Tratamiento de Imágenes

Aprendizaje de maquinas

Nombre: Luis Fernando Orzoco
Grupo: 4
Tutora: Sandra García

***Resumen*— Este documento detalla el proceso de implementación de clasificadores de aprendizaje de máquinas para la identificación de objetos en imágenes, en cumplimiento con la Etapa 3 del curso de Tratamiento de Imágenes (208054). El proyecto se enfoca en la extracción de características morfológicas (centroide en X, centroide en Y, y circularidad) de un conjunto de imágenes de tarjetas de parqueadero (identificando vehículos eléctricos vs. no eléctricos) utilizando MATLAB R2021b o superior. Se implementa y entrena una Máquina de Vectores de Soporte (SVM) con un kernel lineal utilizando los datos etiquetados manualmente de la carpeta Entrenamiento. Posteriormente, se evalúa el rendimiento de la SVM en un conjunto de datos de Prueba también etiquetado manualmente. Adicionalmente, se desarrolla y evalúa un clasificador por umbral (perceptrón simple) utilizando las características de centroide en Y y circularidad, comparando su precisión (42.86%) con la del modelo SVM (100% en este dataset), destacando las limitaciones de los clasificadores lineales simples frente a la posible superposición de datos en el espacio de características.**

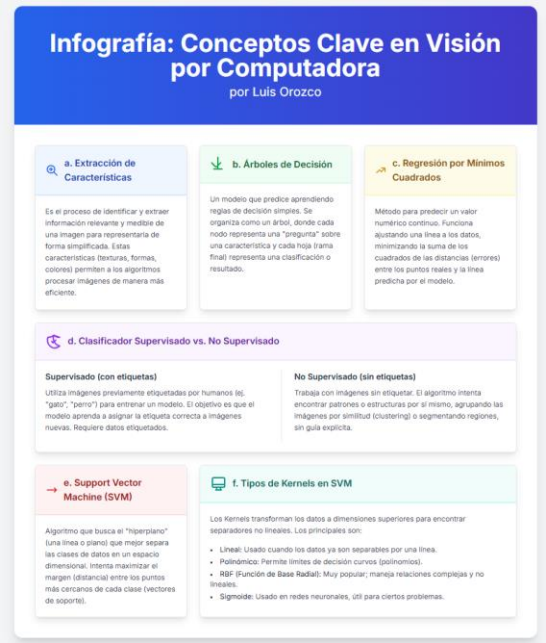
***Abstract* This document details the implementation process of machine learning classifiers for object identification in images, in compliance with Stage 3 of the Image Processing course (208054). The project focuses on extracting morphological features (X-axis centroid, Y-axis centroid, and circularity) from a set of parking card images (identifying electric vs. non-electric vehicles) using MATLAB R2021b or later. A Support Vector Machine (SVM) with a linear kernel is implemented and trained using the manually labeled data in the Training folder. Subsequently, the SVM's performance is evaluated on a manually labeled Test dataset. Additionally, a threshold classifier (simple perceptron) is developed and evaluated using the Y-axis centroid and circularity features, comparing its accuracy (42.86%) with that of the SVM model (100% in this dataset), highlighting the limitations of simple linear classifiers regarding potential data overlap in the feature space.**

I Introducción

El propósito de este informe es documentar el desarrollo sistemático de la Etapa 3, enfocada en la aplicación práctica del Aprendizaje de Máquinas dentro del curso de Tratamiento de Imágenes (208054), cumpliendo rigurosamente con los lineamientos establecidos en la guía de actividades correspondiente. El objetivo fundamental de esta etapa es aplicar los conceptos teóricos del procesamiento digital de imágenes y las técnicas de inteligencia artificial para entrenar y evaluar clasificadores supervisados, tal como lo estipula el Resultado de Aprendizaje 2 (RAP 2) del curso: "Examinar las propiedades principales de los objetos identificados en las imágenes, considerando características como color, área, número de elementos y texturas, con el objetivo de entrenar clasificadores supervisados". Esta fase representa un componente práctico crucial, donde se traducen los fundamentos teóricos en soluciones computacionales concretas capaces de realizar tareas de clasificación binaria sobre datos visuales. Para alcanzar dicho objetivo, se utiliza como herramienta principal el software MATLAB, específicamente su robusto 'Image Processing Toolbox' y el 'Statistics and Machine Learning Toolbox', entornos que proporcionan las funciones necesarias para la manipulación de imágenes, la extracción de descriptores cuantitativos y la implementación de algoritgos de clasificación. El proyecto sigue de manera estructurada las instrucciones detalladas en el Anexo 1 - Implementación de códigos en el software - Etapa 3 , documento que guía al estudiante a través de cuatro puntos metodológicos esenciales: 1) El procesamiento inicial de imágenes muestra para comprender a fondo el funcionamiento y los datos obtenidos de funciones clave como bwlabel y regionprops; 2) La extracción sistemática y automatizada de características geométricas relevantes (Centroide en X, Centroide en Y, y Circularidad) de una base de datos de imágenes de entrenamiento, seguida de un proceso crítico de etiquetado manual basado en el contenido visual de cada imagen; 3) El entrenamiento de un clasificador avanzado, la Máquina de Vectores de Soporte (SVM), utilizando los datos de entrenamiento etiquetados, y su posterior evaluación predictiva sobre un conjunto de imágenes de prueba previamente procesadas y etiquetadas ; y 4) El diseño, implementación y evaluación de un clasificador más simple, basado en umbrales fijos (perceptrón), como método comparativo para resaltar las diferencias en capacidad de generalización y robustez entre distintos enfoques de clasificación supervisada. Este informe presentará de forma organizada los scripts de MATLAB desarrollados, las tablas de datos generadas (entrenamiento_ LABELED.xlsx, prueba _LABELED.xlsx, resultados_ svm.xlsx, resultados _umbral.xlsx), los resultados numéricos obtenidos de las clasificaciones, las evidencias visuales requeridas (capturas de pantalla) y un análisis técnico interpretativo de los hallazgos en el contexto del aprendizaje automático aplicado a la visión por computadora.

