Universidad Nacional Experimental del Táchira

Vicerectorado Académico

Decanato de Docencia

Departamento de Ingeniería en Informática

Trabajo de aplicación profesional

Proyecto especial de grado

Paquete en lenguaje R para la clasificación de tubérculos de papa criolla(*Solanum phureja*) para diferentes densidades de siembra empleando redes neuronales probabilísticas.

Autor(es): Jesús David Escalante Rodríguez

C.I.: 21.220.841

jesusd.escalante@unet.edu.ve

Tutor(es): Rossana Timaure

rttg@unet.edu.ve

San Cristóbal, Julio.2019

Índice general

Introducción

En los cultivos de papa criolla (*Solanum phureja*) son muchas las variables que influyen en la cosecha de los tubérculos, tales como la temperatura, la altura del terreno, el clima, el tipo y la composición del suelo, la densidad de siembra, entre otros. Los encargados y personas interesadas en este proceso siempre han buscado manipularlas, no sólo para mejorar la calidad del producto, si no para obtener características que sean convenientes, como el tamaño de los tubérculos, ya que existen intereses industriales diferentes para su comercialización.

Investigaciones anteriores concluyeron que la densidad de siembra es una variable con una afectación considerable en el tamaño de los tubérculos producidos por las plantas en su ciclo de cosecha, sin embargo la manipulación de esta variable generalmente se maneja de forma lineal sin tener en cuenta ciertas condiciones espaciales que posee como propiedades, por ser distancias entre las calles y los surcos, y la distancia entre plantas a la que se siembra, o expresado de otra manera cuantas plantas son sembradas por metro cuadrado en un terreno.

La densidad de siembra es una variable cualitativa, que aunque se trata de distancias, se definen en densidades predeterminadas como d_1 = 50x30cm, donde d_1 es una densidad definida igual a un espacio entre surcos de 50cm y distancia entre plantas de 30cm.

Por otra parte, las redes neuronales clasifican basado en sus neuronas entrenadas, a través de datos conocidos con anterioridad, y no requiere de supuestos estadísticos a tener en cuenta para su proceso de clasificación.

El uso de redes neuronales probabilísticas es una solución que se planteó para el modelado de los datos recogidos y observados de la cosecha de un cultivo de tubérculos de papa criolla (*Solanum phureja*) realizado en el Centro Agropecuario Marengo de la Universidad Nacional de Colombia, en el departamento de Cundinamarca en Colombia (74°12'58.51"W;4°40'52.92"N), el cual tiene una

altitud de 2516 msnm y una temperatura media de 14 °C. Implementado un paquete en Lenguaje R que permite a través de una entrada, propuesta como una matriz con los datos observados de cultivo, clasificar los tubérculos de papa criolla cosechados en densidades de siembra según el peso total del cultivo, peso total de tubérculos y diamétro de cada tubérculo.

Este trabajo de grado está dividido en los siguientes capítulos:

Capítulo 1. Preliminares.

- Planteamiento del problema; en él se plantean las razones en las cuales esta basado el proyecto de investigación.
- Objetivos, se expone el objetivo general del proyecto de investigación y se identifican los objetivos específicos mediante los cuales se lleva a cabo el desarrollo del proyecto.
- Justificación e importancia, se establecen los motivos por cuales el desarrollo del proyecto de investigación es relevante
- Alcance y limitaciones, se detalla la funcionalidad del proyecto de investigación.

Capítulo 2. Fundamentos teóricos

- Antecedentes, se exponen revisiones de investigaciones previas que posean relación directa con el tema del actual proyecto de investigación.
- Bases teóricas, se definen un conjunto de conceptos que proporcionan conocimiento acerca del objeto de estudio.

Capítulo 3. Fundamentos metodológicos

- Enfoque de la investigación, se plantea el enfoque utilizado para llevar a cabo la investigación.
- Tipo o nivel de investigación, se expone el tipo de investigación y nivel de investigación, según el grado de profundidad que abarca y la naturaleza de los datos.
- Diseño de la investigación, se establece la estrategia general que adopta el investigador para responder al problema planteado.
- Metodología, se describen el conjunto de técnicas, herramientas y pautas utilizadas para llevar a cabo el proyecto de investigación.

Capítulo 4. Desarrollo

- Creación del esqueleto del paquete, se detallan la creación de la estructura del paquete y su documentación.
- Diseño de las soluciones algorítmicas, se plantean los algoritmos que permiten dar solución a la clasificación a través de redes neuronales probabilísticas.
- Codificación de los algoritmos, se da a conocer la manera en que se desarrollaron en lenguaje R los algoritmos planteados para las soluciones de clasificación.
- Chequeo del paquete, método de distribución y registro del método de envío, se explica cómo se distribuye el paquete a repositorios públicos, para permitir su uso.
- Pruebas funcionales del paquete, se observa la funcionalidad del proyecto de investigación, mediante la exposición de tablas comparativas

Capítulo 5. Conclusiones

- Se da a conocer la evaluación e interpretación final de la investigación.

Capítulo 1

Preliminares

1.1. Planteamiento y formulación del problema

La producción de papa criolla (*Solanum phureja*), siempre ha tenido muchos desafíos por parte de la industria, debido a los grandes intereses provenientes de los diferentes consumidores que la misma posee, siendo así que es uno de los productos agrícolas mas consumidos y con mayor importancia en el mundo, después del arroz, el maíz y el trigo. Colombia es el mayor productor de papa criolla en Latinoamerica, por lo cual es un país que posee grandes exigencias industriales para su producción (Ligarreto *et al.*, 2003).

Los tubérculos de papa criolla tienen un tamaño aproximado entre dos y ocho centímetros (2-8 cm), y se pueden clasificar en tres calibres según su diámetro transversal promedio, de donde radican los intereses comerciales de la misma. Los tubérculos de entre dos y medio y cuatro centímetros son preferidos para encurtidos y pre-cocidos, mientras los tubérculos promedio entre cuatro y seis y medio centímetros son preferidos para frituras en hojuelas, o con mas de cinco centímetros para frituras en tiras (CORPOICA, 2009).

Por otra parte, el crecimiento vegetal es definido como el aumento irreversible del tamaño y peso seco de las plantas (altura, área foliar, diámetro, número de células y cantidad de protoplasma) o los cambios que ocurren en una planta o población de plantas a través del tiempo, fenómeno acompañado del aumento en la complejidad estructural metabólica del organismo (diferenciación celular, número de hojas), por procesos de división y alargamiento celular, incorporación de materia y energía del ambiente (fotosíntesis, absorción de agua y de iones) y metabolización subsiguiente (Cabezas, 2005), la cual se traduce en multiplicación y diferenciación celular. Este proceso está ínti-

mamente relacionado con algunos factores internos como fotosíntesis, respiración, transpiración, condiciones de estrés, concentración enzimática, balance hormonal y expresión genética (Piñeros, 2009).

En la papa son dos los procesos fisiológicos asociados directamente al rendimiento de la misma, la fotosíntesis y la respiración. Estos dos son procesos íntimamente asociados, ya que durante la fotosíntesis se producen carbohidratos que son consumidos durante la respiración, pero una gran cantidad de estos carbohidratos contribuyen al proceso de producción incrementando el tamaño de los tubérculos, el nivel y periodo de crecimiento de los tubérculos es una variable que responde directamente a la expresión de rendimiento expresada como producción por día, siendo así que cerca de un noventa por ciento (90 %) del peso acumulado de los tubérculos producidos por una planta es producto de la asimilación de dióxido de carbono, así como otros procesos similares (Beukema, 1979). Sin embargo, estos procesos se ven afectados por condiciones externas tales como la intensidad de luz, temperatura, longitud de los días, condiciones del suelo que son variables no controladas, o por condiciones como la cantidad de riego, fertilizantes aplicados para la mejora de los suelos que son variables controlables pero que requieren de costos adicionales de producción (Piñeros, 2009). Es por ello que se sigue buscando, con el manejo de variables controladas, el control sobre la cantidad total producida y el tamaño de los tubérculos, sin producir costos adicionales, siendo la densidad de siembra una de estas variables, en la cual se enfoca esta investigación (Bernal, 2017).

Sin embargo, la densidad de siembra es una variable de tipo espacial, ya que se define como la distancia entre plantas y entre linderas, siendo una coordenada en X - Y en un plano cartesiano, por lo cual un estudio estadístico de la misma para clasificar o predecir un modelo no puede ser del tipo de regresión lineal o de un análisis de varianza, debido a que los supuestos de estos modelos establecen que los residuales deben ser independientes en cada punto, no tomando en cuenta una relación entre vecinos, como si ocurre en los cultivos de papa, las plantas compiten por los recursos y por consiguiente por producir más, haciendo así los residuales dependientes (Piñeros, 2009).

La identificación siempre ha sido un problema de interés en el área agrícola, existen trabajos sobre la identificación de variedades de papas, sobre la capacidad de las plantas de papas para cobertura del suelo, entre otras variables de estudio sometidas a clasificación, esto debido al impacto que tiene la comercialización y manejo agrícola del producto, entre las técnicas empleadas se encuentran modelos estadísticos basados en probabilidad, las máquinas de soporte vectorial y las redes neuronales, estas últimas técnicas presentan la ventaja de que bajo el proceso de aprendizaje supervisado no presentan supuestos rígidos en cuanto al comportamiento y relación de los datos

de entrada (Pal, 2003; Aziz, 2016).

En otro contexto, una red neuronal artificial (*artificial neural network*, ANN) es un modelo que crea una relación entre una configuración de señales de entrada y una señal de salida usando un modelo derivado de nuestro entendimiento de como el cerebro responde a determinados estímulos de entradas sensoriales. Así como un cerebro usa ese conjunto de células interconectadas llamadas neuronas, así las ANN usan una red de neuronas artificiales o nodos para solucionar problemas aprendidos (Lantz, 2015).

En general, las ANN son aprendices versátiles que pueden ser aplicadas a casi cualquier tarea de aprendizaje en cualquier contexto, clasificación, predicción, reconocimiento de patrones e incluso de tareas de supervisión de procesos. Las ANN, se aplican mejor a los problemas cuyos datos de entrada y salida son bien definidos o bastante simples, sin embargo, el proceso que relaciona la entrada con una determinada salida es en extremo complejo, por eso se conocen como un método de caja negra (Lantz, 2015).

Las ANN han sido usadas desde hace cincuenta años para simular como el cerebro humano es capaz de resolver problemas. Primero surgieron las simples funciones lógicas AND y OR, que ayudaron a los científicos a entender como biológicamente el cerebro opera. Sin embargo, como el avance tecnológico y computacional ha sido incrementado y continua creciendo potencialmente durante los últimos años, las ANN han incrementado su complejidad siendo usadas frecuentemente en múltiples problemas como:

1. Programas de reconocimiento de voz y escritura, como los que usan los correo de voz de los servicios de transcripción y máquinas clasificadoras de correo postal.

