Horizonte Académico
Revista Multidisciplinaria
Recibido:09/10/2025 Aprobado:10/11/2025

DOI: https://doi.org/10.70208/3007.8245.v5.n3.278

Técnicas de inteligencia artificial para el reconocimiento automático del alfabeto braille: un enfoque aplicado a la inclusión educativa

Danilo José Regalado López
danilojoseregalado@gmail.com
Escuela Superior de Ingeniería y Tecnología
Universidad Internacional de la Rioja

RESUMEN

La discapacidad visual constituye una de las principales barreras para el acceso equitativo a la educación, especialmente cuando el uso del sistema braille no puede ser comprendido por docentes y estudiantes videntes. En respuesta a esta problemática, esta investigación desarrolló un software de reconocimiento automático del alfabeto braille mediante técnicas de inteligencia artificial y visión por computador. Se utilizó una base de datos de 1319 imágenes de símbolos braille procesadas en MATLAB, aplicando algoritmos supervisados como Fine Tree, Linear Discriminant, Naive Bayes, Support Vector Machines y K-Nearest Neighbors. Los modelos fueron entrenados, validados y comparados a través de métricas de precisión, sensibilidad y velocidad de procesamiento. Los resultados evidenciaron niveles de exactitud superiores al 99 % en los clasificadores más robustos, destacando el desempeño del modelo SVM. Estos hallazgos confirman la viabilidad de la propuesta como herramienta de traducción en tiempo real entre braille y alfabeto latino, contribuyendo a la reducción de barreras comunicacionales y al fortalecimiento de entornos educativos inclusivos.

Palabras clave: inteligencia artificial, braille, visión por computador, inclusión educativa, reconocimiento de patrones

Artificial Intelligence Techniques for Automatic Recognition of the Braille Alphabet: An Applied Approach to Educational Inclusion

ABSTRACT

Visual disability represents one of the main barriers to equitable access to education, particularly when the use of the Braille system cannot be understood by sighted teachers and students. In response to this challenge, this research developed automatic Braille alphabet recognition software using artificial intelligence and computer vision techniques. A dataset of 1,319 Braille symbol images was processed in MATLAB, applying supervised algorithms such as Fine Tree, Linear Discriminant, Naive Bayes, Support Vector Machines, and K-Nearest Neighbors. Models were trained, validated, and compared using accuracy, sensitivity, and processing speed metrics. Results showed accuracy levels above 99% in the most robust classifiers, with SVM standing out as the best-performing model. These findings confirm the feasibility of the proposal as a real-time translation tool between Braille and the Latin alphabet, contributing to the reduction of communication barriers and the strengthening of inclusive educational environments.

Keywords: artificial intelligence, braille, computer vision, educational inclusion, pattern recognition

INTRODUCCIÓN

La discapacidad visual constituye una de las limitaciones más relevantes en el ámbito educativo y social. Según la Organización Mundial de la Salud (OMS, 2014), las personas con ceguera o baja visión enfrentan dificultades significativas en la realización de actividades cotidianas, lo que repercute directamente en su inclusión social y en sus posibilidades de acceso a la educación. Entre los principales retos, se destaca la barrera comunicacional que se genera cuando los estudiantes invidentes utilizan el sistema braille, pero los docentes no poseen las competencias para interpretarlo. Esta situación no solo limita la interacción pedagógica, sino que también incrementa la dependencia, la inseguridad y el riesgo de exclusión educativa de los estudiantes (Ojeda & Bolaños, 2015).

El braille, creado por Louis Braille en 1824, se basa en la combinación de seis puntos en relieve distribuidos en dos columnas y tres filas. Mediante 64 combinaciones posibles, este sistema permite representar letras, números, signos de puntuación y, en variantes más recientes, símbolos para disciplinas específicas como música, química o informática (Martínez & Polo, 2004). Aunque constituye un medio universalmente reconocido para la lectoescritura de personas con discapacidad visual, su comprensión por parte de la población vidente sigue siendo limitada. De hecho, cuando el sistema braille es excluido de los procesos educativos, se perpetúa la exclusión y aumenta la deserción estudiantil entre jóvenes con discapacidad visual (Lucas, 2018).

En respuesta a estas dificultades, se han desarrollado diversas ayudas técnicas y dispositivos. Ejemplos históricos incluyen la máquina Perkins, las líneas braille y los sintetizadores de voz, que contribuyeron a mejorar la comunicación escrita y oral de los estudiantes invidentes. Sin embargo, estos recursos suelen implicar elevados costos, requieren que el docente domine el braille o presentan limitaciones en cuanto a velocidad y fidelidad en la traducción de los textos

(Escobar et al., 2014).

En los últimos años, el avance de la inteligencia artificial (IA) y la visión por computador ha permitido diseñar aplicaciones innovadoras para mitigar estas barreras. Entre ellas, destaca Seeing AI de Microsoft, lanzada en 2017, que describe el entorno a través de una cámara y un sintetizador de voz, aunque no reconoce braille (Merino, 2019; Microsoft, 2020). Asimismo, en Argentina se desarrolló Braileo, una app en fase de prueba que traduce imágenes de textos en braille al castellano, pero su implementación ha enfrentado problemas relacionados con la calidad de las imágenes capturadas (Di Tulio, 2017; Todo Noticias, 2018). En España, el sistema Drishti combina telefonía inteligente y reconocimiento de imágenes para describir el entorno, aunque tampoco reconoce símbolos braille (Universidad Internacional de Valencia, 2020).

Pese a los avances, la mayoría de estas soluciones se centran en un único actor —ya sea el estudiante invidente o el docente—, dejando sin resolver la necesidad de un espacio inclusivo de comunicación bidireccional. En consecuencia, persiste la necesidad de herramientas tecnológicas capaces de traducir de manera automática y confiable el braille al alfabeto convencional, beneficiando simultáneamente a estudiantes y docentes.

La presente investigación surge en este contexto y propone el diseño de un software de reconocimiento automático de símbolos braille, basado en técnicas de inteligencia artificial y utilizando la plataforma MATLAB. La propuesta se distingue de otras alternativas por tres características clave:

1. Uso de big data: se emplea un conjunto amplio de imágenes de símbolos braille, lo que permite entrenar modelos de clasificación con alto nivel de precisión.
2. Aplicación de algoritmos supervisados: se implementan y comparan clasificadores como Fine Tree, Linear Discriminant, Naive Bayes, Support Vector Machines y K-

Nearest Neighbors, optimizando la fidelidad del reconocimiento.

3. Implementación en tiempo real: a diferencia de sistemas basados únicamente en fotografías, el software procesa imágenes obtenidas por video, lo que facilita el reconocimiento inmediato y dinámico de los caracteres braille.

El impacto esperado de esta herramienta se enmarca en la educación superior, ámbito en el cual los estudiantes invidentes enfrentan mayores dificultades de inclusión (CONADIS, 2019). Al proporcionar un sistema capaz de traducir en tiempo real los textos en braille al alfabeto latino, se fortalece la independencia de los estudiantes, se facilita la labor docente y se contribuye a un entorno académico más inclusivo y equitativo.

METODOLOGÍA

Cuando se definió el tema y se delimitó el problema, se inició el proceso de diseño y desarrollo del software con el cual se soluciona la situación comunicacional de los estudiantes invidentes con profesores videntes.

El proceso se desarrolló por fases.

Fase de recolección de información

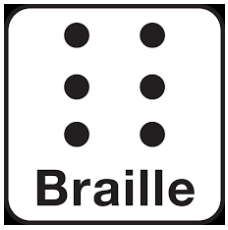
Mediante revisiones documentales, se consultaron diversas fuentes primarias y secundarias para conocer con exactitud los algoritmos que conforman el sistema braille, el cual es considerado mundialmente, como un sistema de lectoescritura, usado por las personas invidentes y los que posee baja visión, siendo su principal medio de comunicación. Desde aquí se comenzó a encontrar un gran volumen de datos (big data), que se tenía que manejar.

Se constató en esta fase que el braille es un conjunto conformado por seis puntos, y que se presentan en relieve. Cada uno un signo, cuyas combinaciones representa palabras. Estos puntos parten de un Signo Generador (SG), donde se ubican con una distribución y tamaño específico, lo que es captado por las terminaciones nerviosas de los dedos, permitiendo su

comprensión de acuerdo con las 64 combinaciones que son posibles realizar.

Se requirió un análisis exacto sobre los puntos que conforman el sistema, estos son pequeños negros, pero al ser escritos para poder comprenderlos se hacen en relieve. Se distribuyen en filas y columnas:

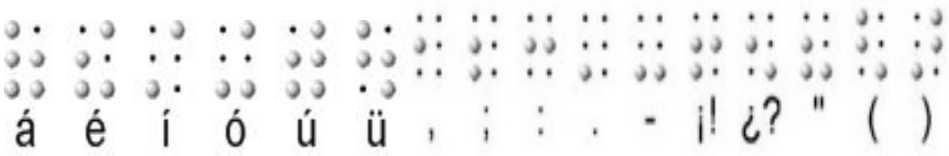
Figura 1 Signo Generador Braille, conformado por seis puntos distribuidos en dos columnas y tres filas



Fuente: (Discapanet.es, 2020)

A partir de este Signo Generador se inician las combinaciones, siendo las principales las correspondientes al alfabeto. Estas combinaciones hacen que el braille sea considerado un sistema binario, lo que genera un gran volumen de datos, llegando a encontrarse 64 caracteres, que cambian su significado de acuerdo al que les antecede o a aquel que les sigue, de allí la big data encontrada como parte de la primera fase del proceso de estudio. Incluso, existen signos para indicar las vocales acentuadas, los signos de puntuación y los números.

Figura 2. Vocales acentuadas y signos de puntuación en Braille



Fuente: (Discapanet.es, 2020)

En esta recolección de información básica se logra determinar que la escritura en braille se representa por la conformación de los signos generadores, es decir, se utiliza como fundamente el SG completo, por cada una de las palabras y letras que se desean transcribir. Así, por ejemplo, cuando se inicia una palabra con mayúscula, se antepone a esta un SG usando solo los puntos 4 y 6.

Fase de análisis de datos

Esta fase inició con la identificación de los patrones de los logaritmos del braille, sus múltiples combinaciones y caracteres creados a partir de estas. Con esto se fue construyendo la información significativa que logra darle cuerpo al software creado. Sin embargo, en el proceso de determinaron algunos inconvenientes que tenían que ser superados con el diseño y programación del software, estos coincidieron con los expuestos en el material bibliográfico consultado.

Entre estos inconvenientes están las diferencias ente las vocales acentuadas y las no acentuadas, por ello su incorporación desde el inicio dentro del aplicativo. Otro de los inconvenientes analizados es que el braille no posee subrayados, epígrafes, tablas, márgenes, y demás caracteres representativos del lenguaje escrito. El análisis develó que se requiere el uso de caracteres especiales para lograr transformar los símbolos relativos a química, física, música y otros idiomas, como el inglés.

Esta diferenciación se resolverá contextualizando lo expresado, lo que permite dar un significado distinto a un mismo signo. En el análisis se detectó que, cuando lo escrito en braille se refiere a música y a informática, el Signo Generador se transforma en un cajetín de 8 puntos, esto permite realizar una mayor cantidad de combinaciones y por tanto de caracteres finales, llegando a ser éstas 256.

Figura 3 Signo Generador Braille de 8 puntos, especialmente diseñado para trascribir y leer temas de música e informática



Fuente: (Martínez & Polo, 2004)

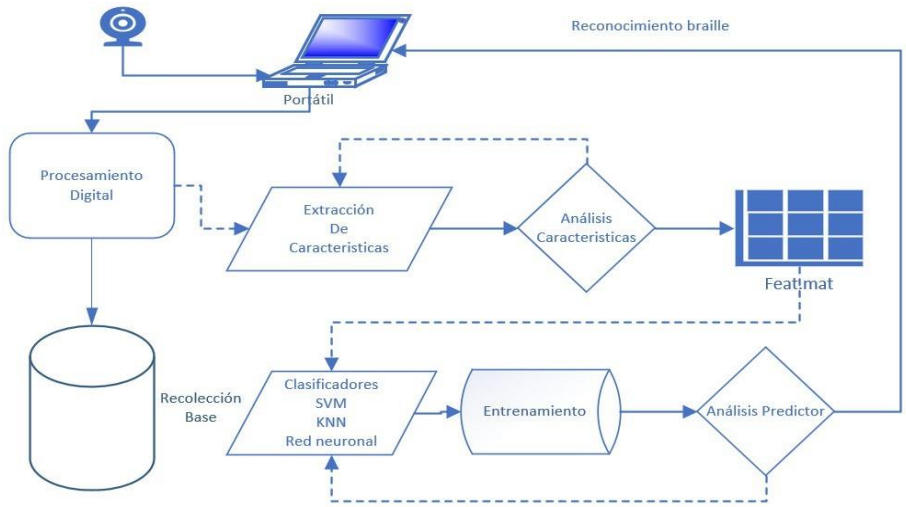
Así mismo, se pudo encontrar que en cada uno de los idiomas existen signos específicos, propios de los idiomas, lo que también es tomado en cuenta para el diseño y programación del software. Por otro lado, se logró determinar que los puntos en relieve del braille con el tiempo y uso del material se van desgastando, por ello que pensó diseñar un software que in situ pueda realizar la lectura del material escrito en este lenguaje. Asimismo, se encontró que la velocidad de la lectura del braille es menor a la que se imprime para leer escrituras en tinta, por tanto, por medio del aplicativo se logrará permitir que los videntes puedan leerlo con mayor rapidez, optimizando el factor tiempo.

Todos los aspectos destacados ayudan a generar las bases con las cuales se procesaron las siguientes fases, siendo primordial entender el braille, sus implicaciones y características.

Fase de Programación.

Esta fase representa el centro innovador del proyecto. Se ejecutó por pasos ver Tabla 1 abarcando los diferentes elementos que se requieren para manejar la big data en el procesamiento y entrenamiento de los datos. Se describen a continuación los nueve pasos que se ejecutaron para alcanzar el desarrollo del software.

Tabla 1 Pasos de programación del software

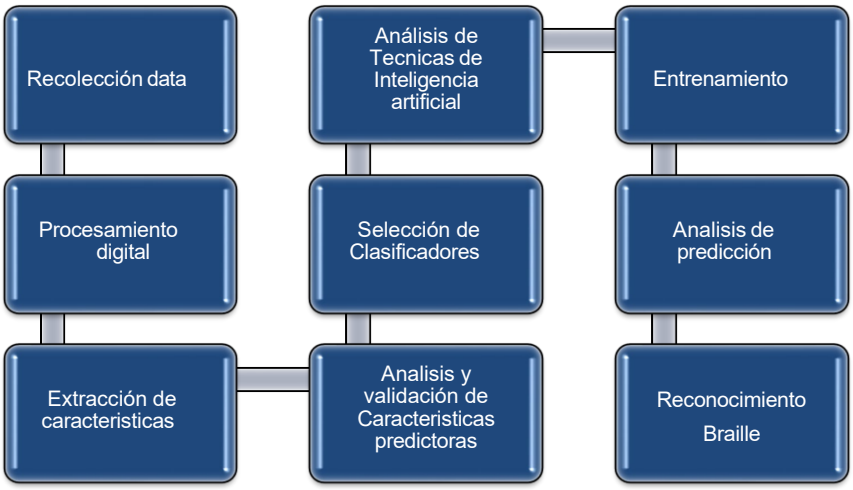


Fuente: Construcción propia

Diseño de la estructura técnico funcional de la interfaz ver Figura 7, aquí se consideran todos

los factores que intervienen en el proceso de análisis, diseño e implementación del proyecto.

Figura 4 Aquí se detalla el diseño de la estructura interfaz gráfica del reconocimiento del lenguaje braille a alfabeto de letras. Fuente: Construcción propia



Fase de ejecución y resultados

El resultado se enfoca en un software traductor de las escrituras en braille al alfabeto de letras, en español. El proceso que genera esta aplicación es.

Primero, se inicia el proceso de recolección de la base de datos, es decir, de los algoritmos del braille que se combinan generando los diferentes caracteres. Este proceso se ejecuta mediante un video el cual capta las imágenes y las almacena en un repositorio denominado "DB" ver Figura 11, adicional tenemos un modelo para entrenar los modelos ver Figura 8.

A partir de la filmación, la aplicación extrae las imágenes y ejecuta el etiquetado de las denominadas clases, en función de lo que corresponda. Es decir, la big data se produce cuando el set de datos crece a medida que se alimenta de imágenes ó se manifiesta con información externa en el formato preestablecido que el sistema requiere para su funcionamiento, el reconocimiento del braille se da en función de la exactitud y precisión de los clasificadores utilizados.