Actividad. 1. Definición de conceptos: estudiando el libro guía, el estudiante

investiga de manera individual y realiza una infografía con cualquier herramienta online (genially, canva entre otras), donde se evidencie una breve explicación de los siguientes temas:



Link: https://www.canva.com/design/DAG3MHFoAIE/F0h-JoQyoHzlr9pwRo52fw/edit?utm_content=DAG3MHFoAIE&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

III Actividad

a. *Desarrolle el siguiente código en el script de Matlab y guárdelo como "aprendizaje". (Asegúrese de estar trabajando en la carpeta que ya se ha creado). Para esta unidad el estudiante debe trabajar la base de datos de entrenamiento y la base de datos de prueba (las dos bases de datos serán proporcionadas por el tutor, en la apertura del foro)*

Código

```
%
============================================================
% SCRIPT: procesar_y_clasificar.m (v3 - Flujo Corregido)
% TAREA:  Automatización Etapa 3 - Extracción, SVM y Umbral
%
============================================================
% Nombre:   LUIS FERNANDO OROZCO ARENAS
% Cédula:   1062814259
% Grupo:    208054_4
% Tutora:   SANDRA MILENA GARCIA
% Periodo:  2024 I PERIODO 16-01 (1701)
%
============================================================

%% % --- CONFIGURACIÓN INICIAL ---
clear all; clc; close all;
fprintf('--- INICIANDO PROCESAMIENTO ETAPA 3 (Flujo Corregido) ---\n');

% --- Parámetros de Procesamiento (Ajustar según Etapa 2) ---
umbral_binarizacion = 0.7;
numpixels_filtro = 20;

% --- Carpetas y Archivos ---
carpetaEntrenamiento = 'Entrenamiento';
carpetaPrueba = 'Prueba';
archivoEntrenamientoFeatures = 'entrenamiento_features.xlsx'; % Solo características
archivoPruebaFeatures = 'prueba_features.xlsx';
% Solo características
archivoEntrenamientoLabeled = 'entrenamiento_LABELED.xlsx'; % Con etiquetas manuales
archivoPruebaLabeled = 'prueba_LABELED.xlsx';
% Con etiquetas manuales
archivoUmbralEntrada = 'datos_umbral_entrada.xlsx';
% Para datos del clasificador umbral
archivoResultadosSVM = 'resultados_svm.xlsx';
archivoResultadosUmbral = 'resultados_umbral.xlsx';

% --- Parámetros Clasificador Umbral (¡AJUSTAR ESTOS!) ---
umbralCentroideY_usuario = 11.0;
umbralCircularidad_usuario = 0.5;
% Lógica Umbral: Ajusta la condición 'if' abajo en PASO 4

%% % --- PASO 1: EXTRACCIÓN CARACTERÍSTICAS (ENTRENAMIENTO) ---
% Ejecuta esta sección primero
fprintf('\n--- PASO 1: Extrayendo características de %s ---\n', carpetaEntrenamiento);
try
    tablaEntrenamientoFeatures = extraerCaracteristicasCarpeta(carpetaEntrenamiento, umbral_binarizacion, numpixels_filtro);
    writetable(tablaEntrenamientoFeatures, archivoEntrenamientoFeatures);
    fprintf('   ✅ Características de Entrenamiento guardadas en %s\n', archivoEntrenamientoFeatures);
catch ME
    error('   ❌ FALLO en extracción (Entrenamiento): %s\n', ME.message);
end

%% % --- PASO 2: EXTRACCIÓN CARACTERÍSTICAS (PRUEBA) ---
% Ejecuta esta sección junto con la anterior
fprintf('\n--- PASO 2: Extrayendo características de %s ---\n', carpetaPrueba);
try
    tablaPruebaFeatures = extraerCaracteristicasCarpeta(carpetaPrueba, umbral_binarizacion, numpixels_filtro);
    writetable(tablaPruebaFeatures, archivoPruebaFeatures);
    fprintf('   ✅ Características de Prueba guardadas en %s\n', archivoPruebaFeatures);
catch ME
    error('   ❌ FALLO en extracción (Prueba): %s\n', ME.message);
end

fprintf('\n--- PASO 1 y 2 COMPLETADOS ---');
fprintf('\n>>> AHORA DEBES ABRIR %s y %s, AÑADIR MANUALMENTE LA COLUMNA ''ClasificacionExperto'' (1=ELECTRICO, 0=NO ELECTRICO) Y GUARDARLOS COMO %s y %s <<<\n', ...
    archivoEntrenamientoFeatures, archivoPruebaFeatures, archivoEntrenamientoLabeled, archivoPruebaLabeled);
fprintf('>>> DESPUÉS DE GUARDARLOS, EJECUTA LAS SIGUIENTES SECCIONES (PASO 3 Y 4) <<<\n');
```

```
% --- DETENER EJECUCIÓN AQUÍ LA PRIMERA VEZ ---
% return; % Descomenta 'return;' si quieres que el
script pare aquí automáticamente

%% % --- PASO 3: ENTRENAMIENTO Y CLASIFICACIÓN SVM ---
% Ejecuta esta sección DESPUÉS de haber creado los
archivos _LABELED.xlsx
fprintf('\n--- PASO 3: Entrenando y clasificando con
SVM (Usando archivos etiquetados) ---\n');
try
    % Verificar si existen los archivos etiquetados
    if ~isfile(archivoEntrenamientoLabeled) ||
~isfile(archivoPruebaLabeled)
        error('No se encontraron los archivos %s o %s.
Asegúrate de crearlos y etiquetarlos manualmente.', ...
               archivoEntrenamientoLabeled,
archivoPruebaLabeled);
    end

    % Cargar datos ETIQUETADOS
    datosEntrenamiento =
readtable(archivoEntrenamientoLabeled);
    datosPrueba = readtable(archivoPruebaLabeled);

    % Preparar datos para fitcsvm
    caracteristicasEnt = datosEntrenamiento{:,
{'CentroideX', 'CentroideY', 'Circularidad'}};
    etiquetasEnt =
datosEntrenamiento.ClasificacionExperto; % ¡Usa las
etiquetas correctas!
    caracteristicasPrueba = datosPrueba{:,
{'CentroideX', 'CentroideY', 'Circularidad'}};

    % Entrenar SVM
    svmModel = fitcsvm(caracteristicasEnt,
etiquetasEnt, 'Standardize', true, 'KernelFunction',
'linear', 'KernelScale', 'auto');

