UNIVERSIDAD NACIONAL JOSÉ MARÍA ARGUEDAS

FACULTAD DE INGENIERÍA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Presentado por

JOEL SAUL MORENO MAYHUIRE

EFICIENCIA DE UN SISTEMA DE CONTROL DE CALIDAD MEDIANTE PROCESAMIENTO DIGITAL DE IMÁGENES EN LA CLASIFICACIÓN DE LA TUNTA EN LA PLANTA DE PRODUCCIÓN DE KISHUARA - ANDAHUAYLAS

Asesor:

Mgtr. JUAN JOSÉ ORÉ CERRÓN

TESIS PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO DE SISTEMAS

ANDAHUAYLAS – APURÍMAC – PERÚ

2020

DEDICATORIA

Dedico este trabajo a mis padres por su apoyo incondicional en mi desarrollo académico y profesional y a mis docentes de la universidad.

AGRADECIMIENTO

A Dios por brindarme salud y bienestar y ayudarme a realizar esta investigación, a mis padres por su apoyo incondicional y a mis hermanos.

A mis amigos que siempre estuvieron conmigo.

Al Mgtr. Juan José Oré Cerrón y al Mgtr. Thomas Ancco Vizcarra por su apoyo y guía en el desarrollo de la presente investigación.

A la Universidad Nacional José María Arguedas por haberme brindado las puertas abiertas para mi formación profesional.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

La clasificación como uno de los procesos más importantes en empresas agroindustriales requieren un adecuado proceso de trabajo por el personal del área. Estos problemas se presentan también en la clasificación de la tunta que es un producto destacado en los platos típicos del Perú y que actualmente están tomando interés no solamente en el altiplano del Perú sino también en la planta de producción de la tunta en Kishuara – Andahuaylas. La tunta con sus características de tamaño, forma, color y sabor enriquecen nuestra cultura culinaria ocupando una posición importante en la gastronomía nacional y también internacional. Teniendo en cuenta esta importancia, se realizó un estudio de eficiencia de clasificación haciendo uso de los algoritmos de machine learning (aprendizaje automático) de tipo supervisado para el entrenamiento de datos y construcción de modelos de clasificación, para la posterior evaluación de su rendimiento.

Se utilizaron 800 unidades de tuntas de las cuatro categorías para la elaboración del banco de imágenes y estas fueron analizadas individualmente mediante un algoritmo de procesamiento digital de imágenes para la extracción de características de tamaño, forma y color de las tuntas y la posterior construcción del corpus de conocimiento. El corpus se entrenó con algoritmos de aprendizaje automático supervisado (K-Nearest Neighbors, Decision Trees, Gaussian Naive Bayes, Multinomial Naive Bayes, Support Vector Machines) y se generaron modelos de clasificación. Se realizaron validaciones cruzadas con 10 iteraciones de estos modelos con el fin de obtener el rendimiento en la clasificación, después de la evaluación de resultados de cada uno, se determinó que el modelo generado con el algoritmo Support Vector Machines es el de mejor rendimiento en la clasificación de tuntas con una Accuracy del 93.13%, un Recall del 93.48% y un F1-Score del 93.18%.

Debido a la eficiencia del modelo de clasificación generado con el algoritmo Support Vector Machines, esta se usó para la construcción de un sistema clasificador de tuntas integrando con los algoritmos de procesamiento digital de imágenes. Al realizar la prueba del sistema clasificador con 120 unidades de tuntas de las cuatro categorías diferentes a las que han sido destinadas al entrenamiento y prueba del modelo, con el fin de validar su funcionamiento, se recurrió a la matriz de confusión y se obtuvo una Accuracy (Exactitud) del 97.5%, un Recall del 97.5% y un F1-Score del 97.51%.

Palabras clave: Procesamiento digital de imágenes, aprendizaje automático supervisado, modelo de clasificación, corpus de conocimiento, validación cruzada.

ABSTRACT

The classification as one of the most important processes in agroindustrial companies requires an adequate work process by the personnel of the area. These problems are also present in the classification of tunta, which is a prominent product in the typical dishes of Peru, and which is currently taking interest not only in the highlands of Peru but also in the tunta production plant in Kishuara - Andahuaylas. The tunta with its characteristics of size, shape, color and flavor enrich our culinary culture, occupying an important position in national and international gastronomy. Taking this importance into account, a classification efficiency study was carried out using supervised machine learning algorithms for data training and construction of models of classification, for the subsequent evaluation of its performance.

800 units of tuntas from the four categories were used for the elaboration of the image bank and these were individually analyzed using a digital image processing algorithm for the extraction of characteristics of size, shape and color of the tuntas and the subsequent construction of the corpus of knowledge. The corpus was trained with supervised machine learning algorithms (K-Nearest Neighbors, Decision Trees, Gaussian Naive Bayes, Multinomial Naive Bayes, Support Vector Machines) and classification models were generated. Cross validations were performed with 10 iterations of these models in order to obtain the performance in the classification, after evaluating the results of each one, it was determined that the model generated with the Support Vector Machines algorithm is the one with the best performance in the classification of tuntas with an Accuracy of 93.13%, a Recall of 93.48% and F1-Score of 93.18%.

Due to the efficiency of the classifiation model generated with the Support Vector Machines algorithm, it was used for the construction of a classifier system of tuntas integrating with the digital image proccesing algorithms. When performing the classifier system test with 120 units of tuntas from the four different categories to which they have been destined for training and testing the model, in order to validate its operation, the confusión matrix was used and an Accuracy was obtained of 97.5%, a Recall of 97.5% and F1-Score of 97.51%.

Keywords: Digital image processing, supervised machine learning, classification model, knowledge corpus, cross-validation.

CHUMASQA

Kay chumasqa yachay taripasqa willariwanchik, akllayqa hatun chiqap purichiy empresas agroinductriales nisqan wasikunapaq, allin chanin yachasqa puririchiq llamkaqkunata munanku. Chaynam kay sasachakuykuna tarikun tunta nisqanchik akllaykunapipas, risqisqanchikpihina kay ruruqa Perú suyupi tukuy ima mikuykuna yanukunapaqmi; kunankunaqa aswanmi yupaychakkanku mana kay altiplano Perú suyullapichu aswantaq kay planta de producción de la tunta nisqan kishuara llaqtapipas. Kay tunta nisqanqa tukuy niraqmi sayayninpas, kasqanpas, niraqninpas, chaynallataq musyayninpas tukuy ima yanukuypi miskipuni, kay tuntaqa tarikun allin riqsiypi chay gastronomía nacional chaymanta gastronomía internacional nisqanpipas. Chaymi allin chanin yachay akllasqa ruranapaq kay algoritmos de machine learning (aprendizaje automático) nisqanwan yanapachikurqa chay ruraykuna qawarispa taripanapaq, chaynallataq chay modelo nisqan akllana qispichinapaq, chaywan qipata qispiynin taripanapaq.

Kay ruranapaqqa 800 pachak tuntakunata tawaman akllasqamanta banco de imágenes nisqanpi churaqqa, chaykuna sapakama allin allin riqsipasqa chay algoritmo de procesamiento digital de imágenes nisqan rurasqawan yanapachikusqa, chaypi sayanin, imayna kasqan, niraqnin tuntakunapata urquspa chay corpus de conocimiento nisqan qispichinapaq. Chay yachaykuna qispichinapaqqa yachapachikurqa kay algoritmos de aprendizaje automático supervisado (K-Nearest Neighbors, Decision Trees, Gaussian Naive Bayes, Multinomial Naive Bayes, Support Vector Machines) nisqanwanmi, modelos de clasificación nisqan qispinanpaq. Kay yachayta qispichira, validaciones cruzadas con 10 iteraciones nisqankunawan yanapachikuspa, allin chay akllana lluqsinanrayku, chaymanta chay akllana modelo nisqanpa tukupayninta allinta taripaspa, chanincharun chay modelo nisqan akllana rurasa chay algoritmo Support Vector Machines nisqanwan aswan allin akllaq lluqsirun, chaywanmi Accuracy 93.13%, Recall 93.48%, F1-Score 93.18% nisqanman chayanchik.

Allin chanin chay modelo de clasificación nisqan rurasa chay algoritmo Support Vector Machines nisqanwan, kaywan yanapachikuspa qispichinapaq chay sistema clasificador nisqantaqa algoritmos de procesamiento digital de imágenes nisqankunawan yanapachikuspa. Tawaman akllasqamanta 120 tuntakunawan akllayta qallaykurqa, chaymi matriz de confusión nisqanman chayarun, chaypi Accuracy (Exactitud) 97.5%, Recall 97.5%, F1-Score 97.51% nisqanta akllaypi tukuparun.

Chanin simikuna: Procesamiento digital de imágenes, aprendizaje automático supervisado, modelo de clasificación, corpus de conocimiento, validación cruzada.

INTRODUCCIÓN

Como la clasificación es un proceso importante y para optimizarlo en las últimas décadas se está recurriendo al procesamiento digital de imágenes para clasificar los objetos de acuerdo a parámetros de calidad que deben cumplir sus características ya sean de tamaño, forma y color y como una herramienta eficaz en el proceso, en la presente investigación se utilizó los algoritmos de machine learning (aprendizaje automático) de tipo supervisado para el entrenamiento de un corpus de características de tuntas, los modelos se entrenaron y se validaron con el método de validación cruzada para la eficaz evaluación de los mismos.

Esta investigación tiene como objetivo principal determinar la eficiencia de un sistema de control de calidad mediante procesamiento digital de imágenes y un modelo de machine learning en la clasificación de la tunta en la planta de producción de Kishuara – Andahuaylas**.** Debido a que se cuenta con 4 clases de tuntas (primera, segunda, tercera, cuarta) que obedecen ciertos parámetros de calidad para pertenecer a un grupo de clase, estos parámetros están establecidos de acuerdo a características de tamaño, forma y color de cada tunta. El presente informe de tesis consta de la siguiente estructura:

CAPÍTULO I: Planteamiento del problema. En donde se parte desde la realidad problemática, para la formulación de los problemas, justificación y por consiguiente establecer los objetivos de la investigación.

CAPÍTULO II: Marco teórico. Donde se encuentra el análisis de los antecedentes de investigaciones similares a la presente, el marco conceptual de donde se basa la presente investigación.

CAPÍTULO III: Materiales y Metodología. Donde se establece la operacionalización de variables, población y muestra, nivel y diseño de investigación, técnicas de instrumentos de acopio de datos y las técnicas de análisis de datos.

CAPÍTULO IV: Resultados y discusión. Se partió desde la recopilación de imágenes para la extracción de características mediante procesamiento digital de imágenes, hasta la evaluación de los modelos de clasificación y con el modelo más eficiente se implementó el sistema clasificador de tuntas integrando con algoritmos de procesamiento digital de imágenes.

Y por último se tiene las conclusiones de la investigación y las recomendaciones para futuros trabajos.

CAPÍTULO I: PLANTEAMIENTO DEL PROBLEMA

1.1. Realidad problemática

La tunta es un producto milenario que su proceso se ha transmitido de generación en generación y su consumo se viene revalorando a nivel local, nacional e internacional. Con el procesado tradicional practicado, se logra un producto uniforme en color en la tunta, debido a que las condiciones ambientales naturales surten efecto por las fuertes corrientes de aire acompañado de temperaturas que están por debajo de -10°C.

La tunta (chuño blanco) un derivado de la papa que se produce principalmente en el altiplano peruano, en Puno, por encima de 4,000 msnm, donde prospera una producción comercial dirigida al mercado peruano y boliviano. Existen sin embargo limitaciones tecnológicas que repercuten en la calidad del producto final, evitando su expansión comercial (Fonseca, Huarachi, & Ordinola, 2011).

La tunta, un alimento muy conocido y destacado en los platos típicos del Perú, con sus características de tamaño, forma, color y sabor que enriquecen nuestra cultura culinaria, originada en el altiplano y ahora ocupan una posición importante en la gastronomía nacional y también internacional, posee atractivas ventajas que pueden posibilitar la ampliación de su consumo a las grandes ciudades, dadas sus bondades nutricionales, su buena adaptación a diversos preparados y su prolongada conservación. Para ello es necesario asegurar una adecuada calidad a fin de ofrecer al mercado un producto homogéneo, manteniendo la limpieza, la textura y su sabor particular (Fonseca C. , Huarachi, Chura, & Cotrado, 2007).

La producción de la tunta no solamente se hace en Puno o en Bolivia, también se está tomando interés en otros lugares del Perú como es el caso de la asociación de productores de Kishuara, que con los avatares del fenómeno de cambio climático y las condiciones ambientales han cambiado drásticamente, han implementado una planta para la producción semi industrial de la tunta.

Sin embargo, la producción de tuntas presenta dificultades específicamente en el proceso de clasificación. El producto final se distingue entre los tamaños grande, mediano y pequeño e implícitamente estas van de la mano con las formas redondas y alargadas que son indicadores de calidad de una tunta, presentándose problemas en la clasificación según a estos aspectos, confundiendo o agrupando muchas veces

las variedades por las semejanzas que presentan. También el producto final presenta tonalidades variadas de color entre blanco y oscuro que a simple vista se podría considerar como perteneciente a un determinado grupo “blanco” de calidad, por lo que es evidente que se cometen errores mediante este criterio de clasificación.

La obtención de un producto de mayor calidad, depende además de la materia prima, de las tecnologías de elaboración y la manipulación que se haga del producto. Si bien en la elaboración de chuño y tunta se usa una metodología tradicional, esta se puede mejorar substancialmente aplicando métodos de selección y clasificación más depuradas y cuidadosas (Guidi, Esprella, Aguilera, & Devaux, 2002). A este problema en la actualiadad se podría solucionar aplicadondo tecnologías de información que permitan analizar las imágenes y tomar decisiones.

Debido a la importancia en tecnologías de información que actualmente está presente y en gran avance para optimizar los procesos. Se ve que la inteligencia artificial ya tiene usos en cuanto a la optimización del rendimiento de la labor humana gracias a los algoritmos eficientes, pero que actualmente no se pone mucho interés en la región para integrar en el manejo de sus procesos de clasificación, pero no solamente ello, sino también la inteligencia artificial interviene en el reconocimiento de objetos, rostros, movimientos, que haciendo uso de estas bondades se optimizaría los trabajos hechos por los seres humanos.

1.2. Formulación del problema

1.2.1. Problema general

¿Cuál es la eficiencia de un sistema de control de calidad mediante procesamiento digital de imágenes y un modelo de machine learning en la clasificación de la tunta en la planta de producción de Kishuara - Andahuaylas?

1.2.2. Problemas específicos

- ¿Cuál es la eficiencia de clasificación del modelo generado con el algoritmo K-Nearest Neighbors de machine learning al evaluar tuntas de cuatro clases?

- ¿Cuál es la eficiencia de clasificación del modelo generado con el algoritmo Decision Trees de machine learning al evaluar tuntas de cuatro clases?

- ¿Cuál es la eficiencia de clasificación del modelo generado con el algoritmo Gaussian Naive Bayes de machine learning al evaluar tuntas de cuatro clases?

- ¿Cuál es la eficiencia de clasificación del modelo generado con el algoritmo Multinomial Naive Bayes de machine learning al evaluar tuntas de cuatro clases?

- ¿Cuál es la eficiencia de clasificación del modelo generado con el algoritmo Support Vector Machines de machine learning al evaluar tuntas de cuatro clases?

1.3. Justificación

Debido a que la producción de la tunta se realiza a partir de variedades de papas nativas procedentes de las cosechas en las alturas, siguiendo con el proceso de preselección de la papa, congelación, lavado minucioso en agua, el descascarado y el secado a la fuerte luz del sol, finalmente la clasificación que al realizarlo manualmente existe la incertidumbre de agrupar erróneamente las tuntas para finalmente envasar.

La clasificación de la tunta por su tamaño, forma y color es dificultosa, demanda tiempo y concentración, requiere de atención y mucha práctica, demanda de mano de obra calificada y el personal debe de desarrollar el sentido de la vista, el tacto y la percepción para obtener un producto estándar para su comercialización.

Para mejorar y acelerar la tarea de clasificación se requiere incrementar el nivel de precisión, lo cual es posible de solucionar con la implementación de un sistema mediante algoritmos de procesamiento digital de imágenes y el entrenamiento de modelos de clasificación. Con cuyas técnicas es posible analizar la imagen y clasificarlos de acuerdo a parámetros de tamaño, forma y color.

Existen algoritmos de machine learning capaces de entrenar grandes cantidades de datos de cualquier origen, solo que estos datos deben estar normalizados en un sistema común para que el algoritmo los reconozca como tal. Debido a que existen algoritmos de clasificación y regresión, cada una posee ciertas

virtudes de clasificación, estos algoritmos entrenan conjuntos de datos con atributos cualitativos o cuantitativos, y cada una posee rendimientos diferentes.

Por otro lado, dada la naturaleza del problema, esta investigación servirá de base para el desarrollo de futuras herramientas de clasificación no solamente de tuntas, sino de cualquier otro producto destinado al comercio.

1.4. Objetivos

1.4.1. Objetivo general

Determinar la eficiencia de un sistema de control de calidad mediante procesamiento digital de imágenes y un modelo de machine learning en la clasificación de la tunta en la planta de producción de Kishuara – Andahuaylas**.**

1.4.2. Objetivos específicos

- Determinar la eficiencia de clasificación del modelo generado con el algoritmo K-Nearest Neighbors de machine learning al evaluar tuntas de cuatro clases.

- Determinar la eficiencia de clasificación del modelo generado con el algoritmo Decision Trees de machine learning al evaluar tuntas de cuatro clases.

- Determinar la eficiencia de clasificación del modelo generado con el algoritmo Gaussian Naive Bayes de machine learning al evaluar tuntas de cuatro clases.

- Determinar la eficiencia de clasificación del modelo generado con el algoritmo Multinomial Naive Bayes de machine learning al evaluar tuntas de cuatro clases.

- Determinar la eficiencia de clasificación del modelo generado con el algoritmo Support Vector Machines de machine learning al evaluar tuntas de cuatro clases.

CAPÍTULO II: MARCO TEÓRICO

2.1. Antecedentes

2.1.1. Antecedentes a nivel nacional

Barriga y Arrasco (2018) en su tesis **"Diagnóstico automático de roya amarilla en hojas de cafeto aplicando procesamiento de imágenes y aprendizaje máquina"** proponen una solución a través del aprendizaje máquina y procesamiento de imágenes, con el fin de automatizar el proceso de detección de la Roya en las hojas y calcular de manera más precisa la severidad del hongo. Para la construcción de los modelos de clasificación se optó por el algoritmo de aprendizaje máquina SVM (Support Vector Machines).

En esta investigación se logró reunir un número suficiente de imágenes, para los experimentos y el desarrollo adecuado de los demás procesos de la investigación. Se implementó en Python el descriptor de textura Haralick, lo cual permitió procesar imágenes y extraer características necesarias para la etapa de clasificación, se implementó también un descriptor basado únicamente en el histograma RGB de la imagen, es decir en la intensidad de los píxeles en cada canal de color. Se generaron diez conjuntos de datos, variando la configuración del descriptor, el espacio de color sobre el cual se aplicaban los métodos y la resolución de las muestras. Se entrenaron diez modelos, todos basados en el algoritmo Support Vector Machines, cuatro de los diez modelos obtuvieron una exactitud en la clasificación superior al 80%, superando el objetivo propuesto. Los dos mejores modelos (87.50% y 95.53%) fueron los entrenados a partir de las características extraídas con el descriptor RGB. El tercer mejor modelo obtuvo una exactitud del 89.40% en muestras de 128x128 de resolución a partir del descriptor de Haralick sobre el espacio de color HSV, con una separación (d=5) entre píxeles de referencia y píxeles vecinos.

Castro (2019) en su tesis titulado **"Aplicación de algoritmos inteligentes para el reconocimiento automático de enfermedades foliares de cultivo de palta"** en el que se aplicaron cuatro algoritmos inteligentes: Naive Bayes, Random Forest, Redes Neuronales y Support Vector Machines. El proyecto tuvo como objetivo general la determinación del algoritmo inteligente más eficaz para el reconocimiento de imágenes de la enfermedad foliar de palta. Se trabajó mediante la metodología de etapas de procesamiento

de imágenes tales como: adquisición de imágenes, preprocesamiento de imágenes, extracción de características y modelado; luego de la evaluación de su eficacia se obtuvo que el algoritmo Máquina de Soporte Vectorial tiene mayor asertividad de 96% en el reconocimiento de enfermedades foliares del cultivo de palta, esto después de ser evaluados con la Matriz de Confusión.

En la etapa de extracción de características se extrajo características de 630 imágenes de hojas de palta sanas y enfermas con 25 características. Las características se establecieron se acuerdo a los canales (R, G, B y Gris) y de estos se extrajo los valores mínimos, primer cuartil, mediana, media, tercer cuartil y máximo de cada canal de color en una imagen, en la fase de modelado se realizó el entrenamiento de datos con los algoritmos mencionados anteriormente y la evaluación de la exactitud. La matriz con los valores extraídos de las imágenes se particionó en dos partes de manera aleatoria: 80% para entrenamiento y 20% para la validación. El dataset de entrenamiento se usó para generar cuatro modelos. De acuerdo al análisis de resultados de cada modelo de clasificación mediante la matriz de confusión se determinó que el algoritmo Support Vector Machines es el más eficaz ya que se obtiene una exactitud (Accuracy) de 96.03%.

Sullca, Molina, Rodríguez y Fernández (2018) presentaron un artículo titulado **"Detección de enfermedades y plagas en las hojas de arándanos utilizando técnicas de visión artificial"** con el objetivo de contruir un sistema experto que detecte la enfermedad o plaga y tomar una acción correctiva a tiempo, se tomó en base a la necesidad de este sector pues este problema afecta críticamente al rendimiento de las cosechas. Una de las áreas consideradas en este trabajo es el uso de visión artificial puesto que se utilizaron técnicas para el procesamiento y análisis de imágenes, imágenes que tuvieron que ser recolectadas por los propios autores de este trabajo recorriendo sembríos de arándanos y tomando fotos a cada planta y con ellos se armó la base de datos; además del uso de machine learning para la utilización de algoritmos de aprendizaje, uno de ellos fue la utilización del clasificador Support Vector Machines(SVM). Los resultados arrojaron un índice de reconocimiento adecuado que fue capaz de clasificar si la planta de arándano estaba siendo afectada por la enfermedad o plaga. La primera fase que se realizó es el input de la base de datos de las fotografías, se recopiló fotos de distintas variedades de plantas de arándanos. Para la extracción de

características se tilizaron los algoritmos HOG (Histograms of Oriented Gradient) y LBP (Local Binary Pattern). Luego de la extracción, se procedió a entrenar la data generada. Los algoritmos que se utilizaron fueron: SVM (Support Vector Machines), Random Forest y una Red Neuronal y probado con las características obtenidas, el mejor resultado obtenido fue una exactitud (Accuracy) del 85.6% demostrada por el modelo entrenado con una Red Neuronal y utilizando vectores característicos extraídos con LBP y que es adecuada para predecir el estado de la planta de arándano. Los autores concluyen, además que el modelo tiene un poder de predicción representativo, es decir puede acertar bien al definir la clase a la cual pertenece (si es una hoja con enfermedad o no) una nueva imagen insertada en el modelo, y debido a que la base de datos de las imágenes fue creada desde cero, fue menos complicado en el preprocesamiento, se tuvo una base de datos limpia y destinada a la elaboración del modelo que se propuso.

2.1.2. Antecedentes a nivel internacional

Sandoval y Prieto (2009) en el artículo denominado **"Procesamiento de imágenes para la clasificación de café cereza"** desarrollaron un sistema que usa técnicas de visión artificial para clasificar frutos de café en ocho categorías según el estado de maduración en el que se encuentre. Las ocho categorías comprenden todo el proceso de maduración del café desde las etapas iniciales hasta sobremaduros y secos. A partir de un conjunto de 9 características que corresponden a 4 de textura, 3 de color y 2 de forma que se obtuvieron mediante la implementación de algoritmos de visión artificial, se logró implementar un clasificador Bayesiano, cuyo desempeño, evaluado mediante el método de validación cruzada, corresponde al 96.88%. El clasificador también muestra una buena capacidad de generalización, que corresponde a la relación entre efectividad y especificidad, como el obtenido en la curva ROC que genera un área del 0.9719 muy cercano al 1 ideal.

Montoya, Cortés y Chaves (2014) en el artículo denominado **"Sistema automático de reconocimiento de frutas basado en visión por computador"** presentan un sistema de reconocimiento capaz de identificar una fruta tropical latinoamericana de entre un conjunto, establecido en una

base de datos, utilizando técnicas de visión por computador. La investigación realizada permitió comparar los clasificadores KNN y bayesiano y los modelos del color RGB y HSV, junto con las características de tamaño y forma usadas previamente por investigadores de esta área en países como Malasia, Brasil y Estados Unidos. Para la clase de frutas definidas en esta investigación se determinó que las características que mejor las describieron fueron los valores medios de los canales RGB y la longitud de los ejes mayor y menor cuando se usaba el clasificador bayesiano, proceso que permitió obtener resultados con una exactitud igual al 90% en las pruebas realizadas, encontrándose que no siempre el seleccionar una mayor cantidad de variables para formar el vector descriptor permite que los clasificadores entreguen una respuesta más acertada, en este sentido es importante considerar que entre las variables de estudio debe presentarse un valor bajo de dependencia o correlación. La síntesis del desarrollo del proyecto dio como resultado la construcción de una báscula electrónica capaz de clasificar frutas, dispositivo que pretende contribuir a la solución del problema de identificación y clasificación de productos agrícolas en los supermercados.

Heras (2017) en un artículo de investigación titulado **"Clasificador de imágenes de frutas basado en inteligencia artificial"** presenta un trabajo en el que se utilizan algoritmos para la construcción de un clasificador de imágenes de frutas basado en la extracción de características del color de las imágenes en determinadas regiones de interés. Para el desarrollo del clasificador de imágenes de frutas se utiliza la técnica de extracción del histograma a color en tres dimensiones y con la implementación de algoritmos de inteligencia artificial se efectúa la clasificación automática de imágenes. El conjunto de imágenes adquiridas consiste en una recolección de internet y la elaboración propia. El conjunto completo resultó en diferente número de fotografías por cada una de las cuatro clases de frutas que en total suman 83 imágenes. Para el set de datos se extrajo las características de imágenes basadas en histogramas de color "3D RGB". Se divide el conjunto de datos en dos subconjuntos y se adquiere uno adicional: conjunto de datos de entrenamiento, prueba y validación. Se implementa el algoritmo de clasificación y se entrena a la máquina de aprendizaje "Random Forest" con el conjunto de datos se entreamiento, luego se evalúa el clasificador con el grupo de datos de prueba. Una vez realizada la prueba se debe analizar el reporte de clasificación, si este tiene un porcentaje alto de recuperación en la

clasificación de imágenes superior al 90% se debe probar el conjunto de datos de validación. Al implementar y ejecutar el algoritmo mencionado, se obtuvo una precisión y un recall del 100%, de esta forma el clasificador esta entrenado y es capaz de identificar y clasificar automáticamente cuatro clases de frutas. El resultado exacto es debido a la poca cantidad de datos que se tiene y además los colores de las frutas (banana, manzana roja, manzana verde y naranja) son muy característicos. La máquina de aprendizaje clasifica las imágenes según al color más cercano al que se le entrenó, por ejemplo, si introducimos una mandarina, es muy probable que se clasifique como naranja, requiriendo el entrenamiento con naranjas y de esa forma se puede aumentar la efectividad del clasificador.

2.2. Marco conceptual

2.2.1. Tunta

Producto similar al chuño con la diferencia de que para la obtención de la tunta se elabora en corrientes de agua, eliminando los compuestos oscuros productos de la oxidación, de color blanquecino, obtenido de tubérculos enteros de papas nativas a través de un proceso de congelación, lavado y secado por exposición al sol. Es una tecnología tradicional de los agricultores del altiplano boliviano – peruano (Guidi et al., 2002).

La tunta se elabora principalmente en la puna, bajo condiciones naturales del clima altiplánico, durante la época seca, cuando la temperatura ambiental está por debajo de 0˚ C. En esta época suceden las heladas, a la vez diariamente ocurre una alta radiación solar. El proceso consiste en el congelamiento de los tubérculos de papa y el remojo de las papas en agua corriente (ríos o lagunillas) por aproximadamente un mes. El proceso es seguido por el pelado manual y el secado hasta obtener tubérculos deshidratados (14% de humedad), de color blanco, forma alargada o redonda y de olor característico (Fonseca et al., 2011).

Figura 1

Muestra de tuntas

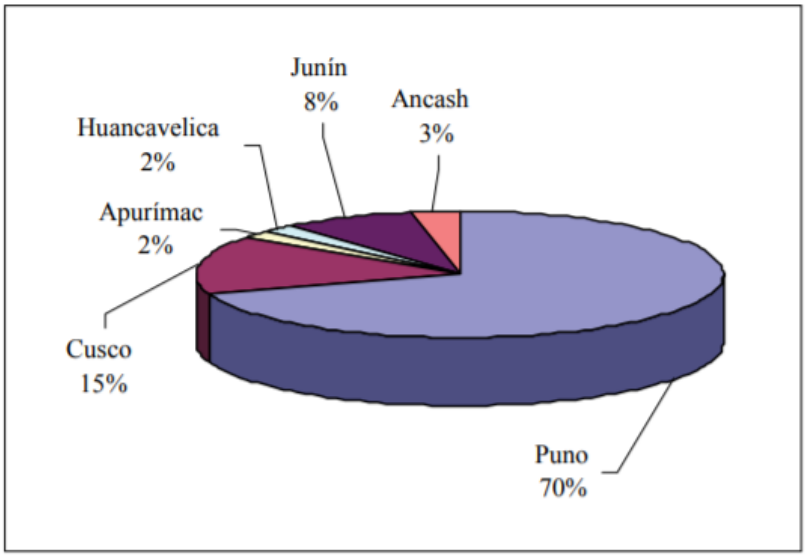
Fuente: (Fonseca et al., 2008).