Como última parte de la fase de ejecución y resultados, se procesa el reconocimiento del lenguaje. Esto inicia con la recolección de la información o datos, pero basados en el video

realizado de los escritos en braille que desean transformar. Así, los datos son calibrados generando una edición adecuada de iluminación de la imagen, a lo que sigue el reconocimiento de los símbolos brailicos, y su posterior transformación al alfabeto de letras, en español.

Desarrollo específico de la contribución

Identificación de requisitos

En el contexto universitario, avalado por docentes, se comprobó que los estudiantes invidentes presentan dificultades al momento estregar sus trabajos escritos los cuales, por naturaleza, son transcritos en el lenguaje braille. Estos algoritmos no son de conocimiento de los docentes por lo que se presenta una barrera comunicacional entre ambos grupos de personas. En este sentido, el braille como lenguaje de comunicación táctil, debe ser conocido por todos en su defecto, se deben generar estrategias, técnicas y dispositivos, que permitan su comprensión por parte de los docentes videntes.

En el mercado existen diferentes dispositivos que contribuyen a la comunicación de los estudiantes invidentes, sin embargo, la concepción de estos ha sido establecida de manera unilateral, es decir, son dispositivos que permiten transformar el alfabeto de letras en el sistema braille. También, transformando en audio lo escrito en otros lenguajes diferentes al braille, no detectándose programas eficientes que permitan al lector vidente comprender lo escrito en braille.

El trabajo previo efectuado permitió identificar requisitos para el desarrollo del software, a saber, tecnologías de inteligencia artificial, conocimiento del alfabeto braille, equipos y dispositivos y procesador de lectura. En este sentido, se establecieron como requisitos el MATLAB, como sistema más adecuado; laptop y cámara Web; dispositivo electrónico matriz, para hacer tangible la traducción del alfabeto Braille al de letras. Por otro lado, permitió

desarrollar el software dirigido al ambiento universitario, ya que es uno de los más desasistidos en esta área.

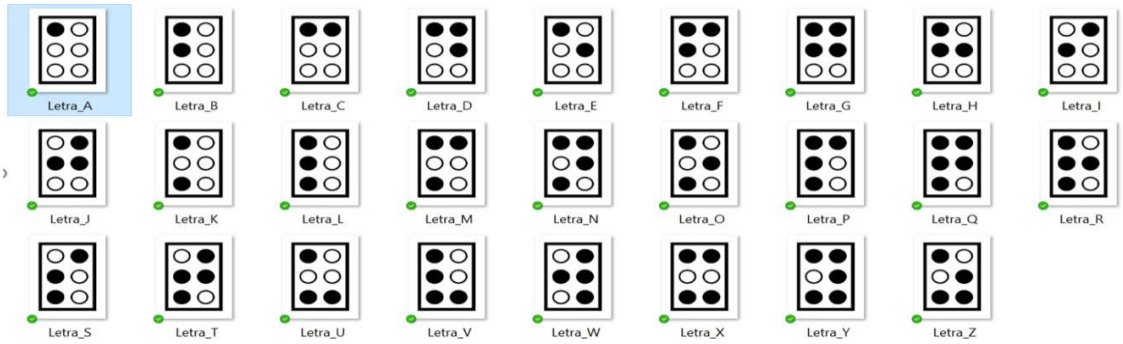
Descripción de la herramienta software desarrollada

A continuación, se presentan de manera detallada cada uno de los pasos efectuados para el desarrollo del software, para lo que se consideraron los hallazgos de la fase de identificación de requisitos.

Recolección de la base de datos

Es la primera etapa del proyecto, consiste en almacenar una base de datos de imágenes símbolos del lenguaje braille en el directorio de MATLAB " DB". Por esta razón hemos seleccionado este modelo de símbolos braille para la detección y reconocimiento luego de haber entrenado a una gran cantidad de imágenes con estos símbolos ver modelo Figura 8.

Figura 5 Modelo de la estructura de los símbolos braille con sus respectivas letras del abecedario que se van a reconocer



Fuente: Construcción propia.

El procedimiento para recopilar las imágenes del lenguaje braille, por medio de la interfaz desarrollada, se da a partir de la lectura a un video que detecta y extrae los objetos que contengan los símbolos braille dentro del rectángulo utilizando herramientas y librerías de visión artificial. Se inició con el grabado de una secuencia de imágenes "Video" en formato "AVI", se le presenta múltiples fotogramas e imágenes de símbolos braille y el sistema por medio de la cámara web capta este video y graba 5 cuadros o imágenes por segundo. El

número de detecciones de imágenes está sujeto a la duración del video. Para este proceso se utilizó como código MATLAB en siguiente ver Códigos 1.

Códigos 1 Aquí se describe el código MATLAB utilizado para el paso recolección de data.

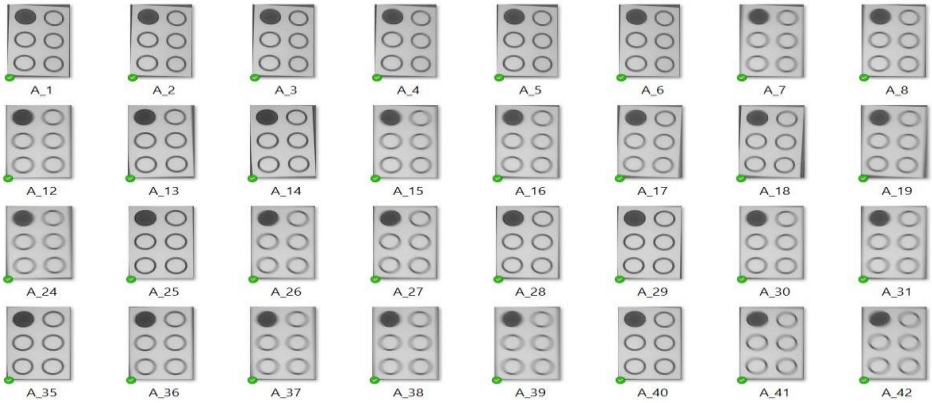
```
% Iniciar webcam
app.cam = webcam(1);
% iniciar video writer
app.v = VideoWriter('video.avi','Uncompressed AVI');
% cuadros por segundo
app.v.FrameRate = 5;
```

Fuente: Construcción propia.

Luego este video se guarda en el directorio o ruta del proyecto MATLAB establecido con formato AVI 'video.avi'.

Ver Figura 11. Las imágenes como se guardan en la base “DB” del directorio de MATLAB.

Figura 6 Modelo de cómo se guarda en la base “DB” por cada símbolo braille



Fuente: Construcción propia.

Paso procesamiento digital

Esta etapa consiste en aplicar el análisis de las diferentes señales desde el emisor, en este caso, desde las imágenes captadas de los símbolos braille, para que puedan ser trasmitidas. En este sentido, la interfaz como receptora, realiza la interpretación de las señales que recibe. En este proyecto se realizaron seis fases para alcanzar el procesamiento:

Crear directorio para almacenar o guardar el archivo. Para ello se utilizó la función en MATLAB “path = "DB/" + name;”.

Luego se realizó una validación, si existe o no la ruta, con un condicionante simple de decisión

ver Códigos 2.

Códigos 2 Se indica el comando de validación utilizó en esta etapa del procesamiento digital

```
if ~exist( path, "dir" )
    mkdir( path );
end
```

Fuente: Construcción propia.

Además, revisa cuantos archivos existen actualmente en esa ruta como una verificación para evitar duplicidad al momento de guardar las imágenes.

“nFiles = length(dir(path)) - 2;”

Esta línea de código nos permite verificar duplicidad en el directorio para almacenar las imágenes braille a partir del vídeo.

Luego damos Inicio de la lectura del video, para esto se utilizó el código para iniciar la lectura del archivo del video que se grabó previamente en MATLAB.

“vid = VideoReader('video.avi');”

La función VideoReader permite dar lectura al vídeo grabado en la interfaz de recolección de datos desde MATLAB.

Posteriormente se lee un cuadro o una imagen del video utilizamos “readFrame”.

“frame = readFrame(vid);”

Se detecta el rectángulo que rodea al símbolo braille en la imagen. Para esta etapa se tiene que realizar la descripción del algoritmo braille ubicado en el recuadro. Así, se realiza la máscara binaria, se transforma a espacio de colores YCbCr, el cuál es un espacio de color utilizado en sistemas de fotografía y video digital. Define el color en términos de un componente de luminancia y dos de crominancia, respectivamente.

Se encuentran en el rango -128 a 127 signado o 0 a 255 no signado” (The MathWorks, 2020).

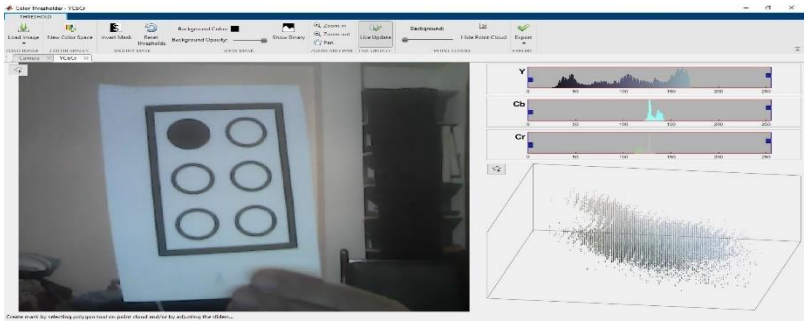
Solamente selecciona los valores de Y dentro de un rango definido por Ymax y Ymin. La

máscara binaria hace referencia al createMask, que no es otra cosa que crear una máscara dentro de la imagen, así “devuelve una máscara del mismo tamaño que la imagen, con “s” dentro del objeto ROI y “s” en todas partes “.himage1h0”. Esta sintaxis es necesaria cuando los ejes que contienen el ROI contienen más de una imagen ver Códigos 3 (The MathWorks, 2020).

Códigos 3 Comandos de máscara binaria para la transformación de los espacios de colores YCbCr

```
function BW = createMask(app,RGB)
            I = rgb2ycbcr(RGB);
            BW = ( I(:,:,1) >= app.YminSpinner.Value ) &...
                ( I(:,:,1) <= app.YmaxSpinner.Value );
        End
```

Figura 7 Binarización convierte los valores de espacio de color RGB en el espacio de color YCbCr. es una matriz -by-3 que contiene los valores de color luminancia () y crominancia (y) como columnas



Fuente: (The MathWorks, 2020).

Seguido a la descripción, se realiza el análisis “hBlobAnalysis” ver Códigos 4. que se encarga de rastrear un conjunto de puntos de la máscara. Esta es “una técnica fundamental de visión artificial basada en el análisis de regiones de imagen consistentes. Como tal, es una herramienta de elección para aplicaciones en las que los objetos que se inspeccionan son claramente discernibles desde el fondo”. (The MathWorks Inc, 2020).

El proceso de visión artificial para (Valdivia, 2016), se convierte en un campo de la inteligencia

artificial, y se ejecuta programando al computador por medio del diseño de detectores que entienden las características que posee una imagen. Esta última es percibida de manera bidimensional y su composición es deducida de manera tridimensional. Bajo este principio se aplica el análisis “hBlobAnalysis”, desde donde se define un área mínima y todo lo que este menor de esa área se descarta.

Códigos 4 Comandos de definición de área mínima para el análisis "hBlobAnalysis"

```
SE = strel('rectangle', app.RectSpinner.Value * [1 1] );
IbwOpen = imopen( BW, SE );

hBlobAnalysis = vision.BlobAnalysis("MinimumBlobArea",
app.AreaSpinner.Value);
[area,~,bboxOut] = step( hBlobAnalysis, IbwOpen );
```

Fuente: (The MathWorks Inc, 2020).

Después de análisis "hBlobAnalysis", se detecta que el rectángulo tenga las proporciones de 3:2, más un cierto rango de tolerancia ver Códigos 5. Esto está definido por Rmax y Rmin.

Códigos 5 Comandos para detectar las proporciones del rectángulo 3:2

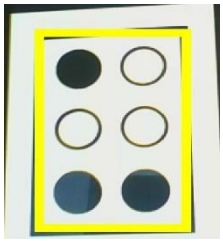
```
HW = double(bboxOut(:,3:4));
            HW_ratio = HW(:,2) ./ HW(:,1);
            area = double(area) ./ HW(:,2) ./ HW(:,1);

            idx = ...
                app.RminSpinner.Value <= HW_ratio &...
                HW_ratio <= app.RmaxSpinner.Value & ...
                0.5 <= area & area <= 0.9;
```

Fuente: Construcción propia.

El Ymax –Ymin, indica el rango basado en el espacio de color YCbCr, que es ampliamente utilizado para el vídeo digital, en este formato. La información de luminancia se almacena como un único componente (Y) y la información de crominancia se almacena como dos componentes de diferencia de color (Cb y Cr). Cb representa la diferencia entre el componente

azul y un valor de referencia. Cr representa la diferencia entre el componente rojo y un valor de referencia. YUV, otro espacio de color ampliamente utilizado para el vídeo digital es muy similar a YCbCr pero no idéntico. Por su parte, Rmax-Rmin indica el rango de la razón entre la cantidad de pixeles verticales y horizontales. Finalmente te retorna todos los rectángulos que cumplan las propiedades previamente dichas o mencionadas, es decir mostramos en color amarillo el objeto detectado ver Figura 10.

Figura 8 Aquí se indica el espacio de color YCbCr y detecta el rectángulo 3:2 de la imagen con el símbolo braille



Fuente: Construcción propia.

En el procesamiento digital se realiza la conversión en escala de grises ver Códigos 6, recortar y guardar todas las colecciones de imágenes definidas por los rectángulos detectados previamente con el algoritmo anterior.

Códigos 6 Comandos para la conversión en escala de grises, recortar y guardar todas las colecciones de imágenes definidas por los rectángulos

```
n = size(bboxOut,1);
for i = 1:n
  nFiles = nFiles + 1;
  I = rgb2gray(frame);
  J = imcrop(I,bboxOut(i,:));
  imwrite( J, path + "/" + name + "_" + nFiles + ".png" );
end
```

Fuente: Construcción propia.

Paso de extracción de características

Esta etapa encierra la identificación de las características propias de las imágenes, estas parten del mapa de bits las cuales están formadas por rejillas de celdas, denominadas píxeles

(Ordoñez, 2005). A cada una de estas rejillas se les asigna un color y también una luminancia específica y propia, por lo que la unión de varias celdas permite percibir los colores de las imágenes como un continuo sólido, a pesar de estar formada por muchas celdas independientes pero ordenadas de manera lógica. En este sentido, los extractores de características, los que se han seleccionado son los Pixeles de la imagen y las características de los patrones locales binarios (LBP).

Para la extracción de los píxeles, se creó una función "T" ver Códigos 7 que extrae los píxeles de las imágenes, básicamente lo que hace este método es obtener una matriz de valores en el rango [0:255] donde 0 es negro y 255 blanco. La obtención de los valores significativos se basa en el método del pixel, el mismo que utiliza parámetros como binarizar la imagen con espacios de color con métodos YCbCr, y para obtener la escala deseada en blanco y negro de la imagen, es decir, se realizó un rescaldado concluyendo como idóneo 54 características de cada imagen 9 filas y 6 columnas, que son guardadas en una tabla en MATLAB.

En este sentido, se establecieron los rangos promedio de los píxeles ver Códigos 7.

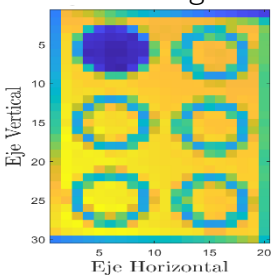
Códigos 7 Línea de código para obtener el promedio de los pixeles en un determinado rango

```
function T = pixelFeat(dataIn,params)
    % Cantidade de pixeles Vertical y Horizontal
    nFeat = [3 2] * params.nFeat;
    % Reescalado de la imagen
    img = imresize(dataIn,nFeat);
    % Tabla de características
    T = table;
    T.feat = img(:)';
End


case "Pixel"
        params.nFeat = 3;
```

Fuente: Construcción propia.

A continuación, se muestra cómo son exactamente los píxeles de las imagen original ver Figura 12, vistos desde la función MATLAB "imagesc(A_1)".

Figura 9 Muestra exactamente como es el pixel de la imagen original utilizando la función MATLAB imagesc

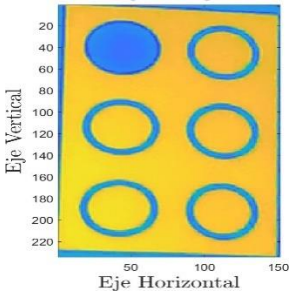


Fuente: Construcción propia.