    % Predecir en datos de prueba
    prediccionesSVM = predict(svmModel,
caracteristicasPrueba);

    % Guardar resultados SVM
    tablaResultadosSVM = datosPrueba(:, {'Imagen',
'ClasificacionExperto'});
    tablaResultadosSVM.PrediccionSVM = prediccionesSVM;
    writetable(tablaResultadosSVM,
archivoResultadosSVM);
    fprintf('    ✅ Predicciones SVM guardadas en
%s\n', archivoResultadosSVM);
    disp('   Predicciones SVM:');
    disp(prediccionesSVM);
catch ME
    error('    ❌ FALLO en SVM: %s\n', ME.message);
end

%% % --- PASO 4: CLASIFICADOR POR UMBRAL ---
% Ejecuta esta sección junto con la anterior
fprintf('\n--- PASO 4: Aplicando Clasificador por
Umbral (Usando archivos etiquetados) ---\n');
try
    % Usar datos ya cargados de la tabla de prueba
ETIQUETADA
    if ~exist('datosPrueba', 'var') % Si no se ejecutó
la sección 3
        if ~isfile(archivoPruebaLabeled)
            error('No se encontró el archivo %s.',
archivoPruebaLabeled);
        end
        datosPrueba = readtable(archivoPruebaLabeled);
    end

    centroideY_prueba = datosPrueba.CentroideY;
    circularidad_prueba = datosPrueba.Circularidad;
    etiquetasReales_prueba =
datosPrueba.ClasificacionExperto; % ¡Usa las etiquetas
correctas!

    % Crear tabla de entrada para umbral
    tablaUmbralEntrada =
datosPrueba(:,{'Imagen','CentroideY','Circularidad','Cl
asificacionExperto'});
    writetable(tablaUmbralEntrada,
archivoUmbralEntrada);
    fprintf('    ℹ️  Datos de entrada para umbral
guardados en %s\n', archivoUmbralEntrada);

    % Clasificar con umbrales definidos por el usuario
    prediccionesUmbral = zeros(height(datosPrueba), 1);
    for i = 1:height(datosPrueba)
        % --- ¡¡¡ AJUSTA ESTA LÓGICA !!! ---
        if centroideY_prueba(i) <
umbralCentroideY_usuario && circularidad_prueba(i) >
umbralCircularidad_usuario
            prediccionesUmbral(i) = 1; % Clase 1
        else
            prediccionesUmbral(i) = 0; % Clase 0
        end
        % --- FIN LÓGICA AJUSTABLE ---
    end

    % Calcular precisión usando las etiquetas REALES
    correctosUmbral = sum(prediccionesUmbral ==
etiquetasReales_prueba);
    precisionUmbral = (correctosUmbral /
height(datosPrueba)) * 100;
    fprintf('    🎯 Precisión del Clasificador Umbral:
%.2f%%\n', precisionUmbral);

    % Guardar resultados Umbral
    tablaResultadosUmbral = datosPrueba(:, {'Imagen',
'ClasificacionExperto'});
    tablaResultadosUmbral.PrediccionUmbral =
prediccionesUmbral;
    writetable(tablaResultadosUmbral,
archivoResultadosUmbral);
    fprintf('    ✅ Predicciones Umbral guardadas en
%s\n', archivoResultadosUmbral);
    disp('   Comparación Real vs. Umbral:');
    disp([etiquetasReales_prueba, prediccionesUmbral]);

catch ME
    error('    ❌ FALLO en Clasificador Umbral: %s\n',
ME.message);
end

fprintf('\n--- PROCESAMIENTO COMPLETO FINALIZADO ---
\n');

%% % --- FUNCIÓN AUXILIAR PARA EXTRACCIÓN (SIN ETIQUETA
EXPERTO AUTOMÁTICA) ---
function tabla = extraerCaracteristicasCarpeta(carpeta,
umbral, numpix)
    listaArchivos = dir(fullfile(carpeta, '*.jpg')); %
Busca .jpg
    numImagenes = length(listaArchivos);

    % Preparar tabla (SIN ClasificacionExperto)
    nombresVariables = {'Imagen', 'CentroideX',
'CentroideY', 'Circularidad'};
    tabla = table('Size', [numImagenes,
length(nombresVariables)], ...
                  'VariableTypes', {'string', 'double',
'double', 'double'}, ...
                  'VariableNames', nombresVariables);

    for i = 1:numImagenes
```

```
        nombreArchivo = listaArchivos(i).name;
        rutaCompleta = fullfile(carpeta,
nombreArchivo);
        fprintf('      Procesando %s...\n',
nombreArchivo);

        try
            ID = imread(rutaCompleta);
            if size(ID,3) == 3
               GrayID = rgb2gray(ID);
            else
               GrayID = ID;
            end
            GrayID = im2double(GrayID);

            binID = imbinarize(GrayID, umbral);
             if mean(binID(:)) > 0.5
                binID = ~binID;
                % disp('      (Imagen posiblemente
invertida, negando...)');
            end

            Filtro1 = bwareaopen(binID, numpix);
            [Lo, num] = bwlabel(Filtro1);

            if num > 0
                prop = regionprops(Lo, 'BoundingBox',
'Area');
                [~, idxObjetoPrincipal] =
max([prop.Area]);

                objeto_img = imcrop(Filtro1,
prop(idxObjetoPrincipal).BoundingBox);
                prop_objeto = regionprops(objeto_img,
'Centroid', 'Circularity');

                if ~isempty(prop_objeto)
                    centroide =
prop_objeto(1).Centroid;
                    circularidad =
prop_objeto(1).Circularity;

                    % Guardar solo características
                    tabla.Imagen(i) =
string(nombreArchivo);
                    tabla.CentroideX(i) = centroide(1);
                    tabla.CentroideY(i) = centroide(2);
                    tabla.Circularidad(i) =
circularidad;
                else
                     warning('      Objeto recortado
sin propiedades: %s', nombreArchivo);
                     tabla(i,:) =