2.2.2. Producción de la tunta en el Perú

La producción de la tunta viene desde las épocas ancestrales, siguiendo con la costumbre los campesinos como una fuente de ingresos además de que el producto es de abundante composición nutricional. Las zonas productoras de la tunta se encuentran en la región de puna o jalca, sobre los 3,800 m.s.n.m, donde las condiciones climáticas son favorables para su producción. Puno representa el 70% de la producción nacional; el 30 % restante se distribuye en los departamentos de Cuzco, Junín, Huancavelica, Ancash y Apurímac. (Fonseca & Ordinola, 2011).

Figura 2

Distribución porcentual de la producción de la tunta en el Perú



Fuente: (Fonseca & Ordinola, 2011).

2.2.3. Materia prima para la tunta

Fonseca y Ordinola (2011) consideran que el chuño y la tunta responden a dos razones básicas en la lógica de los productores andinos: permiten utilizar un producto que no puede ser consumido en forma fresca, tal como las papas amargas, dado su alto contenido de glicoalcaloides, y está relacionada a la seguridad alimentaria, en la cual el procesamiento permite el almacenamiento de la papa por un periodo largo, especialmente en comunidades donde este tubérculo es la base de la dieta y donde la

estacionalidad de la producción impide comer papa fresca todo el año. El proceso de transformación requiere de pocos insumos principalmente energía solar, agua y mano de obra; en contextos geográficos como la puna, a más de 3 800 msnm donde hay escasez de combustible natural o artificial, y una falta de puestos de trabajo.

Actualmente, las variedades de papa nativa amarga siguen vigentes para la elaboración de tunta, entre las más conocidas están: Locka o Lucky, Palita, Piñaza, pero también las papas libres de glicoalcaloides (*solanum tuberosum*), nativas como la variedad Yana Imilla (para el consumo familiar), y mejoradas, de reciente incorporación, tales como Chaska, Canchán y Perricholi (estas variedades provienen de regiones de menor altitud como Andahuaylas en Apurímac), y tienen un propósito básicamente comercial (Fonseca & Ordinola, 2011).

Tabla 1

Variedades de papa empleadas con mayor frecuencia

Tipo de papa	Especie	Nombre común
Nativa amarga	*S. Juzepczukii*	Piñaza, Lucki, Locka
	S. Curtilobum	Choquepito, Parina
Nativa dulce	*S. tuberosum* spp andigena	Imilla negra, Imilla, blanca, Sani imilla, Peruanita, Palita
Mejorada	*S. tuberosum* spp andigena	Canchán, Ch´aska, Perricholi y otras

Fuente: (Fonseca & Ordinola, 2011).

2.2.4. Definición de los productos de chuño y tunta

Guidi et al. (2002) diferencian los productos chuño y tunta:

a. Chuño. Producto alimenticio de color oscuro, obtenido de tubérculos enteros de papas nativas amargas, a través de un proceso de congelación,

deshidratación y secado por exposición al sol. Esta es una tecnología tradicional utilizada por los productores del altiplano boliviano-peruano.

b. Tunta. Producto alimenticio de color blanquecino, obtenido de tubérculos enteros de papas nativas a través de un proceso de congelación, lavado y secado por exposición al sol. Es una tecnología tradicional de los agricultores del altiplano boliviano-peruano.

Figura 3

Chuño y Tunta

Fuente: (Peñarrieta & Alvarado, 2012).

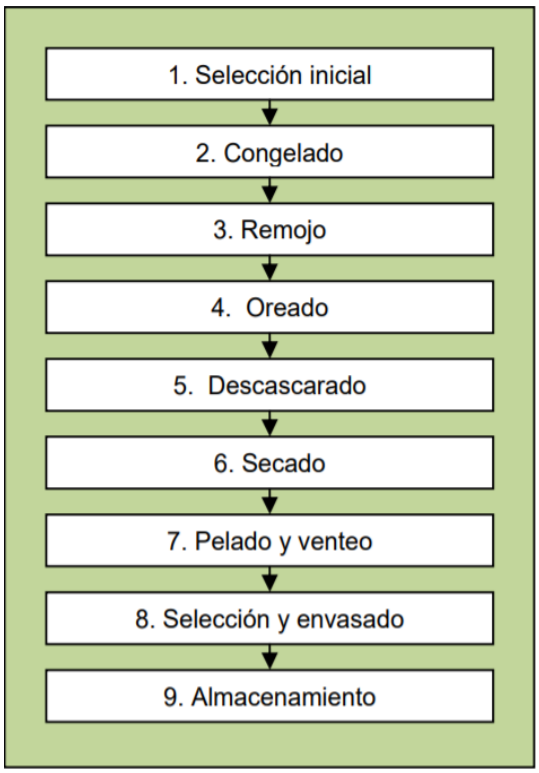
2.2.5. El proceso de elaboración de la tunta

El proceso de elaboración de la tunta se inicia con una selección ligera de los tubérculos de papa recién cosechadas, descartando los tubérculos con daño del gorgojo de los andes (*Prenotripes spp.*) con niveles altos a severos (más del 50% del tubérculo presenta galerías causadas por las larvas). Luego proceden al congelamiento, por 5 a 8 días, para lo cual extienden los tubérculos al aire durante la noche, sobre un piso ligero de paja, y lo recogen al día siguiente muy temprano, cubriéndolo con mantas de lana o plástico para protegerlo del sol y no se oscurezca. Una vez congelada la papa, le sumergen generalmente en el río, durante 15 a 20 días para después realizar el descascarado y el enjuague; de inmediato procede el secado, para lo cual

extienden la tunta sobre una delgada capa de paja, durante 5 a 8 días, bajo una fuerte intensidad solar. Culminan el proceso con el pelado final, la selección y el envasado de la tunta, para luego transportarla a los almacenes acondicionados dentro de sus casas (Fonseca et al., 2011).

Figura 4

Diagrama del proceso de elaboración de la tunta



Fuente: (Fonseca et al., 2011).

2.2.6. Características de la tunta

La tunta se caracteriza por ser un producto deshidratado (14 % de humedad) como indican Fonseca y Ordinola (2011) conserva la forma del tubérculo de la variedad de papa que le dio origen; puede ser de forma redonda o alargada. Presenta ojos profundos o superficiales; es de color blanco, de peso muy ligero y su factor de conversión es de 7 a 1 (requiere 7 kg de papa fresca para producir 1 kg de tunta).

Los expertos en cocina afirman que la tunta combina muy bien con producto de sabor fuerte, por ejemplo, en un chupe de camarones la tunta

suele tomar el gusto a camarones ya que absorbe su sabor, también combina bien con las especies, como las hierbas andinas (romero, huacatay, menta y otros) y el anís chino. Un alimento de fácil digestibilidad por su alto contenido de almidón, especial para la alimentación de infantes y adultos mayores; además los pobladores rurales basados en su experiencia afirman que la tunta alivia los problemas estomacales, como la gastritis (Fonseca & Ordinola, 2011).

Como se verá en la Tabla 2 constituye una fuente de alimento rico en calorías (323 Kcal) apropiado para resistir el clima adverso de los andes y el fuerte trabajo en el campo: contiene minerales como calcio, fósforo y hierro, elementos escasos en muchos alimentos de consumo masivo como el arroz y los fideos.

A continuación, la composición nutricional de la tunta en 100 gramos:

Tabla 2

Composición nutricional de la tunta en 100 gramos

Elementos básicos	Cantidad
Energía (Kcal)	323
Agua (ml)	18.10
Proteína (g)	1.90
Carbohidratos (g)	77.70
Fibra (g)	2.10
Calcio (mg)	92
Fósforo (mg)	54
Hierro (mg)	3.3

Fuente: (Fonseca & Ordinola, 2011).

Fonseca et al. (2008) indican que un lote de tunta de buena calidad desde el punto de vista comercial debe reunir siete características básicas: forma, color, tamaño, rehidratación, sabor, textura y olor.

- **Forma**: De forma redonda o alargada de acuerdo a la variedad de papa empleada. Ejemplo: Chaska: redonda y Locka: alargada.

- **Color**: El color, de preferencia, debe ser blanco intenso, pero la gama de color puede extenderse hasta blanco-mate. No es recomendable la presencia de manchas amarillentas o de color oscuro.

- **Olor**: El olor de un alimento contribuye al placer de comer. Debe tener olor suave, a hierbas acuáticas. No debe presentar olores fuertes o fétidos.

- **Tamaño**: Se consideran 3 categorías, entre grandes, medianos y pequeños.

- **Rehidratación**: Es el tiempo que debe remojarse la tunta para que se ablande y pueda cocinarse fácilmente. Este tiempo puede variar entre 10 minutos a media hora.

- **Textura**: Suave y esponjosa, en especial las variedades nativas.

- **Sabor**: Agradable, ligeramente insípido. Combina bien con comidas de sabores fuertes. Tiene la propiedad de absorber los sabores de los ingredientes que la acompañan durante su cocción.

2.2.7. Clasificación de la tunta

De acuerdo a la Norma Técnica Peruana, los tamaños de la tunta de acuerdo a su forma, se establece de la siguiente manera:

Tabla 3

Tamaños de tunta de acuerdo a su forma

	Forma	
Tamaño	Redondos (cm)	Alargados (cm)
Grande	5,1 a mas	7,0 a mas
Mediana	3,9 – 5,0	5,5 – 7,0
Pequeña	3,9 a menos	5,5 a menos

Tamaño expresado en diámetro ecuatorial.

Fuente: (Norma Técnica Peruana, 2007).

La clasificación de la tunta como labor final, requiere de especial cuidado porque ello influirá en el precio y su destino comercial:

Tabla 4

Criterios de clasificación de la tunta

Primera	Tamaño grande, color blanco intenso, entera, sin grietas.
Segunda	Tamaño mediano, color blanco intenso, entera, sin grietas.
Tercera	Tamaño pequeño, color blanco a cremoso, enteras o con ligeras grietas.
Cuarta	Color crema, con manchas amarillas u oscuras, diferente tamaño, partidas y con grietas (sólo para uso casero).

Fuente: (Fonseca et al., 2008).

2.2.8. Visión Artificial

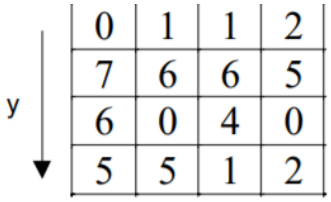
La visión artificial es una disciplina que engloba todos los procesos y elementos que proporcionan ojos a una máquina, deduce la estructura y propiedades geométricas y materiales de imágenes digitales. La visión, tanto para un hombre como para un ordenador, consta principalmente de dos fases: captar la imagen e interpretarla. El ojo del ordenador es la cámara de video, y su *retina* un sensor que es sensible a la intensidad luminosa, así que en la visión artificial lo que resta es interpretar las imágenes, distinguir los objetos de la escena, extraer información de ellos y resolver aspectos más particulares según las necesidades que se deseen satisfacer (Gonzáles et al., 2006).

2.2.9. Imagen digital

Una imagen es un arreglo bidimensional de píxeles con diferente intensidad luminosa (escala gris).

Figura 5

Imagen de 16 píxeles



Fuente: (Esqueda, 2002).

Si la intensidad luminosa de cada píxel se representa por n bits entonces existirán 2^n escala de grises diferentes.

Matemáticamente, una imagen se representa por $r = f(x, y)$, donde r es la intensidad luminosa del píxel cuyas coordenadas son (x, y). Una función matemática para procesar imágenes se representa como $g(x, y) = T[f(x, y)]$ (Esqueda, 2002).

Pixel es cada uno de los puntos que compone la matriz de una imagen digital. Es la unidad mínima de visualización de una imagen digital.

2.2.10. Espacios de color de una imagen

2.2.10.1. Espacio de color RGB

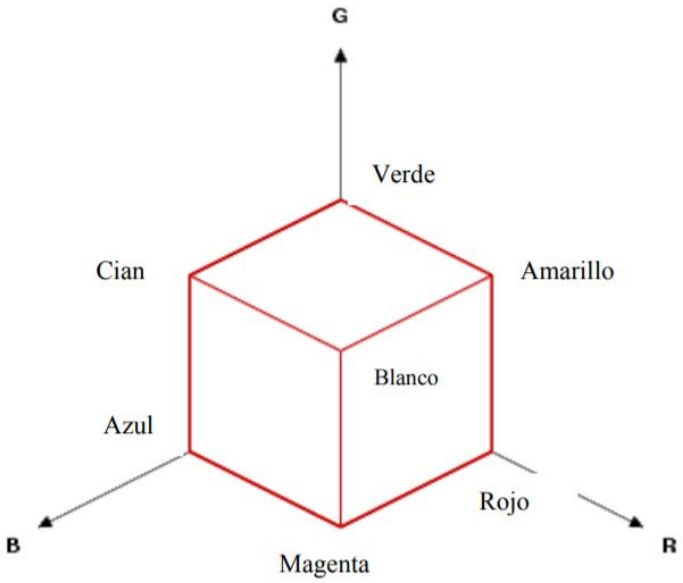
El espacio RGB es el espacio de color más extendido y el que utilizan la mayoría de cámaras de video y fotográficas para construir una imagen de color. Y de ahí, su importancia en visión artificial ya que trabajar con el mismo espacio de color con el que trabaja la cámara con la que se capturan las imágenes permite evitar la alteración de las propiedades del color durante el proceso de segmentación, propia de los errores de conversión y transformación, y por otro lado conseguir una mayor velocidad de segmentación por ahorro de esas operaciones de conversión y redondeo.

El espacio de color RGB se representa como un cubo donde un color viene definido por la mezcla de valores de intensidad de tres colores primarios: rojo, verde y azul. Un color viene descrito por una tupla de 3 coordenadas en el cubo. El color negro se representa por (r=0, g=0, b=0) y el color blanco se

representa por (r=255, g=255, b=255). La gama acromática de escala de grises está representada por la diagonal del cubo (Gil & Torres F, 2004).

Figura 6

Representación del espacio RGB



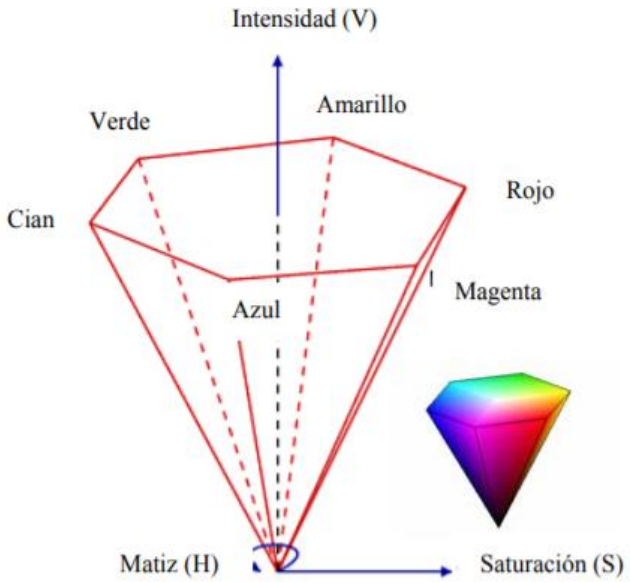
Fuente (Gil & Torres F, 2004).

2.2.10.2. Espacio de color HSV

El espacio de color HSV representa uno de los espacios de coordenadas más clásicos e intuitivos existentes en la literatura. Su interpretación geométrica viene determinada por un cono de base quasi-hexagonal. Con esta representación del espacio de color, cada color trabaja con 3 componentes básicas: matiz, saturación y brillo. El matiz, H, hace referencia al valor de cromaticidad o clase de color. La saturación, S, se refiere a las longitudes de onda que se suman a la frecuencia del color, y determina la cantidad de blanco que contiene un color. Contra menos saturado este un color más cantidad de blanco, y contra más saturado este un color menor cantidad de blanco. En definitiva, la saturación representa la pureza e intensidad de un color. Así, la falta de saturación viene dada por la generatriz en la representación del cono HSV. Esa falta de saturación representa la gama de grises desde el blanco hasta el negro. La luminancia, V, se corresponde con la apreciación subjetiva de claridad y oscuridad (Gil & Torres F, 2004).

Figura 7

Espacio de colores HSV



Fuente: (Gil & Torres F, 2004)

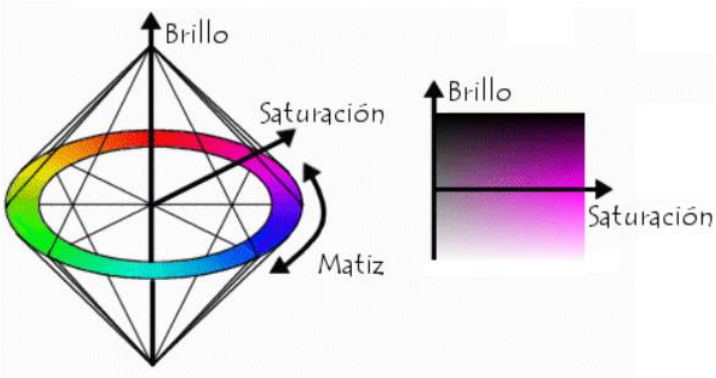
2.2.10.3. Espacio de color HSL

El espacio HSL (del inglés Hue, Saturation, Lightness – Matiz, Saturación, Luminosidad), también conocido como HLS o HSI, define un espacio de color en términos de sus componentes constituyentes. Se representa gráficamente como un cono doble o un doble hexágono. Los dos vértices en el espacio HSL se corresponden con el blanco y el negro, el ángulo se corresponde con el matiz, la distancia al eje con la saturación y la distancia al eje blanco-negro se corresponde a la luminancia. Al igual que el espacio HSV, es una deformación no lineal del espacio de color RGB.

El gráfico siguiente es una representación del modelo HSL, donde el color se representa mediante un círculo cromático, y la luminancia y saturación se representan mediante dos ejes.

Figura 8

Bicono del modelo de color HSL



Fuente: (Pillou, 2008).

Comparación entre HSL y HSV

HSL es similar al modelo HSV, pero refleja mejor la noción intuitiva de la saturación y la luminancia.

- En HSL, la componente de la saturación va desde el completamente saturado hasta el gris equivalente, mientras que en HSV, con V al máximo, va desde el color saturado hasta el blanco, lo que no es muy intuitivo.

- La luminancia en HSL siempre va desde el negro hasta el blanco pasando por la tonalidad deseada, mientras que en HSV la componente V se queda a mitad camino, entre el negro y la tonalidad escogida.

2.2.11. Procesamiento digital de imágenes

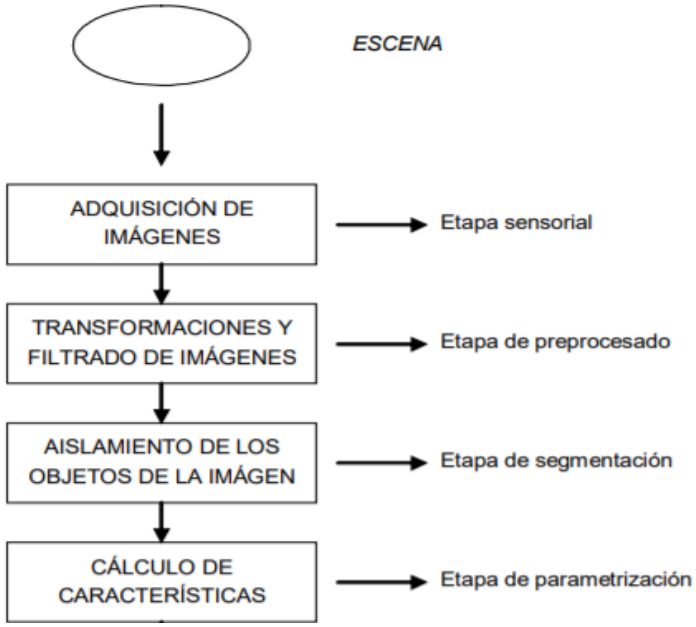
El procesamiento digital de imágenes es un proceso a través del cual se extrae una imagen del mundo real y se produce una versión modificada de la misma por medio de algoritmos para su posterior análisis. Se conoce como análisis de imágenes al proceso mediante el cual se parte de una imagen para poder obtener una medición, interpretación o decisión dentro de un proceso específico (Constante & Gordón, 2015).

2.2.12. Etapas del procesamiento de imágenes

El primer paso en el proceso es adquirir la imagen digital, para el siguiente paso, el preprocesamiento de dicha imagen, para mejorar la imagen de forma que el objetivo final tenga mayores posibilidades de éxito. El paso siguiente es la segmentación, su objetivo es dividir la imagen en las partes que la constituyen o los objetos que la forman. La salida del proceso de segmentación es una imagen de datos que, o bien contiene la frontera de la región o los puntos de ella misma, es necesario convertir estos datos a una forma que sea apropiada para el ordenador. La parametrización, que recibe también el nombre de selección de rasgos, se dedica a extraer rasgos que producen alguna información cuantitativa de interés o rasgos que son básicos para diferenciar una clase de objetos de otra.

Figura 9

Etapas del procesamiento de imágenes



Fuente: (Gonzáles et al., 2006).

2.2.12.1. Adquisición de imágenes

En primer paso se trata de conseguir que la imagen sea lo más adecuada posible para que se pueda continuar con las siguientes etapas, una correcta adquisición de la imagen supone un paso muy importante para que el proceso de reconocimiento tenga éxito. Dentro de esta etapa existen múltiples factores que atañen directamente al proceso de captura de la imagen, formadas fundamentalmente por: el hardware (cámara, óptica, tarjeta de adquisición de imagen, ordenador y software) y el entorno de posicionamiento de los elementos (la iluminación, el fondo, la posición correcta de la cámara, ruido eléctrico-óptico externo, etc.).

2.2.12.2. Transformaciones y filtrado de imágenes

Etapa de preprocesamiento de la imagen:

a. Conversión de imágenes a escala de grises.

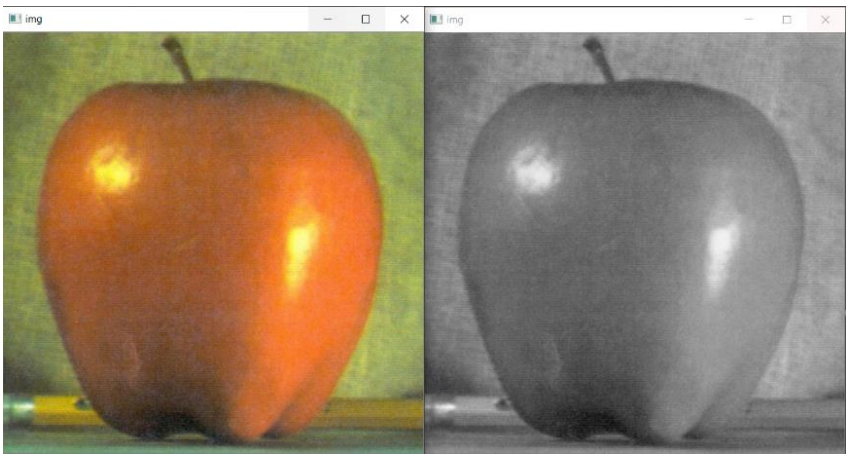
Para la manipulación de imágenes es necesario convertirlas en escala de grises, lo que implica convertirlas en imágenes de tipo binario asignando a cada pixel un solo valor, simplificando y agilizando el procesamiento.

El siguiente algoritmo es una expresión matemática que convierte los valores RGB de cada pixel en valores de escala de grises eliminado la información de matiz y saturación mientras conserva la luminancia, con la suma ponderada de los componentes R, G y B:

$$Gris = 0.2989 * R + 0.5870 * G + 0.1140 * B$$

Figura 10

Conversión de una imagen a escala de grises



Fuente: Elaboración propia.

b. Difuminación de imágenes mediante filtro gaussiano.

El filtro gaussiano es uno de los más aplicados y robustos ya que sigue leyes probabilísticas a través de desviaciones estándares que permiten descartar los factores externos que influyen en una imagen.

Simulan una distribución gaussiana bivariante. El valor máximo aparece en el pixel central y disminuye hacia los extremos tanto más rápido cuanto menor sea el parámetro de desviación típica s. El resultado será un conjunto de valores entre 0 y 1. Para transformar la matriz a una matriz de números enteros se divide toda la matriz por el menor de los valores obtenidos. La ecuación para calcularla es:

$$g(x,y) = e^{-\frac{x^2+y^2}{2*s^2}}$$

$$G(x,y) = \frac{g(x,y)}{min_{x,y}(g(x,y))}$$

1	4	7	4	1
4	20	33	20	4
7	33	55	33	7
4	20	33	20	4
1	4	7	4	1

Filtro gaussiano con s=1 y r=2.

0	0	0
0	1	0
0	0	0

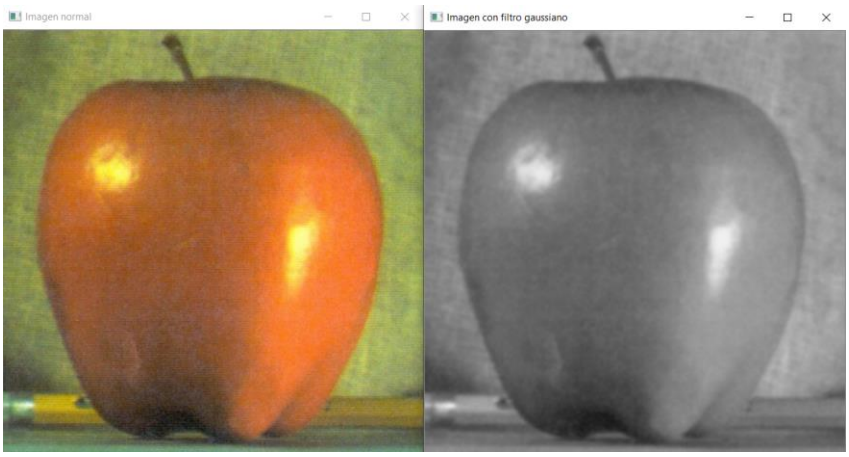
DIV 1

Matriz de filtrado identidad.

Son una manera de obtener filtros de tipo genérico. Pueden ser útiles, por ejemplo, cuando se asume que la respuesta espectral de un pixel es función de la reflectividad de los pixeles vecinos atenuada en función de la distancia. El alcance de esta atenuación (r) viene marcado por el tamaño de la ventana de filtrado (w = 2r + 1) que debe especificarse previamente.

Figura 11

Imagen normal e imagen con filtro gaussiano



Fuente: Elaboración propia.

c. Umbralización

La umbralización es una técnica de segmentación simple y eficiente que permite separar los pixeles de una imagen en escala de grises en dos categorías a partir de un valor umbral de intensidad.

- **Umbral global**

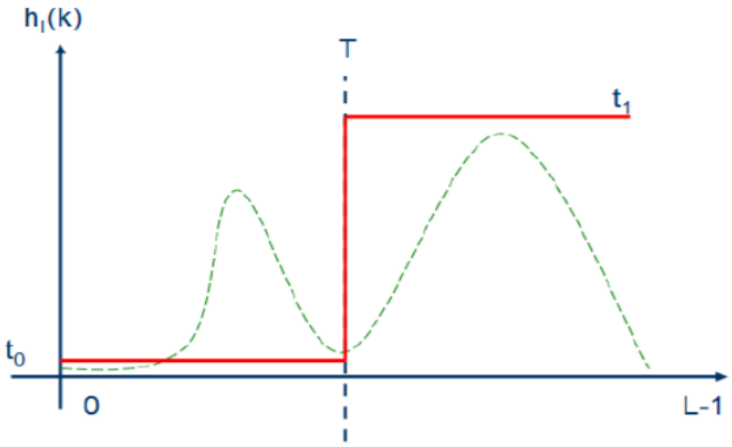
Un umbral fijo o global T, es aquel que es único sobre toda la imagen.

$$b(r,c) = \begin{cases} t_0 \; si \; I(r,c) < T \\ t_1 \; si \; I(r,c) \geq T \end{cases}$$

Donde $b\ (r,c)$ es la intensidad y $t_0\ y\ t_1$ son los dos posibles niveles de gris del pixel umbralizado. Cuando el valor $t_0 = 0$ y el valor $t_1 = 1$ se dice que la imagen se ha binarizado. Esta estrategia puede resultar muy útil cuando se tiene una iluminación homogénea sobre toda la imagen y un histograma bimodal, como el que se muestra en la siguiente figura (Triana, Jaramillo, Gutiérrez, & C, 2016):

Figura 12

Histograma bimodal



Fuente: (Triana, Jaramillo, Gutiérrez, & C, 2016).

T es el valor de intensidad umbral y $t_0\ y\ t_1$ son los dos posibles niveles de gris de los pixeles de la imagen umbralizada.

- **Umbralización de Otsu**

Ideado por Nobuyuki Otsu (1979), este procedimiento no paramétrico selecciona el umbral óptimo maximizando la varianza entre clases mediante una búsqueda exhaustiva. La varianza entre clases se define como una suma

ponderada de las varianzas. El método Otsu no precisa información previa de la imagen antes de su procesamiento, ni supervisión humana para el cálculo de los umbrales (Magro, 2013).

$$P_i = \frac{f_i}{N}$$

P_i es la probabilidad de ocurrencia del nivel de gris de una imagen según Otsu. f_i = frecuencia de repetición del nivel de gris.

En la umbralización de dos niveles se calcula la contingencia de distribución de ambas clases. Los píxeles se dividen en dos clases con diferentes niveles de gris respectivamente:

$$C_1 : \frac{P_1}{\omega_1(t)}, \dots, \frac{P_t}{\omega_1(t)}$$

$$C_2 : \frac{P_{t+1}}{\omega_2(t)}, \frac{P_{t+2}}{\omega_2(t)}, \dots, \frac{P_L}{\omega_2(t)}$$

$$\omega_1(t) = \sum_{i=1}^{t} P_i \qquad \omega_2(t) = \sum_{i=t+1}^{L} P_i$$

$$u_1 = \sum_{i=1}^{t} \frac{iP_i}{\omega_1(t)} \qquad u_2 = \sum_{i=t+1}^{L} \frac{iP_i}{\omega_2(t)}$$

Umbralización de dos niveles según Otsu, los píxeles se dividen en dos clases, C_1 y C_2, con niveles de gris $[1, 2, \dots, t]$ y $[t+1, t+2, \dots, L]$ respectivamente, y se definen las medias para cada una de las clases.

