que la diferencian del método de gradiente clásico y con tasa de aprendizaje variable:

- Se utilizan dos constantes de paso, una por cada parámetro que se estima,
- La constante de aso es negativa (*k<0*) en el cálculo de los parámetros

$$w_1[n+1]w_1[n+1]$$

- La dependencia de a derivada de la función de coste se realiza en términos de la función de coste se realiza en términos del otro parámetro, es decir, que para cálculo de w_0 se tienen en cuenta la derivada de L en función de w_1, y viceversa.

En conclusión, las ecuaciones recursivas quedan como:

$$w_0[n+1] = w_0[n] + k_2 \sum_{i+1}^{m} \big(y_i - f(x;w)\big) * x_i$$

$$para\ k_2 > 0$$

$$w_1[n+1] = w_1[n] + k_3 \sum_{i+1}^{m} (y_i - f(x;w))$$

$$para\ k_3 > 0$$

Método gradiente conjugado: utiliza el valor del gradiente anterior para calcular el paso total entre una iteración y otra.

CAPITULO V

5 RESULTADOS

Al desarrollar el modelo de redes neuronales no existió previamente una estructura preliminar para poder resolver el problema que se plantea. Solo es afirmable de la existencia de una estructura, la cual puede dar respuesta adecuadas al problema.

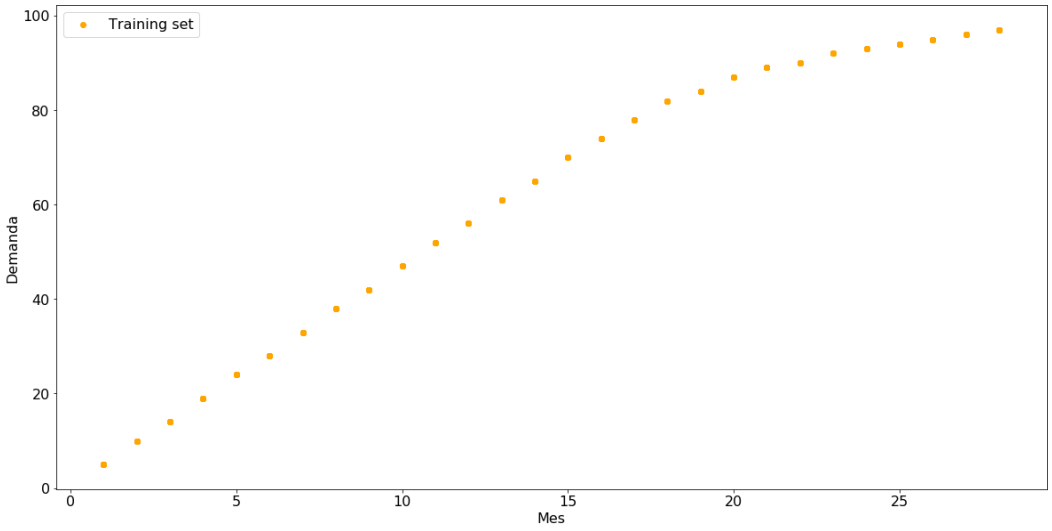

Gráfico01: Datos de entrenamiento.

Fuente: Elaboración propia.

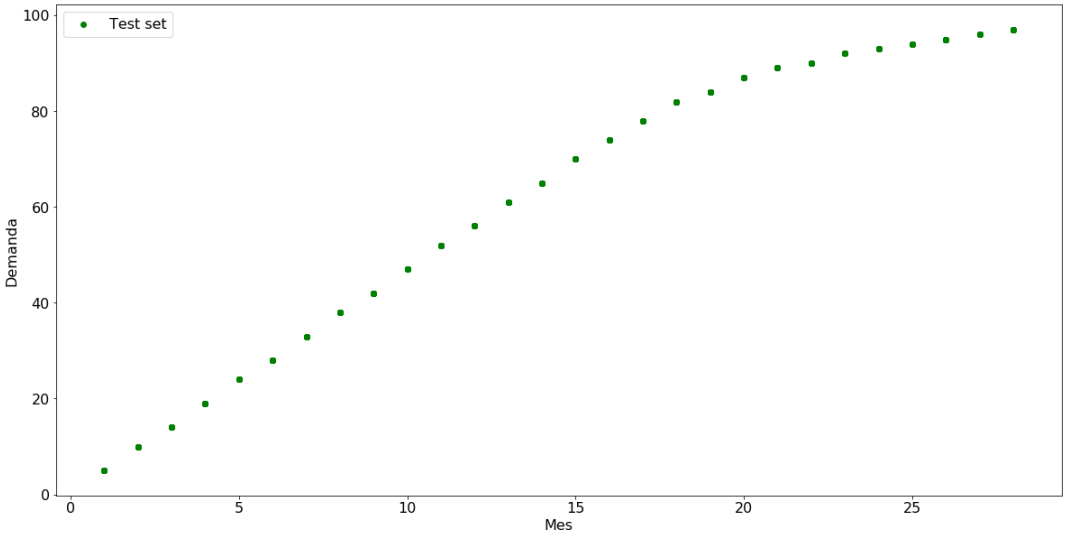

Gráfico02: Datos de prueba.

Fuente: Elaboración propia.

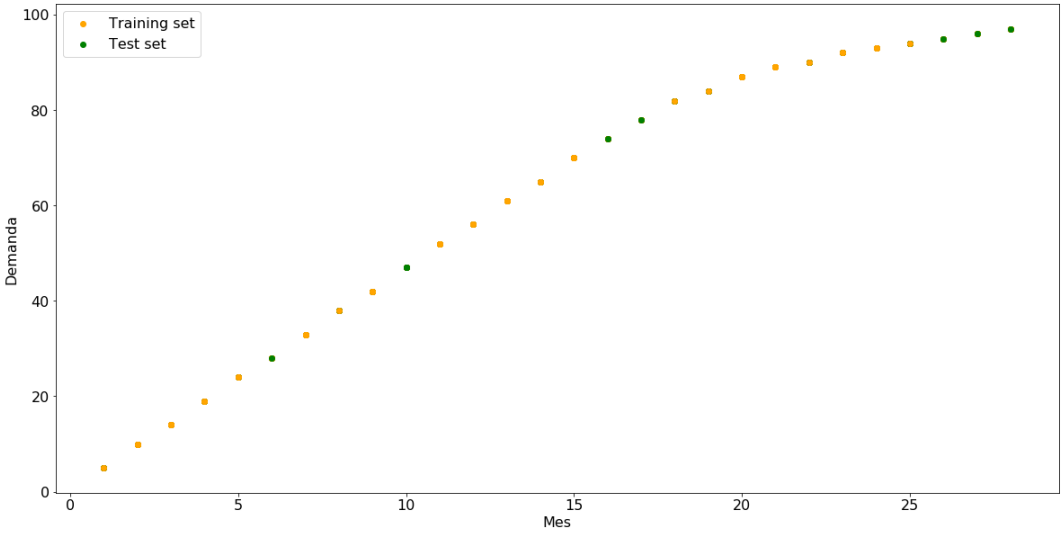

Gráfico 03: Datos de prueba vs datos de entrenamiento.

Fuente: Elaboración propia.

5.1 Normalización y estandarización de datos

Los datos pertenecientes a una distribución normal se pueden estandarizar o normalizar, procedimiento que se obtiene utilizando las siguientes funciones

5.1.1 Para normalizar: se utiliza la siguiente formula

$$z = \frac{x - \min(x)}{\max(x) - \min(x)}$$

En Python se utilizará la siguientes librería y funciones para normalizar:

Librería: "**from** sklearn **import** preprocessing"

Función: scaler=preprocessing.Normalizer(norm='l2',copy=True)

Tabla 03: Datos normalizados

	Mes	Volumen de Demanda (Unidad/Libro)		
		2017	2018	2019
Datos para la Fase de Entrenamiento	Enero	0.99991989	0.98926374	0.9701425
	Febrero	0.99966285	0.98345319	0.95323064
	Marzo	0.99951208	0.97898042	0.96190511
	Abril	0.99896854	0.97908607	0.93200467
	Mayo	0.99867413	0.98498733	
	Junio	0.99783341	0.98099311	
	Julio	0.99705449	0.97159024	
	Agosto	0.99675093	0.96152395	
	Setiembre	0.9957232	0.97492423	
	Octubre	0.99049227	0.97314201	
	Noviembre	0.99044476	0.96750042	
	Diciembre	0.98710548	0.95242415	

Fuente: Elaboración Propia

Resultados obtenidos de los datos Originales (Tabla 02)

Media: 85.21428571428571

Desviación Estándar: 9.837031857587723

Valor Máximo: 100

Valor Mínimo: 70

Resultados obtenidos después de efectuada la normalización (Tabla 03)