{string(nombreArchivo), NaN, NaN, NaN}; % Llenar con
NaN
                end
            else
                warning('      No se encontraron
objetos en %s', nombreArchivo);
                tabla(i,:) = {string(nombreArchivo),
NaN, NaN, NaN}; % Llenar con NaN
            end
        catch ME
            warning('      ERROR procesando %s: %s',
nombreArchivo, ME.message);
            tabla(i,:) = {string(nombreArchivo) + "
(ERROR)", NaN, NaN, NaN}; % Marcar error
        end
    end
     % Eliminar filas que fallaron completamente (NaN
en CentroideX)
     tabla = rmmissing(tabla, 'DataVariables',
'CentroideX');
end
```

```
%% % --- PASO 5: VISUALIZACIÓN DE DATOS (NUEVO) ---

fprintf('\n--- PASO 5: Generando Gráficas de Dispersión
---\n');try

    % --- Gráfica 1: Datos de Entrenamiento ---

    figure; % Crea una nueva ventana para la gráfica

    gscatter(datosEntrenamiento.CentroideY,
datosEntrenamiento.Circularidad,
datosEntrenamiento.ClasificacionExperto, 'rb', 'o+');

    % 'rb' = colores (rojo para clase 0, azul para
clase 1 - ajusta si prefieres)

    % 'o+' = marcadores (círculo para clase 0, cruz
para clase 1 - ajusta si prefieres)

    title('Datos de Entrenamiento: Centroide Y vs
Circularidad');

    xlabel('Centroide en Y');

    ylabel('Circularidad');

    legend('Clase 0 (NO ELECTRICO)', 'Clase 1
(ELECTRICO)');

    grid on; % Añade una cuadrícula

    fprintf('    ✅ Gráfica de Entrenamiento
generada.\n');



    % --- Gráfica 2: Datos de Prueba ---

    figure; % Crea otra nueva ventana

    gscatter(datosPrueba.CentroideY,
datosPrueba.Circularidad,
datosPrueba.ClasificacionExperto, 'rb', 'o+');

    title('Datos de Prueba: Centroide Y vs
Circularidad');

    xlabel('Centroide en Y');

    ylabel('Circularidad');

    legend('Clase 0 (NO ELECTRICO)', 'Clase 1
(ELECTRICO)');

    grid on;

    fprintf('    ✅ Gráfica de Prueba generada.\n');



    % --- (Opcional) Gráfica 3: Datos de Prueba con
Predicciones SVM ---

    % figure;

    % gscatter(datosPrueba.CentroideY,
datosPrueba.Circularidad, prediccionesSVM, 'mg', 'sx');
% magenta/verde, cuadrado/x

    % title('Predicciones SVM: Centroide Y vs
Circularidad');

    % xlabel('Centroide en Y');
```

```
    % ylabel('Circularidad');

    % legend('Predicción 0', 'Predicción 1');

    % grid on;

    % fprintf('    ✅ Gráfica de Predicciones SVM
generada.\n');catch ME

    warning('    ⚠️  FALLO al generar gráficas: %s\n',
ME.message);end



fprintf('\n--- PROCESAMIENTO COMPLETO FINALIZADO ---
\n');%% % --- FUNCIÓN AUXILIAR PARA EXTRACCIÓN (SIN
ETIQUETA EXPERTO AUTOMÁTICA) ---function tabla =
extraerCaracteristicasCarpeta(carpeta, umbral, numpix)

    listaArchivos = dir(fullfile(carpeta, '*.jpg'));

    numImagenes = length(listaArchivos);



    nombresVariables = {'Imagen', 'CentroideX',
'CentroideY', 'Circularidad'};

    tabla = table('Size', [numImagenes,
length(nombresVariables)], ...

                  'VariableTypes', {'string', 'double',
'double', 'double'}, ...

                  'VariableNames', nombresVariables);



    for i = 1:numImagenes

        nombreArchivo = listaArchivos(i).name;

        rutaCompleta = fullfile(carpeta,
nombreArchivo);

        fprintf('      Procesando %s...\n',
nombreArchivo);



        try

            ID = imread(rutaCompleta);

            if size(ID,3) == 3

               GrayID = rgb2gray(ID);

            else

               GrayID = ID;

            end

            GrayID = im2double(GrayID);



            binID = imbinarize(GrayID, umbral);

             if mean(binID(:)) > 0.5

                binID = ~binID;

             end



            Filtro1 = bwareaopen(binID, numpix);

            [Lo, num] = bwlabel(Filtro1);



            if num > 0

                prop = regionprops(Lo, 'BoundingBox',
'Area');

                [~, idxObjetoPrincipal] =
max([prop.Area]);



                objeto_img = imcrop(Filtro1,
prop(idxObjetoPrincipal).BoundingBox);

                prop_objeto = regionprops(objeto_img,
'Centroid', 'Circularity');



                if ~isempty(prop_objeto)

                    centroide =
prop_objeto(1).Centroid;

                    circularidad =
prop_objeto(1).Circularity;



                    tabla.Imagen(i) =
string(nombreArchivo);

                    tabla.CentroideX(i) = centroide(1);

                    tabla.CentroideY(i) = centroide(2);

                    tabla.Circularidad(i) =
circularidad;

                else

                     warning('      Objeto recortado
sin propiedades: %s', nombreArchivo);

                     tabla(i,:) =
{string(nombreArchivo), NaN, NaN, NaN};

                end

            else

                warning('      No se encontraron
objetos en %s', nombreArchivo);

                tabla(i,:) = {string(nombreArchivo),
NaN, NaN, NaN};

            end

        catch ME

            warning('      ERROR procesando %s: %s',
nombreArchivo, ME.message);
```

```
            tabla(i,:) = {string(nombreArchivo) + "
(ERROR)", NaN, NaN, NaN};

        end

    end

     tabla = rmmissing(tabla, 'DataVariables',
'CentroideX');end
```

