MEDICIÓN DE PARÁMETROS DE CALIDAD DE HARINA DE PESCADO USANDO IMÁGENES HIPERESPECTRALES E INTELIGENCIA ARTIFICIAL

Cesar Cherre-Pupuche

Piura, marzo de 2019

FACULTAD DE INGENIERÍA

Departamento de Ingeniería Mecánico-Eléctrica

Cherre, C. (2019). *Medición de parámetros de calidad de harina de pescado usando imágenes hiperespectrales e inteligencia artificial* (Tesis para optar el título de Ingeniero Mecánico-Eléctrico). Universidad de Piura. Facultad de Ingeniería. Programa Académico de Ingeniería Mecánico-Eléctrica. Piura, Perú.

U N I V E R S I D A D D E P I U R A

FACULTAD DE INGENIERÍA

“Medición de parámetros de calidad de harina de pescado usando imágenes hiperespectrales e inteligencia artificial”

Tesis para optar el Título de

Ingeniero Mecánico Eléctrico

CESAR VENADIO CHERRE PUPUCHE

Asesor: Mgtr. Juan Carlos Soto Bohórquez

Piura, Marzo de 2019

A Dios y mis queridos padres
Cesar Cherre y Janet Pupuche
por su apoyo incondicional.

Prólogo

La industria pesquera es una de las principales actividades para la economía peruana actual. Los procesos de producción de harina de pescado involucran un control de calidad. Determinar los parámetros de fisicoquímicos para harina de pescado es una tarea realizada por equipos estacionarios de laboratorio o a través de métodos químicos.

El procedimiento que propone esta tesis, es poder medir los parámetros fisicoquímicos mas relevantes del proceso productivo, de manera no destructiva y de tiempo de adquisición bajo. Los parámetros a correlacionar son: proteína, contenido de humedad, grasas y cenizas. La obtención de estos parámetros facilita y agiliza los procedimientos de control de calidad y permite tomar acciones para asegurar la mejor calidad posible.

Esta tesis también tiene como objetivo formular un procedimiento usando análisis de señales e imágenes hiperespectrales, reducción de dimensionalidad y algoritmos de inteligencia artificial para identificar sistemas de predicción de parámetros fisicoquímico. Este procedimiento puede ser extensible y escalable para diferentes sistemas de identificación, para productos como: harinas de cereales, frutas, verduras, carnes, pescados y otros. Basado en los resultados experimentales, es posible la implementación futura de un sistema en línea y de medición en tiempo real.

Es importante mencionar que todas las muestras de harina de pescado, así como los datos de referencia fueron proporcionados por una empresa dedicado a la produccion de harina y aceite de pescado, planta ubicada en Piura, Perú. Fue posible conocer parte del proceso de producción de harina de pescado, también, se proporcionó información sobre los mecanismos actuales de obtención de parámetros de calidad así como los instrumentos utilizados.

Resumen

En la presente tesis se establece un procedimiento para la identificación de parámetros fisicoquímicos de harina de pescado, aplicando técnicas de visión artificial para imágenes hiperespectrales, procesamiento de señales, modelos de predicción basados en aprendizaje automático supervisado. Estos procedimientos se aplicaron en la salida del proceso productivo de elaboración de harina de pescado.

El primer capítulo, describe estudios sobre investigaciones previas, también se incluyen los sistemas actuales de adquisición de parámetros fisicoquímicos por métodos no invasivos.

En el segundo y tercer capítulo, se estudian los conceptos teóricos acerca de técnicas de reducción de dimensionalidad, procesamiento de imágenes hiperespectrales y modelos de aprendizaje automático.

En el cuarto capítulo, presenta los frameworks utilizados y los resultados obtenidos por el sistema de predicción de parámetros.

Por último, el quinto capítulo, presenta las conclusiones y recomendaciones de la presente tesis.

Índice

Prólogo iii
Resumen iv
Índice v
Introducción 1
Capítulo 1 3
Antecedentes 3
1.1 Estado del arte 3
1.2 Instrumentación actual 7
1.2.1 DA 7250 7
1.2.2 F 750 Quality meter 8
Capítulo 2 11
Estudios teóricos 11
2.1 Procesamiento de imágenes hiperespectrales 11
2.1.1 Preprocesamiento de firma espectral 11
2.2 Sistemas de aprendizaje automático 12
2.2.1 Aprendizaje automático no supervisado 13
2.2.2 Aprendizaje automático supervisado 13
2.2.3 Aprendizaje automático mixto 15
2.3 Reducción de dimensionalidad y transformada Wavelet 15
2.3.1 Metodologías de reducción 16
2.3.2 Transformada Wavelet 17
2.3.3 Validación de reducción 20
Capítulo 3 23
Sistemas de aprendizaje automático 23
3.1 Support Vector Regression 23

3.1.1 Regresión Lineal 23
3.1.2 Regresión No Lineales 25
3.2 Neural Networks 27
3.2.1 Entradas 28
3.2.2 Función de activación 28
3.2.3 Salidas 29
3.2.4 Topología de redes neurales 29
3.2.5 Tipos de redes neuronales artificiales 30
3.2.6 Multilayer Perceptron 32
3.3 Validación cruzada y error de predicción 34
3.3.1 Coeficiente de correlación 35
3.3.2 Normalidad de errores 35
Capítulo 4 37
Desarrollo y experimentación 37
4.1 Desarrollo de software 37
4.2 Reducción de firma espectral 37
4.3 Parámetros de sintonización de modelos de regresión 39
4.3.1 Support vector regression 40
4.3.2 Multilayer perceptron regression 49
4.4 Comprobaciones estadísticas 58
4.4.1 Support vector regression 58
4.4.2 Multilayer perceptron regression 66
4.5 Validación de modelo 74
4.5.1 Prueba por coeficiente de correlación 74
4.5.2 Verificación estadística 75
Capítulo 5 77
Conclusiones y recomendaciones 77
5.1 Conclusiones 77
5.2 Recomendaciones 78
Bibliografía 81
Anexos 83
Anexo A Códigos para identificación de modelos usando SV 83
Anexo B Códigos para modelos de aprendizaje automático MLP bajo Tensorflow 92

Introducción

Actualmente la producción de harina de pescado involucra procesos de control de calidad, la función de estos procesos consiste en medir parámetros fisicoquímicos con fin de obtener métricas sobre la producción en tiempo real. Conocer el proceso en tiempo real permite tomar acciones sobre el proceso, esto a su vez se ve reflejado en un producto de mayor calidad y competitivo para el mercado internacional de harina de pescado.

Los sistemas de medición de parámetros fisicoquímicos en línea y en tiempo real toman gran relevancia para estos procesos, sin embargo, actualmente no se cuenta con equipos comerciales que realicen esta tarea. La presente tesis presenta un modelo de identificación basado en inteligencia artificial usando imágenes hiperespectrales para medición de parámetros fisicoquímicos.

Gracias a las capacidades computacionales modernas es posible procesar información proveniente de imágenes hiperespectrales, utilizar herramientas de inteligencia artificial, abstraer conocimiento en modelos numéricos y con estos poder estimar los parámetros fisicoquímicos. En esta tesis se describen todos los procedimientos teóricos y prácticos para obtener modelos de predicción.

Se adjuntan todos los códigos usados en el procesamiento de información. Es posible replicar estos modelos para diferentes productos, planteando así, un modelo escalable y de gran aporte para el control de calidad de procesos productivos para múltiples sectores.

Capítulo 1
Antecedentes

1.1 Estado del arte

1. El uso de las imágenes hiperespectrales tiene una gran utilidad en el análisis no destructivo para predecir múltiples parámetros de calidad de harina maíz usando modelos de redes neuronales artificiales Multilayer Perceptron (Mutlu, y otros, 2011).

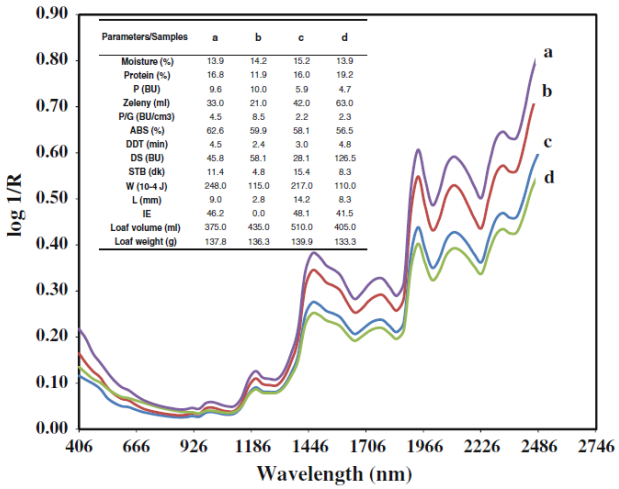

Figura 1 Grafico de firmas de muestras aleatorias
Fuente: Mutlu, y otros, 2011.

Este estudio afirma que, tiene capacidad de predicción para 14 parámetros (proteína, humedad, zeleny[1], entre otros) de manera rápida y eficaz. También, afirma una gran escalabilidad de esta metodología propuesta y la reducción de costos a diferencia del uso de métodos invasivos.

En la figura 1 se muestran las firmas espectrales para diferentes muestras de harina de trigo capturadas por un instrumento hiperespectral desde 406 nm hasta 2746 nm.

[1] También llamado índice de sedimentación es un indicador de calidad determina el numero de partículas sedimentadas generadas por proteínas hinchadas por absorción de agua.

2. Se tiene éxito con el uso de las técnicas de procesamiento de imágenes hiperespectrales al evaluar calidad de granos de cacao permitiendo correlacionar el índice de antocianina AR12[2] a través del análisis de índices espectrales (Mundaca, Soto, & Ipanaqué, 2015).

   La figura 2 presenta una comparativa de una imagen en el espectro NIR[3] versus una imagen en RGB.

Figura 2 Captura de imagen hiperespectral a diferentes bandas (NIR vs. RGB)
Fuente: Mundaca, Soto, & Ipanaqué, 2015.

3. En (Sun, 2015) se describe los procedimientos para desarrollar sistemas de medición usando procesamiento de imágenes, procesamiento de señales, técnicas estadísticas e inteligencia artificial. En la sección A de la figura 3 se muestra el procedimiento de adquisición de información espectral, en la sección B se aprecia una imagen de filete de carne capturado a diferentes bandas. Las aplicaciones presentan técnicas de regresión, clasificación y agrupamiento. En esta tesis se toma importancia las técnicas utilizadas en el procesamiento de imágenes hiperespectrales orientadas para el control de calidad de productos marinos.

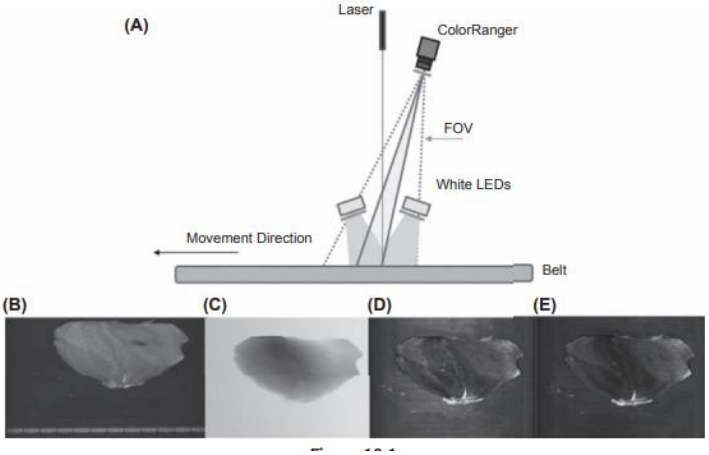

Figura 3 Captura de imágenes hiperespectrales de filetes de pescados
Fuente: Sun, Computer Vision Technology for Food Quality Evaluation, 2015.

[2] Se refiere al contenido de pigmentos hidrosolubles influyentes en la coloración de productos de origen vegetal.
[3] Espectroscopia del infrarrojo cercano, tiene como finalidad medir la interacción de la luz con materia y de esta manera correlacionar radiación electromagnética con índices fisicoquímicos.

4. En (Dai, Cheng, Sun, Zhu, & Pu, 2016) presenta a la transformada wavelet como método de extracción de características y como métodos de regresión presenta a las técnicas de soporte vectorial, con el objetivo de determinar el TBVN [4] para langostinos. En la figura 4 se aprecia la distribución de TBVN para langostinos a través de mapas de calor.

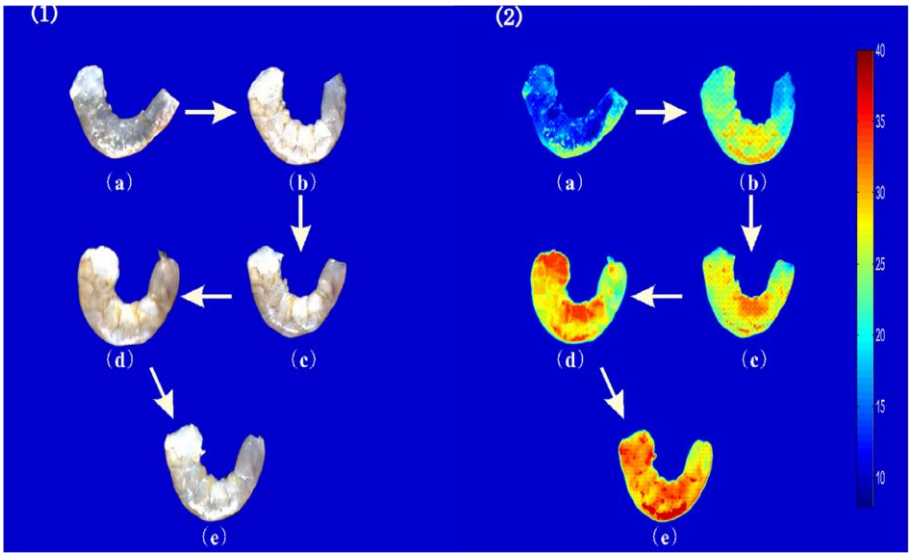

Figura 4 Mapa de distribución de TBVN para langostino
Fuente: Dai, Cheng, Sun, Zhu, & Pu, 2016.

5. En (Sun, Hyperspectral Imaging for Food Quality Analysis and Control, 2010) presenta las aplicaciones de procesamiento de cubos hiperespectrales para clasificar productos comestibles como carnes, pescados, frutas y otros alimentos. Esta clasificación se basa en los parámetros fisicoquímicos.
   La figura 5 presenta imágenes capturadas a distintas bandas para evaluar un sistema de clasificación de pescados basado en frescura, se referencian los pixeles de color rojo como pixeles "no frescos", con el fin de generar mapas de frescura para toda la superficie del pescado.

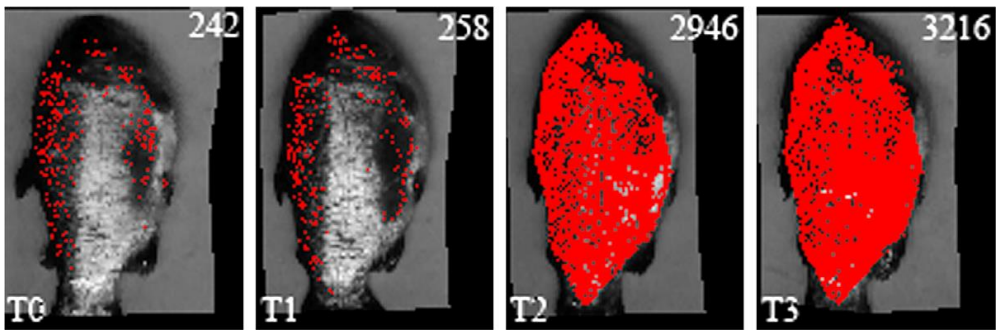

Figura 5 Barrido de imágenes hiperespectrales para clasificación de pescados
Fuente: Sun, Hyperspectral Imaging for Food Quality Analysis and Control, 2010.

[4] Total de bases volátiles nitrogenadas, es un indicador de frescura y calidad de productos marino caracterizado por determinar los componentes volátiles que provocan degradación del producto.

6. En (Fernandes, y otros, 2015) se determina el contenido de PH, grados Brix[5] y antocianina aplicado a uvas usando diferentes algoritmos como redes neuronales, índices espectrales y mínimos parciales cuadrados aplicado a firmas espectrales.

   La figura 6 presenta un gráfico de correlación de valores deseados versus valores obtenidos por el sistema de regresión usando redes neuronales, así mismo presenta índices cuantitativos para determinar la calidad de predicción.

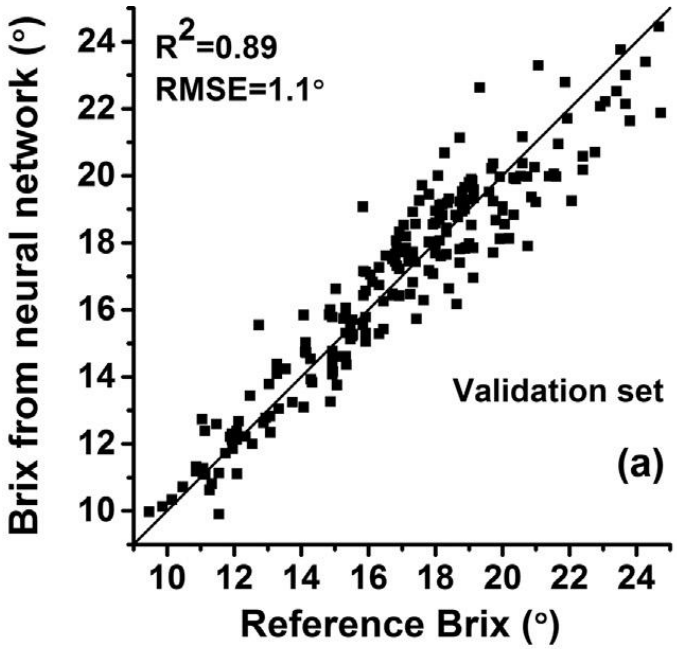

Figura 6 Grafico de validación cruzada para predicción de grados Brix
Fuente: Fernandes, y otros, 2015.

7. (United States Patente nº US 9,176,110 B1, 2015) Plantea un modelo usando indicies espectrales para determinar la concentración de histamina para pescados de manera no destructiva a través del análisis de imágenes hiperespectrales. En la figura 7 se observa diferentes firmas espectrales con sus respectivos valores de histamina.

   Esta patente determina un índice espectral, el cual puede ser correlacionado directamente con el valor de histamina usando muestras de entrenamiento.

[5] Índice para determinar el contenido de contenido de azucares en productos.

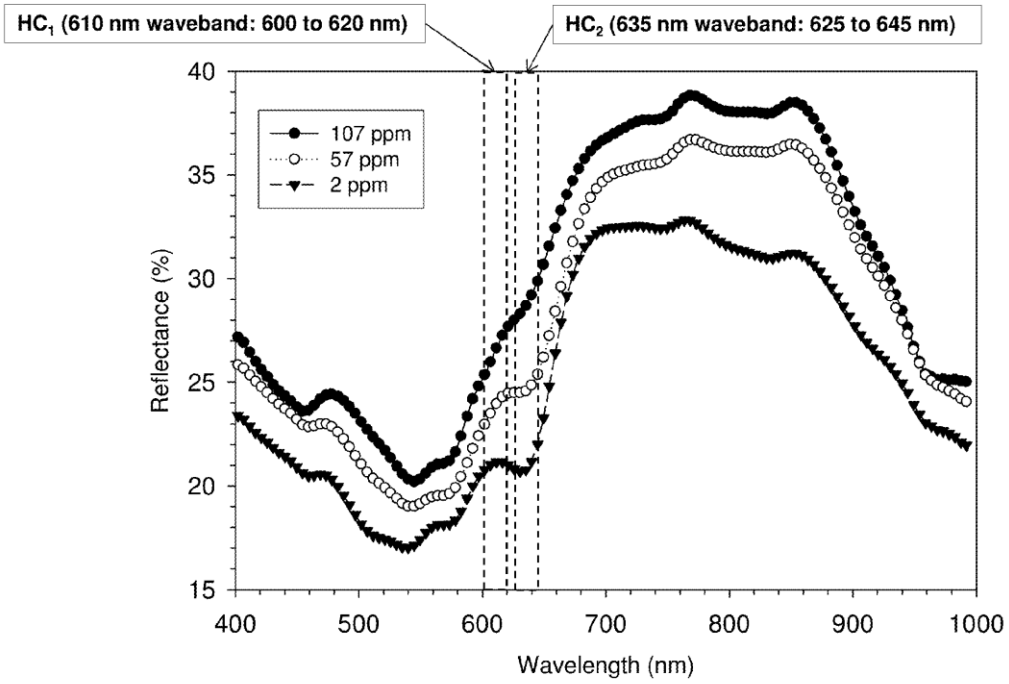

Figura 7 Identificación de índices espectrales
Fuente: United States Patente nº US 9,176,110 B1, 2015.

1.2 Instrumentación actual

El levantamiento de información se realizó en la parte final del proceso productivo de elaboración de harina de pescado. En la información obtenida sobre el proceso de medición de los parámetros de calidad de harina de pescado, se observó que, actualmente, se utiliza un dispositivo de laboratorio de medición por espectroscopia llamado NIR DA 7250.

También, se toma en consideración los sistemas de medición portátiles usados en la industria agrícola para medición de parámetros fenológicos, fisicoquímicos y otros. Estos instrumentos son llamados Food Science Instruments.

1.2.1 DA 7250

El DA 7250 es un analizador NIR de la familia "Perten Instruments" presentado en la figura 8, utilizado para analizar múltiples variables para diversos productos de modo no destructivo. Este dispositivo es capaz de obtener resultados en 6 segundos gracias al software de identificación que incluye. Su tecnología está basada en espectrómetros, lo cuales extraen múltiples firmas o huellas espectrales que posteriormente son procesadas por el software con el objetivo de entregar la información solicitada a través de un display o exportada en distintos formatos. El rango de bandas en las que opera es de 900 nm a 1700 nm con una precisión menor a 0.3 nm, un espacio de pixel de 3.1nm/pixel y una resolución óptica de aproximadamente 7nm.

El dispositivo está disponible para analizar diferentes productos como cereales, granos, pastas, semillas, harinas y otros. Entre los parámetros que puede identificar se encuentran proteína, grasa, almidón, fibra, cenizas y otros.

El software utiliza algoritmos de regresiones como PLS (Mínimos Cuadrados Parciales), NN (redes neuronales artificiales), HR (Regresión Honigs) y análisis discriminante. La interfaz gráfica permite configurar y calibrar los algoritmos de regresión para obtener la mejor precisión posible (Perten Intruments, 2018).

Figura 8 Analizador NIR DA 7250
Fuente: Perten Intruments, 2018.

1.2.2 F 750 Quality meter

Este dispositivo portátil presentado en la figura 9 permite conocer parámetros fisicoquímicos de manera no invasiva en un corto periodo de tiempo, basado en tecnología NIR, permite estimar parámetros tanto para materia sólida como solidos solubles. Entre las aplicaciones que desarrolla se encuentran los análisis de calidad para exportación de frutas y verduras; determina la madurez de productos agrícolas; estima el tiempo de cosecha y entre otras.

Los productos que pueden ser identificados por este instrumento son manzanas, paltas, mandarinas, uvas, kiwi, mango, peras y otros. Gracias al software, incorpora módulos de entrenamiento y calibración para obtener métricas más precisas.

El espectrómetro utilizado es un Carl Zeiss MMS-1 Spectrometer con rango de bandas de 310 nm a 1100nm con resolución espectral de entre 8 a 13 nm, cuenta con sistema de geo localización y almacenamiento de información de 32GB (Instruments, 2018).

Figura 9 Analizador NIR Fruit Scan F 750
Fuente: Felix Instruments, 2018.

Capítulo 2
Estudios teóricos

2.1 Procesamiento de imágenes hiperespectrales

Los sistemas de adquisición de imágenes hiperespectrales son instrumentos que se encargan de convertir la energía reflejada de un entorno especifico, las cuales pasan etapas analógicas y digitales de filtrado y acondicionamiento de señales, los cuales se traducen en estructura de datos en una matriz o arreglo, estos datos se procesan para obtener caracterizas y modelos, con el fin de identificar y predecir diferentes capas de información.

2.1.1 Preprocesamiento de firma espectral

El paso fundamental es identificar la región específica a estudiar, llamado Región de Interés (ROI), por esto, el procedimiento fundamental es la segmentación o separación de ROI en la muestra de harina de pescado, entre las técnicas se encuentran la segmentación por bordes, gradientes, entre otros.

Para este caso de estudio en específico, al contar con una muestra homogénea de harina de pescado, solo es necesario realizar un redimensionamiento de la imagen original, es decir eliminar partes de la imagen con tamaños fijos.

Algunas veces es necesario evaluar alguna etapa de filtrado en la imagen hiperespectral, entre los filtros más comunes se encuentran: filtros medianos adaptativos, filtros de reducción de ruidos, filtros gaussianos. Para el análisis de imágenes de muestras de harina de pescado, en esta tesis, no se aplicó ningún tipo de filtro en la imagen hiperespectral (Sun, 2015).

En términos computacionales, una imagen hiperespectral es un arreglo de 3 dimensiones, es decir un cubo de datos de (NxMxP), donde N y M se refiere a la resolución espacial, el tamaño de la imagen y P a la resolución espectral, el número de bandas del espectro electromagnético.

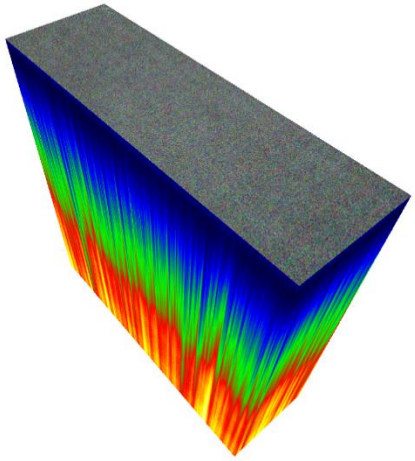

Figura 10 Representación de cubo hiperespectral
Fuente: Elaboración propia.

Se aprecia en la figura 10 los datos numéricos de reflectancia son representados a partir de un mapa de colores de 8 bits, es decir, cada valor de reflectancia es renderizado por un color con el objetivo de tener una idea visual acerca de un cubo hiperespectral.

El área de interés tiene dimensiones de 100 pixeles de largo y 100 pixeles de ancho, ubicado en el centro de la imagen hiperespectral. Esta firma promedio se calcula con el objetivo de adquirir una firma que pueda generalizar un comportamiento de la muestra, este grupo de datos es el punto de partida para el diseño de los modelos de identificación.

2.2 Sistemas de aprendizaje automático

Los sistemas que utilizan inteligencia artificial están compuestos de múltiples algoritmos cuyo objetivo es encontrar comportamiento generalizado identificando patrones en banco de datos, con el fin de clasificar, predecir, identificar modelos y agrupándolos convenientemente. El proceso de aprendizaje automático toma importancia cuando no se tiene un modelo matemático o es muy complicado de modelar por métodos de identificación. Este proceso tiene como objetivo encontrar un modelo a partir de información pasada, datos de entrada y salida; buscar una relación entre ellos y proporcionar información suficiente para validar la confiabilidad de las predicciones a través de experimentos.

Dependiendo de las fuentes de datos (instrumentación, bases de datos, imágenes, etc), es necesario realizar procesos de reducción de dimensiones o extracción de características, para diseñar un algoritmo de aprendizaje.

2.2.1 Aprendizaje automático no supervisado

El aprendizaje automático no supervisado tiene como objetivo identificar patrones entre un conjunto de datos de entrada X de (m x n), evaluar la frecuencia y la interactividad de los patrones, de esta manera generalizar un patrón base, con el fin de predecir nuevos comportamiento o tendencias. (Shobha & Rangaswamy, 2018)|

Los algoritmos de aprendizaje no supervisado tienden a buscar patrones potencialmente útiles, eligiendo un grupo de datos originales, estos datos, deben poseer ciertas características con el fin de extraer características útiles para mejorar la precisión de las predicciones, evitando redundancias que podrían desviar la calidad de predicción, agrupamiento y clasificación (Brodley & Dy, 2004).

2.2.2 Aprendizaje automático supervisado

El aprendizaje supervisado consta en generalizar modelos (clasificación, agrupamiento, regresión) donde se cuenta con información de entrada X e información de salida Y, esta última actúa de supervisor entrenar e identificar el modelo. Una vez identificado el modelo, es posible predecir información $\hat{Y}$ frente nueva información $\hat{X}$.