Una vez realizado el proceso de reescalada, los píxeles se observan de manera diferente se observan 6 cuadrantes con distintos colores y un cuadrante representativo de color azul, le sigue el turquesa, tonos amarillos, naranja y mostaza ver Figura 13.

“imagesc (imresize (A_1, [3.5 2.5]))”.

Figura 10 Descripción imagen reescalada: "Muestra exactamente como es el pixel resscalado de la imagen original utilizando la función MATLAB imagesc (imresize)

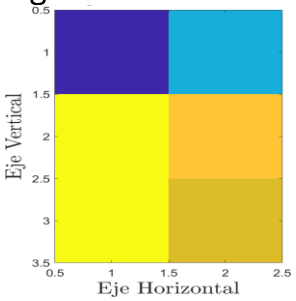


Fuente: Construcción propia.

Se realiza algunos experimentos para destacar los pixeles idóneos una vez realizado el proceso de reescalada, los píxeles se observan de manera diferente se observa una matriz de dimensiones [30X20] cuadrantes con distintos colores y varios cuadrantes representativos de tono azul esto nos indica que a medida que vayamos incrementado el número de pixeles la imagen va retomando la originalidad y para este ejercicio lo que necesitamos es minimizar el consumo de recursos.

“imagesc (imresize (A_1, [30 20]))”.

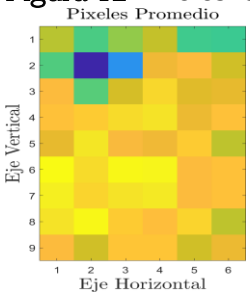
Figura 11 Reescalado 30X20 imagen con distorsión pixelado próximo al original



Fuente: Construcción propia.

Finalmente, luego de experimentar varios ejercicios se determinó el proceso de reescalada con píxeles que permite ser eficientes en la extracción de características de las imágenes almacenas en el set de datos. Esto a su vez se realiza de manera masiva a través del algoritmo del pixel. Ver Figura 15 además, se observa una matriz de dimensiones [9X6] con un total de 54 pixeles por imagen útiles y uno de tono azul en la posición 11 el cual será considerado como punto clave en la extracción, este procedimiento se lo realiza uno a uno para clasificar, entrenar y detectar el símbolo braille a lenguaje de alfabeto de letras. "imagesc (imresize (A_1, [9 6]))".

Figura 12 Pixeles idóneos para extraer características de la imagen



Fuente: Construcción propia.

Se creó una función técnica simple pero efectiva para clasificar objetos, filtra pixeles mediante parámetros y obtenemos valores binarios muy representativos, basada en los patrones locales binarios (LPB). Estos, según (Alvarado & Fernández, 2012), es un "método para el reconocimiento de texturas usado en la detección del movimiento de objetos en tiempo real, detección de caras y el reconocimiento de expresiones faciales con excelentes resultados".

El LPB, utiliza un pixel central como medida de comparación con el resto de los píxeles vecinos, asignándoles a cada uno de estos un valor de acuerdo con su semejanza o diferencia con este. Así, "está definido como una medida invariante a la escala de grises, derivado de una definición general de la textura en una vecindad local" (Alvarado & Fernández, 2012).

Para ello se codifica la relación del píxel central con la intensidad de color de los píxeles aledaños, por medio de la función ver Códigos 8.

Códigos 8 Función para la codificación del pixel central con la intensidad de color de los píxeles aledaños, desde la LPB

```
function T = readLPBfeat(dataIn,params)

    % Reescalado de la imagen
    img = imresize(dataIn,params.size);
    % Extracción de características
    lbpFeatures                                                    =
extractLBPFeatures(img,'CellSize',params.cell,'Normalization','None');
    % Tabla de características
    T = table;
    T.feat = lbpFeatures;
End


case "LPB"
        % Parámetros LPB
        params.size = [ 150 100 ];
        params.cell = [ 32 32 ];

    otherwise
end
```

Fuente: Construcción propia.

Posterior se lee toda la base de datos existente en el directorio "DB" ver Códigos 9 se le aplica la función que extrae las características previamente seleccionadas con el algoritmo "Pixel" ó "LBP" las mismas que son etiquetadas para hacer referencia a clase que corresponden es decir a cada letra braille con su respectiva equivalencia a alfabeto latino.

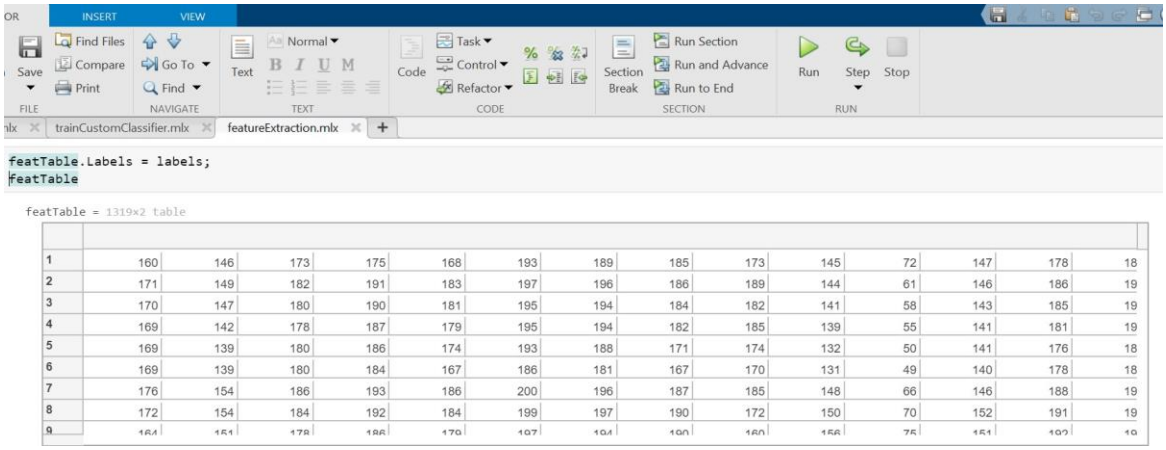
Códigos 9 Función para extraer las características seleccionadas de las imágenes

```
imds                                                                    =
imageDatastore("DB\","IncludeSubfolders",true,"LabelSource","foldernames");
files = imds.Files;
labels = imds.Labels;
% Función de lectura
imds = transform(imds, @(x)featFun(x,params) );
```

Fuente: Construcción propia.

Luego se guardan características en el directorio de MATLAB "featureTable.mat" ver Figura 16. Cada fila es una muestra o una imagen y cada columna es una característica y la última columna es la clase a la que pertenece (supervisado).

Figura 13 Directorio de MATLAB "featureTable.mat", donde cada fila es una muestra o una imagen y cada columna es una característica y la última columna es la clase a la que pertenece.



1	160	146	173	175	168	193	189	185	173	145	72	147	178	18
2	171	149	182	191	183	197	196	186	189	144	61	146	186	19
3	170	147	180	190	181	195	194	184	182	141	58	143	185	19
4	169	142	178	187	179	195	194	182	185	139	55	141	181	19
5	169	139	180	186	174	193	188	171	174	132	50	141	176	18
6	169	139	180	184	167	186	181	167	170	131	49	140	178	18
7	176	154	186	193	186	200	196	187	185	148	66	146	188	19
8	172	154	184	192	184	199	197	190	172	150	70	152	191	19
9	164	151	178	186	179	197	194	190	160	156	75	151	192	19

Fuente: Construcción propia.

Luego se realizó una función que permitió seleccionar cualquiera de los métodos "Pixel" ó "LBP" de extracción de características de las imágenes Códigos 10. En esta etapa lo que se requiere es tener una tabla de características que ayuden como entrada a los clasificadores seleccionados, para ello se parte de la premisa de tomar los clasificadores idóneos, es decir aquellos que basado en algoritmos de técnicas de inteligencia artificial, los modelos permitan clasificar las imágenes de braille, en función de los comandos desarrollados como parámetros únicos de identificación y clasificación de imagen (Heras, 2017).

Códigos 10 Función que permite seleccionar cualquiera de estos métodos de extracción de características de las imágenes

```
function T = featFun( dataIn, params )
    switch params.name
        case "Pixel"
            T = pixelFeat(dataIn,params);
        case "LPB"
            T = readLPBfeat(dataIn,params);
        otherwise
            T = [];
    end
end
```

Fuente: Construcción propia

Una vez establecidos los métodos para extraer los datos de las imágenes realizamos una nueva función con script en vivo con el fin de utilizar cualquiera de estos algoritmos en la extracción de características de toda la base de datos almacenada en "DB" la misma que consta de todas las letras del abecedario braille desde la A-Z. En definitiva utilizo para extraer todas las caracteristicas de todas las 1319 imágenes que tenemos en la base de datos "DB", el mismo que nos da como resultado una tabla de total de valores con los pixeles mas significativos.

4.2.4.- Paso de Entrenamiento

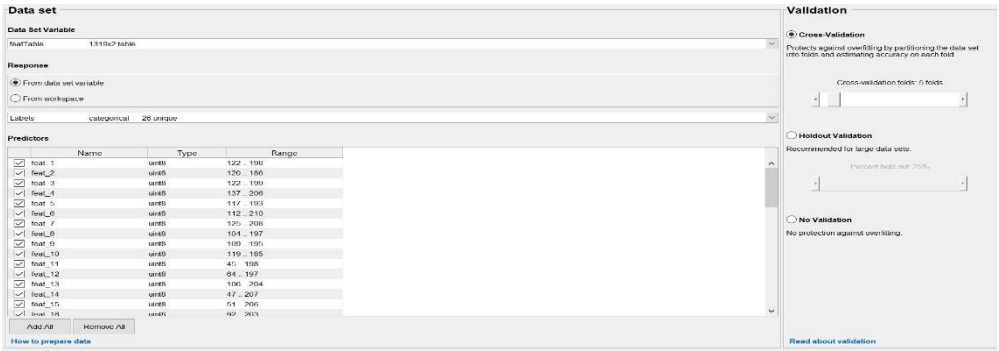
Esta etapa incluye una serie de subprocesos que se explican a continuación.

Selección de clasificadores: Ante la multiplicidad de clasificadores que se encuentran disponibles en el APP de MATLAB, se realizaron las pruebas para determinar su aplicación. A este proceso se le conoce como classification learner, esto según (The MathWorks Inc, 2020), se utiliza para entrenar modelos que son útiles para clasificar datos ver Figura 17, así se puede explorar automáticamente el aprendizaje supervisado, utilizando varios clasificadores. Para (Sancho, 2020), el aprendizaje supervisado se encuentra referido a aquello donde se aprenden funciones, las cuales son enmarcadas a un grupo de ejemplos, por lo que se establece una relación deseada de entrada y salida. Con esta aplicación se logra explorar

esquemas de validación, entrenar módulos, entre otras funciones. Con todo proceso conlleva una serie de etapas, que fueran desarrolladas para alcanzar el diseño del software:

- **Preparación de Datos.** En primer lugar se selecciónó la tabla de caracteristica guardadas previamente "featureTable.mat", ademas se puede observar el tipo de variable que es un unit8(entero de 8 bits).
- **Cross-Validation.-** Se divide el data set en n partes, por ejemplo 5 partes o folds. Para (Amat, 2016a), es un método de resamplins, es decir, de remuestro, que se utiliza para evaluar los diferentes modeles estadísticos. Este se emplea con el fin de estimar el test error rate (tasa de error de prueba), de un determinado modelo en función de evaluar el nivel predictivo de este. Por otro lado, también se utiliza para seleccionar el nivel de flexibilidad adecuado del modelo.
- **Holdout Validation.-** Esta sección es la validación clásica donde se separa un porcentaje del dataset para la validación.
- **No Validation.** Sirve para entrenar con todo el dataset.

Figura 14 Configuración previa y selección de la tabla de características que se utiliza para buscar el clasificador



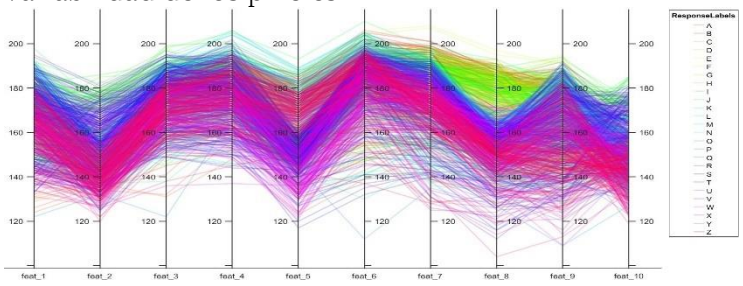
Fuente: Construcción propia.

Análisis de las cacaterísticas: El Data set "featTable" almacena una tabla de 1319 observaciones cuyo tamaño es de 86 kB que corresponden a 54 características obtenidas de las 26 clases etiquetadas en el modelo para aprendizaje supervisado.

Las características las graficamos con el fin de entender como están distribuidos los datos.

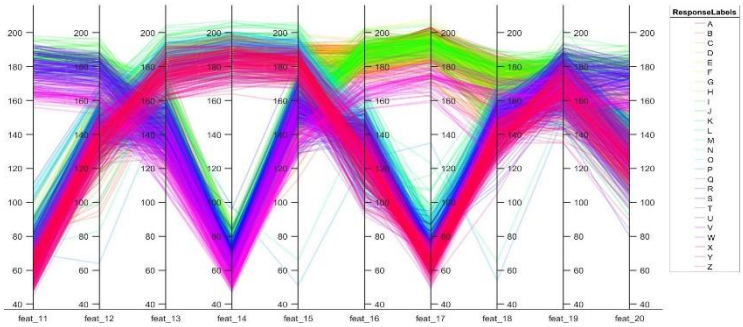
Para esto seleccionamos gráfica “Parallel Coordinates Plot” nos permite visualizar como se componen las características “feat”, clases[A-Z], pixeles [0-255] es así que la Gráfica 1 nos muestra que existe homogeneidad en los datos, aún no se puede observar ni precisar si la información es relevante para utilizarla en su clasificación y entrenamiento. A medida que vamos observando la Gráfica 2 [11-14-17] Gráfica 3 [23-26-29] Gráfica 4 [32- 35-38] Gráfica 5 [41-44] estas cumplen cierta variabilidad con respecto a su eje de pixeles y clases nos indican que el pixel es más obscuro ya que este tiende a [0] y se convierte en un buen candidato para entrada a los clasificadores que seleccionemos posteriormente y también al evaluar las técnicas de inteligencia artificial nativas de MATLAB mediante el “Clasification Learner” permite también evaluar si los “Pixel” ó “LBP” son las que necesitamos, además se puede comparar características mediante las gráficas “Scatter Plot” cuya dimensión es 2 variables.

Gráfica 1 Parallel Plot de las características 1-10 de las clases A-Z, podemos observar la variabilidad de los pixeles



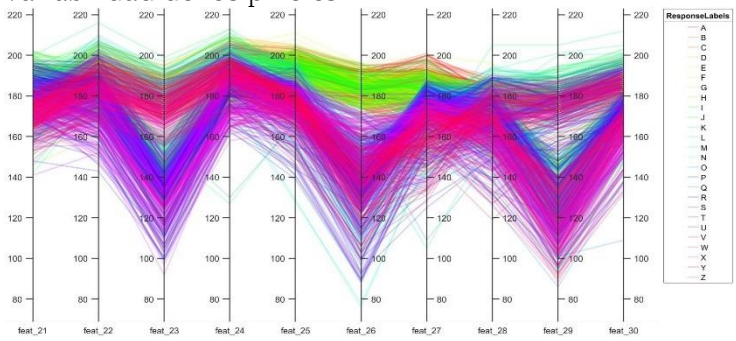
Fuente: Construcción propia

Gráfica 2 Parallel Plot de las características 11-20 de las clases A-Z, podemos observar la variabilidad de los pixeles



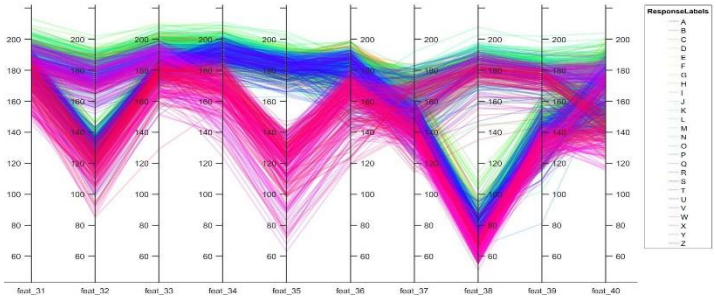
Fuente: Construcción propia.