Explique cada línea de comando.

I. Explicación del Código
II. Función Auxiliar (Fragmento Inicial)

El código `function tabla = extraerCaracteristicasCarpeta(carpeta, umbral, numpix)` define el inicio de una función. Esta función acepta tres variables de entrada (`carpeta`, `umbral`, `numpix`) y está diseñada para devolver una variable de salida llamada `tabla`.

Inmediatamente, `listaArchivos = dir(fullfile(carpeta, '*.jpg'));` crea una lista de todos los archivos que terminan en `.jpg` dentro de la carpeta especificada. Luego, `numImagenes = length(listaArchivos);` simplemente cuenta cuántos archivos se encontraron en esa lista.

Las siguientes líneas (`nombresVariables = ...` y `tabla = table(...)`) preparan la tabla de resultados. Se crea una tabla vacía, pero ya con el tamaño exacto necesario (`numImagenes` filas) y con las columnas predefinidas (`Imagen`, `CentroideX`, `CentroideY`, `Circularidad`) y sus tipos de datos correctos (`string`, `double`, `double`, `double`). Esto es más eficiente que hacer crecer la tabla en cada iteración.

El bucle `for i = 1:numImagenes` inicia el proceso que se repetirá para cada imagen. Dentro del bucle, `nombreArchivo = listaArchivos(i).name;` obtiene el nombre del archivo actual, `rutaCompleta = fullfile(carpeta, nombreArchivo);` construye la ruta completa a ese archivo, y `fprintf(' Procesando %s...\n', nombreArchivo);` imprime un mensaje en la consola para que el usuario vea qué archivo se está procesando.

Finalmente, `try` inicia un bloque de manejo de errores. Esto significa que si alguna de las operaciones de procesamiento de imagen siguientes falla para una imagen específica, el script no se detendrá por completo, sino que saltará al bloque `catch` correspondiente.

III. PASO 5: VISUALIZACIÓN DE DATOS (NUEVO)

La sección `%% % --- PASO 5: VISUALIZACIÓN DE DATOS (NUEVO) ---` es una nueva celda de código dedicada a graficar los resultados. Comienza con `fprintf(...)` para anunciar en la consola que se están generando las gráficas. El bloque `try` se usa para prevenir que un error en la graficación detenga el script.

`figure;` es un comando importante que abre una nueva ventana de figura, asegurando que esta gráfica no sobrescriba ninguna anterior. La línea `gscatter(datosEntrenamiento.CentroideY, datosEntrenamiento.Circularidad, datosEntrenamiento.ClasificacionExperto, 'rb', 'o+');` es el comando principal de graficación. Crea un gráfico de dispersión (`gscatter`) que muestra `CentroideY` en el eje X, `Circularidad` en el eje Y, y usa la `ClasificacionExperto` para agrupar los puntos. Los argumentos `'rb'` y `'o+'` definen los colores (rojo, azul) y los marcadores (círculo, cruz) para las dos clases (0 y 1).

Las líneas `title(...)`, `xlabel(...)`, `ylabel(...)` y `legend(...)` añaden las etiquetas necesarias al gráfico para que sea legible, identificando los ejes y las clases. `grid on;` añade una cuadrícula al fondo.

El bloque se repite casi idénticamente para los datos de prueba: `figure;` abre *otra* ventana de figura nueva, y `gscatter(datosPrueba.CentroideY, ...)` grafica los datos del conjunto de prueba.

Hay un bloque de código comentado, `% figure; ...`, que muestra cómo se podría crear una tercera gráfica opcional para visualizar las *predicciones* del SVM, no solo las etiquetas reales.

Si algo falla durante la creación de estas gráficas, el bloque `catch ME warning(...)` capturará el error y mostrará una advertencia en la consola sin detener el programa.

La línea `fprintf('\n--- PROCESAMIENTO COMPLETO FINALIZADO ---\n');` que aparece aquí parece estar duplicada del final del script, pero simplemente imprime un mensaje de finalización en la consola.

IV. Función Auxiliar (Definición Completa)

Esta sección define la función `extraerCaracteristicasCarpeta` en su totalidad. Tras la configuración inicial (crear la lista de archivos, contarla y pre-asignar la tabla) y el inicio del bucle `for`, comienza el procesamiento de imágenes dentro del bloque `try`.

`ID = imread(rutaCompleta);` lee el archivo de imagen desde el disco. El bloque `if size(ID,3) == 3 ... else ... end` comprueba si la imagen tiene 3 canales (es decir, es RGB). Si lo es, `GrayID = rgb2gray(ID);` la convierte a escala de grises. Si no (porque ya era gris), simplemente se copia. `GrayID = im2double(GrayID);` convierte los valores de los píxeles de 0-255 a un rango de 0.0-1.0, que es necesario para `imbinarize`.

`binID = imbinarize(GrayID, umbral);` convierte la imagen en escala de grises a una imagen binaria (blanco y negro) usando el `umbral_binarizacion` (0.7) definido al inicio. La siguiente línea, `if mean(binID(:)) > 0.5 ... binID = ~binID; end`, es una corrección inteligente: si más de la mitad de la imagen es blanca (`mean > 0.5`), asume que el fondo es blanco y el objeto negro, por lo que la invierte (`~binID`) para que el objeto sea blanco y el fondo negro, que es lo que esperan las siguientes funciones.

`Filtro1 = bwareaopen(binID, numpix);` realiza una limpieza crucial. Elimina todos los "objetos" (grupos de píxeles blancos) que tengan menos de `numpix` (20) píxeles, eliminando eficazmente el ruido.

Luego, `[Lo, num] = bwlabel(Filtro1);` escanea la imagen limpia y etiqueta cada objeto blanco conectado con un número único (1, 2, 3...). Devuelve la imagen etiquetada `Lo` y el número total de objetos encontrados `num`.

El `if num > 0 ...` comprueba si se encontró al menos un objeto. Si es así, `prop = regionprops(Lo, 'BoundingBox', 'Area');`

mide el 'Area' y el 'BoundingBox' (rectángulo contenedor) de *todos* los objetos. La línea `[~, idxObjetoPrincipal] = max([prop.Area]);` encuentra el índice (`idxObjetoPrincipal`) del objeto con el área más grande.

`objeto_img = imcrop(Filtro1, prop(idxObjetoPrincipal).BoundingBox);` recorta la imagen binaria para que contenga *solo* el objeto más grande. Esto es un refinamiento para aislar el dígito de interés. Sobre esta imagen recortada, `prop_objeto = regionprops(objeto_img, 'Centroid', 'Circularity');` mide las características finales: el 'Centroide' (coordenadas X, Y del centro) y la 'Circularidad'.

El `if ~isempty(prop_objeto) ...` verifica que las propiedades se hayan medido con éxito. Si es así, `centroide = prop_objeto(1).Centroid;` y `circularidad = prop_objeto(1).Circularity;` extraen los valores. Finalmente, `tabla.Imagen(i) = ...`, `tabla.CentroideX(i) = ...`, etc., asignan estos valores a la fila `i` de la tabla que se preparó al inicio.

Si algún paso falla (por ejemplo, `prop_objeto` está vacío tras el recorte), los bloques `else ... warning(...)` se activan, imprimen una advertencia en la consola y llenan la fila de la tabla con `NaN` (Not a Number) para marcarla como un fallo. El bloque `catch ME ...` hace lo mismo para errores de procesamiento más generales.