Si la naturaleza de Y es discreta, se afirma que el sistema de aprendizaje automático es un sistema de clasificación supervisada.

Si la naturaleza de Y es vector o matriz de valores numéricos, se afirma que el sistema de aprendizaje automático es un modelo de regresión supervisada. Sobre estos algoritmos se plantea el desarrollo del estudio de los modelos para predecir los parámetros de físico químicos de harina de pescado.

Los modelos de regresiones se basan en poder obtener un modelo donde se analiza históricos de múltiples variables relacionadas, y con ello efectuar predicciones numéricas. Es importante tener criterios de selección y construcción de modelos, entre los criterios más importantes se tiene la complejidad de los sistemas a predecir, el tiempo de cálculo de predicción, la efectividad del modelo de regresión.

La finalidad de una regresión es ofrecer que tenga el mejor ajuste entre las variables relacionables, y con ella cuantificar nuevos valores. Existen múltiples naturalezas de ecuaciones de regresión, en este apartado se centrará en las ecuaciones de tipo lineales.

Basado en la ecuación de la recta

$$Y_i = B_0 + B_1 X_i + \varepsilon \tag{2.1}$$

Donde Y_i representa los valores de salida del modelo de regresión, B_0 es desplazamiento de la recta ajustada, B_1 la pendiente de la recta o la relación proporcional entre la variable de entrada y salida, X_i se refiere a los valores de entrada y ε representa el error aleatorio propio del ajuste. Al estudiar ε se determina que tan dependiente o independiente son las variables por relacionar. (Montgomerz, Peck, & Geoffrey, 2012)

En el caso ideal, se trata de que la media de los errores de la regresión sea 0 y que los errores cumplan una distribución normal.
Para el cálculo de recta de ajuste se tiene:

$$Y_i = [y_1, y_2, y_3, \dots, y_n] \tag{2.2}$$

$$X_i = [x_1, x_2, x_3, \dots, x_n] \tag{2.3}$$

Entre los métodos más usados de estimación de parámetros de regresiones lineales se encuentra el método de mínimos cuadrados, el cual obtiene los coeficientes de la recta de regresión de la siguiente manera:

$$B_1 = \frac{SS_{xy}}{SS_{xx}} \tag{2.4}$$

$$B_0 = \bar{y} - B_1\bar{x} \tag{2.5}$$

Donde:

$$SS_{xy} = \sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) \tag{2.6}$$

$$SS_{xy} = \sum_{i=1}^{n}(x_i y_i - n\bar{x}\bar{y}) \tag{2.7}$$

$$SS_{xx} = \sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{2.8}$$

$$SS_{xx} = \sum_{i=1}^{n} {x_i}^2 - n\bar{x}^2 \tag{2.9}$$

Usando dichas ecuaciones se construye la recta de ajuste de regresión, en adelante a partir de esta ecuación se pueden realizar predicciones a partir de datos de entrada.

2.2.2.1 Regresiones Múltiples

En muchas ocasiones de aplicaciones reales es necesario contar con información multivariable para predecir y relacionar la salida con los datos de entrada, también es necesario utilizar funciones más complejas para predecir valores discretos con mayor exactitud y precisión, estas funciones van desde polinomiales hasta complejas redes neuronales.

Para identificar los posibles tipos de regresiones es necesario estudiar y analizar los datos de entrada, a partir de este punto empieza el análisis para determinar el mejor tipo de regresión.

Se evalúa la dimensión de los datos, puede ser escalares (valores numéricos), vectores (señales de instrumentación), matrices (imágenes monocromáticas), cubos (imágenes hiperespectrales). Esto es importante por 2 aspectos fundamentales, el primero es para contemplar las dimensiones de los datos de entrada en el diseño de la regresión y el segundo es para considerar los tiempos de ejecución.

Es indispensable tener en cuenta la complejidad del modelo, para ello se toman en cuenta estudios previos, y así poder terminar las opciones más viables.

Para el presente estudio de los sistemas de regresiones se plantean 2 tipos de regresiones no lineales, usando diferentes metodologías heredadas del aprendizaje automático supervisado.

2.2.3 Aprendizaje automático mixto

El aprendizaje automático mixto o también llamado aprendizaje semisupervisado se refiere a sistemas cuentan con gran volumen de datos, de los cuales solo una pequeña parte están correlacionado con datos de salida y la mayoría de los datos no poseen ningún tipo de etiquetado, es ahí donde el sistema semisupervisado buscar encontrar interacción entre estos dos grupos de datos (Wang & Shen, 2007).

Estos sistemas utilizan procesos iterativos con el fin de obtener nuevas etiquetas para los datos no correlacionados, evalúan la calidad de etiquetado usando predicciones, retroalimentando la calidad de predicción cambiando las prioridades de generalización de modelo usando datos etiquetados.

Este algoritmo posee básicamente 2 tipos de aprendizaje, transductivo e inductivo; el primero se refiere predecir etiquetas nuevas para los datos no etiquetado, mientas que el aprendizaje inductivo se encarga de predecir nuevos datos partir del conocimiento adquirido anteriormente (Triguero, García, & Herrera, 2013).

2.3 Reducción de dimensionalidad y transformada Wavelet

Basado en la hipótesis principal en la que se afirma que es posible modelar la señal adquirida por medio de una imagen hiperespectral, generalizar modelos de forma ágil y poder implementar algoritmos de cuantificación. Para todo esto, es importante tener en cuenta los tiempos de cálculo, lo cual dirige a buscar posible soluciones y estrategias que permitan disminuir la alta dimensionalidad implícita en una imagen hiperespectral.

Existen variaciones no deseadas resultantes de la física, interacciones ambientales, errores en los instrumentos durante la adquisición de datos, en algunos casos puede disminuir la robustez y precisión de la predicción del modelo. (Pu, Su, Ma, Liu, & Cheng, 2014)

En múltiples ocasiones, los modelos que cuentan con un número específico de características o variables de observación, la presión y eficiencia de los algoritmos tiende a disminuir. Esto se da principalmente cuando dichas características no ofrecen información relevante al sistema. Entre las posibles causas de estos problemas se encuentran la degradación de los datos por ruidos, disturbios, inadecuada calibración y excesiva dimensionalidad.

Si se mantiene ambientes controlados durante la recolección de datos y una adecuada calibración, se puede obtener datos originales de calidad aceptable; sin embargo, los problemas de alta dimensionalidad pueden seguir presente. Una de las consecuencias de la excesiva dimensionalidad es el deterioro de la capacidad de generalización del modelo, lo que causa problemas como el *subentrenamiento* o *sobreentrenamiento*.

2.3.1 Metodologías de reducción

Los métodos de reducción de dimensionalidad se clasifican en dos: selección de características y extracción de características. El primer método pretende una selección de un subconjunto de características provenientes de los datos originales basado en test estadísticos, filtrado de datos y métodos envolventes. Para la extracción de características, se propone el uso de un conjunto de datos extraídos a partir de las transformaciones de la data original. La aplicación de este segundo método toma relevancia cuando el número de variables o tamaño de datos es elevado y/o cuando existe una alta correlación presenta información redundante, lo que produce un alto nivel de *multicolienalidad*[6] (Rodríguez, 2016)

Por las razones expuestas anteriormente, el procesamiento de los datos originales para minimizar variaciones no deseadas es necesario para un robusto sistema de predicción. Recientemente, la transformada wavelet ha sido aplicada en la separación de bandas superpuestas, eliminación de ruido, suavizado, corrección de línea de base y eliminación de multicolinealidad. (Gributs & Burns, 2006)

2.3.1.1 PCA

El algoritmo de análisis de componentes principales o “PCA” tiene como objetivo seleccionar características útiles, reducir la dimensión con respecto a la dimensión de la data original, todo esto sin perder información relevante.

A partir de un grupo de datos de entrada X de dimensión (m), se pretende representarlos a través de $\hat{X}$ de dimensión (p), donde generalmente p < m, esto realiza por medio de combinaciones lineales con el objetivo deshacer datos dependientes unos de otros, es decir, minimizar la multicolinealidad (Rahiala, 2011).

Este algoritmo es uno de los más usados en la reducción de dimensionalidad, sin embargo, para este caso de estudio se utilizará otra metodología conocida como Wavelets.

[6]Término que habitualmente se emplea en campos como la econometría o la regresión multivariante, y que indica la existencia de interrelaciones entre variables explicativas de un determinado modelo, lo que puede ocasionar coeficientes de regresión poco estables, de gran variabilidad, e intervalos de confianza más amplios.

2.3.1.2 Wavelets

Los wavelets se refiere a un conjunto de herramientas matemáticas y algoritmos; los cuales permiten procesar de forma precisa, robusta y eficiente gran variedad de señales. Además, es muy utilizada en aplicaciones de ciencias como la Física, Análisis Numérico, Procesamiento de Señales, Estadística y Probabilidad. Esta transformada es usada para extraer características de especial interés que están localizadas en una señal, esto se traduce como una metodología de procesamiento que permite extraer características de interés para contextos de clasificación y/o regresión. Existen varias técnicas para extraer características, una de ellas es usar la técnica del filtrado para la atenuación de las componentes de alta frecuencia presente en los datos. Otra técnica es represar la información de manera reducida, ya que esta modalidad es ideal para la compresión de datos.

2.3.2 Transformada Wavelet

Se afirma que la Transformada Wavelet (TW) es una evolución de la transformada Fourier. La TW surge para fortalecer las debilidades de la Transformada de Fourier que posee en el dominio del tiempo. Entre las debilidades más relevantes se refiere a que la Transformada de Fourier (TF) fue ideada para señales estacionarias, periódicas e infinitas. En el dominio de la frecuencia no existe información disponible del tiempo; es imposible determinar que en el instante de tiempo en el que tiene lugar un evento particular a una frecuencia concreta. El análisis Wavelet surge como un método que permite representar, descomponer y reconstruir una señal, donde se puede presentar cambios muy significativos en sus componentes de tiempo y frecuencia en forma instantánea; gracias al análisis multiresolución con ventanas de longitud variable, que se adaptan al cambio de frecuencia.

Para la TF representan a las funciones en bases de senos y cosenos de permanencia infinita, mientras que la TW representa en base a funciones localizadas en frecuencia(dilatación) y en el tiempo(traslación).

2.3.2.1 Wavelet continua

La Transformada Wavelet Continua (CWT) se define como el análisis de una señal en una sección específica de esta, y es expresada mediante coeficientes del producto interno, entre la señal y la función Wavelet Madre (WM). La WM es una función localizada que contiene todas las funciones con energía finita y funciones de cuadrado integrable definidas (Nieto & Orozco, 2008).

$$\int |f(t)|^2 \, dt = E; E < \infty \tag{2.10}$$

La función WM viene representada por la siguiente expresión matemática:

$$\psi_{u,s}(t) = \frac{1}{\sqrt{s}} \, \psi \left(\frac{t-u}{s} \right) \tag{2.11}$$

Donde **s** se refiere al factor de escala y **u** se refiere al factor de traslación. Cuando s > 1 se afirma que las wavelets son dilatadas y cuando s < 1 son contraídas; es decir, variando el

factor s, se envuelven espectros de frecuencias. Además. el valor de s es inversamente proporcional al rango de frecuencia que abarca.

La CWT se expresa como la sumatoria para todos los instantes de tiempo de f(t) multiplicada por versiones escaladas y desplazadas de una WM.

$$W_f = \int_{-\infty}^{\infty} f(t)\psi_{u,s}(t)dt \tag{2.12}$$

Lo que representa la expresión 3.12 es la descomposición de la señal f(t) en interpretaciones de la mismas señales escaladas y desplazadas de la función base. Si se observa la metodología de la TF, la señal se descompone en funciones senoidales de diferentes frecuencias. Para la CWT, la función WM es análoga a las funciones senoidales.

A partir de los coeficientes wavelets (u, s) se puede reconstruir la señal original. Este concepto es interesante, ya que es la base de la compresión de señales con poca entropía, es decir, la señal original es muy similar a la señal reconstruida.

Se puede afirmar que un escalado pequeño, es decir, una wavelet comprimida es capaz de capturar información que se encuentran en altas frecuencias, mientras una wavelet extendida captura detalles que varían lentamente.

2.3.2.2 Wavelet discreta

En la actualidad la adquisición de señales y datos son de tipo digitales, es decir se dispone de banco de datos digitalizados provenientes de sistemas de instrumentación, por ello es necesario contar con una herramienta matemática que permita trabajar con señales digitales, de esta manera surge la necesidad de emplear la Transformada Wavelet Discreta (DWT).

La DWT (Transformada Wavelet Discrete) es una formulación matemática que transforma un arreglo de datos de dimensión en otro vector de coeficientes.

Basado en la transformada wavelet continua, se presentan 2 tipos de coeficientes (u,s), por lo que es necesario llevar estos coeficientes a un plano discreto.

$$s = a^i \tag{2.13}$$

$$u = kna^i \tag{2.14}$$

El término a^i se refiere a un paso de dilatación fijo, i y k son enteros. A partir de esto se remplaza en la ecuación de la WM:

$$\psi_{i,k}(t) = a^{-i/2}\psi\left(a^{-i}\left(t - kna^i\right)\right) \tag{2.15}$$

$$\psi_{i,k}(t) = a^{-i/2}\psi(a^{-i}t - kn)) \tag{2.16}$$

El factor de escala s puede ser usado para identificar singularidades de las señales, es decir, puede expandir o contraer el ancho de ventana de análisis, mientras que el valor de u representa en que parte de la señal se analiza.

La DWT se expresa como:

$$DW_f = \int_{-\infty}^{\infty} f(t)\,\psi_{i,k}(t)d(t) \tag{2.17}$$

Se puede representar la función $f(t)$ en función de la sumatoria de los coeficientes wavelet discretos multiplicados por las funciones base.

$$f(t) = \sum_{s}\sum_{u} W_f(s,u)\,\psi_{u,s}(t) \tag{2.18}$$

Basado en la Transformada Wavelet, la reconstrucción de la señal no contiene información redundante; por lo tanto, al aplicar estos planteamientos, el problema de sobre dimensionalidad queda resuelto.

2.3.2.3 Reconstrucción de señal

La Transformada Wavelet se expresa como una serie de filtros agrupados por etapas, a esto se le denomina "Niveles de descomposición". En cada etapa se encuentran dos procesos: filtrado para alta frecuencia y baja frecuencia. Cada proceso de filtrado adquiere coeficientes de detalle y aproximación.

En la figura 11 de observa una representación gráfica de la descomposición de la señal original, donde p y q representan las etapas de filtrado.

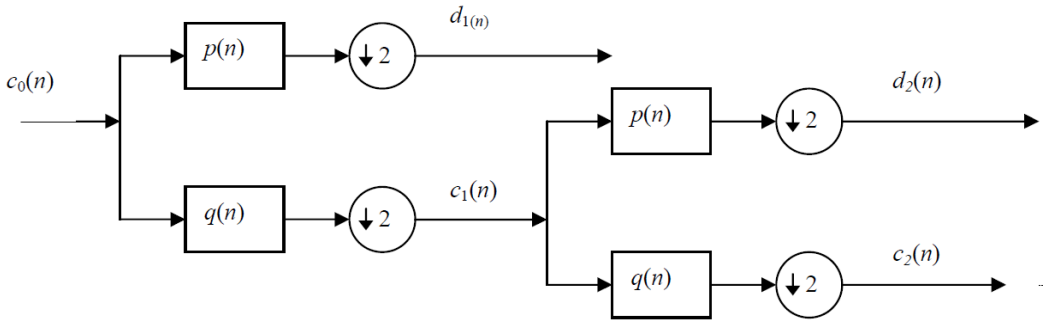

Figura 11 Diagrama de bloques de disposición de señal
Fuente: Nieto & Orozco, 2008.

El proceso denominado como p se refiere a un filtro de alta frecuencia y el proceso q a un filtro de baja frecuencia. Se debe considerar el orden de coeficientes, C_0 se refiere a la señal original, c_i se refiere a los coeficientes de aproximación y d_i a los coeficientes de detalle.

Se puede reconstruir la señal usando la ecuación discreta de c se expresa como:

$$c_{i-1}(n) = 2^{-1/2} \sum_k c_i(k)p(n-2k) + 2^{-1/2} \sum_k d_i(k)q(n-2k) \tag{2.19}$$

La ecuación 2.19 se expresa de manera gráfica en la figura 12:

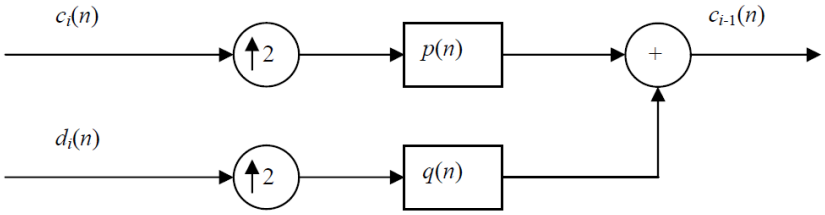

Figura 12 Diagrama de bloques de reconstrucción de señal
Fuente: Nieto & Orozco, 2008.

2.3.3 Validación de reducción

Para comparar 2 señales digitales de diferentes dimensiones es necesario realizar un proceso de interpolación con el objetivo de completar puntos intermedios para la señal de menor dimensión. Tras realizarse este procedimiento, se procede a calcular la diferencia de cada uno de los puntos, es decir se calcula todos los errores por cada componente, al final se obtiene un vector de 240 errores.

La figura 13 muestra un subsistema de todo del proceso de reducción de bandas por transformada wavelet discreta, este subsistema tiene una entrada (firma espectral original) y y una salida (firma espectral reducida).

Figura 13 Diagrama de bloque para reducción usando DWT
Fuente: Elaboración propia.

$$SS_{orignal,\ n=240} = \left[x_1, x_2, x_3, \ldots, x_n\right] \tag{2.20}$$

$$SS_{wavelet,\ n=60} = [x_1, x_2, x_{3,\ldots,} x_n] \tag{2.21}$$

A partir de los vectores anteriores, se construye un nuevo vector de datos de tamaño de 240 términos a partir del vector original reducido de 60 términos por interpolación.

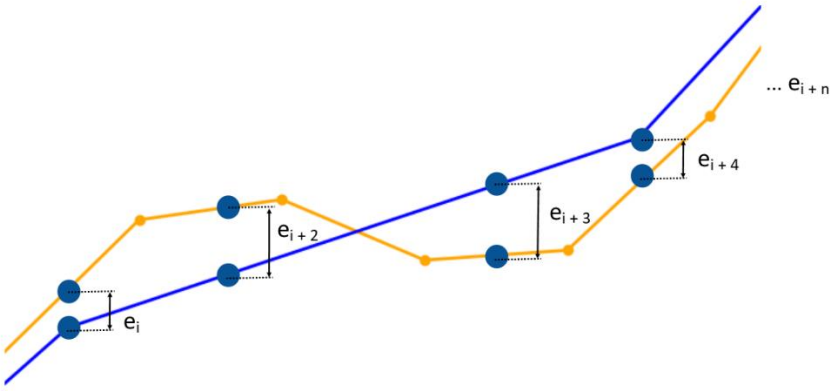

Figura 14 Gráfico de muestras para firmas espectrales
Fuente: Elaboración propia.

El grafico presentado en la figura 14 se muestran las 2 firmas: de color anaranjado la firma original y de azul la firma reducida. Se obtiene el vector de errores para evaluar la similitud de ambas firmas.

Se procede a evaluar el Error Medio Cuadrático o Root Mean Squared (RMS), el cual plantea medir el error promedio al cuadrado y evita que las diferencias negativas y positivas se compensen. El criterio más importante para evaluar la similitud es que el RMS sea pequeño. El caso ideal es ser muy cercano a 0.

$$RMS = \frac{1}{n}\sum_{k=1}^{n}(\hat{X}_k - X_k)^2 \tag{2.22}$$

Con el apoyo de las herramientas estadísticas, se propone otro modelo de validación de similitud entre las firmas espectrales originales y reducidas. La prueba de muestras pareadas es utilizada para identificar una relación entre las observaciones de las muestras. Se plantea una hipótesis estadística.

H_o = no tiene relación las 2 muestras de datos (t > 2, α = 95%)

H_1 = si tiene relación las 2 muestras de datos (t < 2, α = 95%)

$$t = \frac{\bar{x}_1 - \bar{x}_2}{S\sqrt{n}} \tag{2.23}$$

$\bar{x}_1$ se refiere a la media de la muestra original

$\bar{x}_2$ se refiere a la media de la muestra reconstruida

S se refiere a la desviación estándar la diferencia de las muestras

n se refiere al tamaño de muestra

α se refiere a la confiabilidad

En la figura 15 muestra de manera grafica la distribucion formada por la muestra de datos, y la region donde es aceptable la hipoteisis basada en la confiablidad α.

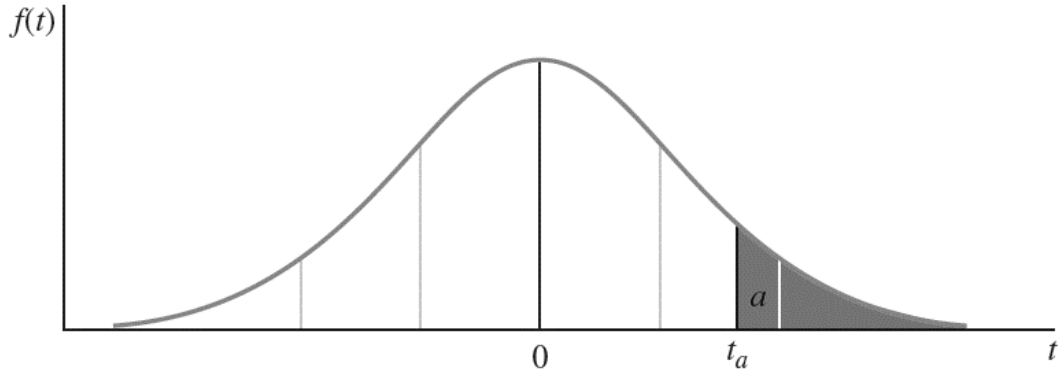

Figura 15 Gráfico de distribución T student
Fuente: Elaboración propia.

Capítulo 3
Sistemas de aprendizaje automático

3.1 Support Vector Regression

Las máquinas de soporte vectorial son unas de las metodologías de optimización matemática más relevantes dentro de la familia de algoritmos del aprendizaje automático supervisado con aplicaciones reales. En la utilización de las máquinas de soporte orientado a regresiones se encuentran sistemas de optimización lineal y no lineal, es decir es necesario contar con herramientas matemáticas de optimización para desarrollar los algoritmos de regresión usando vectores de soporte.

Se presentan 2 tipos de regresiones: Lineales y no Lineales, se evaluará las condiciones matemáticas de ambas, pero es preciso reafirmar que el objetivo de este estudio es la aplicación de las regresiones no lineales usando soporte vectorial.

3.1.1 Regresión Lineal

La expresión matemática general de regresión lineal simple posee la forma:

$$f(x) = (w_1 x_1 + w_2 x_2 + w_3 x_3 + \cdots w_i x_i) + b \tag{3.1}$$

$$f(x) = \langle w, x \rangle + b \tag{3.2}$$

Para encontrar $f(x)$ es necesario calcular $\overline{w}$, para ello se recurre a las técnicas de optimización, la cual debe mantener los puntos de predicción dentro de una banda de confiabilidad.

Donde se busca minimizar la función:

$$\min(\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i + {\xi^*}_i)) \tag{3.3}$$

Se satisfacen las siguientes expresiones:

$$y_i - (\langle w, x_i \rangle + b) \leq \varepsilon + \xi_i \tag{3.4}$$

$$(\langle w, x_i \rangle + b) - y_i \leq \varepsilon + \xi^*{}_i \tag{3.5}$$

$$\xi_i \geq 0, \xi^*{}_i \geq 0, \ i = 1, 2, 3 \ \ldots, n \tag{3.6}$$

La constante C es fundamentar para determinar la tolerancia cercana a ε; ξ_i y ξ_i se refiere a los términos que regulan el error cometido por la función $f(x)$.

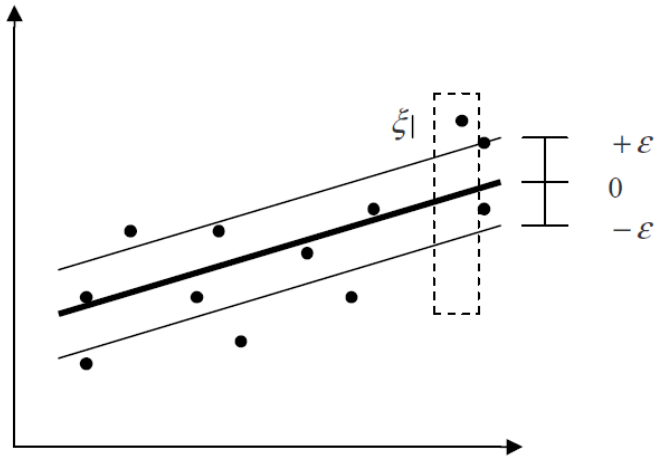

Figura 16 Gráfico de pérdida de margen de SVM lineal
Fuente: Basak, Pal, & Patranabis, 2007.

La interpretacion grafica de C y ξ_i junto a los hiperplanos de separacion se aprecian en la figura 16.

Consideraciones matemáticas:

- Caso 1: cuando el valor de $C \uparrow\uparrow$, $C \rightarrow \infty$ tiene como consecuencia que $\xi_i \rightarrow 0$
- Caso 2: cuando el valor de $C \rightarrow 0$ tiene como consecuencia que $\xi_i \uparrow\uparrow$

Interpretando:

Para el *caso 1* se conceptualiza una regresión ideal, es decir una perfecta correlación.

Para el *caso 2* se refiere una regresión de elementos mal representados. La posible conclusión de esto es elaborar un criterio una convención con el fin de elegir parámetros de sintonización adecuados.

Se plantea una un método de optimización usando Lagrange:

$$L(w, b, \xi, \xi^*, \alpha, \alpha^*, \eta, \eta^*) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\left(\xi_i + \xi^*{}_i\right) - \sum_{i=1}^{n}\alpha_i(\varepsilon + \xi_i - y_i + \langle w, x_i\rangle + b) - \sum_{i=1}^{n}\alpha^*{}_i(\varepsilon + \xi_i + y_i - \langle w, x_i\rangle - b) - \sum_{i=1}^{n}\left(\eta_i\xi_i + \eta^*{}_i\xi^*{}_i\right) \quad (3.7)$$

Donde las variables w, b, ξ, ξ^* a obtener y $\alpha, \alpha^*, \eta, \eta^*$ son variables provenientes de las restricciones, donde el objetivo es maximizar (w, b, ξ, ξ^*) y minimizar $(\alpha, \alpha^*, \eta, \eta^*)$ usando la ecuación de Lagrange.

Usando expresiones matemáticas como derivadas parciales y condiciones de punto de silla, resulta una expresión de optimización dual y la función de predicción:

$$\max\left(-\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i - \alpha^*{}_i)\left(\alpha_j - \alpha^*{}_j\right)\langle x_i, x_j\rangle - \varepsilon\sum_{i=1}^{n}(\alpha_i - \alpha^*{}_i) + \sum_{i=1}^{n} y_i(\alpha_i - \alpha^*{}_i)\right) \quad (3.8)$$

$$f(x) = \sum_{i=1}^{n}(\alpha_i - \alpha^*{}_i)\,\langle x_i, x\rangle + b \quad (3.9)$$

En la función de predicción $f(x)$ se observa que depende de los vectores de soporte y no depende de la dimensión de las muestras de entrada.