Figura 13

Imagen normal e imagen umbralizada con el método Otsu

Fuente: Elaboración propia.

d. Reducción del ruido en imágenes mediante transformaciones morfológicas

- **Erosión binaria**

La transformación de la erosión es el resultado de comprobar si el elemento estructurante B está completamente incluido dentro del conjunto X. Cuando no ocurre, el resultado de la erosión es el conjunto vacío: ε

$$\varepsilon_B(X) = X\Theta B = \{x \mid B_x \subseteq X\}$$

Cuando los objetos de la escena sean menores que el elemento estructurante, éstos desaparecerán. Otra interpretación de la erosión supone tomar el valor mínimo de la imagen en el entorno de vecindad definido por el elemento estructurante.

La erosión supone una degradación de la imagen. La aplicación iterativa de esta transformación hará que se eliminen todos los objetos existentes en la imagen. La erosión es una transformación antiextensiva:

$$\varepsilon_B(X) \subseteq X$$

- **Dilatación binaria**

La dilatación es la transformación dual a la erosión. El resultado de la dilatación es el conjunto de elementos tal que al menos algún elemento del

conjunto estructurante B está contenido en el conjunto X, cuando B se desplaza sobre el conjunto X:

$$\delta_B(X) = X \oplus B = \{x \mid X \cap B_x \neq \emptyset\}$$

Esta operación representa un crecimiento progresivo del conjunto X. Al pasar el elemento estructurante dentro del conjunto, éste no se modificará. Sin embargo, en la frontera del conjunto X, al desplazar a B, el conjunto resultado se expansionará. La aplicación iterada de este operador haría degradar la imagen, haciendo coincidir el conjunto dilatado con la imagen. La dilatación es una transformación extensiva:

$$X \subseteq \delta_B(X)$$

La dilatación también se interpreta como el valor máximo del entorno de vecindad definido por el elemento estructurante.

- **Apertura binaria**

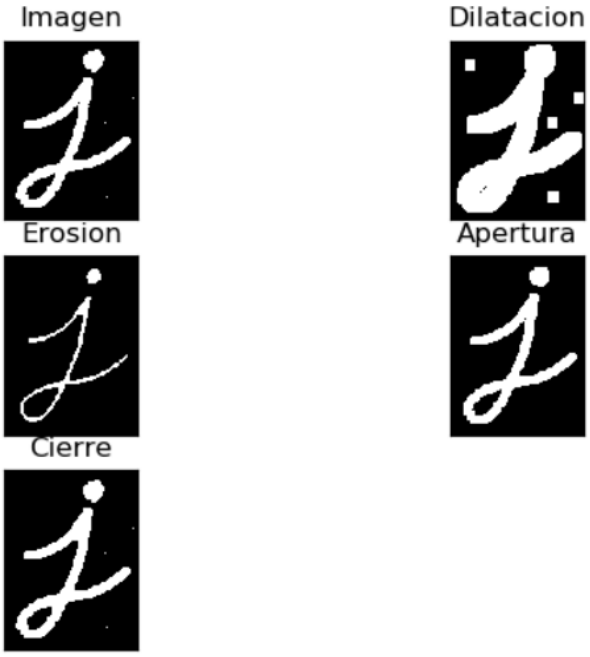
La apertura binaria elimina todos los objetos que no están completamente contenidos en el elemento estructurante, esta operación puede ser ideal para la eliminación de ruido.

- **Cierre binario**

El cierre binario produce que la dilatación rellene las estructuras que la erosión no puede separar. Los contornos de los objetos también serán suavizados, pero habiendo rellanado las fisuras.

Figura 14

Imagen con transformaciones morfológicas



Fuente: Elaboración propia.

2.2.12.3. Segmentación

El separar la imagen en unidades significativas es un paso importante en visión computacional para llegar al reconocimiento de objetos. Este proceso se conoce como segmentación. El objetivo de este paso es determinar las regiones; es decir, las partes o segmentos que puedan considerarse como unidades significativas. Esto puede ayudar a obtener una visión más compacta de la información de bajo nivel, ya que, en vez de miles o millones de píxeles, se puede llegar a decenas de regiones (Sucar & Gómez).

2.2.12.4. Cálculo de características

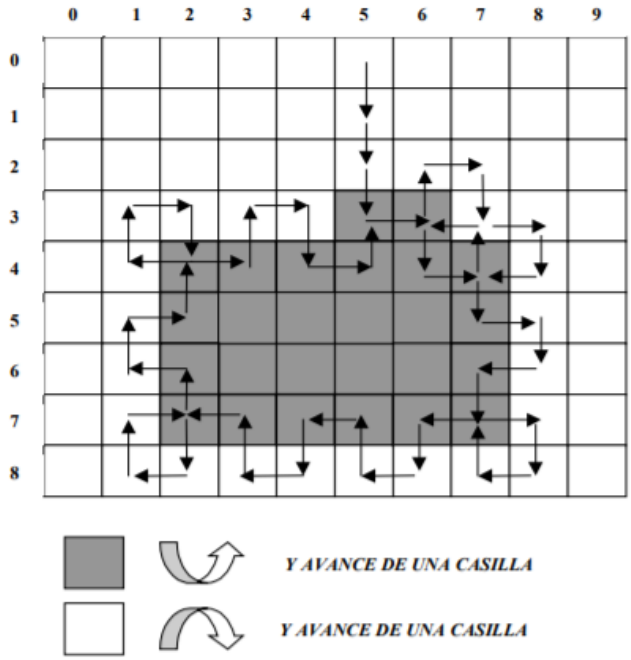
Seguimiento de contorno para el cálculo de características

Como el nombre indica, los algoritmos de seguimiento de contorno trazan fronteras ordenando los puntos de tipo borde sucesivo. Un algoritmo muy utilizado para determinar los puntos del contorno de un objeto es el algoritmo de la tortuga. El algoritmo realiza los siguientes pasos: Se avanza pixel a pixel

hasta encontrar un pixel negro correspondiente al contorno a determinar. En la figura 15 se puede apreciar que el objeto está formado por puntos negros. El algoritmo avanza de la siguiente manera: si está en un punto negro gira 90° en sentido contrario a las agujas del reloj y avanza un pixel, si en cambio está en un punto blanco gira 90° en sentido de las agujas del reloj y avanza un pixel. Se van almacenando las coordenadas de todos los pixeles negros que se van encontrando. El algoritmo termina cuando llega de nuevo al primer punto negro encontrado. Como en el vector del contorno aparecen algunos puntos repetidos (algunas esquinas), se eliminan. En la figura 15 se puede observar que este algoritmo se desplaza bordeando todo el objeto, una vez obtenido el vector de contornos se puede analizar para determinar gran cantidad de parámetros del objeto, como: área, perímetro, diámetro equivalente, ancho, altura, etc (Gonzáles et al., 2006).

Figura 15

Funcionamiento del algoritmo de la tortuga



Contorno= (5,3), (6,3), (6,3), (6,4), (7,4), (7,4), (7,5), (7,6), (7,7), (7,7), (6,7), (5,7), (4,7), (3,7), (2,7), (2,7), (2,6), (2,5), (2,4), (2,4), (3,4), (4,4), (5,4),(5,3)
Contorno Filtrado= (5,3), (6,3), (6,4), (7,4), (7,5), (7,6), (7,7), (6,7), (5,7), (4,7), (3,7), (2,7), (2,6), (2,5), (2,4), (3,4), (4,4), (5,4)

Fuente: (Gonzáles et al. 2006).

2.2.13. Machine Learning

2.2.13.1. Concepto de machine learning

El aprendizaje automático es parte de la IA (Inteligencia Artificial) que permite que un sistema aprenda de los datos en lugar de a través de la programación explícita. El aprendizaje automático utiliza una variedad de algoritmos que aprenden iterativamente de los datos para mejorar, describir datos y predecir resultados. A medida que los algoritmos ingieren datos de entrenamiento, es posible producir modelos más precisos basados en esos datos (Hurwitz & Kirsch, 2018).

2.2.13.2. La muestra de aprendizaje

Para poder realizar el cálculo de las funciones discriminantes suele ser precisa la existencia de un conjunto de patrones similares a los que se desea reconocer, que se denomina conjunto de aprendizaje. El conjunto de aprendizaje debe ser por tanto un subconjunto representativo del universo de trabajo. Cuando la muestra es abundante suele crearse otro conjunto con ella. Este segundo conjunto se utiliza para probar los resultados, y se conoce como conjunto de test. Es importante que estos dos conjuntos sean independientes, como norma general, en el caso de un universo de trabajo grande, es suficiente con asegurar que el conjunto de aprendizaje y el test no tengan elementos en común. De esta forma puede probarse que el clasificador desarrollado ha adquirido la propiedad de generalización. Esta propiedad garantiza que un sistema clasifica correctamente patrones que no ha visto durante el proceso del entrenamiento (Velez et al., 2003).

2.2.13.3. Modelo de datos

Un modelo de aprendizaje automático es el resultado generado cuando entrena un algoritmo de aprendizaje automático con datos. Después del entrenamiento, cuando proporciona un modelo con una entrada, se le dará una salida. Por ejemplo, un algoritmo predictivo creará un modelo predictivo. Luego, cuando proporcione datos al modelo predictivo, recibirá una predicción basada en los datos que capacitaron al modelo. El aprendizaje automático ahora es esencial para crear modelos analíticos (Hurwitz & Kirsch, 2018).

2.2.13.4. Aprendizaje supervisado

En el entrenamiento de los algoritmos de aprendizaje supervisado, además de los datos necesarios para realizar la predicción, es necesario disponer de una característica objetivo para cada una de las instancias. Siendo este el valor que el modelo ha de reproducir. Pudiendo ser este el valor tanto de tipo numérico como categórico. Una vez finalizado el proceso de entrenamiento, el valor objetivo ya no es necesario, ya que es la predicción que realiza el modelo. Solamente es necesario el resto de las características. A partir de este momento, el valor objetivo únicamente se suele utilizar en los procesos de validación. Para comprobar si el modelo sigue siendo válido (Rodríguez, 2018).

2.2.13.5. Aprendizaje no supervisado

A diferencia de los algoritmos de aprendizaje supervisado, en los no supervisados no es necesario disponer de la respuesta correcta en los datos de entrenamiento. Ya que no se busca la reproducción de un resultado conocido, sino el descubrimiento de nuevos patrones o resultados (Rodríguez, 2018).

2.2.13.6. Support Vector Machines

Las máquinas de vectores de soporte (Support Vector Machines SVM) tienen su origen en los trabajos sobre la teoría del aprendizaje estadístico y fueron introducidas en los años 90 por Vapnik y sus colaboradores. Aunque originalmente las SVMs fueron pensadas para resolver problemas de clasificación binaria, actualmente se utilizan para resolver otros tipos de problemas (regresión, agrupamiento, clasificación multiclase). Pertenecen a la categoría de los clasificadores lineales, puesto que inducen separadores lineales, también llamados hiperplanos en un espacio. La idea es seleccionar un hiperplano de separación que equidista de los ejemplos más cercanos de cada clase para, de esta forma, conseguir lo que se denomina un margen máximo a cada lado del hiperplano. Además, a la hora de definir el hiperplano, sólo se consideran los ejemplos de entrenamiento de cada clase que caen justo en la frontera de dichos márgenes. Estos ejemplos reciben el nombre de vectores soporte (Carmona, 2016).

Hiperplano de margen máximo

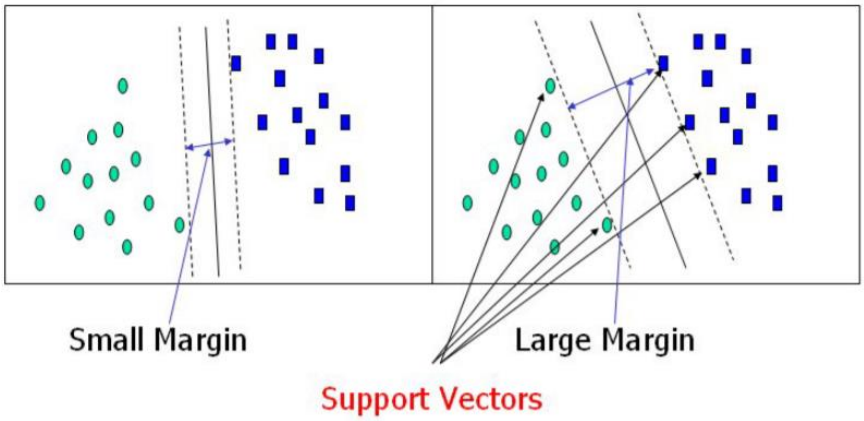
Hiperplano que proporciona la máxima separación entre dos clases linealmente separables (Gonzales, 2011).

Vectores de soporte

Instancias más próximas, de cada clase, al hiperplano de margen máximo (al menos uno por clase). El conjunto de Vectores de Soporte define de forma única el hiperplano de margen máximo (Gonzales, 2011).

Figura 16

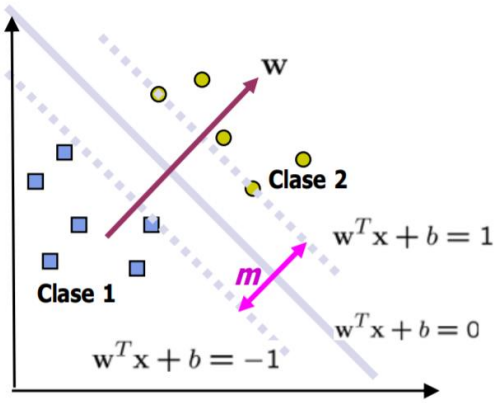
Hiperplanos de margen pequeño y máximo



Fuente: (Gonzales, 2011).

Figura 17

Ejemplo del hiperplano construido por un SVM para separar dos clases



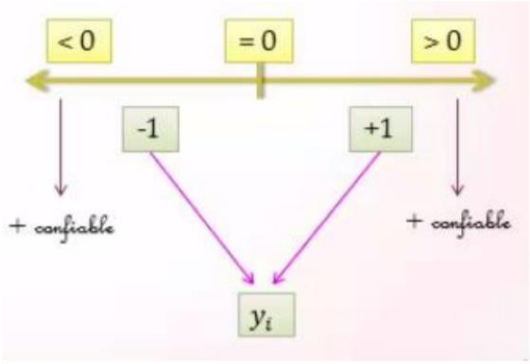
Fuente: (Betancourt, 2005).

El hiperplano es un subespacio unidimensional descrito por la siguiente ecuación: $b + W1 * X1 + W2 * X2 + W3 * X3 \ldots = 0$. O como se puede observar en la figura 17; simplemente un vector de pesos (W) transpuesto por un vector de características (X).

Como se puede ver en la figura 18, el valor del hiperplano puede tomar valores positivos o negativos, haciendo referencia esto a cada una de las clases. Es aquí donde surge el concepto de confiabilidad, entre más positivo sea el valor tendrá más confiabilidad de ser de una clase y entre más negativo, más confiabilidad habrá de que pertenezca a la otra clase.

Figura 18

Imagen representativa de la confiabilidad de acuerdo con el resultado arrojado por la ecuación del hiperplano



Fuente: (Gamarra & Ríos).

2.2.13.7. K-Nearest Neighbors

K vecinos más cercanos, la idea sobre la que se fundamenta este paradigma es que un nuevo caso se va a clasificar en la clase más frecuente a la que pertenecen sus K vecinos más cercanos (Moujahid, Inza, & Larrañaga).

El algoritmo K-NN

La notación a utilizar es el siguiente:

Figura 19

Notación para el paradigma K-NN

		X_1	...	X_j	...	X_n	C
$(\mathbf{x}_1, c_1)$	1	x_{11}	...	x_{1j}	...	x_{1n}	c_1
	⋮	⋮		⋮		⋮	⋮
$(\mathbf{x}_i, c_i)$	i	x_{i1}	...	x_{ij}	...	x_{in}	c_i
	⋮	⋮		⋮		⋮	⋮
$(\mathbf{x}_N, c_N)$	N	x_{N1}	...	x_{Nj}	...	x_{Nn}	c_N
$\mathbf{x}$	$N+1$	$x_{N+1,1}$	...	$x_{N+1,j}$	...	$X_{N+1,n}$	?

Fuente: (Moujahid, Inza, & Larrañaga).

- D indica un fichero de N casos, cada uno de los cuales está caracterizado por n variables predictoras, $X1,\ldots,Xn$ y una variable a predecir, la clase C.
- Los N casos se denotan por:

$$(x_1, c_1), \ldots, (x_N, c_N) \quad donde$$
$$x_i = (x_{i,1}, x_{i,n}) \text{ para todo } i = 1, \ldots, N$$
$$c_i \in \{c^1, \ldots, c^m\} \text{ para todo } i = 1, \ldots, N$$

$c^1, \ldots, c^m$ denotan los m posibles valores de la variable C.

- El nuevo caso que se pretende clasificar se denota por $x = (x1,\ldots,x_n)$.

Pseudocódigo para el clasificador K-NN:

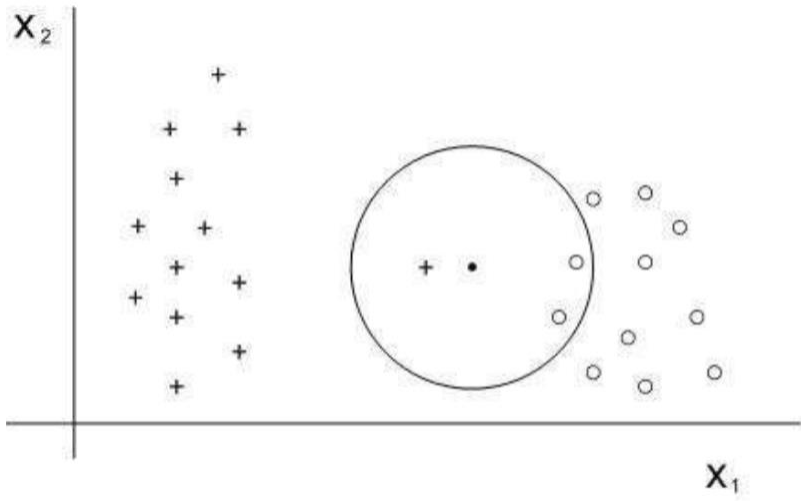
COMIENZO
  Entrada: $D = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_N, c_N)\}$
    $\mathbf{x} = (x_1, \ldots, x_n)$ nuevo caso a clasificar
  PARA todo objeto ya clasificado (x_i, c_i)
    calcular $d_i = d(\mathbf{x}_i, \mathbf{x})$
  Ordenar $d_i (i = 1, \ldots, N)$ en orden ascendente
  Quedarnos con los K casos $D_{\mathbf{x}}^K$ ya clasificados más cercanos a $\mathbf{x}$
  Asignar a $\mathbf{x}$ la clase más frecuente en $D_{\mathbf{x}}^K$
FIN

Moujahid, Inza, y Larrañaga presentan un pseudocódigo para el clasificador K-NN básico. Tal y como puede observarse en el mismo, se

calculan las distancias de todos los casos ya clasificados al nuevo caso x, que se pretende clasificar. Una vez seleccionados los K casos ya clasificados, D_x^k más cercanos al nuevo caso x, a éste se le asignará la clase (valor de la variable C) más frecuente de entre los K objetos, D_x^k.

Figura 20

Ejemplo de aplicación del algoritmo K-NN básico



Fuente: (Moujahid, Inza, & Larrañaga).

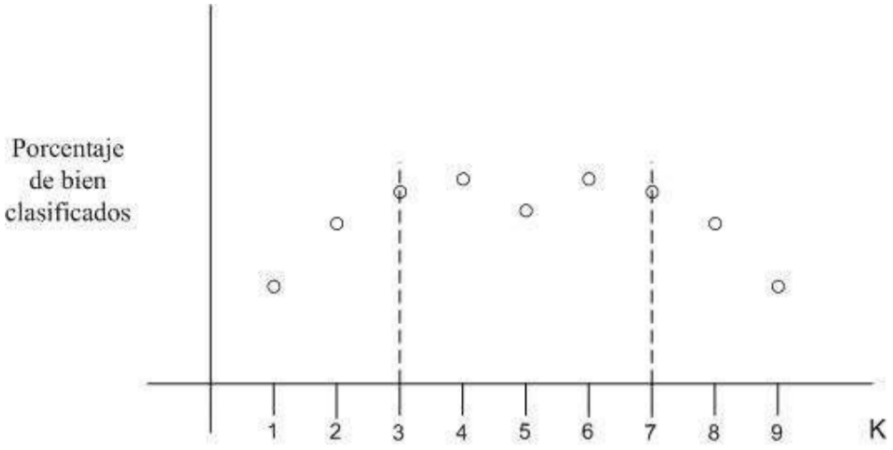
La Figura 20 muestra de manera gráfica un ejemplo de lo anterior. Tal y como puede verse, en esta figura tenemos 24 casos ya clasificados en dos posibles valores $(m = 2)$. Las variables predictoras son X_1 y X_2, y se ha seleccionado $K = 3$. De los 3 casos ya clasificados que se encuentran más cercanos al nuevo caso a clasificar, x (representado por •), dos de ellos pertenecen a la clase ◦, por tanto el clasificador 3-NN predice la clase ◦ para el nuevo caso. Nótese que el caso más cercano a x pertenece a la clase +. Es decir, que si hubiésemos utilizado un clasificador 1-NN, x se hubiese asignado a +.

En caso de que se produzca un empate entre dos o más clases, conviene tener una regla heurística para su ruptura. Ejemplos de reglas heurísticas para la ruptura de empates pueden ser: seleccionar la clase que contenga al vecino más próximo, seleccionar la clase con distancia media menor, etc.

Una cuestión importante es la determinación del valor de K. Se constata empíricamente que el porcentaje de casos bien clasificados es no monótono con respecto de K (véase Figura 21), siendo una buena elección valores de K comprendidos entre 3 y 7.

Figura 21

Ejemplo de la no monotocidad del porcentaje de clasificados en función de K



Fuente: (Moujahid, Inza, & Larrañaga).

2.2.13.8. Naive Bayes

Un clasificador bayesiano obtiene la probabilidad posterior de cada clase, C_i , usando la regla de Bayes, como el producto de la probabilidad a priori de la clase por la probabilidad condicional de los atributos (E) dada la clase, dividido por la probabilidad de los atributos (Sucar L. , 2006).

$$P(C_i|E) = \frac{P(C_i)P(E|C_i)}{P(E)}$$

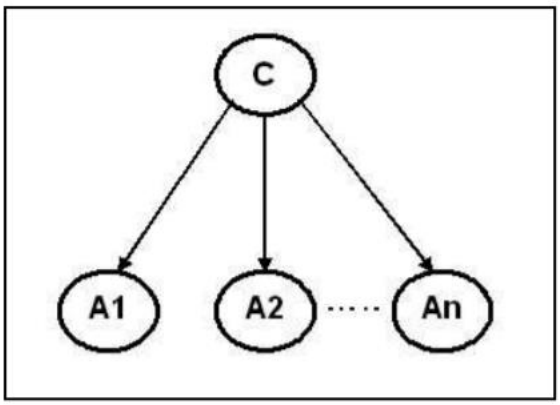
Este clasificador asume que los atributos son independientes entre sí dada la clase, así que la probabilidad se puede obtener por el producto de las probabilidades condicionales individuales de cada atributo dado el nodo clase:

$$P(C_i|E) = \frac{P(C_i)P(E_1|C_i)P(E_2|C_i) \dots P(E_n|C_i)}{P(E)}$$

Donde n es el número de atributos. Gráficamente, un NBC se puede representar como una red bayesiana en forma de estrella, con un nodo de la raíz, C, que corresponde a la variable de la clase, que está conectada con los atributos, $E1, E2, \ldots, E_n$. Los atributos son condicionalmente independientes dada la clase, de tal manera que no existen arcos entre ellos. Los parámetros se pueden estimar fácilmente, a partir de los datos, en base a frecuencias. El denominador en la ecuación no se requiere, ya que es una constante; es decir, no depende de la clase. Al final se pueden simplemente normalizar las probabilidades posteriores de cada clase (haciendo que sumen uno) (Sucar L. , 2006).

Figura 22

Clasificador bayesiano simple, A1,A2,...,An son condicionalmente independientes dada la clase C



Fuente: (Sucar L. , 2006).

2.2.13.9. Gaussian Naive Bayes

Implementa un modelo para la clasificación y supone que la probabilidad de las características es gaussiana.

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} e^{-\frac{(x_i - u_y)^2}{2\sigma_y^2}}$$

2.2.13.10. Multinomial Naive Bayes

Es una de las variantes Bayes ingenuas clásicas utilizadas en la clasificación con características discretas (por ejemplo, recuentos de palabras para la clasificación de texto). Se basa en la aplicación de la Regla de Bayes para predecir la probabilidad condicional de que un documento pertenezca a una clase $P(c_i \mid d_j)$ (Anguiano, 2009).

$$P(c_i|d_j) = \frac{P(c_i)P(d_j|c_i)}{P(d_j)}$$

Dado que la probabilidad de cada documento $P(d_j)$ no aporta información para la clasificación, el término suele omitirse. La probabilidad de un documento dada la clase suele asumirse como la probabilidad conjunta de los términos que aparecen en dichos documentos dada la clase y se calculan como:

$$P(d_j|c_i) = \prod_{t=1}^{V} P(w_t|c_i)$$

El término $P(w_t|c_i)$ se calcula a partir del número de apariciones de cada término w_t en una clase c_i pero para evitar el problema de las probabilidades 0 se usa la estimación de Laplace.

$$P(w_t|c_i) = \frac{1 + n(w_t, c_i)}{V + n(c_i)}$$

Donde $n(w_t, c_i)$ es el número de ocurrencias de w_t en c_i, V es el tamaño del vocabulario y $n(c_i)$ es el conteo total de palabras en c_i. De este modo, la clasificación se hace buscando el argumento que maximiza la función:

$$c^*(d) = \underset{ci}{\operatorname{argmax}}\, P(c_i) \prod_{t=1}^{V} P(w_t|c_i)$$

2.2.13.11. Decision Trees

Los denominados árboles de decisión, constituyen uno de los métodos del aprendizaje inductivo supervisado más utilizados. Una de sus principales virtudes, es la sencillez de los modelos obtenidos. Dado un conjunto de ejemplos de entrenamiento, se construye una partición del espacio de entrada y se asigna a cada región un determinado modelo. Luego, dado un nuevo dato, a partir de los valores de las variables de entrada se determina una región y el predictor del modelo construido le asigna un valor a la variable de salida. Existen muchos métodos de árboles de decisión, como los introducidos por Quinlan, el algoritmo ID3 de 1986 y el C4.5 también de Quinlan del año 1993. Nos encontramos aquí en los llamados Árboles de Clasificación y Regresión, en inglés *Classification and Regression Trees* (CART), que deben su desarrollo a L. Breiman, J. Friedman, R. Olshen y C. Stone, autores del libro del mismo nombre, publicado en 1984 (Roche, 2009).

CART (Classification and Regression Trees)

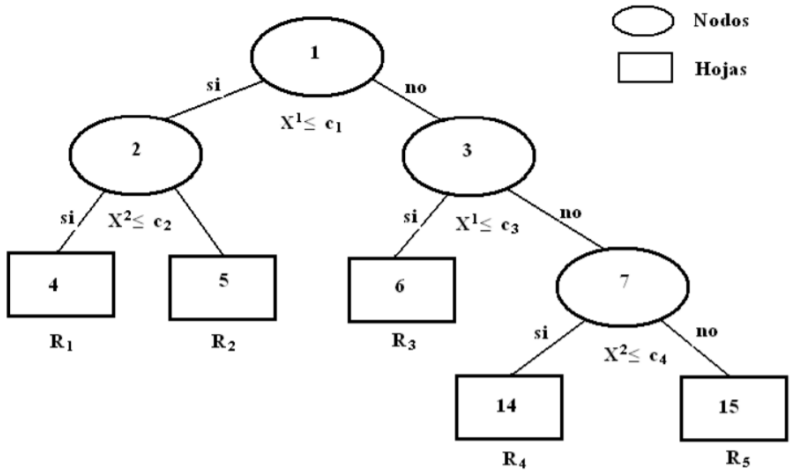
Partimos de una muestra de entrenamiento (1.1). $(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)$, donde cada $X_i = (X_i^1, \ldots, X_i^p)$ es un vector con p variables aleatorias, que pueden ser todas continuas, todas discretas o mezclas de ambas. Las variables Y_i son unidimensionales, discretas o continuas. Con la muestra de entrenamiento, construímos una estructura del tipo árbol en dos etapas bien diferenciadas, en la primera, determinamos el llamado **árbol maximal** y en la segunda, aplicamos un procedimiento denominado de **poda**.

Para construir el árbol, comenzamos con toda la muestra en el **nodo raíz** y vamos obteniendo los **nodos interiores** por particiones sucesivas, mediante una cierta pregunta o regla que involucra a uno de los p atributos. Se trata de árboles binarios, por lo cual en función de la respuesta, cada nodo se parte en dos **nodos hijos**. Por convención, asignamos el nodo izquierdo al caso afirmativo y el derecho al contrario. Por último, se elige algún criterio de parada, para saber cuando un nodo deja de partirse y queda constituyendo un nodo terminal llamado **hoja**.

Supongamos una muestra de entrenamiento como en (1.1), con $p = 2$, y un árbol de 5 hojas como se representa en la Figura 23, donde las particiones se hacen en base a preguntas sobre los atributos X^1 y X^2 y los c_j son números reales. En la Figura 24, se aprecia como el espacio R^2 queda partido en regiones R_i , con $i = 1,\ldots,5$, donde cada R_i es un rectángulo de lados paralelos a los ejes.