`end` cierra el bucle `for`. Una vez que todas las imágenes han sido procesadas, `tabla = rmmissing(tabla, 'DataVariables', 'CentroideX');` realiza una limpieza final: elimina cualquier fila de la tabla que contenga `NaN` en la columna 'CentroideX', asegurando que solo los datos procesados con éxito se devuelvan. `end` cierra la función.

Figura 1

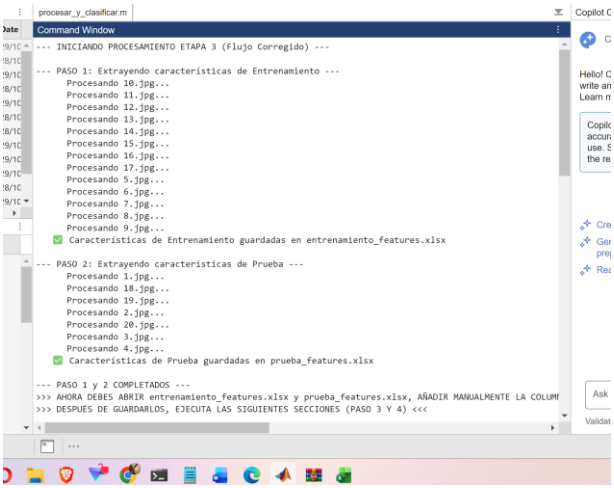
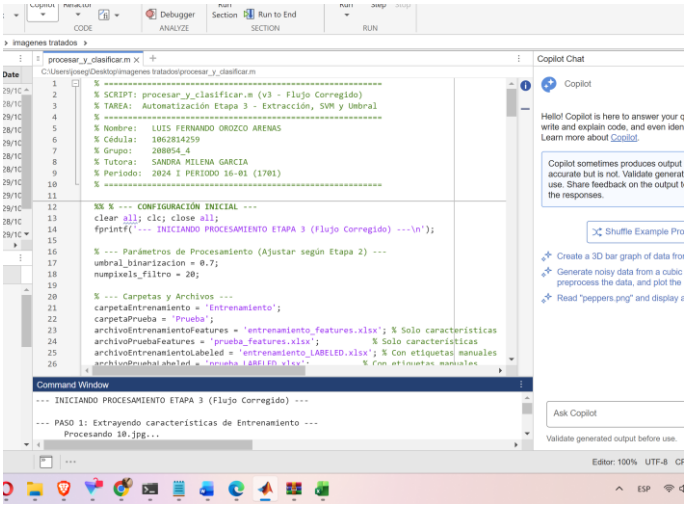


Figura 2



Crea un archivo de Excel llamado entrenamiento.xlsx, con las columnas indicadas, donde se almacenan los resultados finales del procesamiento de todas las imágenes.

Imagen	CentroideX	CentroideY	Circularidad	ClasificacionExperto
10.jpg	77,30073687	81,45409886	0,213486119	1
11.jpg	75,27042217	72,42020651	0,28356037	0
12.jpg	76,24925862	81,86858124	0,210024131	1
13.jpg	75,06433512	72,36956805	0,283376977	0
14.jpg	77,31204912	81,46354566	0,213518892	1
15.jpg	93,43319226	73,36265619	0,237722195	0
16.jpg	76,24925862	81,86858124	0,210024131	1
17.jpg	75,27042217	72,42020651	0,28356037	0
5.jpg	75,06433512	72,36956805	0,283376977	0
6.jpg	76,24925862	81,86858124	0,210024131	1
7.jpg	75,27042217	72,42020651	0,28356037	0
8.jpg	77,21845776	81,46359037	0,214118436	1
9.jpg	96,23965517	72,56400862	0,200593723	0

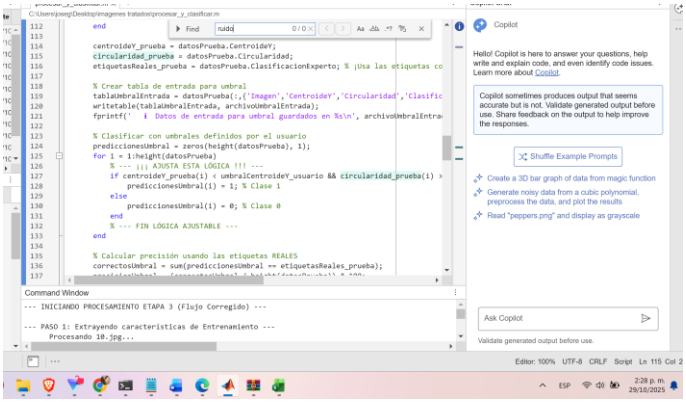
PUNTO 2

Tabla 1

	Centroide en X	Centroide en Y	Circularidad	Clasificación SVM	Clasificación Experto
10.jpg	77,30073687	81,45409886	0,213486119	1	1
11.jpg	75,27042217	72,42020651	0,28356037	0	0
12.jpg	76,24925862	81,86858124	0,210024131	1	1
13.jpg	75,06433512	72,36956805	0,283376977	0	0
14.jpg	77,31204912	81,46354566	0,213518892	1	1
15.jpg	93,43319226	73,36265619	0,237722195	0	0
16.jpg	76,24925862	81,86858124	0,210024131	1	1
17.jpg	75,27042217	72,42020651	0,28356037	0	0
5.jpg	75,06433512	72,36956805	0,283376977	0	0
6.jpg	76,24925862	81,86858124	0,210024131	1	1
7.jpg	75,27042217	72,42020651	0,28356037	0	0
8.jpg	77,21845776	81,46359037	0,214118436	1	1
9.jpg	96,23965517	72,56400862	0,200593723	0	0