El valor de b se encuentra usando la siguiente expresión:

$$b = y_i - \langle w, x_i\rangle - \varepsilon \quad (3.10)$$

3.1.2 Regresión No Lineales

Basados en los conceptos anteriores, es posible heredar y extender la metodología de las regresiones lineales múltiples a las regresiones no lineales múltiples, a través de la obtención de un espacio de dimensiones muy altas y hasta infinitas, denotaremos a este espacio como Φ, este espacio será aplicado a todos los datos de entrada con el fin de aproximar a comportamiento lineal, es decir se hace uso de $\Phi(x)$ como parámetros de entrada. (Basak, Pal, & Patranabis, 2007)

La nueva función que encontrar:

$$f(x) = \langle w, \Phi(x)\rangle + b \quad (3.11)$$

De igual manera se plantea:

$$\min(\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi^*_i)) \tag{3.12}$$

Se satisfacen las siguientes expresiones:

$$y_i - (\langle w, \Phi(x_i)\rangle + b) \leq \varepsilon + \xi_i \tag{3.13}$$

$$(\langle w, \Phi(x_i)\rangle + b) - y_i \leq \varepsilon + \xi^*_i \tag{3.14}$$

$$\xi_i \geq 0, \quad \xi^*_i \geq 0, \quad i = 1, 2, 3 \ \ldots, n \tag{3.15}$$

Usando la misma analogía y extendiendo al caso no lineal, se tiene la siguiente función objetivo:

$$\max(-\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i - \alpha^*_i)\big(\alpha_j - \alpha^*_j\big)\langle\Phi(x_i), \Phi(x_j)\rangle - \varepsilon\sum_{i=1}^{n}(\alpha_i - \alpha^*_i) + \sum_{i=1}^{n} y_i(\alpha_i - \alpha^*_i)) \tag{3.16}$$

Unos de los problemas para proceder a resolver estas expresiones matemáticas es que complicado encontrar el espacio Φ, al analizar la expresión de la función objetivo es posible observar que depende exclusivamente de producto de la proyección del espacio, es decir de $\langle\Phi(x), \Phi(x')\rangle$

Al definir una función Kernel que represente el producto mencionando, le da un valor real que correspondería a las imágenes de los elementos del nuevo espacio.

$$K(x, x') = \langle\Phi(x), \Phi(x')\rangle \tag{3.17}$$

La función objetivo se representa de la siguiente forma:

$$\max(-\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i - \alpha^*_i)\big(\alpha_j - \alpha^*_j\big)K(x_i, x_j) - \varepsilon\sum_{i=1}^{n}(\alpha_i - \alpha^*_i) + \sum_{i=1}^{n} y_i(\alpha_i - \alpha^*_i)) \tag{3.18}$$

De igual manera para la función de predicción:

$$f(x) = \sum_{i=1}^{n}(\alpha_i - \alpha^*_i)\, K(x_i, x) + b \tag{3.19}$$

Existen muchos tipos de kernels, sin embargo, el más utilizado es el kernel de función base radial gaussiana, el kernel depende de una constante γ que controla la flexibilidad del kernel, es decir es otro parámetro de sintonización del modelo regresión.

$$K(x, x') = \exp(-\gamma \|x - x'\|^2) \tag{3.20}$$

3.2 Neural Networks

Las redes neuronales artificiales se encuentran entre los métodos más utilizados por la ciencia de datos e inteligencia artificial, gracias a su tropológica de capas y neuronas permite generar modelos complejos en situaciones donde no se cuente con algún modelo matemático o no admite un tratamiento algorítmico (Gabriel, 2016).

Los sistemas que tienen como base las redes neuronales artificiales tienden a simular el comportamiento del cerebro humano, es decir pueden abstraer conocimiento a partir de información observada y la experiencia adquirida anteriormente. Todo esto se reduce a tratar de emular un sistema biológico, similar al presentado en la figura 17, en un sistema computacional, por ello es indispensable conocer la primera instancia el proceso biológico más elemental de una neurona.

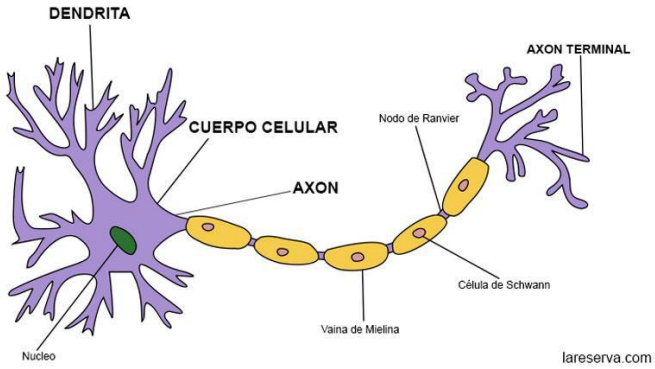

Figura 17 Representación gráfica de una neurona biológica
Fuente:
http://lareserva.com/home/sites/default/files/styles/article_image/public/field/image/neuronasf.jpg.

Básicamente una neurona es una unidad de procesamiento de información elemental, y como todo sistema de procesamiento posee entradas y salidas, para las neuronas biológicas, las encargadas de recibir la información proveniente de múltiples neuronal son las Dendritas, la parte encargada de abstraer y procesar información es el Cuerpo Celular, con esta información procesada, es necesario comunicarla hacia las demás neuronas o unidades de procesamiento, esta tarea es realizada por el Axón Terminal. Esta función de procesamiento de información se ajusta con cada interacción de neuronas. (Kriesel, 2007)

3.2.1 Entradas

En el plano computacional, es necesario definir una etapa de entradas, esta etapa de entradas es multivariable, es decir es un vector de n datos de entrada.

$$\hat{X} = [x_1, x_2, x_3, \ldots, x_n] \tag{3.21}$$

Dentro de las múltiples entradas, se tiene en cuenta la influencia de cada una de las variables, por consecuencia es necesario establecer pesos o números que, al multiplicar por la variable, incrementen o disminuya la importancia en la unidad procesadora.

$$Xi = w_1 * x_1 + w_2 * x_2 + w_3 * x_3 + .. . + w_n * x_n \tag{3.22}$$

3.2.2 Función de activación

Las funciones de activación son las encargadas de la actividad de la neurona, recibe la información de la entrada, pasa por una función de activación para obtener un valor de salida, una función de activación también llamada función de transferencia, toma cierto comportamiento respecto a la entrada. Entre las funciones de transferencias más comunes tenemos:

Función de activación lineal:

$$f(x) = \begin{cases} -1; & x \leq -1/a \\ a * x; & -1/a < x < 1/a \\ 1; & x \geq 1/a \end{cases} \tag{3.23}$$

Función de activación sigmoid:

$$f(x) = \frac{1}{1 + e^{-g*x}} \tag{3.24}$$

Función de activación tanh:

$$f(x) = \frac{e^{g*x} - e^{-g*x}}{e^{g*x} + e^{-g*x}} \tag{3.25}$$

3.2.3 Salidas

La última etapa del proceso de interacción de una neurona se encarga de entregar la información hacia las demás neuronas, con cada paso de neurona se procesa la información, creando una red de neuronas.

Una neurona por si sola tiene muy poca probabilidad de éxito para relacionar patrones, pero trabajando en conjunto múltiples redes neuronales, si es capaz de abstraer conocimiento del proceso. Los sistemas de aprendizaje automático utilizan en su gran mayoría varias neuronas interconectadas o redes neuronales, estas redes poseen diferentes tipos de organización, también llamada arquitectura de redes neuronales o topología de redes de neuronales.

3.2.4 Topología de redes neurales

Las redes neuronales conectadas entre sí poseen diferentes disposiciones, estas disposiciones son de tipo de capa o formaciones de neuronas, estas capas se encuentran entre las entradas y salidas del sistema.

Es importante conocer el número de capas, el número de neuronas por capa; los procesos de selección de estos parámetros descritos anteriormente influyen directamente en la efectividad del sistema con redes neuronales.

Los sistemas multicapa son los más comunes para desarrollar sistemas de predicción y disponen de un grupo de neuronas agrupadas, para determinar a qué capa pertenece cada grupo de neuronas es necesario observar las entradas y salidas de las neuronas. En la figura 18 se observa de manera generalizada las entradas y salidas de los grupos de capas, usualmente las capas más cercanas a la entrada le envían información a siguiente capa, hasta llegar a la salida.

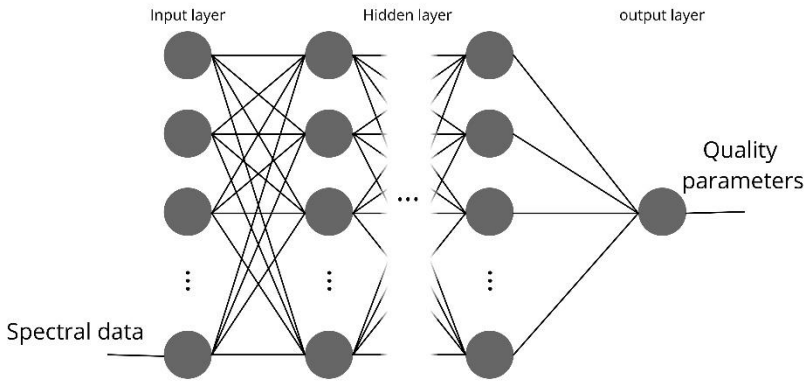

Figura 18 Representación de redes neuronales multicapa
Fuente: Elaboración propia.

Existen grupos que engloban a ciertos tipos de capas y estas están clasificadas en tres: capa de entrada, capas ocultas y capa de salida, la capa de entrada es un conjunto de redes

neuronales que se encargan de recibir y procesar la información de entrada, esta capa realiza un procedimiento de extracción de características, las capas ocultas son las encargadas de la mayor parte de la generalización del modelo y por último la capa de salida presenta las variables de predicción, para el caso de clasificación se tienen valores binarios o discretos en las salidas y para el caso de regresión se tienen valores numéricos flotantes (Gurney, 2004).

3.2.5 Tipos de redes neuronales artificiales

Para poder determinar el mejor modelo de arquitectura de redes neuronales es necesario analizar las ventajas y desventajas de cada una de ellas, por ello se presentan los principales tipos de redes neuronales.

3.2.5.1 Perceptron

Las redes neuronales de tipo perceptron tiene una topología donde se realiza una suma ponderada o sumatoria de las entradas con determinados pesos, esta sumatoria pasa por una función de activación escalonada con umbral determinado, es decir la salida es binaria. Este tipo de redes neuronales es muy utilizado para modelos de clasificación. El perceptron separa a través de hiperplanos donde su ecuación está bajo el régimen de pesos y de la función de activación. En la figura 19 se aprecia una representación gráfica de un neuronal perceptrón.

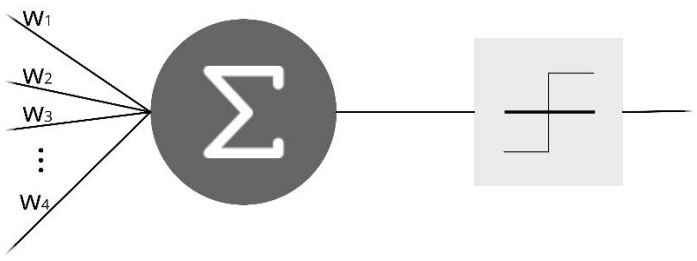

Figura 19 Representación de una neurona perceptron.
Fuente: Elaboración propia

La desventaja principal de este tipo de arquitectura es que es efectivo siempre y cuando lo que se desee modelar es linealmente separable. Una de las soluciones más comunes es llevar los datos de entrada hacia otro espacio con el fin de poder tener un grupo de datos linealmente separable.

3.2.5.2 Backpropagation

Backpropagation es un tipo de red neuronales que utiliza etapas de propagación, donde la información recibida en la entrada es propagada por todas las redes, hasta generar una salida, como primera iteración. Esta salida de la primera iteración se compara con la salida de referencia y con ello se calcula un error. Los errores calculados se propagan en sentido inverso hasta la entrada, se ajustan los pesos de las redes neuronales para obtener en cada

iteración un error menor, estos procesos iterativos se realiza hasta que converja hasta un error muy cercano a cero. Como se aprecia en la figura 20, la capa de cálculo de errores es ingresado para el reajuste de pesos. Los métodos de optimización cobran gran importancia para optimizar el tiempo de cada iteración, minimizar el número de iteraciones y evitar divergencias.

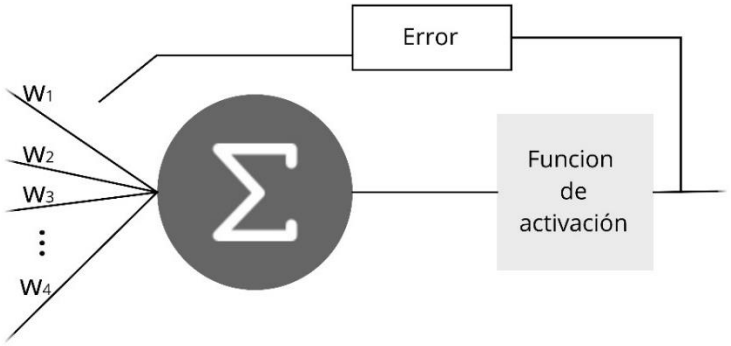

Figura 20 Representación de una neurona Backpropagation
Fuente: Elaboración propia.

3.2.5.3 Adeline

Los sistemas de redes Adeline tienen aplicación para el procesamiento de señales, diseño de filtros adaptativos, este sistema trabaja por corrección de errores, las funciones de activación son de tipo lineal. Al igual que la red neuronal perceptron tiene la desventaja de modelar solo sistemas linealmente separables, sin embargo, el método de minimización de errores incrementa la efectividad respecto a la red de tipo perceptron. Se considera que la red neuronal Adeline es una arquitectura heredada de la red neuronal perceptron (Zhang, 2007).

En la figura 21 muestra el diagrama de la arquitectura de la red neuronal Adeline, es posible observar una referencia y un algoritmo adaptativo para la corrección de pesos.

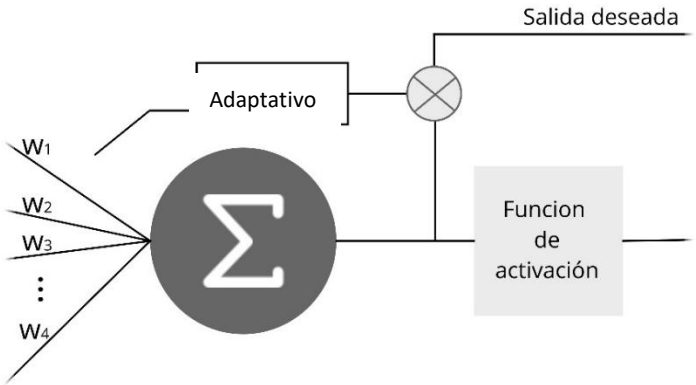

Figura 21 Representación de una neurona Adeline
Fuente: Elaboración propia

3.5.2.4 Aprendizaje asociativo

Estos sistemas de aprendizaje están orientados para el aprendizaje no supervisado, donde las redes neuronales artificial no reciben información para contrastar las salidas frente a las entradas. Estos sistemas deben ser capaces de extraer correlaciones y características en los datos de entradas a fin de encontrar cierto grado de similitud entre la entrada de la red, encontrar grupos con características en común y agruparlos convenientemente, realizar un proceso de compresión de datos, tiendo en la salida un menor número de bits sin perder la información original.

El aprendizaje asociativo se refiere a encontrar una correlación o vínculos entre las entradas y sus salidas, estas correlaciones son vinculadas por asociaciones.

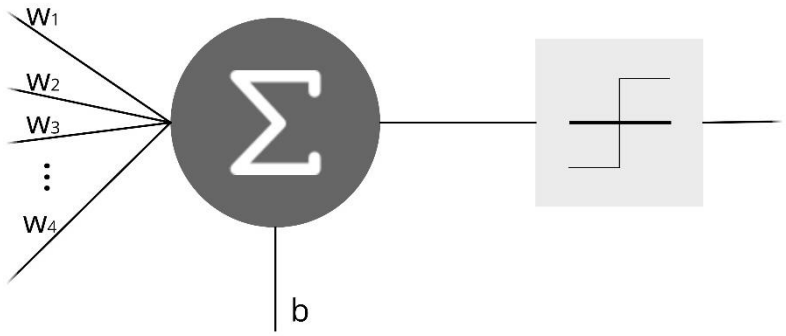

Figura 22 Representación de una neurona de aprendizaje asociativo
Fuente: Elaboración propia

Para este tipo de arquitectura la salida tiene una ecuación:

$$out = hardlim(\overline{w} * \overline{inputs} + bias) \tag{3.26}$$

Se tiene que $\overline{inputs}$ indica presencia de estímulos, donde la neurona responderá a los estímulos si $\overline{w} * \overline{inputs} > -b$ias.

Es posible visualizar en la figura 22 el b o bias, al modificar esta constante es posible modificar la salida de la neurona con el objetivo reducir el número de iteraciones para obtener los pesos óptimos así como su convergencia.

3.2.6 Multilayer Perceptron

Los sistemas de predicción usando las redes neuronales Multi Layer Perceptron (MLP), recientemente lo consideran como una herramienta valiosa para la predicción de múltiples parámetros de calidad de harina de trigo (Mutlu, y otros, 2011). La arquitectura MLP o también llamada BackPropagtion (BP) es la arquitectura más utilizadas por la ciencia de datos para la identificación de modelos, es considerada como “Aproximador Universal”, considera modelos lineales y no lineales (Kashaninejad, Dehghani, & Kashiri, 2009).

En la MLP recibe los datos a través de las neuronas de las capas de entrada, después de proceso de multiplicación de pesos, acoplamientos del bias (b) y pasar por la función de transferencia, son transmitidos a las capas ocultas, y estas a su vez transmitidos a la capa de salida.

Entre las aplicaciones de las redes MLP se tiene la clasificación, regresiones y agrupamiento selectivo, para este caso de estudio se tendrá en consideración el análisis de las regresiones usando MLP.

3.2.6.1 Regresiones

La expresión matemática generalizada multivariable de regresión posee la forma:

$$f(x_{li}) = g_i(x_l) + \varepsilon_{li}$$
$$l = 1, 2, 3 \ \dots, n; \quad i = 1, 2, 3 \ \dots, q \tag{3.27}$$

Donde $f(x_{li})$ es la salida o respuesta multivariable, x_l se refiere a las variables independientes de entradas, g_i es un modelo no paramétrico desconocido y ε_{li} es el error aleatorio.

Para el desarrollo de los sistemas de regresiones para predicción de parámetros de harina de pescado se tienen las siguientes consideraciones:

- Se genera una regresión por cada parámetro, es decir se desarrolla un sistema de múltiples entradas (firma espectral) y una salida (parámetros físicoquímicos).
- Se asume el error de tipo ruido blanco, el error generado por la predicción se considera nulo o cero.

Para encontrar valores estimados es necesario construir un vector de estimadores:

$$\hat{g}_i = [\hat{g}_1, \hat{g}_2, \hat{g}_3, \dots, \hat{g}_q]$$
$$i = 1, 2, 3 \ \dots, q \tag{3.28}$$

Por consiguiente, la predicción viene dada por:

$$\hat{y}_i = \hat{g}_i(x) \tag{3.29}$$

Las salidas dependen exclusivamente de los estimadores y esos a su vez depende de los parámetros de las redes neuronales MLP, se describe en la formulación matemática de los estimadores:

$$\hat{g}_i(x) = \sum_{k=1}^{m} \beta_{ik}\sigma(\sum_{j=1}^{P} w_{kj}\, x_j \; - \; \omega_{k0}) \tag{3.30}$$

Donde β_{ik} se refiere a los pesos de la capa de oculta y salida de i- neuronas de salida y k-neuronas ocultas, $\sigma(z)$ se refiere a la función de activación, w_{kj} se refiere a los pesos de la capa de entrada de k-neuronas ocultas y j-neuronas de entrada, x_j vector de entrada y ω_{k0} se refiere a el bias de k-neuronas ocultas (Hwang, Lay, Maechler, Martin, & Schimer, 1994).

3.2.6.2 LBFGS

El algoritmo LBFGS o Limited memory Broyden Fletcher Goldfarb Shanno es utilizado como método para resolver problemas de optimización no lineales, para el objetivo de esta tesis será empleado para el cálculo de obtención de pesos para el modelo de redes neuronales MLP. El método LBFGS tiene como objetivo minimizar funciones no lineales muy extensas, inicialmente este algoritmo se planteó para sistemas numéricos relacionados a la mecánica de fluidos y biología molecular.

Para LBFGS se tiene:

$$\min f(x)\,; \quad x \in R^n \tag{3.31}$$

Donde $f(x)$ es una función que tiene segunda derivada continua.

Para el caso de la obtención de pesos, la $f(x)$ es un vector o matriz de pesos y errores, donde el principal propósito es minimizar el error del modelo de redes neuronales MLP. Este proceso es iterativo, por consiguiente, es necesario la implementación de métodos numéricos. Para la implementación de este algoritmo se hace uso del método numérico quasi-Newton (Gilbert & Lemarechal, 1989).

3.3 Validación cruzada y error de predicción

En el presente apartado hace referencia a las técnicas utilizadas para determinar la calidad de los modelos de regresiones. La validación cruzada consiste en contrastar información obtenida por el modelo de predicción con información de prueba para evaluar la efectividad del modelo.

Existen múltiples métodos estadísticos para validar métodos de regresiones, una de ellas son la comparación de parámetros obtenidos con los parámetros obtenidos por modelos teóricos (Kozak & Kozak, 2003).

3.3.1 Coeficiente de correlación

Teóricamente los datos obtenidos por el modelo de predicción deben ser iguales a los datos empleados en la validación cruzada, es decir guardan una relación lineal. Es importante medir la fuerza de esta relación, es aquí donde toma protagonismo el coeficiente de correlación, un parámetro estadístico que mide la fuerza de una relación lineal sin importar las unidades (Pita & Pértega, 1997).

El coeficiente de correlación tiene la forma:

$$R = \frac{1}{n-1}\sum_{i=1}^{n}(\frac{X_i}{S_x})(\frac{Y_i}{S_y}) \tag{3.32}$$

Donde:

$$S_x = \sqrt{\frac{\sum {X_i}^2}{n-1}} \tag{3.33}$$

$$S_y = \sqrt{\frac{\sum {Y_i}^2}{n-1}} \tag{3.34}$$

Donde n se refiere al tamaño de la muestra, X_i e Y_i se refieren a la variables que se desea encontrar el grado de asociaron lineal.

El valor de R toma valores de entre 0 y 1. Se tienen las siguientes interpretaciones de coeficiente de correlación:

- Caso 1: cuando el valor de $R = 1$, ambas muestras son exactamente iguales.
- Caso 2: cuando el valor de $R \rightarrow 1$, se obtiene una relación linear fuerte.
- Caso 3: cuando el valor de $R \rightarrow 0$, se obtiene una relación linear débil.
- Caso 4: cuando el valor de $R = 0$, no existe relación lineal.

3.3.2 Normalidad de errores

Otro índice para determinar la calidad de predicción obtenida por el modelo, es estudiar la distribución que forman los errores del modelo de predicción. Para un modelo de regresión es deseable obtener errores aleatorios (no dependen del modelo) y de media cero. La distribución que asegura la aleatoriedad de los errores es la distribución normal, donde:

$$\varepsilon_i \approx N(0, \sigma^2) \tag{3.35}$$

Para determinar si la distribución formada por los errores del modelo se ajusta a la una distribución normal es necesario realizar una prueba de ajuste. Se selecciona la prueba Shapiro Wilk para contrastar la hipótesis de normalidad.

Se plantea una hipótesis estadística.

H_o = se ajusta a una distribución normal (p-value > α, α = 2% y w >> 0)

H_1 = no se ajusta a una distribución normal (p-value > α, α = 2% y w << 0)

$$w = \frac{1}{ns^2}(\sum_{i=1}^{h} a_{in}(x_{(n-i+1)} - x_{(i)}))^2 \tag{3.36}$$

$$h = \begin{cases} \frac{n}{2}; & si\ n\ es\ par \\ \frac{n-1}{2}; & si\ n\ es\ impar \end{cases} \tag{3.37}$$

$x_{(i)}$ es el vector ordenado de menor a mayor del vector muestral

s^2 se refiere a la varianza muestral

a_{in} se refiere a los coeficientes de estadístico Shapiro Wilk.

α se refiere al nivel de significación

Capítulo 4
Desarrollo y experimentación

4.1 Desarrollo de software

El desarrollo algoritmos de reducción de dimensionalidad (transformada wavalet), algoritmos de regresión (support vector regression y multilayer perceptron regression), validaciones estadísticas, visualización de información, prototipo de modelo, sistema en producción y conexiones de bases de datos en tiempo real fueron implementados en el lenguaje de programación Python.

Se utiliza el framework TensorFlow como biblioteca principal para el desarrollo y puesta en producción de los sistemas de aprendizaje automático. TensorFlow es un framework de código libre que realiza cálculos numéricos mediante flujo de datos, cuenta con un potente compilador de algebra lineal que optimiza la ejecución del código en los procesadores y diferentes hardware de procesamiento. Este framework es utilizador por ingenieros e investigadores para desarrollo de software de aprendizaje automático en la empresa Google. Los sistemas desarrollados por TensorFlow se adapta a los diferentes escenarios de uso, desde modelos prototipo a sistemas en producción, escalables y de gran demanda (Google, 2018).

El tiempo de ejecución del algoritmo de desarrollo por parámetro es de 2.5 s y el tiempo de ejecución del algoritmo en producción por parámetro es de 1.8 s.

4.2 Reducción de firma espectral

En el apartado 3.2 se analiza el estudio teórico para la validación de la reducción de la firma espectral, en este apartado se presentan los resultados de las pruebas.

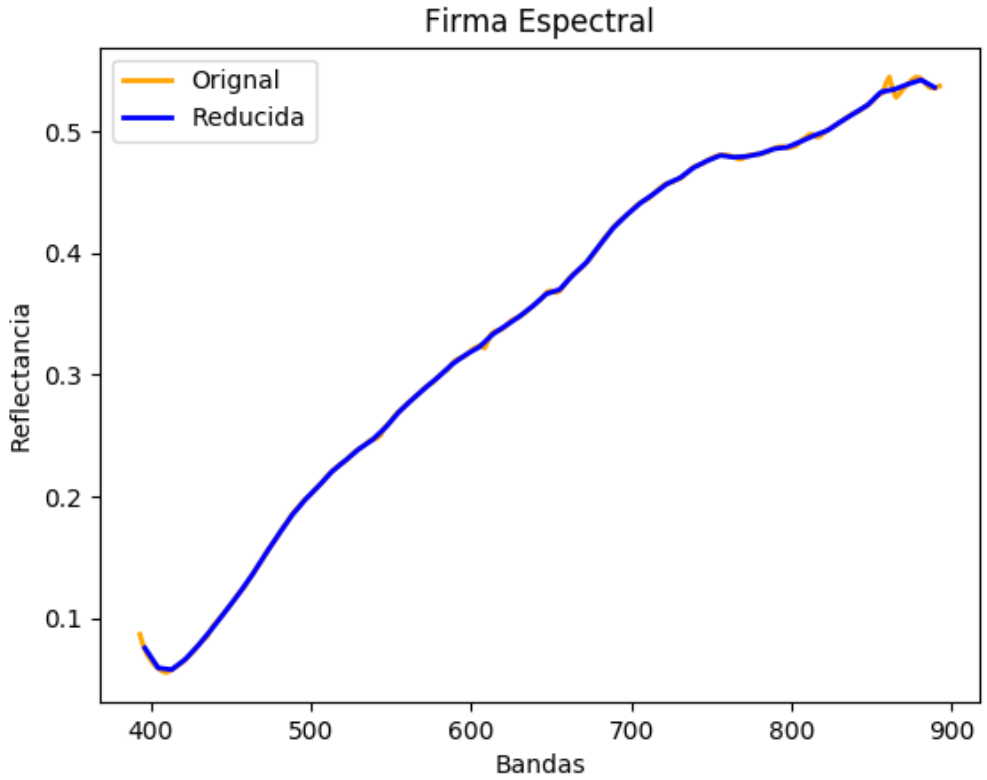

Figura 23 Grafico de firmas espectrales superpuestas
Fuente: Elaboración propia.

En la figura 23 se aprecia 2 firmas espectrales de diferente número de compontes, la firma espectral original posee 240 componentes, mientras que la firma espectral reducida posee 60 componentes.
Calculo y resultados de RMS:

- Número de firmas espectrales = 50
- RMS promedio $\approx 1e-05$

El valor de RMS es muy cercano a 0, lo cual confirma que ambas firmas espectrales (original y reducida) poseen alta similitud, poca perdida de información eliminando el problema de dimensionalidad de manera eficiente. Para verificar la generalización de este comportamiento, se utilizaron 50 firmas espectrales de muestras de harina de pescado.