Gráfica 3 Parallel Plot de las características 21-30 de las clases A-Z, podemos observar la variabilidad de los pixeles



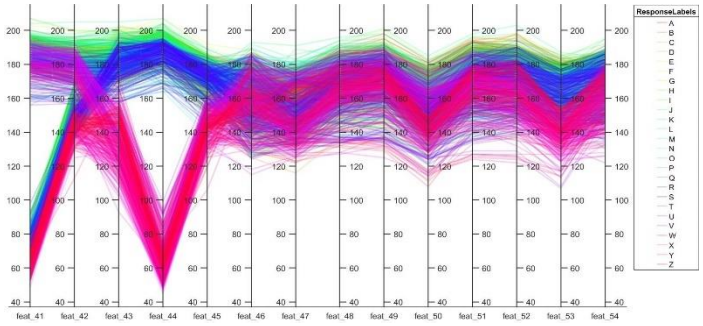
Fuente: Construcción propia.

Gráfica 4 Parallel Plot de las características 31-40 de las clases A-Z, podemos observar la variabilidad de los pixeles



Fuente: Construcción propia.

Gráfica 5 Parallel Plot de las características 41-54 de las clases A-Z, podemos observar la variabilidad de los pixeles.



Fuente: Construcción propia.

Comparativo de las características: Una vez realizado el análisis gráfico del comportamiento de las "feat" se procede a realizar un entrenamiento múltiple, que permite MATLAB por medio del "Clasification Learner" para entender qué clasificador ofrece los mayores porcentajes de precisión, sensibilidad y exactitud se analizaron varios modelos.

Modelo Fine Tree. De acuerdo con (The MathWorks Inc, 2020), este modelo se basa es el diseño de un árbol de clasificación, el cual crece de manera interactiva, usando la aplicación aprendizaje de clasificación. Esto ofrece mayor flexibilidad usando fitctree Códigos 11, ajusta el árbol de decisión binario para la clasificación multiclase, a medida que va creciendo permite predecir las etiquetas y los nuevos datos predictores.

Códigos 11. Devuelve un árbol de decisión de clasificación binaria ajustado y basado en las variables de entrada conocidas como predictores, atributos contenidas en la tabla y la salida.

```
classificationTree = fitctree(...
        predictors, ...
        response, ...
        'SplitCriterion', 'gdi', ...
        'MaxNumSplits', 100, ...
        'Surrogate', 'off', ...
        'ClassNames', unique(response));
```

Fuente: (The MathWorks Inc, 2020)

Las características de este modelo se pueden resumir en:

Tabla 2 Datos cuantitativos y cualitativos del modelo Fine Tree. En esta tabla podemos observar métricas de clasificación en donde seleccionamos el Accuracy del 99,0% mientras más cerca al 100% su predicción es la requerida

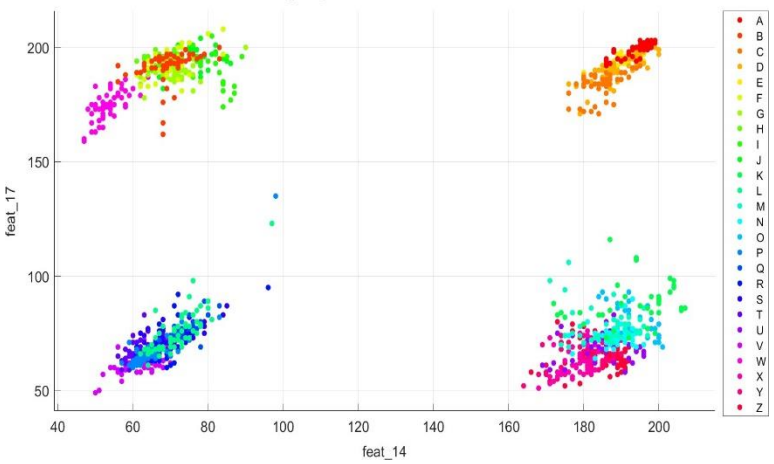
Results	Accuracy:99,0% Total misclassification cost: 13 Prediction speed:3900 obs/sec Training time: 17.534 sec
Model Type	Fine Tree Maximum number of splits: 100 Splits criterion: Gini´s diversity index Surrogate decisión splits: Off
Optimizer options	Hyperparameter options disabled
Feature Selection	Al features used in the model, before PCA
PCA	PCA disabled
Misclassification costs	Cost matrix: default

Fuente: Construcción propia.

Esta Gráfica 6 Agrupación de datos en el Modelo Fine Tree, excelente demostración de variabilidad de pixeles entre dos características. Indica que se acopla al modelo y que las características tendrán una excelente predicción debido a la variabilidad de los datos de los pixeles detectados por medio del procesamiento digital “Pixel” ó “LBP” utilizados, además las

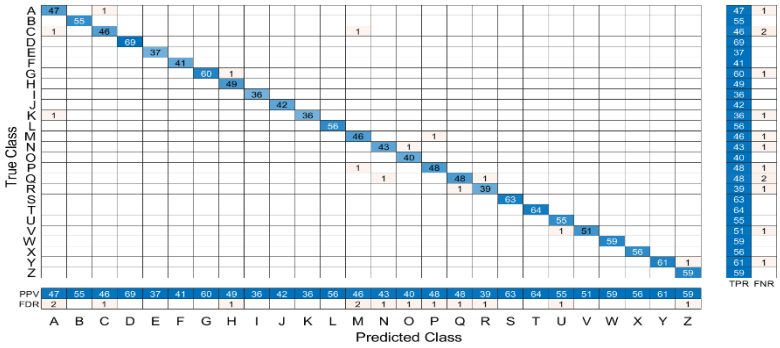
Gráfica 7 nos indica que las características dan síntomas de no detectar pixeles obscuros donde se pueda determinar que exista la puntuación de la imagen con símbolo braille.

Gráfica 6 Agrupación de datos en el Modelo Fine Tree, excelente demostración de variabilidad de pixeles entre dos características



Fuente: Construcción propia.

Gráfica 7 Dispersión de datos en el Modelo Fine Tree, demostración de baja variabilidad de pixeles entre dos características



Fuente: Construcción propia.

Matrices de confusión Fine Tree: Para (Ariza, Alba, & Rodríguez, 2018), las matrices de confusión "son la forma más usual y estándar de informar sobre la exactitud temática de productos derivados de la clasificación de datos procedentes de imágenes". Estas utilizan los índices de porcentaje y el de Kappa, los cuales son globales y sin permitir la manipulación de las categorías ni la creación de condiciones determinadas.

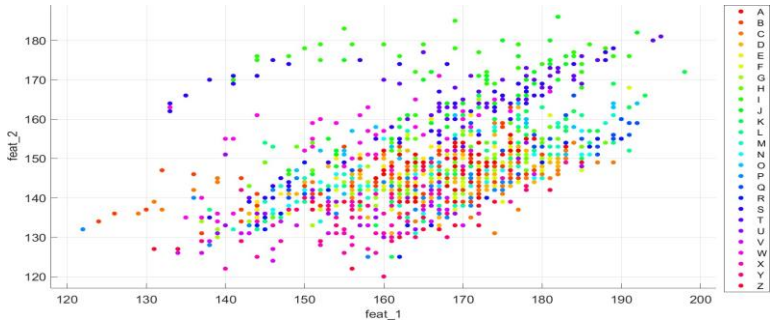
Por otro lado, las matrices de confusión contienen una serie de valores que sirven para contar y determinar el grado de semejanza entre aquellos datos observados de manera pareja. Maneja dos conjuntos de datos, uno llamado conjunto de datos bajo control y el otro llamado,

conjunto de datos de referencia.

El método aplica una distribución multinomial, clase por clase, permitiendo también aproximaciones binomiales a determinadas categorías. No es un método de control no de predicción, basándose en el contraste de los datos reales con los hipotéticos, controlando constantemente el error tipo I, es decir, el riesgo del productor.

Además, es una herramienta que permite visualizar el desempeño de un algoritmo de aprendizaje supervisado, cada columna de la matriz representa el número de predicciones de las clases (símbolos braille en letras del abecedario), mientras que cada fila representa las clases reales o verdaderas es decir prácticamente nos permite observar los aciertos y errores en el modelo de aprendizaje con el set de datos observar Gráfica 8.

Gráfica 8 Matriz de confusión Fine Tree permite observar los aciertos y errores en el modelo de aprendizaje con el set de datos TPR: FNR y PPV: FDR. De acuerdo con esta matriz existen 13 clases[A-C-H-G-M-N-O-P-Q-R-U-Z] que al momento de predecir o reconocer en línea el modelo nos indica habrá errores con menor proporción



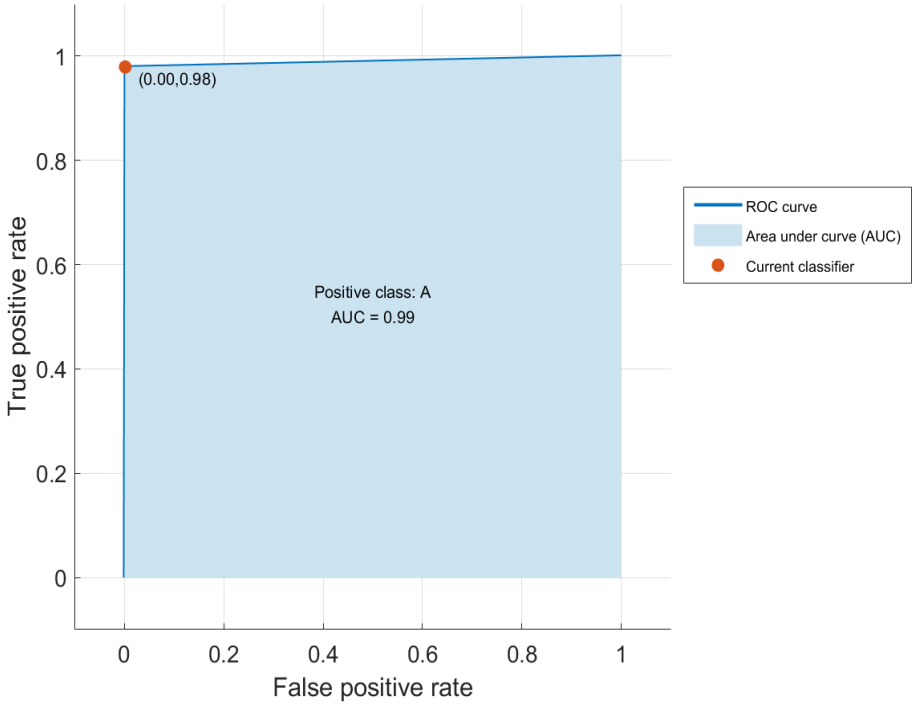
Fuente: Construcción propia.

Curva ROC: De acuerdo con (Benavides, 2017), la curva ROC "representa una especificidad frente a la sensibilidad para cada posible valor umbral o punto de corte en la escala de resultados de la prueba en estudio". Es decir, este modelo estadístico compara la eficiencia de dos algoritmos, con resultado dicotómico, así, se comparan la lectura de las características de las imágenes en braille y se comparan con las predefinidas, dando como resultado la validación o no del símbolo recibido.

(Brownlee, 2018), describe que las curvas ROC, resumen la compensación entre la tasa

positiva verdadera y la tasa positiva falsa para un modelo predictivo que utiliza diferentes umbrales de probabilidad. Por otro lado, estas son apropiadas cuando las observaciones están equilibradas entre cada clase, mientras que las curvas de recuperación de precisión son apropiadas para conjuntos de datos desequilibrados.

Gráfica 9 Cada punto de la curva ROC (círculos rojos) modelo Fine Tree corresponde a un posible punto de corte entre la sensibilidad en el eje (True positive rate) y 1-especificidad en el eje (False positive trate), nos informa la relación de verdaderos y falsos positivos. Además de la proximidad a 1 del área (AUC=0,99) bajo la curva, es una excelente métrica para evaluar un clasificador.



Fuente: Construcción propia.

Modelo Linear Discriminant: Amat (2016b), el análisis discriminante lineal, o sus siglas en inglés LDA, es un método taxonómico supervisado de variables de corte cualitativo. Este procesa a más de un grupo de datos conocidos previamente, junto a nuevos datos, estos últimos son clasificadas de acuerdo con las características que presentan similares a las de los grupos conocidos. Para esto utiliza el teorema de Bayes, de manera que el modelo "estima la probabilidad de que una observación, dado un determinado valor de los predictores, pertenezca a cada una de las clases de la variable cualitativa. Finalmente se asigna la

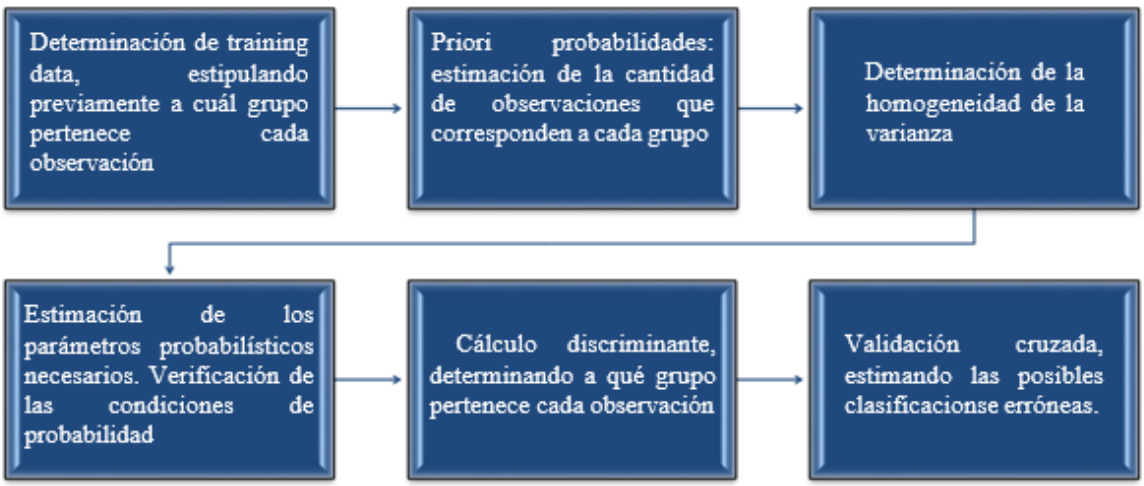
observación a la clase k para la que la probabilidad predicha es mayor" (Amat, 2016b). Es considerado una alternativa ante la presencia de variables cualitativas.

Códigos 12 Línea de código para entrenar con el modelo Linear Discriminant

```
classificationDiscriminant = fitcdiscr(...
    predictors, ...
    response, ...
    'DiscrimType', 'linear', ...
    'Gamma', 0, ...
    'FillCoeffs', 'off', ...
    'ClassNames', unique(response) );
```

Amat (2016b), describe el proceso del modelo LDA, en seis pasos tal y como se efectuó para este proyecto. Observar la técnica nos indica las variables cuantitativas y cualitativas que intervienen, además revisar las Gráfica 10 que muestra agrupación de datos de 2 características comparadas, mientras que la Gráfica 11 muestra dispersión donde indica que esos pixeles no van a predecir el braille, la Gráfica 12 nos muestra la matriz de confusión de este modelo y la Gráfica 13 indica la curva ROC donde podemos darnos cuenta que mientras más cerca de uno el AUC su predicción es al 100%.

Tabla 3 Proceso del modelo linear discriminant



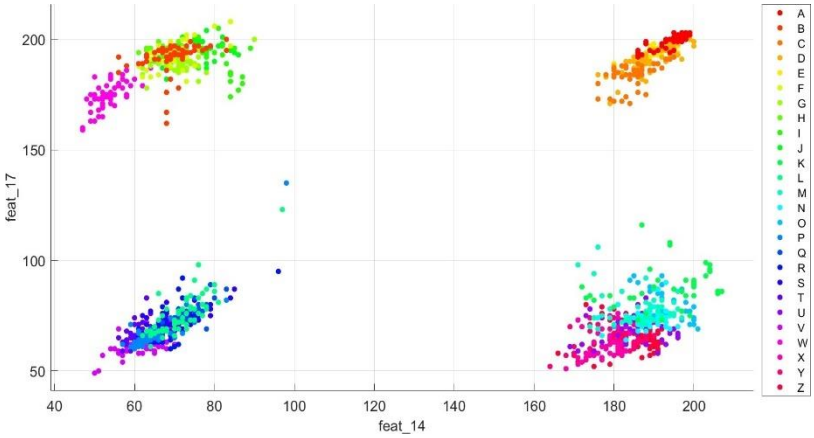
Fuente: Amat(2016b).