Figura 23

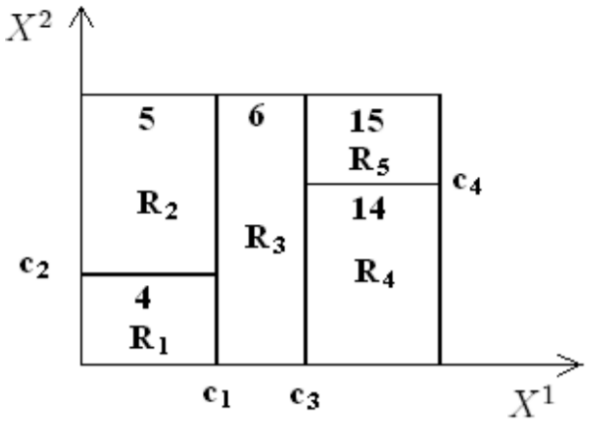
Ejemplo de un árbol de 5 hojas



Fuente: (Roche, 2009).

Figura 24

Partición del espacio R^2, según el árbol de la Figura 23



Fuente: (Roche, 2009).

2.2.14. Python

Python es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. La elegante sintaxis de Python y su tipado dinámico, junto con una naturaleza interpretada, hacen de éste un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en diversas áreas y sobre todo la mayoría de las plataformas (Van Rossum, 2017).

2.2.15. OpenCV

OpenCV (Open Source Computer Vision Library) es una biblioteca de software de visión artificial y aprendizaje automático de código abierto. OpenCV se creó para proporcionar una infraestructura común para aplicaciones de visión por computador y para acelerar el uso de la percepción de máquinas en los productos comerciales.

La biblioteca tiene más de 2500 algoritmos, que incluye algoritmos de machine learning y de visión artificial. Estos algoritmos permiten identificar objetos, caras, clasificar acciones humanas en video, hacer tracking de movimientos de objetos, extraer modelos 3D, encontrar imágenes similares, eliminar ojos rojos, reconocer escenarios, etc. Se usa en aplicaciones como la detección de intrusos en videos, monitorización de equipamientos, ayuda a navegación de robots, inspeccionar etiquetas en productos. Está escrito en C++, tiene interfaces en C++, C, Python, Java y Matlab y funciona en Windows, Linux, Android y Mac OS (Gracia, 2013).

2.2.16. Scikit Learn

Es una librería de python para Machine Learning y Análisis de Datos. Está basada en NumPy, SciPy y Matplotlib. Las ventajas principales de scikit-learn son su facilidad de uso y la gran cantidad de técnicas de aprendizaje automático que implementa. Con scikit-learn podemos realizar aprendizaje supervisado y no supervisado. Podemos usarlo para resolver problemas tanto de clasificación y como de regresión. Es muy fácil de usar porque tiene una interfaz simple y muy consistente. La interfaz es muy fácil de aprender. Te das cuenta que el interfaz es consistente cuando puedes cambiar de técnica de machine learning cambiando sólo una línea de código. Otro punto a favor de

scikit-learn es que los valores de los hiper-parámetros tienen unos valores por defecto adecuados para la mayoría de los casos (Matínez, 2020).

2.2.17. MySQL Workbench

Esta es una herramienta visual de diseño de base de datos que integra desarrollo de software, administración de base de datos, diseño de base de datos, creación y mantenimiento para el sistema de base de datos MySQL, este software da al usuario herramientas de administración complejas para la configuración de servidores, administración de usuarios, copias de seguridad y mucho más. MySQL Workbench está disponible en Windows, Gnu/Linux, y MacOS X (Amoedo, s.f.).

2.2.18. Metodología SEMMA

La metodología SEMMA (Sample, Explore, Modify, Model, Assess) fue propuesta por SAS Institute, el cual la define como el proceso de selección, exploración y modelado aplicado a cantidades significativas de datos almacenados que permitan el descubrimiento de patrones como herramientas de apoyo para el negocio. Según SAS más que una metodología para minería de datos, SEMMA es un conjunto de herramientas funcionales enfocadas hacia los aspectos de desarrollo propio de un modelo de minería y se compone de cinco fases (Hernández & Dueñas, 2009):

Muestreo. En esta etapa se realiza la extracción de una muestra de los datos que permita representar características comunes de la población para posteriormente comenzar el análisis de los mismos. Con esta fase se logra facilitar los procesos de minado sobre los datos, reduciendo costes y tiempo para la organización.

Exploración. La exploración de datos a través de técnicas estadísticas permite realizar un seguimiento a los mismos logrando detectar, identificar y posteriormente eliminar datos que representen anomalías o deficiencias en las fases siguientes hacia el descubrimiento de información.

Modificación. En esta fase se realiza una selección y transformación de los datos de acuerdo con las variables seleccionadas para el proceso de minado, la cual permitirá de acuerdo con éstas adaptar el enfoque de selección y diseño del modelo.

Modelado. En este punto de la metodología se hace uso de herramientas de software que permitan la utilización de técnicas y métodos propios de la minería de datos, las cuales tiendan hacia el descubrimiento de asociaciones o combinaciones entre los datos, logrando así la predicción de resultados con un alto nivel de confianza.

Evaluación. Uno de los pasos principales dentro de una metodología es la valoración de la solución. A partir del modelo obtenido en la fase anterior se realiza una evaluación de resultados para verificar el éxito del proyecto. Una buena práctica para comprobar la validez del modelo es seleccionar otra muestra de datos y aplicarlo para verificación de resultados, si este resulta optimo se procede con el proceso de producción, en caso contrario se desarrollará otro modelo.

Figura 25

Metodología SEMMA

Fuente: (Hernández & Dueñas, 2009).

CAPÍTULO III: MATERIALES Y METODOLOGÍA

3.1. Operacionalización de variables

La presente investigación posee una sola variable, por lo tanto, la operacionalización es la siguiente:

Tabla 5

Operacionalización de variables

VARIABLE	DEFINICIÓN CONCEPTUAL	DIMENSIONES	INDICADORES
Univariado Eficiencia de un sistema de control de calidad mediante procesamiento digital de imágenes y un modelo de machine learning en la clasificación de la tunta.	Un sistema que utiliza algoritmos de procesamiento digital de imágenes las cuales extraen en tiempo real las características específicas de las tuntas mediante una webcam y un modelo de clasificación entrenado con un algoritmo de machine learning de tipo supervisado.	Modelo de Support Vector Machines. Modelo de K-Nearest Neighbors. Modelo de Gaussian Naive Bayes Modelo de Multinomial Naive Bayes. Modelo de Decision Trees. Clasificación	Accuracy (Exactitud) Recall(Recuperación) F1 - Score (Puntuación F1) Matriz de confusión

Fuente: Elaboración propia.

3.2. Población y muestra

3.2.1. Población

La población está conformada por tuntas procesadas por los productores de tunta en la Planta de Producción de Kishuara – Andahuaylas.

3.2.2. Muestra

La muestra fue generada de manera no probabilística, utilizando el muestreo intencional, de tal manera que se tomaron tuntas de las cuatro categorías (primera, segunda, tercera, cuarta), y que estas están determinadas para su clasificación mediante las características del tamaño, forma y color.

Se adquirió un total de 800 unidades de tuntas de las cuatro categorías.

3.3. Nivel de investigación

Es una investigación **descriptiva.**

Hernández, Fernández y Baptista (2016) mencionan que en la investigación descriptiva se busca especificar las propiedades, las características y los perfiles de personas, grupos, comunidades, procesos, objetos o cualquier otro fenómeno que se someta al análisis. Es decir, únicamente pretenden medir o recoger información de manera independiente o conjunta sobre los conceptos o las variables a las que se refieren, esto es, su objetivo no es indicar cómo se relacionan éstas.

3.4. Diseño de investigación

Es una **investigación no experimental** y según Hernández et al. (2016) son estudios que se realizan sin la manipulación deliberada de variables y en los que sólo se observan los fenómenos en su ambiente natural para analizarlos.

3.5. Técnica de instrumentos de acopio de datos

Para el acopio de datos se construyó un corpus con las características y etiquetas de clase de tuntas mediante los siguientes pasos:

1. Obtención de tuntas de diferentes categorías (Primera, Segunda, Tercera, Cuarta) que sean representativos a la categoría que pertenecen.

2. Establecimiento de características específicas, que diferencie entre tuntas de distinta categoría.

3. Extracción de características externas del tamaño, forma y color de cada tunta mediante algoritmos de procesamiento digital de imágenes.

4. Construcción del corpus con la estructura de características extraídas y etiquetas de clase de tuntas.

Tabla 6

Estructura para la construcción del corpus de conocimiento

Columnas de características de tuntas	Clase
Características de tuntas representadas en números.	Primera (0) Segunda (1) Tercera (2) Cuarta (3)

Fuente: Elaboración propia.

- Las etiquetas Primera, Segunda, Tercera, Cuarta se representaron mediante números (0, 1, 2, 3).

- En el vector de características se estableció cada uno de los atributos obtenidos de las imágenes de tuntas.

- Las características **del tamaño** son: Ancho, Altura, Perímetro, Área, Diámetro equivalente.

- La característica **de la forma** es: Relación de aspecto.

- Las características **del color** son las medias de los canales de color RGB, Gris, HSV y HSL del objeto en la imagen.

3.6. Técnicas de análisis de datos

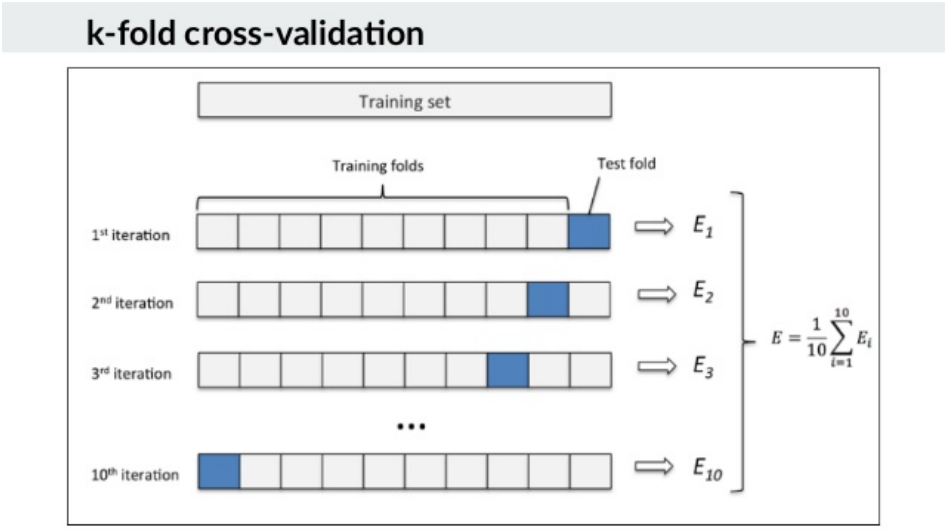
Para el análisis de datos, durante el entrenamiento y generación de los modelos de clasificación se realizó la validación cruzada *k-fold* con 10 iteraciones. Para determinar el rendimiento de los modelos de clasificación, se recurrió a las métricas: Matriz de Confusión, Exactitud (Accuracy), Error Absoluto Medio.

- **Validación cruzada k-fold**

La validación cruzada de k iteraciones o *k-fold cross validation* consiste en tomar los datos de un conjunto y dividirlo en *k subconjuntos* y, al momento de realizar el entrenamiento, se va a tomar cada *k* subconjunto como datos de *prueba* del modelo, mientras que el resto se tomará como conjunto de *entrenamiento*. Este proceso se repetirá *k veces*, y en cada iteración se seleccionará un conjunto de *prueba* diferente. Una vez finalizadas las iteraciones, se calcula la precisión y el error para cada uno de los modelos producidos, y para obtener la precisión y el error final se calcula el promedio de los resultados de los *k* modelos entrenados (Delgado, 2018).

Figura 26

Validación cruzada k-fold



Fuente: (Vásquez, 2017).

- **Matriz de confusión**

Una matriz de confusión resume el rendimiento de clasificación de un clasificador con respecto a algunos datos de prueba. Es una matriz bidimensional, indexada en una dimensión por la clase verdadera de un objeto y la otra por la clase que asigna el clasificador (Sammut & Webb, 2010).

Aplicado en el campo del aprendizaje automático y, específicamente, el problema de la clasificación estadística, la columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real. Uno de los beneficios de las matrices de confusión es que facilita ver si el sistema está confundiendo dos clases.

Para entender mejor este concepto se procede a mostrar cómo sería esta matriz para una clasificación binaria:

Tabla 7

Matriz de confusión para clasificación binaria

		Assigned Class	
		Positive	Negative
Actual Class	Positive	TP	FN
	Negative	FP	TN

Fuente: (Sammut & Webb, 2010).

Como podemos observar, las etiquetas conocidas corresponden a las filas, y las que el algoritmo predice a las columnas. Para definir correctamente estos valores podemos indicar que:

True Positives (TP): Número de casos positivos a los cuales se les asignó la clase positiva.

True Negatives (TN). Número de casos negativos a los cuales se les asignó la clase negativa.

False Positives (FP): Número de casos negativos a los cuales se les asignó la clase positiva.

False Negatives (FN): Número de casos positivos a los cuales se les asignó la clase negativa.

- **Accuracy (Exactitud)**

Este término se aplica en el contexto de los modelos de clasificación. La exactitud de un modelo a menudo se evalúa o estima aplicándolo a los datos de prueba para los que se conocen las etiquetas (valores Y). La exactitud de un clasificador en los datos de prueba se puede calcular como el número de objetos clasificados correctamente/el número total de objetos (Sammut & Webb, 2010).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Recall (Recuperación)**

Es la medida de los casos positivos correctamente identificados de todos los casos positivos reales. Es importante cuando el costo de los falsos negativos es alto (Huilgol, 2019).

$$Recall = \frac{TP}{TP + FN}$$

- **Precision(Precisión)**

Se implica como la medida de los casos positivos correctamente identificados de todos los casos positivos predichos. Por tando, resulta útil cuando los costes de los falsos positivos son elevados (Huilgol, 2019).

$$Precision = \frac{TP}{TP + FP}$$

- **F1 - Score (Puntuación F1)**

El valor F1 se utiliza para combinar las medidas de precisión y recall en un solo valor. Esto es práctico porque hace más fácil el poder comparar el rendimiento combinado de la precisión y la exhaustividad (Martinez, 2020).

$$F1 - Score = 2\,x\,\frac{Precision\,x\,Recall}{Precision + Recall}$$

CAPÍTULO IV: RESULTADOS Y DISCUSIÓN

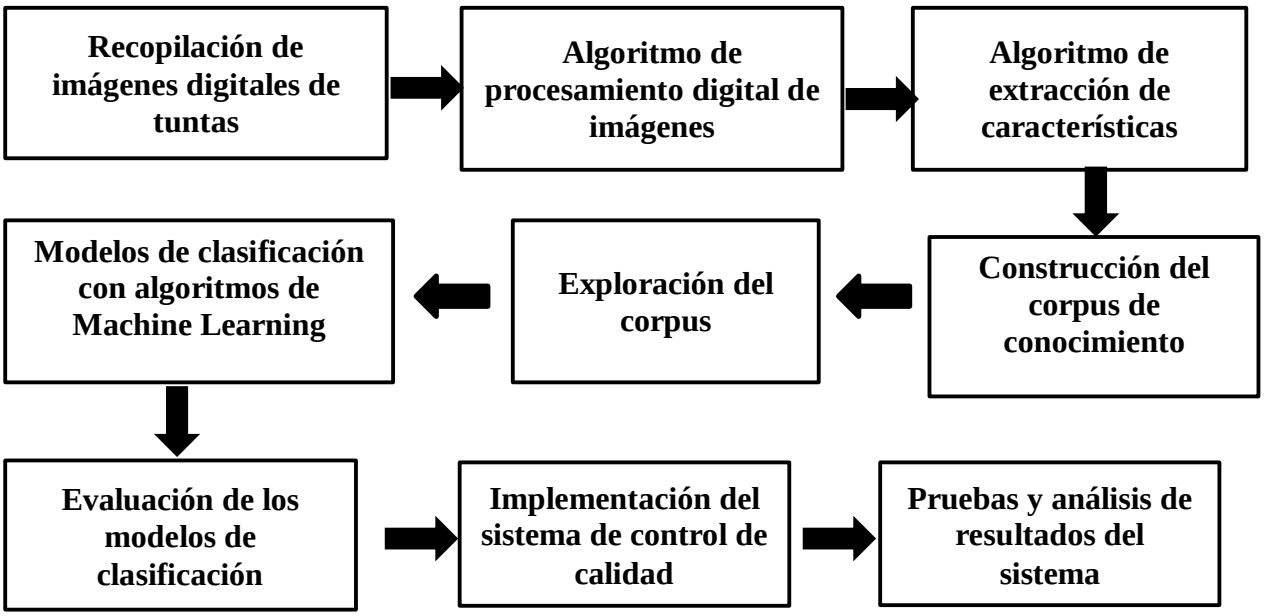
Como el Machine Learning (Aprendizaje Automático) es parte de la minería de datos, el desarrollo de la solución se guió de la metodología SEMMA (Sample, Explore, Modify, Model, Assess) que cuenta con las siguientes fases: Muestreo, Exploración, Modificación, Modelado y Evaluación.

Se partió desde la recopilación de imágenes para la extracción de características mediante procesamiento digital de imágenes, hasta la construcción del sistema y las pruebas.

Diagrama de bloques

Figura 27

Diagrama de bloques del desarrollo de la solución



Fuente: Elaboración propia.

Dentro de la etapa de machine learning que consta desde la Recopilación de imágenes digitales de tuntas hasta la Evaluación de los modelos de clasificación, se planteó la siguiente metodología para el desarrollo de la solución:

Tabla 8

Diagrama de la metodología para el desarrollo de la investigación

Muestreo				Exploración y Modificación	Modelado	Evaluación
Recopilación de imágenes digitales de tuntas	Algoritmo de procesamiento digital de imágenes	Algoritmo de extracción de características	Construcción del corpus de conocimiento	Exploración del corpus	Modelos de clasificación con algoritmos de Machine Learning	Evaluación de los modelos de clasificación

Fuente: Elaboración propia.

4.1. Recopilación de imágenes digitales

Para construir el banco de imágenes, se obtuvo tuntas de distintas categorías (Primera, Segunda, Tercera, Cuarta), con características de tamaño, forma y color distintivas.

Para la captura de imágenes se usó una webcam Microsoft LifeCam Cinema de alta definición con una resolución de 1280 x 720 que permite grabar videos en HD de 720p con 30 fps (fotogramas por segundo).

Figura 28

Webcam Microsoft LifeCam Cinema utilizado en la

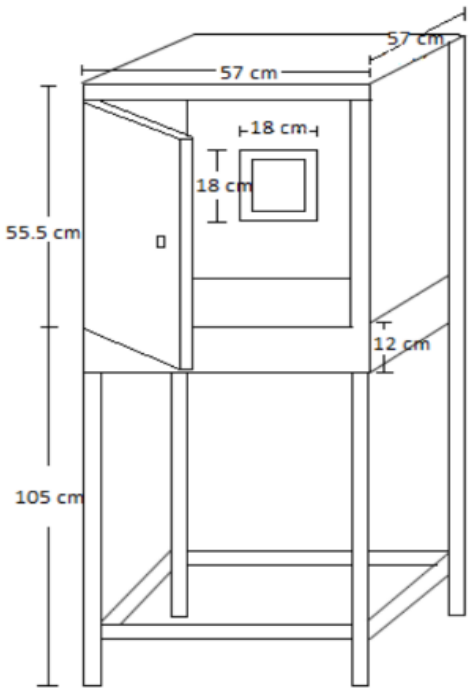
Fuente: Elaboración propia.

4.1.1. Prototipo para la captura de imágenes

Para la captura de imágenes se construyó un prototipo especial para tal fin. El prototipo es una estructura cerrada, con un fondo negro y paredes blancas, y focos fluorescentes en las paredes para ayudar en la iluminación de la escena que es uno de los factores muy importantes en la obtención de imágenes para resaltar e identificar el objeto. Encima del prototipo se colocó una webcam Microsoft LifeCam Cinema a 15 cm de altura, conectada a la Laptop mediante puerto USB.

Figura 29

Medidas del prototipo construido



Fuente: Elaboración propia.

Figura 30

Prototipo para la adquisición de imágenes

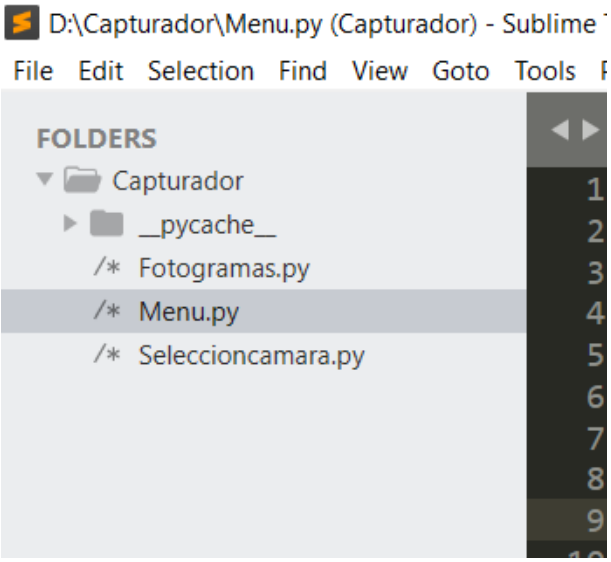
Fuente: Elaboración propia.

4.1.2. Software de captura de imágenes

El software de captura de imágenes se construyó en el lenguaje de programación Python 3.7.1 y la biblioteca OpenCV:

Figura 31

Archivos del capturador



Fuente: Elaboración propia.

- En el archivo Fotogramas.py se encuentra la función para la captura de la secuencia de imágenes tomadas con una cámara, empleando una velocidad de sustitución de imágenes de hasta 30 fps (fotogramas por segundo).

- En el archivo Seleccionacamara.py se encuentra la función que da la opción de elegir la cámara preferencial, siendo una cámara externa (webcam) la indicada para este proceso.

- El archivo Menu.py contiene la estructura de códigos para la construcción de la interfaz gráfica de la aplicación, aquí se encuentran todos los métodos y funciones para la invocación y la ejecución de las diferentes tareas que se encuentran en los diferentes archivos de la aplicación, estas constan de:

    1. Buscar cámara
    2. Iniciar captura
    3. Guardar imagen
    4. Terminar captura

Buscar cámara

En la búsqueda de la cámara, al usuario se le da por defecto 3 índices para la selección. Una vez seleccionada, se guarda ese valor para la iniciación de la captura del video.

Figura 32

Búsqueda y selección de la cámara

```
def __init__(self, root):
    self.ventanaprincipal=root
    self.ventana=tk.Toplevel(self.ventanaprincipal, bg="#1A8EA6", relief="groove")
    self.ventana.title("Seleccione la camara")
    self.ventana.geometry("500x170+80+120")
    self.ventana.overrideredirect(1)
    self.ventana.wm_attributes("-topmost",1)
    self.ventanaprincipal.wm_attributes("-disabled",1)
    self.labelseleccioncamara=tk.Label(self.ventana, text="Seleccione cámara", font=("Arial",14))
    self.labelseleccioncamara.place(x=10, y=10)
    text_font = ("Arial", 13)
    self.combobox=ttk.Combobox(self.ventana, font=text_font)
    self.ventana.option_add("*TCombobox*Listbox*Font", text_font)
    self.combobox["values"]=[0,1,2]
    self.combobox.current(0)
    self.combobox.place(x=250, y=10)
    self.boton=tk.Button(self.ventana, text="Aceptar", width=10, height=1)
    self.boton.place(x=220, y=70)
    self.boton.config(command=self.aceptar, overrelief="raised", bg="#232928", cursor="hand2", font=("Arial",14))

def aceptar(self):
    self.ventanaprincipal.wm_attributes("-disabled",0)
    self.ventana.destroy()

def capturarvalor(self):
    return self.combobox.get()
```

Fuente: Elaboración propia.

Lectura de video

Las secuencias de imágenes son capturadas por la cámara y convertidas al espacio de color RGB y se usa el módulo ImageTk para la visualización del video como una secuencia de imágenes.

Figura 33

Lectura del video

```
class camara:
    camara = None
    frame=None
    image=None
    frame1=None
    image1=None
    frameprocesado=None
    imagenprocesado=None
    def __init__(self):
        pass
    def lectura(self, matrix):
        self.frame=cv2.cvtColor(matrix, cv2.COLOR_BGR2RGB)
        self.image=Image.fromarray(self.frame)
        self.image=ImageTk.PhotoImage(self.image)
        return self.image
```

Fuente: Elaboración propia.

Ventana del video

Figura 34

Ventana del video

```
def Ventanavideo(self, mirror=False):
    video = None
    if self.numcamara!=None:
        self.cap=cv2.VideoCapture(int(self.numcamara))
        self.objVideo=Fotogramas.camara()
        while True:
            ret, FrameMatrix =self.cap.read()
            self.fotocapturada=FrameMatrix

            if mirror is True:
                FrameMatrix=FrameMatrix[:,::-1]
            video=self.objVideo.lectura(FrameMatrix)
            self.labelvideo.configure(image=video)
            self.labelvideo.image=video

            if self.hilo[0]:
                self.hilo[0]=False
                self.labelvideo.configure(image='')
                self.labelvideo.configure(text='PANTALLA PRINCIPAL')
                break
        self.cap.release()
        self.cap=None
    else:
        msg.showinfo("Alerta", "Seleccione la cámara")

def buscarcamara(self):
    camara=sc.Seleccioncamara(self.root)
    self.numcamara=camara.capturarvalor()

def iniciarcaptura(self, hilo):
    hilo1=threading.Thread(target=self.Ventanavideo,args=(hilo,))
    hilo1.start()

def terminarcaptura(self, hilo):
    hilo[0] = True

def guardarimagen(self):
    cv2.imwrite("D:/FOTOS2/tunta"+str(self.i)+".JPG", self.fotocapturada)
    self.i=self.i+1
```

Fuente: Elaboración propia.

Una vez capturada el video y convertida a RGB, se empieza a emitir mediante una ventana para su visualización, se establece también las funciones de buscar cámara, iniciar captura, terminar captura, guardar imagen.

Para guardar una imagen, se estableció la ruta en una unidad del computador para su almacenamiento, y en una extensión, en este caso se consideró el .JPG.

Figura 35

Interfaz para la captura de imágenes

Fuente: Elaboración propia.

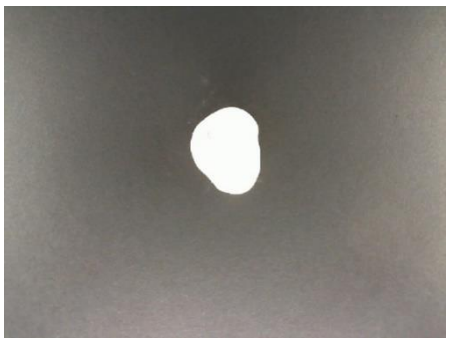
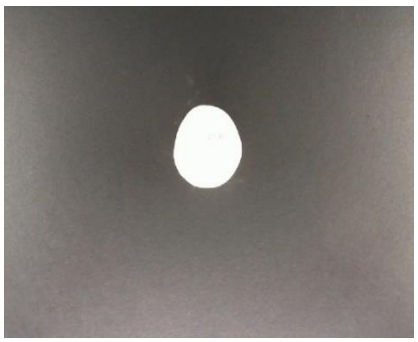
A continuación, se muestran algunas imágenes de tuntas de cada categoría:

a. Tuntas grandes de primera categoría:

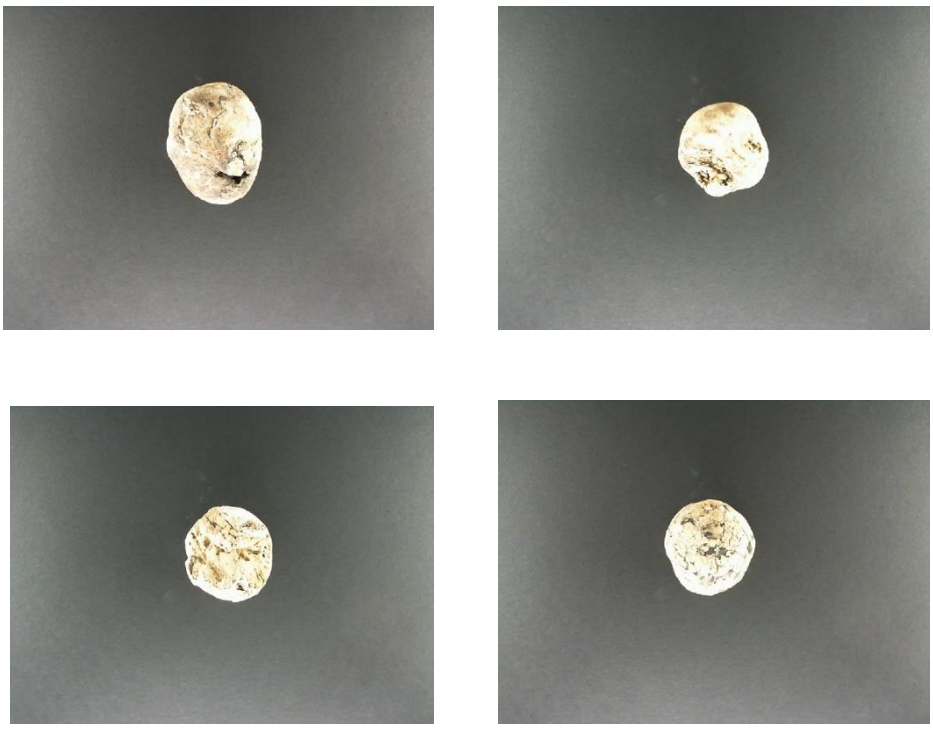

b. Tuntas medianas de segunda categoría.




c. Tuntas pequeñas de tercera categoría:

d. Tuntas oscuras de cuarta categoría.