Calculo y resultados de prueba de muestras pareadas:

- Grados de libertad = 240
- Región de aceptación = 0.025
- $t_a = 1.96$
- $t_0 = 0.76$

Donde:

$$t_0 < t_a$$

H_o = no tiene relación las 2 muestras de datos (t > t_a, α = 95%)

H_1 = si tiene relación las 2 muestras de datos (t < t_a, α = 95%)

Se verifica la hipótesis H_1.

Para verificar la generalización de este comportamiento, se utilizaron 50 firmas espectrales de muestras de harina de pescado diferente, con lo cual se obtuvo que, para todas las muestras cumplen con la hipótesis estadística.

Todas las firmas preprocesadas no poseen pérdida de datos y se encuentran en el mismo subespacio; sin embargo, el volumen de datos es menor, ya que es la cuarta parte del volumen original.

4.3 Parámetros de sintonización de modelos de regresión

Para la implementación del algoritmo de aprendizaje automático se utilizó el 70% del tamaño de los datos de entrada, para el grupo A se seleccionó 130 muestras y para el grupo B 84 muestras. Los datos restantes se emplearon para pruebas de validación.

En total se seleccionaron muestras provenientes de diferentes lotes de producción, los cuales provee la siguiente información (Proteína, Grasas, Humedad, Cenizas, Cloruros, Arena, TVBN, Histamina, REMAO[7]). Estos parámetros de calidad fueron analizados químicamente por una empresa reguladora de calidad de producto.

Para el análisis de parámetros de calidad se escogieron Proteína, Grasas, Humedad y Cenizas; estos parámetros son seleccionados debido a su gran aporte en la calificación comercial.

Para Grupo A:
- Muestras: 190
- Año: 2016 - 2017
- Inicio: 01/12/2016
- Fin: 27/04/2017

[7] Remanente de antioxidantes, utilizado para preservación de harina de pescado.

Tabla 1. Descripción de parámetros de harina de pescado para muestra A

Parámetro	Unidad	Rango	Media	D.S.
Proteína	%	64.3–71.5	68.5	1.2
Grasas	%	6.9 – 10.5	8.7	0.8
Humedad	%	5.4-9.2	7.05	0.7
Cenizas	%	14.2-19.2	15.8	0.9

Fuente: Elaboración propia.

En la tabla 1 presentan todos los parámetros identificados en el grupo A usando modelos de regresión junto a su unidad y métricas estadísticas de interés. Estas métricas darán una idea sobre la calidad de predicción.

Para Grupo B:
- Muestras: 122
- Año: 2018
- Inicio: 07/04/2018
- Fin: 22/04/2018

Tabla 2. Descripción de parámetros de harina de pescado para muestra B

Parámetro	Unidad	Rango	Media	D.S.
Proteína	%	65.5–70.8	68.1	1.04
Grasas	%	7.4 – 10.3	8.9	0.6
Humedad	%	5.8-10.1	7.5	0.6
Cenizas	%	14.7-17.02	15.9	0.4

Fuente: Elaboración propia.

En la tabla 2 presentan todos los parámetros identificados en el grupo B usando modelos de regresión junto a su unidad y métricas estadísticas de interés. Estas métricas darán una idea sobre la calidad de predicción.

4.3.1 Support vector regression

Parámetros de sintonización para modelo de regresión usando Support Vector:

Para todos los modelos se emplean el kernel “RBF”. Los parámetros de sintonización por parámetro se presentan en la tabla 3

Tabla 3. Parámetros de sintonización de modelo SV

Parámetro	C	gamma	epsilon
Proteína	1e4	200	1e-8
Grasas	1e1	80	1e-1
Humedad	1e2	100	1e-3
Cenizas	1e1	50	1e-3

Fuente: Elaboración propia.

Los parámetros de sintonización de la tabla 3 determinan la calidad de ajuste y predicción del modelo Support Vector Regression. Estas constantes se determinaron de manera empírica y determinan la mejor calidad de predicción.

Para grupo A:

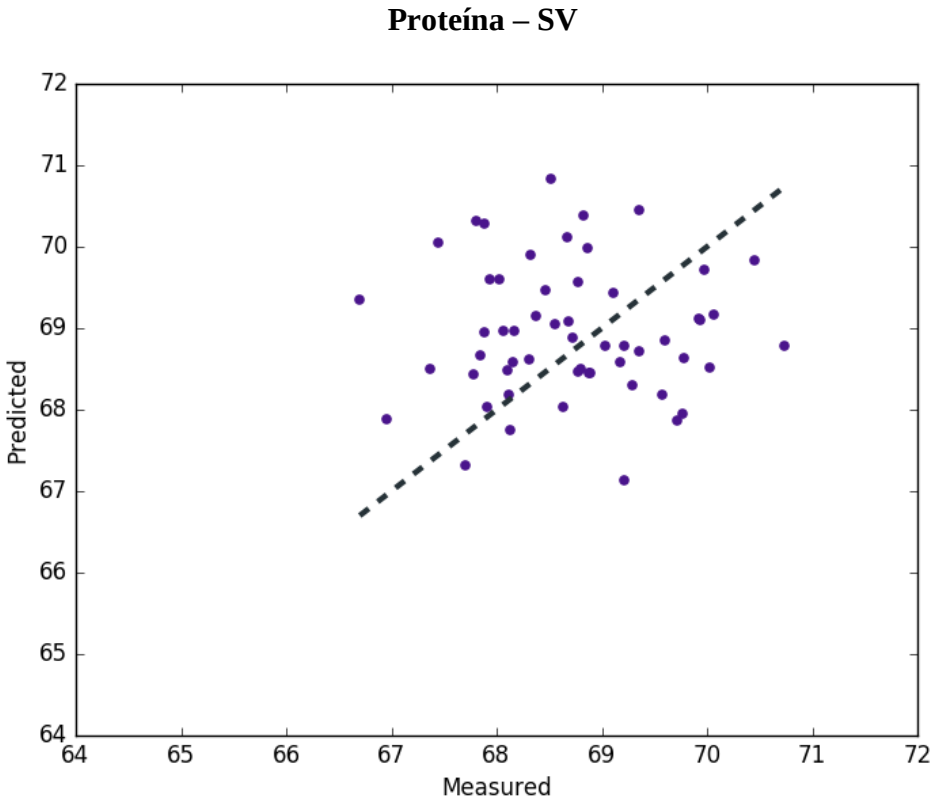

Figura 24 Gráfico de validación de modelo para Proteína
Fuente: Elaboración propia.

La figura 24 pertenece a un gráfico de validación cruzada para evaluación del grupo A, donde se aprecia la comparativa de los datos medidos y los datos calculados por modelo de regresión. Este grafico también incluye una línea de tendencia ideal. El modelo empleado se refiere Support Vector Regression para predecir proteína.

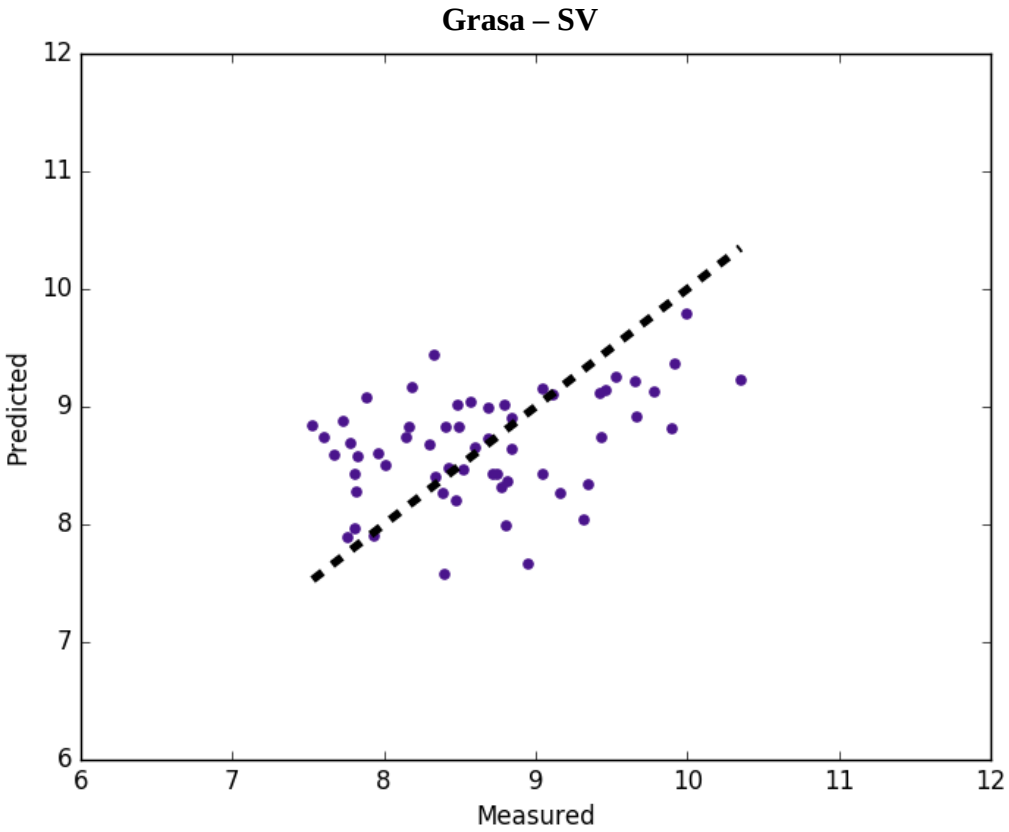

Figura 25 Gráfico de validación de modelo para Grasa
Fuente: Elaboración propia.

La figura 25 pertenece a un gráfico de validación cruzada para evaluación del grupo A, donde se aprecia la comparativa de los datos medidos y los datos calculados por modelo de regresión. Este grafico también incluye una línea de tendencia ideal. El modelo empleado se refiere Support Vector Regression para predecir grasa.

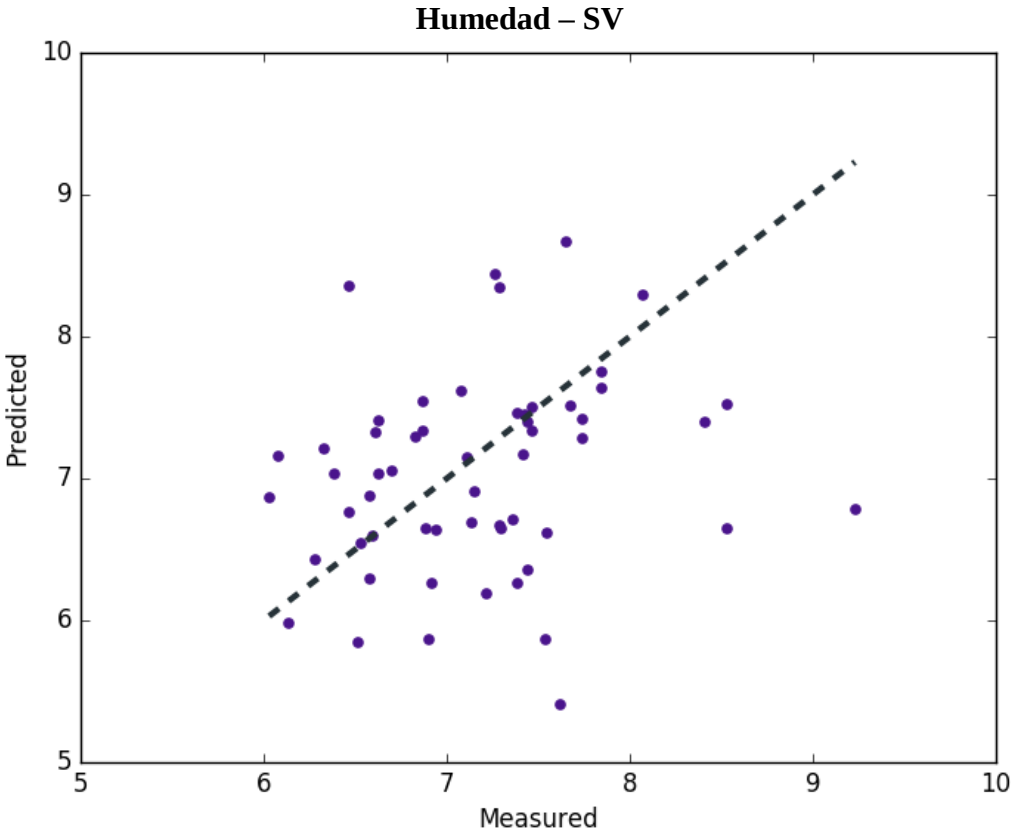

Figura 26 Gráfico de validación de modelo para Humedad
Fuente: Elaboración propia.

La figura 26 pertenece a un gráfico de validación cruzada para evaluación del grupo A, donde se aprecia la comparativa de los datos medidos y los datos calculados por modelo de regresión. Este grafico también incluye una línea de tendencia ideal. El modelo empleado se refiere Support Vector Regression para predecir humedad.

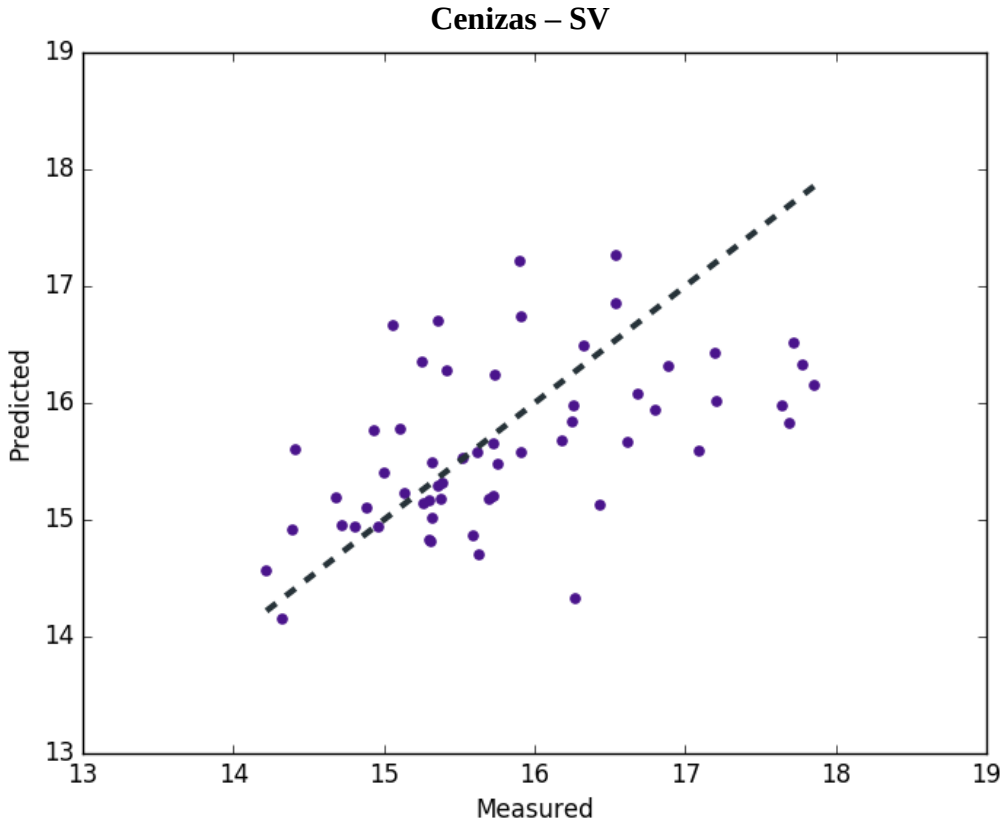

Figura 27 Gráfico de validación de modelo para Cenizas
Fuente: Elaboración propia.

La figura 27 pertenece a un gráfico de validación cruzada para evaluación del grupo A, donde se aprecia la comparativa de los datos medidos y los datos calculados por modelo de regresión. Este grafico también incluye una línea de tendencia ideal. El modelo empleado se refiere Support Vector Regression para predecir cenizas.

En la tabla 4 se presentan todos los parámetros que determinan la calidad de predicción del modelo de regresiones usando Support Vector.

Tabla 4. Resumen de validación cruzada para modelo de predicción

Parámetro	R	RMS	Error(%)
Proteína	0.01	1.46	1.55
Grasas	0.38	0.44	6.35
Humedad	0.25	0.79	9.28
Cenizas	0.51	0.71	4.1

Autor: Elaboración propia.

Para grupo B:

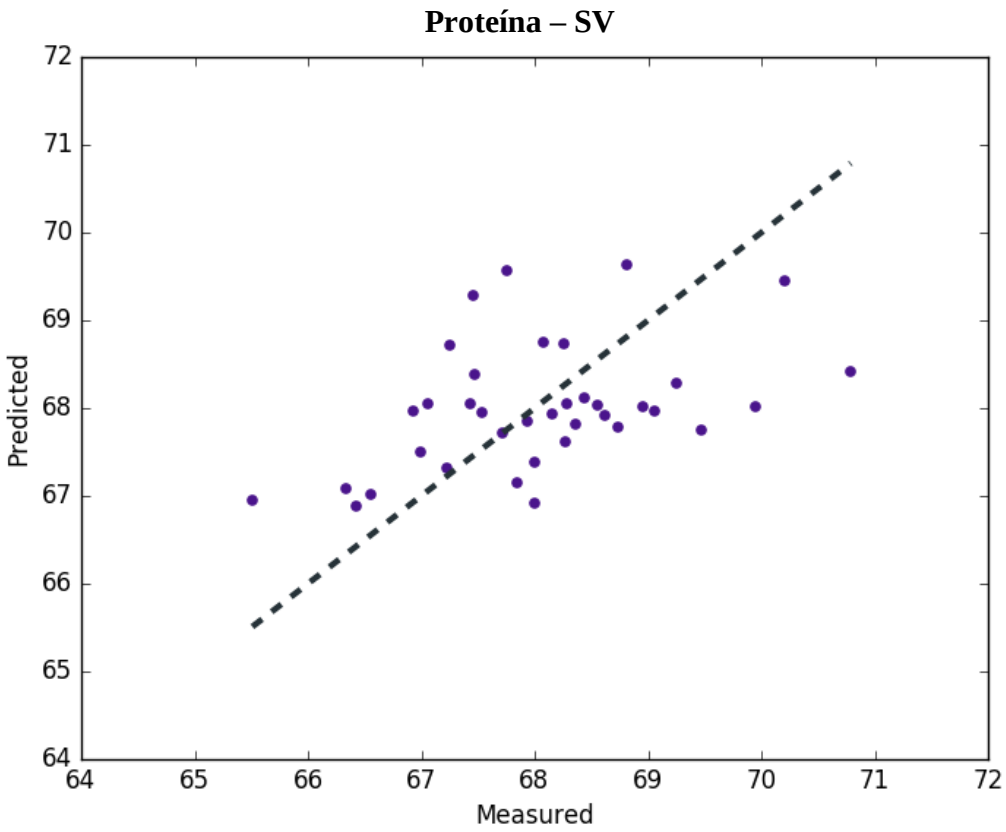

Figura 28 Gráfico de validación de modelo para Proteína
Fuente: Elaboración propia.

La figura 28 pertenece a un gráfico de validación cruzada para evaluación del grupo B, donde se aprecia la comparativa de los datos medidos y los datos calculados por modelo de regresión. Este grafico también incluye una línea de tendencia ideal. El modelo empleado se refiere Support Vector Regression para predecir proteína.

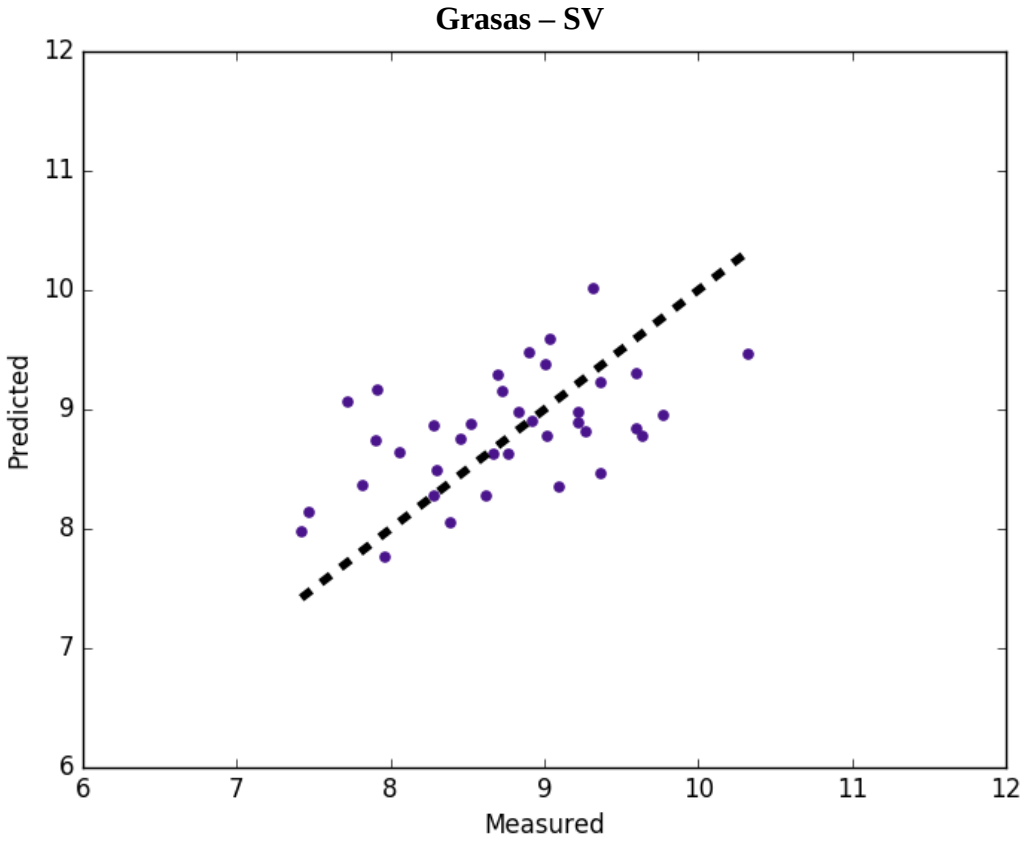

Figura 29 Gráfico de validación de modelo para Grasas
Fuente: Elaboración propia.

La figura 29 pertenece a un gráfico de validación cruzada para evaluación del grupo B, donde se aprecia la comparativa de los datos medidos y los datos calculados por modelo de regresión. Este grafico también incluye una línea de tendencia ideal. El modelo empleado se refiere Support Vector Regression para predecir grasas.

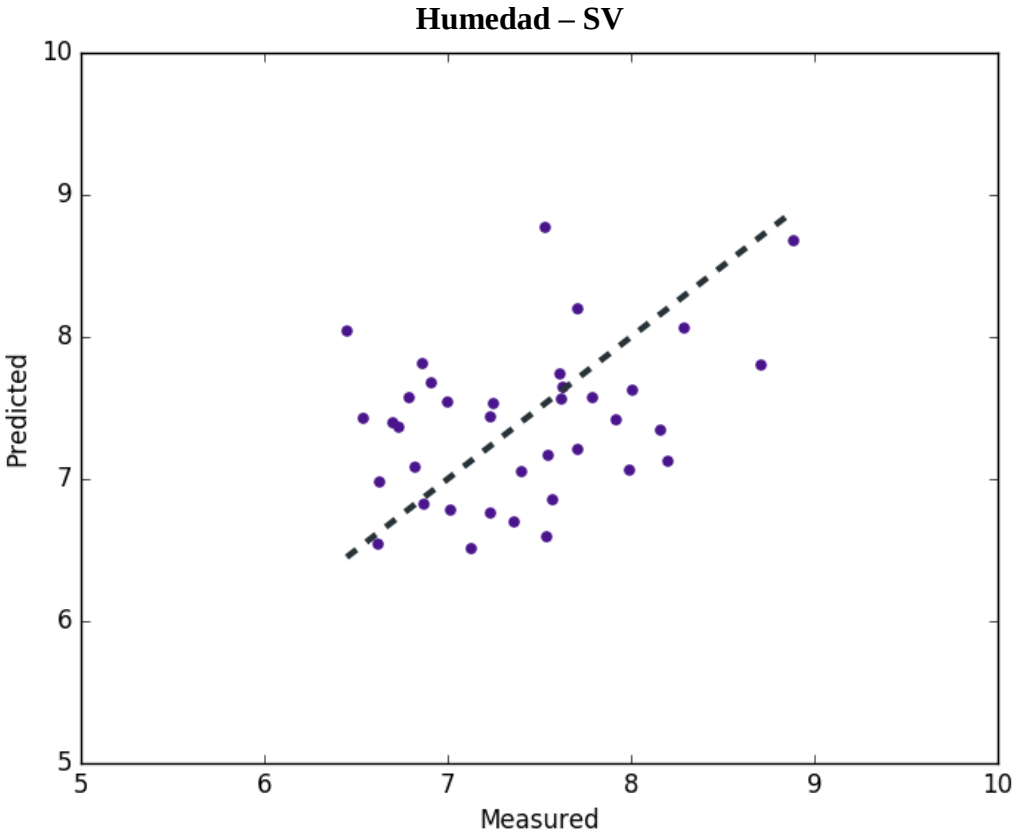

Figura 30 Gráfico de validación de modelo para Humedad
Fuente: Elaboración propia.

La figura 30 pertenece a un gráfico de validación cruzada para evaluación del grupo B, donde se aprecia la comparativa de los datos medidos y los datos calculados por modelo de regresión. Este grafico también incluye una línea de tendencia ideal. El modelo empleado se refiere Support Vector Regression para predecir humedad.

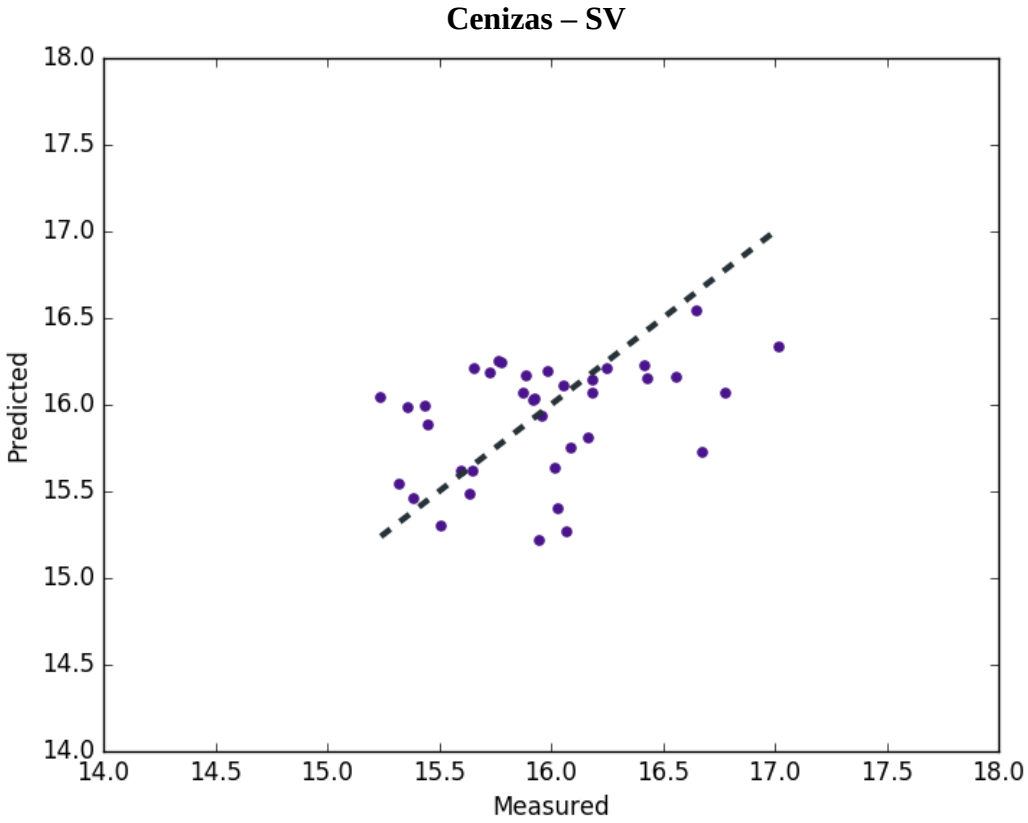

Figura 31 Gráfico de validación de modelo para Cenizas
Fuente: Elaboración propia.

La figura 31 pertenece a un gráfico de validación cruzada para evaluación del grupo B, donde se aprecia la comparativa de los datos medidos y los datos calculados por modelo de regresión. Este grafico también incluye una línea de tendencia ideal. El modelo empleado se refiere Support Vector Regression para predecir cenizas.