Tabla 4 Datos cuantitativos y cualitativos del modelo linear discriminant. en esta tabla podemos observar métricas de clasificación en donde seleccionamos el Accuracy del 100,0% mientras más cerca al 100% su predicción es la requerida

Results	Accuracy:100,0% Total misclassification cost: 0 Prediction speed: 10000 obs/sec Training time: 22.041 sec
Model Type	Linear Discriminant Covariance structure: Full
Optimizer options	Hyperparameter options disabled
Feature Selection	Al features used in the model, before PCA
PCA	PCA disabled
Misclassification costs	Cost matrix: default

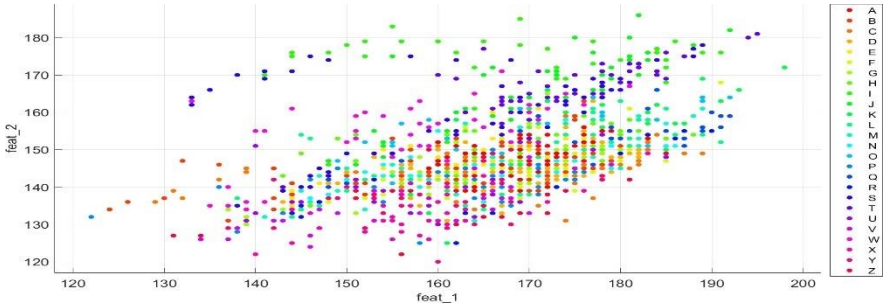
Fuente: Construcción propia.

Gráfica 10 Agrupación de datos en el Modelo linear discriminant, excelente demostración de variabilidad de pixeles entre dos características (feat_14 y feat_17)



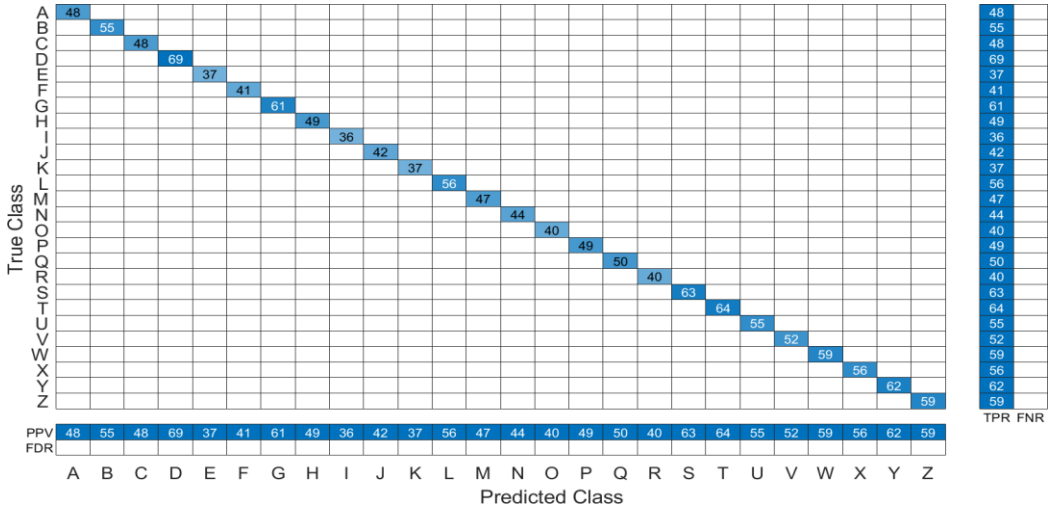
Fuente: Construcción propia.

Gráfica 11. Dispersión de datos en el Modelo linear discriminant, demostración de baja variabilidad de pixeles entre dos características (feat_1 y feat_2)



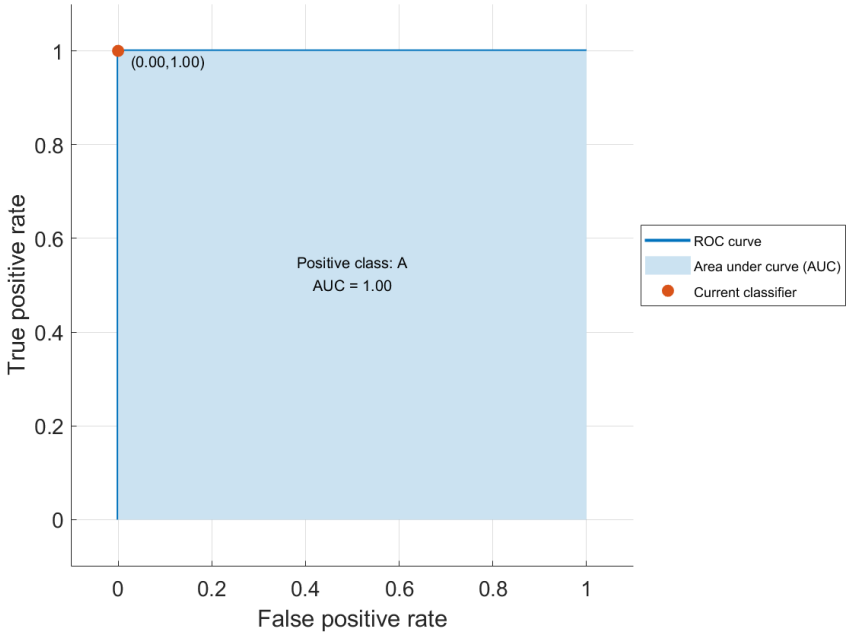
Fuente: Construcción propia

Gráfica 12 Matriz de confusión linear discriminant permite observar los aciertos y errores en el modelo de aprendizaje con el set de datos TPR: FNR y PPV: FDR en esta matriz de confusión su exactitud es al 100% lo cual es una señal de recocimiento braille sin inconvenientes



Fuente: Construcción propia

***Gráfica* 13** Cada punto de la curva ROC (círculos rojos) modelo Fine Tree corresponde a un posible punto de corte entre la sensibilidad en el eje (True positive rate) y 1-especificidad en el eje (False positive trate), nos informa la relación de verdaderos y falsos positivos. Además de la proximidad a 1 del área (AUC=1.00) bajo la curva, es una excelente métrica para evaluar un clasificador.



Fuente: Construcción propia.

Modelo Naive Bayes: Según (Chaparro, Giraldo, & Rodón, 2013), este modelo, como otros, se sustenta principalmente en el teorema de Bayes, este se encuentra referido al cálculo de la probabilidad de la ocurrencia, bajo ciertas condiciones, de un evento X, debido a que ocurrió

un evento Y. Este modelo utiliza los clasificadores bayesianos, los cuales tienen la función de clasificar, bajo supervisión, datos discretos. Esto ha permitido generar diferentes tipos de clasificadores dentro del teorema, lo que depende de la relación de dependencia de las variables, incluyendo a predictivas.

(Chaparro, Giraldo, & Rodón, 2013), describen que este modelo se construye asumiendo a las variables como predictoras y que las mismas son independientes debido a la variable de la clase. (Fernández, 2014), expone que el modelo se representa por medio de un gráfico acíclico, direccionado por cada uno de los nodos, los cuales representan a las diferentes variables. Estas se encuentran unidas por arcos lo que significa que existe una dependencia probabilística. Esto permite presentar de una forma concreta el conocimiento y las estrategias para posibles análisis.

Códigos 13 Línea de código para entrenar las "feat" con la técnica NaiveBayes

```
if any(strcmp(distributionNames,'Kernel'))

     classificationNaiveBayes = fitcnb(...
         predictors, ...
         response, ...
         'Kernel', 'Normal', ...
         'Support', 'Unbounded', ...
         'DistributionNames', distributionNames, ...
         'ClassNames', unique(response) );
 else
     classificationNaiveBayes = fitcnb(...
         predictors, ...
         response, ...
         'DistributionNames', distributionNames, ...
         'ClassNames', unique(response) );
 end
```

Fuente: (The MathWorks Inc, 2020)

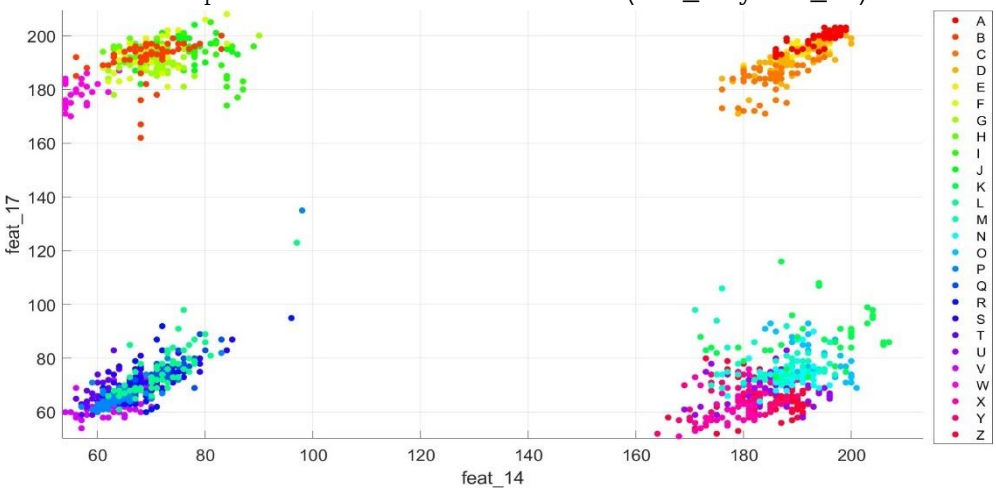
Observar son datos cualitativos y cuantitativos que arroja esta técnica que de igual forma propone dos graficas Gráfica 14 agrupación de puntos comparando 2 características, mientras que la Gráfica 15 manifiesta dispersión de puntos.

Tabla 5 Datos cuantitativos y cualitativos del del modelo Naive Bayes, en esta tabla podemos observar métricas de clasificación en donde seleccionamos el Accuracy del 99,8% mientras más cerca al 100% su predicción es la requerid

Results	Accuracy:99,8% Total misclassification cost: 3 Prediction speed: 150 obs/sec Training time: 123.41 sec
Model Type	Kernel Naive Bayes Distributio name for numeric predictors: Kernel Distribuitor name for categorical predictors: MVMN Kernel type: Gaussian Support: Unbounded
Optimizer options	Hyperparameter options disabled
Feature Selection	Al features used in the model, before PCA
PCA	PCA disabled
Misclassification costs	Cost matrix: default

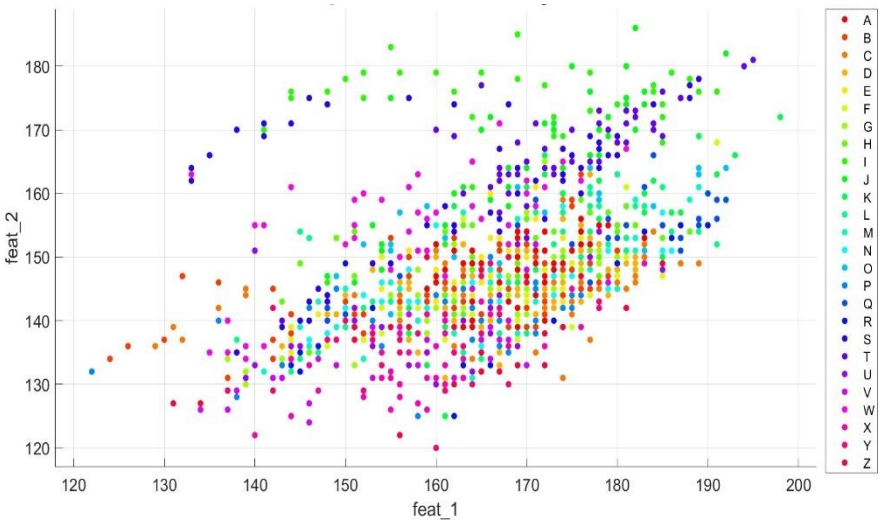
Fuente: Construcción propia.

Gráfica 14 Agrupación de datos en el Modelo Naive Bayes, excelente demostración de variabilidad de pixeles entre dos características (feat_14 y feat_17).



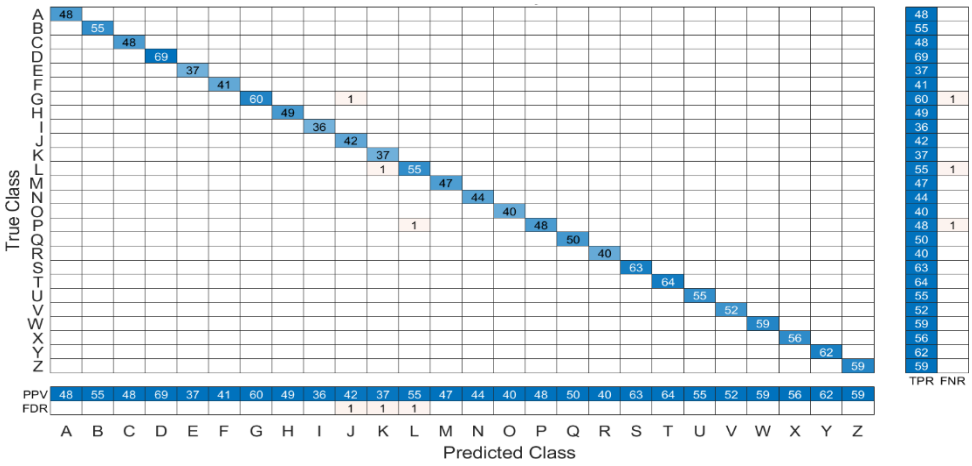
Fuente: Construcción propia.

Gráfica 15 Dispersión de datos en el Modelo Naive Bayes, demostración de baja variabilidad de pixeles entre dos características (feat_1 y feat_2)



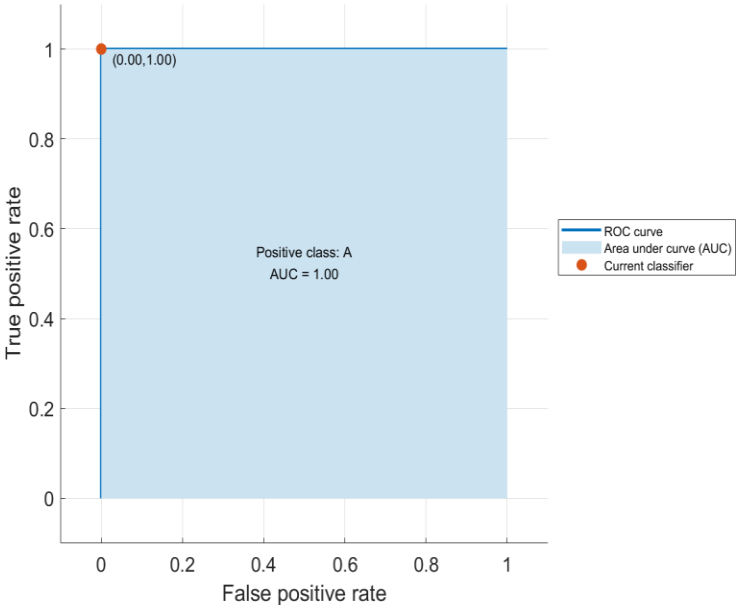
Fuente: Construcción propia.

Gráfica 16 Matriz de confusión Naive Bayes permite observar los aciertos y errores en el modelo de aprendizaje con el set de datos TPR: FNR y PPV: FDR



Fuente: Construcción propia.

Gráfica 17 Cada punto de la curva ROC (círculos rojos) modelo Fine Tree corresponde a un posible punto de corte entre la sensibilidad en el eje (True positive rate) y 1-especificidad en el eje (False positive trate), nos informa la relación de verdaderos y falsos positivos. Además de la proximidad a 1 del área (AUC=1.00) bajo la curva, es una excelente métrica para evaluar un clasificador



Fuente: Construcción propia

Modelo SVM: Para (Maldonado & Weber, 2012), el Support Vector Machines (SVM), es un modelo que logra procesar grandes volúmenes de datos, generalizando frente a aquellos elementos nuevos, así como, la construcción de una serie de funciones no lineales, lo que permite realizar una clasificación o regresión de estos. Entre las características básicas de este modelo están:

Consigue un mejor desempeño predictivo que otras estrategias de filtro y wrapper, debido a su habilidad para ajustarse mejor a los datos, gracias a la medida de desempeño en validación, pero evitando caer en sobreajuste. Presenta un criterio de parada explícito, indicando claramente cuando la eliminación de atributos comienza a afectar negativamente el desempeño del método. Se puede utilizar con cualquier función de kernel. Se puede extender de forma simple a variaciones de SVM, como SVM multiclase, y a otros métodos de clasificación. (Maldonado & Weber, 2012). (Martín, 2016), indica que el procedimiento que se realiza desde el SVM parte de hacer la selección de un hiperplano con una ubicación

equidistante de aquellos considerados como ejemplos aledaños de cada clase. Así se logra establecer el llamado margen máximo del hiperplano, de cada uno de los lados. Otra parte del proceso es que para definir el hiperplano solo son considerados aquellos ejemplos, primero que se establecen en el entrenamiento y segundo, que se encuentran en los márgenes de este.