Media: 0.9809747778321599

Desviación Estándar: 0.017452808873137327

Valor Máximo: 0.99991989426

Valor Mínimo: 0.932004671541

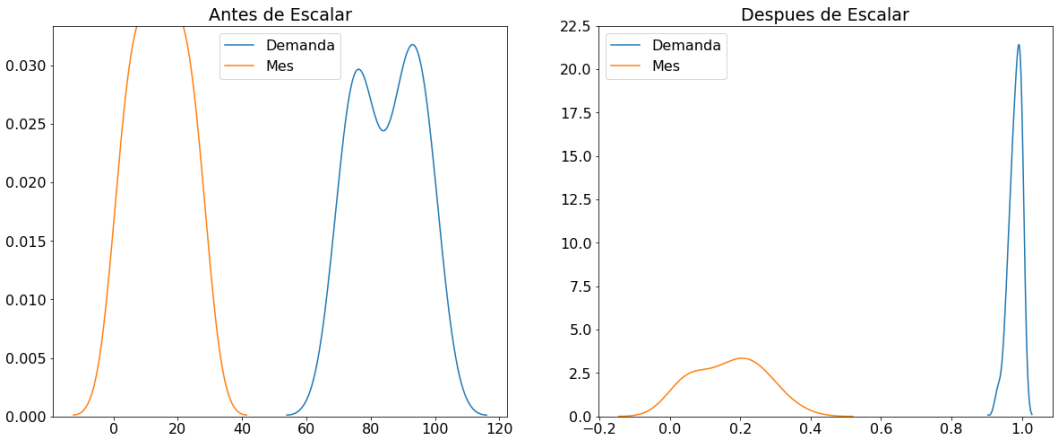

Gráfico 04: Datos de prueba vs datos de entrenamiento Normalizados

Fuente: Elaboración propia.

5.1.2 Para estandarizar: se utiliza la siguiente formula

$$z = \frac{x_i - \mu}{\sigma}$$

En Python se utilizará la siguientes librería y funciones para normalizar:

Librería: "**from** sklearn **import** preprocessing"

Función: scaler=preprocessing.Normalizer(norm='l2',copy=True)

Tabla 04: Datos Estandarizados

	Mes	Volumen de Demanda (Unidad/Libro)		
		2017	2018	2019
Datos para la Fase de Entrenamiento	Enero	0.01265721	0.14614123	0.24253563
	Febrero	0.02596527	0.18116243	0.30224386
	Marzo	0.03123475	0.20395425	0.27338356

Abril	0.04540766	0.20344646	0.36244626
Mayo	0.05147805	0.17262665	
Junio	0.06579121	0.19404259	
Julio	0.0766965	0.23666942	
Agosto	0.08054553	0.27472113	
Setiembre	0.09238669	0.22253705	
Octubre	0.13756837	0.23020564	
Noviembre	0.13791003	0.25286943	
Diciembre	0.16007116	0.30477573	

Fuente: Elaboración Propia

Resultados obtenidos de los datos Originales (Tabla 02)

Media: 85.21428571428571

Desviación Estándar: 9.837031857587723

Valor Máximo: 100

Valor Mínimo: 70

Resultados obtenidos después de efectuada la Estandarización (Tabla 03)

Media: 6.740639792367022e-16

Desviación Estándar: 1.0183501544346312

Valor Máximo: 1.53064813089

Valor Mínimo: -1.57501474338

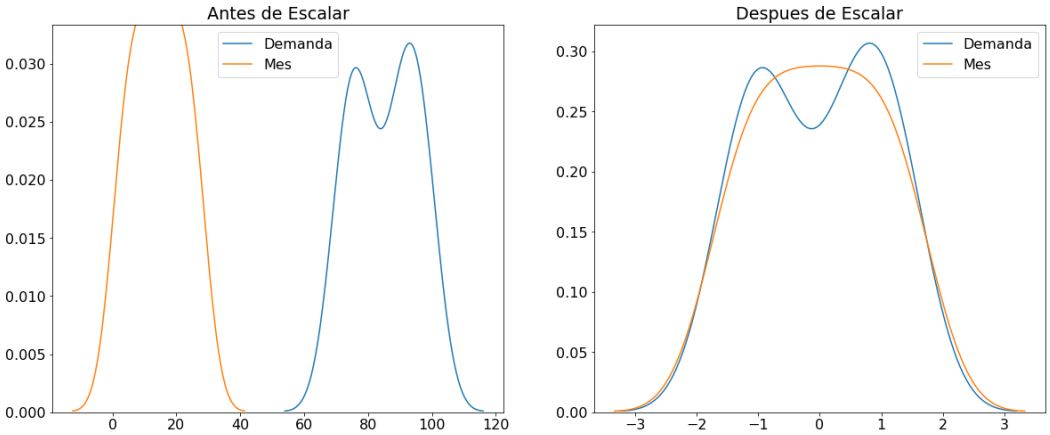

Gráfico 05: Datos de prueba vs datos de entrenamiento Estandarizados

Fuente: Elaboración propia.

5.2 Presentación de los resultados obtenidos

Datos de los meses

0	1	10	11	20	21
1	2	11	12	21	22
2	3	12	13	22	23
3	4	13	14	23	24
4	5	14	15	24	25
5	6	15	16	25	26
6	7	16	17	26	27
7	8	17	18	27	28
8	9	18	19		
9	10	19	20		

Datos Demanda

0	5	4	24	8	42
1	10	5	28	9	47
2	14	6	33	10	52
3	19	7	38	11	56

12	61	18	84	24	94
13	65	19	87	25	95
14	70	20	89	26	96
15	74	21	90	27	97
16	78	22	92		
17	82	23	93		

5.2.1 Segmentación de datos de entrenamiento y prueba

Para la segmentación de datos de entrenamiento y prueba se efectua mediante la siguiente función:

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.25)

Es de mencionar que dicha segmentación de datos de entrenamiento X_train y los datos de Control X_test se efectúan mediante dicha función, considerando para ello que del total de datos ingresados el 0.75 serán considerado como datos de entrenamiento y el 0.25 como datos de control.

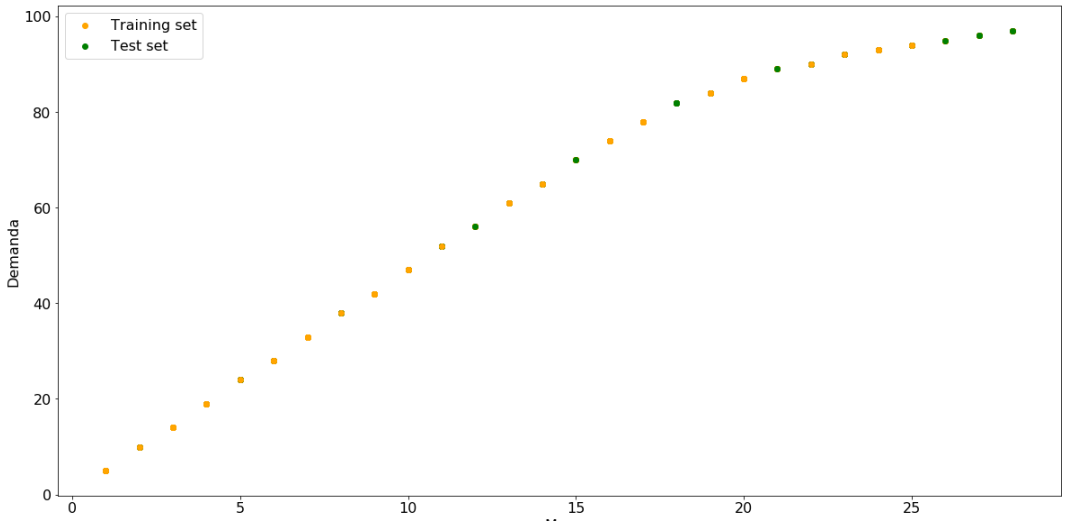

Grafico 06: Segmentación de datos para predicción de la demanda del mes 27.

Fuente: Elaboración propia.