2. La automatización de dispositivos inteligentes como los controles ambientales de un edificio de oficinas o automóviles autodirigidos y drones autoguiados.

3. Modelos sofisticados de patrones climáticos y otros fenómenos científicos, sociales o económicos.

4. La clasificación y reconocimiento de cualquier tipo de frutos y hojas de plantas, basado en sus atributos y características.

Las redes neuronales probabilísticas (*Probabilistic neural network*, PNN) se derivan de las ANN y se diferencian en que estas hacen uso de redes de funciones de base radial. Son adoptadas en

muchos casos por su fácil entrenamiento y velocidad de análisis. Las PNN tienen tres capas, una primera de entrada, una de base radial y otra de competición. Básicamente la capa de base radial evalúa las distancias entre los valores de un vector de entrada y estas distancias son escaladas no linealmente, para que la capa de competitividad encuentre las distancia mas corta, a través de la cual realiza la clasificación.

En otro orden de ideas, R es un lenguaje de programación y entornos para gráficos y cálculos estadísticos, es una implementación diferente al lenguaje S (Chambers, 1998), y se desarrolla bajo un proyecto de *software* bajo Licencia Pública General (*General Public License*, GNU) de la *Free Software Foundation* (`https://www.fsf.org/`) en forma de codigo fuente. Se compila y se ejecuta en una amplia variedad de plataformas entre ellas *FreeBSD*,*Linux*, *Windows* y*MacOS*. (The R Project for Statistical Computing, 2018).

Esta investigación tuvo como objetivo desarrollar un paquete en lenguaje R que permita ajustar y entrenar una red neuronal artificial probabilística para la clasificación de densidades de siembra de papa criolla (*Solanum phureja*) teniendo como entrada las siguientes variables:

1. **Peso total del cultivo.** Variable cuantitativa expresada en gramos (g/cosecha), que indica el peso de todos los tubérculos recogidos en la siembra.

2. **Total de tubérculos recogidos.** Variable cuantitativa que indica el número/cosecha de tubérculos recogidos en la siembra.

3. **Diámetro medio ponderado de todos los tubérculos.** Variable cuantitativa expresada en centímetros (cm.), que indica el cálculo del diámetro ponderado (por los tubérculos ser ovoides) de cada tubérculo recogido en la siembra.

Esta herramienta permité, entre otras cosas, a investigadores del área agrícola y a bioestadísticos comparar los resultados de clasificación alimentando la red con datos observados y con los mismos datos normalizados, para observar si es necesario el tratamiento estadístico de los datos recogidos antes de la clasificación y a su vez concluir gracias a los gráficos presentados y correlacion entre los datos hasta que punto la densidad de siembra afecta las variables antes mencionadas.

1.2. Objetivos

1.2.1. Objetivo general

Desarrollar un paquete en lenguaje R que permita la clasificación de tubérculos de papa criolla (*Solanum phureja*) para diferentes densidades de siembra empleando redes neuronales probabilísticas.

1.2.2. Objetivos específicos

- Estudiar la parametrización de la densidad de siembra como parámetro de entrada al clasificador.
- Diseñar los algoritmos para las funciones principales que permitan el entrenamiento de una red neuronal probabilística para la clasificación de tubérculos de papa criolla (*Solanum phureja*) para diferentes densidades de siembra.
- Implementar las funciones para la clasificación de tubérculos de papa criolla para diferentes densidades de siembra, basadas en los algorítmos desarrollados.
- Realizar la pruebas del paquete desarrollado bajo diferente escenarios.

1.3. Aportes de la investigación

Colombia es el mayor productor, consumidor y exportador de papas diploides en el mundo; tiene una ventaja competitiva notable debido a ser centro de diversidad y poseer gran aceptación por los consumidores debido a las características del tubérculo. Adicionalmente, en este país se ha desarrollado una amplia tradición como cultivo tecnificado, con potencial de industrialización y exportación (Rodríguez *et al*, 2009). Ademas de contar con uno de los recursos genéticos que ofrece las mayores oportunidades de exportación como alimento procesado exclusivo y sin competencia. Desafortunadamente, muchas de las estrategias de manejo del cultivo de la papa, han sido adaptadas a papa criolla (*Solanum phureja*), creando un sistema productivo poco eficiente (Piñeros, 2009). Los altos precios de los insumos agrícolas y el mal manejo de los cultivos agronómicamente provoca baja productividad y amenaza la competitividad del sistema de producción, por lo que es importante identificar factores limitantes del rendimiento y desarrollar practicas innovadoras para el cultivo, tales como una gestión nutricional integrada y equilibrada, que es una de las practicas mas eficientes para garantizar a la planta la oportunidad de expresar su potencial genético que

eventualmente se reflejara en una mejor calidad y rendimiento (López *et al*, 2014).

Varias investigaciones revelan que el tamaño promedio de los tubérculos, su variabilidad y su conteo, definen una única distribución, presentando mayor variabilidad de rangos de tamaño en la etapa final de llenado, con aumento del rendimiento de los mayores tamaños por cuenta de los tamaños inferiores, lo que sugiere que un tubérculo que pasa de un calibre inferior al siguiente, no es sustituido por otro de calibre anterior, además, parece existir una correlación negativa entre la variabilidad relativa del tamaño del tubérculo y el número de tubérculos por unidad de área (Struik, 1991).

Es importante reconocer, que debe tenerse cuidado con la interpretación de esta correlación, pues hasta cierto punto, la naturaleza de los datos asociados al número de tubérculos por planta pudiera ser similar a la de los datos composicionales, es decir, si un componente aumenta, el otro esta forzado a disminuir, lo que genera correlaciones sin conexión lógica en datos verdaderamente composicionales, sin embargo, el número de tubérculos por calibre no suma una constante, pudiera ocurrir que su máximo posea una distribución específica, entonces una mayor cantidad de tubérculos de un calibre podría provocar un número menor de tubérculos en otro calibre específico, lo que en esencia puede generar correlaciones en las que debe tenerse mucho cuidado al momento de su interpretación.

La importancia del cultivo obliga a muchos investigadores a realizar diferentes estudios para mejorar la práctica agrícola, la descripción y clasificación de las variables que impacten en el estudio del crecimiento de los tubérculos, ya que de eso depende el uso a darles; procurando siempre hacer uso del espacio de siembra de forma óptima, por lo que varios estudios evidencian la necesidad del análisis de la densidad de siembra para evaluar sobre todo el rendimiento. Bernal(2017) evaluó en lugar del rendimiento, un indicador que se asocia directamente como lo es el calibre de los tubérculos, los cuales en la práctica se manipulan en cuatro categorías de diámetro promedio (hasta 2 cm, de 2 a 4 cm, de 4 a 6 cm y más de 6 cm). La naturaleza de esta variable imposibilita usualmente la comparación de la respuesta (conteos de tubérculos) para cada densidad de siembra (definidas como 30cm*100cm, 40cm*100cm y 50cm*100cm) mediante análisis de varianza, pues en el caso de conteos, otras distribuciones como la poisson y la binomial negativa se adaptan mucho mejor al tipo de dato generado. Los modelos clásicos de regresión poisson y binomial negativa, en sus modalidades usuales o en la opción inflada por ceros (Cameron, 1998) en datos de conteo pertenecen a la familia de modelos lineales generalizados (Zeileis, 2008) y los desarrollos recientes permiten generar varias estadísticas que permiten su comparación con sus contrapartes sin ceros en

exceso, lo que resulta útil en la elección del mejor modelo para relacionar predictores y respuesta en conteos.

Sin embargo, los estudios realizados hasta ahora son todos haciendo suposiciones del tipo lineal y haciendo modificaciones que permitan así su estudio a través de *ANOVA* o regresión lineal, tomando en cuenta los patrones de vecindad, más la propuesta consiste en clasificar los datos a través de una red neuronal que no amerita de suposiciones estadísticas o matemáticas, que sólo aprendiendo de lo observado es capaz de realizar una clasificación, a partir de la información existente de la relación latente entre las densidades de siembra y el tamaño del tubérculo, que como herramienta descriptiva pueda ayudar a la planificación de los procesos de siembra.

Además dado que el paquete de clasificación basado en redes neuronales probabilísticas se ha desarrollado en lenguaje R, cuyo paradigma es de código abierto y colaborativo, esta investigación ha permitido desarrollar una herramienta computacional y estadística que sea escalable a la incorporación de nuevas funciones.

El paquete se ha desarrollado con las funcionalidades de clasificación de cualquier tipo de datos sin importar la cantidad de entradas o el número de clases a clasificar, así como con funcionalidades que permiten evaluar mencionada clasificación a traves de graficos y estadisticas como la sensibilidad y especificidad de la misma.

Capítulo 2

Fundamentos teóricos

2.1. Antecedentes

La clasificación por medio de redes neuronales ha sido un hito ya marcado en el campo agronómico, muchos estudios se han realizado con el objetivo de analizar ciertos comportamientos de plantas y los beneficios que se puedan sacar de ellas. En el año 2011, Mayabiro E. presento en la Universidad Nacional Experimental del Táchira, un prototipo sobre el entorno *MATLAB* para el cálculo de la tasa de germinación de plántulas de pimentón previamente segmentadas. El entorno desarrollado permitió establecer una clasificación de las plántulas, hojas u objetos de la misma por medio de redes neuronales. La investigadora realizó el entrenamiento de múltiples redes neuronales multicapas con algoritmos de retropropagación, donde aunque variaban las capas intermedias de las redes y sus funciones de transferencia fueron entrenadas con los mismos datos de entrada, validandolas con una base de datos de pruebas para seleccionar al final una con salidas similares a las deseadas.

En el año 2011, el grupo de investigación de sistemas de procesamiento y control de señales de la Universidad Nacional Tenaga de Malasia desarrollo un sistema de inteligencia con un enfoque novedoso para la clasificación de frutas usando técnicas de procesamiento de imágenes digitales y redes neuronales artificiales, con el objetivo de desarrollar un método de clasificación rápido con una meta del 100 % de eficiencia. El estudio se realizo con cinco especies de frutas, manzanas, platanos, zanahorias, mangos y naranjas, extrayendo de ellas siete características en función de la forma y el color. La captura de las imágenes se realizo con una cámara digital convencional y las manipulaciones a los datos y construcción de la red con el software *MATLAB*. Los resultados obtenidos durante esta investigación fueron de gran avance en el campo de reconocimiento de patrones en imágenes.