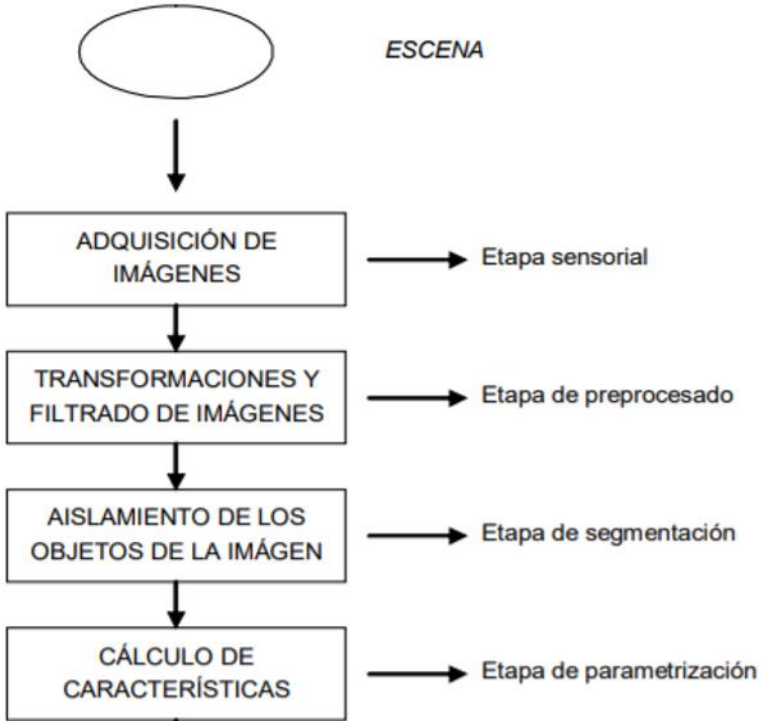


4.2. Procesamiento digital de imágenes

Para la construcción del corpus de datos con las características de tamaño, forma y color, se hizo el análisis de cada una de las 800 imágenes de tuntas destinadas para este proceso, mediante un algoritmo de procesamiento digital de imágenes, con la finalidad de extraer sus características.

Figura 36

Algoritmo para el procesamiento digital de imágenes



Fuente: Elaboración propia.

4.2.1. Adquisición de imágenes

Se tiene una imagen de entrada, el algoritmo lee y lo prepara para los posteriores procesamientos.

Figura 37

Leer una nueva imagen

```
8       def __init__(self,img):
9           img = cv2.imread("D:/BDTUNTA/TUNTAS/tunta0.jpg")
```

Fuente: Elaboración propia.

Figura 38

Imagen de entrada al procesamiento

Fuente: Elaboración propia.

4.2.2. Transformaciones y filtrado de imágenes

a. Conversión de la imagen a escala de grises

Se realiza la conversión de imágenes a escala de grises para que sus pixeles estén en un solo canal de color, una vez realizada esta transformación los valores de cada pixel de una imagen van a estar en un intervalo de 0 a 255.

Figura 39

Conversión de una imagen a escala de grises

```
13          imgray=cv2.cvtColor(imagen,cv2.COLOR_BGR2GRAY)
```

Fuente: Elaboración propia.

Figura 40

Imagen convertida a escala de grises

Fuente: Elaboración propia

b. Difuminación de imágenes mediante filtro gaussiano.

El filtro gaussiano se utilizó para para reducir el ruido en una imagen por la presencia de objetos que no son relevantes en la imagen o cambios bruscos en el color.

Una vez realizada el filtro gaussiano a una imagen, los pixeles que contiene estarán más uniformes, ayudando en el eficaz procesamiento de imágenes.

Figura 41

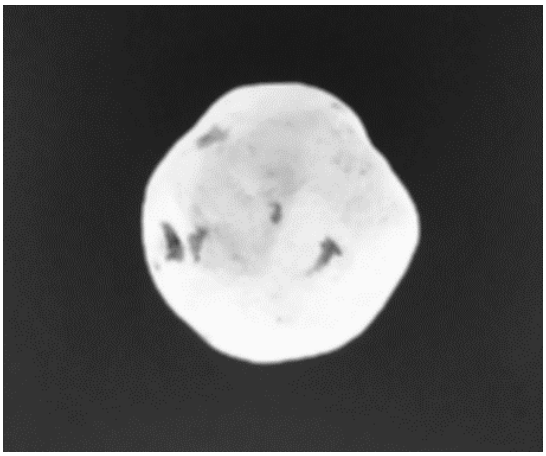
Difuminación con filtro gaussiano

```
14          dif = cv2.GaussianBlur(imgray, (5,5), 5)
```

Fuente: Elaboración propia.

Figura 42

Imagen suavizada con filtro gaussiano



Fuente: Elaboración propia.

c. Umbralización

Los valores de los pixeles de una imagen son convertidos en 0 (blanco) ó 1 (negro) de acuerdo a la intensidad de su escala de grises, permitiendo distinguir con mayor claridad un objeto en una imagen.

Figura 43

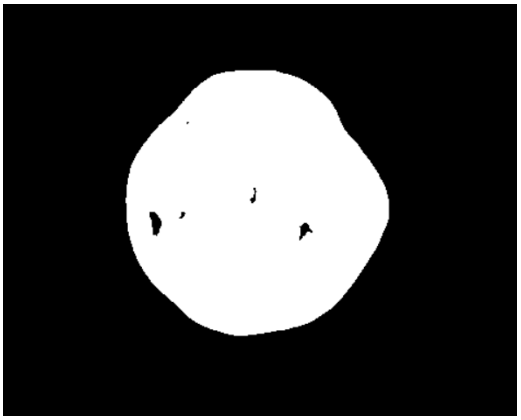
Umbralización de una imagen con algoritmo Otsu

```
15          ret, thresh = cv2.threshold(dif, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)
```

Fuente: Elaboración propia.

Figura 44

Imagen umbralizada con el algoritmo



Fuente: Elaboración propia.

d. Transformaciones morfológicas

Para la correcta segmentación de la imagen se debe tener un fondo y un objeto separados mediante la umbralización, que asigna valores a los píxeles de acuerdo a la intensidad de su escala de grises, valor máximo (color blanco) y valor mínimo (negro), pero en ocasiones cuando el objeto presenta aún manchas negras en su contenido por la presencia de fisuras y huecos y en algunos casos el algoritmo podría confundir con el fondo, también existe la presencia de manchas blancas en el fondo de la imagen y el algoritmo podría confundir con el objeto de la imagen. Para tapar estos huecos se recurren a los métodos de **Cierre** y **Apertura.**

Cierre. Elimina ruido en el objeto de la imagen.
Apertura. Elimina ruido en el fondo de la imagen

Figura 45

Algoritmo de Cierre y Apertura

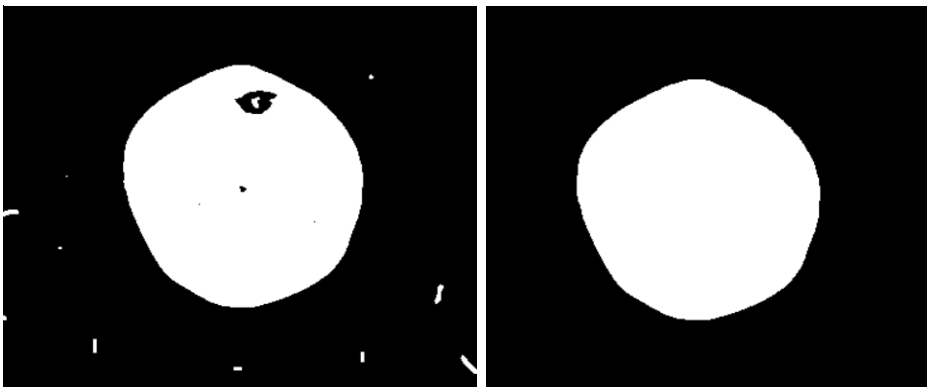
```
17          kernel = np.ones((7,7),np.uint8)
18          closing = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, kernel,iterations = 5)
19          opening = cv2.morphologyEx(closing, cv2.MORPH_OPEN, kernel,iterations = 1)
```

Fuente: Elaboración propia.

En lo siguiente, la primera imagen umbralizada con Otsu. La segunda imagen sin ruidos, resultado de los algoritmos de cierre y apertura:

Figura 46

Imagen umbralizada con Otsu, e imagen sin ruido después de los



Fuente: Elaboración propia.

4.2.3. Segmentación

La segmentación es para obtener el objeto de interés separándolo del fondo de la imagen. La segmentación se realiza mediante una función “Contornos” que devuelve la matriz de contornos del objeto en la imagen y se puede graficar para la visualización del resultado. Con la matriz de contornos se puede calcular las características de la imagen.

Figura 47

Cálculo y gráfica de los contornos del objeto

```
class Procesamiento:
    imagen = None
    def __init__(self,img):
        self.imagen = img
    def Contornos(self, imagen):
        #global cnt
        imgray=cv2.cvtColor(imagen,cv2.COLOR_BGR2GRAY)
        dif = cv2.GaussianBlur(imgray, (5,5), 5)
        ret, thresh = cv2.threshold(dif, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)
        kernel = np.ones((7,7),np.uint8)
        closing = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, kernel,iterations = 5)
        opening = cv2.morphologyEx(closing, cv2.MORPH_OPEN, kernel,iterations = 1)
        contours, hierarchy = cv2.findContours(opening,1,2)
        cv2.drawContours.(self.imagen, contours,-1, (0, 255, 0), 2)
        return contours
```

Fuente: Elaboración propia.

Figura 48

Imagen segmentada y los contornos dibujado alrededor del objeto

Fuente: Elaboración propia.

4.3. Extracción de características

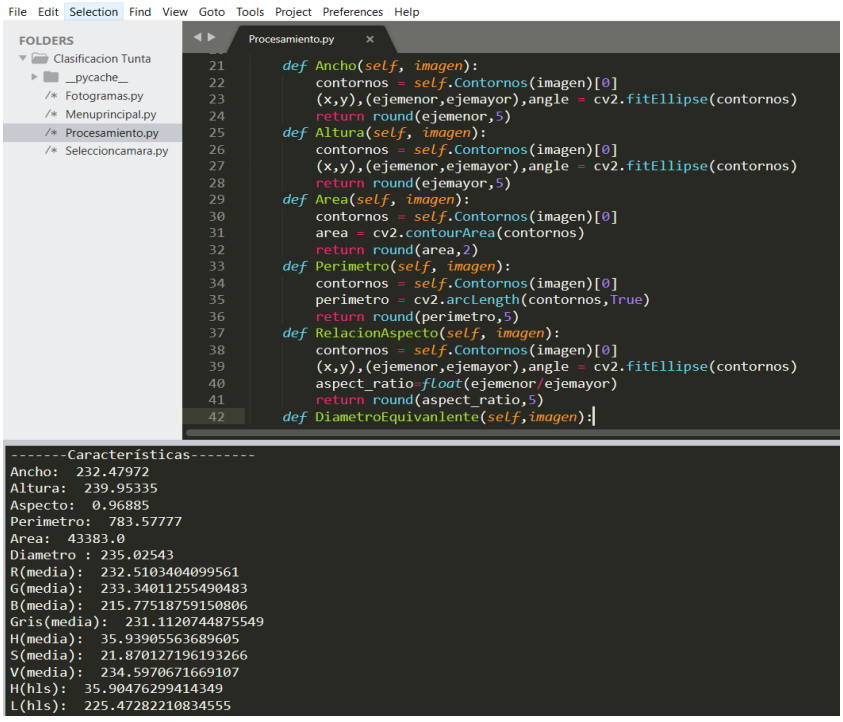
Las características se obtuvieron después de haber obtenido los contornos del objeto en la imagen. Una vez obtenido la matriz de características, se procedió a implementar un algoritmo mediante el uso de dicha matriz, que extrae las dimensiones geométricas de un objeto en una imagen, del mismo modo, para extraer las características del color:

- Las características del tamaño extraídas son: **Ancho, Altura, Perímetro, Área.**
- La característica de la forma es: **Relación de aspecto.**
- Las características del color son:
    - Media de los canales de color RGB del objeto.
    - Media del canal de color Gris del objeto.
    - Media de los canales de color HSV del objeto.
    - Media de los canales de color HSL del objeto.

En la Figura 52 se muestra la codificación y resultados de la extracción de características individuales de tuntas, los resultados se muestran en consola:

Figura 49

Imagen de un fragmento de la extracción de características de una tunta



Fuente: Elaboración propia.

4.4. Construcción del corpus de conocimiento

El corpus de características y etiquetas de clase de las tuntas se construyó con los datos obtenidos de la extracción de características. Individualmente cada tunta se sometió al análisis mediante procesamiento digital de imágenes para la eficaz obtención de características.

Tabla 9

Estructura general del corpus de conocimiento

Columnas de características de tuntas ***Ancho, Altura, Relacion_de_aspecto, Perímetro, Área, Diámetro_equivalente, Canales de color(R(media),G(media),B(media)), Canal de color Gris(media), Canales de color(H(media),S(media),V(media)), Canales de color(H(media),L(media),S(media))***	Clase
Muestras de características de tamaño, forma y color de tuntas representadas en números.	Primera (0) Segunda (1) Tercera (2) Cuarta (3)

Fuente: Elaboración propia

- Las etiquetas Primera, Segunda, Tercera, Cuarta se representaron mediante números (0,1, 2, 3).

- Las características del **tamaño** (Ancho, Altura, Relación de aspecto, Perímetro, Área, Diámetro equivalente) y la **forma** (Relación de aspecto), extraídas después del procesamiento de imágenes están representadas mediante píxeles como unidad de medida.

- Las características del **color** se estructuraron mediante la obtención de la media de los canales de color RGB, Gris, HSV, HLS del objeto en la imagen:

    - Canales de color R(media), G(media), B(media).
    - Canal de color Gris(media).
    - Canales de color H(media), S(media), V(media).
    - Canales de color H(media), L(media), S(media).

En total se extrajo características de 800 imágenes de tuntas clasificadas según su categoría:

Tabla 10

Corpus de conocimiento

CORPUS DE CONOCIMIENTO	
CATEGORÍA	TOTAL
PRIMERA	200
SEGUNDA	200
TERCERA	200
CUARTA	200
TOTAL	**800**

Fuente: Elaboración propia.

A continuación, se visualiza algunas muestras de características extraídas de tuntas de PRIMERA, SEGUNDA y TERCERA clase:

1. **Muestra de las primeras 10 instancias de características de tuntas de Primera clase**

- **Características de tamaño y forma**

Nro	Ancho	Altura	Relación de aspecto	Perímetro	Area	Diámetro equivalente
1	209.20271	229.81787	0.9103	732.28131	37395.5	218.20502
2	239.13063	274.48254	0.87121	864.60721	51745	256.67836
3	238.86713	285.8689	0.83558	871.63664	53118	260.06141
4	232.54617	275.57385	0.84386	850.84985	50459.5	253.46998
5	229.50528	270.47272	0.84853	833.92092	48829	249.34116
6	209.98175	248.64717	0.8445	756.92597	38832	222.35656
7	206.73712	250.41702	0.82557	770.90872	40078.5	225.89717
8	200.31981	241.01994	0.83113	733.35237	37670	219.00441
9	198.30946	247.28738	0.80194	745.69552	38254	220.6955
10	233.18977	239.34779	0.97427	788.52395	43629	235.69083

- **Características de color**

Nro	B(media)	G(media)	R(media)	Gris(media)	H(media)	S(media)	V(media)	H(hls)	L(hls)	S(hls)
1	211.51611	232.01315	233.55353	230.12379	34.20905	27.15315	234.82463	34.20186	223.49124	113.22751
2	217.66147	234.49719	234.57820	232.59819	35.04958	21.55161	236.13539	35.03784	227.19894	113.53681
3	225.48030	237.06273	234.71590	235.02430	40.52325	13.93656	237.50303	40.49535	231.79901	99.38083
4	221.55678	234.03280	234.61414	232.76704	37.62291	17.97001	236.38004	37.60300	229.27261	101.83293
5	220.75805	234.90928	235.70072	233.50366	37.20989	19.09837	237.01803	37.20102	229.24097	102.54345
6	231.19498	241.10336	241.22756	239.97163	40.78370	13.10941	242.66098	40.77908	237.28619	104.40413
7	230.42605	243.01510	243.53046	241.70709	37.32244	15.67596	244.84506	37.31254	237.95986	119.06038
8	236.89508	246.08807	246.87616	245.24845	38.82153	12.00013	248.26946	38.80444	242.89527	121.87821
9	238.32789	245.10590	241.74011	243.28856	49.34412	8.76598	245.50246	49.32281	242.02471	108.54303
10	222.32063	235.54992	234.96322	233.84919	38.35840	17.56520	237.04633	38.33814	229.98956	103.43678

2. Muestra de las primeras 10 instancias de características de tuntas de Segunda clase.

- **Características de tamaño y forma**

Nro	Ancho	Altura	Relacion de aspecto	Perimetro	Area	Diametro equivalente
1	157.40701	180.37489	0.87267	565.12698	22099	167.74183
2	164.13818	183.78979	0.89308	577.38686	23409	172.64201
3	163.72432	186.50731	0.87784	586.25692	24009.5	174.84234
4	151.36292	167.98042	0.90107	527.57063	19708.5	158.40973
5	159.97336	175.07318	0.91375	561.38686	21959	167.20965
6	140.29825	185.79024	0.75514	545.81327	20482	161.48837
7	143.89798	178.96483	0.80406	543.77164	20150.5	160.1762
8	166.86081	173.84224	0.95984	566.78383	22561	169.48616
9	164.25658	173.36665	0.94745	570.70057	22102	167.75321
10	173.56229	179.86452	0.96496	591.77164	24431.5	176.3722

- **Características de color**

Nro	B(media)	G(media)	R(media)	Gris(media)	H(media)	S(media)	V(media)	H(hls)	L(hls)	S(hls)
1	245.84720	251.57890	250.12768	250.45637	45.12092	6.44120	252.08443	45.11716	249.33509	126.25666
2	234.87236	246.82122	247.00689	245.49410	37.01154	14.36402	248.44512	36.99856	241.93158	136.52529
3	248.62217	252.43480	250.74656	251.43962	48.28196	4.20691	252.71243	48.28126	251.10337	112.52283
4	244.43904	251.25068	250.31661	250.15093	44.73422	7.88339	252.11520	44.73161	248.66628	125.42454
5	240.72833	248.20213	247.82246	247.19097	44.96131	10.19466	249.63847	44.95942	245.57569	120.64099
6	248.36153	252.13089	250.62795	251.19621	48.15923	4.41488	252.57744	48.15841	250.89866	112.64161
7	247.39459	251.84335	250.29696	250.82392	47.58995	5.06648	252.26694	47.58907	250.23888	116.10764
8	248.01987	251.89986	250.33971	250.93780	48.04119	4.61349	252.29128	48.04062	250.58770	113.50770
9	241.50839	249.60173	249.24878	248.53215	43.38883	10.36753	251.04313	43.38668	246.64880	133.46476
10	245.78852	251.05594	250.04650	250.10738	44.58927	6.17142	251.77309	44.57874	249.16530	119.27787

3. **Muestra de las primeras 10 instancias de características de tuntas de Tercera clase**

- **Características de tamaño y forma**

Nro	Ancho	Altura	Relacion de aspecto	Perimetro	Area	Diametro equivalente
1	134.96524	149.57655	0.90232	468.37467	15667.5	141.23909
2	135.00676	145.05539	0.93073	466.65894	15414.5	140.09408
3	138.43257	158.61978	0.87273	492.85995	16590.5	145.33988
4	139.93205	155.34641	0.90077	493.44574	16689	145.77069
5	132.25793	137.35905	0.96286	441.50461	13926	133.1583
6	128.68416	137.68826	0.93461	445.44574	13924	133.14874
7	127.68587	138.21327	0.92383	438.71782	13755.5	132.34065
8	128.64384	163.03842	0.78904	487.00209	16372.5	144.38184
9	126.55498	159.96396	0.79115	480.21529	15655	141.18274
10	112.33866	121.82746	0.92211	383.80613	10473.5	115.47846

- **Características de color**

Nro	B(media)	G(media)	R(media)	Gris(media)	H(media)	S(media)	V(media)	H(hls)	L(hls)	S(hls)
1	246.21990	251.45971	249.94014	250.37446	45.82024	6.03774	251.94758	45.81816	249.44521	121.85754
2	248.22590	252.09851	250.40934	251.10402	47.62422	4.32274	252.39736	47.62173	250.70480	115.23916
3	246.24320	250.80149	249.42321	249.82142	45.98298	5.36043	251.24784	45.97816	249.15305	116.44136
4	241.59542	249.16872	248.33755	248.02254	42.18339	9.09128	250.13659	42.17428	246.16446	129.23160
5	244.52960	250.27663	249.19292	249.26204	45.12635	7.22720	251.13343	45.12401	248.19632	124.83718
6	247.15402	251.04378	249.83585	250.18392	47.32710	4.96784	251.73121	47.32554	249.85158	112.47538
7	247.06526	251.80724	250.41427	250.80265	46.67150	5.35138	252.26038	46.66913	250.06210	116.96293
8	245.25932	251.60686	250.20860	250.42407	44.94802	7.03305	252.10710	44.94506	249.05880	126.53908
9	236.27986	246.56792	248.03714	245.80180	38.61105	13.80660	249.40112	38.59465	243.14251	131.57170
10	249.48182	252.45175	250.64568	251.51724	48.72179	3.26788	252.61280	48.72104	251.46951	108.28977

4.5. Exploración del corpus

En la etapa de exploración del corpus se hizo el análisis de datos mediante el uso de las bibliotecas Pandas, Matplotlib y Seaborn de Python:

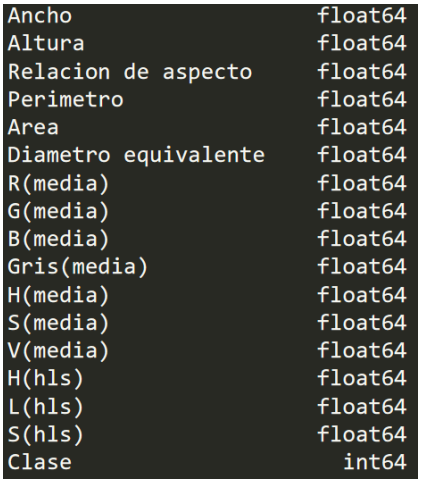
Los pasos que se elaboraron fueron:

1. Tipo de datos.
2. Verificar la cantidad de datos y búsqueda de datos nulos.
3. Correlación de las variables predictoras y la variable objetivo dentro del corpus.

1. Tipo de datos

Figura 50

Verificación de tipo de datos



```
Ancho                   float64
Altura                  float64
Relacion de aspecto     float64
Perimetro               float64
Area                    float64
Diametro equivalente    float64
R(media)                float64
G(media)                float64
B(media)                float64
Gris(media)             float64
H(media)                float64
S(media)                float64
V(media)                float64
H(hls)                  float64
L(hls)                  float64
S(hls)                  float64
Clase                     int64
```

Fuente: Elaboracion propia.

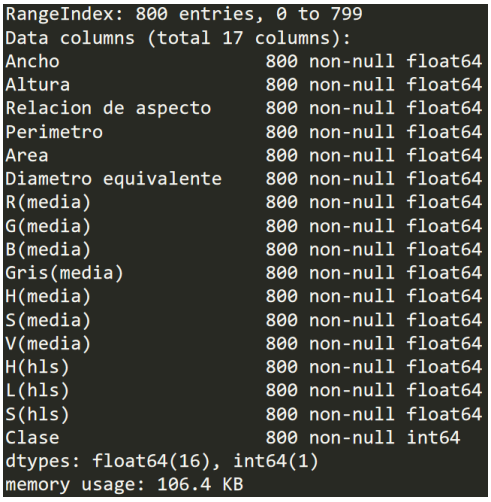
En los resultados verificamos que las variables son numéricas y adecuadas para los siguientes procesos.

La Clase es una variable categórica nominal pero para ser parte de un corpus y generación de un modelo matemático, se procedió a la coversión a una variable cuantitativa de acuerdo a la cantidad de clases que se tiene en el análisis de la clasificación.

2. Verificar la cantidad de datos y búsqueda de datos nulos.

Figura 51

Cantidad de datos y búsqueda de datos nulos



```
RangeIndex: 800 entries, 0 to 799
Data columns (total 17 columns):
Ancho                   800 non-null float64
Altura                  800 non-null float64
Relacion de aspecto     800 non-null float64
Perimetro               800 non-null float64
Area                    800 non-null float64
Diametro equivalente    800 non-null float64
R(media)                800 non-null float64
G(media)                800 non-null float64
B(media)                800 non-null float64
Gris(media)             800 non-null float64
H(media)                800 non-null float64
S(media)                800 non-null float64
V(media)                800 non-null float64
H(hls)                  800 non-null float64
L(hls)                  800 non-null float64
S(hls)                  800 non-null float64
Clase                   800 non-null int64
dtypes: float64(16), int64(1)
memory usage: 106.4 KB
```

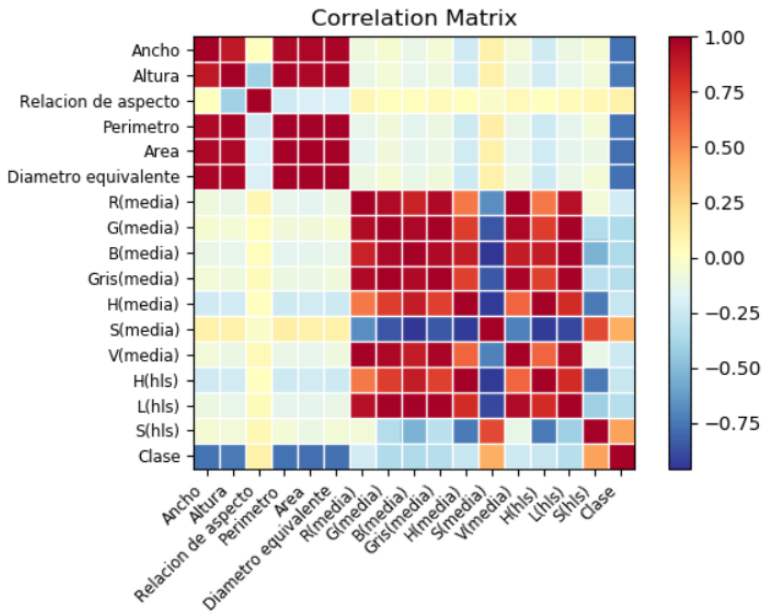
Fuente: Elaboracion propia.

En los resultados verificamos que se tiene en total 17 variables y 800 datos en cada una y no se presenta ningún dato nulo.

3. Correlación entre las variables

Figura 52

Correlación entre variables



Fuente: Elaboración propia.

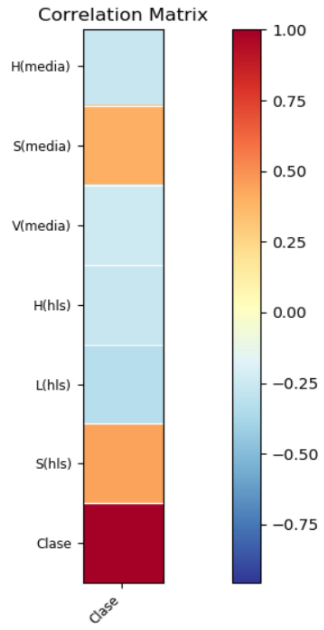
De los resultados se puede desprender lo siguiente:

- Existen correlaciones fuertes y correlaciones débiles entre las variables.

- Lo que sobresale de este análisis es que las variables H(media), S(media), V(media) pertenecientes al espacio de color H,S,V y las variables H(hls), L(hls), S(hls) pertenecientes al espacio de color H,L,S presentan ligera coincidencia en las correlaciones con respecto a la variable objetivo Clase:

Figura 53

Correlación de HSV y HLS con la Clase



Fuente: Elaboración propia.

- Debido a la coincidencia de correlación entre estas variables con la variable objetivo Clase, se mantendrá solo uno de ellos en este caso se considerará solo el espacio de color H, S, V como parte de las variables predictoras que ingresarán al entrenamiento de datos.

4. Correlación de la variable objetivo con las variables predictoras.

Figura 54

Correlaciones de la variable objetivo con las demás variables

	Clase
Clase	1.000000
S(media)	0.400565
Relacion de aspecto	0.111810
R(media)	-0.216206
V(media)	-0.243119
H(media)	-0.263641
Gris(media)	-0.326529
G(media)	-0.351077
B(media)	-0.354897
Altura	-0.737643
Perimetro	-0.760580
Ancho	-0.761830
Diametro equivalente	-0.771894
Area	-0.778283

Fuente: Elaboracion propia.

Solo quedan 14 variables y de los resultados de correlaciones, se desprende lo siguiente:

- Area, Diámetro equivalente, Ancho, Perímetro y Altura están fuertemente correladas con la Clase.

- Las demás variables están correladas ligeramente con la Clase.

4.6. Modelos de clasificación con Machine Learning

Se realizó el entrenamiento del corpus de conocimiento que contiene las muestras de características y etiquetas de clase de las tuntas que fueron sometidas al procesamiento digital de imágenes. Este corpus contiene 800 instancias de características de tuntas de primera, segunda, tercera y cuarta clase, con 14 columnas.

Los algoritmos de machine learning de tipo supervisado que se usaron para el entrenamiento fueron los siguientes: **K-Nearest Neighbors, Decision Trees, Gaussian Naive Bayes, Multinomial Naive Bayes, Support Vector Machines**. Estos algoritmos toman como entrada dos matrices: una matriz "X": de tamaño (#muestras, #características) que contiene las muestras de entrenamiento y una matriz "y" de etiquetas de clase (cadenas o enteros) de tamaño (#muestras).

En el entrenamiento de los modelos dividiremos la base de datos en dos conjuntos de datos, uno para entrenar el modelo (train) y el otro para evaluarlo (test). Para evitar que esta partición pueda influenciar en la evaluación final del modelo utilizaremos la validación cruzada (*cross-validation*) y así, aseguraremos que los resultados son independientes de la partición.