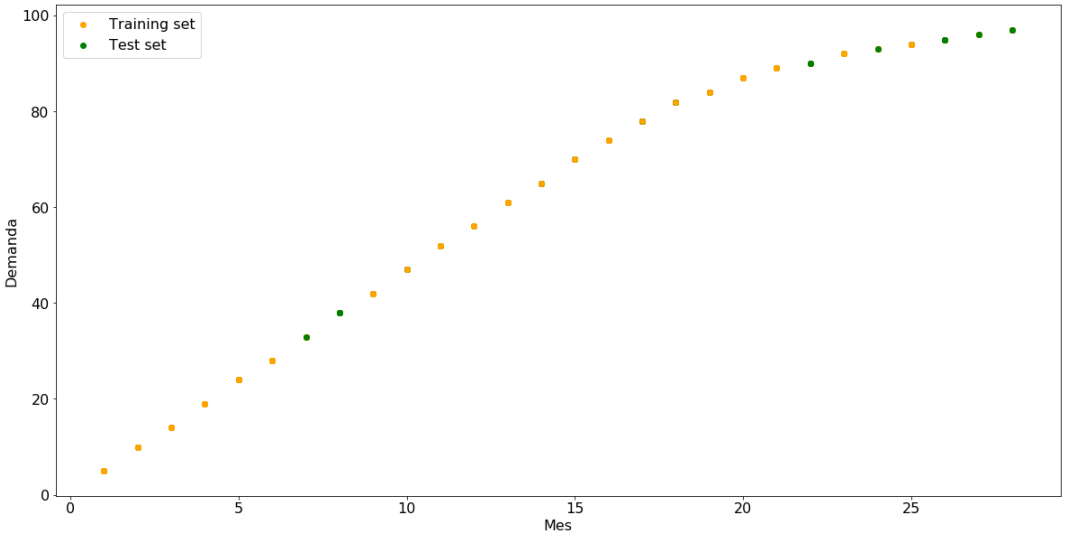

Grafico 07: Segmentación de datos para predicción de la demanda del mes 27.

Fuente: Elaboración propia.

De las figuras mostradas podemos precisar que toda vez que se efectué una predicción, la segmentación de datos control y los datos de control se efectuarán de forma aleatoria, de igual forma ocurrirá para la predicción de demanda de meses futuros como es el caso del mes 29

Figura: Segmentación de datos para predicción de la demanda del mes 29, teniendo un tamaño de Datos control de 0.25 y un SCORE de 0.98.

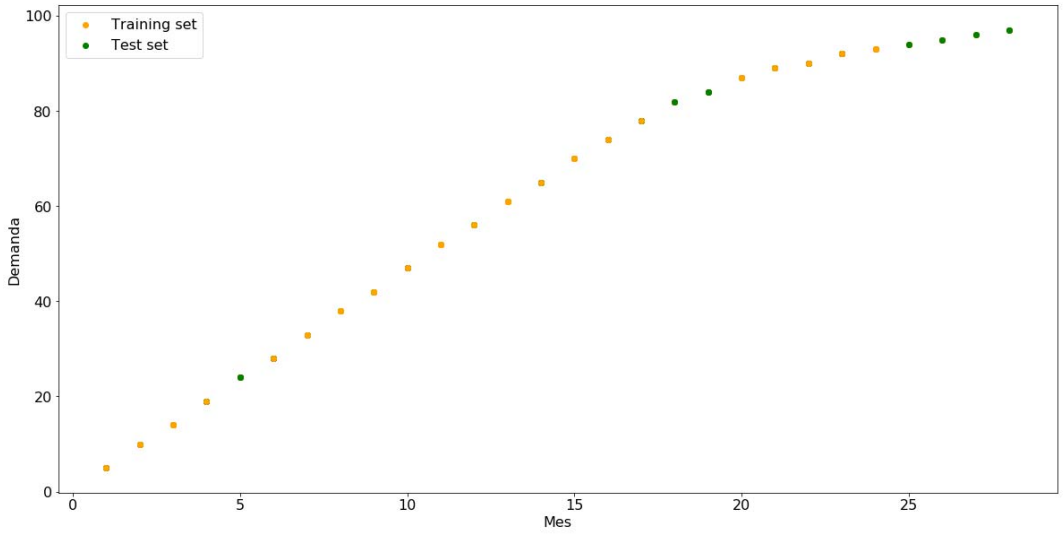

Grafico 08: Segmentación de datos para predicción de la demanda del mes 29.

Fuente: Elaboración propia

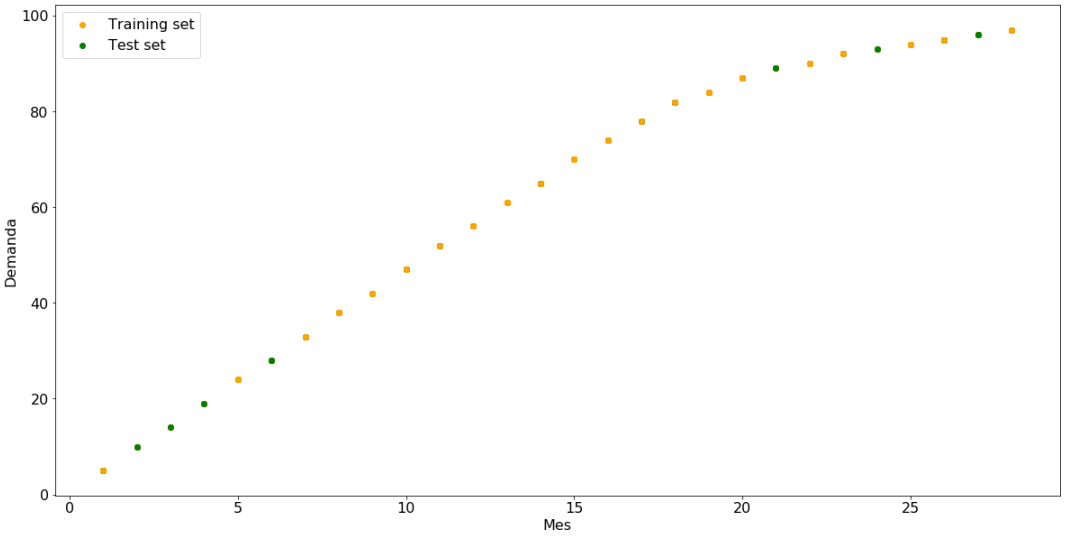

Grafico 09: Segmentación de datos para predicción de la demanda del mes 29.

Fuente: Elaboración propia

En desarrollo del presente proyecto de investigación fue importante contar con la información necesaria, debemos también mencionar que se contó con información de 28 meses, para efectuar las predicciones se efectuó la segmentación en datos

de entrenamiento y datos de test considerándose para ello 0.75 para datos de entrenamiento y 0.25 como datos de control, segmentación que fue realizada de forma aleatoria. Es por ello que los resultados de las predicciones de demanda para un mismo mes serán diferentes como también el número de iteraciones, pero toda predicción mediante iteraciones sucesivas debe superar el SCORE de 0.98, para una mejor precisión en el resultado de la predicción.

5.2.2 Predicción de la demanda para el mes 27

Aproximaciones del SCORE: la condición principal que se debe de cumplir es que el SCORE de predicción sea mayor a 0.98, esto precisa que se ajusta la predicción de la demanda del mes 27 a lo más real posible a partir de los datos de entrenamiento y prueba ingresados para la predicción:

mlr.score(X_train,y_train) > 0.98

Obteniéndose como se muestra a continuación seis iteraciones en donde la Iteración 06 es mayor a 0.98

Iteración 01: 0.97386265118

Iteración 02: 0.959201885589

Iteración 03: 0.962575761323

Iteración 04: 0.974274482742

Iteración 05: 0.959930441142

Iteración 06: 0.984709286263

El presente resultado calculado para la predicción del mes 27 de los datos ingresado fue efectuado mediante el siguiente comando:

"Predicción para el Mes=27 ",mlr.predict(27)

Teniendo en consideración el SCORE mayor a 0.98, el resultado obtenido fue el siguiente:

Predicción para el Mes=27 [97.1201887]

Ahora debemos verificar y validar con los datos ingresados en el mes 27 su demanda es de 97 y el resultado de la predicción de la demanda para el mes 27 es de [97.1201887] con un SCORE de 0.984709286263 en la Iteración 06.