En la tabla 5 se presentan todos los parámetros que determinan la calidad de predicción del modelo de regresiones usando Support Vector.

Tabla 5. Resumen de validación cruzada para modelo de predicción

Parámetro	R	RMS	Error(%)
Proteína	0.44	1.01	1.23
Grasas	0.53	0.34	5.71
Humedad	0.34	0.43	7.45
Cenizas	0.37	0.19	2.17

Fuente: Elaboración propia.

4.3.2 Multilayer perceptron regression

Parámetros de sintonización para modelo de regresión usando Multilayer Perceptron:

La capa de entrada se utilizaron 60 neuronas para todos los parámetros, con función de activación “Lineal”. La capa oculta presenta diferentes configuraciones y se muestran en la tabla 6. La capa de salida presenta una solo una neurona para cada parámetro con función de activación “Lineal”.

Tabla 6. Parámetros de sintonización de capa oculta

Parámetro	Capas ocultas	Neuronas	Función de activación
Proteína	4	15 – 15 – 3	Lineal
Grasas	5	5 – 5 – 3	Lineal
Humedad	3	10 – 25 – 10	Tanh
Cenizas	3	15 – 15 – 6	Lineal

Fuente: Elaboración propia

Se observa que para todos los parámetros a identificar se tiene una topología de capas ocultas con un numero de neuronas relativamente bajo y que en su mayoría la función de activación corresponde a una función Lineal.

Visualización de modelos de regresión usando Multilayer Perceptron:

Para grupo A:

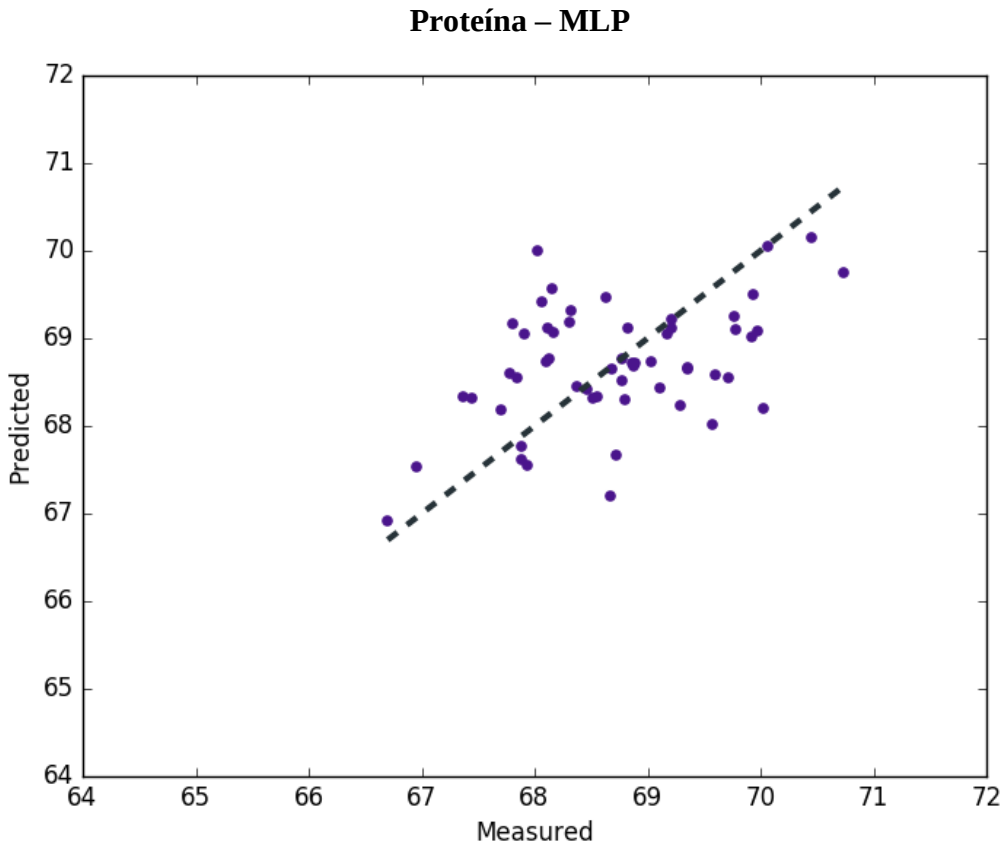

Figura 32 Gráfico de validación de modelo para Proteína
Fuente: Elaboración propia.

La figura 32 pertenece a un gráfico de validación cruzada para evaluación del grupo A, donde se aprecia la comparativa de los datos medidos y los datos calculados por modelo de regresión. Este grafico también incluye una línea de tendencia ideal. El modelo empleado se refiere Multilayer Perceptron para predecir proteína.

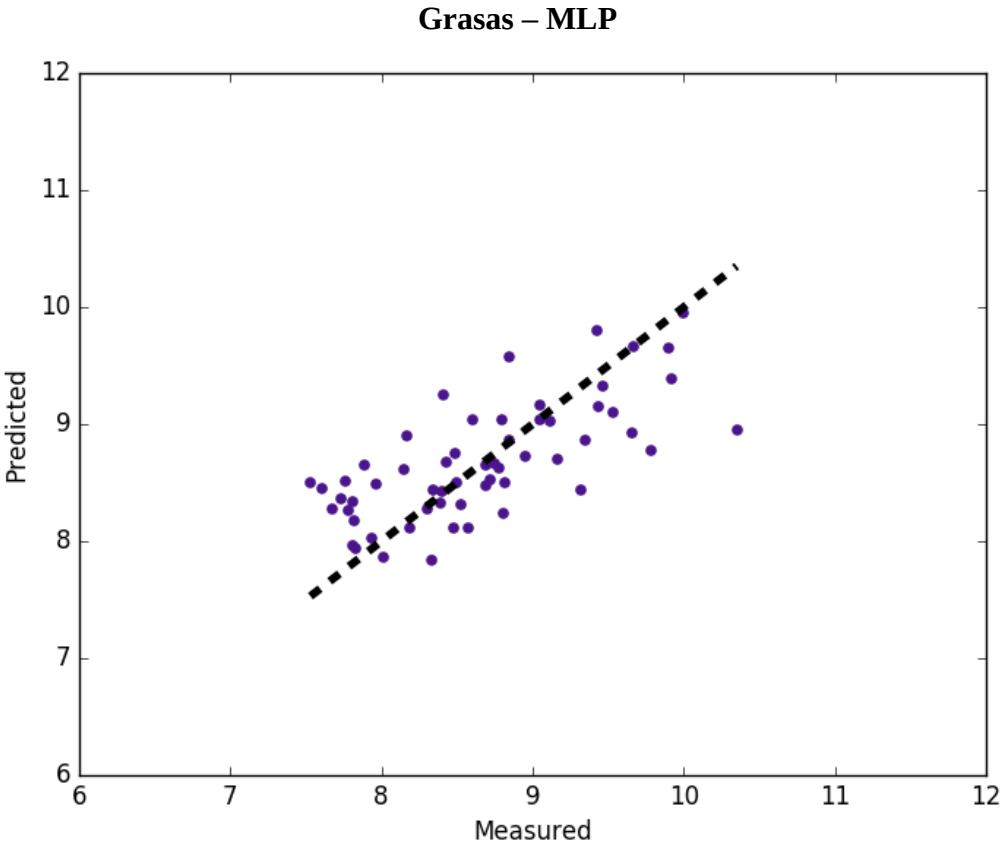

Figura 33 Gráfico de validación de modelo para Grasas
Fuente: Elaboración propia.

La figura 33 pertenece a un gráfico de validación cruzada para evaluación del grupo A, donde se aprecia la comparativa de los datos medidos y los datos calculados por modelo de regresión. Este grafico también incluye una línea de tendencia ideal. El modelo empleado se refiere Multilayer Perceptron para predecir grasas.

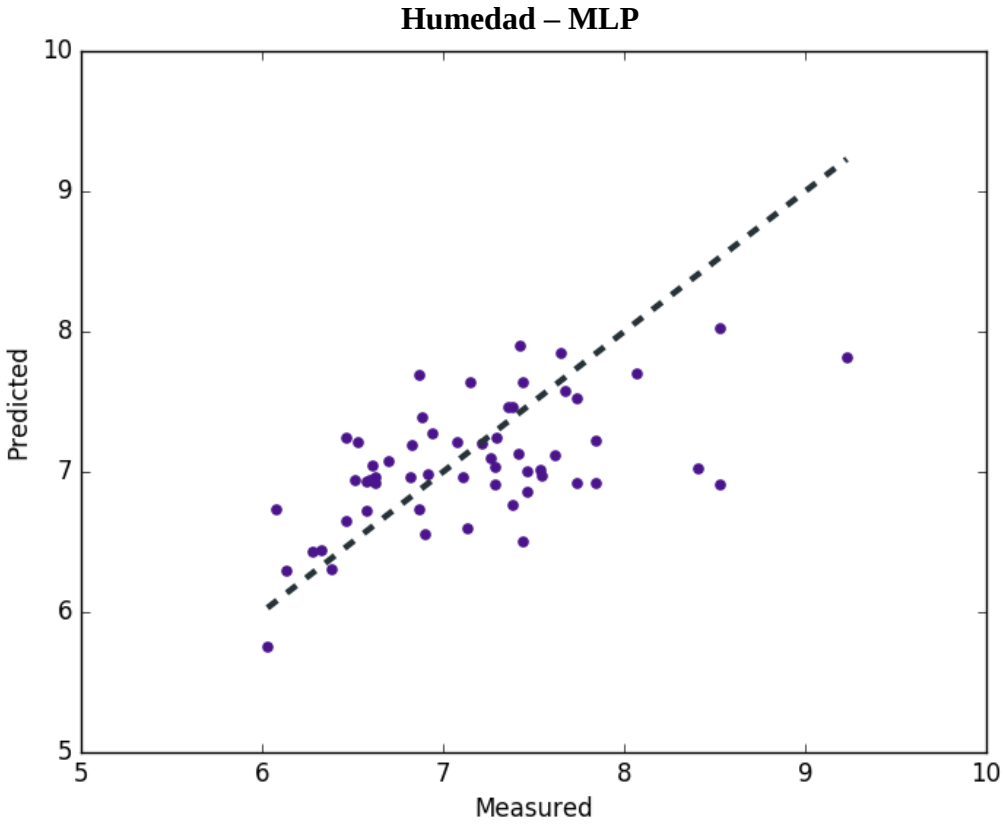

Figura 34 Gráfico de validación de modelo para Humedad
Fuente: Elaboración propia.

La figura 34 pertenece a un gráfico de validación cruzada para evaluación del grupo A, donde se aprecia la comparativa de los datos medidos y los datos calculados por modelo de regresión. Este grafico también incluye una línea de tendencia ideal. El modelo empleado se refiere Multilayer Perceptron para predecir humedad.

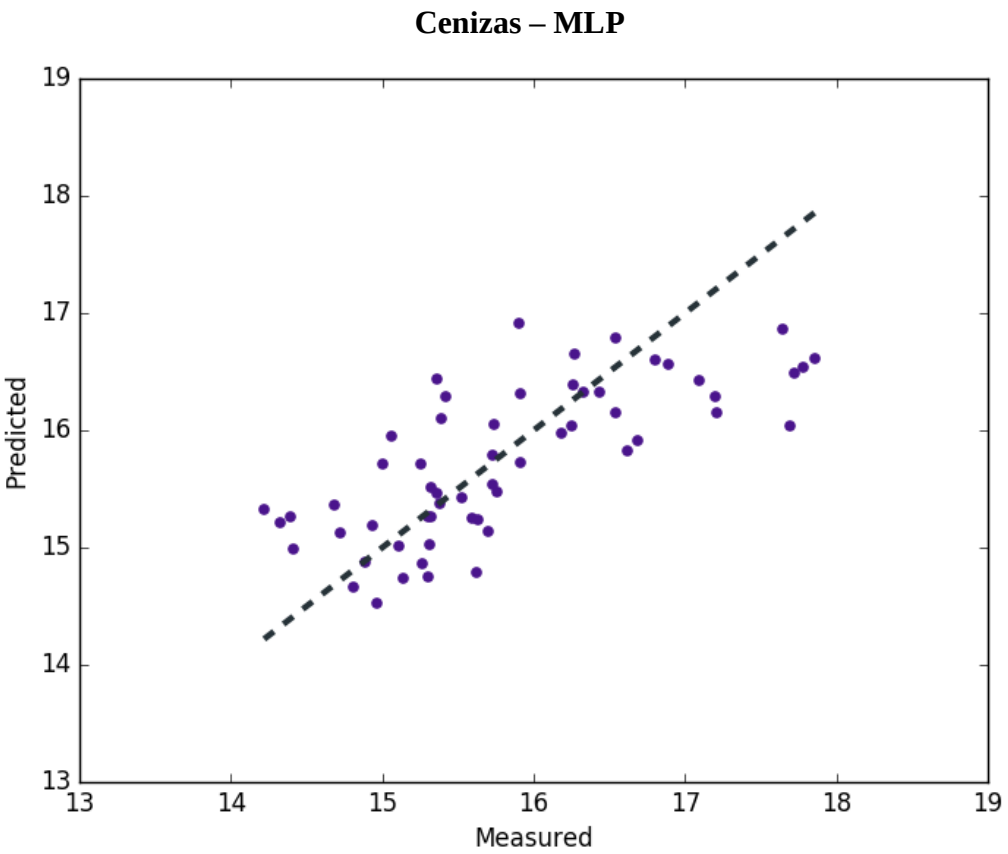

Figura 35 Gráfico de validación de modelo para Cenizas
Fuente: Elaboración propia.

La figura 35 pertenece a un gráfico de validación cruzada para evaluación del grupo A, donde se aprecia la comparativa de los datos medidos y los datos calculados por modelo de regresión. Este grafico también incluye una línea de tendencia ideal. El modelo empleado se refiere Multilayer Perceptron para predecir cenizas.

En la tabla 7 se presentan todos los parámetros que determinan la calidad de predicción del modelo de regresiones usando Multilayer Perceptron.

Tabla 7. Resumen de validación cruzada para modelo de predicción

Parámetro	R	RMS	Error(%)
Proteína	0.45	0.67	0.96
Grasas	0.71	0.24	4.37
Humedad	0.6	0.3	5.77
Cenizas	0.73	0.41	3.18

Autor: Elaboración propia.

Para grupo B:

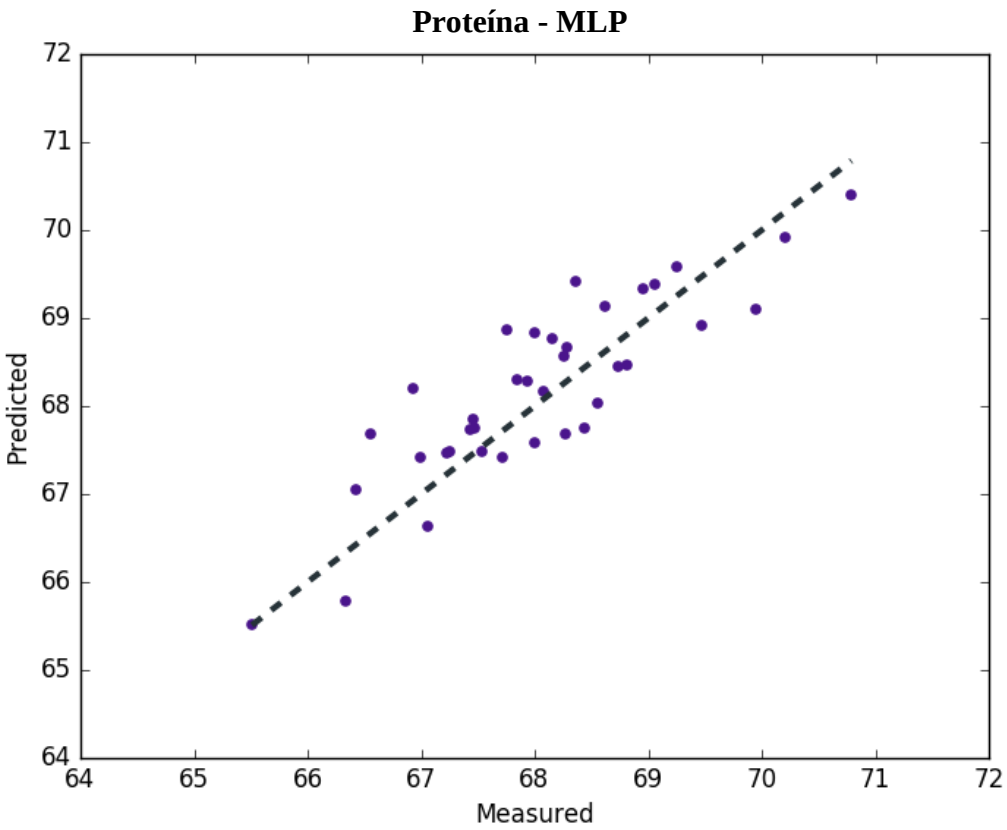

Figura 36 Gráfico de validación de modelo para Proteína
Fuente: Elaboración propia.

La figura 36 pertenece a un gráfico de validación cruzada para evaluación del grupo B, donde se aprecia la comparativa de los datos medidos y los datos calculados por modelo de regresión. Este grafico también incluye una línea de tendencia ideal. El modelo empleado se refiere Multilayer Perceptron para predecir proteína.

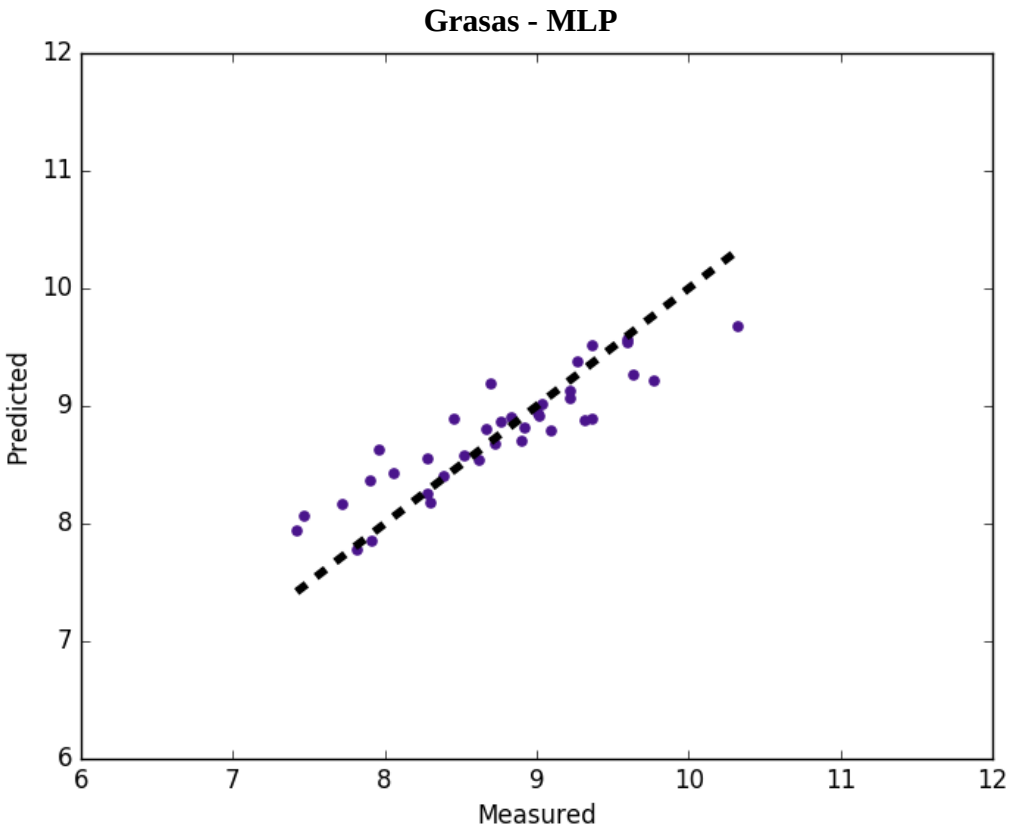

Figura 37 Gráfico de validación de modelo para Grasas
Fuente: Elaboración propia.

La figura 37 pertenece a un gráfico de validación cruzada para evaluación del grupo B, donde se aprecia la comparativa de los datos medidos y los datos calculados por modelo de regresión. Este grafico también incluye una línea de tendencia ideal. El modelo empleado se refiere Multilayer Perceptron para predecir grasas.

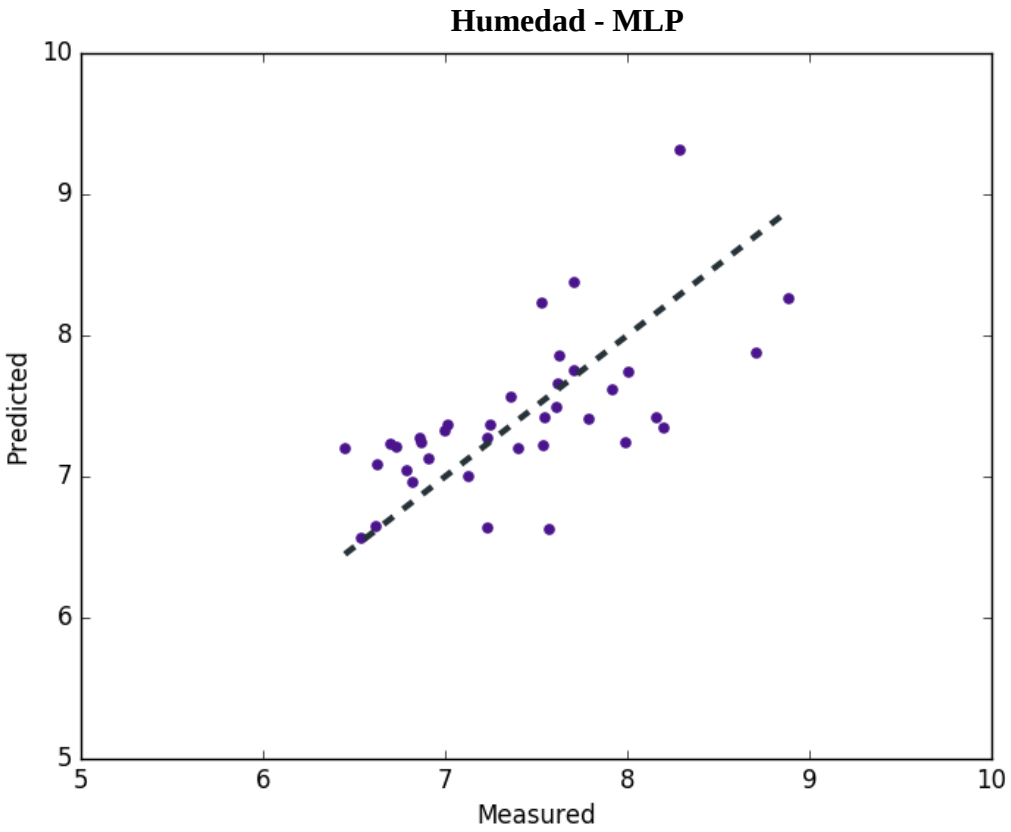

Figura 38 Gráfico de validación de modelo para Humedad
Fuente: Elaboración propia.

La figura 38 pertenece a un gráfico de validación cruzada para evaluación del grupo B, donde se aprecia la comparativa de los datos medidos y los datos calculados por modelo de regresión. Este grafico también incluye una línea de tendencia ideal. El modelo empleado se refiere Multilayer Perceptron para predecir humedad.

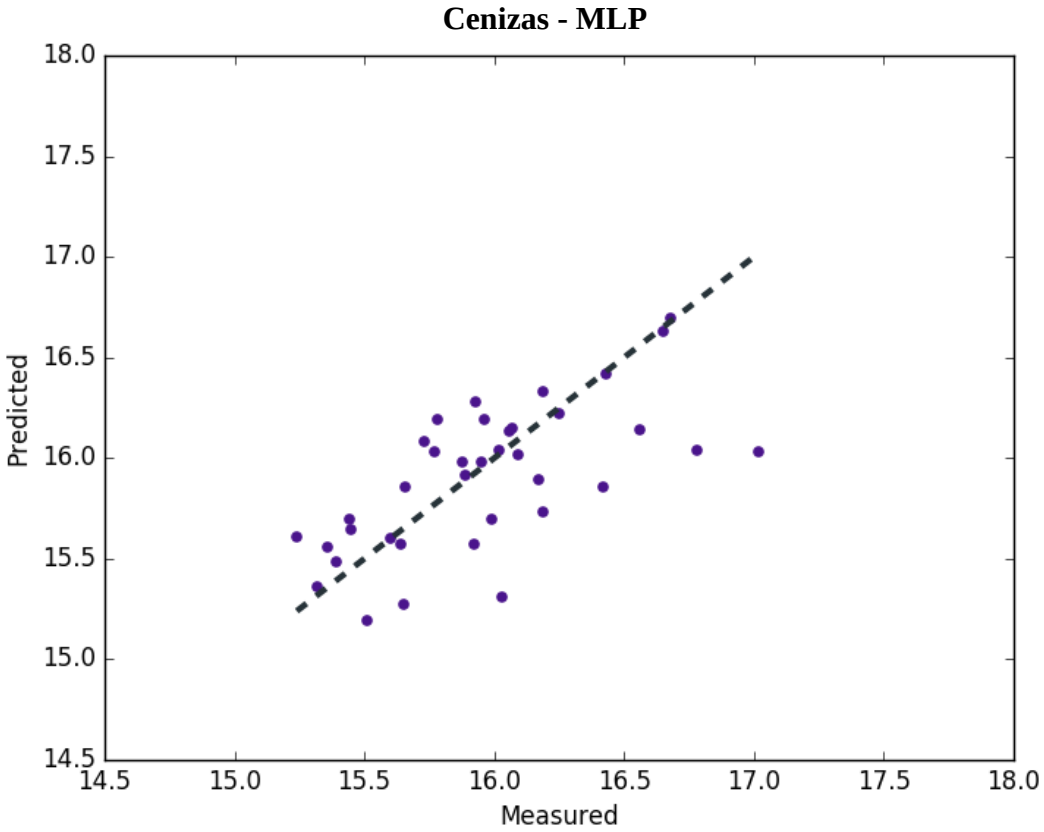

Figura 39 Gráfico de validación de modelo para Cenizas
Fuente: Elaboración propia.

La figura 39 pertenece a un gráfico de validación cruzada para evaluación del grupo B, donde se aprecia la comparativa de los datos medidos y los datos calculados por modelo de regresión. Este grafico también incluye una línea de tendencia ideal. El modelo empleado se refiere Multilayer Perceptron para predecir cenizas.

En la tabla 8 se presentan todos los parámetros que determinan la calidad de predicción del modelo de regresiones usando Multilayer Perceptron.

Tabla 8. Resumen de validación cruzada para modelo de predicción

Parámetro	R	RMS	Error(%)
Proteína	0.89	0.32	0.71
Grasas	0.9	0.1	2.79
Humedad	0.67	0.22	5.12
Cenizas	0.67	0.11	1.54

Autor: Elaboración propia.

4.4 Comprobaciones estadísticas

La comprobación estadística presentada en el presente apartado tiene como objetivo confirmar la calidad de predicción de los modelos descritos anteriormente, con esta información es posible determinar el modelo más adecuado para cada parámetro, en el Capítulo 5 se presentan los modelos escogidos según criterios mencionados anteriormente.

Se toma como muestra de datos al error del modelo de predicción, es decir se obtiene el error que existe entre en los valores calculados por modelo y los valores de referencia. Este error es estudiado encontrando la distribución que forma, con esta información es posible determinar la robustez que posee el sistema de predicción de parámetros.