4.6.1. Entrenamiento con K-Nearest Neighbors

La clasificación basada en los vecinos más cercanos requiere el parámetro k que es la cantidad de vecinos cercanos a un punto de consulta. En esta investigación se entrenó variando los k vecinos cercanos entre valores del 1 al 10 con el fin de obtener el mejor modelo.

La clasificación se calcula a partir de un voto mayoritario simple de los vecinos más cercanos de cada punto, a un punto de consulta se le asigna la clase de datos que tiene la mayor cantidad de representantes dentro de sus k vecinos más cercanos.

Como primer paso se realiza la carga del corpus que tiene una extensión .csv con la biblioteca Pandas de Python, se particiona las columnas de características y la columna de etiquetas.

Se carga también el validador k – fold para que el modelo se entrene de acuerdo al parámetro n_splits = 10, con el que le indicamos que se entrene el modelo y se valide 10 veces.

En la Figura 51: La variable classifier carga el algoritmo entrenador, en este caso el n_neighbors = 3, indica tres vecinos cercanos y de esa manera a lo largo de los entrenamientos se variará este valor a partir del 1 al 10.

Figura 55

Entrenamiento con el algoritmo K-Nearest Neighbors

```
30  #Carga de datos
31  dataset = pd.read_csv("D:/CORPUS/bdtuntas.csv")
32  arreglox = dataset[dataset.columns[:-1]].values
33  arregloy = dataset[dataset.columns[-1]].values
34  #Algoritmo entrenador
35  classifier = KNeighborsClassifier(n_neighbors = 3)
36  #Validador K-Fold
37  kf = KFold(n_splits=10, shuffle=True, random_state=2)
38
39  #Entrenamiento
40  for train_index, test_index in kf.split(arreglox):
41      X_train, X_test = arreglox[train_index], arreglox[test_index]
42      y_train, y_test = arregloy[train_index], arregloy[test_index]
43
44      classification_model = classifier.fit(X_train, y_train)
```

Fuente: Elaboración propia.

Después de haberse realizado el entrenamiento, se obtienen las puntuaciones de rendimiento del modelo: Accuracy, Recall, Precision, F1-Score.

Estos resultados se van a obtener en cada una de las iteraciones de la construcción y validación del modelo de clasificación:

Figura 56

Cálculo del rendimiento del modelo

```
    #Rendimiento del modelo
    y_pred=classification_model.predict(X_test)
    ACC = round(accuracy_score(y_test, y_pred),4)
    RECALL = round(recall_score(y_test, y_pred, average='macro'),4)
    PC = round(precision_score(y_test, y_pred, average='macro'),4)
    F1 = round(f1_score(y_test, y_pred, average='macro'),4)
```

Fuente: Elaboración propia.

Del mismo modo se realizó el procedimiento con los demás algoritmos de machine learning:

4.6.2. Entrenamiento con Decision Trees

Al igual que otros clasificadores, toma como entrada dos matrices: una matriz “X”, dispersa o densa, de tamaño (#características, #muestras) que contiene las muestras de entrenamiento, y una matriz “Y” de valores enteros de tamaño (#muestras), que contiene las etiquetas de clase para las muestras de entrenamiento.

Figura 57

Entrenamiento con el algoritmo Decision Trees

```
#Algoritmo entrenador
classifier = tree.DecisionTreeClassifier()
#Validador K-Fold
kf = KFold(n_splits=10, shuffle=True, random_state=2)

#Entrenamiento
for train_index, test_index in kf.split(arreglox):
    X_train, X_test = arreglox[train_index], arreglox[test_index]
    y_train, y_test = arregloy[train_index], arregloy[test_index]

    classification_model = classifier.fit(X_train, y_train)
```

Fuente: Elaboración propia.

4.6.3. Entrenamiento con Gaussian Naive Bayes

Figura 58

Entrenamiento con el algoritmo Gaussian Naive Bayes

```
#Algoritmo entrenador
classifier = GaussianNB()
#Validador K-Fold
kf = KFold(n_splits=10, shuffle=True, random_state=2)

#Entrenamiento
for train_index, test_index in kf.split(arreglox):
    X_train, X_test = arreglox[train_index], arreglox[test_index]
    y_train, y_test = arregloy[train_index], arregloy[test_index]

    classification_model = classifier.fit(X_train, y_train)
```

Fuente: Elaboración propia.

4.6.4. Entrenamiento con Multinomial Naive Bayes

Figura 59

Entrenamiento con el algoritmo Multinomial Naive Bayes

```
#Algoritmo entrenador
classifier = MultinomialNB()
#Validador K-Fold
kf = KFold(n_splits=10, shuffle=True, random_state=2)

#Entrenamiento
for train_index, test_index in kf.split(arreglox):
    X_train, X_test = arreglox[train_index], arreglox[test_index]
    y_train, y_test = arregloy[train_index], arregloy[test_index]

    classification_model = classifier.fit(X_train, y_train)
```

Fuente: Elaboración propia.

4.6.5. Entrenamiento con Support Vector Machines

En el entrenamiento con este algoritmo se debe especificar el tipo de clasificador SVM.

LinearSVC una implementación de la clasificación de vectores de soporte para el caso de un kernel lineal. Implementa una estrategia de clases múltiples “uno contra el resto”, entrenando así modelos de n clases, este método adminte entradas densas y dispersas.

Por otro lado, SVC y NuSVC implementan el enfoque “uno contra uno” y “uno contra el resto” esta última viene por defecto.

A continuación, se realizan los entrenamientos con cada uno de los métodos y al final se tomará la mejor, ya que todas están destinadas a clasificaciones multiclase acorde con la situación que se tiene.

Figura 60

Entrenamiento con el algoritmo Support Vector Machines

```
#Algoritmo entrenador
classifier = svm.LinearSVC(dual = False)
#Validador K-Fold
kf = KFold(n_splits=10, shuffle=True, random_state=2)

#Entrenamiento
for train_index, test_index in kf.split(arreglox):
    X_train, X_test = arreglox[train_index], arreglox[test_index]
    y_train, y_test = arregloy[train_index], arregloy[test_index]

    classification_model = classifier.fit(X_train, y_train)
```

Fuente: Elaboración propia.

Se realizó también la prueba con el método SVC:

```
classifier = svm.SVC()
```

Y finalmente se realizó la prueba con el método NuSVC:

```
classifier = svm.NuSVC()
```

Se guardó el modelo de clasificación con una extensión. pkl:

Figura 61

Guardado del modelo de clasificación

```
#Guardar modelo entrenado
dump(classification_model, 'modelo_entrenado.pkl')
```

Fuente: Elaboración propia.

4.7. Evaluación de los modelos de clasificación

La técnica de estimación de rendimiento a ser usada en la presente investigación es la validación cruzada, la cual es una técnica que divide el conjunto de datos de manera aleatoria en k partes exclusivas de aproximadamente el mismo tamaño. Es decir, siendo nuestro conjunto de datos D, la validación cruzada divide este en k partes de la siguiente manera: D1, D2, D3,…, Dk. El modelo es construido y evaluado k veces, siendo construido usando D - Di instancias y evaluado en las Di instancias.

Se realizó 5 pruebas mediante la validación cruzada con k = 10, es decir el modelo se entrena y evalúa 10 veces con conjuntos de prueba diferentes al conjunto de entrenamiento.

Los resultados se plasman mediante las métricas: Accuracy, Recall, Precision, F1-Score de cada modelo de clasificación, pero daremos mayor énfasis en la Accuracy, Recall y F1-Score.

En las siguientes tablas se muestran los resultados de las pruebas de los modelos generados con los algoritmos K-Nearest Neighbors, Decision Trees, Gaussian Naive Bayes, Multinomial Naive Bayes, Support Vector Machines:

4.7.1. Prueba del modelo K-Nearest Neighbors

A continuación, se presentan los resultados de rendimiento obtenidos por los modelos construidos a partir del algoritmo K-NN, las pruebas se realizaron variando los vecinos cercanos a partir del 1 al 10 con el fin de obtener la mejor puntuación:

Tabla 11

Resultados con K-Nearest Neighbors

K vecinos	Accuracy	Recall	Precision	F1-Score
1	0.8088	0.8126	0.8097	0.8055
2	0.7775	0.7820	0.7818	0.7594
3	0.7763	0.7755	0.7726	0.7697
4	0.7563	0.7575	0.7497	0.7376
5	0.7600	0.7583	0.7568	0.7527
6	0.7488	0.7485	0.7372	0.7322
7	0.7413	0.7372	0.7372	0.7316
8	0.7450	0.7454	0.7373	0.7305
9	0.7438	0.7414	0.7395	0.7341
10	0.7475	0.7464	0.7371	0.7313

Fuente: Elaboración propia.

Al analizar los resultados, se obtiene un mejor rendimiento al entrenar con 1 vecino cercano a diferencia del resto de los modelos, se obtiene un Accuracy del 80.88%, Recall del 81.26% y un F1-Score del 80.55%.

De la misma forma se obtuvieron los resultados de los otros modelos de clasificación.

4.7.2. Prueba del modelo Decision Trees

Tabla 12

Resultados con Decision Trees

Modelo	Accuracy	Recall	Precision	F1-Score
Decision Trees	0.9288	0.9311	0.9314	0.9293

Fuente: Elaboración propia.

4.7.3. Prueba del modelo Gaussian Naive Bayes

Tabla 13

Resultados con Gaussian Naive Bayes

Modelo	Accuracy	Recall	Precision	F1-Score
Gaussian Naive Bayes	0.8938	0.8976	0.9033	0.8943

Fuente: Elaboración propia.

4.7.4. Prueba del modelo Multinomial Naive Bayes

Tabla 14

Resultados con Multinomial Naive Bayes

Modelo	Accuracy	Recall	Precision	F1-Score
Multinomial Naive Bayes	0.8063	0.8069	0.8120	0.7982

Fuente: Elaboración propia.

4.7.5. Prueba del modelo Support Vector Machines

El entrenamiento de este modelo se realizó con los métodos LinearSVC, SVC y NuSVC.

Tabla 15

Resultados con Support Vector Machines

Modelo	Accuracy	Recall	Precision	F1-Score
LinearSVC	0.9313	0.9348	0.9341	0.9318
SVC	0.7588	0.7586	0.7529	0.7441
NuSVC	0.7900	0.7883	0.7948	0.7818

Fuente: Elaboración propia.

En los modelos de clasificación generados con el algoritmo Support Vector Machines se obtiene buenos resultados en la clasificación con el método LinearSVC, con un Accuracy del 93.13%, Recall del 93.48% y un F1-Score del 93.18%.

Se puede ver que este modelo es el de mejor rendimiento en comparación con el resto de los modelos de clasificación.

4.7.6. Resumen de las pruebas de los modelos

Tabla 16

Resumen de las pruebas

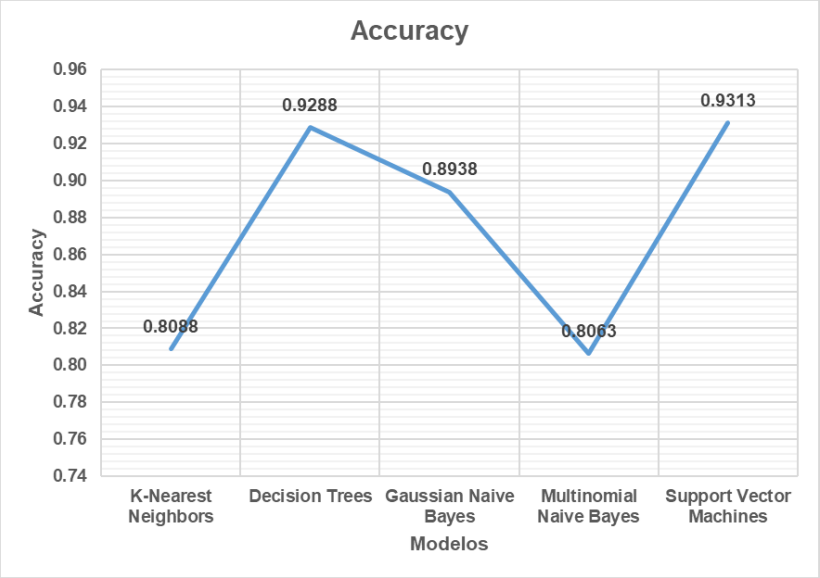
	MODELOS DE CLASIFICACIÓN				
Métricas	**K-Nearest Neighbors**	**Decision Trees**	**Gaussian Naive Bayes**	**Multinomial Naive Bayes**	**Support Vector Machines**
Accuracy	0.8088	0.9288	0.8938	0.8063	0.9313
Recall	0.8126	0.9311	0.8976	0.8069	0.9348
Precision	0.8097	0.9314	0.9033	0.8120	0.9341
F1-Score	0.8055	0.9293	0.8943	0.7982	0.9318

Fuente: Elaboración propia.

Gráfico de Accuracies

Figura 62

Gráfico de Accuracies

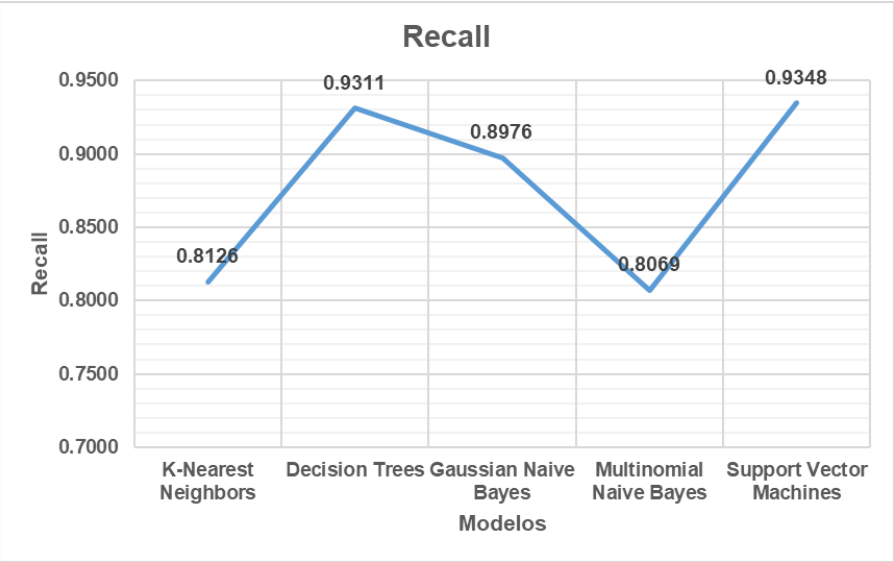


Fuente: Elaboración propia.

Gráfico de Recalls

Figura 63

Gráfico de Recalls

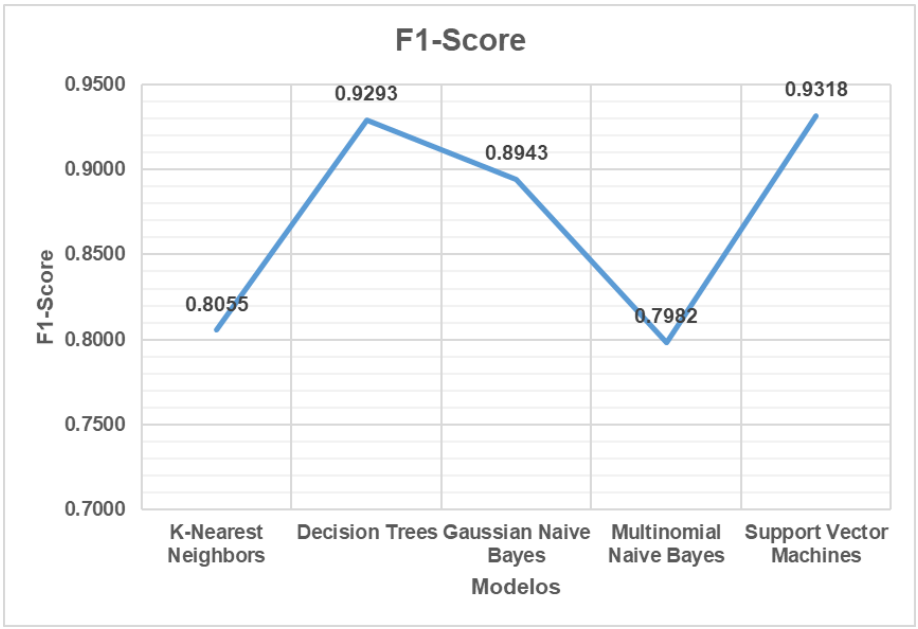


Fuente: Elaboración propia.

Gráfico de F1 - Scores

Figura 64

Gráfico de F1 - Scores



Fuente: Elaboración propia.

Después de la evaluación de los modelos mediante las métricas de clasificación, se observa que el modelo generado con el algoritmo Support Vector Machines obtiene mejores resultados en la clasificación de tuntas de las cuatro clases (primera, segunda, tercera, cuarta), con un Accuracy del 93.13%, Recall del 93.48% y un F1-Score del 93.18% y como segundo mejor modelo es el generado con el algoritmo Decision trees.

Por la obtención de mejores resultados con el modelo Support Vector Machines, este se implantó en el sistema clasificador de tuntas.

4.8. Análisis de la evaluación de los modelos

Se utilizó 800 unidades de tuntas de las cuatro categorías para la obtención del banco de imágenes y estas fueron analizadas individualmente mediante un algoritmo de procesamiento digital de imágenes para la extracción de características de tamaño, forma y color de las tuntas, con estos datos se construyó el corpus de conocimiento. El corpus se entrenó con algoritmos de machine learning de tipo supervisado (K-Nearest Neighbors, Decision Trees, Gaussian Naive Bayes, Multinomial Naive Bayes, Support Vector Machines), y se generaron modelos de clasificación.

Se realizó la prueba del rendimiento de los modelos de clasificación mediante la validación cruzada, los resultados de estas pruebas se plasmaron mediante las métricas de clasificación: Accuracy, Recall y F1-Score:

1. En la primera prueba se tiene los resultados de los modelos generados con el algoritmo K-Nearest Neighbors, probado con vecinos cercanos que oscilaban entre 1 y 10 y determinado con un vecino cercano como el más óptimo, obteniéndose un Accuracy del 80.88%, Recall del 81.26% y un F1-Score del 80.55%.

2. En la segunda prueba se tiene los resultados del modelo con el algoritmo Decision Trees, obteniéndose un Accuracy del 92.88%, Recall del 93.11% y un F1-Score del 92.93%.

3. En la tercera prueba se tiene los resultados del modelo con el algoritmo Gaussian Naive Bayes, obteniéndose un Accuracy del 89.38%, Recall del 89.76% y un F1-Score del 89.43%.

4. En la cuarta prueba se tiene los resultados del modelo con el algoritmo Multinomial Naive Bayes, obteniéndose un Accuracy del 80.63%, Recall del 80.69% y un F1-Score del 79.82%.

5. En la quinta prueba se tiene los resultados de los modelos generados con el algoritmo Support Vector Machines, probado con los métodos LinearSVC, SVC y NuSVC y determinado el método LinearSVC como el más óptimo, obteniéndose un Accuracy del 93.13%, Recall del 93.48% y un F1-Score del 93.18%.

Después del análisis de la evaluación de los modelos, se determina que el modelo generado con el algoritmo Support Vector Machines es el más eficiente en la clasificación de tuntas y el modelo generado con el algoritmo Multinomial Naive Bayes es el menos eficiente.

4.9. Implementación de la interfaz

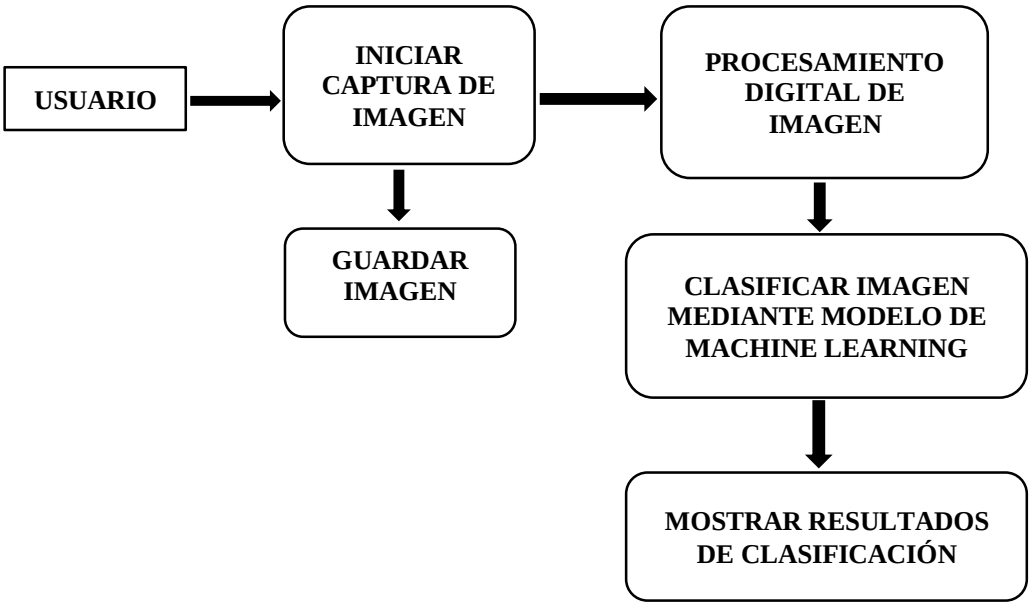
Después de haber obtenido un modelo de clasificación con el algoritmo Support Vector Machines y determinado como el más eficiente entre el resto de los modelos, se realizó la implementación del sistema clasificador de tuntas.

1. Diagramas

Mediante el **diagrama de contexto** se planteó el esquema general del funcionamiento del sistema, para el detalle de algunos procesos se implementaron los **diagramas de flujo**.

Figura 65

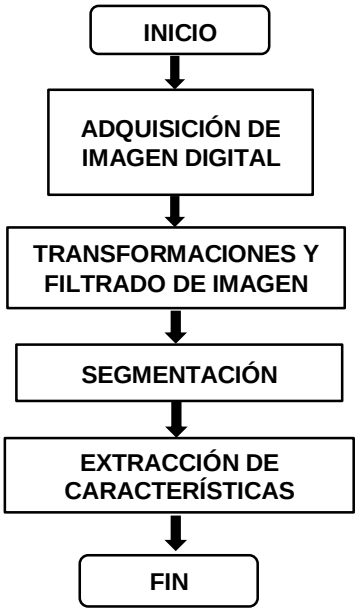
Diagrama de contexto del sistema



Fuente: Elaboración propia.

Figura 66

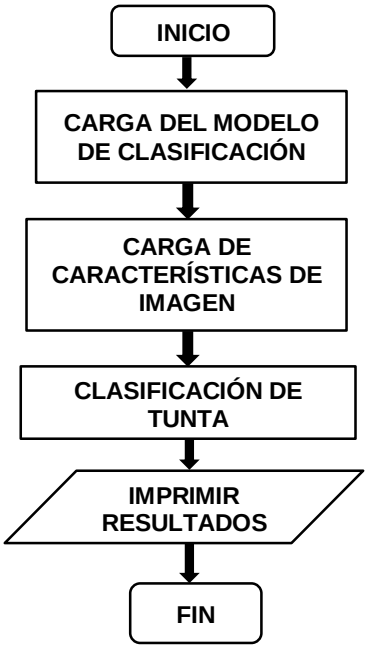
Diagrama de flujo del procesamiento digital de imágenes



Fuente: Elaboración propia.

Figura 67

Diagrama de flujo de la clasificación mediante modelo de machine

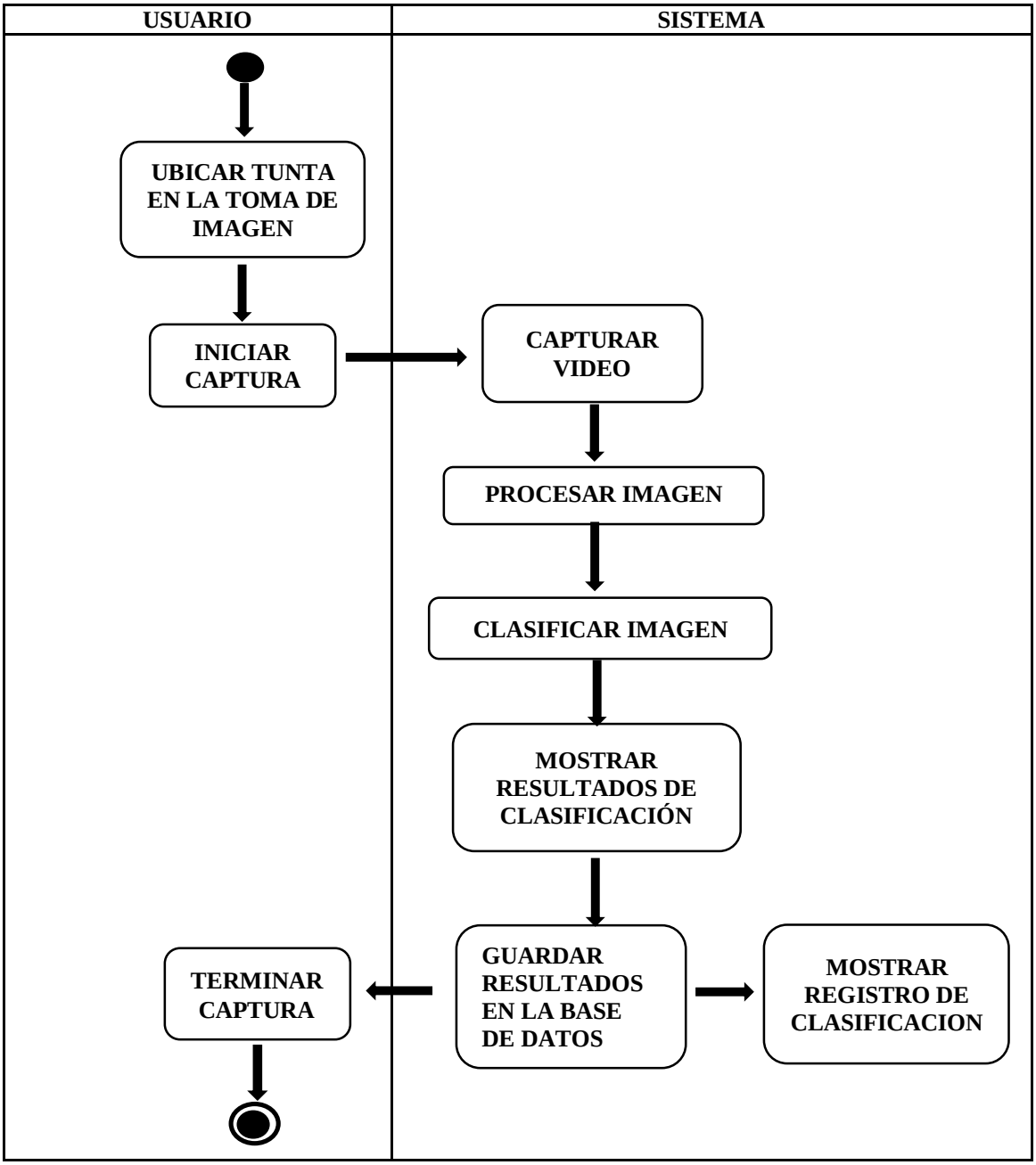


Fuente: Elaboración propia.

Mediante el **diagrama de canal** que es una variación del diagrama de actividades, permite visualizar los procesos del sistema, divididos mediante canales de separación de procesos, y las acciones del usuario que interviene en el funcionamiento del sistema.

Figura 68

Diagrama de canal del sistema



Fuente: Elaboración propia.

2. Construcción

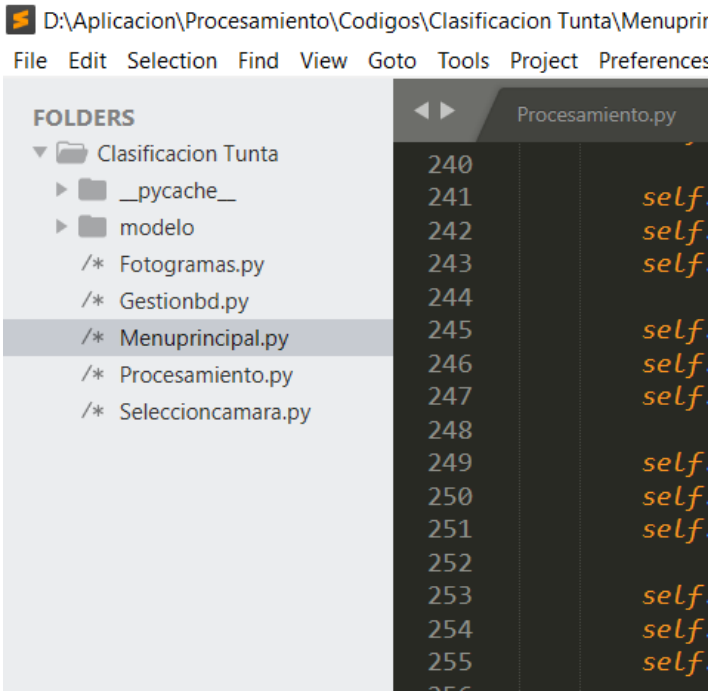
Para la implementación del sistema se usó el lenguaje de programación Python en su versión 3.7.4 con el editor de texto Sublime Text 3.

En la implementación de la interfaz de usuario se usó la biblioteca gráfica Tkinter de Python.

La estructura principal de archivos de la aplicación es el siguiente:

Figura 69

Estructura principal de archivos de la aplicación

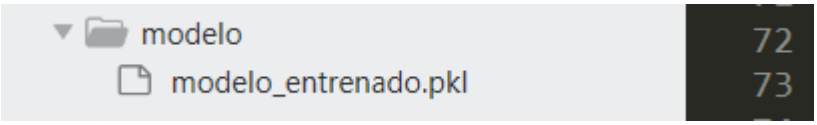


Fuente: Elaboración propia

- En la carpeta **modelo** se encuentra el archivo del modelo de clasificación entrenado con el corpus de conocimiento.

Figura 70

Archivo del modelo de clasificación



Fuente: Elaboración propia.