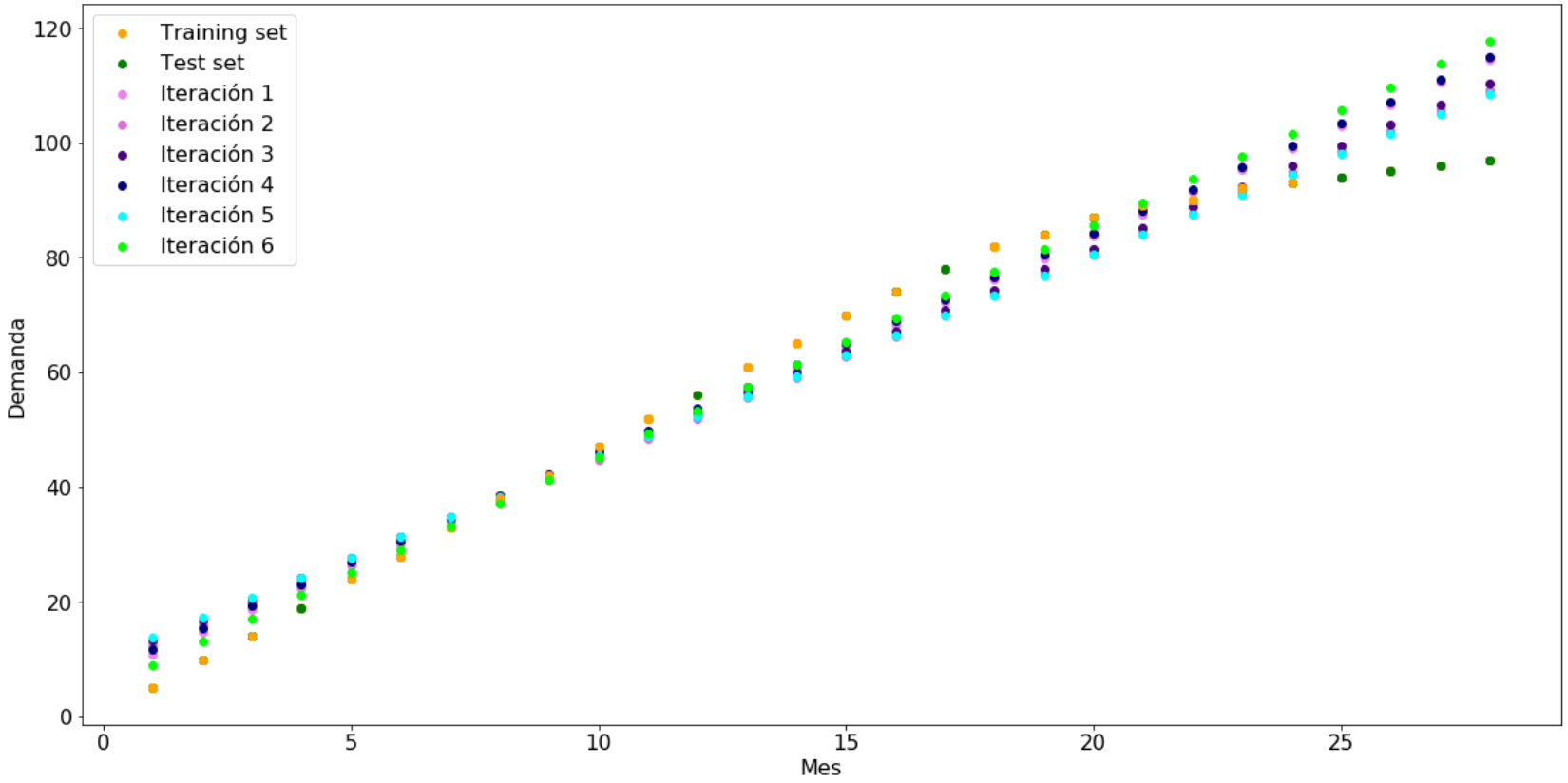

Grafico 10: Primera predicción de la demanda del mes 27.

Fuente: Elaboración propia

Volvamos a predecir el nuevamente la demanda del mes 27 y observemos los resultados

Aproximaciones del SCORE: nuevamente la principal condición a cumplir es que el SCORE de predicción sea mayor a 0.98, esto para precisar que el resultado obtenido en la predicción de la demanda del mes 27 se ajusta lo más posible al resultado real a partir de los datos de entrenamiento y prueba ingresados para la predicción:

:

mlr.score(X_train,y_train) > 0.98

A diferencia de la predicción anterior de la demanda del mes 27 esta se obtuvo en la Iteración 06 con una predicción de 97.1201887, realizaremos un nuevo proceso de predicción teniendo en consideración los mismos datos, hecho ello se obtuvo que la predicción se obtuvo en la Iteración 07 con un SCORE mayor a 0.98 como se muestra a continuación:

Iteración 01: 0.961871606966

Iteración 02: 0.967511750253

Iteración 03: 0.958063423853

Iteración 04: 0.956053144051

Iteración 05: 0.966899054325

Iteración 06: 0.95601526745

Iteración 07: 0.999380853742

El presente resultado calculado para la predicción del mes 27 de los datos ingresado fue efectuado mediante el siguiente comando:

"Predicción para el Mes=27 ",mlr.predict(27)

Para esta nueva predicción de la demanda del mes 27 es nuevamente considerado el SCORE mayor a 0.98, y de ello el resultado obtenido fue el siguiente:

Predicción para el Mes=27 [96.65001538]

Ahora se procede a verificar y validar con los datos ingresados en el mes 27 su demanda es de 97 y el resultado para esta nueva predicción de la demanda del mes 27 es de [96.65001538] con un SCORE de 0.999380853742 en la Iteración 07.

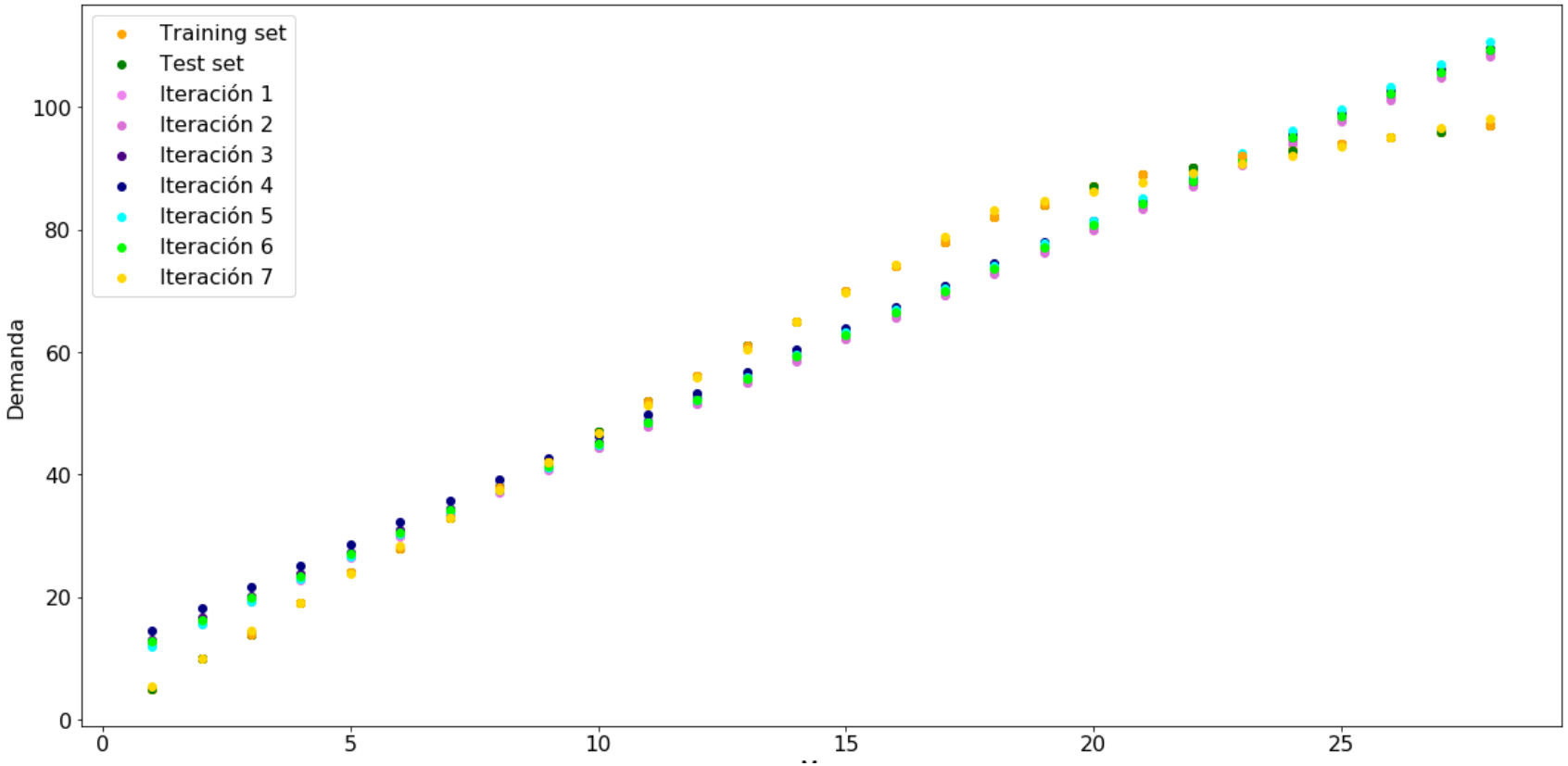

Grafico 11: Segunda predicción de la demanda del mes 27.

Fuente: Elaboración propia

Por lo mencionado en la segmentación de los datos para Entrenamiento y Test en 0.75 y 0.25 respectivamente y siendo estos aleatorios los resultados de las predicciones varían de la siguiente panera, para la primera prueba se obtiene una predicción para el Mes 27 se obtuvo una demanda de [96.65001538] en la Iteración 06 con un SCORE de 0.984709286263 y en la segunda prueba de predicción para el mismo Mes obtuvo una demanda de [96.65001538] en la iteración 07 con un SCORE de 0.999380853742 en ambos casos el SCORE supera al solicitado que es de 0.98, mientras el SCORE se aproxime más a la unidad (1) el grado de precisión de la predicción es mayor.