4.4.1 Support vector regression

Para grupo A:

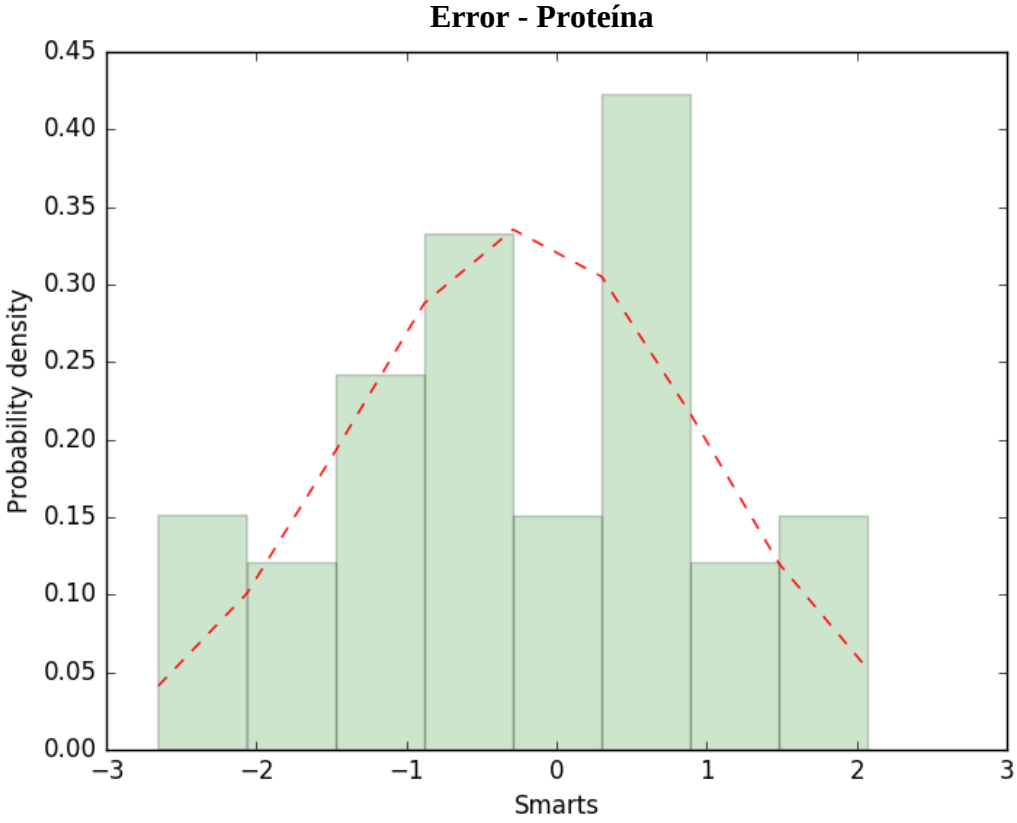

Figura 40 Gráfico de distribución de error de modelo de predicción para Proteína
Fuente: Elaboración propia.

El grafico mostrado en la figura 40 presenta la distribución que forma los errores generados entre las muestras del grupo A de referencia y obtenidas por los modelos de regresión Support Vector Regression para proteína. Este grafico cuenta con un tamaño de muestra 10 y media -0.3.

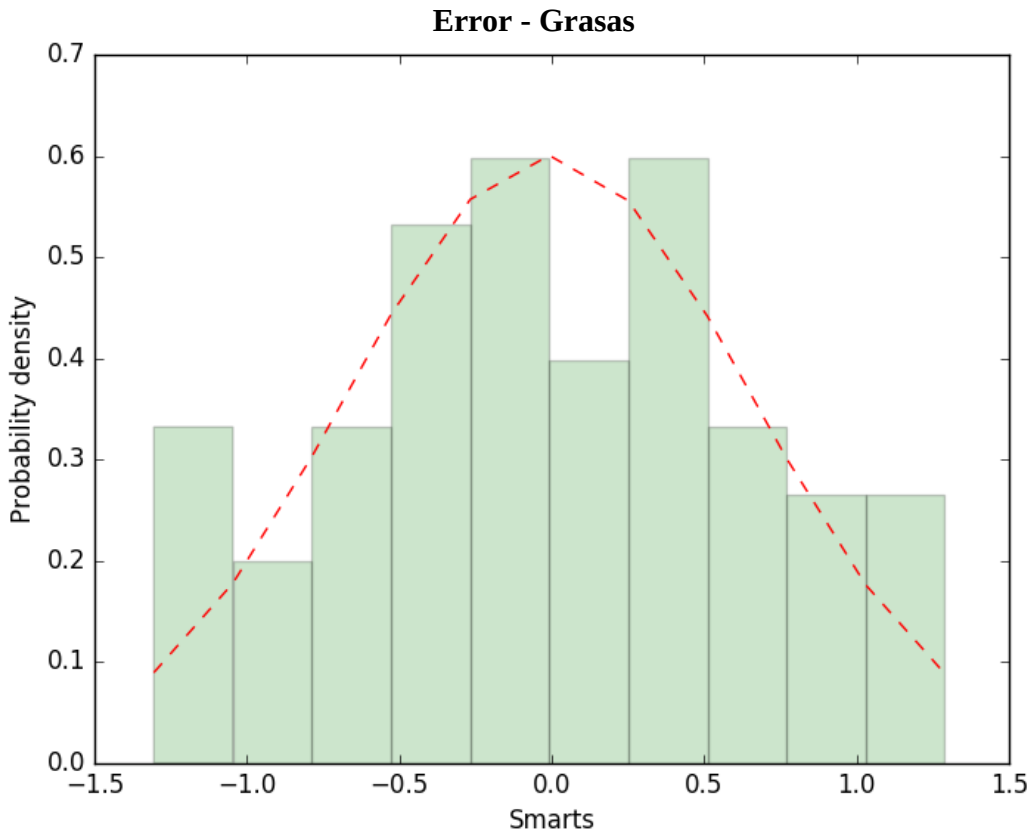

Figura 41 Gráfico de distribución de error de modelo de predicción para Grasas
Fuente: Elaboración propia.

El grafico mostrado en la figura 41 presenta la distribución que forma los errores generados entre las muestras del grupo A de referencia y obtenidas por los modelos de regresión Support Vector Regression para grasas. Este grafico cuenta con un tamaño de muestra 10 y media 0.

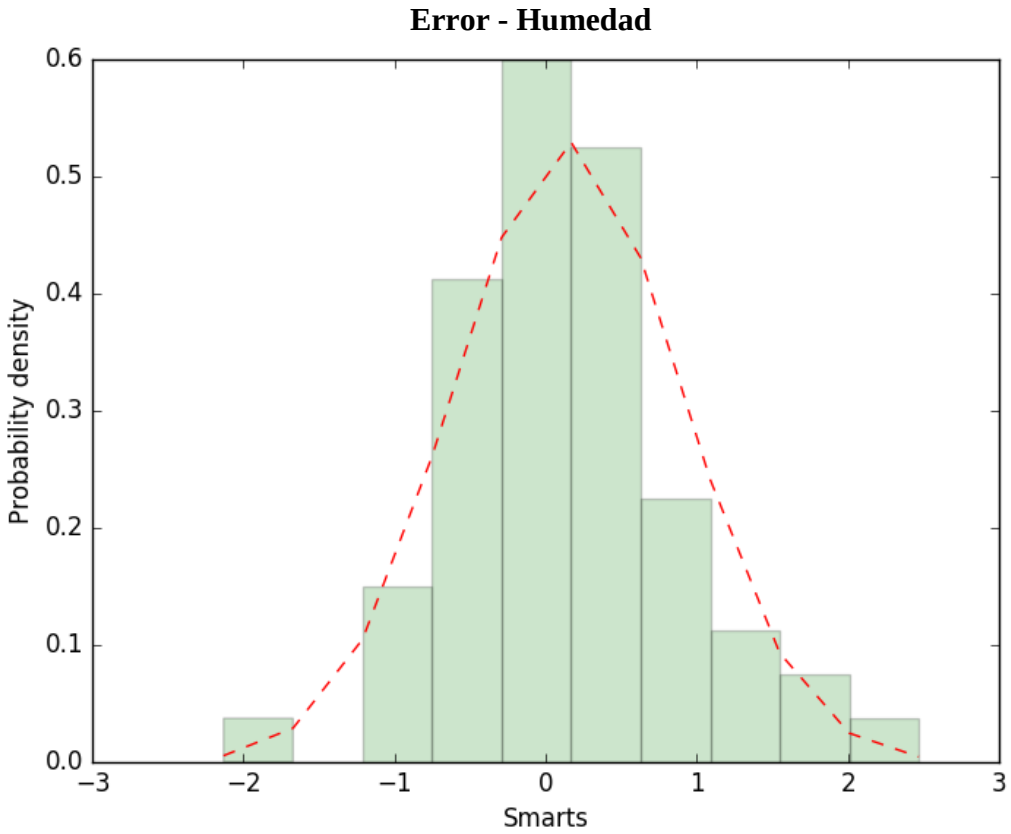

Figura 42 Gráfico de distribución de error de modelo de predicción para Humedad
Fuente: Elaboración propia.

El grafico mostrado en la figura 42 presenta la distribución que forma los errores generados entre las muestras del grupo A de referencia y obtenidas por los modelos de regresión Support Vector Regression para humedad. Este grafico cuenta con un tamaño de muestra 10 y media 0.

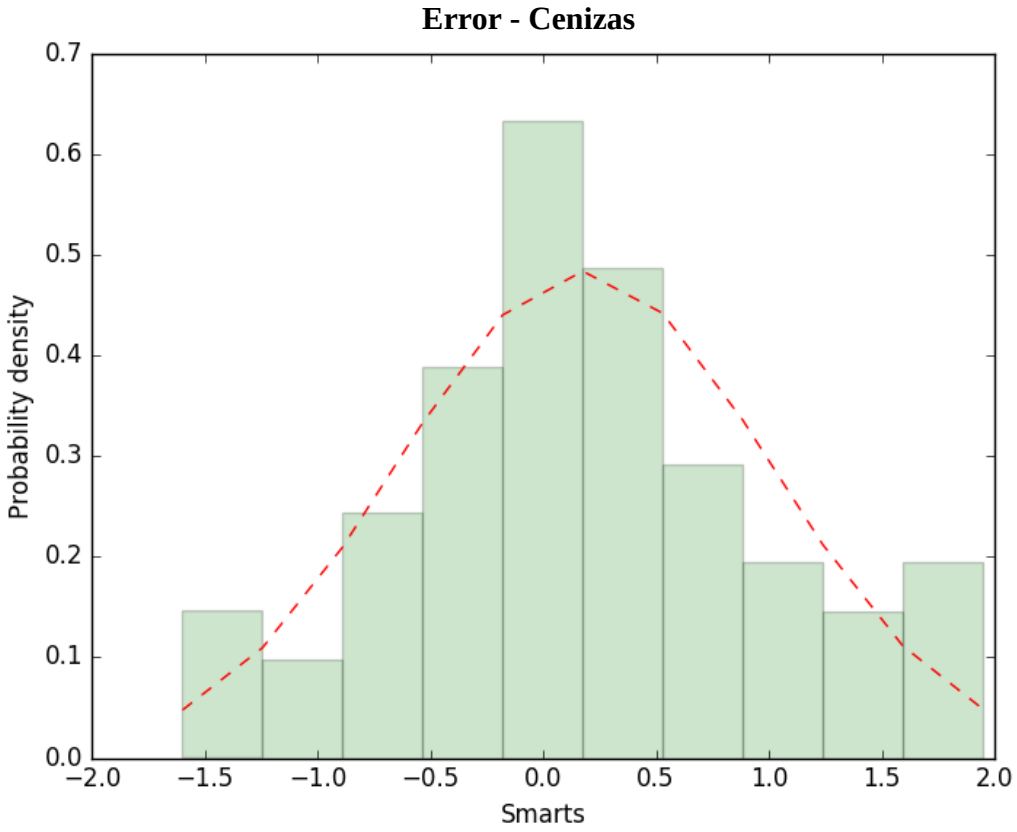

Figura 43 Gráfico de distribución de error de modelo de predicción para Cenizas
Fuente: Elaboración propia.

El grafico mostrado en la figura 43 presenta la distribución que forma los errores generados entre las muestras del grupo A de referencia y obtenidas por los modelos de regresión Support Vector Regression para cenizas. Este grafico cuenta con un tamaño de muestra 10 y media 0.25.

En la tabla 9 se presentan los resultados de la prueba Shapiro Wilk.

Tabla 9. Resumen de prueba de ajuste Shapiro Wilk

Parámetro	P-value	W
Proteína	0.38	0.97
Grasas	0.51	0.98
Humedad	0.09	0.96
Cenizas	0.75	0.98

Autor: Elaboración propia.

Para grupo B:

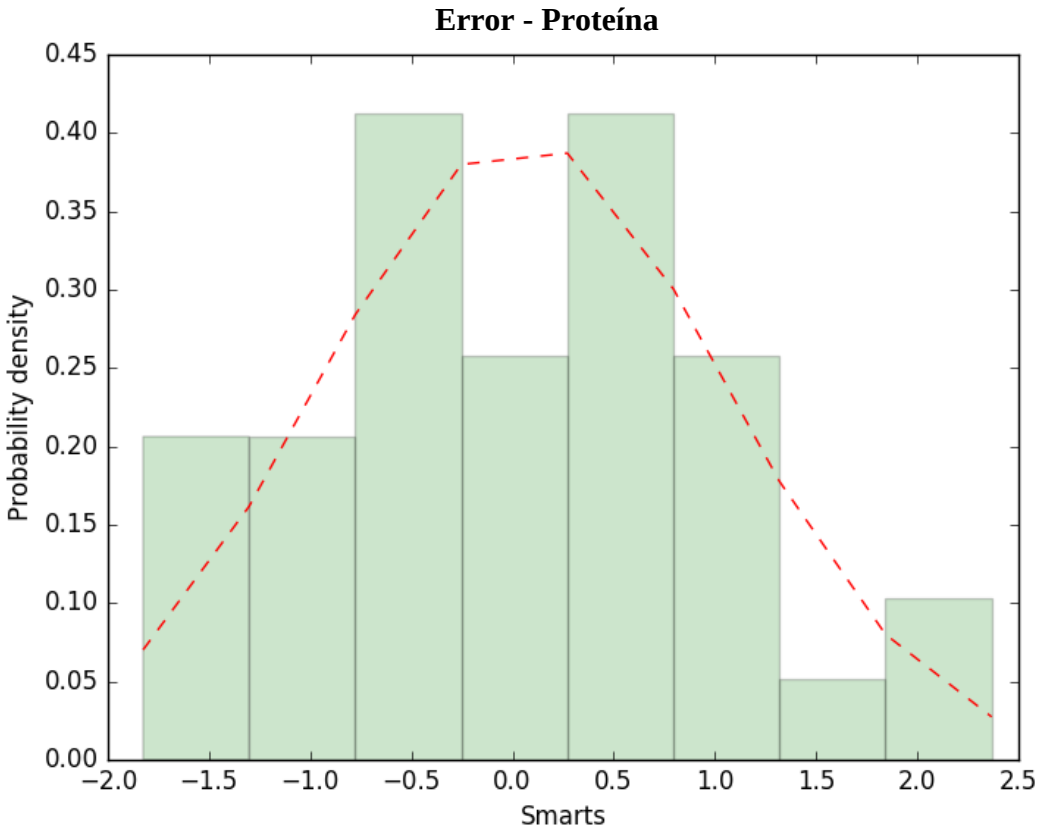

Figura 44 Gráfico de distribución de error de modelo de predicción para Proteína
Fuente: Elaboración propia.

El grafico mostrado en la figura 44 presenta la distribución que forma los errores generados entre las muestras del grupo B de referencia y obtenidas por los modelos de regresión Support Vector Regression para proteína. Este grafico cuenta con un tamaño de muestra 10 y media 0.25.

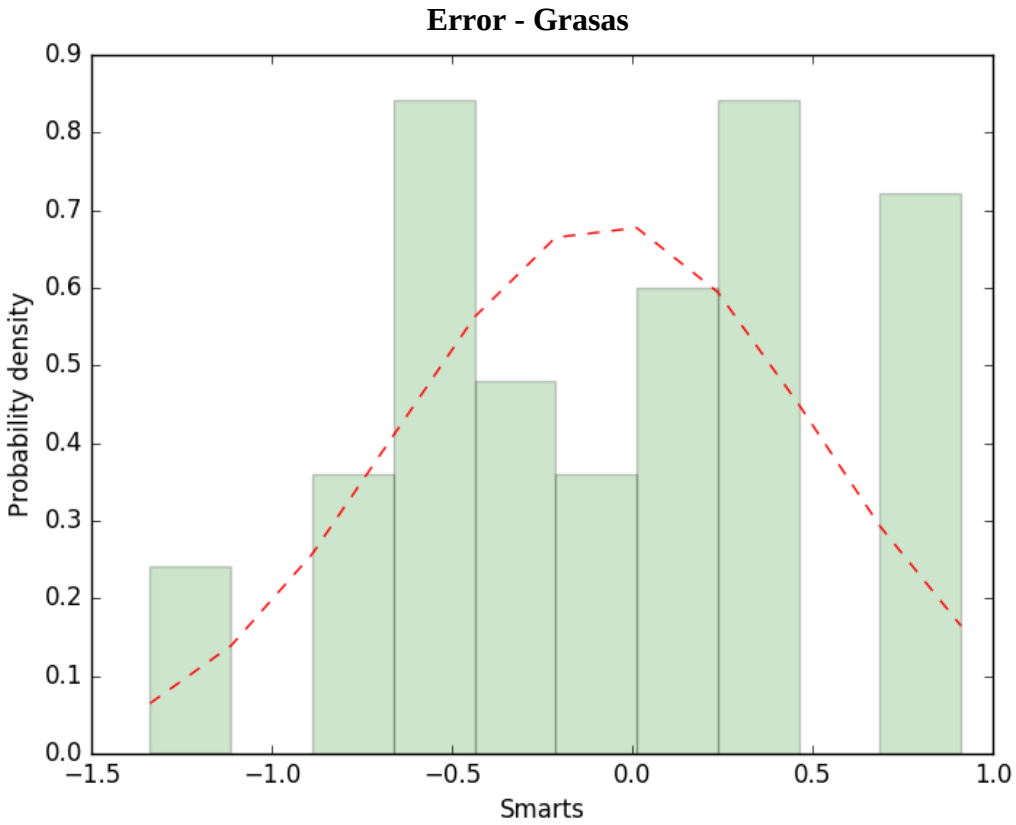

Figura 45 Gráfico de distribución de error de modelo de predicción para Grasas
Fuente: Elaboración propia.

El grafico mostrado en la figura 45 presenta la distribución que forma los errores generados entre las muestras del grupo B de referencia y obtenidas por los modelos de regresión Support Vector Regression para grasas. Este grafico cuenta con un tamaño de muestra 10 y media -0.25.

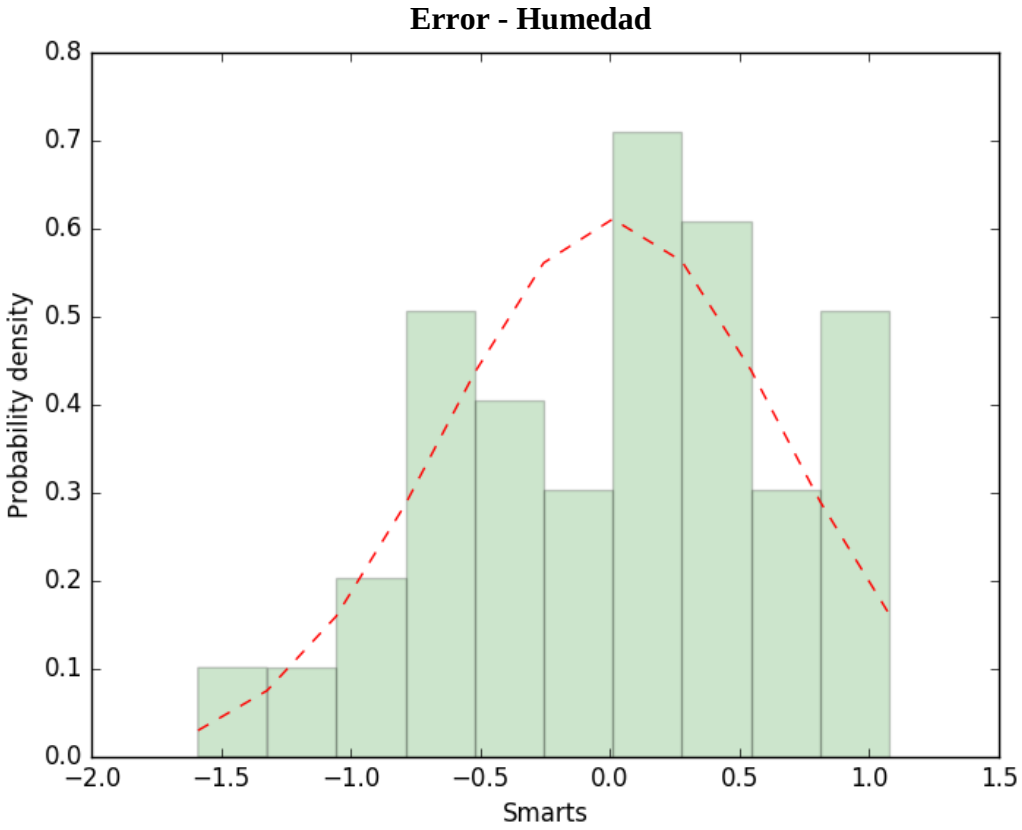

Figura 46 Gráfico de distribución de error de modelo de predicción para Humedad
Fuente: Elaboración propia.

El grafico mostrado en la figura 46 presenta la distribución que forma los errores generados entre las muestras del grupo B de referencia y obtenidas por los modelos de regresión Support Vector Regression para humedad. Este grafico cuenta con un tamaño de muestra 10 y media 0.

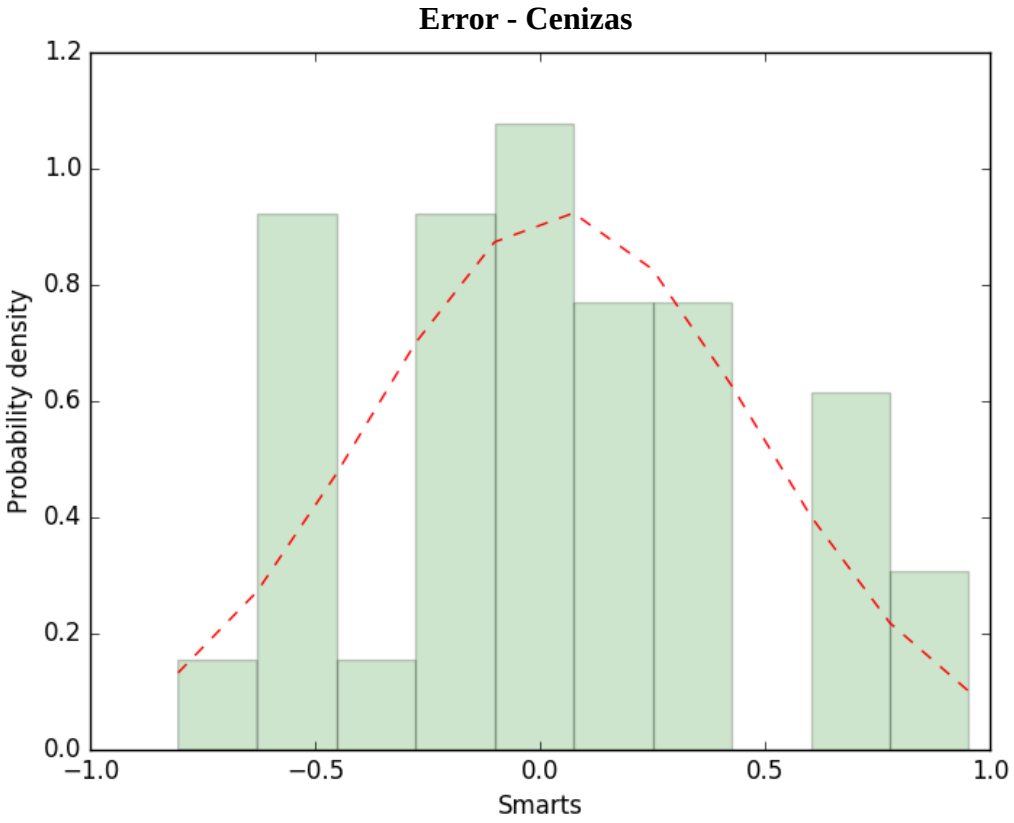

Figura 47 Gráfico de distribución de error de modelo de predicción para Cenizas
Fuente: Elaboración propia.

El grafico mostrado en la figura 47 presenta la distribución que forma los errores generados entre las muestras del grupo B de referencia y obtenidas por los modelos de regresión Support Vector Regression para cenizas. Este grafico cuenta con un tamaño de muestra 10 y media 0.1.

En la tabla 10 se presentan los resultados de la prueba Shapiro Wilk.

Tabla 10. Resumen de prueba de ajuste Shapiro Wilk

Parámetro	P-value	W
Proteína	0.71	0.97
Grasas	0.21	0.96
Humedad	0.42	0.97
Cenizas	0.69	0.97

Fuente: Elaboración propia.

4.4.2 Multilayer perceptron regression

Para grupo A:

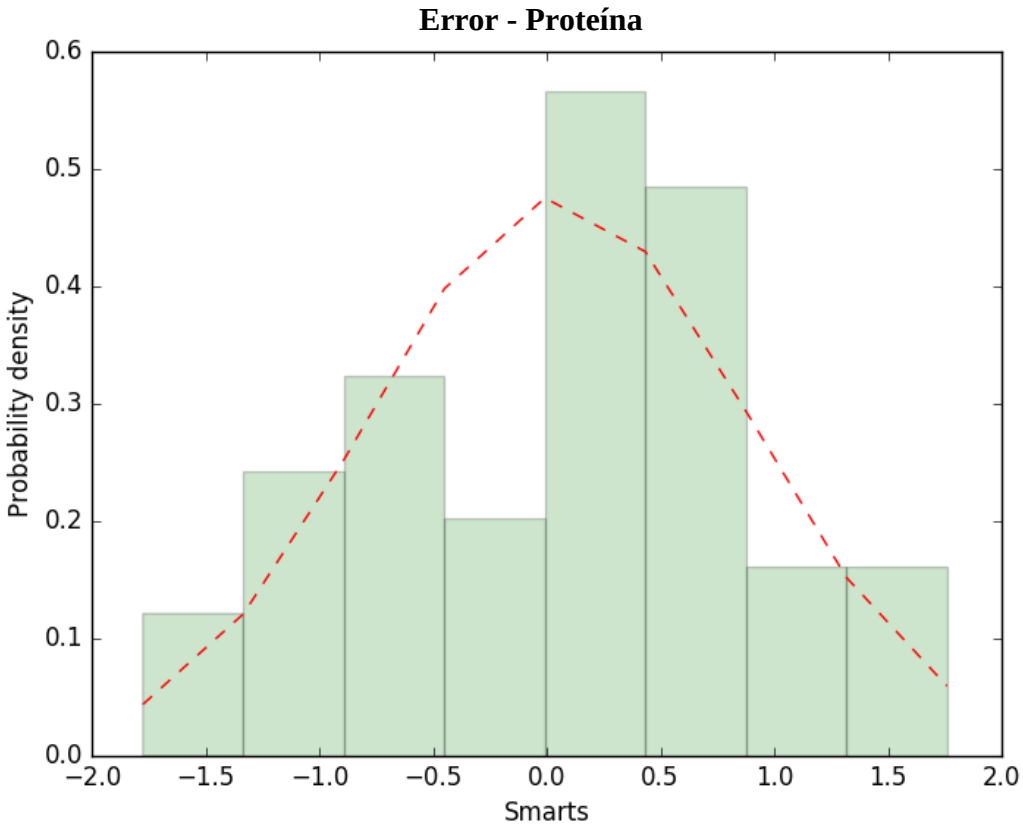

Figura 48 Gráfico de distribución de error de modelo de predicción para Proteína
Fuente: Elaboración propia.

El grafico mostrado en la figura 48 presenta la distribución que forma los errores generados entre las muestras del grupo A de referencia y obtenidas por los modelos de regresión Multilayer Perceptron Regression para proteína. Este grafico cuenta con un tamaño de muestra 10 y media 0.