Códigos 14. Línea de código para entrenar las "feat" con la técnica SVM

```
template = templateSVM(...
    'KernelFunction', 'linear', ...
    'PolynomialOrder', [], ...
    'KernelScale', 'auto', ...
    'BoxConstraint', 1, ...
    'Standardize', true);
classificationSVM = fitcecoc(...
    predictors, ...
    response, ...
    'Learners', template, ...
    'Coding', 'onevsone', ...
    'ClassNames', unique(response) );
```

Fuente: (The MathWorks Inc, 2020)

Observar son datos cualitativos y cuantitativos que arroja esta técnica que de igual forma propone dos graficas Gráfica 18 agrupación de puntos comparando 2 características, mientras que la Gráfica 19 manifiesta dispersión de puntos.

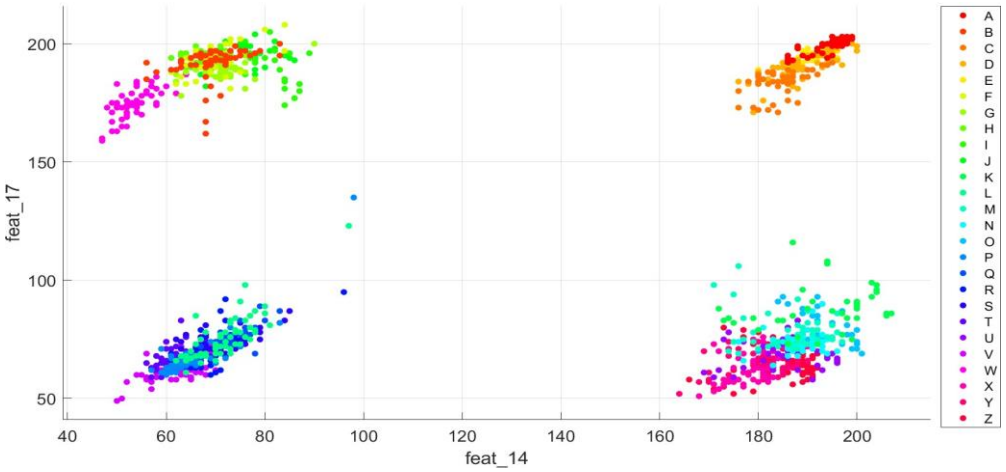
Tabla 6. Datos cuantitativos y cualitativos del del modelo SVM, en esta tabla podemos observar métricas de clasificación en donde seleccionamos el Accuracy del 99,8% mientras más cerca al 100% su predicción es la requerida

Results	Accuracy:100,0% Total misclassification cost: 0 Prediction speed: 630 obs/sec Training time: 76.189 sec
Model Type	Linear SVM Kernel function: Linear Kernel scale: Automatic Box constraint level: 1 Multiclass method: One-vs-One Standardiza data: true
Optimizer options	Hyperparameter options disabled
Feature Selection	Al features used in the model, before PCA

PCA	PCA disabled
Misclassification costs	Cost matrix: default

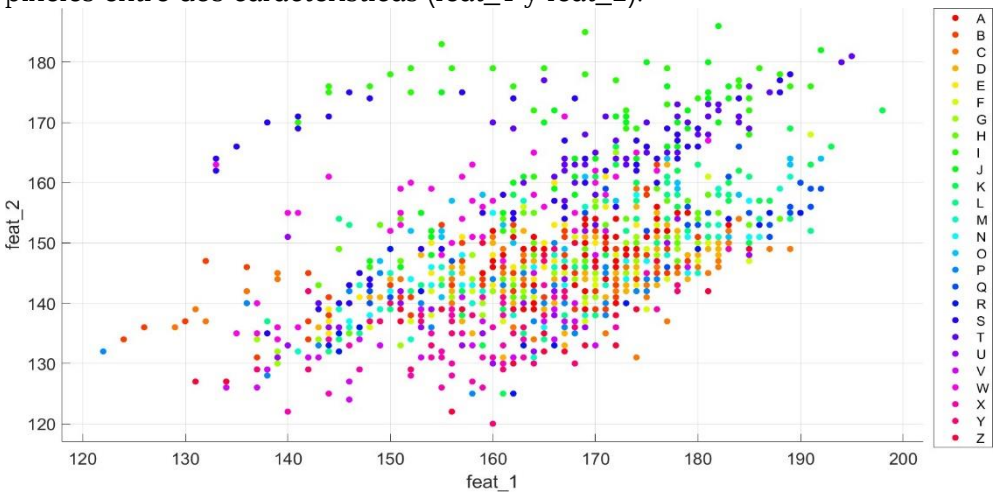
Fuente: Construcción propia.

Gráfica 18 Agrupación de datos en el Modelo SVM, excelente demostración de variabilidad de pixeles entre dos características (feat_14 y feat_17)



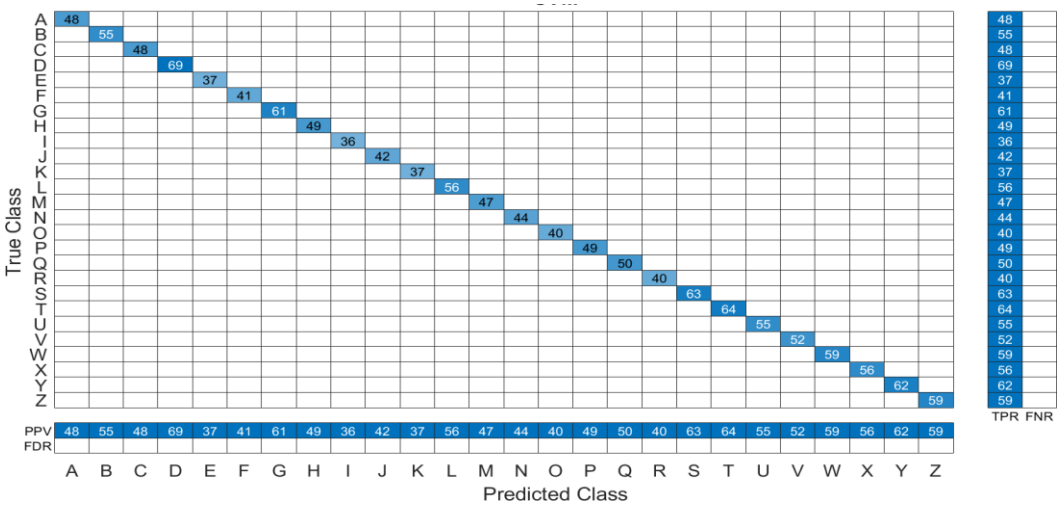
Fuente: Construcción propia.

Gráfica 19 Dispersión de datos en el Modelo SVM, demostración de baja variabilidad de pixeles entre dos características (feat_1 y feat_2).



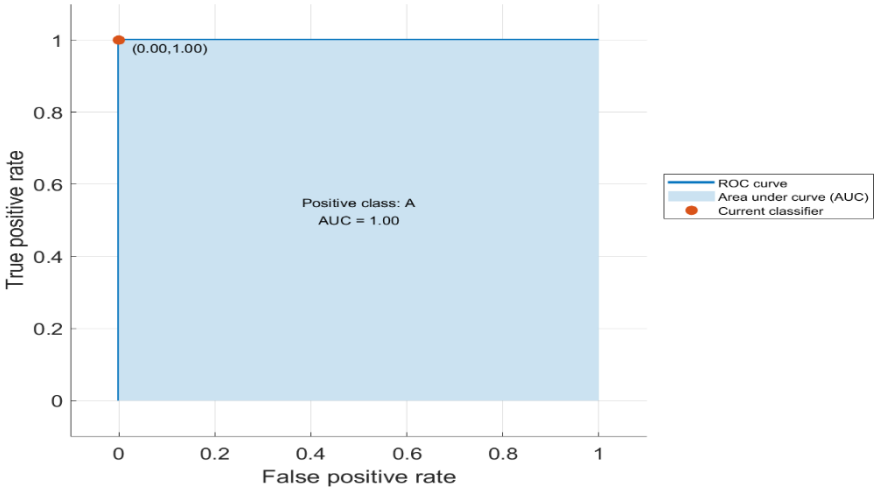
Fuente: Construcción propia.

Gráfica 20. Matriz de confusión SVM permite observar los aciertos y errores en el modelo de aprendizaje con el set de datos TPR: FNR y PPV: FDR. Modelo perfecto sin errores de predicción.



Fuente: Construcción propia.

Gráfica 21 Cada punto de la curva ROC (círculos rojos) modelo SVM corresponde a un posible punto de corte entre la sensibilidad en el eje (True positive rate) y 1- especificidad en el eje (False positive trate), nos informa la relación de verdaderos y falsos positivos. (AUC=1,00) excelente predictor



Fuente: Construcción propia.

Modelo KNN: (Villar & Cuevas, 2015), describen a este modelo como un modelo no paramétrico para realizar clasificaciones estadísticas, las cuales se destacan por ser precisos y confiables, sin tener ningún tipo de dependencia de supuesto distribucional alguno. Las clasificaciones se realizan por medio de algoritmos, a saber, k-NN, k-Nearest Neighbors, en español los k vecinos más cercanos. Sus características teóricas logran garantizar que "su probabilidad de error está acotada por el doble de la probabilidad del error Bayesiano" (Villar

& Cuevas, 2015). De este modo, se asigna un elemento ω, del grupo estudiado y una métrica de distancia. Esta norma logra determinar los KNN ubicados en la base de entrenamiento, asignando a la categoría con mayor cantidad de datos agrupados, un símbolo ω. sí, de acuerdo con (García, Verdú, Larrey, & Morales, 2008), el proceso enunciado por (Villar & Cuevas, 2015), "dado un patrón incompleto x, el algoritmo KNNimpute selecciona los K1 casos más cercanos del conjunto de entrenamiento con valores conocidos en las características a ser imputadas (i.e., características con valores perdidos en x), de tal forma que se minimice una métrica de distancia". En este sentido, al hablar de K1 se hace referencia a los KNN, utilizados para la llamada imputación. Se usa K1 para referir a los K vecinos más cercanos usados para imputación, es decir, para la sustitución de valores. Estos también pueden ser seleccionados de aquellos patrones sin valores perdidos, sin embargo, no es recomendable por autores como (García, Verdú, Larrey, & Morales, 2008). Asimismo, el valor adecuado de los K1 se puede obtener utilizando la validación cruzada (Cross-Validation).

Códigos 15. Línea de código para entrenar con la técnica KNN el vecino más cercano. Fuente: (The MathWorks Inc, 2020)

```
classificationKNN = fitcknn(...
    predictors, ...
    response, ...
    'Distance', 'Euclidean', ...
    'Exponent', [], ...
    'NumNeighbors', 1, ...
    'DistanceWeight', 'Equal', ...
    'Standardize', true, ...
    'ClassNames', unique(response));
```

Observar son datos cualitativos y cuantitativos que arroja esta técnica que de igual forma propone dos gráficas Gráfica 22 agrupación de puntos comparando 2 características que muestran que los datos de las "feat" serán buenos predictores, mientras que la Gráfica 23 manifiesta dispersión de puntos.

Con base a los datos cuantitativos de entrenamiento la técnica KNN es quien realiza por medio

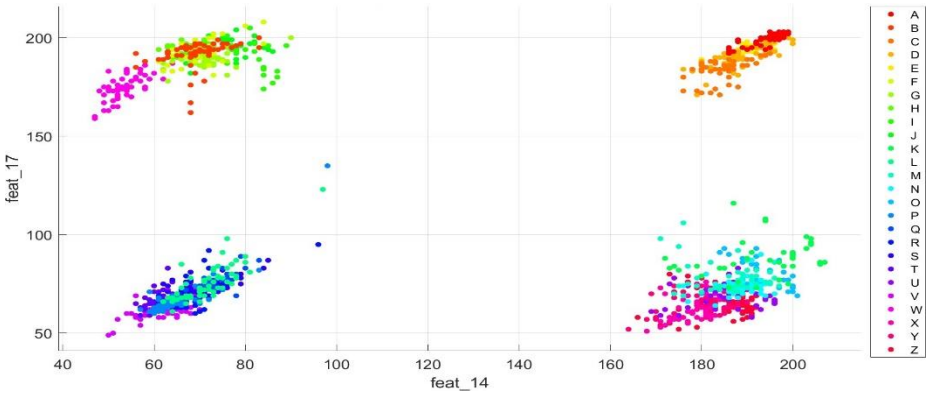
de su algoritmo una alta cantidad de 6700 obs/sec esto en relación con el resto de las técnicas.

Tabla 7 Datos cuantitativos y cualitativos del del modelo SVM, en esta tabla podemos observar métricas de clasificación en donde seleccionamos el Accuracy del 98,9% mientras más cerca al 100% su predicción es la requerida

Results	Accuracy:98,9% Total misclassification cost: 14 Prediction speed: 6700 obs/sec Training time: 199.51 sec
Model Type	Fine KNN Number of neighbors: 1 Distance metric: Euclidean Distance weight: Equal Standardize data: true
Optimizer options	Hyperparameter options disabled
Feature Selection	Al features used in the model, before PCA
PCA	PCA disabled
Misclassification costs	Cost matrix: default

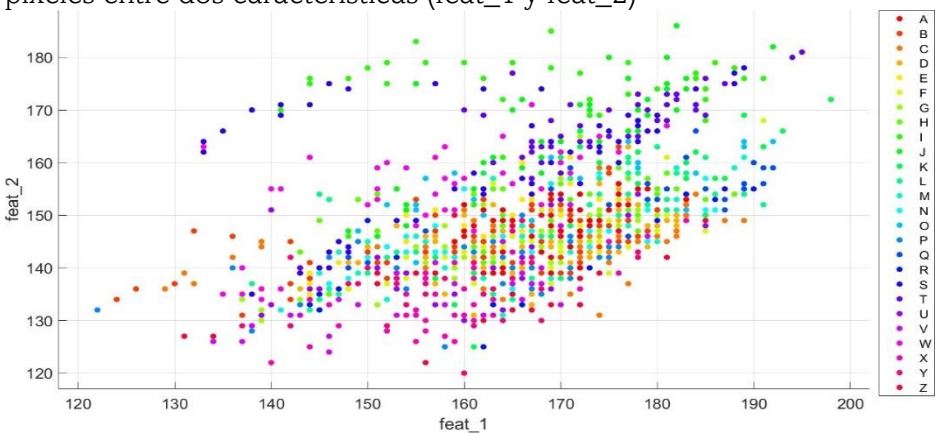
Fuente: Construcción propia.

Gráfica 22 Agrupación de datos en el Modelo KNN, excelente demostración de variabilidad de pixeles entre dos características (feat_14 y feat_17).



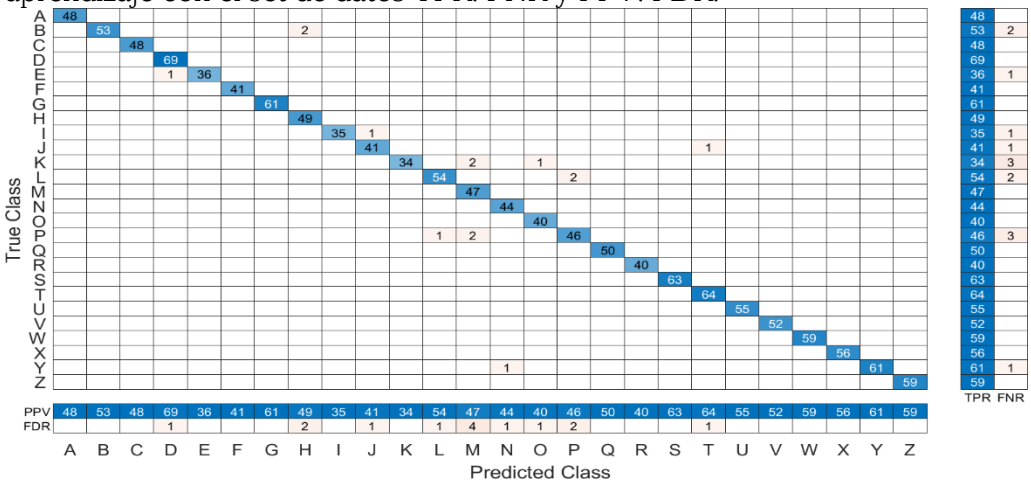
Fuente: Construcción propia.