- En el archivo Fotogramas.py se encuentra la función para la captura de la secuencia de imágenes tomadas con una cámara, empleando una velocidad de sustitución de imágenes de hasta 30 fps (fotogramas por segundo).

- En el archivo Gestionbd.py se encuentra los códigos fuente de la estructura de la gestión del registro de clasificación de tuntas, las funciones para la conexión a la base datos, muestra de datos, búsqueda de datos, eliminar datos, etc.

- En el archivo Seleccionacamara.py se encuentra la función que da la opción de elegir la cámara preferencial, siendo una cámara externa (webcam) la indicada para esta investigación.

- El archivo Procesamiento.py contiene el cerebro de la aplicación por así decirlo, donde se encuentra el algoritmo implementado para el procesamiento digital de imágenes, extracción de características y la carga del modelo de clasificación entrenado.

- El archivo Menuprincipal.py contiene la estructura de códigos para la construcción de la interfaz gráfica de la aplicación, aquí se encuentran todos los métodos y funciones para la invocación y la ejecución de las diferentes tareas que se encuentran en los diferentes archivos de la aplicación mencionadas, estas constan de:

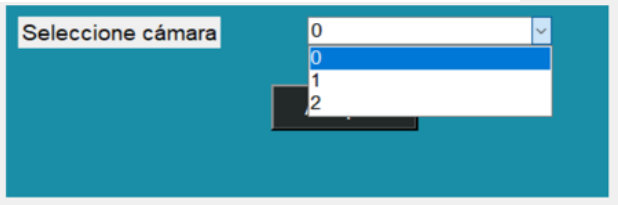
    1. Buscar cámara
    2. Iniciar captura
    3. Procesar (clasificación)
    4. Guardar captura
    5. Terminar captura
    6. Guardar clasificación
    7. Ver registro

Como ya se construyó el software de captura de imágenes como se detalla en el subcapítulo 4.1., esta se utilizó para los módulos de buscar cámara e iniciar captura, a partir de allí se construyeron los demás módulos del sistema clasificador de tuntas.

1. **Buscar cámara.** En la búsqueda de cámara se da la opción al usuario de usar una cámara externa al computador, la más adecuada para el sistema es una webcam.

Figura 71

Buscar cámara

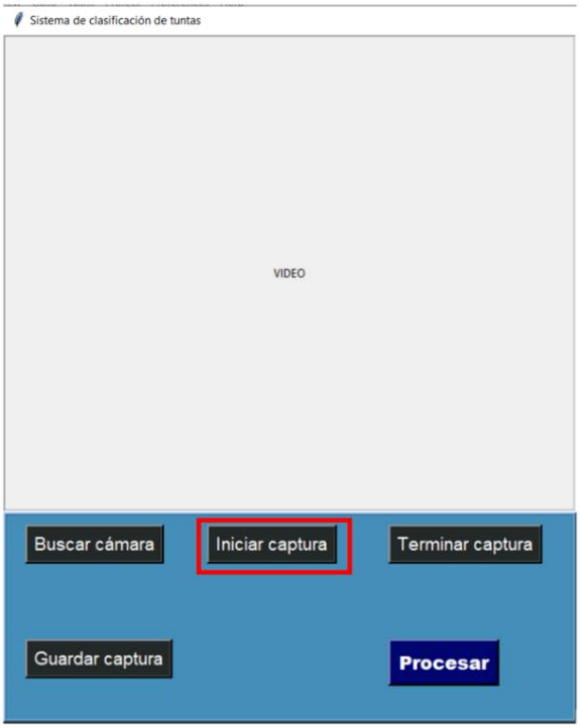


Fuente: Elaboración propia.

2. **Iniciar captura.** El usuario inicia su cámara elegida y el sistema empieza a detectar el objetivo.

Figura 72

Interfaz del inicio de captura de video



Fuente: Elaboración propia.

3. Procesar

Este procedimiento está constituido por dos subprocesos principales:

3.1. Procesamiento digital de imagen

En este proceso la imagen de la tunta capturada mediante la webcam es analizada en tiempo real mediante algoritmos de procesamiento digital de imágenes y se extraen las características geométricas y características del color. En este proceso fueron usados los algoritmos construidos y mostrados en el subcapítulo 4.2.

3.2. Clasificación

Las características de la imagen procesada son los datos de entrada para el modelo de clasificación generado mediante entrenamiento del corpus de datos con un algoritmo de machine learning. Según al resultado de la predicción del modelo se establece la categoría a la que pertenece una nueva tunta.

El modelo entrenado se encuentra en un archivo con extensión .pkl, se procedió a la carga del modelo y alimentación con las características extraídas de la tunta.

Figura 73

Predicción de la clase de una tunta

```
80  def prediccion(self):
81      modelo = joblib.load('modelo/modelo_entrenado.pkl')
82      clase = modelo.predict([[self.Ancho(self.imagen),self.Altura(self.imagen),
83          self.RelacionAspecto(self.imagen),self.Perimetro(self.imagen),self.Area(self.imagen),
84          self.DiametroEquivanlente(self.imagen),self.ColormeanRGB(self.imagen)[0],
85          self.ColormeanRGB(self.imagen)[1],self.ColormeanRGB(self.imagen)[2],
86          self.ColormeanGris(self.imagen)[0],self.ColormeanHSV(self.imagen)[0],
87          self.ColormeanHSV(self.imagen)[1],self.ColormeanHSV(self.imagen)[2]]])
88
89      return clase
```

Fuente: Elaboración propia.

Según al resultado de la predicción se estableció la categoría a la que pertenece una nueva tunta (Primera, Segunda, Tercera, Cuarta).

4. **Guardar captura.**

El usuario tiene la opción de guardar la captura actual en una unidad del computador.

5. **Guardar clasificación**

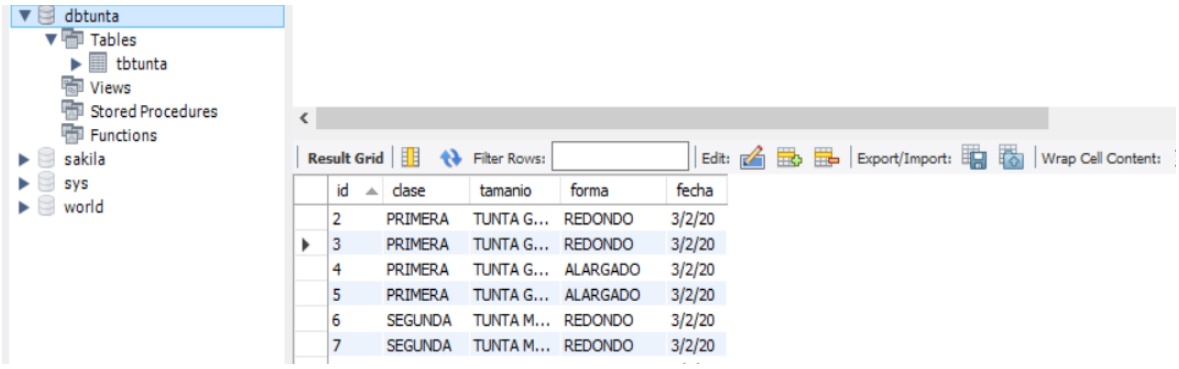
Después de realizar el proceso de clasificación de una tunta, existe la opción de guardar dichos resultados en una base de datos, para su posterior filtrado y búsqueda de datos.

Al guardar los resultados también se guarda automáticamente la fecha actual en la que se está realizando la clasificación.

Se implementó una base de datos con la herramienta de diseño Mysql Workbench para el gestor MySql.

Figura 74

Base de datos de resultados



Fuente: Elaboración propia.

Figura 75

Guardado de los resultados del procesamiento

```
def guardaresults(self):

    day = str(date.today().day)
    mes = str(date.today().month)
    anio = date.today().year
    anio = anio%(10**2)
    anio = str(anio)
    fecha1 = str(mes+"/"+day+"/"+anio)

    self.id=tkinter.StringVar()
    self.clase=tkinter.StringVar(value = self.etiquetaclase)
    self.tamanio=tkinter.StringVar(value = self.etiquetamanio)
    self.forma=tkinter.StringVar(value = self.etiquetaforma)
    self.fecha=tkinter.StringVar(value = fecha1)

    con = pymysql.connect(host="localhost", user="root", password="1234",database="dbtunta")
    cur = con.cursor()

    if self.clase.get() =="":
        messagebox.showerror("Mensaje", "Procece la tunta")
    else:
        cur.execute("insert into tbtunta values (NULL,%s,%s,%s,%s)", (self.clase.get(), self.tamanio.get(), self.forma.get(), self.fecha.get()))
        con.commit()
        con.close()
        messagebox.showinfo("Mensaje", "Insertado correctamente")
```

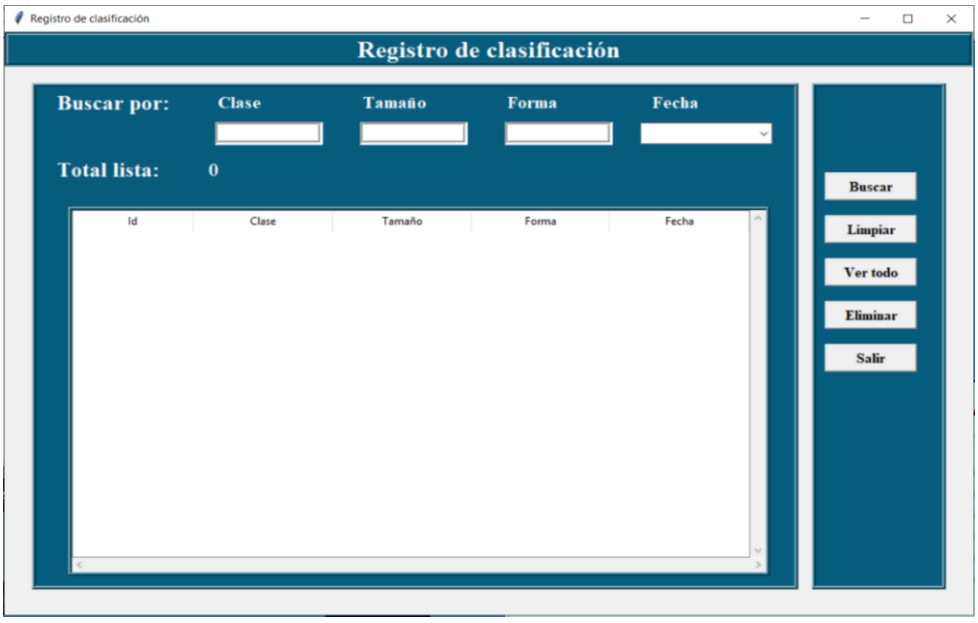
Fuente: Elaboración propia.

6. Ver registro

Mediante esta opción se visualiza otra interfaz que muestra en una tabla todo el registro de clasificación, mediante la cual se puede filtrar con diferentes opciones de búsqueda: Por clase, tamaño, forma, fecha. Se muestra también el total de clasificaciones de acuerdo al tipo de listado que se desee.

Figura 76

Interfaz del registro de clasificación



Fuente: Elaboración propia.

7. Terminar captura

Mediante esta operación se cierra la ejecución de la cámara.

8. Visualización general del sistema clasificador de tuntas

El sistema está establecido en 2 partes: Procesamiento de imagen en tiempo real y resultados del procesamiento.

En los resultados del procesamiento se muestran los indicadores de calidad de una tunta (tamaño, forma, color) y las características geométricas principales y características del color.

Figura 77

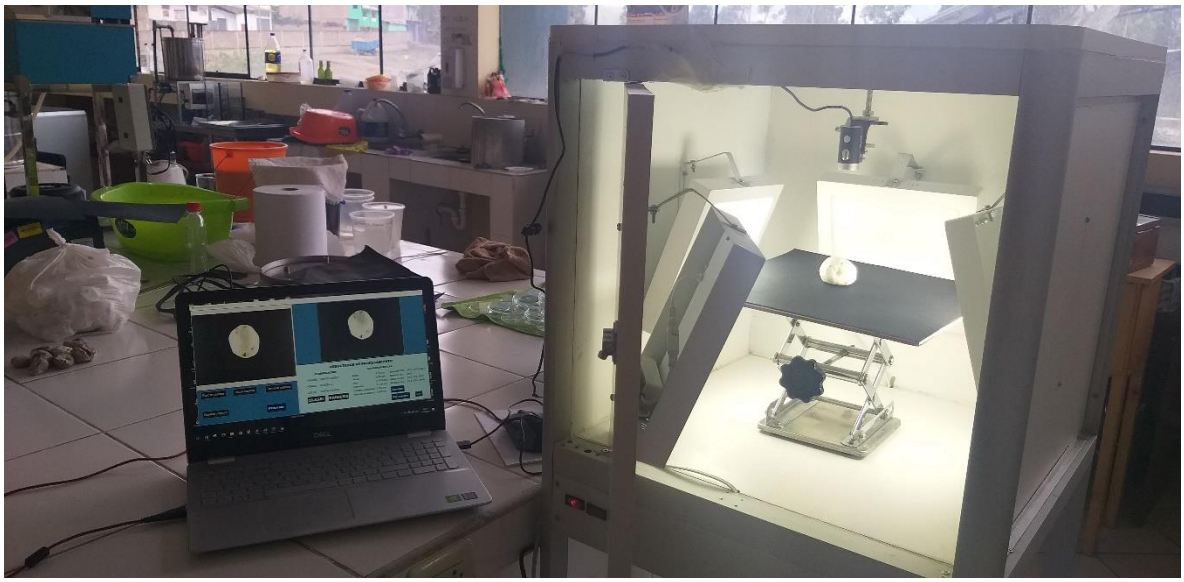
Visualización general del sistema clasificador

Fuente: Elaboración propia.

4.10. Pruebas del sistema

Figura 78

Pruebas del sistema



Fuente: Elaboración propia.

Se realizó las pruebas de clasificación de tuntas de las 4 categorías, obteniéndose resultados que se visualizan en la parte derecha de la interfaz general, de acuerdo al tamaño, forma y color de una tunta, y al final se muestra la clasificación de la tunta (Primera, Segunda, Tercera, Cuarta). También se visualizan las características geométricas de la tunta y las características del color.

Después de realizar el proceso de clasificación en tiempo real, los datos obtenidos se pueden **guardar** en una base de datos y **ver el registro** de clasificación realizadas.

Primera categoría:

Segunda categoría:

Tercera categoría

Cuarta categoría

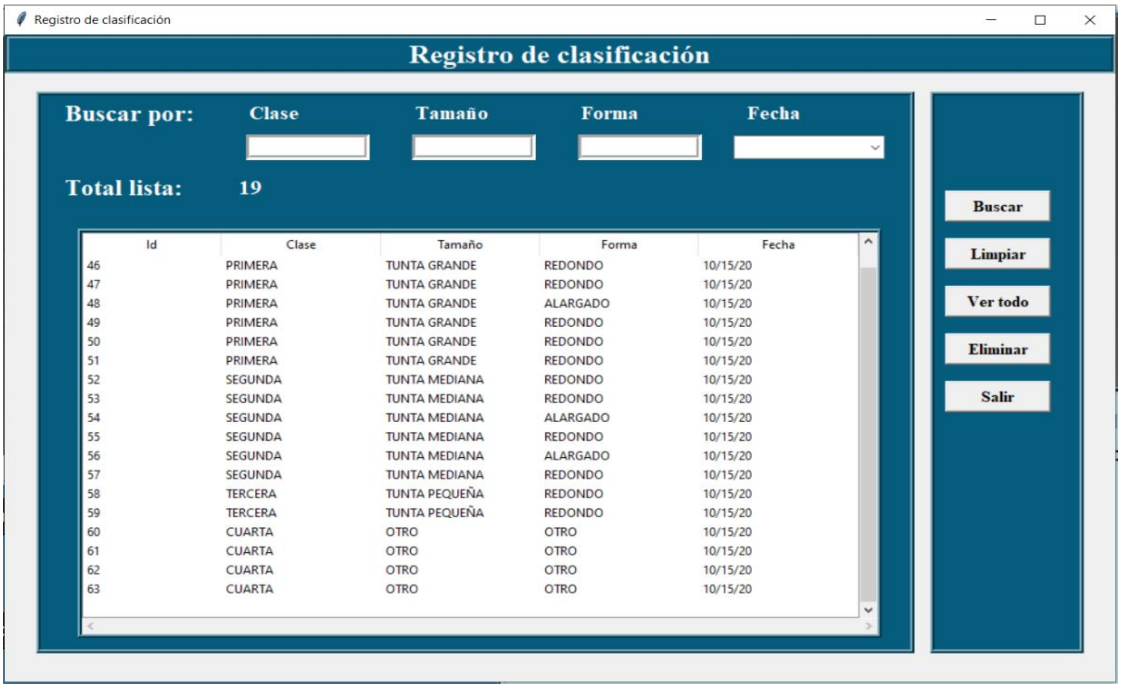
Las tuntas de cuarta categoría son aquellas que poseen manchas oscuras en su textura, partidas y con grietas. Una tunta de cualquier tamaño o forma, si posee manchas oscuras se considera de cuarta categoría y no está apta para la comercialización.

Resultados de clasificación

Se muestra en una tabla los resultados de clasificación realizadas con el sistema en la fecha actual o anteriores, en esta interfaz se pueden filtrar las búsquedas mediante la clase, tamaño, forma y la fecha de clasificación, así como también eliminar registros según conveniencia del usuario.

Figura 79

Resultados de clasificación



Id	Clase	Tamaño	Forma	Fecha
46	PRIMERA	TUNTA GRANDE	REDONDO	10/15/20
47	PRIMERA	TUNTA GRANDE	REDONDO	10/15/20
48	PRIMERA	TUNTA GRANDE	ALARGADO	10/15/20
49	PRIMERA	TUNTA GRANDE	REDONDO	10/15/20
50	PRIMERA	TUNTA GRANDE	REDONDO	10/15/20
51	PRIMERA	TUNTA GRANDE	REDONDO	10/15/20
52	SEGUNDA	TUNTA MEDIANA	REDONDO	10/15/20
53	SEGUNDA	TUNTA MEDIANA	REDONDO	10/15/20
54	SEGUNDA	TUNTA MEDIANA	ALARGADO	10/15/20
55	SEGUNDA	TUNTA MEDIANA	REDONDO	10/15/20
56	SEGUNDA	TUNTA MEDIANA	ALARGADO	10/15/20
57	SEGUNDA	TUNTA MEDIANA	REDONDO	10/15/20
58	TERCERA	TUNTA PEQUEÑA	REDONDO	10/15/20
59	TERCERA	TUNTA PEQUEÑA	REDONDO	10/15/20
60	CUARTA	OTRO	OTRO	10/15/20
61	CUARTA	OTRO	OTRO	10/15/20
62	CUARTA	OTRO	OTRO	10/15/20
63	CUARTA	OTRO	OTRO	10/15/20

Fuente: Elaboración propia.

4.11. Análisis de resultados del sistema

Como se vió en el anterior subcapítulo, se procedió a clasificar las tuntas de las cuatro categorías diferentes a las que han sido destinadas al entrenamiento y prueba del modelo, es decir con tuntas que el sistema no haya visto aún, con el fin de validar su funcionalidad. Para este proceso se dispuso de 120 tuntas de las cuatro categorías.

Se dispuso de 30 unidades de cada categoría para la prueba de clasificación con el sistema.

Mediante la matriz de confusión se visualiza el comportamiento del sistema construido mediante algoritmos de procesamiento digital de imágenes y un modelo entrenado con un algoritmo de machine learning.

Tabla 17

Matriz de confusión de resultados del sistema

		RESULTADOS DEL SISTEMA			
		Primera	**Segunda**	**Tercera**	**Cuarta**
CLASE REAL	**Primera**	29	1	0	0
	Segunda	2	28	0	0
	Tercera	0	0	30	0
	Cuarta	0	0	0	30

Fuente: Elaboración propia.

Al analizar la matriz se llega a los siguientes resultados en el rendimiento de clasificación del sistema construido con algoritmos de procesamiento de imágenes y un modelo de machine learning.

Con respecto al Accuracy, existe una proporción del 97.5% entre el número total de tuntas clasificadas correctamente con respecto a la totalidad de tuntas sometidas a la prueba del clasificador.

Con respecto al Recall, existe también una proporción del 97.5% en el número de tuntas clasificadas como una clase (primera, segunda, tercera, cuarta) dividida por el número de tuntas que realmente pertenecen a esa clase.

Con respecto al F1-Score, esta métrica es la media armónica entre la Precisión y Recall, se obtine una puntuación del 97.51%.

La Precisión es la proporción de tuntas que son verdaderamente de una clase (primera, segunda, tercera, cuarta) dividida por el total de tuntas clasificadas como esa clase.

CONCLUSIONES

En la presente investigación debido a la realidad problemática que existe y directamente con la clasificación de tuntas de acuerdo a sus características de tamaño, forma y color que poseen y que estos son los indicadores principales de la categoría a la que pertenecen (primera, segunda, tercera, cuarta), se recurrió al procesamiento digital de imágenes y el machine learning (aprendizaje automático) para el entrenamiento y evaluación de modelos de clasificación y la posterior contrucción de un sistema clasificador.

La cámara como un componente principal en la entrada de datos al proceso fue la óptima en la presente investigación, la altura desde su posición al lugar del objeto (tunta) fue de 15 cm y es primordial mantener esa altura ya que los algoritmos y el modelo de clasificación se adaptan a la misma. El prototipo de la captura de imágenes que contiene a la cámara, el fondo y el sistema de iluminación se construyó con las medidas específicas. El sistema de iluminación fue de mucha importancia para la captura de imágenes y se requirió de un fondo opuesto al color del producto para facilitar la eficacia del procesamiento digital de imágenes, reduciendo el ruido digital.

Después de haber realizado el procesamiento de imágenes de 800 unidades de tuntas de primera, segunda, tercera y cuarta categoría, y esto permitió la obtención de características del tamaño, forma y color, con estos datos se contruyó el corpus de conocimiento para luego entrenar y obtener modelos de clasificación haciendo uso de algoritmos de aprendizaje automático supervisado: K-Nearest Neighbors, Decision Trees, Gaussian Naive Bayes, Multinomial Naive Bayes, Support Vector Machines. La evaluación de estos modelos se realizó mediante la validación cruzada con el método k-fold.

Después de los procesos mencionados en los anteriores párrafos, se alcanzó los objetivos de la investigación:

1. Se determinó la eficiencia de un sistema de control de calidad mediante procesamiento digital de imágenes y un modelo de machine learning en la clasificación de la tunta, esta mediante las pruebas del sistema que se realizó con 120 unidades de tuntas de las cuatro categorías diferentes a las que han sido destinadas al entrenamiento y prueba del modelo, es decir con tuntas que el sistema no haya visto aún con el fin de validar su rendimiento. Al final de la prueba se obtuvo un Accuracy (exactitud) del 97.5%, Recall del 97.5% y un F1-Score del 97.51%.

La bondad de este sistema es que clasifica las tuntas en tiempo real optimizando los recursos de hardware y software, reduciendo el tiempo de ejecución y aumentando la precisión de clasificación ya que está construida con un modelo entrenado de mejor rendimiento.

2. Se determinó la eficiencia de clasificación del modelo generado con el algoritmo K-Nearest Neighbors de machine learning al evaluar tuntas de cuatro clases. Se llegó a la conclusión que, el modelo con el algoritmo mencionado tiene un Accuracy del 80.88%, Recall del 81.26% y un F1-Score del 80.55%. Esto después de haber probado con vecinos cercanos que oscilaban entre 1 y 10, y determinado que se obtienen mejores resultados con un vecino cercano.

3. Se determinó la eficiencia de clasificación del modelo generado con el algoritmo Decision Trees de machine learning al evaluar tuntas de cuatro clases. Se llegó a la conclusión que, el modelo generado con el algoritmo mencionado tiene un Accuracy del 92.88%, Recall del 93.11% y un F1-Score del 92.93%.

4. Se determinó la eficiencia de clasificación del modelo generado con el algoritmo Gaussian Naive Bayes de machine learning al evaluar tuntas de cuatro clases. Se llegó a la conclusión que, el modelo generado con el algoritmo mencionado tiene un Accuracy del 89.38%, Recall del 89.76% y un F1-Score del 89.43%.

5. Se determinó la eficiencia de clasificación del modelo generado con el algoritmo Multinomial Naive Bayes de machine learning al evaluar tuntas de cuatro clases. Se llegó a la conclusión que, el modelo generado con el algoritmo mencionado tiene un Accuracy del 80.63%, Recall del 80.69% y un F1-Score del 79.82%.

6. Se determinó la eficiencia de clasificación del modelo generado con el algoritmo Support Vector Machines de machine learning al evaluar tuntas de cuatro clases. Se llegó a la conclusión que, el modelo con el algoritmo mencionado tiene un Accuracy del 93.13%, Recall del 93.48% y un F1-Score del 93.18%. Esto después de haber probado con los métodos LinearSVC, SVC y NuSVC, y determinado que se obtienen mejores resultados con el método LinearSVC.

El sistema clasificador de tuntas se construyó haciéndose uso de las siguientes herramientas: lenguaje de programación Python, y las bibliotecas OpenCV, Tkinter, Scikit-learn, PyMySQL, Joblib, integrando algoritmos de procesamiento digital de imágenes y el modelo de clasificación más eficiente.

Además, del análisis de los modelos de clasificación se determinó que el modelo generado con el algoritmo Support Vector Machines es el más eficiente en la clasificación de tuntas.

Por lo tanto, se cumplieron satisfactoriamente los objetivos de la investigación.

RECOMENDACIONES

Después de culminada la investigación se recomienda lo siguiente:

- Cuanto mayor sea la cantidad de datos de cualquier producto que se desee analizar y clasificar, es mejor para la generación de un modelo clasificador, en esta investigación se usó 800 unidades de tuntas.

- Tener una cámara de mejor resolución para la obtención de imágenes, ya que es uno de los primeros pasos y de los más importantes en este tipo de investigaciones.

- Los algoritmos de machine learning, tecnología que actualmente está cautivando a la mayoría de los grandes investigadores en la clasificación y reconocimiento de objetos para la toma de decisiones, es muy eficiente en investigaciones como el que se desarrolló.

- El procesamiento digital de imágenes una herramienta muy importante y que tiene proyecciones en el futuro en las empresas ya que la mayoría utiliza el control de calidad de sus productos de manera manual, y con esta tecnología se aumentaría la precisión de clasificación y la reducción del tiempo del proceso.

- El sistema clasificador hecho con modelos de clasificación de machine learning y procesamiento digital de imágenes se puede implementar en cualquier empresa que tenga la necesidad de analizar imágenes para cualquier fin, como las empresas productoras y agroindustriales, para el control de calidad de los productos destinados al comercio. Las empresas que fabrican piezas para el mercado también pueden utilizar este tipo de sistema para la clasificación, reconocimiento o comparación de objetos mediante análisis de imágenes.