5.2.3 Predicción de la demanda para el mes 29

Para la predicción del mes 29, se debe tener en claro que este mes no se encuentra ingresado como datos de entrenamiento y control. Como se muestra a continuación:

Datos Mes - Demanda

0	5	10	52		
1	10	11	56	20	89
2	14	12	61	21	90
3	19	13	65	22	92
4	24	14	70	23	93
5	28	15	74	24	94
6	33	16	78	25	95
7	38	17	82	26	96
8	42	18	84	27	97
9	47	19	87		

Aproximaciones del SCORE: la condición principal que se debe de cumplir es que el SCORE de predicción sea mayor a 0.98, esto precisa que se ajusta la predicción de la demanda del mes 29 a lo más real posible a partir de los datos de entrenamiento y prueba ingresados para la predicción:

mlr.score(X_train,y_train) > 0.98

Obteniéndose como se muestra a continuación seis iteraciones en donde la Iteración 11 es mayor a 0.98 SCORE establecido para la predicción.

Iteración 01: 0.961553541847

Iteración 02: 0.960269160468

Iteración 03: 0.953376143781

Iteración 04: 0.965374511913

Iteración 05: 0.95884843422

Iteración 06: 0.963883082135

Iteración 07: 0.96798460815

Iteración 08: 0.962402810366

Iteración 09: 0.960851569348

Iteración 10: 0.961043146594

Iteración 11: 0.98221143858

El presente resultado calculado para la predicción de la demanda del mes 29 de los datos ingresado fue efectuado mediante el siguiente comando:

"Predicción para el Mes=29 ",mlr.predict(29)

Teniendo en consideración el SCORE mayor a 0.98, el resultado obtenido fue el siguiente:

Predicción para el Mes=29 [122.33246166]

De los datos ingresados para la predicción de la demanda del mes 29 se obtuvo un resultado de predicción de demanda de [122.33246166] con un SCORE de 0.98221143858 en la Iteración 11.

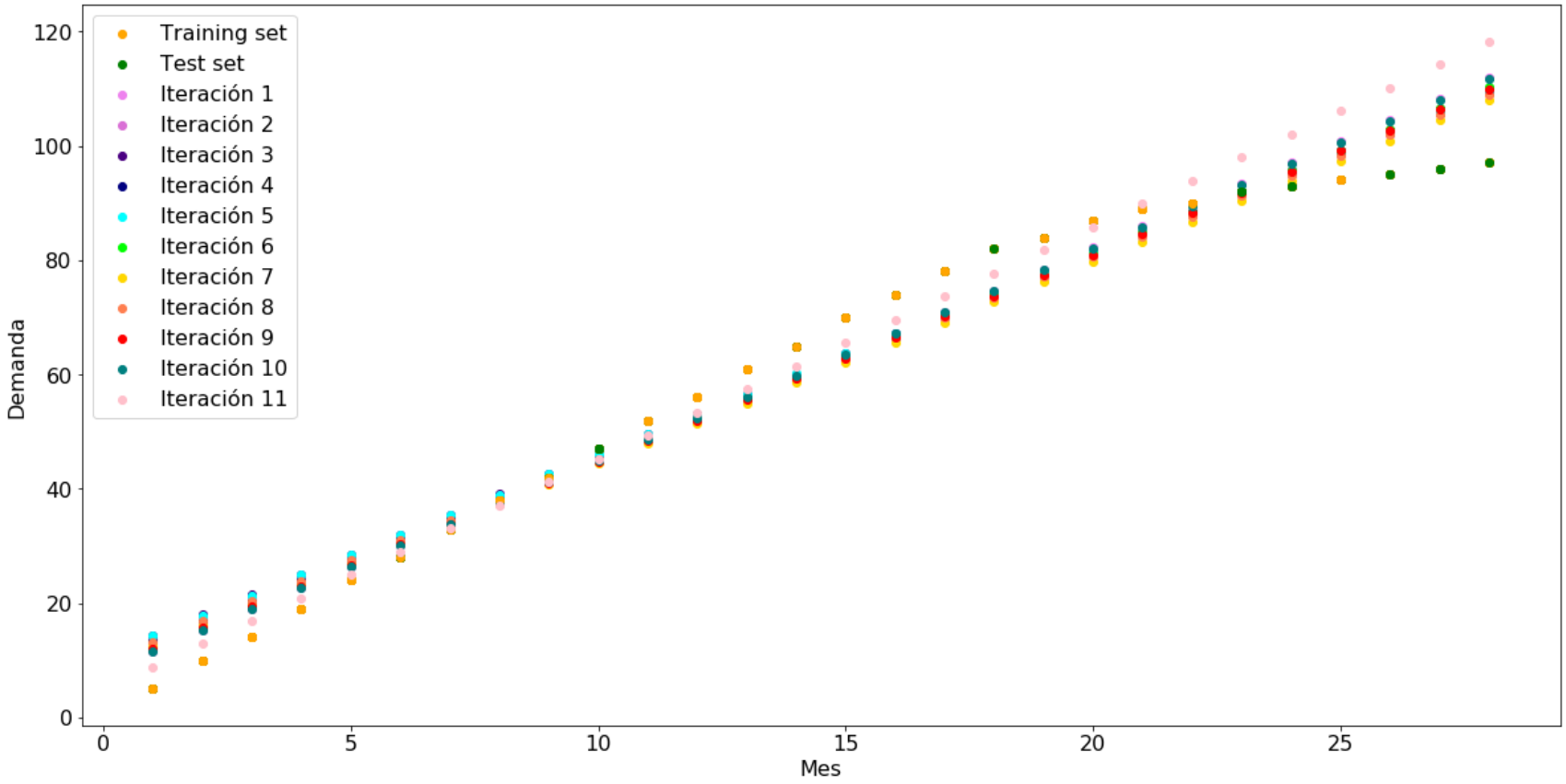

Grafico 12: Primera predicción de la demanda del mes 29.

Fuente: Elaboración propia

Volvamos a predecir el nuevamente la demanda del mes 29 y observemos los resultados

Aproximaciones del SCORE: nuevamente la principal condición a cumplir es que el SCORE de predicción sea mayor a 0.98, esto para precisar que el resultado obtenido en la predicción de la demanda del mes 29 se ajusta lo más posible al resultado real a partir de los datos de entrenamiento y prueba ingresados para la predicción:

:

mlr.score(X_train,y_train) > 0.98

A diferencia de la predicción anterior de la demanda del mes 29 esta se obtuvo en la Iteración 06 con una predicción de 97.1201887, realizaremos un nuevo proceso de predicción teniendo en consideración los mismos datos, hecho ello se obtuvo que la predicción se obtuvo en la Iteración 13 con un SCORE mayor a 0.98 como se muestra a continuación:

Iteración 01: 0.967217908052

Iteración 02: 0.956189134517

Iteración 03: 0.956253937424

Iteración 04: 0.966433442897

Iteración 05: 0.974380856525

Iteración 06: 0.96442876492

Iteración 07: 0.968902569489

Iteración 08: 0.964594123839

Iteración 09: 0.964360489521

Iteración 10: 0.97761064557

Iteración 11: 0.967364551853

Iteración 12: 0.954232342734

Iteración 13: 0.999466003519

El presente resultado calculado para la predicción del mes 29 de los datos ingresado fue efectuado mediante el siguiente comando:

"Predicción para el Mes=29 ",mlr.predict(29)

Para esta nueva predicción de la demanda del mes 29 es nuevamente considerado el SCORE mayor a 0.98, y de ello el resultado obtenido fue el siguiente:

Predicción para el Mes=29 [99.500008]

Ahora se procede a verificar y validar con los datos ingresados en el mes 27 su demanda es de 97 y el resultado para esta nueva predicción de la demanda del mes 29 es de [99.500008] con un SCORE de 0.999466003519 en la Iteración 13.