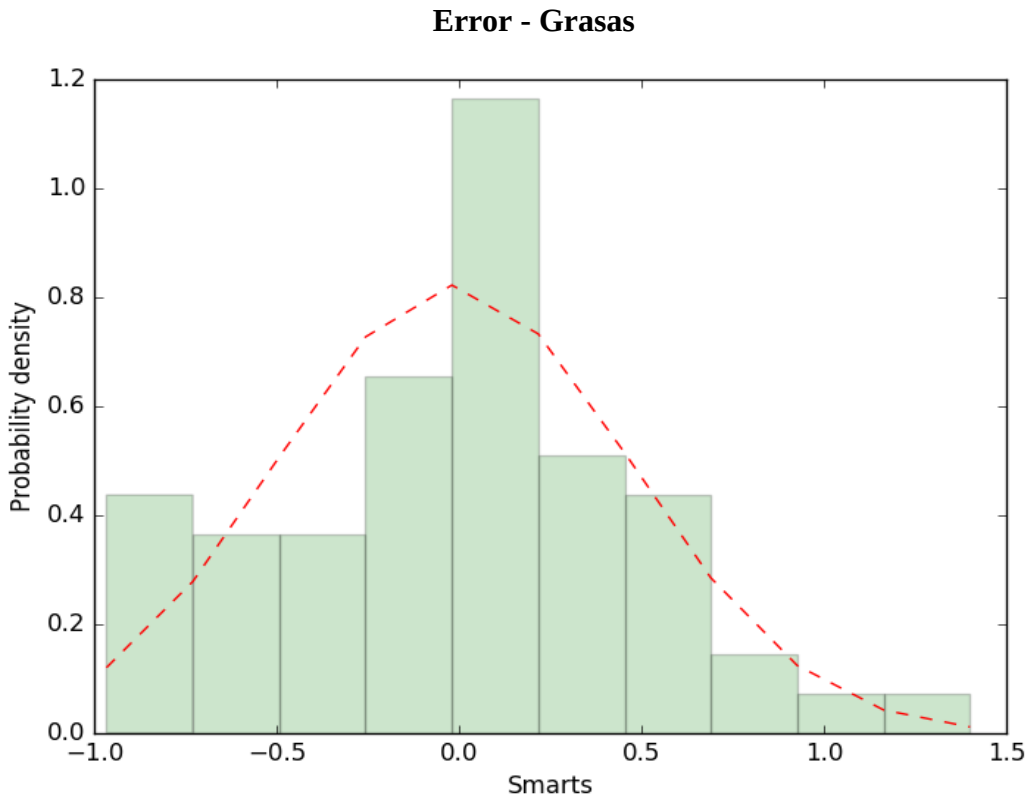

Figura 49 Gráfico de distribución de error de modelo de predicción para Grasas
Fuente: Elaboración propia.

El grafico mostrado en la figura 49 presenta la distribución que forma los errores generados entre las muestras del grupo A de referencia y obtenidas por los modelos de regresión Multilayer Perceptron Regression para grasas. Este grafico cuenta con un tamaño de muestra 10 y media -0.05.

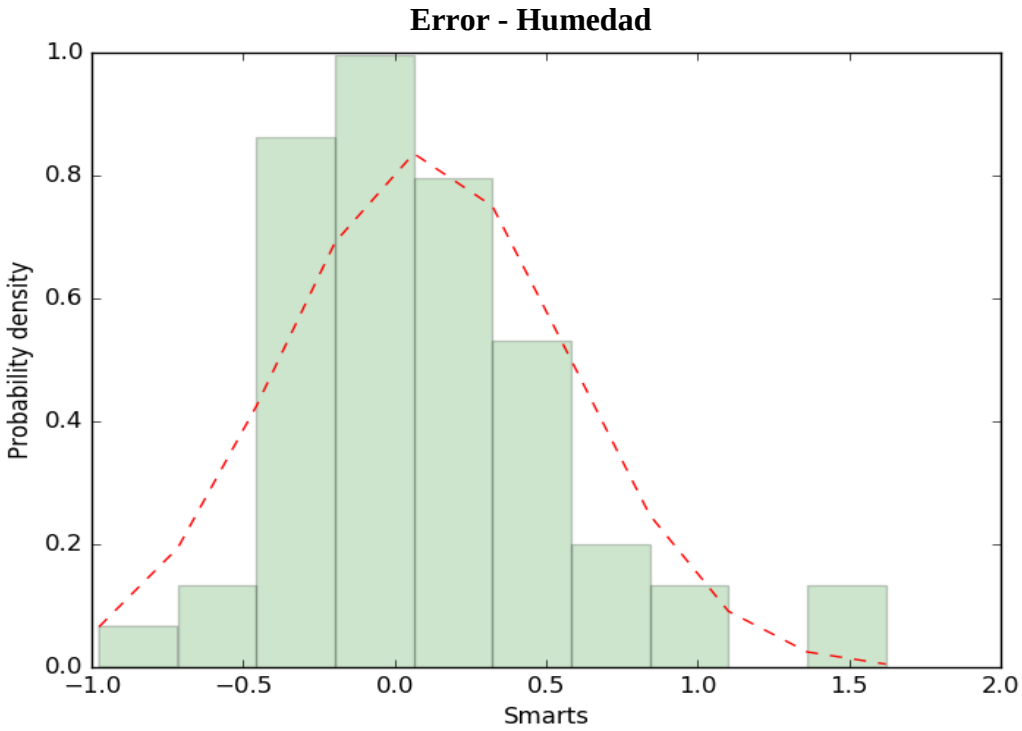

Figura 50 Gráfico de distribución de error de modelo de predicción para Humedad
Fuente: Elaboración propia.

El grafico mostrado en la figura 50 presenta la distribución que forma los errores generados entre las muestras del grupo A de referencia y obtenidas por los modelos de regresión Multilayer Perceptron Regression para humedad. Este grafico cuenta con un tamaño de muestra 10 y media 0.05.

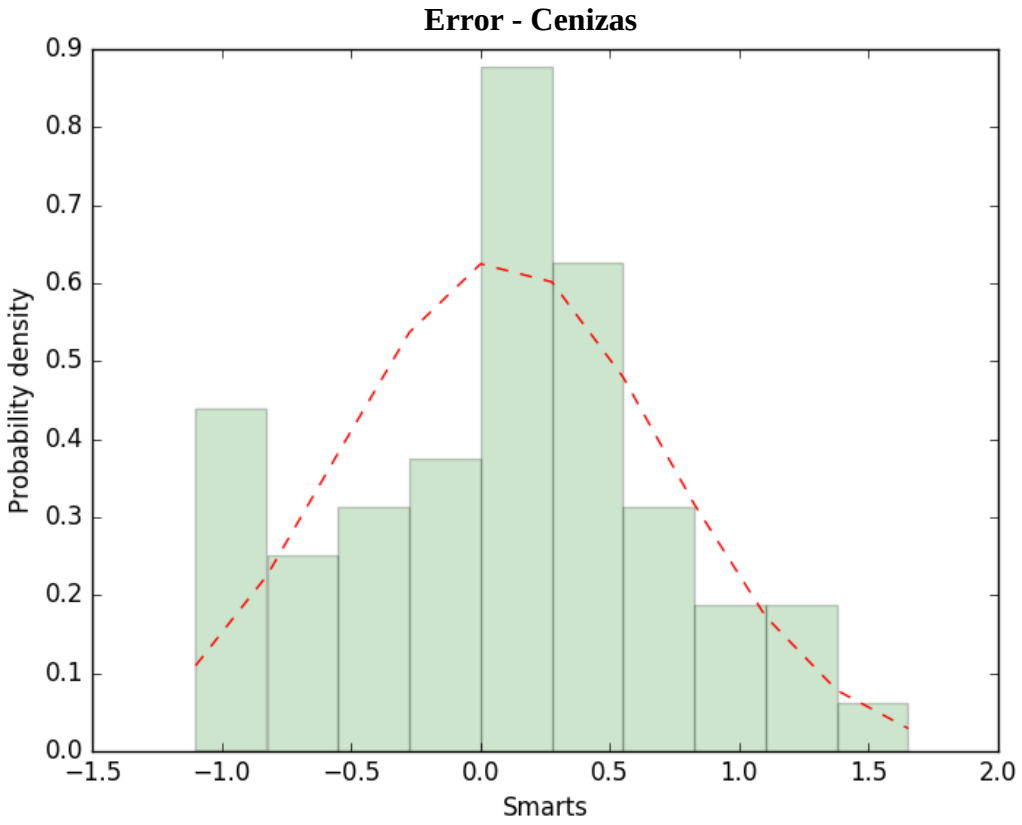

Figura 51 Gráfico de distribución de error de modelo de predicción para Cenizas
Fuente: Elaboración propia.

El grafico mostrado en la figura 51 presenta la distribución que forma los errores generados entre las muestras del grupo A de referencia y obtenidas por los modelos de regresión Multilayer Perceptron Regression para cenizas. Este grafico cuenta con un tamaño de muestra 10 y media 0.

En la tabla 11 se presentan los resultados de la prueba Shapiro Wilk.

Tabla 11. Resumen de prueba de ajuste Shapiro Wilk

Parámetro	P-value	W
Proteína	0.72	0.98
Grasas	0.41	0.97
Humedad	0.03	0.95
Cenizas	0.51	0.98

Fuente: Elaboración propia.

Para grupo B:

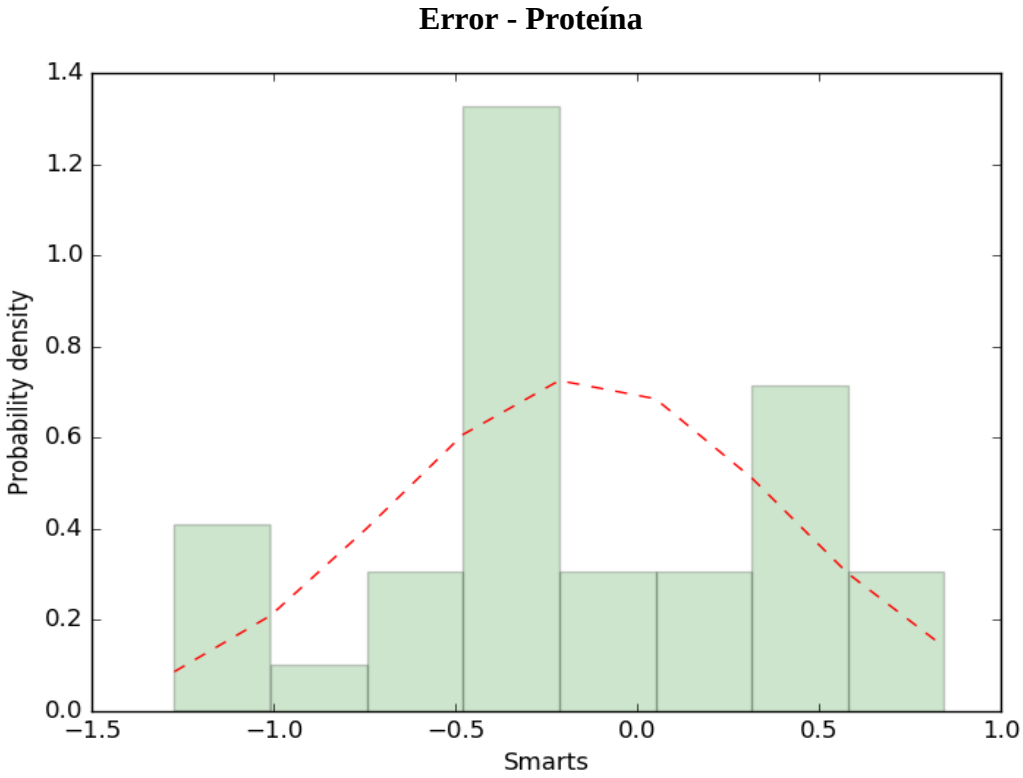

Figura 52 Gráfico de distribución de error de modelo de predicción para Proteína
Fuente: Elaboración propia.

El grafico mostrado en la figura 52 presenta la distribución que forma los errores generados entre las muestras del grupo B de referencia y obtenidas por los modelos de regresión Multilayer Perceptron Regression para proteína. Este grafico cuenta con un tamaño de muestra 10 y media -0.2.

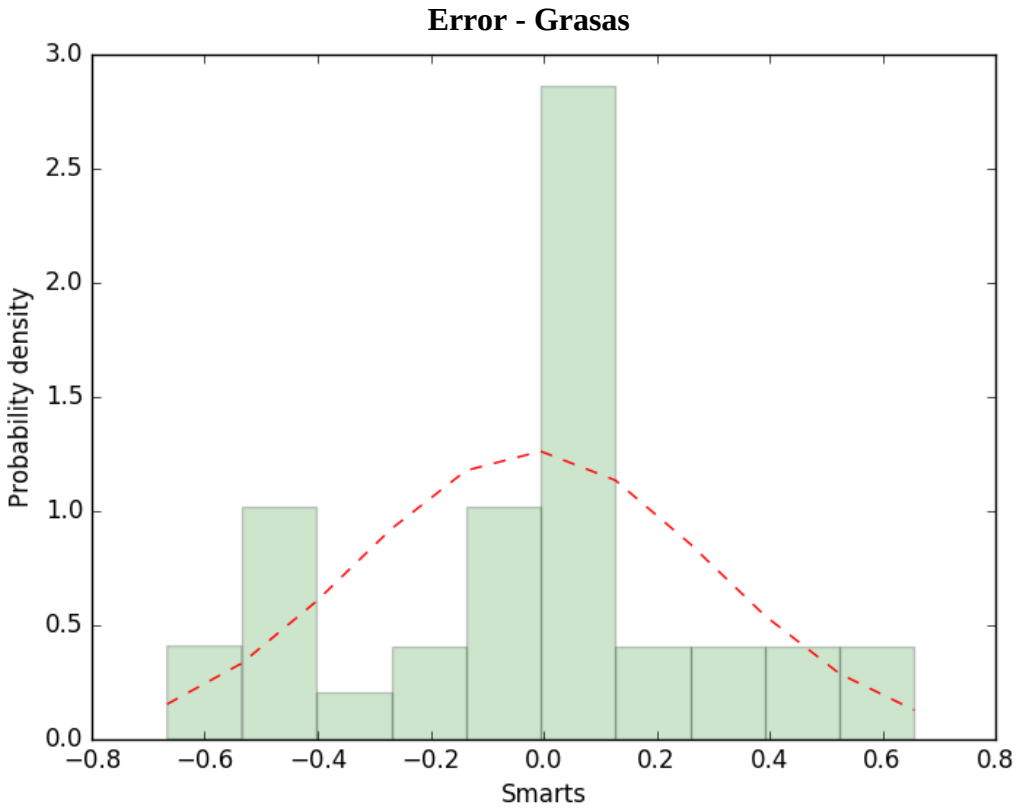

Figura 53 Gráfico de distribución de error de modelo de predicción para Grasas
Fuente: Elaboración propia.

El grafico mostrado en la figura 53 presenta la distribución que forma los errores generados entre las muestras del grupo B de referencia y obtenidas por los modelos de regresión Multilayer Perceptron Regression para grasas. Este grafico cuenta con un tamaño de muestra 10 y media 0.

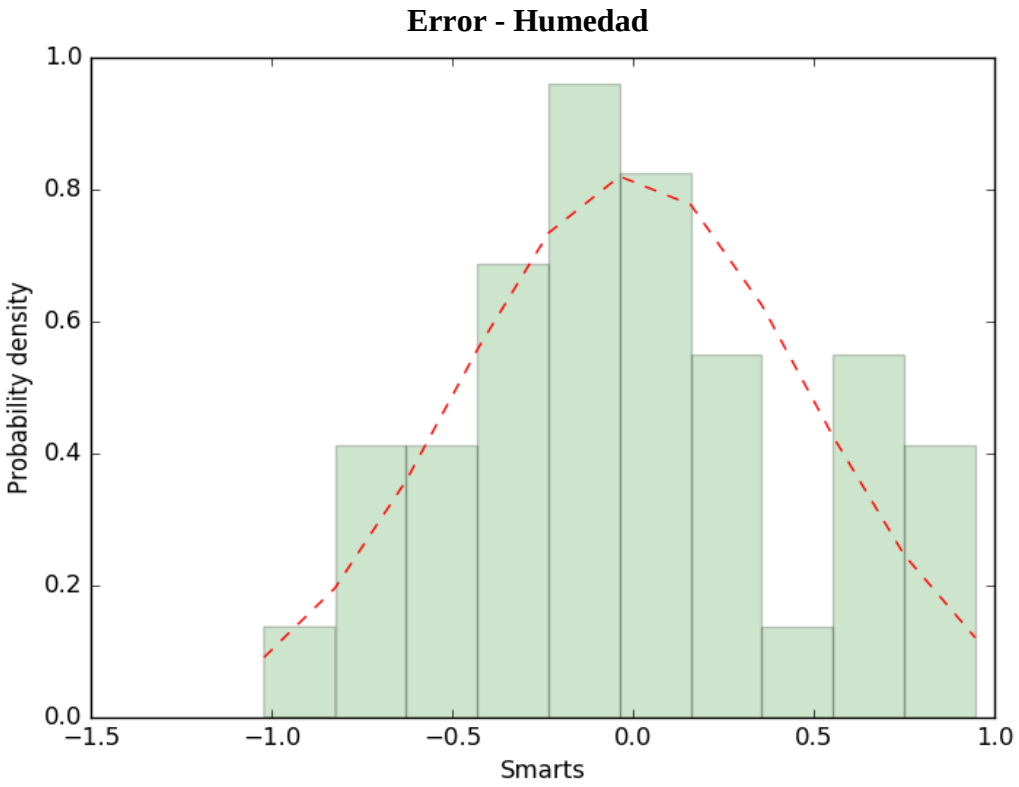

Figura 54 Gráfico de distribución de error de modelo de predicción para Humedad
Fuente: Elaboración propia.

El grafico mostrado en la figura 54 presenta la distribución que forma los errores generados entre las muestras del grupo B de referencia y obtenidas por los modelos de regresión Multilayer Perceptron Regression para humedad. Este grafico cuenta con un tamaño de muestra 10 y media -0.05.

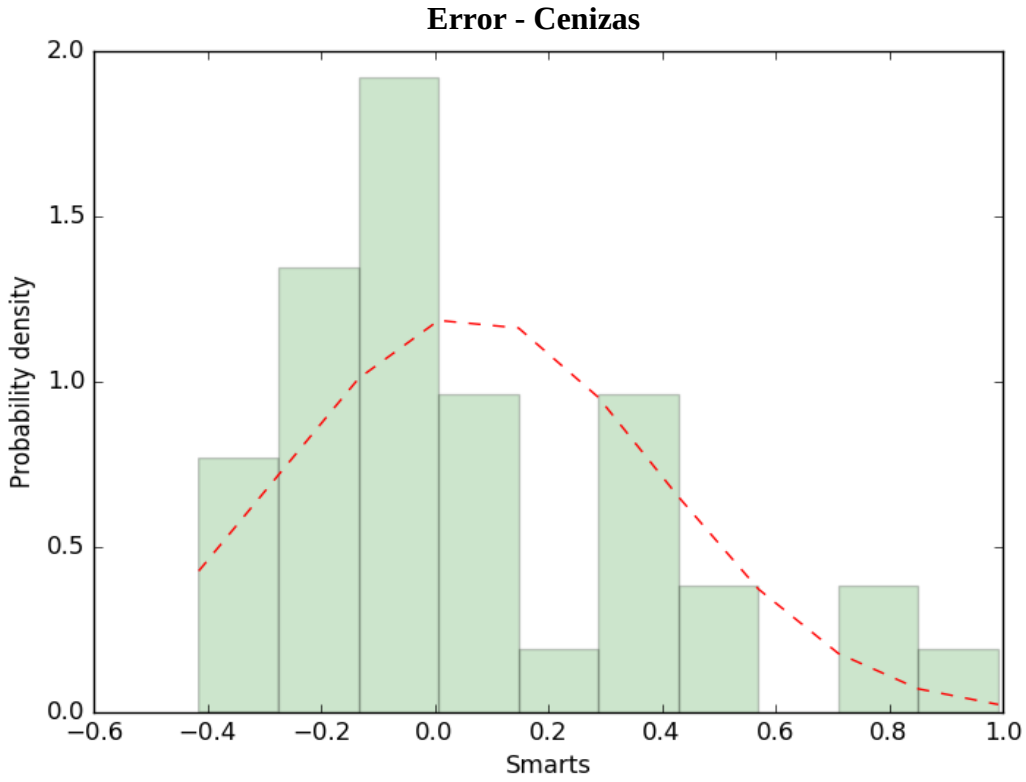

Figura 55 Gráfico de distribución de error de modelo de predicción para Cenizas
Fuente: Elaboración propia.

El grafico mostrado en la figura 55 presenta la distribución que forma los errores generados entre las muestras del grupo B de referencia y obtenidas por los modelos de regresión Multilayer Perceptron Regression para cenizas. Este grafico cuenta con un tamaño de muestra 10 y media 0.1.

En la tabla 12 se presentan los resultados de la prueba Shapiro Wilk.

Tabla 12. Resumen de prueba de ajuste Shapiro Wilk

Parámetro	P-value	W
Proteína	0.15	0.95
Grasas	0.3	0.96
Humedad	0.44	0.97
Cenizas	0.12	0.92

Fuente: Elaboración propia.

4.5 Validación de modelo

Para identificar los modelos con mejor calidad de predicción de parámetros se tomará en cuenta el apartado 3.5 del capítulo 3. Al realizar las pruebas descritas, es posible determinar el modelo con mejores prestaciones para predecir parámetros de harina de pecado.

4.5.1 Prueba por coeficiente de correlación

En la tabla 13 y 14 se detalla el resumen de las métricas indicadoras de calidad de predicción para el algoritmo de regresión MLP. En esta sección solo se toma en cuenta el algoritmo MLP debido a que tiene mayor calidad de predicción que el algoritmo SV, esta información es respaldada por el apartado 4.3.1.

Para grupo A:

Tabla 13. Resumen de modelos de predicción

Parámetro	Algoritmo	R	RMS	Error(%)
Proteína	MLP	0.45	0.67	0.96
Grasas	MLP	0.71	0.24	4.37
Humedad	MLP	0.71	0.24	4.37
Cenizas	MLP	0.6	0.3	5.77

Fuente: Elaboración propia.

Para grupo B:

Tabla 14. Resumen de modelos de predicción

Parámetro	Algoritmo	R	RMS	Error(%)
Proteína	MLP	0.89	0.32	0.71
Grasas	MLP	0.9	0.1	2.79
Humedad	MLP	0.67	0.22	5.12
Cenizas	MLP	0.67	0.11	1.54

Fuente: Elaboración propia.

Se observa que los modelos de regresión para el grupo B de la tabla 13, tiene mayor calidad de predicción. Esto es posible debido a que la diferencia de tiempo baja (como máximo 2 meses) de entre la fecha de muestreo de parámetros fisicoquímicos respecto a la fecha de captura de imágenes hiperespectrales, existe una degradación de la materia, por lo

tanto, la composición química y física varia, al tomar la imagen, las firmas espectrales tienden a ser diferentes.

También es posible identificar que, para todos los modelos, el algoritmo de regresión usando redes neuronales artificiales MLP obtuvo los valores más altos de coeficiente de correlación (cercanos a 1).

4.5.2 Verificación estadística

Para verificar la aleatoriedad y normalidad de error se presenta la tabla 15 y 16 respectivamente.

Para grupo A:

Tabla 15. Resumen de modelos de predicción

Parámetro	Algoritmo	P-value	W
Proteína	MLP	0.72	0.98
Grasas	MLP	0.41	0.97
Humedad	MLP	0.03	0.95
Cenizas	MLP	0.51	0.98

Fuente: Elaboración propia.

Para grupo B:

Tabla 16. Resumen de modelos de predicción

Parámetro	Algoritmo	P-value	W
Proteína	MLP	0.15	0.95
Grasas	MLP	0.2	0.96
Humedad	MLP	0.44	0.97
Cenizas	MLP	0.12	0.92

Fuente: Elaboración propia.

Se observa que para todos los casos se tiene:

$$W \cong 1$$

$$Pvalue > 0.02$$

H_o = se ajusta a una distribución normal (p-value > α, α = 2% y W >> 0)

H_1 = no se ajusta a una distribución normal (p-value < α, α = 2% y W << 0)

Se verifica la hipótesis H_0.

Capítulo 5
Conclusiones y recomendaciones

5.1 Conclusiones

Basados en los resultados mostrados en el capítulo 4, se presentan las siguientes conclusiones:

1. Los sistemas de adquisición de imágenes hiperespectrales tienen una influencia significativa en el desarrollo de sistemas no invasivos para medir parámetros fisicoquímicos, relacionados con la calidad de productos de origen marino, como la harina de pescado.

2. El desarrollo de algoritmos de reducción de dimensionalidad ayudó a reducir el número de bandas, por lo que demuestra que es posible desarrollar sistemas no invasivos con menor número de bandas (60). Esta premisa va relacionada directamente con el costo de los instrumentos.

3. Para diseñar un sistema de predicción de parámetros fisicoquímicos para harina de pescado basta con utilizar un espectrómetro y no una cámara hiperespectral. Al ser una muestra homogénea no es necesario identificar formas específicas, por lo tanto, un sistema de adquisición de imágenes queda sobredimensionado para la aplicación, al igual que la conclusión anteriormente descrita, influye directamente en el costo de desarrollo de los instrumentos.

4. El sistema de identificación tiene mayor precisión para el grupo B. Esta afirmación toma sentido, debido a que, la fecha de medición de parámetros por técnicas químicas y físicas, comparado con la fecha de adquisición de imagen por muestra es bajo. El tiempo aproximado es entre 2 semanas a 1 mes. Para las muestras del grupo A, las fechas de adquisición de los parámetros por los diferentes mecanismos tienen una diferencia aproximada de 2 años. Esto se debe a que, existe una degradación de la harina de pescado, por ello en esta presente tesis se tomará mayor relevancia en los resultados obtenidos por el grupo B.

5. Es posible predecir con precisión aceptable los siguientes parámetros: proteína, humedad, grasas y cenizas. Los modelos de regresión más exactos se dieron gracias a los sistemas de identificación usando redes neuronales artificiales, como se puede observar en las tablas 8 y tabla 7. Los valores del coeficiente de correlación R son aceptables ya que son cercanos a 1, el RMS e1 bajo porque es cercano a 0; es decir los modelos se ajustan para asegurar una buena calidad de predicción.

6. Los índices estadísticos determinaron y confirmaron la conclusión anterior. Se tiene como consideración, el comportamiento del error como una manera alternativa de verificar la precisión y robustez de los sistemas de predicción. Es importante verificar que la distribución que forma el error del modelo sea una distribución normal.

7. La implementación y el desarrollo del software de código, así como la utilización de frameworks comerciales, tienen una positiva influencia en la escalabilidad, robustez de sistema y reducción de costos. Es posible desarrollar sistemas en producción, añadir protocolos de comunicaciones industriales con computadores de bajo costo.

8. El tiempo de identificación por parámetros es de aproximadamente 2 segundos para obtener los datos en tiempo real y en línea. La obtención de estos parámetros ayuda a tomar acciones para mantener la mejor calidad de harina de pescado.

5.2 Recomendaciones

Estas recomendaciones están basadas en los resultados y conclusiones presentadas en el capítulo 4 y 5 respectivamente, se presentan las siguientes recomendaciones:

1. Para los futuros trabajos, es necesario tener en cuenta que, las fechas de adquisición de imágenes con las fechas de ensayos deben tener una diferencia de máximo 2 meses. Es importante considerar la degradación de la materia de harina de pescado.

2. Realizar pruebas con espectrómetros de lectura en tiempo real podría reducir el tiempo de procesamiento. Además, incluir módulos de comunicaciones industriales haría posible la conexión con equipos de instrumentación industrial.

3. Se debe plantear el desarrollo de un prototipo funcional de medición como trabajo futuro. Asimismo, realizar un proceso de identificación del proceso de producción de harina de pescado, analizar las variables más influyentes y construir un sistema de control automático que asegure la mejor calidad de harina de pescado.

4. Al diseñar un prototipo funcional es necesario e indispensable tener en cuenta el ambiente de operación. Los procesos de producción de harina de pescado se dan

sobre condiciones salinas y húmedas. La aparición de corrosión en el prototipo afectará la calidad de medición y la vida útil del instrumento.

Bibliografía

Vannesch, L., & Castelli, M. (2018). Multilayer Perceptrons. *Encyclopedia of Bioinformatics and Computational Biology*, 1-9.

Basak, D., Pal, S., & Patranabis, C. (2007). Support Vector Regression. *Neural Information Processing*, 203 - 224.

Brodley, C., & Dy, J. (2004). Feature Selection for Unsupervised Learning. *Journal of Machine Learning Research*, 845–889.

Dai, Q., Cheng, J., Sun, D., Zhu, Z., & Pu, H. (2016). Prediction of total volatile basic nitrogen contents using wavelet features from visible/near-infrared hyperspectral images of prawn (Metapenaeus. *Food Chemistry*, 257–265.