Otro estudio, realizado en el año 2013, por Stephen Gang Wu consistía en el estudio teórico de técnicas de procesamiento de imágenes y datos para el reconocimiento automático de hojas para la clasificación de plantas. Doce características de las plantas fueron extraídas y distribuidas en cinco variables principales que constituían el vector de entrada de una red neuronal artificial probabilista, que había sido entrenada con 1800 hojas para clasificar 32 tipos de plantas con una precision superior al 90 %, el autor asegura que su metodología de implementación de la PNN era fácil y rápida en comparación de otras investigaciones similares.

En el año 2017, Bernal N realizó un estudio de campo con el cultivo de papa criolla para evaluar la influencia de la densidad de siembra asociada a distancias entre plantas de 30,40 y 50 cm y distancias entre surcos de 100 cm sobre el conteo de tubérculos de calibres inferiores a 2 cm, entre 2 y 4 cm, entre 4 y 6 cm, y de mas de 6 cm de diámetro ponderado y sobre el peso fresco en gramos de los tubérculos. Los tubérculos cosechados se clasificaron y contaron mediante tamizado y se pesaron en su totalidad sin discriminar por calibre. Los modelos estadísticos empleados para modelar el comportamiento de la cosecha, evidenciaron el efecto significativo de la densidad de siembra sobre el conteo de tubérculos y calibre y se observo una razón aproximada de 40:40:20:1 desde el calibre menor al mayor. El efecto de la competición, en todos los modelos probados resulto significativo, aumentando en la mayoría de los casos a medida que disminuía la distancia entre plantas, tanto en el patrón de vecindad intrahileras como en el caso de inter e intrahileras.

2.2. Bases Teóricas

2.2.1. Papa criolla (*Solanum Phureja*).

Solanum tuberosum grupo *phureja*, o papa criolla, hace referencia a los fenotipos conocidos como yema de huevo, los cuales presentan color de piel y carne amarilla (Rodriguez *et al.*, 2009). La estructura de la planta y los fenotipos puede ser observado en la figura 2.1.

Las plantas de papa criolla (*Solanum phureja*) suelen ser sembradas en calles separadas por una determinada distancia que es lo que se conoce como la densidad de siembra y generalmente con espacios entre surcos de un metro (Bernal, 2017). En la figura 2.2 son identificadas las calles y los surcos en un sembradío.

En los cultivos de papa criolla (*Solanum phureja*) son muchas las variables que influyen en la

Figura 2.1: Estructura de la planta *Solanum phureja*.

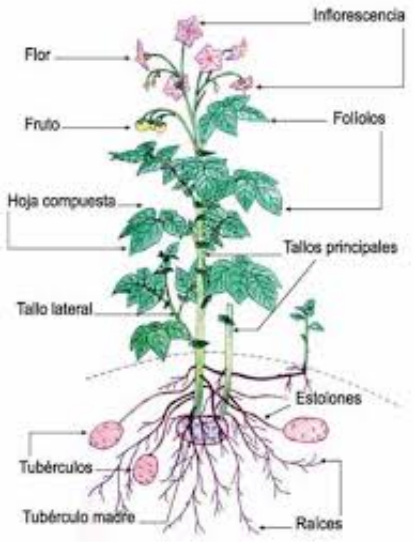


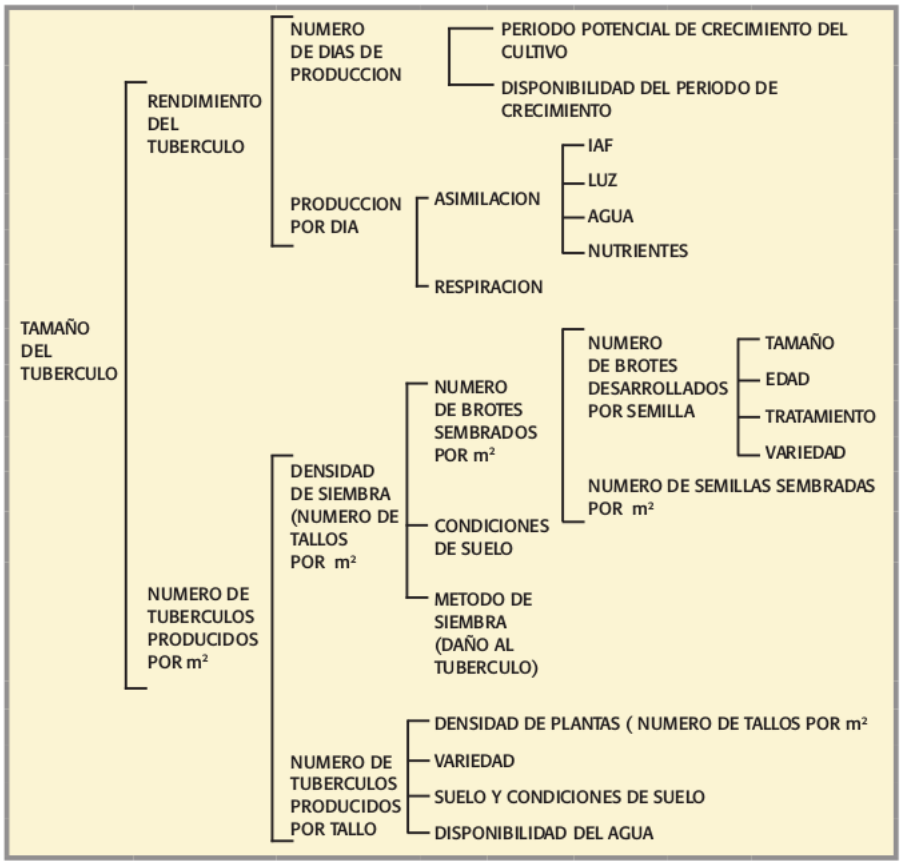
Figura 2.2: Sembradío de *Solanum phureja*.

cosecha de los tubérculos, tales como la temperatura, la altura del terreno, el clima, el tipo y la composición del suelo, la densidad de siembra, entre otros. Y de ellas dependen el crecimiento de los tubérculos producidos por la planta (Bernal, 2017). Las variables que influyen en el rendimiento de la papa pueden ser observados en la figura 2.3.

2.2.2. Redes neuronales artificiales.

Una red neuronal artificial, es un conjunto de nodos de un programa (neuronas) interconectados entre si, simulando el proceso de pensamiento humano, se pudiera considerar como una caja negra

Figura 2.3: Variables de influencia sobre el rendimiento de *Solanum phureja*.



entrenada previamente para esperar una entrada y basado en las características o comportamiento de la misma proporcionar una determinada salida, eliminando así la necesidad de diferentes algoritmos que deban analizar comportamientos cada uno por separado (Stephen, 2007).

Entre las definiciones más recientes de inteligencia artificial se expresa, en forma general, la inteligencia artificial como la capacidad que tienen las máquinas para realizar tareas que en el momento son realizadas por seres humanos; otros autores como Nebendah (1988) y Delgado (1998) dan definiciones más completas y las definen como el campo de estudio que se enfoca en la explicación y emulación de la conducta inteligente en función de procesos computacionales basados en la experiencia y el conocimiento continuo del ambiente. Autores como Marr (1977), Mompin (1987), Rolston (1992), en sus definiciones involucran los términos de soluciones a problemas muy complejos.

El nacimiento de la inteligencia artificial se sitúa en los años cincuenta; en esa fecha la informáti-

ca apenas se había desarrollado, y ya se planteaba la posibilidad de diseñar máquinas inteligentes. Hoy en día se habla de vida artificial, algoritmos genéticos, computación molecular o redes neuronales. En algunas de estas ramas los resultados teóricos van muy por encima de las realizaciones prácticas.

A través de los años, se han utilizado diversas técnicas de inteligencia artificial para emular "comportamientos inteligentes". Al software que hace uso de dichas técnicas se le denomina, de forma genérica, "sistema inteligente", y es cada vez más amplia la gama de aplicaciones donde incide la inteligencia artificial.

Las redes neuronales artificiales son eficientes en tareas tales como: el reconocimiento de patrones, problemas de optimización o clasificación, y se pueden integrar en un sistema de ayuda a la toma de decisiones, pero no son una panacea capaz de resolver todos los problemas: todo lo contrario, son modelos muy especializados que pueden aplicarse en dominios muy concretos.

Las redes neuronales emulan la estructura y el comportamiento del cerebro, utilizando los procesos de aprendizaje para buscar una solución a diferentes problemas; son un conjunto de algoritmos matemáticos que encuentran las relaciones no lineales entre conjuntos de datos; suelen ser utilizadas como herramientas para la predicción de tendencias y como clasificadoras de conjuntos de datos. Se denominan neuronales porque están basadas en el funcionamiento de una neurona biológica cuando procesa información.

2.2.3. Redes neuronales probabilísticas.

Las redes neuronales se utilizan para referirse a una amplia clase de modelos y algoritmos. Las neuronas ocultas se generan en base a alguna combinación de los datos observados, similar a una expansión de base en otras técnicas estadísticas. Sin embargo, en lugar de elegir la forma de la expansión, los pesos utilizados para crear las neuronas ocultas se calculan a partir de los datos. Las redes neuronales pueden implicar una variedad de funciones de activación, que son las transformaciones de las entradas de datos brutos ponderados para crear las neuronas ocultas (Wiley, 2016).

Una red neuronal probabilística (PNN) no es mas que una ANN que usa funciones estadísticas que escalan la variable no linealmente como una forma de campana o una distribución normal (Stephen, 2007).

Una opción común para las funciones de activación de las redes neuronales es la función sigmoide: $\sigma(x) = \frac{1}{1+e^{-x}}$ y la función tangente hiperbólica $f(x) = tanh(x)$. Sin embargo para las redes neuronales probabilísticas que la decisión de cada una de sus neuronas está basada en probabilidades se suelen usar funciones de base radial y aunque suelen existir una variedad de ellas, las PNN usan la forma *Gaussiana*:

$$f(x) = exp(-\frac{||x-c||^2}{2\sigma^2})$$

2.2.4. Curvas ROC.

Una amplia gama de tests diagnósticos reportan sus resultados cuantitativamente, utilizando escalas contínuas. El análisis de curvas ROC (*receiver operating characteristic curve*) constituye un método estadístico para determinar la exactitud diagnóstica de estos *tests*, siendo utilizadas con tres propósitos específicos: determinar el punto de corte de una escala continua en el que se alcanza la sensibilidad y especificidad más alta, evaluar la capacidad discriminativa del test diagnóstico y comparar la capacidad discriminativa de dos o más *test* diagnósticos que expresan sus resultados como escalas continuas.