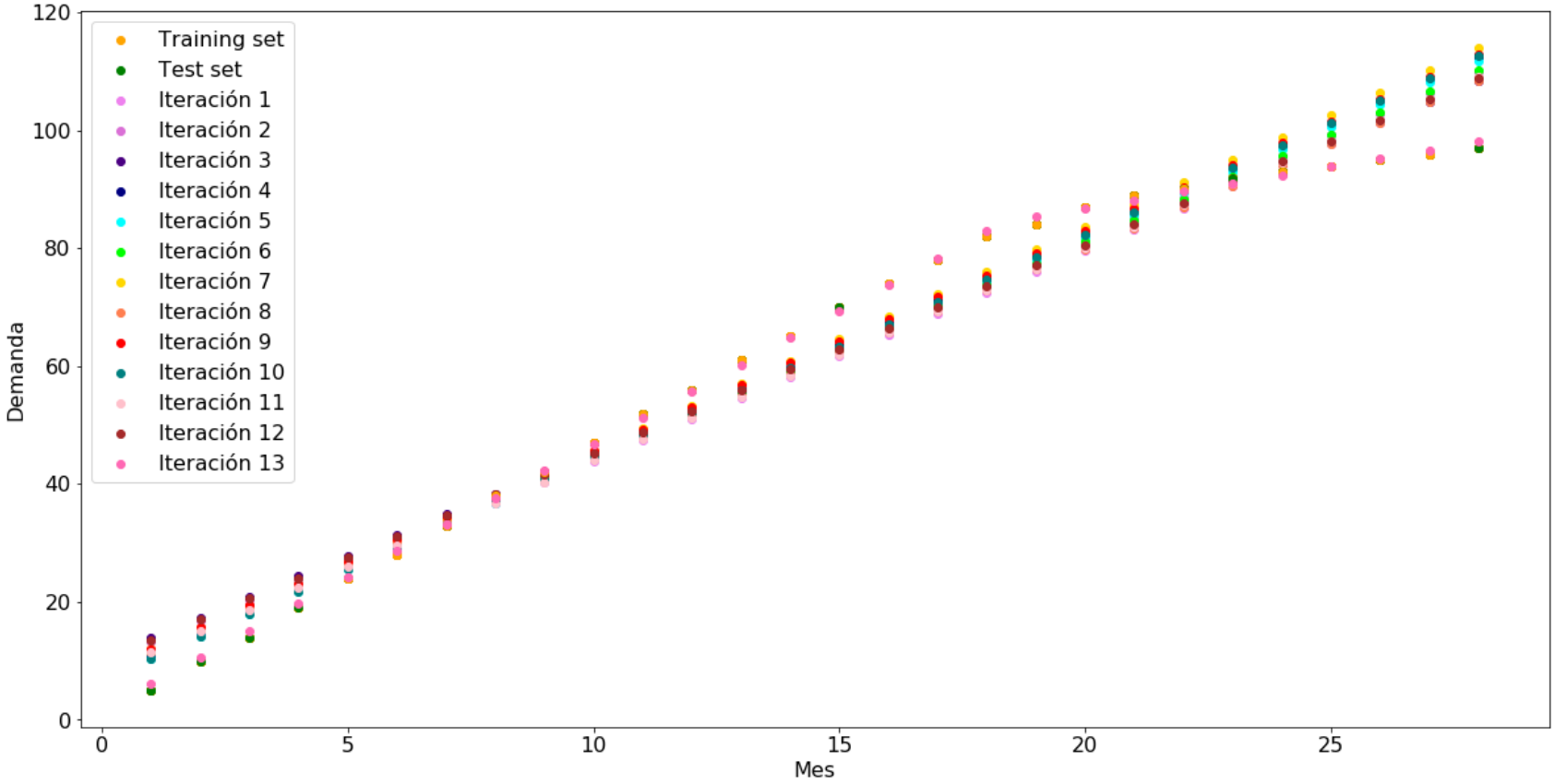

Grafico 13: Segunda predicción de la demanda del mes 29.

Fuente: Elaboración propia

De igual manera para la predicciones futura se considera la segmentación de los datos para Entrenamiento y Test en 0.75 y 0.25 respectivamente y siendo estos aleatorios los resultados de las predicciones varían de la siguiente panera, para la primera prueba se obtiene una predicción para el Mes 29 se obtuvo una demanda de [122.33246166] en la Iteración 11 con un SCORE de 0.98221143858 y en la segunda prueba de predicción para el mismo Mes obtuvo una demanda de [99.500008] en la iteración 13 con un SCORE de 0.999466003519 en ambos casos el SCORE supera al solicitado que es de 0.98, mientras el SCORE se aproxime más a la unidad (1) el grado de precisión de la predicción es mayor.

CAPITULO VI

6 DISCUSIÓN

En el transcurso del desarrollo de la investigación fue muy importante la recopilación de información, en búsqueda de una visión panorámica de los principios, desarrollo y aplicaciones de una red neuronal artificial enfatizando la capacidad para resolver problemas de predicción de series de tiempo.

Gómez (2010) concluye que las redes neuronales artificiales proporcionan buenas predicciones comparadas con las predicciones del modelo de serie de tiempo salvo que se tuvo que buscar la estructura idónea 1:2:16:12:1. Según el resultado de la presente tesis se obtuvo una estructura de hidden_layer_sizes=(3,3); se ratifica que la topología es distinta de acuerdo al tipo de problema. Sin embargo, se difiere el resultado, el número de neuronas de la capa oculta debe ser menor a las de la capa de entrada.

Jiménez (2013) hizo una comparación y evaluación con el método de descomposición, Winter y ARIMA de sus resultados determinando que las redes neuronales artificiales son un modelo que obtiene mejores resultados de predicción con un error cuadrático medio del 2%. Según el resultado de la presente tesis se obtuvo una confiabilidad (SCORE) de resultado superior a 98%; se acepta la determinación.

Finalmente, se obtuvo resultados con una validación establecida fue de 98%. Según el resultado de la presente tesis se obtuvo una validación se superó el mismo.

CONCLUSIONES

Al profundizar los conocimientos y al desarrollarlos, facilitó el diseño del modelo de aprendizaje para el funcionamiento de la red neuronal artificial, se emplearon conceptos aprendidos durante la carrera, como: inteligencia artificial, programación matemática, algebra lineal, simulación de sistemas, entre otros. Al tener un enfoque con sólida fundamentación matemática.

Se calculó los valores futuros a partir del modelo de predicción de demanda, fueron muy buenos, con el test de validación del 98% y el error cuadrático medio del 2%. Una de las características principales que se observó en el uso de la red neuronal artificial fue que funciona como una caja negra, es decir la interacción que se hizo con la red se centró en el inicio y en el final.

La investigación ha demostrado que se puede realizar la predicción de la demanda de libros de la biblioteca especializada de sistemas. utilizando redes neuronales artificiales, se pudo predecir los valores futuros. Los resultados de la investigación apoyan la utilización de las redes neuronales artificiales como técnica confiable en la predicción de series.

RECOMENDACIONES

Profundizar la investigación para definir el conjunto de datos razonables, dotar a la red neuronal artificial de suficientes parámetros para que sea capaz de aprender y evitar el sobreaprendizaje, que es el principal aspecto a tener en cuenta, perdiendo su habilidad de generalizar su aprendizaje a casos nuevos. Los modelos de tipo caja negra pueden producir buenos resultados durante el aprendizaje y validación. Sin embargo, si el fenómeno es complejo, la generalización podría ser decepcionante.

Potenciar el programa agregando nuevas líneas de código que puedan asignar nuevas variables de entrada como los indicadores de mes, temperatura, punto de congelación, salinidad, refrigerante, capacidad de la planta, entre otras. Para obtener una mayor precisión en la predicción de acuerdo a los cambios en el entorno.

Comparar la red neuronal multicapa, con otros modelos, como las redes de alto orden, las redes neuronales probabilísticas, las redes neuronales difusas, las redes neuronales basadas en Wavelets o redes funcionales. Sometiendo a pruebas de predicción de series de tiempo, así como su facilidad de construcción y diseño.

REFERENCIAS BIBLIOGRAFIA

Heaton, J. (2008). Introduction to Neural Networks for Java. St. Louis: Heaton Research, Inc.

Heaton, J. (2008). Introduction to Neural Networks with C#. U.S.: Heaton Research, Inc.

Heaton, J. (2010). Programming Neural Networks with Encog 2 in Java. St. Louis: Heaton Research, Inc.

Joseph Howse (2013). OpenCV Computer Vision with Python, PACKT PUBLISHING, open source comunity experience distilled.

Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, P. (2010). Metodología de la Investigación. México: McGRAW - HILL.

Hu, Y. H., & Hwang, J.-N. (2002). Handbook of Neural Network Signal Processing. U.S.: CRC Press LLC.

Isasi Viñuela, P., & Galván León, I. M. (2004). Redes de neuronas artificiales: Un enfoque práctico. España: Pearson.

ANEXO

ANEXO 01: Datos Utilizado de para la predicción

Universidad Nacional José María Arguedas

Identidad y Excelencia para el Trabajo Productivo y el Desarrollo

DIRECCIÓN DE INFORMACIÓN ACADEMICA Y BIBLIOTECA

INFORME Nº 002-2019-UNAJMA/DIAB/RBE-EPIS/RCVD

A : **Dra. Ing. Norma Lorena Catacora Flores**
Directora de Información Académica y Biblioteca

DE : ROBERTO CARLOS VEGA DÁVILA
Responsable Biblioteca Especializada EPIS

ASUNTO : Informe Solicitado

REFERENCIA : **Memorando Múltiple. N° 004-2019-DIAB/UNAJMA**

FECHA : 18 de Marzo de 2019

Es grato dirigirme a usted, para saludarla muy cordialmente y a la vez elevar la respuesta al informe solicitado según documento en referencia, los datos

El total de Alumnos que consultaron libros es de aproximadamente 43.
El total de Docentes que realizan consultas es de 04.
El total de libros prestados a los estudiantes, para sus domicilios es de: 475 movimientos.
El total de libros prestados a los docentes, para sus domicilios es de 11 movimientos.