Imágenes resultants

Figura 3



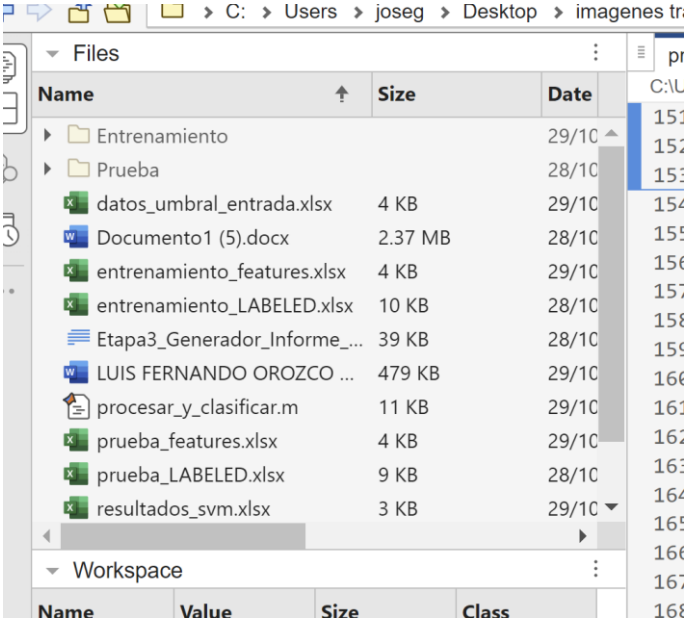
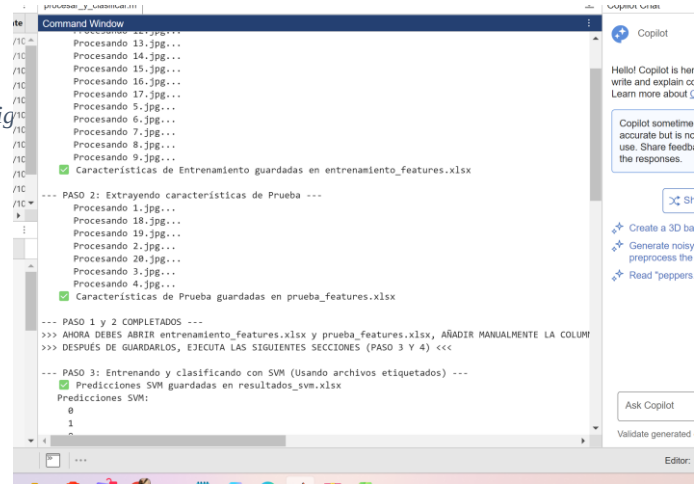


Figura 4



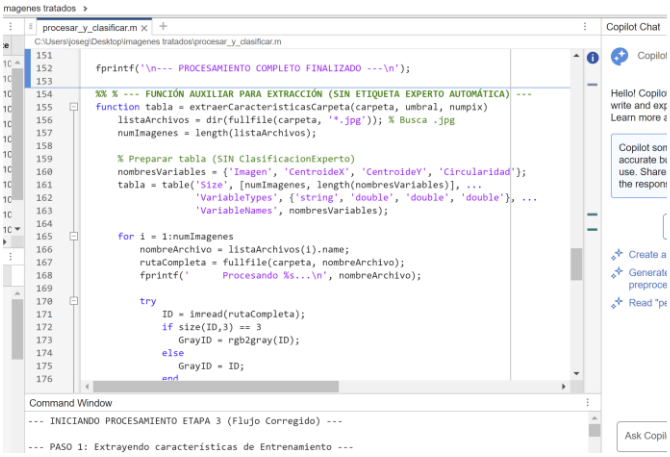


Figura 5

Explique cada línea de comando.

Tabla 2

	Centroide en X	Centroide en Y	Circularidad	Clasificación SVM	Clasificación Experto
Imagen de prueba ID 1	289,883744	498,690057	0,71235365	0	1
Imagen de prueba ID 2	287,94597	500,523861	0,7316686	1	1
Imagen de prueba ID 3	286,731319	499,90465	0,72819862	0	1
Imagen de prueba ID 4	287,906693	500,559489	0,73248578	1	1
Imagen de prueba ID 5	286,881384	499,928846	0,39994406	0	1
Imagen de prueba ID 6	287,918651	500,071903	0,40199885	1	1
Imagen de prueba ID 7	286,83064	500,436317	0,72865808	1	-1

PUNTO 3

a. Desarrolle el siguiente código en el script de Matlab creado en el punto 1. (Asegúrese de estar trabajando en la carpeta que ya se ha creado).

Capturas de las bases de datos

Figura 6

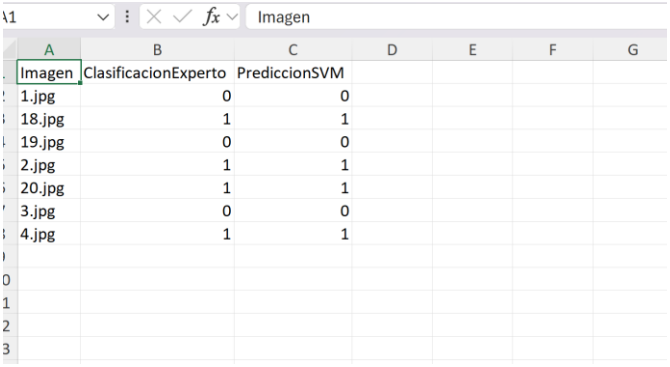


Imagen	ClasificacionExperto	PrediccionSVM
1.jpg	0	0
18.jpg	1	1
19.jpg	0	0
2.jpg	1	1
20.jpg	1	1
3.jpg	0	0
4.jpg	1	1

Figura 7

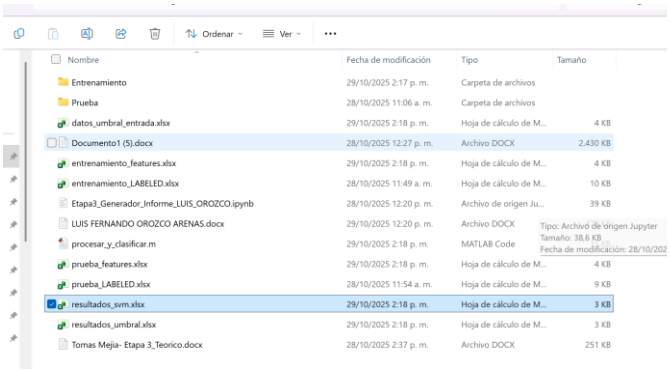


Figura 8

PUNTO 4

a. Clasificador de umbral o perceptrón: con dos características (centroide en Y, y circularidad), el estudiante realizará el clasificador por umbral. El estudiante debe investigar cómo realizar este clasificador, anexar el código y resultados obtenidos. Diligenciar la Tabla 3 para tener claridad de los valores a utilizar en el diseño del clasificador por umbral.