Gráfica 23. Dispersión de datos en el Modelo KNN, demostración de baja variabilidad de pixeles entre dos características (feat_1 y feat_2)



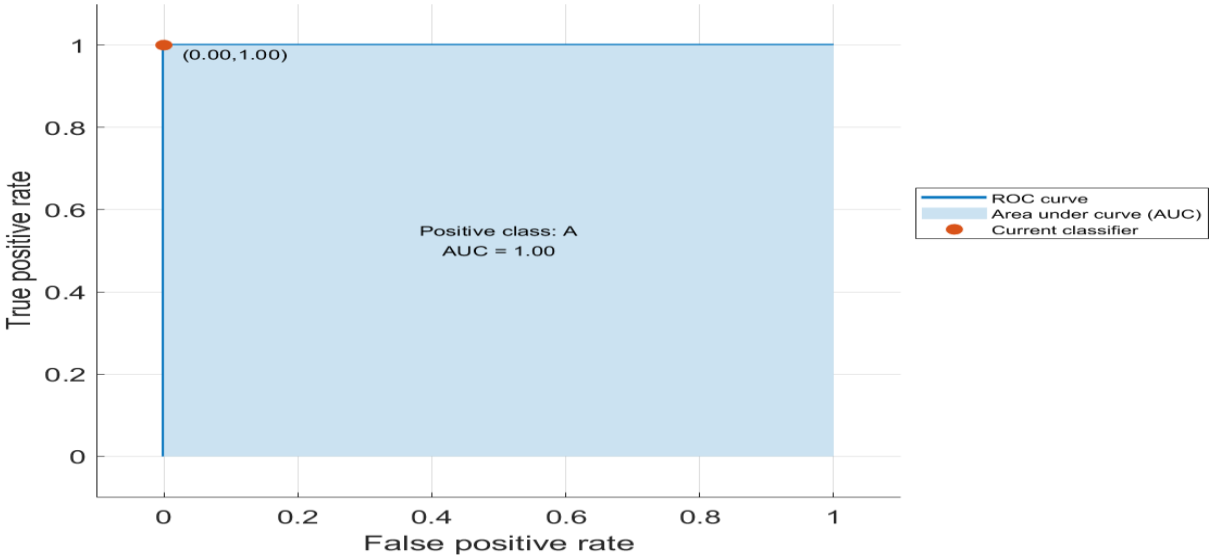
Fuente: Construcción propia

Gráfica 24 Matriz de confusión KNN permite observar los aciertos y errores en el modelo de aprendizaje con el set de datos TPR: FNR y PPV: FDR.



Fuente: Construcción propia.

Gráfica 25 Cada punto de la curva ROC (círculos rojos) modelo KNN corresponde a un posible punto de corte entre la sensibilidad en el eje (True positive rate) y 1- especificidad en el eje (False positive trate), nos informa la relación de verdaderos y falsos positivos.



Fuente: Construcción propia

Modelo Subspace Discriminant: (Radhika & Varadarajan, 2018), señalan que el propósito de

esta clasificación es identificar los nuevos grupos de categorías, para ello utiliza los mapas autoorganizados, a partir de la aplicación de la red neuronal. Para (Basogaín, 2009), la red neuronal, simulando a la red biológica, se encuentra conformada por un conjunto de preceptrones, cada uno de los cuales procesa las entradas y salidas de información, éstas se procesan "normalmente con una suma básica. La suma de las entradas es modificada por una función de transferencia y el valor de la salida de esta función de transferencia se pasa directamente a la salida del elemento procesador".

En este método, según (Radhika & Varadarajan, 2018), realiza el siguiente proceso: se toma a primera imagen RGB, que es "es un modelo de color basado en la síntesis aditiva, el cual permite representar un color mediante la mezcla por adición de los tres colores primarios de luz, rojo, verde y azul" (Rodríguez, 2020). Esta se convierte en Imagen LAB, "es un sistema de color que representa la totalidad del espectro visible." (Rodríguez, 2020), para separar luminancia y crominancia, componentes. Los valores de crominancia se usan para entrenar el mapa auto organizador de la agrupación. Todos los valores A y B han sido modificados en espacio vectorial. El A y B valores más cercanos a interpretación visual, así usando sub-clasificador espacial pueden usar algunas de las mejores características del total de características y discriminar, completando la clasificación para tomar decisiones para diferentes clases. Finalmente, los datos vectoriales se pueden convertir a datos de imagen para fines de visualización; para evitar una clasificación incorrecta de áreas pequeñas y para mejorar, se utiliza el concepto morfológico de precisión de clasificación. El operador morfológico decidirá el valor de la clase. Eso depende del tipo de imagen, así este proceso ha sido hecho en clasificación de cuatro clases y se puede extender a clases superiores también.

Códigos 16 Línea de código para entrenar con la técnica Subspace Discriminant

```
classificationEnsemble = fitcensemble(...
    predictors, ...
    response, ...
    'Method', 'Subspace', ...
    'NumLearningCycles', 30, ...
    'Learners', 'discriminant', ...
    'NPredToSample', subspaceDimension, ...
    'ClassNames', unique(response) );
```

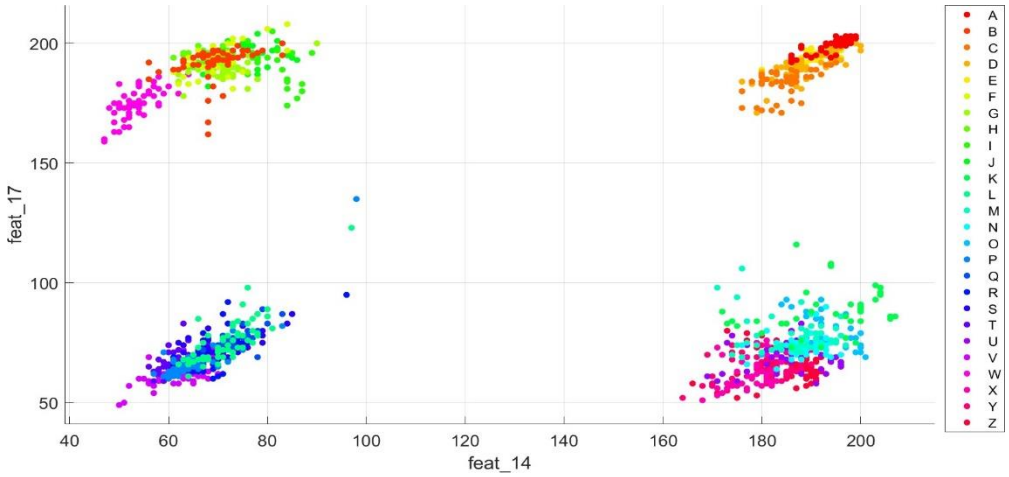
Fuente: (The MathWorks Inc, 2020)

Tabla 8 Datos cuantitativos y cualitativos del del modelo subspace discriminant, en esta tabla podemos observar métricas de clasificación en donde seleccionamos el Accuracy del 98,9% mientras más cerca al 100% su predicción es la requerida

Results	Accuracy:99,9% Total misclassification cost: 1 Prediction speed: 980 obs/sec Training time: 243.86 sec
Model Type	Subspace Discriminant Ensemble method: Subspace Learner type: Discriminant Mimber of learners: 30 Subspace dimensión: 27
Optimizer options	Hyperparameter options disabled
Feature Selection	Al features used in the model, before PCA
PCA	PCA disabled
Misclassification costs	Cost matrix: default

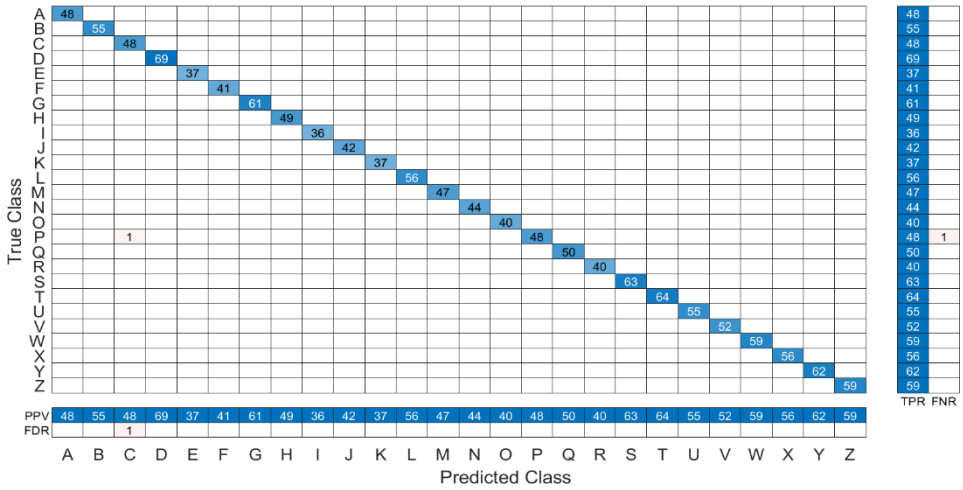
Fuente: Construcción propia.

Gráfica 26. Agrupación de datos en el Modelo subspace discriminant, excelente demostración de variabilidad de pixeles entre dos características (feat_14 y feat_17).



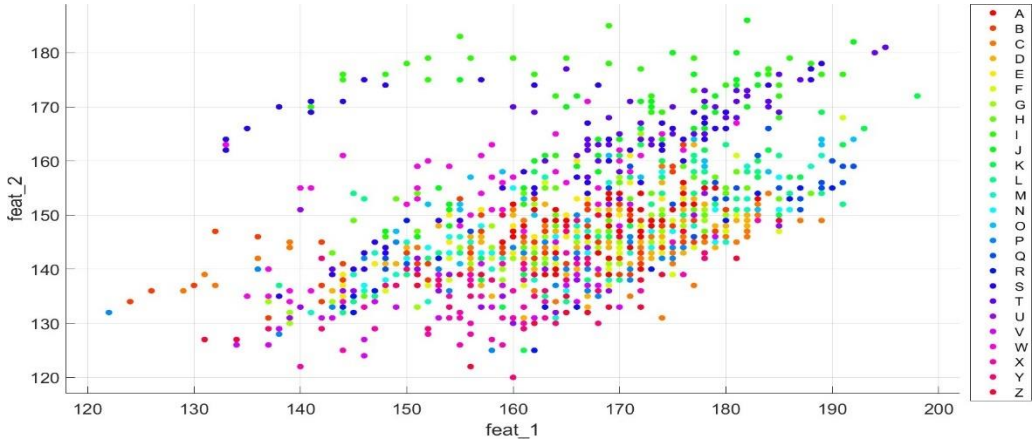
Fuente: Construcción propia.

Gráfica 27 Dispersión de datos en el Modelo subspace discriminant, demostración de baja variabilidad de pixeles entre dos características (feat_1 y feat_2).



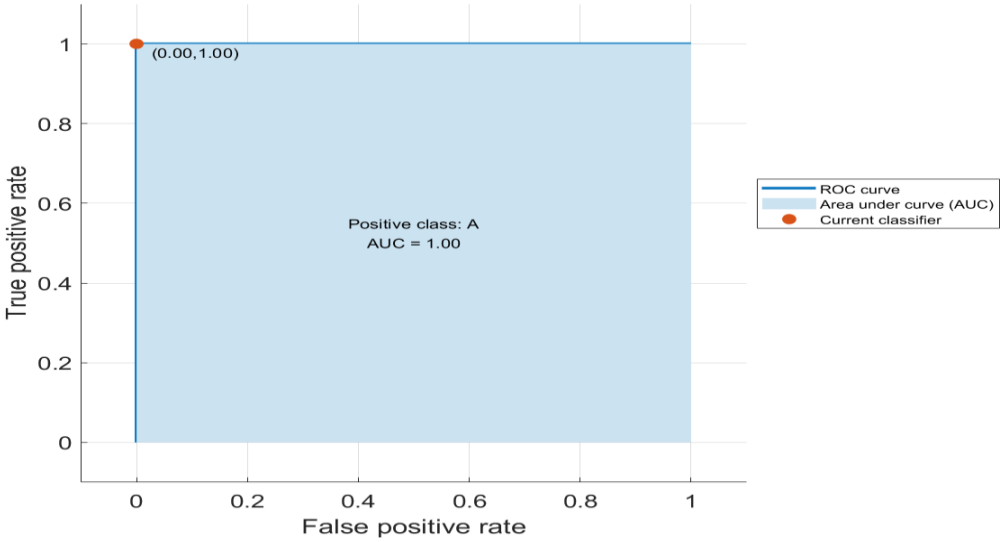
Fuente: Construcción propia.

Gráfica 28 Matriz de confusión subspace discriminant permite observar los aciertos y errores en el modelo de aprendizaje con el set de datos TPR: FNR y PPV: FDR existe una clase que el modelo posible no sea detectada True Class (P) por que el modelo Predicted Class(C)



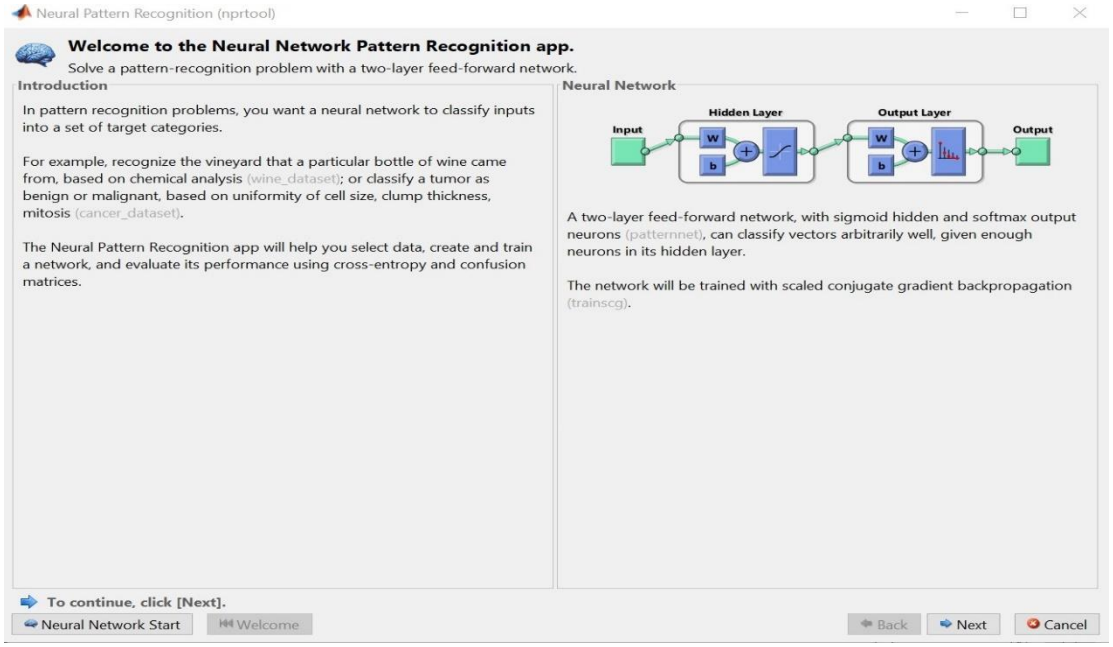
Fuente: Construcción propia.

Gráfica 29 Cada punto de la curva ROC (círculos rojos) modelo subspace discriminant corresponde a un posible punto de corte entre la sensibilidad en el eje (True positive rate) y 1-especificidad en el eje (False positive trate), nos informa la relación de verdaderos y falsos positivos, AUC (1,00) buen predictor.



Redes Neuronales: De acuerdo con (Aldabas & Rubira, 2002) las redes neuronales son las que más utilizadas, son las redes con múltiples capas que funcionan hacia delante. Esta red está compuesta por un conjunto de nodos de entrada que componen la capa de entrada, un conjunto de una o más capas ocultas de neuronas y una capa de neuronas de salida. Los datos de entrada se propaga hacia adelante desde la capa de entrada por la oculta hasta la salida, este tipo de configuración se conoce como MLP o "MultiLayer Perceptrons". También se aplica el proceso "back propagation" que consiste en dos pasadas a través de las diferentes capas de la red, una corrida hacia adelante y una corrida hacia atrás. En la corrida hacia adelante, se aplica en la capa de entrada un patrón o vector de entrada, este propaga su efecto a través de las diferentes capas y como consecuencia produce un vector de salida. Durante este proceso, los pesos sinápticos de la red son fijos y no se modifican.