Los libros que fueron adquiridos en detalle son:

Hasta el Año 2015 : 1612 Libros.
Libros Catalogados en Totalidad.
Faltan renovar algunas etiquetas y terminar reparación.

Año 2018 : 87 Libros para la biblioteca de EPIS
Estos Libros Aun no Fueron catalogados. Se Adjunta Lista.

Total 1699 Libros Adquiridos hasta antes del 18 de Febrero.

Los libros que adeudan los docentes a la biblioteca de EPAE y que no han sido devueltos hasta la fecha son 1.
La totalidad de libros que adeudan los alumnos y no han sido devueltos hasta la fecha son 57.

Sin otro en particular me despido de usted no sin antes dejarle las muestras de mi estima personal, es todo cuanto informo a usted para los fines que estime por conveniente.

Atentamente;

UNIVERSIDAD NACIONAL JOSÉ MARÍA ARGUEDAS

Roberto Carlos Vega Dávila

ROBERTO CARLOS VEGA DAVILA
Responsable de Biblioteca EPIS

Recibido
26-03-2019
Hora: 9:10 a.

C. Archivo.
CC.

LIBROS ADQUIRIDOS 2018-I EPIS

N° ITEM	CANT.	TITULO	EDITORIAL
02	01	MATEMATICAS DISCRETAS. APLICACIONES Y EJERCICIOS	PATRIA
03	01	INTRODUCCION AL CALCULO.	UNIVERSITAT POLITECNICA DE VALENCIA
04	01	CALCULO DIFERENCIAL. PROBLEMAS RESUELTOS.	REVERTE
10	01	CALCULO INTEGRAL. TECNICAS DE INTEGRACION	EDICIONES DE LA U
11	01	CALCULO DE VARIAS VARIABLES.	PATRIA
13	01	METODO DE LAS 6'D. MODELAMIENTO-ALGORITMO-PROGRAMACION	UNIV. SAN MARTIN DE PORRES
24	01	ESTADISTICA INFERENCIAL 1. PARA INGENIERIA Y CIENCIAS	PATRIA
71	01	METODOLOGIA DE LA INVESTIGACION	UNIV. SAN IGNACIO DE LOYOLA (USIL)
73	01	METODOLOGIA DE LA INVESTIGACION. EL PROCESO Y SUS TECNICAS.	LIMUSA
159	01	EXCEL FOR MASTER 2016	MACRO
160	01	DESARROLLO DE APLICACIONES CON VISUAL BASIC 2015	MACRO
161	01	DESARROLLO DE APLICACIONES MEDIANTE EL FRAMEWORK DE SPRING	EDICIONES DE LA U
162	01	DESARROLLO DE APLICACIONES MOVILES CON ANDROID	MACRO
163	01	DESARROLLO DE APLICACIONES WEB CON PHP Y MYSQL.	MACRO
164	01	APRENDER RASPBERRY PI. CON 100 EJERCICIOS	MARCOMBO
165	01	PYTHON FACIL	MARCOMBO
166	01	LINUX. PRINCIPIOS BASICOS DE USO DEL SISTEMA	EDICIONES ENI
167	01	ADMINISTRACION HARDWARE DE UN SISTEMA INFORMATICO.	MACRO
168	01	ADMINISTRACION SOFTWARE DE UN SISTEMA INFORMATICO.	MACRO
169	01	CCNA SECURITY 210-260. OFFICIAL CERT GUIDE	CISCO PRESS
170	01	REDES OPTICAS DE ACCESO CONVERGENTE	UNIV. DISTRITAL FRANCISCO JOSE DE CALDAS
171	01	HACKING CON INGENIERIA SOCIAL TECNICAS PARA HACKEAR HUMANOS	EDICIONES DE LA U
172	01	PHP - CREACION DE PAGINAS WEB DINAMICAS 2A ED.	ALFAOMEGA
173	01	PROGRAMACION TEORIA Y APLICACIONES	UNIV. DE MEDELLIN
174	01	PROGRAMACION PARA INGENIERIA Y CIENCIAS CON MATLAB Y OCTAVE	BELLISCO
175	01	DISEÑO DE INTERFACES.	PARRAMON
176	01	OPTIMIZACION ALGORITMOS PROGRAMADOS CON MATLAB	ALFAOMEGA
177	01	ANALISIS Y DISEÑO ALGORITIMOS CON IMPLEMENTACION EN C++, JACA + PHP.	AUTORES NACIONALES
180	01	PRINCIPIOS DE ECONOMIA	UNIV. SAN IGNACIO DE LOYOLA (USIL)
181	01	MATEMATICA DISCRETA	CEF. (CENTRO DE ESTUDIOS FINANCIEROS)
182	01	MATEMATICA DISCRETA.	CEF. (CENTRO DE ESTUDIOS FINANCIEROS)
183	01	ESTRUCTURA DE DATOS CON C++.	UNIV. PRIVADA DEL NORTE
184	01	FUNDAMENTOS DE PROGRAMACION.ALGORITMOS, ESTRUCTURA DE DATOS Y OBJETOS.	MCGRAW-HILL
185	01	METODOS NUMERICOS APLICADOS A LA INGENIERIA	ALFAOMEGA
186	01	METODOS NUMERICOS PARA INGENIEROS	MCGRAW-HILL
190	01	ADMINISTRACION BASICA DE BASES DE DATOS CON ORACLE 12C SQL PRACTICAS Y EJERCICIO	ALFAOMEGA
191	01	BIG DATA ANALYTICS CON HERRAMIENTAS DE SAS, IBM, ORACLE Y MICROSOFT	GARCETA GRUPO EDITORIAL
192	01	INVESTIGACION OPERATIVA	MACRO
193	01	INVESTIGACION DE OPERACIONES PROGRAMACION LINEAL.	MACRO
199	01	SISTEMAS OPERATIVOS. PANORAMA PARA INGENIERIA EN COMPUTACION E INFORMATICA	PATRIA
200	01	LOGICA MATEMATICA E INTELIGENCIA ARTIFICIAL.	DYKINSON
201	01	PROYECTOS FORMULACION Y EVALUACION.	MACRO
202	01	FORMULACION Y EVALUACION DE PROYECTOS DE INVERSION.	MACRO
205	01	SISTEMAS OPERATIVOS	ALFAOMEGA
209	01	JAVA A FONDO CURSO DE PROGRAMACION	ALFAOMEGA
210	01	JAVA 8 Y ANDROID STUDIO.	AUTORES NACIONALES