2.2.5. Lenguaje R.

R es un conjunto integrado de *software* de código abierto para el almacenamiento, manipulación, cálculo y visualización de datos para computación y gráficación estadística, puede ser compilado y ejecutado en Windows, Mac OS X y otras plataformas UNIX (como Linux), se distribuye usualmente en formato binario (`https://www.r-project.org/about.html`, 2018). El proyecto de *software* R fue iniciado por Robert Gentleman y Ross Ihaka. El lenguaje fue influenciado por lenguaje S desarrollado originalmente en *Bell Laboratories* por John Chambers y sus colegas. Desde entonces ha evolucionado para el cálculo estadístico asociado a diversas disciplinas para contextos académicos y comerciales. En R, la unidad fundamental de código compartible es el paquete, el cual agrupa código, datos, documentación y pruebas, y resulta simple de compartir con otros. Para enero del 2015 ya habían más de 6.000 paquetes disponibles en la Red Integral de Archivos de R, conocido comúnmente por su acrónimo CRAN, el cual es el repositorio de paquetes . Esta gran variedad de paquetes es una de las razones por las cuales R es tan exitoso, pues es probable que algún investigador o académico ya haya resuelto un problema en su propio campo usando esta herramienta, por lo que otros usuarios simplemente podrán recurrir a ella para su uso directo o para llamarla en un nuevo código (Wickham,2015).

2.2.6. Estructura de paquetes en R/RStudio.

Los requerimientos de núcleo (*core*) para que un desarrollo de software pueda considerarse y compilarse como un paquete de lenguaje R debe cumplir con las siguiente estructura.

1. DESCRIPTION: metadatos del package.
   La tarea del archivo *Description* es de gran importancia ya que es en el donde se registra la metadata, las dependencias que utiliza el paquete, la licencia y el soporte en caso de ocurrir errores con el mismo. La estructura mínima para realizar un paquete en R es la siguiente:

   - Package: mypackage
   - Title: What The Package Does (one line, title case required)
   - Version: 0.1
   - Authors@R: person("First", "Las", email = "first.last@example.com",
   - role = c("aut", "cre"))
   - Description: What the package does (one paragraph)
   - Depends: R (>= 3.1.0)
   - License: What license is it under?
   - LazyData: true

2. `R/`: dirección del repositorio donde se encuentra el código del paquete (.R files).
   Se expondrán las buenas prácticas a la hora de realizar todo nuestro código en R, desde organización de las funciones, estilos de código y nombre de variables

   Organizar funciones en R: aunque se pueden organizar los archivos como se desee, los dos extremos son malos, no colocar todas las funciones en el mismo archivo y no crear un archivo para cada función, aunque si una función es muy grande o tiene mucha documentación se puede dar el caso, los nombres de los archivos tienen que ser significativo y deben de terminar en *.R*.

   - Bien
     - fit_models.R

    - utility_functions.R
- Mal
    - foo.r
    - stuff.r

Se puede recomendar de acuerdo al número de función utilizar prefijo.

Nombres de Objetos: Los nombres de las variables y funciones deben de ser en minúsculas, usar el guión bajo (_) para separar palabras.

- Bien
    - day_one
    - day_1
- Mal
    - first_day_of_the_month
    - DayOne
    - dayone
    - djm1

En lo posible no usar nombres de variables existentes, ya que esto causará confusión.

Espaciado: Se recomienda colocar espacios alrededor de todos los operadores lógicos y aritméticos (=, +, -, <-, etc.). siempre coloque un espacio después de una coma, y nunca antes de ella.

- Bien
    - average <- mean(feet / 12 + inches, na.rm = TRUE)
- Mal
    - average<-mean(feet/12+inches,na.rm=TRUE)

Hay una pequeña excepción a esta regla: (:, :: y :::) no necesitan espacios alrededor de ellos.

- Bien
    - x <- 1:10
    - base::get

- Mal
  - x <- 1 : 10
  - base :: get

Dejar un espacio antes del paréntesis izquierdo, excepto en la llamada a una función.

- Bien
  - if (debug) do(x)
  - plot(x,y)
- Mal
  - if(debug)do(x)
  - plot(x, y)

Se utiliza más de un espacio en caso de que esto mejore a la alineación, por ejemplo:

```
list(
    total  = a + b + c,
    mean   = (a + b + c) / n
)
```

No coloque espacios alrededor del código entre paréntesis o corchetes (a menos que haya una coma)

- Bien
  - if (debug) do(x)
  - diamonds[5,]
- Mal
  - if (debug) do(x) *# no espacios alrededor de debug*
  - x[1,] *# necesita un espacio despues de la coma*
  - x[1 ,] *# el espacio va despues de la coma no antes*

Llaves: Una llave de apertura nunca debe ir en su propia línea y siempre debe ir seguida de un nueva línea. Una llave siempre debe ir en su propia línea, a menos que sea seguida por otra y siempre sangría en el código dentro de las llaves

- Bien
  - if (y < 0 && debug) {
    message("Y es negativo")
    }
  - if (y == 0) {
    log(x)
    } else {
    y ^ x
    }
- Mal
  - if (y < 0 && debug)
    message("Y es negativo")
  - if (y == 0) {
    log(x)
    }
    else {
    y ^ x
    }

Sentencias que son muy cortas esta bien dejarlas en la misma línea.
if(y < 0 && debug) message("Y es negativo")

Longitud de Línea: cada línea debe de llevar máximo 80 carateres, si se queda sin espacio es recomendable utilzar una función separada

Sangria: Utilize sangria de 2 espacios, nunca use tabulador o multiples tabuladores o espacios. La unica excepción es cuando se define una sentencia en multiples líneas.

```
long_function_name <- function(    a = "a long argument",
                                   b = "another argument",
                                   b = "another argument",
```

Asignación: Usar el <-, y no =

- Bien
  - x <- 5
- Mal
  - x = 5

3. man/: documentación.

4. NAMESPACE: específica que objetos conforman el paquete.

Comentarios: Comente su codigó, el comentario comienza #, los comentarios deben de explicar el porque, no el que.
Use los caracteres (-) y (=) para separar lineas
Load data -
Plot data -

2.2.7. RStudio.

RStudio es un ambiente de desarrollo integrado (*Integrated Development Environment*, IDE) que ofrece herramientas de desarrollo vía consola, editor de sintaxis que apoya la ejecución de código, así como herramientas para el trazado, la depuración y la gestión del espacio de trabajo. RStudio está disponible para Windows, Mac y Linux o para navegadores conectados a RStudio Server o RStudio Server Pro (Debian / Ubuntu, RedHat / CentOS, y SUSE Linux) (https://www.rstudio.com/about/, 2018).

Capítulo 3

Fundamentos metodológicos

A continuación se plantea la estructura a seguida por el presente trabajo, detallando el enfoque, tipo, nivel y diseño de la investigación y la metodología a implementar, entre otros.

3.1. Enfoque de la investigación

En función de los objetivos planteados en la presente investigación, la misma se adecua a los parámetros de una investigación de tipo proyectivo. La Universidad Pedagógica Experimental Libertador UPEL (2006) expone que la investigación proyectiva consiste en encontrar la solución a los problemas prácticos, se ocupa de cómo deberían ser las cosas para alcanzar los fines y funcionar adecuadamente. Consiste en la elaboración de una propuesta o de un modelo, para solucionar problemas o necesidades de tipo práctico, ya sea de un grupo social, institución, o un área en particular del conocimiento, partiendo de un diagnóstico preciso de las necesidades del momento, los procesos explicativos o generadores involucrados y las tendencias futuras.

3.2. Diseño de la investigación

El diseño de la investigación es la estrategia general que adopta el investigador para responder al problema planteado, por lo que es vital establecer una correcta secuencia de pasos para elaborar el prototipo de software que dará solución a la problemática principal de la investigación (Arias, 2012).

3.3. Fases metodológicas

3.3.1. Desarrollo del paquete en R

Los pasos seguidos en el desarrollo de esta investigación serán descritos a continuación siendo basados en los antecedentes y estudios realizados y las pautas estándar establecidas para la creación de paquetes y extensiones en R.

Creación del esqueleto del paquete.
En esta etapa se diseñó y creó los directorios, ficheros y objetos que conformaron el paquete.

Diseño de las soluciones algorítmicas.
Los métodos para el diseño de las funciones fueron los diagramas de flujo, definiendo sus entradas y salidas.

Codificación de los algoritmos.
Para su codificación se seguieron las normas de estilo para codificación en R, sugeridas por Wickham (2015) y por el creador del paquete *formatR* Xie(2017), además se establecieron la dependencia con las funciones de código base y las recomendadas para desarrollo en R.

Registrar el método para el envío y uso de funciones.

Es esta etapa del desarrollo se establecerán las dependencias sobre los paquetes de la base fuente de código R y sus métodos de conexión, considerando el manejo de versiones y los criterios de mantenimiento, además se establecerán los espacios de nombre o las estrategias para la búsqueda y utilización de las variables; unificando estos criterios a las funciones que serán diseñadas.

Diseñar y entrenar la red neuronal probabilística.

En esta etapa se definirán las variables de entrada y se declararan las neuronas patrones y las clases de salida que permitirán la clasificación, para proceder con el entrenamiento de la red que se hará a través del método de *jackknifing* para determinar el parámetro de escalamiento correcto que permita la mejor clasificación.

Pruebas unitarias de las funciones.

Debido a que los paquetes en R están conformados, entre otros elementos por las funciones primarias, a cada una de ellas se les realizaran pruebas unitarias en dos fases, la primera con datos sintéticos que permitan comprobar cada estado del diagrama de flujo, que esquematiza la solución numérica y/o lógica que permite la construcción de la PNN y su entrenamiento. Entre las herramientas ha utilizarse se encuentran los paquetes de R *RUnit* (Zenka, 2015) y *testthat* (Wickham, 2017), y la segunda etapa donde se realizaran las pruebas funcionales del paquete con el conjunto de datos de prueba que formaran parte integral del paquete y con los cuales se desarrollaran los ejemplos prácticos que conformaran la documentación que acompaña al paquete R.

Implementar y realizar pruebas de clasificación.

El objetivo de esta etapa es definir los casos de entrenamiento y prueba que serán usados para buscar la mejor clasificación de densidad de siembra de tubérculos de papa criolla ante diferentes escenarios.

Chequear la carga del paquete.

En esta etapa del desarrollo se utilizaran las funciones de chequear paquete que ofrece el código R; cuya finalidad es verificar cada fichero del árbol de carpetas asociadas a cada elemento de la estructura o esqueleto del paquete, que a su vez creara el archivo de documentación en LaTeX y/o HTML, compilará el código fuente y creará las librerías de enlace dinámico (*dynamic link library* DLL).

Construcción del método de distribución del paquete.

Se seleccionara la forma de distribución del paquete desde el repositorio local, creando los ficheros fuentes (en formato *tarball*) y en binario.