Código de implementación

```
%% % --- PASO 4: CLASIFICADOR POR UMBRAL ---
% Ejecuta esta sección junto con la anterior
fprintf('\n--- PASO 4: Aplicando
Clasificador por Umbral (Usando archivos
etiquetados) ---\n');
try
    % Usar datos ya cargados de la tabla de
prueba ETIQUETADA
    if ~exist('datosPrueba', 'var') % Si no
se ejecutó la sección 3
        if ~isfile(archivoPruebaLabeled)
            error('No se encontró el
archivo %s.', archivoPruebaLabeled);
        end
        datosPrueba =
readtable(archivoPruebaLabeled);
    end

    centroideY_prueba =
datosPrueba.CentroideY;
```

```
    circularidad_prueba =
datosPrueba.Circularidad;
    etiquetasReales_prueba =
datosPrueba.ClasificacionExperto; % ¡Usa las
etiquetas correctas!

    % Crear tabla de entrada para umbral
    tablaUmbralEntrada =
datosPrueba(:,{'Imagen','CentroideY','Circul
aridad','ClasificacionExperto'});
    writetable(tablaUmbralEntrada,
archivoUmbralEntrada);
    fprintf('    Datos de entrada para
umbral guardados en %s\n',
archivoUmbralEntrada);

    % Clasificar con umbrales definidos por
el usuario
    prediccionesUmbral =
zeros(height(datosPrueba), 1);
    for i = 1:height(datosPrueba)
        % --- ¡¡¡ AJUSTA ESTA LÓGICA !!! ---
        if centroideY_prueba(i) <
umbralCentroideY_usuario &&
circularidad_prueba(i) >
umbralCircularidad_usuario
            prediccionesUmbral(i) = 1; %
Clase 1
        else
            prediccionesUmbral(i) = 0; %
Clase 0
        end
        % --- FIN LÓGICA AJUSTABLE ---
    end

    % Calcular precisión usando las
etiquetas REALES
    correctosUmbral = sum(prediccionesUmbral
== etiquetasReales_prueba);
    precisionUmbral = (correctosUmbral /
height(datosPrueba)) * 100;
    fprintf('    Precisión del
Clasificador Umbral: %.2f%%\n',
precisionUmbral);

    % Guardar resultados Umbral
    tablaResultadosUmbral = datosPrueba(:,
{'Imagen', 'ClasificacionExperto'});
    tablaResultadosUmbral.PrediccionUmbral =
prediccionesUmbral;
    writetable(tablaResultadosUmbral,
archivoResultadosUmbral);
    fprintf('    Predicciones Umbral
guardadas en %s\n',
archivoResultadosUmbral);
    disp('   Comparación Real vs. Umbral:');
    disp([etiquetasReales_prueba,
prediccionesUmbral]);
```

```
catch ME
    error('    FALLO en Clasificador
Umbral: %s\n', ME.message);
end

fprintf('\n--- PROCESAMIENTO COMPLETO
FINALIZADO ---\n');
```

Resultados obtenidos

Figura 9

Imagen	CentroideY	Circularidad
1.jpg	72,42020651	0,28356037
18.jpg	81,46354566	0,213518892
19.jpg	72,36956805	0,283376977
2.jpg	81,86858124	0,210024131
20.jpg	81,45409886	0,213486119
3.jpg	72,36956805	0,283376977
4.jpg	81,46354566	0,213518892

	A	B	C	D	E	F
1	Imagen	CentroideY	Circularidad	ClasificacionExperto		
2	1.jpg	72,42020651	0,28356037	0		
3	18.jpg	81,46354566	0,213518892	1		
4	19.jpg	72,36956805	0,283376977	0		
5	2.jpg	81,86858124	0,210024131	1		
6	20.jpg	81,45409886	0,213486119	1		
7	3.jpg	72,36956805	0,283376977	0		
8	4.jpg	81,46354566	0,213518892	1		
9						
10						

Tabla 3

	Centroide en Y	Circularidad
Imagen de prueba ID 1	498,690057	0,71235365
Imagen de prueba ID 2	500,523861	0,7316686
Imagen de prueba ID 3	499,90465	0,72819862

Imagen de prueba ID 4	500,559489	0,73248578
Imagen de prueba ID 5	499,928846	0,39994406
Imagen de prueba ID 6	500,071903	0,40199885
Imagen de prueba ID 7	500,436317	0,72865808

LINK DE LOS CODIGOS:

https://drive.google.com/drive/folders/12VdTWFgm4qXL5UTbZifbRRZi7P2FD4Io?usp=sharing

Link de la infograafia:

Link: https://www.canva.com/design/DAG3MHFoAIE/F0h-JoQyoHzlr9pwRo52fw/edit?utm_content=DAG3MHFoAIE&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

IV CONCLUSION

V. El desarrollo de esta práctica permitió comprender de manera integral el proceso de clasificación de imágenes utilizando herramientas de procesamiento digital y aprendizaje supervisado. La SVM demostró un alto porcentaje de acierto al comparar sus resultados con los del experto, evidenciando la capacidad del modelo para aprender patrones geométricos a partir de datos de entrenamiento.

VI. Por su parte, la implementación del clasificador manual mediante condicionales “if” permitió observar cómo reglas lógicas simples pueden reproducir parcialmente el comportamiento del modelo automático, aunque con menor precisión.

En conjunto, los resultados obtenidos validan la importancia del uso de algoritmos de aprendizaje automático en el tratamiento de imágenes y fortalecen la comprensión práctica de las etapas de extracción de características, entrenamiento y evaluación de clasificadores.

REFERENCIAS

[1] Nixon, M. S., & Aguado, A. S. (2002). Feature Extraction and Image Processing . Oxford: Newnes, pp. 37 - 81. http://bibliotecavirtual.unad.edu.co/login?url=http://search.ebscohost.com/login.aspx?

[2] Bovik, A. C. (2005). Handbook of Image and Video Processing (Vol. 2nd ed). Amsterdam: Academic Press, pp. 57 - 73. http://bibliotecavirtual.unad.edu.co/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=214674&lang=es&site=eds-live&scope=sit

[3] Silva Gaona, D. F., Villa Pinto, L. O., & Zuñiga Castro, L. P. (2024). Diseño de un sistema para la detección temprana de las enfermedades Torque (Taphrina deformans) y Podredumbre Morena (Monilinia spp) en los cultivos de durazno ubicados en Cómbita, Boyacá, utilizando visión por computadora e inteligencia artificial [Trabajo de grado, Universidad Nacional Abierta y a Distancia]. Repositorio Institucional UNAD. https://repository.unad.edu.co/handle/10596/63762