Figura 18 Configuración Neural Network Pattern Recognition app que nos permite ingresar los parámetros de capas "Entrada-ocultas-Salidas" con 2 nodos y la función de activación logística se obtuvo una exactitud del 99,92% y velocidad de entrenamiento de 13,15 segundos



Fuente: (The MathWorks Inc, 2020)

En esta sección se configura las capas de entrada, ocultas y de salida además de los nodos, se utiliza 2 capas y 10 neuronas cada una, la capa de entra recibe la tabla “featureTable” y como salida recibe las clases esto debido que se realiza un aprendizaje supervisado, MATLAB genera código automático con la App “Neural network Pattern Recogniton” Códigos 17

Códigos 17 Línea de código del App Neural network Pattern Recogniton, entrenamiento-validación-test

```
    % Setup Division of Data for Training, Validation, Testing
    % For a list of all data division functions type: help nndivision
    net.divideFcn = 'dividerand';  % Divide data randomly
    net.divideMode = 'sample';  % Divide up every sample
    net.divideParam.trainRatio = 70/100;
    net.divideParam.valRatio = 15/100;
    net.divideParam.testRatio = 15/100;
    % Train the Network
    net.trainParam.showWindow = false;
    net = train(net,x,t);
    trainedClassifier.net = net;
    trainedClassifier.Labels = Letters;
    trainedClassifier.predictFcn = @(x) predictNN( trainedClassifier, x );
    trainedClassifier.RequiredVariables = {'feat'};
    predictedLabels = trainedClassifier.predictFcn(featTable);
    trueLabels = featTable.Labels;
    validationAccuracy = sum( predictedLabels == trueLabels ) ./ numel(
trueLabels );
    C = confusionmat( trueLabels, predictedLabels );
```

Fuente: (The MathWorks Inc, 2020).

Predicción en Línea: En esta fase se toma una foto, en la que, por medio de los comandos creados, se buscan los rectángulos que envuelven al símbolo braille, una vez que se detectaron los bordes se recorta la imagen de acuerdo con los rectángulos detectados. Luego se extrae las características de cada imagen recortada con el tipo de extractor (Pixeles ó LBP) con las que fueron entrenadas. Todas las características se guardan en una tabla "T" y a estas se le envían al clasificador previamente seleccionado el cual es parámetro de entrada. Guardar video es necesario si se quiere generar la base de datos para el reconocimiento, sino no es necesario.

El clasificador devuelve la letra detectada y finalmente esta aplicación concatena las letras detectadas de acuerdo con la posición donde se detectaron y las une para formar una palabra esto en cuanto el sistema puede reconocer varios símbolos ya que el fin es leer lo que el invidente escribe en lenguaje braille.

Códigos 18. Comandos para la predicción en línea, detecta el objeto braile los señala en un rectángulo extrae características, además nos permite binarizar el video para utilizar los recursos del procesamiento y posterior utiliza la función classifierFun el cual procede a reconocer las imágenes de símbolos braille al alfabeto latino

```
function Ishape = processFrame(app,frame)
[bboxOut,BW] = app.detectForm(frame);
if app.ClasificarCheckBox.Value
   [ Ishape, txt ] = classifierFun(app,frame,BW,bboxOut);
else
   txt = [];
   if app.VistaBinariaCheckBox.Value
       Ishape = insertShape( BW, "Rectangle", bboxOut,...
       "LineWidth", 10 );
  else
      Ishape = insertShape( frame, "Rectangle", bboxOut,...
      "LineWidth", 10 );
  end
end
imshow(Ishape,'Parent',app.UIAxes);
if isempty(txt)
   title( app.UIAxes, '' );
else
   title( app.UIAxes, strjoin(txt) );
end
end
```

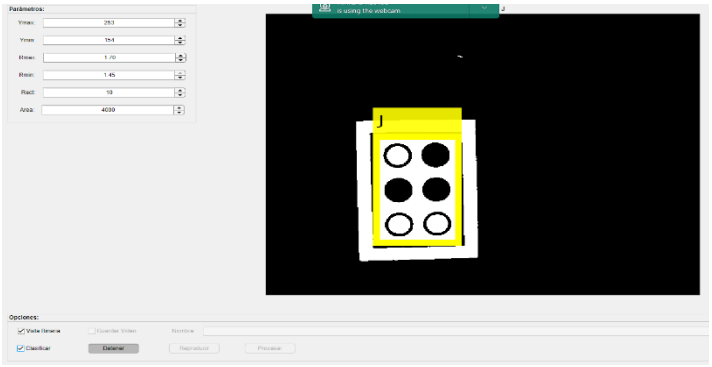
Fuente: Construcción propia.

4.2.5.- Paso de Reconocimiento.

Como última parte de la fase de ejecución y resultados, se procesa el reconocimiento del lenguaje. Esto inicia con la recolección de la información o datos, pero basados en el video realizado de los escritos en braille que desean transformar. Así, los datos son calibrados generando una edición adecuada de iluminación de la imagen aquí interviene la binarización, a lo que sigue el reconocimiento de los símbolos braille, y su posterior transformación al alfabeto de letras ver Figura 19, aquí se puede observar que al mostrar por medio de una imagen recopilada en el celular y esta mostrada a la cámara web del laptop el sistema identifica el objeto mediante un rectángulo color “amarillo”, extrae las características de la letra en braille “J” esta es enviada al modelo del clasificador previamente elegido que puede ser “Fine tree-Linear discriminant-Naive Bayes-SVM-KNN-Subspce discriminant-Redes

neuronales” aquí inicia el proceso de reconocimiento al alfabeto de letras, además con base a los resultados de las matrices de confusión de cada uno de los modelos de entrenamiento utilizados en esta etapa se cumple el reconocer el objeto cabe recalcar que se deben considerar las especificaciones como imagen a detectar, ángulo de inclinación de la imagen expuesta y capacidad de procesamiento de la máquina para soportar MATLAB, en definitiva los resultados no se basan a experimentos de cada letra se lo determina en el entrenamiento y la exactitud de predicción que arroja el modelo ver página de entrenamiento 42.

Figura 19. Interfaz gráfica momento en el cual reconoce el símbolo braille que corresponde a la letra del alfabeto "J", además el sistema está adaptado para reconocer los símbolos que puedan entrar en pantalla y traducirlos al alfabeto de letras



Fuente: Construcción propia.

Evaluación

En esta fase del desarrollo del software, este fue facilitado de docentes universitarios, que cuentan entre sus estudiantes con personas invidentes, algunos con baja visión y otros con ceguera total. Estos realizan el proceso protocolar indicado realizan lecturas del braille, y obtuvieron un reconocimiento total de estas, por lo que su nivel de compresión fue plena.

Para el proceso de evaluación, una vez desarrollado el software, se contó con la participación de docentes. A estos se les permitió usar el dispositivo de captura de imagen de acuerdo con el modelo propuesto, el reconocimiento y transformación al alfabeto de letras fue catalogado como exacto, por parte de los docentes. Estos mostraron aprobación del software exponiendo que contribuirá al desarrollo del proceso de enseñanza-aprendizaje óptimo eliminado las

barreras comunicacionales con sus estudiantes invidentes.

CONCLUSIONES

El proceso dio como resultado desarrollar un modelo de predicción utilizando técnicas de inteligencia artificial y big data para el reconocimiento del lenguaje braille al alfabeto de letras, disminuyendo de forma sustancial la brecha comunicacional entre docentes y alumnos invidentes, gracias a la implementación de tecnología inclusiva en la comunidad educativa.

Se recolectó un set de 1319 imágenes braille de las 26 letras del alfabeto y cada una con 54 características detectadas en el procesamiento digital "Pixel" ó "LBP" obteniendo la tabla "featureTable.mat" utilizada en el entrenamiento de los datos.

Se logró seleccionar los mejores clasificadores que MATLAB and Simulink Student Suite proporciona nativamente con base a su porcentaje de exactitud y velocidad de predicción obteniendo la matriz "trainedModel.mat" que almacena los datos del modelo utilizado en la etapa de reconocimiento.

Fine Tree 99,0%-3900 obs/sec, linear discriminant 100,0%-10000 obs/sec, Naive Bayes 99,8% - 150 obs/sec, SVM 100,0%- 630 obs/sec, KNN 99,0%- 6700 obs/sec, Subspace Discriminant 99,9% - 980 obs/sec, Neural Network 99,92%-13.15 sec.

Se verificó que la interfaz realiza el reconocimiento del símbolo braille a lenguaje de alfabeto de letras en línea, a través de una cámara web y con parámetros específicos de binarización YCbCr se determinó que Ymax debe estar en [253] y Ymin en [154] y a su vez Rmax [1.70] y Rmin [1.45] con un área de objetos no menores a A[4000] son determinantes en el algoritmo de reconocimiento del lenguaje braille ver Figura 19.

Se evaluó la usabilidad con un grupo de docentes que mediante una encuesta el 88% consideran que la solución facilita la enseñanza a los alumnos invidentes en la comunidad educativa del Ecuador, mientras que el 67% se encuentra indecisa y el 33% está dispuesto a

solicitar esta herramientas tecnológicas, además los docentes que desean interactuar con la interfaz son más del 71%lo cual ha generado interés en probar estas soluciones orientadas a ser inclusivos en la comunidad educativa, también indicaron que recomendarían esta solución en otras universidades o colegas más del 71% y el 14% se encuentra indeciso por el momento, finalmente proyecto se lo valora como excelente.

REFERENCIAS BIBLIOGRÁFICAS

Aldabas, E., & Rubira, E. (2002). *Introducción al reconocimiento de patrones*. Barcelona: UPC-Campus Terrassa-DEE-EUETIT Colom. Recuperado de https://d1wqtxts1xzle7.cloudfront.net/46151977/06_reconocimiento_de_patrones.pdf

Alvarado, J., & Fernández, J. (2012). *Análisis de textura en imágenes a escala de grises, utilizando patrones locales binarios (LBP)*. Recuperado de https://www.researchgate.net/publication/303674884

Amat, J. (2016a). *Validación de modelos de regresión: Cross-validation, OneLeaveOut, Bootstrap*. Recuperado de https://www.cienciadedatos.net/documentos/30_cross-validation_oneleaveout_bootstrap

Amat, J. (2016b). *Análisis discriminante lineal*. Recuperado de https://www.cienciadedatos.net/documentos/28_linear_discriminant_analysis_lda_y_quadratic_discriminant_analysis_qda

Ariza, F., Alba, V., & Rodríguez, J. (2018). *Control estricto de matrices de confusión por medio de distribuciones multinomiales*. Recuperado de https://www.researchgate.net/publication/326718169

Basogaín, X. (2009). *Redes neuronales artificiales y sus aplicaciones*. Recuperado de https://ocw.ehu.eus/pluginfile.php/9047/mod_resource/content/1/redes_neuro/c

ontenidos/pdf/libro-del-curso.pdf

Benavides, R. (2017). *Curvas ROC (Receiver-Operating-Characteristic) y sus aplicaciones*. Universidad de Sevilla. Recuperado de https://idus.us.es/bitstream/handle/11441/63201

Brownlee, J. (2018, agosto 31). *Cómo utilizar curvas ROC y curvas de recuperación de precisión para la clasificación en Python*. Recuperado de https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/

Chaparro, J., Giraldo, B., & Rodón, S. (2013). Evaluación del clasificador Naive Bayes como herramienta de diagnóstico en unidades de cuidados intensivos. *Revista de Tecnología / Journal Technology*, 87–93. Recuperado de https://www.researchgate.net/publication/294724781

CONADIS. (2019). *Estadísticas de discapacidad*. Recuperado de https://www.consejodiscapacidades.gob.ec/estadisticas-de-discapacidad/

Di Tulio, N. (2017, noviembre 30). *Braileo: una app que apuesta a la inclusión*. Educar. Recuperado de http://globalbackend.educ.ar/sitios/educar/recursos/ver?id=131881

Discapanet.es. (2020). *El alfabeto braille*. Recuperado de https://www.discapnet.es/areas-tematicas/tecnologia-inclusiva/productos-de-apoyo/guias-tecnicas/el-alfabeto-braille

Escobar, A., et al. (2014). *Análisis comparativo de técnicas de accesibilidad para personas ciegas*. Recuperado de http://www.esvial.org/wp-content/files/CAFVIR2014pp187-195.pdf

Fernández, E. (2014). *Análisis de clasificadores bayesianos*. Universidad de Buenos Aires. Recuperado de http://materias.fi.uba.ar/7550/clasificadores-bayesianos.pdf

García, P., Verdú, R., Larrey, J., & Morales, J. (2008). *Algoritmo KNN basado en información mutua para clasificación de patrones con valores perdidos*. Recuperado de https://www.researchgate.net/publication/260909129

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

IT User. (2017, agosto 7). *La Inteligencia Artificial ayuda a las personas con discapacidad visual*. Recuperado de https://www.ituser.es/it-user/2017/08/la-inteligencia-artificial-ayuda-a-las-personas-con-discapacidad-visual

Lucas, M. (2018). *Tecnología en el aula de educación musical con alumnos ciegos: Proceso de ida y vuelta* (Tesis de maestría). Universidad Internacional de Valencia. Recuperado de https://www.mercedeslucas.com/images/Tecnologia_en_el_aula_de_educacion_musical.pdf

Maldonado, S., & Weber, R. (2012). Modelos de selección de atributos para Support Vector Machines. *Revista Ingeniería de Sistemas*, 26, 49–70. Recuperado de http://www.dii.uchile.cl/~ris/RISXXVI/maldonado1.pdf

Martín, J. (2016). *Support Vector Regression: Propiedades y aplicaciones*. Universidad de Sevilla. Recuperado de https://idus.us.es/bitstream/handle/11441/43808

Martínez, A., & Polo, J. (2004). *Guía didáctica para la lectoescritura braille*. Madrid: Organización Nacional de Ciegos Españoles (ONCE).

Merino, M. (2019, marzo 13). Seeing AI, la app de Microsoft que va describiendo el contenido de las fotos a las personas ciegas. *Xataka*. Recuperado de https://www.xataka.com/inteligencia-artificial/seeing-ai-app-microsoft

Microsoft. (2020, mayo 21). *Seeing AI*. Recuperado de https://www.microsoft.com/en-

us/ai/seeing-ai

Ojeda, J., & Bolaños, J. (2015). *Uso de representaciones semióticas en el aprendizaje de operaciones básicas en aulas con integración de población ciega* (Trabajo de grado). Universidad Distrital Francisco José de Caldas. Recuperado de http://repository.udistrital.edu.co/bitstream/11349/2407

ONU. (2011, diciembre 3). *Día Internacional de las Personas con Discapacidad*. Naciones Unidas. Recuperado de https://www.un.org/es/events/disabilitiesday/2011/

Ordoñez, C. (2005). Formatos de imagen digital. *Revista Digital Universitaria, 6*(5). Recuperado de http://www.revista.unam.mx/vol.6/num5/art50/may_art50.pdf

Radhika, K., & Varadarajan, S. (2018). Ensemble subspace discriminant classification of satellite images. *Journal of Scientific & Industrial Research, 77*(11), 633–638. Recuperado de http://nopr.niscair.res.in/bitstream/123456789/45383/1/JSIR%2077%2811%29%20633-638.pdf

Rodríguez, J. (2020). *Modelos reproductivos del color: RGB, CMYK y LAB*. Recuperado de https://www.logo-arte.com/blog-6.htm

Sancho, F. (2020). *Aprendizaje supervisado y no supervisado*. Recuperado de http://www.cs.us.es/~fsancho/?e=77

Servimedia. (2020). *Tecnología accesible e inclusión*. Recuperado de https://www.servimedia.es/noticias/1135159

The MathWorks Inc. (2020). *Classification Learner App: Train classification models*. Recuperado de https://es.mathworks.com/help/stats/train-classification-models-in-classification-learner-app.html

The MathWorks Inc. (2020). *Vision Toolbox: Blob analysis system object*. Recuperado de

https://es.mathworks.com/help/vision/ref/vision.blobanalysis-system-object.html

The MathWorks Inc. (2020). *MATLAB software*. Recuperado de https://es.mathworks.com/products/matlab.html

Todo Noticias. (2018, abril 9). *Braileo: la app que traduce el lenguaje braille al español*. Recuperado de https://tn.com.ar/salud/lo-ultimo/braileo-la-app-que-traduce-el-lenguaje-braille-al-espanol_861753

Universia. (2016, diciembre 15). *Braileo: una app para interpretar textos en braille creada por estudiantes de la UTN*. Recuperado de https://noticias.universia.com.ar/educacion/noticia/2016/12/15/1147519/braileo-app-interpretar-textos-braille-creada-estudiantes-utn.html

Universidad Internacional de Valencia. (2020). *Cómo funciona el software de reconocimiento de imágenes*. Recuperado de https://www.universidadviu.com/reconocimiento-de-imagenes-software-y-ejemplos/

Valdivia, C. (2016). *Diseño de un sistema de visión artificial para la clasificación de chirimoyas basado en medidas*. Pontificia Universidad Católica del Perú. Recuperado de https://pdfs.semanticscholar.org/74ef/fba2ced3d12bfdbcb1078b4987827ab00319.pdf

Villar, C., & Cuevas, C. (2015). Condensación controlada en k-nn y su aplicación para la identificación del color en tiempo real. *Revista de Matemática, 23*(1), 143–154. Recuperado de https://www.scielo.sa.cr/pdf/rmta/v23n1/1409-2433-rmta-23-01-00143.pdf