3.3.2. Comparación de resultados.

En este paso se procederá a la construcción de las curvas ROC y gráficos de nubes de puntos asociadas a los casos de prueba planteados para el paquete desarrollado, permitiendo concluir de forma descriptiva sobre relaciones entre variables, el mejor manejo de los datos.

Capítulo 4

Desarrollo

A continuación se describe de forma detallada, el diseño y desarrollo del paquete **apnnClassifier** disponible en: https://github.com/ghouljd/apnnClassifier para la clasificación de tubérculos de papa criolla(*Solanum phureja*) para diferentes densidades de siembra empleando redes neuronales probabilísticas, siguiendo la metodología descrita en el Capítulo 3.

4.1. Creación del esqueleto del paquete.

Para esta fase se utilizó el ambiente de desarrollo RStudio, el cual es un entorno integrado de fuente abierta para R que agrega muchas características y herramientas de productividad, además de facilitar el uso de R integrando la ayuda y la documentación.

En esta primera etapa se creó la estructura del paquete, la cual está conformada por la identificación por nombre y descripción del paquete (Figura 4.1), además, se cargan los paquetes del CRAN de R y sus extensiones, de los cuales dependen los métodos y funciones que permitiran el desarrollo del paquete **apnnClassifier**; los mismos fueron almacenados en el repositorio local que se crea para este paquete durante el proceso de desarrollo; los principales fueron:

- pnn(Chasset, P. *et. al*, 2013;https://cran.r-project.org/web/packages/pnn/index.html)
- pROC(Robin, X. *et. al*, 2019; ;https://cran.r-project.org/web/packages/pROC/index.html)

- dplyr(Wickham, H. *et. al*, 2019;;https://cran.r-project.org/web/packages/dplyr/index.html)
- roxygen2(Ooms, J. *et. al*, 2018;;https://cran.r-project.org/web/packages/roxygen2/index.html)

Figura 4.1: Descripción del paquete

```
Package: apnnClassifier
Type: Package
Title: Multiclass classifier based on neural networks.
Version: 1.0.0
Author: Jesús Escalante <jesusd.escalante@unet.edu.ve>
Maintainer: Jesús Escalante <jesusd.escalante@unet.edu.ve>, Rossana Timaure <rttg@unet.edu.ve>
Description: Classifier using probabilistic neural networks.
License: AGPLv3
Encoding: UTF-8
LazyData: true
Imports:
  pnn,
  dplyr,
  pROC
RoxygenNote: 6.1.1
```

4.2. Diseño de las soluciones algorítmicas.

4.2.1. Diseño de las soluciones algorítmicas para el entrenamiento de una red neuronal probabilística que permita la clasificación de tubérculos de papa.

El objetivo funcional del paquete desarrollado es permitir al usuario realizar el entrenamiento de una red neuronal probabilística para la clasificación de datos estandarizados o no estandarizados, así como graficar y evaluar los resultados de la clasificación.

Para este fin se desarrollaron 2 funciones, necesarias para llegar a los resultados esperados y una opcional para estandarizar los datos de entrada.

Como entrada para iniciar el proceso de entrenamiento y clasificación, es necesario tener un conjunto de datos de entrenamiento en una estructura *data.frame*, teniendo conocida cual es la

columna que identifica las clases del conjunto; así como tener un conjunto de datos de entrenamiento en forma de matriz con la misma cantidad de columnas como variables clasificadoras tenga el conjunto de entrenamiento.

Las funciones antes mencionadas junto a sus entradas, salidas y diagramas de flujos serán descritas a continuación.

4.2.2. Diseño del algoritmo de la función de entrenamiento y clasificación de la red (trainNeuralNet).

Para el caso de la creación y entrenamiento de la red neuronal probabilística para permitir la clasificación de los datos, se diseñó el ingreso de los datos en dos conjuntos de datos, uno de entrenamiento con la columna de clase incluida y un conjunto de pruebas con la misma cantidad de datos de entrenamiento que el primer conjunto, además de, el índice de la columna indicadora de clase en el conjunto de entrenamiento y el valor óptimo de la función de activación en caso de conocerse (Cuadro 4.1.). La función diseñada hace uso del paquete **PNN** (Chasset, P) de R para crear y entrenar una red neuronal probabilística, el algoritmo de la función se encarga de crear, entrenar, optimizar y clasificar los datos del conjunto de pruebas recibido, obteniendo al final una red neuronal entrenada capaz de clasificar y lista para ser evaluada junto a su clasificación (Figura 4.2).

Entradas		
Nombre	**Descripción**	**Reglas**
train_set	Conjunto de entrenamiento	Parámetro de tipo *data.frame*. Requerido.
test_set	Conjunto de pruebas	Parámetro de tipo *matriz*. Requerido.
category_column	Índice de la columna que identifica la categoría o clase en el conjunto de entrenamiento	Parámetro de tipo *entero*. No requerido. Valor por defecto: 1.
sigma	Valor óptimo de la función de activación	Parámetro de tipo *flotante*. No requerido. Es calculado en caso de ausencia del parámetro.

Cuadro 4.1: Entradas de la función de entrenamiento - **trainNeuralNet**.

4.2.3. Diseño del algoritmo de la función de evaluación de la red neuronal probabilística y su clasificación (evaluate).

Para el caso de la evaluación de la red neuronal probabilística y su clasificación, se unificó el ingreso de los datos en la lista de R que se traduce a la red neuronal probabilística que es salida de la función **trainNeuralNet**, descrita anteriormente (Cuadro 4.2). La función diseñada realiza la creación del gráfico de nubes de puntos, para demostrar la correlación entre las variables de entrenamiento y hace uso del paquete **pROC** (Robin, X) de R para crear el gráfico de curvas características operativas (Curvas ROC) para identificar si la clasificación fue buena, el algoritmo también permite observar datos de análisis de la red como su sensibilidad, efectividad y área bajo la curva, para determinar un mal o un buen análisis (Figura 4.3.).

Entradas		
Nombre	**Descripción**	**Reglas**
pnn	Red neuronal probabilística	Parámetro de tipo *lista*, que es salida de la función descrita en la sección 4.3. Requerido.

Cuadro 4.2: Entradas de la función de evaluación - **evaluate**.

La predicción del conjunto de pruebas se puede observar en el atributo "output" de la red neuronal devuelta por la función y es un *data.frame* con dos columnas, una que tiene la predicción y otra con la probabilidad con la que se obtuvo dicha predicción, como se puede observar en la figura 4.4.

4.2.4. Diseño del algoritmo de la función de estandarización de datos (standardize).

Para el caso de la estandarización opcional de los datos, se diseñó el ingreso de los datos en un conjunto en formato *lista* o *data.frame*, para este paso se debe tener en cuenta que no tiene sentido alguno estandarizar la columna clase. El tipo de estandarización a usar, que puede ser puntual por media y varianza o escalar por minimos y máximos (Cuadro4.3). La función diseñada dependiendo del tipo de estandarización indicado en la entrada, aplica la función de estandarización columna por columna del conjunto de datos (figura 4.5).

Entradas		
Nombre	**Descripción**	**Reglas**
set	Conjunto de datos	Parámetro de tipo *lista* o *data.frame*. Requerido.
type	Tipo de estandarización	Parámetro de tipo cadena de caracteres. No requerido. Valores posibles: "punctual-"scale". Valor por defecto: "punctual".

Cuadro 4.3: Entradas de la función de estandarización - **evaluate**.

4.3. Codificación de los algoritmos

La codificación de los algoritmos se realizo en el IDE(Entorno de desarrollo integrado) RStudio, siguiendo las especificaciones algoritmicas detalladas con anterioridad.

A continuación se describen las principales funciones creadas para el paquete **apnnClassifier**.

- **Función de entrenamiento y clasificación de la red (trainNeuralNet)**, crea y entrena una red neuronal probabilística y depende del paquete de R Pnn, la codificación de dicha función se muestra en la figura 4.6.
- **Función de evaluación de la red (evaluate)**, evalua la especificidad, efectividad y sensibilidad de la clasificación realizada por la red neuronal probabilística y depende del paquete de R **pROC** y **dPlyr**, la codificación de dicha función se muestra en la figura 4.7.
- **Función de estandarización de datos (standardize)**, se encarga de estandarizar los datos basado en dos tipos de estandarización para datos discretos o continuos, la codificación de dicha función se muestra en la figura 4.8.

4.4. Chequeo del paquete, método de distribución y registro del método de envío.

Para realizar el chequeo del paquete se utilizaron las pruebas unitarias proporcionadas por los paquetes R *RUnit* (Zenka, 2015) y *testthat* (Wickham, 2017). Ver Figura 4.9.

Para el método de distribución se utilizara *tar.gz*, que es utilizado por defecto por RStudio. Ver figura 4.10.

Como método de envío local se utilizará la plataforma de *GitHub*.

4.5. Pruebas funcionales del paquete

El objetivo de este trabajo investigativo consiste en realizar un clasificador de tubérculos de papa criolla para diferentes densidades de siembra y la solución se planteó desarrollando un paquete en R que empleando redes neuronales probabilísticas permitiera dicha clasificación. Para las pruebas funcionales del paquete se realizó la corrida de un algoritmo que permitiera a través del paquete clasificar tubérculos de papa con los datos recogidos en la investigación de Bernal, 2017.

Las características que se tomaron para dicha prueba fueron el diámetro medio ponderado, el peso fresco y la cantidad de papas de un conjunto de 2840 datos de campo recogidos en la investigación de Bernal en 2017 (figura 4.11). Dicho conjunto de datos fue dividido para la ejecución de las pruebas funcionales en un 75 % para el conjunto de entrenamiento y un 25 % para el conjunto de pruebas. Dicha segmentación se realizó de manera aleatoria por densidad de siembra para asegurar obtener resultados de valor.

La codificación de la anterior división, así como el uso de cada una de las funciones del paquete para permitir la clasificación de tubérculos de papa criolla (Solanum phureja) con mencionados conjuntos de datos como entradas se puede observar en la figura 4.12.

Los resultados obtenidos por el algoritmo se observan en las figuras 4.13 y 4.14. El concepto de las curvas ROC es contrastar la especificidad (fracción de verdaderos negativos) con la sensibilidad (fracción de verdaderos positivos). Una buena clasificación es caracterizada porque estos dos cálculos se comporten de manera inversamente proporcional, por lo que aunque las correlaciones sean altas entre las variables de entrada como se puede observar en el gráfico de nubes de puntos, no se obtiene una buena clasificación ya que para valores altos de sensibilidad se pueden observar valores altos de especificidad.