REFERENCIAS BIBLIOGRÁFICAS

Amoedo, D. (s.f.). *MySQL Workbench, herramienta visual para el diseño de bases de datos*. Obtenido de https://ubunlog.com/mysql-workbench-bases-datos/

Anguiano, E. (2009). Naive Bayes Multinomial para Clasificación de Texto usando un esquema de pesado por clases. 1-8. Obtenido de https://ccc.inaoep.mx/~esucar/Clases-mgp/Proyectos/MGP_RepProy_Abr_29.pdf

Barriga, A., & Arrasco, C. (2018). *Diagnóstico automático de roya amarilla en hojas de cafeto aplicando técnicas de procesamiento de imágenes y aprendizaje máquina.* Pontificia Universidad Católica del Perú, Lima, Perú. Obtenido de http://tesis.pucp.edu.pe/repositorio/bitstream/handle/20.500.12404/10343/BARRIGA_ALFONSO_DIAGNOSTICO_PROCESAMIENTO_IMAGENES.pdf?sequence=1&isAllowed=y

Betancourt, G. (2005). LAS MÁQUINAS DE SOPORTE VECTORIAL (SVMs). Obtenido de https://www.redalyc.org/pdf/849/84911698014.pdf

Carmona, E. (2016). Tutorial sobre Máquinas de Vectores de Soporte (SVM). Obtenido de https://www.researchgate.net/publication/263817587_Tutorial_sobre_Maquinas_de_Vectores_Soporte_SVM

Castro, G. (2019). *Aplicación de algoritmos inteligentes para el reconocimiento automático de enfermedades foliares de cultivo de palta.* Universidad Nacional de Moquegua, Moquegua, Perú. Obtenido de https://repositorio.unam.edu.pe/bitstream/handle/UNAM/98/T095_46910098_T.pdf?sequence=1&isAllowed=y

Constante, P., & Gordón, A. (2015). Diseño e implementación de un sistema de visión artificial para clasificación de al menos tres tipos de frutas. *Tesis de titulación.* Escuela Politécnica Nacional, Quito, Ecuador. Obtenido de https://bibdigital.epn.edu.ec/bitstream/15000/11368/1/CD-6457.pdf

Delgado, R. (18 de 07 de 2018). *Introducción a la Validación Cruzada (k-fold Cross Validation) en R*. Obtenido de http://rstudio-pubs-static.s3.amazonaws.com/405322_6d94d05e54b24ba99438f49a6f8662a9.html

Esqueda, J. (2002). Fundamentos de Procesamiento de Imágenes. 1-52. Obtenido de https://www.researchgate.net/publication/316440843_Fundamentos_de_Procesamiento_de_Imagenes

Fonseca, C., & Ordinola, M. (2011). *Mejorando la competitividad de la agroindustria rural: El caso de la tunta en el altiplano peruano.* Lima, perú. Obtenido de https://cgspace.cgiar.org/bitstream/handle/10568/66337/75524.pdf?sequence=2&isAllowed=y

Fonseca, C., Huarachi, E., & Ordinola, M. (2011). Una experiencia de innovación tecnológica y difusión en la producción artesanal de la papa deshidratada: Tunta. *Revista latinoamericana de la papa*, 99-127. Obtenido de https://dialnet.unirioja.es/servlet/articulo?codigo=5512149

Fonseca, C., Huarachi, E., Chura, W., & Cotrado, G. (2007). Guía de las buenas prácticas de procesamiento para la producción artesanal de la tunta. 1-36. Obtenido de http://cipotato.org/wp-content/uploads/2014/09/004295.pdf

Gamarra, C. A., & Ríos, M. S. (s.f.). Aplicación de técnicas de aprendizaje profundo para la clasificación y reconocimiento de objetos en imágenes. *Bogotá.* Universidad Santo

Tomás. Obtenido de https://repository.usta.edu.co/bitstream/handle/11634/10680/2018Gamarracamilo.pdf?sequence=1

Gil, P., & Torres F, O. G. (2004). *Detección de objetos por segmentación multinivel combinada de espacios de color.* Obtenido de https://intranet.ceautomatica.es/old/actividades/jornadas/XXV/documentos/19-abtsufilva.pdf

Gonzales, C. (2011). Máquinas de Vectores Soporte. *Grupo de Sistemas Inteligentes*, 1-52. Obtenido de https://www.infor.uva.es/~calonso/MUI-TIC/MineriaDatos/SVM.pdf

González, A., Martínez, F., Pernía, A., Alba, F., Castelón, M., Ordieres, J., & Vergara, E. (2006). *Técnicas y algoritmos básicos de visión artificial.* Logroño, España: Universidad de la Rioja. Obtenido de https://www.researchgate.net/publication/231521316_Tecnicas_y_algoritmos_basicos_de_vision_artificial_Recurso_electronico_-_En_linea

Gracia, L. (2013). *Un poco de Java y* +. Obtenido de https://unpocodejava.com/2013/10/09/que-es-opencv/

Grant, M. (24 de abril de 2019). *Investopedia*. Obtenido de Investopedia: https://www.investopedia.com/terms/c/coefficient-of-determination.asp

Guidi, A., Esprella, R., Aguilera, J., & Devaux, A. (2002). *Características de la Cadena Agroalimentaria de Chuño y Tunta para el Altiplano Central de Bolivia.* Cochabamba, Bolivia. Obtenido de https://www.proinpa.org/tic/pdf/Papa/Varios%20Papa/Caracteristicas%20de%20la%20cadena%20agroalimentaria%20de%20chuno%20y%20tunta%20para%20el%20Altiplano%20Central%20de%20Bolivia.pdf

Heras, D. (2017). Clasificador de imágenes de frutas basado en inteligencia artificial. *Revista Killkana Técnica, 1*(2), 21-30. Obtenido de https://www.researchgate.net/publication/321176883_Clasificador_de_imagenes_de_frutas_basado_en_inteligencia_artificial

Hernández, C., & Dueñas, M. (2009). Hacia una metodología de gestión del conocimiento basada en minería de datos. Obtenido de http://repositorio.uigv.edu.pe/bitstream/handle/20.500.11818/982/COMTEL-2009-80-96.pdf?sequence=1&isAllowed=y

Hernández, R., Fernández, C., & Baptista, M. d. (2016). *Metodología de la investigación.* México D.F., México: McGraw-Hill. Obtenido de https://www.uca.ac.cr/wp-content/uploads/2017/10/Investigacion.pdf

Huilgol, P. (24 de Agosto de 2019). *Analytics Vidhya*. Obtenido de https://medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2

Hurwitz, J., & Kirsch, D. (2018). *Machine Learning.* John Wiley. Obtenido de https://www.ibm.com/downloads/cas/GB8ZMQZ3

Magro, R. (2013). Binarización de imágenes digitales y su algoritmia como herramienta aplicada a la ilustración entomlógica. 443-464. Obtenido de http://sea-entomologia.org/PDF/Boletin53/443464BSEA53BinarizacionRMagro.pdf

Martinez, J. (09 de 10 de 2020). *AIrtificial.net*. Obtenido de https://www.iartificial.net/precision-recall-f1-accuracy-en-clasificacion/

Matínez, J. (2020). *Librerías de Python para Machine Learning*. Obtenido de https://iartificial.net/librerias-de-python-para-machine-learning/#scikitlearn

Montoya, C., Cortés, J., & Chaves, J. (2014). Sistema automático de reconocimiento de frutas basado en visión por computador. *Ingeniare. Revista chilena de ingeniería, 22*(4), 504-516. doi:10.4067/S0718-33052014000400006

Moujahid, A., Inza, I., & Larrañaga, P. (s.f.). *Clasificadores K-NN.* Obtenido de http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/t9knn.pdf

Norma Técnica Peruana. (2007). Papa deshidratada. Tunta. Obtenido de http://www.asocam.org/sites/default/files/publicaciones/files/086d8d077f3eaa3bdc5c221b2918238d.pdf

Peñarrieta, M., & Alvarado, J. (2012). Chuño and Tunta; the traditional Andean sun-dried Potatoes. Obtenido de https://www.researchgate.net/publication/236577348_Chuno_and_Tunta_the_traditional_Andean_sun-dried_Potatoes

Pillou, J.-F. (16 de 10 de 2008). *CCM*. Obtenido de https://es.ccm.net/contents/727-codigo-hsl

Roche, A. (2009). *Árboles de decisión y Series de tiempo.* UDELAR, Montevideo. Obtenido de http://premat.fing.edu.uy/ingenieriamatematica/archivos/tesis_ariel_roche.pdf

Rodríguez, D. (2018). *Analytics Lane*. Obtenido de https://www.analyticslane.com/2018/07/13/aprendizaje-supervisado-y-aprendizaje-no-supervisado/

Sammut, C., & Webb, G. (2010). *Encyclopedia of Machine Learning.* doi:10.1007/978-0-387-30164-8

Sandoval, Z., & Prieto, F. (2009). Procesamiento de imágenes para la clasificación de café cereza. *Prospectiva, 7*(1), 67-73. Obtenido de https://www.redalyc.org/articulo.oa?id=496250975010

Sucar, E., & Gómez, G. (s.f.). *Visión computacional.* Puebla, México. Obtenido de https://ccc.inaoep.mx/~esucar/Libros/vision-sucar-gomez.pdf

Sucar, L. (2006). Redes bayesianas. *Aprendizaje Automático: conceptos*. Obtenido de https://ccc.inaoep.mx/~esucar/Clases-mgp/caprb.pdf

Sullca, C., Molina, C., Rodríguez, C., & Fernández, T. (2018). Detección de enfermedades y plagas en las hojas de arándanos utilizando técnicas de visión artificial. *Perspectivas: Revista de Tecnología e Información*, 32-39. Obtenido de http://revistas.uigv.edu.pe/index.php/perspectiva/article/view/590/513

Triana, N., Jaramillo, A., Gutiérrez, R., & C, R. (2016). Técnicas de umbralización para el procesamiento digital de imágenes de GEM-Foils. *Scientia Et Technica, 21*, 351-359. Obtenido de https://revistas.utp.edu.co/index.php/revistaciencia/article/view/13271

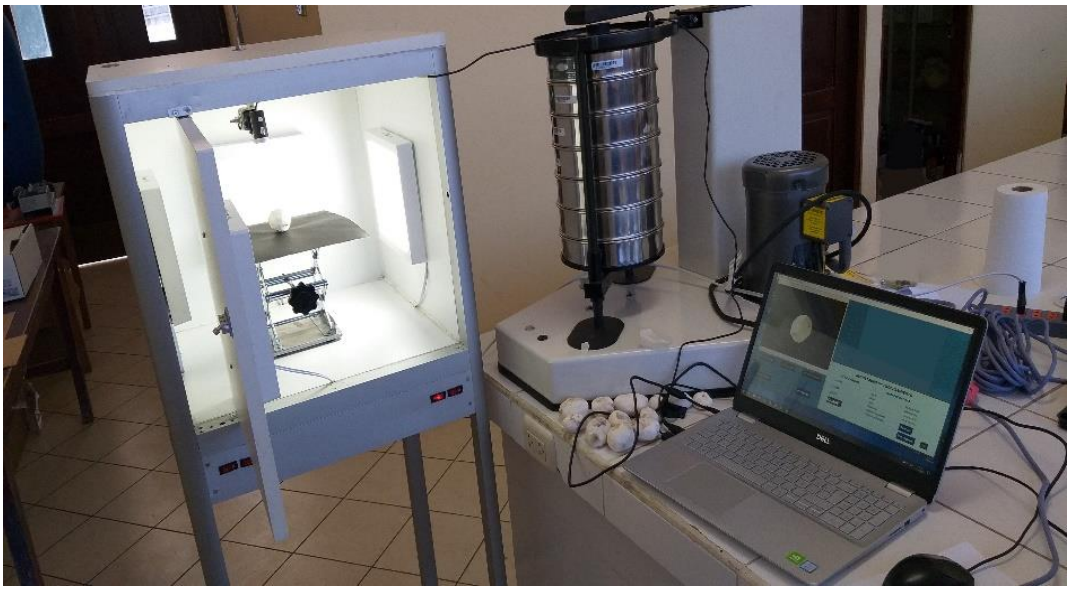
Van Rossum, G. (2017). *El tutorial de Python.* (J. Fred L. Drake, Ed.) Argentina. Obtenido de https://argentinaenpython.com/quiero-aprender-python/TutorialPython3.pdf

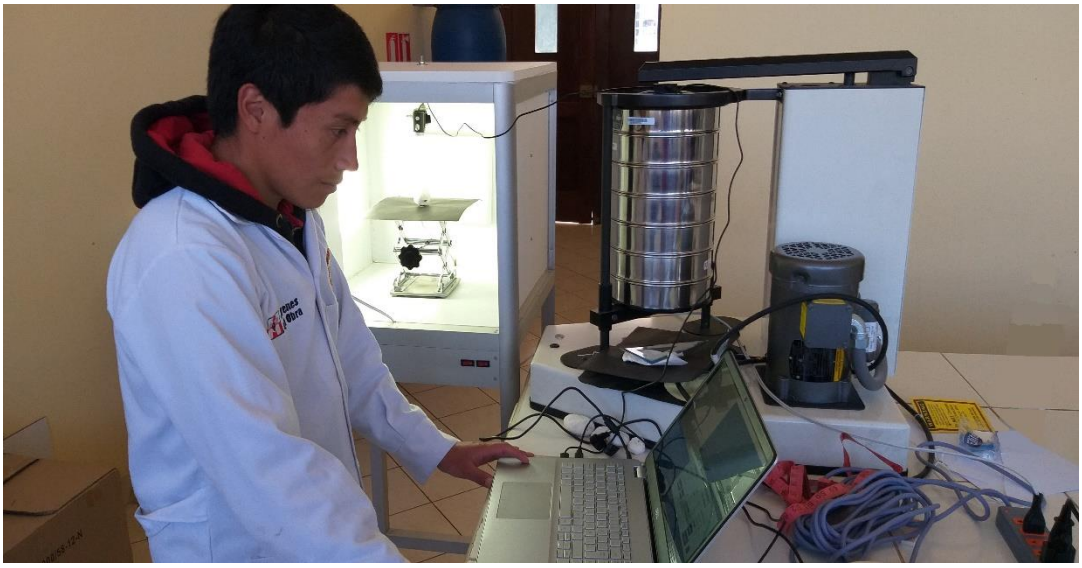
Vásquez, Á. (2017). *Métodos para evaluar hipótesis (resultado de un algoritmo de aprendizaje).* Obtenido de https://es.slideshare.net/angenio2/machine-learning-evaluacin-de-hiptesis

Velez, J., Moreno, A., Sáchez, J., & Sáchez, Á. (2003). *Visión por computador.* (Dykinson, Ed.) Obtenido de http://www.visionporcomputador.es/libroVision/VisionPorComputador.pdf

ANEXOS

Anexo 1: Fotografías de la investigación





Anexo 2: Resultados de las validaciones cruzadas de los modelos

A continuación, se presentan los resultados de las primeras 5 iteraciones de la validación cruzada de cada modelo de clasificación:

1. Resultado del modelo K-Nearest Neighbors con 1 vecino cercano

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=1, p=2,
                     weights='uniform')
KFold(n_splits=10, random_state=2, shuffle=True)
------------Iteración 1---------------
ACC:     0.85
RECALL:  0.8368
PC:      0.8435
F1:      0.839
------------Iteración 2---------------
ACC:     0.8
RECALL:  0.8009
PC:      0.8034
F1:      0.8009
------------Iteración 3---------------
ACC:     0.8
RECALL:  0.7961
PC:      0.8033
F1:      0.7985
------------Iteración 4---------------
ACC:     0.75
RECALL:  0.7527
PC:      0.7659
F1:      0.7468
------------Iteración 5---------------
ACC:     0.9
RECALL:  0.9054
PC:      0.9029
F1:      0.903
```

2. Resultado del modelo Decision Trees

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None,
                       max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
KFold(n_splits=10, random_state=2, shuffle=True)
------------Iteración 1---------------
ACC:     0.925
RECALL:  0.9252
PC:      0.921
F1:      0.9228
------------Iteración 2---------------
ACC:     0.9625
RECALL:  0.9628
PC:      0.963
F1:      0.962
------------Iteración 3---------------
ACC:     0.9375
RECALL:  0.9325
PC:      0.9335
F1:      0.9327
------------Iteración 4---------------
ACC:     0.8875
RECALL:  0.9027
PC:      0.9005
F1:      0.8861
------------Iteración 5---------------
ACC:     0.95
RECALL:  0.9464
PC:      0.9594
F1:      0.9522
```

3. Resultado del modelo Gaussian Naive Bayes

```
GaussianNB(priors=None, var_smoothing=1e-09)
KFold(n_splits=10, random_state=2, shuffle=True)
------------Iteración 1---------------
ACC:     0.925
RECALL:  0.9231
PC:      0.9317
F1:      0.9236
------------Iteración 2---------------
ACC:     0.875
RECALL:  0.8753
PC:      0.8843
F1:      0.8769
------------Iteración 3---------------
ACC:     0.9
RECALL:  0.8988
PC:      0.8974
F1:      0.897
------------Iteración 4---------------
ACC:     0.8125
RECALL:  0.8342
PC:      0.8479
F1:      0.8119
------------Iteración 5---------------
ACC:     0.95
RECALL:  0.9452
PC:      0.9569
F1:      0.9503
```

4. Resultado del modelo Multinomial Naive Bayes

```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
KFold(n_splits=10, random_state=2, shuffle=True)
------------Iteración 1---------------
ACC:     0.875
RECALL:  0.8528
PC:      0.8734
F1:      0.8572
------------Iteración 2---------------
ACC:     0.75
RECALL:  0.7506
PC:      0.7638
F1:      0.7533
------------Iteración 3---------------
ACC:     0.8
RECALL:  0.7999
PC:      0.8005
F1:      0.7984
------------Iteración 4---------------
ACC:     0.8
RECALL:  0.8137
PC:      0.8208
F1:      0.7942
------------Iteración 5---------------
ACC:     0.8
RECALL:  0.7924
PC:      0.8059
F1:      0.7755
```

5. Resultado del modelo SVM con el método LinearSVC

```
LinearSVC(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
KFold(n_splits=10, random_state=2, shuffle=True)
------------Iteración 1---------------
ACC:     0.975
RECALL:  0.975
PC:      0.98
F1:      0.9764
------------Iteración 2---------------
ACC:     0.9125
RECALL:  0.9148
PC:      0.9281
F1:      0.915
------------Iteración 3---------------
ACC:     0.95
RECALL:  0.9511
PC:      0.948
F1:      0.9484
------------Iteración 4---------------
ACC:     0.9
RECALL:  0.9054
PC:      0.9006
F1:      0.8972
------------Iteración 5---------------
ACC:     0.95
RECALL:  0.9425
PC:      0.9631
F1:      0.951
```

6. Resultado del modelo SVM con el método SVC

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
KFold(n_splits=10, random_state=2, shuffle=True)
------------Iteración 1---------------
ACC:     0.825
RECALL:  0.7967
PC:      0.8147
F1:      0.7989
------------Iteración 2---------------
ACC:     0.7125
RECALL:  0.7134
PC:      0.7233
F1:      0.715
------------Iteración 3---------------
ACC:     0.7625
RECALL:  0.7624
PC:      0.7552
F1:      0.7549
------------Iteración 4---------------
ACC:     0.725
RECALL:  0.733
PC:      0.7314
F1:      0.7036
------------Iteración 5---------------
ACC:     0.75
RECALL:  0.7374
PC:      0.7411
F1:      0.7101
```

7. Resultado del modelo SVM con el método NuSVC

```
NuSVC(break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
      decision_function_shape='ovr', degree=3, gamma='scale',
      kernel='rbf',
      max_iter=-1, nu=0.5, probability=False, random_state=None,
      shrinking=True, tol=0.001, verbose=False)
KFold(n_splits=10, random_state=2, shuffle=True)
------------Iteración 1---------------
ACC:     0.875
RECALL:  0.8528
PC:      0.8828
F1:      0.8601
------------Iteración 2---------------
ACC:     0.7125
RECALL:  0.7157
PC:      0.7331
F1:      0.7182
------------Iteración 3---------------
ACC:     0.8125
RECALL:  0.8158
PC:      0.8367
F1:      0.7939

------------Iteración 4---------------
ACC:     0.75
RECALL:  0.7619
PC:      0.7685
F1:      0.7483
------------Iteración 5---------------
ACC:     0.7875
RECALL:  0.7658
PC:      0.7821
F1:      0.7634
```

Anexo 3: Código fuente de las principales funciones del sistema clasificador

Archivo Fotogramas.py

```
import cv2
from PIL import Image
from PIL import ImageTk
import tkinter.messagebox as msg

class camara:
	camara = None
	frame=None
	image=None
	frame1=None
	image1=None
	frameprocesado=None
	imagenprocesado=None

	def __init__(self):
		pass

	def lectura(self, matrix):

		self.frame=cv2.resize(matrix, (600,440),
		interpolation=cv2.INTER_LINEAR)
		self.frame1=cv2.cvtColor(self.frame, cv2.COLOR_BGR2RGB)
		self.image1=Image.fromarray(self.frame1)
		self.image1=ImageTk.PhotoImage(self.image1)
		return self.image1

	def procesoimagen(self, matrix):

		self.frameprocesado=cv2.resize(matrix, (540,400),
		interpolation=cv2.INTER_LINEAR)
		self.frameprocesado=cv2.cvtColor(self.frameprocesado,
		cv2.COLOR_BGR2RGB)
		self.imagenprocesado=Image.fromarray(self.frameprocesado)

		self.imagenprocesado=ImageTk.PhotoImage(self.imagenprocesado)
		return self.imagenprocesado
```

Archivo Gestionbd.py

```
from tkinter import *
from tkinter import ttk
import pymysql
from tkinter import messagebox
from datetime import date, time
from tkcalendar import *

class Gestion:
       Detail_Frame=None
       lbl_conteo=None
       txt_fechab= None
       txt_claseb=None
       txt_tamaniob=None
       txt_formab = None

       def total(self, cont):

            self.lbl_conteo = Label(self.Detail_Frame, text = cont, bg = "#055C7D",
            fg="white", font=("times new roman", 16, "bold"))
            self.lbl_conteo.grid(row=2, column=1, pady=10, padx=10, sticky="w")

       def fetch_data(self):
           cont = 0
           con = pymysql.connect(host="localhost", user="root",
            password="1234",database="dbtunta")
           cur = con.cursor()
           cur.execute("select * from tbtunta")
           rows = cur.fetchall()
           if len(rows)!=0:

              self.tunta_table.delete(*self.tunta_table.get_children())
                     self.lbl_conteo.destroy()
                     for row in rows:
                            self.tunta_table.insert('',END, values=row)
                            cont=cont+1
                     con.commit()
           self.total(cont)
           self.clear()
           con.close()


       def get_cursor(self,ev):
           cursor_row=self.tunta_table.focus()
           contents = self.tunta_table.item(cursor_row)
           row=contents["values"]

           self.id.set(row[0])
           self.clase.set(row[1])
           self.tamanio.set(row[2])
```

```
    self.forma.set(row[3])
    self.fecha.set(row[4])

def delete_data(self):
     con = pymysql.connect(host="localhost", user="root",
     password="1234",database="dbtunta")
     cur = con.cursor()
     cur.execute("delete from tbtunta where id=%s", self.id.get())
     con.commit()
     con.close()
     self.fetch_data()
     self.clear()

def search_data(self):
   fecha = str(self.txt_fechab.get())
   cont= 0
    con = pymysql.connect(host="localhost", user="root",     password =
    "1234",database="dbtunta")
    cur = con.cursor()
   if (self.txt_claseb.get()=="" and self.txt_tamaniob.get()=="" and
   self.txt_formab.get() =="" and self.txt_fechab.get() == ""):
       messagebox.showerror("Mensaje", "Ingrese búsqueda")
   else:
      self.lbl_conteo.destroy()
      cur.execute("select * from tbtunta where clase=%s or tamanio=%s or
      forma=%s or fecha=%s", (self.txt_claseb.get(),
      self.txt_tamaniob.get(),
      self.txt_formab.get(),
      fecha))
      rows = cur.fetchall()
      if len(rows)!=0:

            self.tunta_table.delete(*self.tunta_table.get_children()))
            for row in rows:
                self.tunta_table.insert('',END, values=row)
              cont=cont+1
              con.commit()
      else:
            self.tunta_table.delete(*self.tunta_table.get_children())
      self.clear()
      self.total(cont)
      con.close()
```

Archivo Procesamiento.py

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
import joblib
import Fotogramas

class Procesamiento:
	imagen = None
	def __init__(self,img):
		self.imagen = img
	def Contornos(self, imagen):
		imgray=cv2.cvtColor(imagen,cv2.COLOR_BGR2GRAY)
		dif = cv2.GaussianBlur(imgray, (5,5), 5)
		ret, thresh = cv2.threshold(dif, 0, 255,
		cv2.THRESH_BINARY+cv2.THRESH_OTSU)
		kernel = np.ones((7,7),np.uint8)
		closing = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, kernel,iterations
		= 5)
		opening = cv2.morphologyEx(closing, cv2.MORPH_OPEN, kernel,iterations
		= 1)
		contours, hierarchy = cv2.findContours(opening,1,2)
		return contours

	def Ancho(self, imagen):
		contornos = self.Contornos(imagen)[0] (x,y),(ejemenor,ejemayor),angle
		= cv2.fitEllipse(contornos)
		return round(ejemenor,5)
	def Altura(self, imagen):
		contornos = self.Contornos(imagen)[0]
		(x,y),(ejemenor,ejemayor),angle = cv2.fitEllipse(contornos)
		return round(ejemayor,5)
	def Area(self, imagen):
		contornos = self.Contornos(imagen)[0]
		area = cv2.contourArea(contornos)
		return round(area,2)
	def Perimetro(self, imagen):
		contornos = self.Contornos(imagen)[0]
		perimetro = cv2.arcLength(contornos,True)
		return round(perimetro,5)
	def RelacionAspecto(self, imagen):
		contornos = self.Contornos(imagen)[0]
		(x,y),(ejemenor,ejemayor),angle = cv2.fitEllipse(contornos)
		aspect_ratio=float(ejemenor/ejemayor)
		return round(aspect_ratio,5)
	def DiametroEquivanlente(self,imagen):
		contornos = self.Contornos(imagen)[0]
		area = cv2.contourArea(contornos)
		equi_diametro = np.sqrt(4*area/np.pi)
		return round(equi_diametro,5)
	def ColormeanRGB(self,imagen):
```

```
        imagena=cv2.cvtColor(imagen, cv2.COLOR_BGR2RGB)
        contornos = self.Contornos(imagen)[0]
        imgray = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)
        mask = np.zeros(imgray.shape,np.uint8)
        cv2.drawContours(mask,[contornos],0,255,-1)
        mean_rgb = cv2.mean(imagena,mask = mask)
        return mean_rgb
    def ColormeanGris(self, imagen):
        contornos = self.Contornos(imagen)[0]
        imgray=cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)
        #cv2.imshow("d",imgray)
        mask=np.zeros(imgray.shape, np.uint8)
        cv2.drawContours(mask, [contornos],0,255,-1)
        mean_gris = cv2.mean(imgray,mask=mask)
        return mean_gris
    def ColormeanHSV(self,imagen):
        contornos = self.Contornos(imagen)[0]
        imgray = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)
        hsv=cv2.cvtColor(imagen, cv2.COLOR_BGR2HSV)
        mask = np.zeros(imgray.shape, np.uint8)
        cv2.drawContours(mask, [contornos],0,255,-1)
        mean_hsv = cv2.mean(hsv, mask = mask)
        return mean_hsv
    def ColormeanHLS(self,imagen):
        contornos = self.Contornos(imagen)[0]
        imgray = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)
        hls = cv2.cvtColor(imagen, cv2.COLOR_BGR2HLS)
        mask = np.zeros(imgray.shape, np.uint8)
        cv2.drawContours(mask, [contornos],0,255,-1)
        mean_hls = cv2.mean(hls, mask = mask)
        return mean_hls

    def prediccion(self):
        modelo = joblib.load('modelo/modelo_entrenado.pkl')
        clase = modelo.predict([[self.Ancho(self.imagen),
                self.Altura(self.imagen),
                self.RelacionAspecto(self.imagen),
                self.Perimetro(self.imagen),
                self.Area(self.imagen),
                self.DiametroEquivanlente(self.imagen),
                self.ColormeanRGB(self.imagen)[0],
                self.ColormeanRGB(self.imagen)[1],
                self.ColormeanRGB(self.imagen)[2],
                self.ColormeanGris(self.imagen)[0],
                self.ColormeanHSV(self.imagen)[0],
                self.ColormeanHSV(self.imagen)[1],
                self.ColormeanHSV(self.imagen)[2],
                self.ColormeanHLS(self.imagen)[0],
                self.ColormeanHLS(self.imagen)[1],
                self.ColormeanHLS(self.imagen)[2]]])
```

```
        print("-------Características--------")
        print('Ancho: ',self.Ancho(self.imagen))
        print('Altura: ',self.Altura(self.imagen))
        print('Aspecto: ',self.RelacionAspecto(self.imagen))
        print('Perimetro: ',self.Perimetro(self.imagen))
        print('Area: ',self.Area(self.imagen))
        print('Diametro :',self.DiametroEquivanlente(self.imagen))
        print('R(media): ',self.ColormeanRGB(self.imagen)[0])
        print('G(media): ',self.ColormeanRGB(self.imagen)[1])
        print('B(media): ',self.ColormeanRGB(self.imagen)[2])
        print('Gris(media): ',self.ColormeanGris(self.imagen)[0])
        print('H(media): ',self.ColormeanHSV(self.imagen)[0])
        print('S(media): ',self.ColormeanHSV(self.imagen)[1])
        print('V(media): ',self.ColormeanHSV(self.imagen)[2])
        print('H(hls): ',self.ColormeanHLS(self.imagen)[0])
        print('L(hls): ',self.ColormeanHLS(self.imagen)[1])
        print('S(hls): ',self.ColormeanHLS(self.imagen)[2])

        return clase

    def atributos(self):

        ancho = self.Ancho(self.imagen)
        altura = self.Altura(self.imagen)
        area = self.Area(self.imagen)
        perimetro = self.Perimetro(self.imagen)
        diaequival = self.DiametroEquivanlente(self.imagen)
        relaspecto = self.RelacionAspecto(self.imagen)
        rgb =
        round(self.ColormeanRGB(self.imagen)[0],1),round(self.ColormeanRGB(se
        lf.imagen)[1],1),round(self.ColormeanRGB(self.imagen)[2],1)
        gris = round(self.ColormeanGris(self.imagen)[0],1)
        hsv =
        round(self.ColormeanHSV(self.imagen)[0],1),round(self.ColormeanHSV(se
        lf.imagen)[1],1),round(self.ColormeanHSV(self.imagen)[2],1)
        hls =
        round(self.ColormeanHLS(self.imagen)[0],1),round(self.ColormeanHLS(se
        lf.imagen)[1],1),round(self.ColormeanHLS(self.imagen)[2],1)

        ancho = round(ancho*0.025,3)
        altura = round(altura*0.025,3)
        perimetro = round(perimetro*0.025,3)
        diaequival = round(diaequival*0.025,3)
        area = round(((diaequival*diaequival/4)*3.1415),3)
        relaspecto = round(relaspecto,3)

        return ancho, altura, area, perimetro, diaequival, relaspecto, rgb,
        gris, hsv, hls

    def imagendibujada(self):
        obj=Fotogramas.camara()
```

```
            imagenc = self.imagen
            var  = obj.procesoimagen(imagenc)
            return var
```

Archivo Seleccionacamara.py

```
import tkinter as tk
from tkinter import ttk
class Seleccioncamara:

      ventana=None
      labelseleccioncamara=None
      boton=None
      ventanaprincipal=None
      combobox=None
      def __init__(self, root):
            self.ventanaprincipal=root
            self.ventana=tk.Toplevel(self.ventanaprincipal, bg="#1A8EA6",
            relief="groove")
            self.ventana.title("Seleccione la camara")
            self.ventana.geometry("500x170+80+120")
            self.ventana.overrideredirect(1)
            self.ventana.wm_attributes("-topmost",1)
            self.ventanaprincipal.wm_attributes("-disabled",1)
            self.labelseleccioncamara=tk.Label(self.ventana, text="Seleccione
            cámara", font=("Arial",14))
            self.labelseleccioncamara.place(x=10, y=10)

            #Combobox
            text_font = ("Arial", 13)
            self.combobox=ttk.Combobox(self.ventana, font=text_font)
            self.ventana.option_add("*TCombobox*Listbox*Font", text_font)
            self.combobox["values"]=[0,1,2]
            self.combobox.current(0)
            self.combobox.place(x=250, y=10)
            self.boton=tk.Button(self.ventana, text="Aceptar", width=10,
            height=1)
            self.boton.place(x=220, y=70)
            self.boton.config(command=self.eventoaceptar, overrelief="raised",
            bg="#232928", cursor="hand2", font=("Arial",14), bd=3, fg="#fff")

      def aceptar(self):
            self.ventanaprincipal.wm_attributes("-disabled",0)
            self.ventana.destroy()

      def capturarvalor(self):
            return self.combobox.get()
```