Elmasry, G., Kamruzzaman, M., Sun, D., & Allen, P. (2012). Principles and applications of hyperspectral imaging in quality evaluation of agro-food products: A review. *Critical Reviews in Food Science and Nutrition*, 999–1023.

Fernandes, A., Franco, C., Mendes-Ferreira, A., Mendes-Faia, A., Leal da Costa, P., & Melo-Pinto, P. (2015). Brix, pH and anthocyanin content determination in whole Port wine. *Computers and Electronics in Agriculture*, 88-96.

Gabriel, J. (2016). *Artificial Intelligence: Artificial Intelligence for Humans.* USA: CreateSpace Independent Publishing Platform.

Gilbert, J., & Lemarechal, C. (1989). Some numerical experiments with variable-storage quasi-Newton algorithms. *Mathematical Programming*, 407-435.

Google. (2 de Noviembre de 2018). *TensorFlow*. Obtenido de https://www.tensorflow.org/?hl=es

Gributs, C. E., & Burns, D. H. (2006). Parsimonious calibration models for nearinfrared spectroscopy using wavelets and scaling functions. *Chemometrics and Intelligent Laboratory Systems, 83*, 44–53.

Gurney, K. (2004). *An introduction to neural networks.* London: Taylor & Francis.

Hwang, J., Lay, S., Maechler, M., Martin, D., & Schimer, J. (1994). Regression modeling in back-propagation and projection pursuit learning. *IEEE Transactions on Neural Networks* , 342 - 353.

Instruments, F. (16 de 11 de 2018). *Felix Instruments*. Obtenido de https://felixinstruments.com/food-science-instruments/portable-nir-analyzers/f-750-produce-quality-meter/#

Kashaninejad, M., Dehghani, A., & Kashiri, M. (2009). Modeling of wheat soaking using two artificial neural networks (MLP and RBF). *Journal of Food Engineering*, 602–607.

Kim, M., Chao, K., & Bannon, D. (2015). *United States Patente nº US 9,176,110 B1*.

Kozak, A., & Kozak, R. (2003). Does cross validation provide additional information in the evaluation of regression models? *Canadian Journal of Forest Research*, 976-987.

Kriesel, D. (2007). *A Brief Introduction to Neural Networks.* Bonn.

Montgomerz, D., Peck, E., & Geoffrey, G. (2012). *Introduction to Linear Regression Analysis.*

Mundaca, G., Soto, J., & Ipanaqué, W. (2015). Evolution of the spectral index anthocyanin RI2 in the cocoa beans drying process. *CONGRESO SALESIANO DE CIENCIA, TECNOLOGÍA E INNOVACIÓN PARA LA SOCIEDAD*, 49-53.

Mutlu, A., Hakki, I., Genis, H., Ozturk, R., Basaran-Akgul, N., Sanal, T., & Kaplan, A. (2011). Prediction of wheat quality parameters using near-infrared spectroscopy and artificial neural networks. *Eur Food Res Technol*, 267–274.

Nieto, N., & Orozco, D. M. (2008). The use of the discrete Wavelet transform in the reconstruction of sinusoidal signals. *Scientia et Technica Año XIV, 38*, 381 - 386.

Perten Intruments. (13 de 11 de 2018). *https://www.perten.com*. Obtenido de https://www.perten.com/es/Productos/Analizador-NIR-DA-7250/Especificaciones/

Pita , S., & Pértega, S. (1997). Relación entre variables cuantitativas. *Unidad de Epidemiología Clínica y Bioestadística*, 141-144.

Pu, H., Su, D. W., Ma, J., Liu, D., & Cheng, J. H. (2014). Wavelet Textural Features of Visible and Near Infrared Hyperspectral Image to Differentiate Between Fresh and Frozen-Thawed Pork. *Food and Bioprocess Technology, 7(11)*, 3088–3099.

Rahiala, M. (2011). Parametric and Nonparametric Reliability Analysis. *International Encyclopedia of Statistical Science*, 1047 - 1149.

Rodríguez, M. Á. (2016). *Teledetección de gases mediante imagen hiperespectral por transformada de Fourier en el infrarrojo medio. Una contribución al problema de la cuantificación y la reducción de la dimensionalidad.* Leganés: Universidad Carlos III de Madrid.

Romero, M. (2016). Pruebas de bondad de ajuste a una. *Revista Enfermería del Trabajo*, 105-114.

Shobha, G., & Rangaswamy, S. (2018). *Computational Analysis and Understanding of Natural Languages: Principles, Methods and Applications.* North Holland: ElSEVIER.

Sun, D. (2010). *Hyperspectral Imaging for Food Quality Analysis and Control.* Academic Press.

Sun, D. (2015). *Computer Vision Technology for Food Quality Evaluation.* San Diego: Elsevier Science Publishing.

Triguero, I., García, S., & Herrera, F. (2013). Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Journal Knowledge and Information Systems*, 245-284 .

Wang, J., & Shen, X. (2007). Large Margin Semi-supervised Learning. *Journal of Machine Learning Research 8*, 1867-1891.

Zhang, W. (2007). A Generalized ADALINE Neural Network for System Identification. *IEEE International Conference on Control and Automation* , 2705-2709.

Anexos

Anexo A Códigos para identificación de modelos usando SV

Proteína

```
# -*- coding: utf-8 -*-

from sklearn.svm import SVR
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
from sklearn.model_selection import train_test_split


from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error, explained_variance_score, matthews_corrcoef
from sklearn.neural_network import MLPRegressor
from sklearn.externals import joblib
import pickle


dataset = np.loadtxt("db2.csv", delimiter=",")
param = 1
```

```
X_raw_log = dataset[:,12:72]
X_raw_r = dataset[:,73:133]
Y_raw = dataset[:,param]
X_train, X_test, Y_train, Y_test =  train_test_split(X_raw_log, Y_raw, test_size=0.30, random_state=42)
x_train_log = X_train
x_test = X_test
y_train = Y_train
y_test = Y_test
X = x_train_log
Y = y_train

svr_poly = SVR(kernel='rbf', C=1e4,  gamma=200, epsilon=1e-8)
svr_poly = svr_poly.fit(X, Y)
model = svr_poly
joblib.dump(model,"SVR-proteina",compress=9)
y_poly = svr_poly.predict(x_test)
params = svr_poly.get_params()
corr = np.corrcoef(y_test,y_poly)


fig, ax = plt.subplots()
ax.scatter(y_test, y_poly, edgecolors='#4A148C', alpha=1, c='#4A148C')
ax.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], '--', lw=3, color='#263238')
ax.set_xlabel('Measured')
ax.set_ylabel('Predicted')
plt.ylim(64, 72)
plt.xlim(64, 72)

if(param == 1):
        print("Parameter : Proteina")
```

```
print("----------------------------------")
print("Length of dataset : " + str(len(dataset)))
print("Train Samples :" + str(len(Y)))
print("Test Samples :" + str(len(y_test)))
print("Correlation : " + str(round(corr[0,1],2)))
print("----------------------------------")

error_array_por = np.divide(np.absolute(np.subtract(y_test, y_poly)),y_test)
error_max = np.nanmax(np.absolute(np.subtract(y_test, y_poly)))
error_array = np.absolute(np.subtract(y_test, y_poly))
_error_array = np.subtract(y_test, y_poly)
error_array_por = np.multiply(error_array_por,100)
error_por = np.mean(error_array_por)
print("error(%) : " + str(round(error_por,2)))
# The mean squared error
print("Mean squared error : %.2f" % mean_squared_error(y_test, y_poly))
#Best possible score is 1.0
print('Variance score : %.2f' % abs(r2_score(y_test, y_poly)))
print('Mean absolute error : %.2f' % mean_absolute_error(y_test, y_poly))
print ("Max error: %0.2f" % error_max)
print ("Samples with error < 1 : %d" % (error_array < 1).sum())
y_poly = np.round(y_poly,2)
plt.show()
```

Grasa

```
# -*- coding: utf-8 -*-

from sklearn.svm import SVR
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import RectBivariateSpline
from sklearn.model_selection import train_test_split
```

```
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error, explained_variance_score, matthews_corrcoef
from sklearn.neural_network import MLPRegressor
from sklearn.externals import joblib
import pickle

dataset = np.loadtxt("db2.csv", delimiter=",")
param = 2
X_raw_log = dataset[:,12:72]
X_raw_r = dataset[:,73:133]
Y_raw = dataset[:,param]
X_train, X_test, Y_train, Y_test =  train_test_split(X_raw_log, Y_raw, test_size=0.3, random_state=42)
x_train_log = X_train
x_test = X_test
y_train = Y_train
y_test = Y_test
X = x_train_log
Y = y_train

svr_poly = SVR(kernel='rbf', C=1e1,  gamma=80, epsilon=1e-1)
svr_poly = svr_poly.fit(X, Y)
model = svr_poly
joblib.dump(model,"SVR-grease",compress=9)
y_poly = svr_poly.predict(x_test)
params = svr_poly.get_params()
corr = np.corrcoef(y_test,y_poly)

fig, ax = plt.subplots()
ax.scatter(y_test, y_poly, edgecolors='#4A148C', alpha=1, c='#4A148C')
ax.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=4)
ax.set_xlabel('Measured')
ax.set_ylabel('Predicted')
plt.ylim(6, 12)
```

```
plt.xlim(6, 12)

if(param == 2):
        print("Grasa")

print("----------------------------------")
print("Length of dataset : " + str(len(dataset)))
print("Train Samples :" + str(len(Y)))
print("Test Samples :" + str(len(y_test)))
print("Correlation : " + str(round(corr[0,1],2)))
print("----------------------------------")

error_array_por = np.divide(np.absolute(np.subtract(y_test, y_poly)),y_test)
error_array = np.absolute(np.subtract(y_test, y_poly))
error_max = np.nanmax(np.absolute(np.subtract(y_test, y_poly)))
error_array_por = np.multiply(error_array_por,100)
error_por = np.mean(error_array_por)
print("error(%) : " + str(round(error_por,2)))
# The mean squared error
print("Mean squared error : %.2f" % mean_squared_error(y_test, y_poly))
#Best possible score is 1.0
print('Variance score : %.2f' % abs(r2_score(y_test, y_poly)))
print('Mean absolute error : %.2f' % mean_absolute_error(y_test, y_poly))
print ("Max error: %0.2f" % error_max)
print ("Samples with error < 1 : %d" % (error_array < 1).sum())
y_poly = np.round(y_poly,2)
plt.show()
```

Humedad

```
# -*- coding: utf-8 -*-

from sklearn.svm import SVR
import numpy as np
```

```
import matplotlib.pyplot as plt
from scipy.interpolate import RectBivariateSpline
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error, explained_variance_score, matthews_corrcoef
from sklearn.neural_network import MLPRegressor
from sklearn.externals import joblib
import pickle

dataset = np.loadtxt("db2.csv", delimiter=",")
param = 3
X_raw_log = dataset[:,12:72]
X_raw_r = dataset[:,73:133]
Y_raw = dataset[:,param]
X_train, X_test, Y_train, Y_test =  train_test_split(X_raw_log, Y_raw, test_size=0.3, random_state=42)
x_train_log = X_train
x_test = X_test
y_train = Y_train
y_test = Y_test
X = x_train_log
Y = y_train
svr_poly = SVR(kernel='rbf', C=1e2,  gamma=100, epsilon=1e-3)
svr_poly = svr_poly.fit(X, Y)
model = svr_poly
joblib.dump(model,"SRV-humedad",compress=9)
y_poly = svr_poly.predict(x_test)
params = svr_poly.get_params()
corr = np.corrcoef(y_test,y_poly)


fig, ax = plt.subplots()
ax.scatter(y_test, y_poly, edgecolors='#4A148C', alpha=1, c='#4A148C')
```

```
ax.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], '--', lw=3, color='#263238')
ax.set_xlabel('Measured')
ax.set_ylabel('Predicted')
plt.ylim(5, 10)
plt.xlim(5, 10)

if(param == 3):
        print("Humeadad")

print("----------------------------------")
print("Length of dataset : " + str(len(dataset)))
print("Train Samples :" + str(len(Y)))
print("Test Samples :" + str(len(y_test)))
print("Correlation : " + str(round(corr[0,1],2)))
print("----------------------------------")

error_array_por = np.divide(np.absolute(np.subtract(y_test, y_poly)),y_test)
error_max = np.nanmax(np.absolute(np.subtract(y_test, y_poly)))
error_array = np.absolute(np.subtract(y_test, y_poly))
error_array_por = np.multiply(error_array_por,100)
error_por = np.mean(error_array_por)
print("error(%) : " + str(round(error_por,2)))
# The mean squared error
print("Mean squared error : %.2f" % mean_squared_error(y_test, y_poly))
#Best possible score is 1.0
print('Variance score : %.2f' % abs(r2_score(y_test, y_poly)))
print('Mean absolute error : %.2f' % mean_absolute_error(y_test, y_poly))
print ("Max error: %0.2f" % error_max)
print ("Samples with error < 1 : %d" % (error_array < 1).sum())
y_poly = np.round(y_poly,2)
plt.show()
```

Cenizas

```
# -*- coding: utf-8 -*-

from sklearn.svm import SVR
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import RectBivariateSpline
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error, explained_variance_score, matthews_corrcoef
from sklearn.neural_network import MLPRegressor
from sklearn.externals import joblib
import pickle

dataset = np.loadtxt("db2.csv", delimiter=",")
param = 4
X_raw_log = dataset[:,12:72]
X_raw_r = dataset[:,73:133]
Y_raw = dataset[:,param]
X_train, X_test, Y_train, Y_test =  train_test_split(X_raw_log, Y_raw, test_size=0.3, random_state=42)
x_train_log = X_train
x_test = X_test
y_train = Y_train
y_test = Y_test
X = x_train_log
Y = y_train
svr_poly = SVR(kernel='rbf', C=1e1,  gamma=50, epsilon=1e-3)
svr_poly = svr_poly.fit(X, Y)
model = svr_poly
joblib.dump(model,"SVR-cenizas",compress=9)
y_poly = svr_poly.predict(x_test)
params = svr_poly.get_params()
corr = np.corrcoef(y_test,y_poly)
```

```
fig, ax = plt.subplots()
ax.scatter(y_test, y_poly, edgecolors='#4A148C', alpha=1, c='#4A148C')
ax.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], '--', lw=3, color='#263238')
ax.set_xlabel('Measured')
ax.set_ylabel('Predicted')
plt.ylim(14, 18)
plt.xlim(14, 18)

if(param == 5):
        print("Cenizas")

print("----------------------------------")
print("Length of dataset : " + str(len(dataset)))
print("Train Samples :" + str(len(Y)))
print("Test Samples :" + str(len(y_test)))
print("Correlation : " + str(round(corr[0,1],2)))
print("----------------------------------")

error_array_por = np.divide(np.absolute(np.subtract(y_test, y_poly)),y_test)
error_max = np.nanmax(np.absolute(np.subtract(y_test, y_poly)))
error_array_por = np.multiply(error_array_por,100)
error_por = np.mean(error_array_por)
print("error(%) : " + str(round(error_por,2)))
# The mean squared error
print("Mean squared error : %.2f" % mean_squared_error(y_test, y_poly))
#Best possible score is 1.0
print('Variance score : %.2f' % abs(r2_score(y_test, y_poly)))
print('Mean absolute error : %.2f' % mean_absolute_error(y_test, y_poly))
print(error_max)
y_poly = np.round(y_poly,2)
plt.show()
```

Anexo B Códigos para modelos de aprendizaje automático MLP bajo Tensorflow

Proteína

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense
from keras import regularizers
import keras
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error, explained_variance_score, matthews_corrcoef

def denormalize(param, norm_data):
    df = dataset[:,param].reshape(-1,1)
    norm_data = norm_data.reshape(-1,1)
    scl = MinMaxScaler()
    a = scl.fit_transform(df)
    new = scl.inverse_transform(norm_data)
    return new

dataset = np.loadtxt("db2.csv", delimiter=",")
param = 1
X_raw_log = dataset[:,12:72]
X_raw_r = dataset[:,73:133]
Y_raw = dataset[:,param]
X_train, X_test, Y_train, Y_test =  train_test_split(X_raw_log, Y_raw, test_size=0.30, random_state=42)
scaler = MinMaxScaler(feature_range=(0,1))
X_train = scaler.fit_transform(X_train)
```

```
Y_train = scaler.fit_transform(Y_train.reshape(-1,1))
X_test = scaler.fit_transform(X_test)
Y_test = scaler.fit_transform(Y_test.reshape(-1,1))

model = Sequential()
model.add(Dense(60, input_dim=60, activation='linear'))
model.add(Dense(15, activation='linear', activity_regularizer=regularizers.l2(2e-1)))
model.add(Dense(15, activation='linear'))
model.add(Dense(3, activation='linear'))
model.add(Dense(1, activation='linear'))
model.compile(loss='mean_squared_error', optimizer='Adamax', metrics=['acc'])
model.fit(X_train, Y_train, epochs=10000, batch_size=1000, verbose=0)
pred = model.predict(X_test)
pred = scaler.inverse_transform(pred)
Y_test = scaler.inverse_transform(Y_test)
corr = np.corrcoef(Y_test,pred)

fig, ax = plt.subplots()
ax.scatter(Y_test, pred, edgecolors='#4A148C', alpha=1, c='#4A148C')
ax.plot([Y_test.min(), Y_test.max()], [Y_test.min(), Y_test.max()], '--', lw=3,
color='#263238')
ax.set_xlabel('Measured')
ax.set_ylabel('Predicted')
plt.ylim(64, 72)
plt.xlim(64, 72)

print("----------------------------------")
print("Length of dataset : " + str(len(dataset)))
print("Train Samples :" + str(len(Y_train)))
print("Test Samples :" + str(len(Y_test)))
print("Correlation : " + str(round(corr[0,1],2)))
print("----------------------------------")

error_array_por = np.divide(np.absolute(np.subtract(Y_test, pred)),Y_test)
```

```
error_max = np.nanmax(np.absolute(np.subtract(Y_test, pred)))
error_array = np.absolute(np.subtract(Y_test, pred))
_error_array = np.subtract(Y_test, pred)
error_array_por = np.multiply(error_array_por,100)
error_por = np.mean(error_array_por)
print("error(%) : " + str(round(error_por,2)))
# The mean squared error
print("Mean squared error : %.2f" % mean_squared_error(Y_test, pred))
#Best possible score is 1.0
print('Variance score : %.2f' % abs(r2_score(Y_test, pred)))
print('Mean absolute error : %.2f' % mean_absolute_error(Y_test, pred))
print ("Max error: %0.2f" % error_max)
print ("Samples with error < 1 : %d" % (error_array < 1).sum())
plt.show()
```

Grasas

```
# -*- coding: utf-8 -*-

import numpy as np
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense
from keras import regularizers
import keras
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error,
explained_variance_score, matthews_corrcoef

def denormalize(param, norm_data):
    df = dataset[:,param].reshape(-1,1)
```

```
    norm_data = norm_data.reshape(-1,1)
    scl = MinMaxScaler()
    a = scl.fit_transform(df)
    new = scl.inverse_transform(norm_data)
    return new

dataset = np.loadtxt("db2.csv", delimiter=",")
param = 2
X_raw_log = dataset[:,12:72]
X_raw_r = dataset[:,73:133]
Y_raw = dataset[:,param]
X_train, X_test, Y_train, Y_test =  train_test_split(X_raw_log, Y_raw, test_size=0.30, random_state=42)
scaler = MinMaxScaler(feature_range=(0,1))
X_train = scaler.fit_transform(X_train)
Y_train = scaler.fit_transform(Y_train.reshape(-1,1))
X_test = scaler.fit_transform(X_test)
Y_test = scaler.fit_transform(Y_test.reshape(-1,1))

model = Sequential()
activation = 'tanh'
model.add(Dense(60, input_dim=60, activation='linear'))
model.add(Dense(5, activation=activation, activity_regularizer=regularizers.l2(2e-1)))
model.add(Dense(5, activation=activation))
model.add(Dense(3, activation=activation))
model.add(Dense(1, activation='linear'))
model.compile(loss='mean_squared_error', optimizer='Adamax', metrics=['acc'])
model.fit(X_train, Y_train, epochs=10000, batch_size=1000, verbose=0)
pred = model.predict(X_test)
pred = scaler.inverse_transform(pred)
Y_test = scaler.inverse_transform(Y_test)
corr = np.corrcoef(Y_test,pred)

fig, ax = plt.subplots()
```

```
ax.scatter(Y_test, pred, edgecolors='#4A148C', alpha=1, c='#4A148C')
ax.plot([Y_test.min(), Y_test.max()], [Y_test.min(), Y_test.max()], '--', lw=3, color='#263238')
ax.set_xlabel('Measured')
ax.set_ylabel('Predicted')
plt.ylim(6, 12)
plt.xlim(6, 12)

print("----------------------------------")
print("Length of dataset : " + str(len(dataset)))
print("Train Samples :" + str(len(Y_train)))
print("Test Samples :" + str(len(Y_test)))
print("Correlation : " + str(round(corr[0,1],2)))
print("----------------------------------")

error_array_por = np.divide(np.absolute(np.subtract(Y_test, pred)),Y_test)
error_max = np.nanmax(np.absolute(np.subtract(Y_test, pred)))
error_array = np.absolute(np.subtract(Y_test, pred))
_error_array = np.subtract(Y_test, pred)
error_array_por = np.multiply(error_array_por,100)
error_por = np.mean(error_array_por)
print("error(%) : " + str(round(error_por,2)))
# The mean squared error
print("Mean squared error : %.2f" % mean_squared_error(Y_test, pred))
#Best possible score is 1.0
print('Variance score : %.2f' % abs(r2_score(Y_test, pred)))
print('Mean absolute error : %.2f' % mean_absolute_error(Y_test, pred))
print ("Max error: %0.2f" % error_max)
print ("Samples with error < 1 : %d" % (error_array < 1).sum())
plt.show()
```

Humedad

```
# -*- coding: utf-8 -*-

import numpy as np
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense
from keras import regularizers
import keras
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error, explained_variance_score, matthews_corrcoef

def denormalize(param, norm_data):
    df = dataset[:,param].reshape(-1,1)
    norm_data = norm_data.reshape(-1,1)
    scl = MinMaxScaler()
    a = scl.fit_transform(df)
    new = scl.inverse_transform(norm_data)
    return new

dataset = np.loadtxt("db2.csv", delimiter=",")
param = 3
X_raw_log = dataset[:,12:72]
X_raw_r = dataset[:,73:133]
Y_raw = dataset[:,param]
X_train, X_test, Y_train, Y_test =  train_test_split(X_raw_log, Y_raw, test_size=0.30, random_state=42)
scaler = MinMaxScaler(feature_range=(0,1))
X_train = scaler.fit_transform(X_train)
```

```
Y_train = scaler.fit_transform(Y_train.reshape(-1,1))
X_test = scaler.fit_transform(X_test)
Y_test = scaler.fit_transform(Y_test.reshape(-1,1))

model = Sequential()
activation = 'tanh'
model.add(Dense(60, input_dim=60, activation='linear'))
model.add(Dense(10, activation=activation, activity_regularizer=regularizers.l2(2e-1)))
model.add(Dense(25, activation=activation))
model.add(Dense(10, activation=activation))
model.add(Dense(1, activation='linear'))
model.compile(loss='mean_squared_error', optimizer='Adamax', metrics=['acc'])
model.fit(X_train, Y_train, epochs=10000, batch_size=10000, verbose=1)
pred = model.predict(X_test)
pred = scaler.inverse_transform(pred)
Y_test = scaler.inverse_transform(Y_test)
corr = np.corrcoef(Y_test,pred)

fig, ax = plt.subplots()
ax.scatter(Y_test, pred, edgecolors='#4A148C', alpha=1, c='#4A148C')
ax.plot([Y_test.min(), Y_test.max()], [Y_test.min(), Y_test.max()], '--', lw=3,
color='#263238')
ax.set_xlabel('Measured')
ax.set_ylabel('Predicted')
plt.ylim(5, 10)
plt.xlim(5, 10)

print("----------------------------------")
print("Length of dataset : " + str(len(dataset)))
print("Train Samples :" + str(len(Y_train)))
print("Test Samples :" + str(len(Y_test)))
print("Correlation : " + str(round(corr[0,1],2)))
print("----------------------------------")
```

```
error_array_por = np.divide(np.absolute(np.subtract(Y_test, pred)),Y_test)
error_max = np.nanmax(np.absolute(np.subtract(Y_test, pred)))
error_array = np.absolute(np.subtract(Y_test, pred))
_error_array = np.subtract(Y_test, pred)
error_array_por = np.multiply(error_array_por,100)
error_por = np.mean(error_array_por)
print("error(%) : " + str(round(error_por,2)))
# The mean squared error
print("Mean squared error : %.2f" % mean_squared_error(Y_test, pred))
#Best possible score is 1.0
print('Variance score : %.2f' % abs(r2_score(Y_test, pred)))
print('Mean absolute error : %.2f' % mean_absolute_error(Y_test, pred))
print ("Max error: %0.2f" % error_max)
print ("Samples with error < 1 : %d" % (error_array < 1).sum())
plt.show()
```

Cenizas

```
# -*- coding: utf-8 -*-

import numpy as np
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense
from keras import regularizers
import keras
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error,
explained_variance_score, matthews_corrcoef

def denormalize(param, norm_data):
```

```
    df = dataset[:,param].reshape(-1,1)
    norm_data = norm_data.reshape(-1,1)
    scl = MinMaxScaler()
    a = scl.fit_transform(df)
    new = scl.inverse_transform(norm_data)
    return new

dataset = np.loadtxt("db2.csv", delimiter=",")
param = 4
X_raw_log = dataset[:,12:72]
X_raw_r = dataset[:,73:133]
Y_raw = dataset[:,param]
X_train, X_test, Y_train, Y_test =  train_test_split(X_raw_log, Y_raw, test_size=0.30,
random_state=42)
scaler = MinMaxScaler(feature_range=(0,1))
X_train = scaler.fit_transform(X_train)
Y_train = scaler.fit_transform(Y_train.reshape(-1,1))
X_test = scaler.fit_transform(X_test)
Y_test = scaler.fit_transform(Y_test.reshape(-1,1))

model = Sequential()
activation = 'linear'
model.add(Dense(60, input_dim=60, activation='linear'))
model.add(Dense(15, activation=activation, activity_regularizer=regularizers.l2(2e-1)))
model.add(Dense(15, activation=activation))
model.add(Dense(6, activation=activation))
model.add(Dense(1, activation='linear'))
model.compile(loss='mean_squared_error', optimizer='Adamax', metrics=['acc'])
model.fit(X_train, Y_train, epochs=20000, batch_size=10000, verbose=1)
pred = model.predict(X_test)
pred = scaler.inverse_transform(pred)
Y_test = scaler.inverse_transform(Y_test)
corr = np.corrcoef(Y_test,pred)
```

```
fig, ax = plt.subplots()
ax.scatter(Y_test, pred, edgecolors='#4A148C', alpha=1, c='#4A148C')
ax.plot([Y_test.min(), Y_test.max()], [Y_test.min(), Y_test.max()], '--', lw=3, color='#263238')
ax.set_xlabel('Measured')
ax.set_ylabel('Predicted')
plt.ylim(14.5, 18)
plt.xlim(14.5, 18)

print("----------------------------------")
print("Length of dataset : " + str(len(dataset)))
print("Train Samples :" + str(len(Y_train)))
print("Test Samples :" + str(len(Y_test)))
print("Correlation : " + str(round(corr[0,1],2)))
print("----------------------------------")

error_array_por = np.divide(np.absolute(np.subtract(Y_test, pred)),Y_test)
error_max = np.nanmax(np.absolute(np.subtract(Y_test, pred)))
error_array = np.absolute(np.subtract(Y_test, pred))
_error_array = np.subtract(Y_test, pred)
error_array_por = np.multiply(error_array_por,100)
error_por = np.mean(error_array_por)
print("error(%) : " + str(round(error_por,2)))
# The mean squared error
print("Mean squared error : %.2f" % mean_squared_error(Y_test, pred))
#Best possible score is 1.0
print('Variance score : %.2f' % abs(r2_score(Y_test, pred)))
print('Mean absolute error : %.2f' % mean_absolute_error(Y_test, pred))
print ("Max error: %0.2f" % error_max)
print ("Samples with error < 1 : %d" % (error_array < 1).sum())
plt.show()
```