Figura 4.2: Diagrama de flujo para función de entrenamiento y clasificación de la red neuronal probabilística

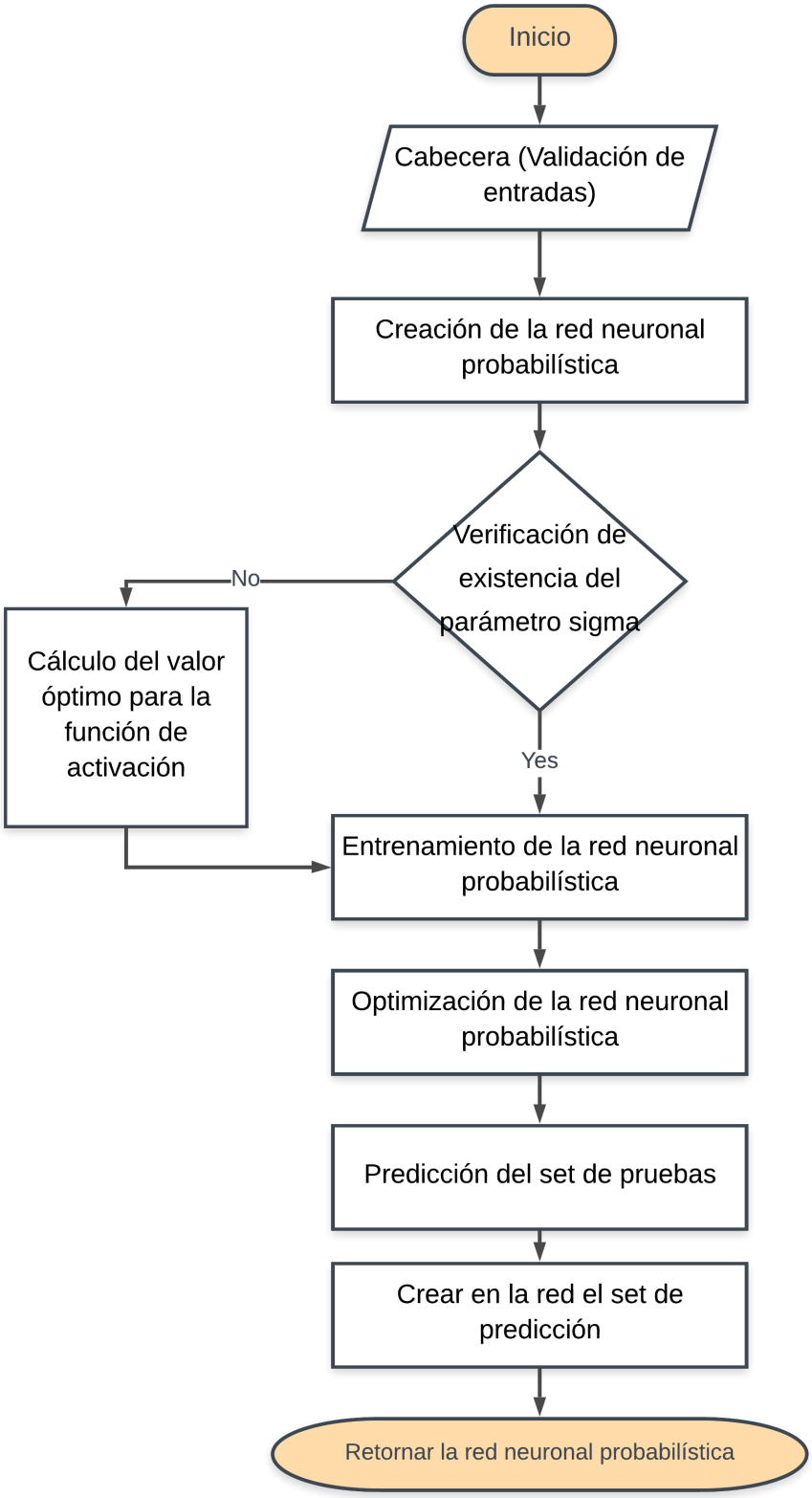


Figura 4.3: Diagrama de flujo para función de evaluación de la red neuronal probabilística

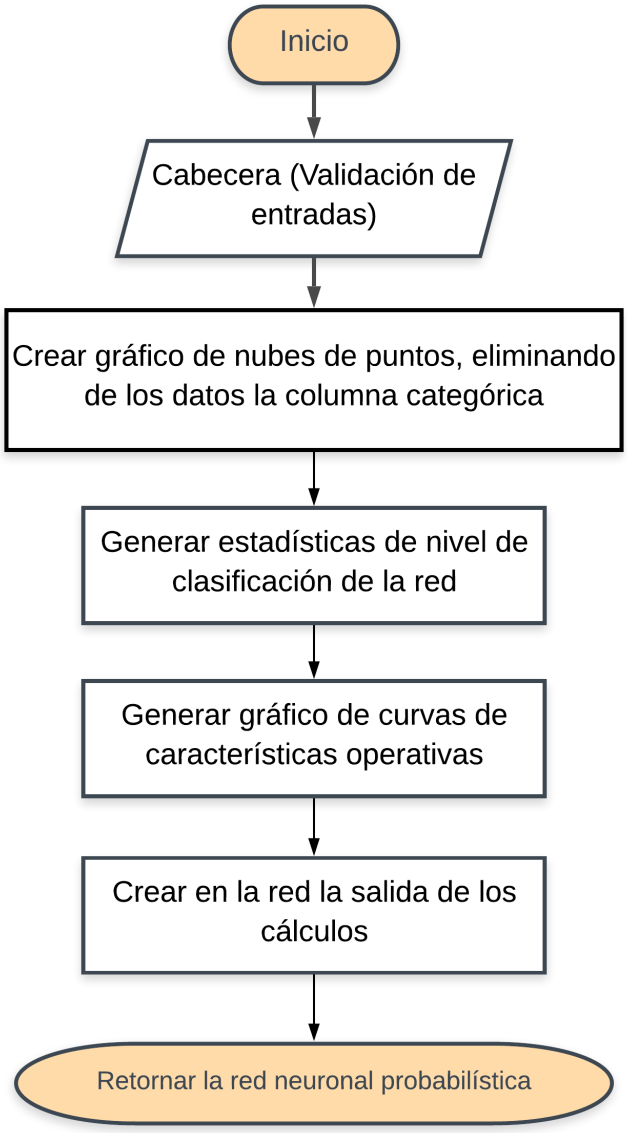


Figura 4.4: Ejemplo de predicción de set de pruebas.

	category	propability
1	1	0.3333852
2	1	0.3334506
3	1	0.3333979
4	1	0.3334506
5	1	0.3334027
6	1	0.3334027
7	1	0.3333727
8	1	0.3334002
9	1	0.3334506
10	1	0.3334399
11	1	0.3333860
12	1	0.3334336
13	1	0.3334516
14	1	0.3333765
15	1	0.3334348
16	3	0.3334156
17	3	0.3333636
18	1	0.3333665
19	1	0.3334506
20	1	0.3334109
21	1	0.3333907
22	3	0.3334152