211	01	APRENDER PROGRAMAR ANDROID CON 100 EJERCICIOS PRACTICOS	MARCOMBO
213	01	DATA MINING. MINERIA DE DATOS	MACRO
214	01	BIG DATA Y PERIODISMO EN LA SOCIEDAD RED	SINTESIS
216	01	PROGRAMACION ORIENTADA A OBJETOS MF0227_3	EDICIONES DE LA U
217	01	ESTRUCTURAS DE DATOS BASICAS	ALFAOMEGA
218	01	METODOLOGIA DE LA PROGRAMACION	CEF. (CENTRO DE ESTUDIOS FINANCIEROS)
219	01	VBA EXCEL 2016. PACK DE 2 LIBROS: DOMINE LA PROGRAMACION EN EXCEL: TEORIA, EJER	EDICIONES ENI
220	01	LENGUAJES DE PROGRAMACION Y PROCESADORES	RAMON ARECES
222	01	MANEJO DE TECNICAS DE PROGRAMACION	PEARSON
223	01	LENGUAJE DE PROGRAMACION CON JAVA.	MACRO
230	01	MANUAL IMPRESCINDIBLE DESARROLLO DE APLICACIONES PARA ANDROID	ANAYA MULTIMEDIA
231	01	MANUAL IMPRESCINDIBLE MICROSOFT ACCESS 2016	ANAYA MULTIMEDIA
232	01	MANUAL IMPRENSCINDIBLE MICROSOFT WORD 2016	ANAYA MULTIMEDIA
233	01	MICROSOFT EXCEL 2016. DOMINE LAS FUNCIONES AVANZADAS DE LA HOJA DE CALCULO	EDICIONES ENI
234	01	APLICACION PRACTICA DE LAS TABLAS DINAMICAS CON EXCEL	PROFIT EDITORIAL
237	01	SISTEMAS PROGRAMABLES AVANZADOS	MARCOMBO
238	01	POSICIONAMIENTO WEB ESTRATEGIAS DE SEO. GOOGLE Y OTROS BUSCADORES	EDICIONES ENI
239	01	ASP.NET C# - PACK DE 2 LIBROS: APRENDER EL LENGUAJE C# Y EL DESARROLLO ASP.NET	EDICIONES ENI
240	01	MICROSOFT POWERPOINT 2016. FUNCIONES BASICAS	EDICIONES ENI
241	01	LENGUAJE ENSAMBLADOR.	MACRO
246	01	CCNA VOICE 640-461 OFFICIAL CERT GUIDE.	CISCO PRESS
247	01	EL LIBRO OFICIAL ILLUSTRATOR CS5	ANAYA MULTIMEDIA
248	01	MODELADO DIGITAL.	ANAYA MULTIMEDIA
250	01	CLOUD COMPUTING, TECNOLOGIA Y NEGOCIO.	PARANINFO
249	01	REDES Y SEGURIDAD	ALFAOMEGA
252	01	SEGURIDAD INFORMATICA Y MALWARES. ANALISIS DE AMENAZAS E IMPLEMENTACION DE CONT	EDICIONES ENI
253	01	PROGRAMACION EN C, C++, JAVA Y UML.	MCGRAW-HILL
254	01	ANDROID. GUIA DE DESARROLLO DE APLICACIONES PARA SMARTPHONES Y TABLETAS	EDICIONES ENI
255	01	INTRODUCCION A LA PROGRAMACION.	PATRIA
256	01	SISTEMAS OPERATIVOS MONOPUESTO	MACMILLAN
257	01	MATEMATICA DISCRETA	CEF. (CENTRO DE ESTUDIOS FINANCIEROS)
258	01	LAN INALAMBRICA Y CONMUTADA. GUIA DE ESTUDIO DE CCNA EXPLORATION.	PEARSON
260	01	CALCULUS II. CALCULO CON FUNCIONES DE VARIAS VARIABLES Y ALGEBRA LINEAL, CON APL	REVERTE
261	01	OPTIMIZACION ALGORITMOS PROGRAMADOS CON MATLAB	ALFAOMEGA
262	01	PROGRAMACION GRAFICA PARA INGENIEROS.	ALFAOMEGA
263	01	PROJECT 2016. CURSO PRACTICO PASO A PASO	ALFAOMEGA
264	01	EL MUNDO GENUINO - ARDUINO. CURSO PRACTICO DE FORMACION	ALFAOMEGA
265	01	CREACION DE JUEGOS Y APLICACIONES PARA ANDROID	EDICIONES DE LA U
270	01	AUDITORIA DE SEGURIDAD INFORMATICA.	EDICIONES DE LA U
272	01	GESTION DE BASES DE DATOS.	EDICIONES DE LA U
273	01	DIDACTICA DE LA ROBOTICA EDUCATIVA. UN ENFOQUE CONSTRUCTIVISTA	DEXTRA EDITORIAL

87

VERIFICAR LA LISTA SEGÚN EL NUMERO DE ITEM , ES COINCIDENTE CON EL NUMERO DE LA PECOSA

ANEXO 02: Selección de Datos de Entrenamiento y Datos de control

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


datos=pd.read_csv("bater001.csv")
x= datos['Mes']
y= datos['Demanda']

print (x)
print (y)

X=x[:,np.newaxis]
i=0
plt.figure(figsize=(20,10))
plt.rc('font',size=16)

colors =['dark
blue','violet','orchid','indigo','navy','cyan','lime','gold','coral','red','teal','pink','brown','hotpin
k','orchid','aqua','green','blue','yellow','purple','black','tomato','salmon']

   #aqui se hace la particion para los datos de entrenamiento y de prueba
   from sklearn.model_selection import train_test_split
   X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.50)

   mlr=MLPRegressor(solver='lbfgs', alpha=1e-
5,hidden_layer_sizes=(3,3),random_state=1)
   mlr.fit(X_train,y_train)
   print (mlr.score(X_train,y_train))
   plt.scatter(X_train,y_train,color='orange', label="Training set"if i==1 else "")
   plt.scatter(X_test,y_test,color='green', label="Test set"if i==1 else "")
   if mlr.score(X_train,y_train) > 0.98:

plt.xlabel('Mes')
plt.ylabel('Demanda')
plt.legend(loc='lower rigth')
plt.show()
print ("Prediccion en Mes=27 ",mlr.predict(29))
```

ANEXO 02: Código de Estandarización y Normalización

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

datos=pd.read_csv("bater1.csv")
datos=datos.replace(np.nan,"0")
df=pd.DataFrame(datos)

print("Media")
print(df['Demanda'].mean())
print("Desviacion Estandar")
print(df['Demanda'].std())
print("Valor Maximo")
print(df['Demanda'].max())
print("Valor Minimo")
print(df['Demanda'].min())

fig,(ax1, ax2)=plt.subplots(ncols=2, figsize=(20,8))
ax1.set_title('Antes de Escalar')
sns.kdeplot(df['Demanda'],ax=ax1)
sns.kdeplot(df['Mes'],ax=ax1)
#con esta Funcion Standarizamos los datos para el entrenamiento
#scaler=preprocessing.StandardScaler()
#con esta Funcion normalizar los datos para el entrenamiento
scaler=preprocessing.Normalizer(norm='l2',copy=True)

df[['Mes','Demanda']]=scaler.fit_transform(df[['Mes','Demanda']])

print("*"*10)
print(df['Demanda'].iloc[0])
print("Media")
print(df['Demanda'].mean())
print("Desviacion Estandar")
print(df['Demanda'].std())
print("Valor Maximo")
print(df['Demanda'].max())
print("Valor Minimo")
print(df['Demanda'].min())

df.to_csv('bater2.csv',sep='\t')
ax2.set_title('Despues de Escalar')
sns.kdeplot(df['Demanda'],ax=ax2)
```

```
sns.kdeplot(df['Mes'],ax=ax2)

plt.show()
```

ANEXO 03: Código de predicción de con redes neuronales

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


datos=pd.read_csv("bater.csv")
x= datos['Minutos']
y= datos['carga']

print (x)
print (y)

X=x[:,np.newaxis]
i=0
plt.figure(figsize=(20,10))
plt.rc('font',size=16)

colors =['dark
blue','violet','orchid','indigo','navy','cyan','lime','gold','coral','red','teal','pink','brown','hotpin
k','orchid','aqua','green','blue','yellow','purple','black','tomato','salmon']

while True:
   i=i+1
   #aqui se hace la particion para los datos de entrenamiento y de prueba
   from sklearn.model_selection import train_test_split
   X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.50)

   mlr=MLPRegressor(solver='lbfgs', alpha=1e-
5,hidden_layer_sizes=(3,3),random_state=1)
   mlr.fit(X_train,y_train)
   print (mlr.score(X_train,y_train))
   plt.scatter(X_train,y_train,color='orange', label="Training set"if i==1 else "")
   plt.scatter(X_test,y_test,color='green', label="Test set"if i==1 else "")
   plt.scatter(X,mlr.predict(X),color=colors[i], label="Iteración "+str(i))
   if mlr.score(X_train,y_train) > 0.98:
      break
plt.xlabel('Mes')
plt.ylabel('Demanda')
plt.legend(loc='lower rigth')
plt.show()
print ("Predicción en Mes=29 ",mlr.predict(28))
```

ANEXO 04: klearn.linear_modelRegresión lineal

class sklearn.linear_model.**LinearRegression**(*fit_intercept = True* , *normalize = False* , *copy_X = True* , *n_jobs = None*)

Regresión lineal de mínimos cuadrados ordinarios.

Parámetros:

fit_intercept : ***booleano, opcional, predeterminado True***

Si calcular el intercepto para este modelo. Si se establece en Falso, no se utilizará ninguna intercepción en los cálculos (por ejemplo, se espera que los datos ya estén centrados).

normalizar : ***booleano, opcional, predeterminado Falso***

Este parámetro se ignora cuando fit_interceptse establece en False. Si es verdadero, los regresores X se normalizarán antes de la regresión restando la media y dividiendo por la norma l2. Si desea estandarizar,
utilice **sklearn.preprocessing.StandardScaler**antes de llamar fita un estimador con normalize=False.

copy_X : ***booleano, opcional, predeterminado True***

Si es True, se copiará X; de lo contrario, puede ser sobrescrito.

n_jobs : ***int o None, opcional (por defecto = None)***

El número de trabajos a utilizar para el cálculo. Esto solo proporcionará aceleración para n_targets> 1 y suficientes problemas grandes. Nonesignifica 1 a menos que en un **joblib.parallel_backend**contexto. -1Significa usar todos los procesadores. Ver Glosario para más detalles.

Atributos:

coef_ : ***array, shape (n_features) o (n_targets, n_features)***

Coeficientes estimados para el problema de regresión lineal. Si se pasan varios objetivos durante el ajuste (y 2D), esta es una matriz 2D de forma (n_targets, n_features), mientras que si solo se pasa un objetivo, esta es una matriz 1D de longitud n_features.

intercept_ : *array*

Término independiente en el modelo lineal.