Figura 4.5: Diagrama de flujo para función de estandarización de datos

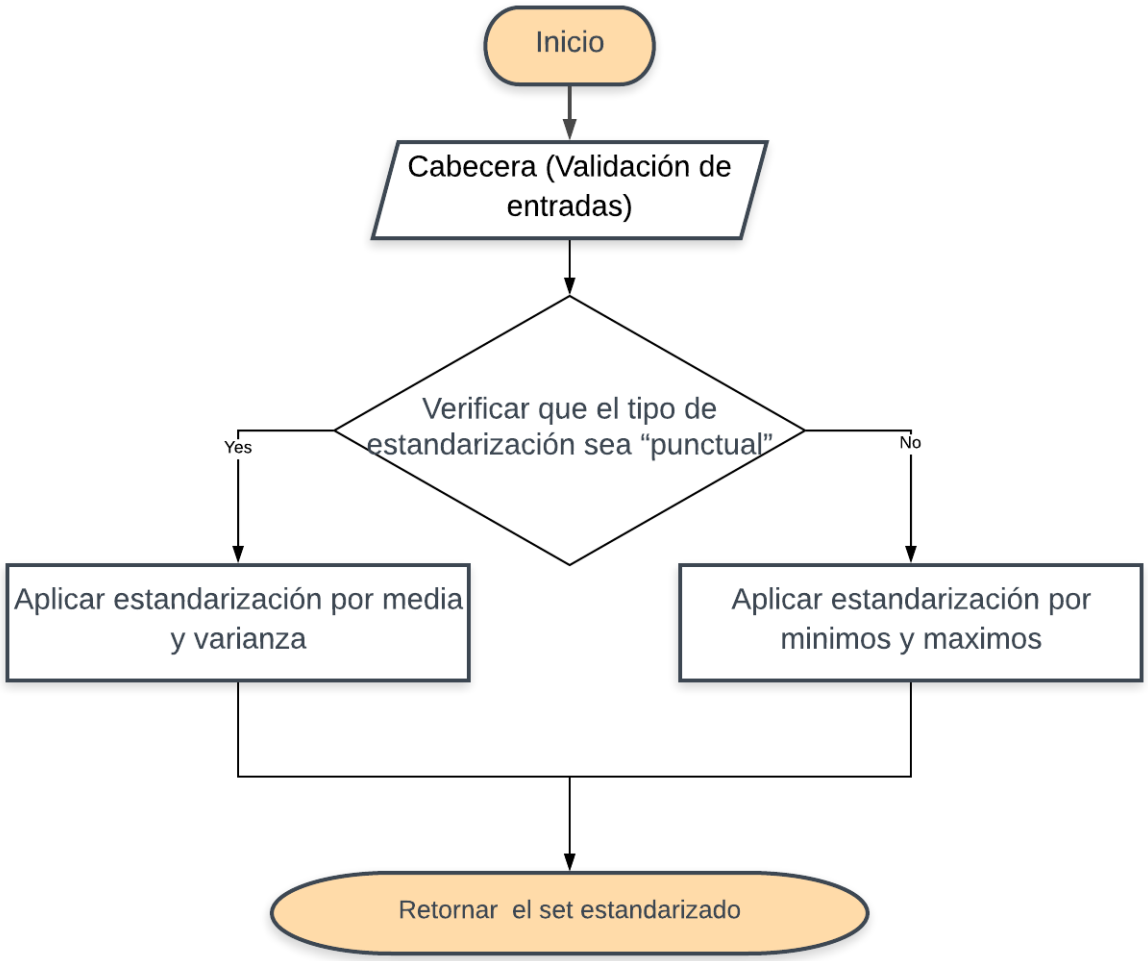


Figura 4.6: Codificación de la función de entrenamiento y clasificación de la red neuronal probabilística.

```
#' Neural network training of the pnn package for classification
#' @param train_set Training set. (Required)
#' @param test_set Testing set. (Required)
#' @param category_column Category column (Not required. Default value: 1)
#' @param sigma Optimun value for the activation function of the neural network. (Not required)
#' @return pnn \code{list} trained with the classified testing set and network performance statistics.
#' @examples
#' library(apnnClassifier)
#' data(trainData, testData)
#' testData <- as.matrix(testData)
#' # Basic usage.
#' pnn <- trainNeuralNet(train_set = trainData, test_set = testData)
#' # If you know the approximate optimal value and the sorter column is not in the first position of the set.
#' pnn <- trainNeuralNet(train_set = trainData, test_set = testData, category_column =  *sorter column*, sigma = *sigma value*)
#' View(pnn)
#' @export
# It is responsible for executing the prediction of the probabilistic neural network.
trainNeuralNet <- function(train_set, test_set, category_column = 1, sigma) {
  if(missing(train_set))
    stop("The training set shouldn't be empty or null.")
  else if(missing(test_set))
    stop("The testing set shouldn't be empty or null.")
  else if(typeof(train_set) != "list")
    stop("The training set isn't valid.")
  else if (!requireNamespace("pnn", quietly = TRUE))
    stop("Package \"pnn\" needed for this function to work. Please install it.", call. = FALSE)
  print("Begin learning process.")
  pnn = pnn::learn(train_set, category.column = category_column)
  if(missing(sigma)){
    print("Finding optimized minimum value.")
    if (!requireNamespace("rgenoud", quietly = TRUE))
      stop("Package \"rgenoud\" needed for this function to work. Please install it.", call. = FALSE)
    pnn = pnn::smooth(pnn)
  }
  else
    pnn = pnn::smooth(pnn, sigma)
  pnn = pnn::perf(pnn)
  print("End learning process.")
  print(paste("Success rate: ", pnn$success_rate * 100))
  print("Begin testing set evaluation.")
  size <- length(test_set[,1])
  output <- data.frame(category = numeric(size), propability = numeric(size))
  for (index in 1:size) {
    predict <- pnn::guess(pnn, test_set[index,])
    output$category[index] <- ifelse(!is.na(predict), predict$category, predict)
    output$propability[index] <- ifelse(!is.na(predict), max(predict$probabilities), 0)
  }
  pnn$output = output
  print("End process.")
  return(pnn)
}
```

Figura 4.7: Codificación de la función de evaluación de la red neuronal probabilística.

```
#' Evaluation of the classification of the probabilistic neural network.
#' @param pnn Trained probabilistic neural network.(Required)
#' @return pnn \code{list} with evaluated input and analysis charts
#' @examples
#' library(apnnClassifier)
#' data(trainData, testData)
#' testData <- as.matrix(testData)
#' pnn <- trainNeuralNet(train_set = trainData, test_set = testData, sigma = 0.5)
#' pnn <- evaluate(pnn)
#' View(pnn)
#' @export
# Responsible of analyze the classification of the probabilistic neural network and generating the corresponding analysis graphs.
evaluate <- function(pnn) {
  if(missing(pnn))
    stop("The pnn parameter is required.")
  else if(is.null(pnn$output))
    stop("The pnn parameter should be a neural net trained.")
  else if (!requireNamespace("pROC", quietly = TRUE))
    stop("Package \"pROC\" needed for this function to work. Please install it.", call. = FALSE)
  else if (!requireNamespace("dplyr", quietly = TRUE))
    stop("Package \"dplyr\" needed for this function to work. Please install it.", call. = FALSE)

  plot(dplyr::select(pnn$set, -pnn$category.column))

  roc.multi <- pROC::multiclass.roc(pnn$output$category, pnn$output$propability, percent=TRUE)
  rs <-roc.multi$rocs
  for (index in 1:length(rs)) {
    pROC::plot.roc(rs[[index]])
  }
  sapply(1:length(rs),function(i) pROC::lines.roc(rs[[i]],col=i))

  pnn$evaluation = roc.multi

  return(pnn)
}
```

Figura 4.8: Codificación de la función de estandarización de datos.

```r
#' Data standardization
#'
#' @param set Data set. (Required)
#' @param type standardization type (Not required. Default value: "punctual". Valid values: "punctual" - "scale")
#'
#' @return standardized set.
#'
#' @examples
#' library(apnnClassifier)
#' data(testData)
#' Basic usage
#' testData <- standardize(testData)
#' For classification by minimum and maximum
#' testData <- standardize(testData, type = "scale")
#' View(testData)
#' @export
# It is responsible for standardizing a set of data.

standardize <- function(set, type = "punctual") {
  if(missing(set))
    stop("The set parameter shouldn't be empty or null.")
  else if(type != "punctual" && type != "scale")
    stop("Type parameter isn't valid.")

  if (type == "punctual"){
    for (index in 1:length(set)) {
      set[[index]] = (set[[index]] - mean(set[[index]])) / sd(set[[index]])
    }
    return(set)
  }
  else return ((set - min(set)) / (max(set) - min(set)))
}
```

Figura 4.9: Opción check para realizar el chequeo del paquete.

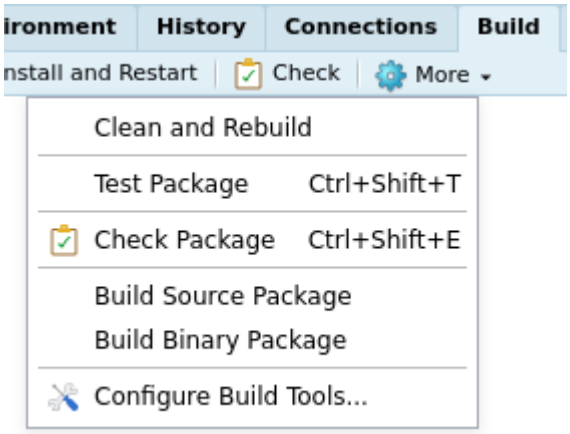


Figura 4.10: Opción Build Source Package.

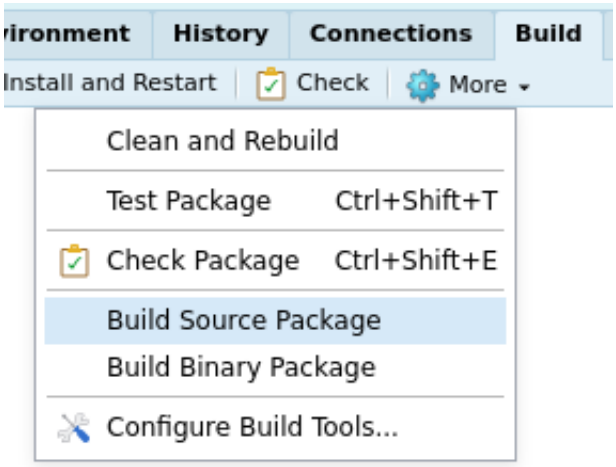


Figura 4.11: Conjunto de datos de entrenamiento.

PesoFresco	CantPapas	DMP	Densidad
120	8	0.7777778	1
1235	59	7.8333333	3
830	40	6.3333333	2
1315	42	7.0000000	3
315	15	2.5000000	3
1015	32	5.7777778	1
460	15	2.3888889	2
65	5	0.7222222	3
450	21	2.2777778	2
635	23	3.6111111	2
260	20	2.0000000	3
80	11	0.7222222	2
645	21	3.1666667	2
115	7	0.6111111	2
185	12	1.1111111	1
770	18	3.3333333	3
515	21	3.3888889	1
815	31	5.0555556	1
1240	49	8.0555556	1
365	17	2.6111111	3
715	26	4.4444444	3
40	8	0.7777778	2

Figura 4.12: Codificación del algoritmo de clasificación usando *apnnClassifier*.

```
completeData = data.frame(read_excel("~/Documents/neuronalNetworkRPackageInvestigation/datagnel.xlsx",

tdata <- completeData[3:5]
tdata <- standardize(tdata)
tdata$Densidad <- completeData[2]$Densidad
tdata$Densidad = as.factor(tdata$Densidad)
smp_size <- floor(0.75 * nrow(tdata))
set.seed(123)
train_ind <- sample(seq_len(nrow(tdata)), size = smp_size)
train_set <- tdata[train_ind, ]
test_set <- data.matrix(tdata[-train_ind, 2:3])

pnn <- trainNeuralNet(train_set = train_set, test_set = test_set, category_column = 4)
pnn <- evaluate(pnn)
View(pnn)
```

Figura 4.13: Gráfico de nubes de puntos.

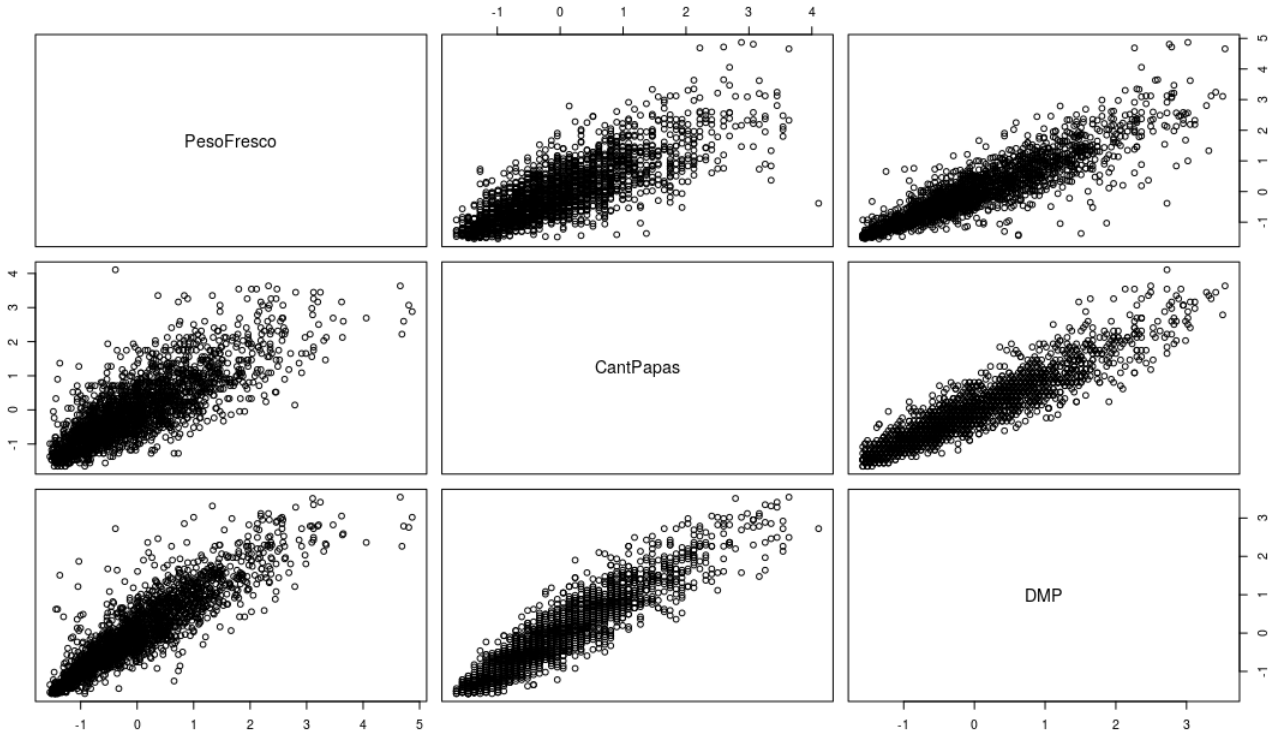
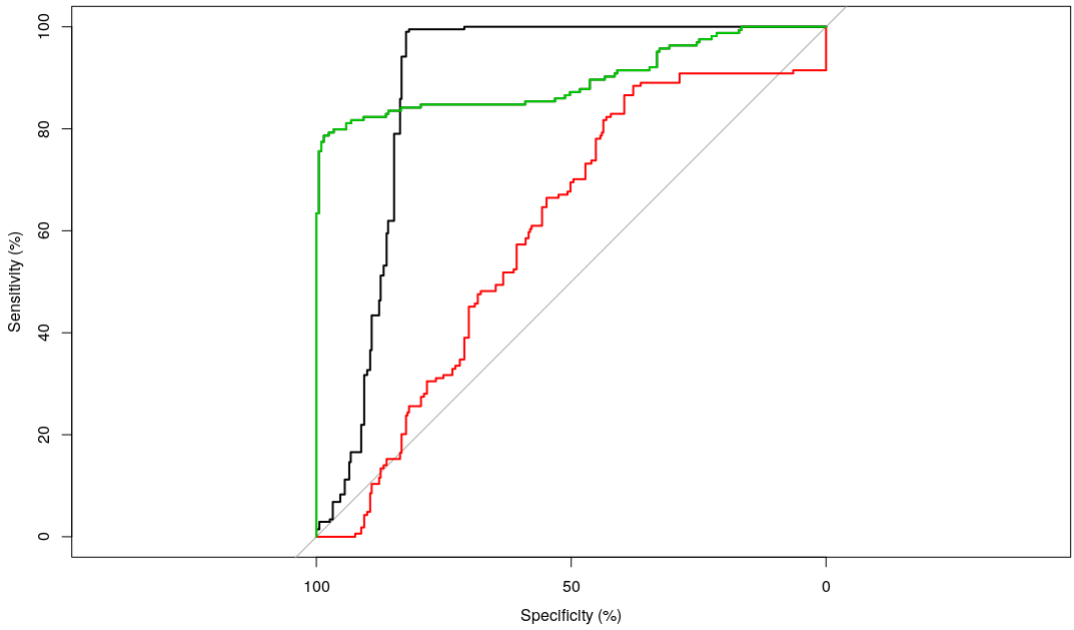


Figura 4.14: Gráfico de curva característica operativa de receptor.



Capítulo 5

Conclusiones y recomendaciones

Se diseñaron las soluciones algorítmicas para el entrenamiento de redes neuronales probabilísticas, lográndose ajustar a cualquier tipo de clasificación sin sesgar los algoritmos al objetivo general que es la clasificación de tubérculos de papas, esto con el fin de que el paquete pueda ser usado independiente del tipo de datos que se tengan para clasificar. Se observó que el algoritmo funciona de manera mucho más rápida cuando se conoce o se prueban manualmente valores para el punto óptimo de la función de activación, ya que como cualquier clasificador, para un set de datos grande los tiempos calculando este punto podrían elevarse considerablemente.

Se codificaron tres funciones que permiten resolver los algoritmos y la problemática planteada. Estas funciones son parametrizables, permitiendo el entrenamiento de redes neuronales probabilísticas para clasificar cualquier tipo de datos y en tantas clases como se requiera en otras investigaciones independientemente de su complejidad o cantidad de datos.

Se realizaron pruebas del paquete comenzando por clasificaciones sencillas con dos entradas y dos clases hasta una cantidad de cinco entradas y tres clases. Determinando así que se realizó un buen clasificador con buenos resultados, pues se evidencian buenas áreas bajo la curva en los gráficos de curvas características.

Se concluyó basado en los estudios realizados por Bernal en 2017, que las variables que más afectan la densidad de siembra en los cultivos de papa criolla (Solanum phureja), son el peso fresco, la cantidad de papas por planta y el diámetro medio ponderado. Se recomienda realizar pruebas con otras variables a fin de validar si existen otras combinaciones de parámetros de entrada que brinden una mejor clasificación.

Con respecto al análisis realizado con el paquete en la clasificación realizada para los datos de tubérculos de papa realizado por Bernal, 2017, no se obtuvo una buena clasificación, debido a dos razones, los datos fueron tomados mediante conteos afectando la clasificación bajo los supuestos de muestreo, además se puede concluir que aunque exista correlación entre las variables que fueron usadas como entrada al algoritmo (peso fresco, diámetro medio ponderado y cantidad de papas) no son buenos clasificadores para la densidad de siembra. Se recomienda trabajar los datos de entrada estadísticamente, eliminando datos atípicos y probar otros sistemas de muestreo en la recolección de datos con el fin de obtener mejores clasificaciones.

Con respecto al desarrollo de software, bajo el paradigma del desarrollo de paquetes en lenguaje R, apnnClassifier es fácilmente escalable, dado que la red neuronal probabilística puede ser parametrizable, debido a que es de código abierto, puede anexarse en nuevas versiones a la librería de funciones del paquete.

Como recomendación, se puede evaluar el desarrollo de un ambiente para gráficos más interactivo con el usuario o en plataforma WEB, elementos que se encontraban fuera del alcance de este proyecto de investigación.

Referencias Bibliográficas

CORPOICA (2009). Siembras en Colombia. Disponible en:http://www.corpoica.org.co. Consultada Octubre 2018.

Piñeros, C . (2009). «RECOPILACIÓN DE LA INVESTIGACIÓN DEL SISTEMA PRODUCTIVO PAPA CRIOLLA». Convenio SADE 045/06.

CRAN Packages pnn. Disponible en:https://cran.r-project.org/web/packages/pnn/index.html. Consultada Junio 2019.

CRAN Packages roxygen2. Disponible en:https://cran.r-project.org/web/packages/roxygen2/index.html. Consultada Junio 2019.

CRAN Packages dplyr. Disponible en:https://cran.r-project.org/web/packages/dplyr/index.html. Consultada Junio 2019.

CRAN Packages pROC. Disponible en:https://cran.r-project.org/web/packages/pROC/index.html. Consultada Junio 2019.

The R Project for Statistical Computing. Disponible en:https://www.r-project.org/. Consultada Octubre 2018.

Lantz, B.(2015).«Machine Learning with R». Reino Unido: Packt Publishing Ltd.

Afshin, A., Abbaspour-Gilandeh, Y., Nooshyar, M., Afkari-Sayah, A. (2016). «Identifying Potato Varieties Using Machine Vision and Artificial Neural Networks». International Journal of Food Properties, Vol 19(3), pp. 618-635.

Stephen, W., Forrest, B., Eric, X., Yu-Xuan, W., Yi-Fan, C., Qiao-Liang, X. (2007). «A Leaf Recognition Algorithm for Plant Classification Using Probabilistic Neural Network». IEEE International Symposium on Signal Processing and Information Technology.

Nur Badariah, A., Kumutha, A., Syed, A., Zainul, A. (2011). «Classification of Fruits using Probabilistic Neural Networks - Improvement using Color Features». TENCON IEEE Region 10 Conference.

Bernal, N., Darghan, AE., Rodríguez, LE. (2017). «Modelado del Calibre de tubérculos de papa Solanum phureja bajo diferentes densidades de siembra mediante regresión binomial negativa cero-inflada».

Acevedo, I., Velásquez, E. (2008).«Algunos conceptos de la econometría espacial y el análisis exploratorio de datos espaciales». Ecos de Economía, No. 27, pp. 9-34.

Wiley, J. (2016). «R Deep Learning Essentials». Reino Unido: Packt Publishing Ltd.

Arias, V., Bustos, P., Ñustéz, C. (1996). «Evaluación del Rendimiento en papa criolla (Solanum phureja) variedad "Yema de Huevo", bajo diferentes Densidades de Siembra en la Sabana de Bogotá.» Agronomía Colombiana, Vol. XIII, No 2, pp. 152-161.

Salomón, J., Estévez, A., Castillo, J., Cordero, M., Varela, M. (2009). «ESTUDIO DE LA COMPOSICIÓN DE CALIBRES EN VARIEDADES DE PAPA (Solanum tuberosum, L.) PARA LA PRODUCCIÓN NACIONAL DE TUBÉRCULOS-SEMILLA». Cultivos Tropicales, vol. 30, No. 1, pp. 69-72.

Cotes, J., Ñustez, C., Pachón, J. (2000). «EVALUACION DE LA DENSIDAD DE SIEMBRA Y EL TAMAÑO DEL TUBERCULO SEMILLA EN LA PRODUCCION DE SEMILLA BASICA DE PAPA CRIOLLA, VARIEDAD «YEMA DE HUEVO» (Solanum phureja Juz. et Buk.)». Agronomía Colombiana, No. 17, pp. 57-57.

Piñeros, C. (2009). «Recopilación de la Investigación del Sistema Productivo Papa Criolla». Secretaria de Agricultura y Desarrollo Economico (Gobernación de Cundinamarca), Federación Colombiana de Productos de Papa.

Buitrago, G., López, A., Coronado, A., Osorno, F. (2004). «Determinación de las características físicas y propiedades mecánicas de papa cultivada en Colombia». Revista Brasileira de Engenharia Agrícola e Ambiental, Vol.8, No.1, pp.102-110.

Ligarreto, G., Suárez, M. (2003). «EVALUACION DEL POTENCIAL DE LOS RECURSOS GENETICOS DE PAPA CRIOLLA (Solanum phureja) POR CALIDAD INDUSTRIAL». Agronomía Colombiana, Vol. 21, No. (1-2), pp. 83-94.

Patel, J., Seung-Kyum, C. (2012). «Classification approach for reliability-based topology optimization using probabilistic neural networks». Struct Multidisc Optim, No. 45, pp. 529–543.

Benavides, A. «Curvas ROC (Receiver-Operating-Characteristic) y sus aplicaciones». Universidad de Sevilla.