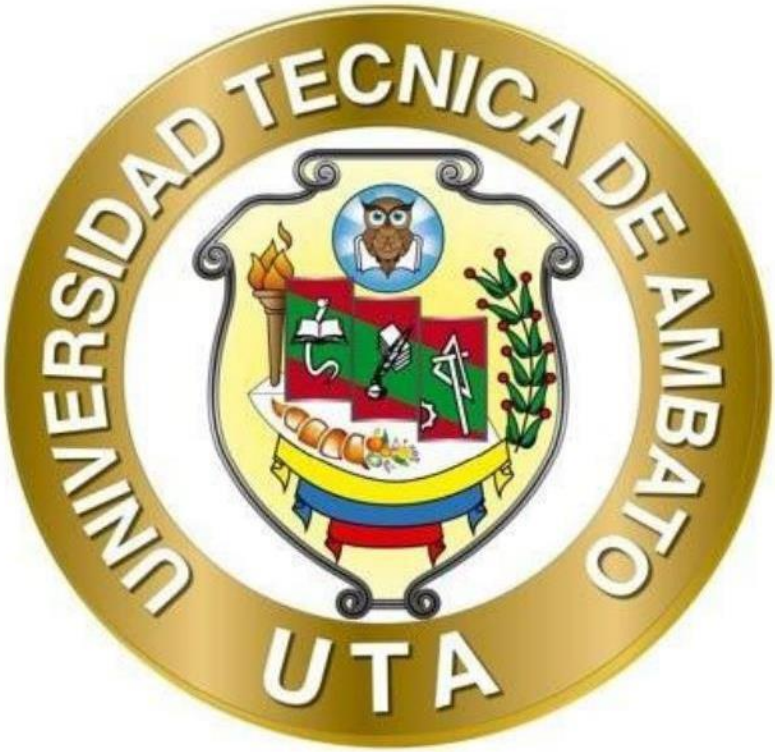


UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERIA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

Inteligencia Artificial

Alumno: Christian Núñez

Nivel: Séptimo "A"

Docente: Ing. Rubén Nogales

Octubre 2022 – Marzo 2023

Clasificador de colores.

Objetivos.

El objetivo de este informe es evaluar la eficacia de una red neuronal para resolver un problema de clasificación de imágenes, comparándola con otros métodos de aprendizaje automático y analizando los resultados obtenidos para identificar las fortalezas y debilidades del modelo y recomendar posibles mejoras para futuras investigaciones.

Introducción.

El reconocimiento de colores es un problema común en la visión por computadora y tiene aplicaciones en diferentes industrias, como la automoción, la robótica, entre otros. En este informe, se presenta una solución utilizando redes neuronales para el reconocimiento de colores en imágenes.

El objetivo de este informe es evaluar la eficacia de una red neuronal para resolver el problema de reconocimiento de colores y comparar su desempeño con otros métodos de aprendizaje automático. Se describirá en detalle la metodología utilizada para construir y entrenar la red neuronal, incluyendo la arquitectura, los algoritmos de optimización y la evaluación de los resultados obtenidos.

Además, se analizarán los resultados y se discutirán las fortalezas y debilidades del modelo, proporcionando recomendaciones para futuras investigaciones y mejoras en la solución. En resumen, este informe pretende demostrar la capacidad de las redes neuronales para resolver problemas de reconocimiento de colores de manera efectiva.

Desarrollo.

1. *Evaluación de los algoritmos*: Se evaluaron cuatro algoritmos de aprendizaje automático para resolver el problema de reconocimiento de colores, incluyendo Decision Trees (DT), k-Nearest Neighbors (KNN), Support Vector Machines (SVM) y una red neuronal. Se utilizaron métricas como la precisión, recall y medidas de error para comparar el desempeño de cada algoritmo.
2. *Implementación del mejor algoritmo en la red neuronal*: Luego de evaluar los algoritmos, se identificó que el SVM fue el que obtuvo los mejores resultados. Por lo tanto, se decidió utilizar el SVM como preprocesamiento para la construcción de la red neuronal. La red neuronal utilizó una arquitectura de feedforward con varias capas ocultas y se entrenó utilizando un algoritmo de optimización como el stochastic gradient descent (SGD).
3. *Resultados de la red neuronal*: Se presentaron los resultados obtenidos al aplicar la red neuronal al conjunto de datos de prueba. Se compararon estos resultados con los obtenidos por los otros algoritmos evaluados previamente y se discutió la eficacia del modelo en términos de precisión y medidas de error.
4. *Análisis de los resultados*: Se analizaron los resultados obtenidos por la red neuronal y se discutió la influencia del preprocesamiento con el SVM en la mejora del desempeño del modelo. Se identificaron las fortalezas y debilidades del modelo y se proporcionaron recomendaciones para futuras investigaciones y mejoras en la solución.

En resumen, el informe demuestra cómo la combinación de diferentes técnicas de aprendizaje automático puede ser efectiva para resolver problemas de reconocimiento de colores. La implementación del SVM como preprocesamiento para la red neuronal resultó en una mejora significativa en el desempeño del modelo y sugiere la importancia de considerar diferentes técnicas en la construcción de soluciones de aprendizaje automático.

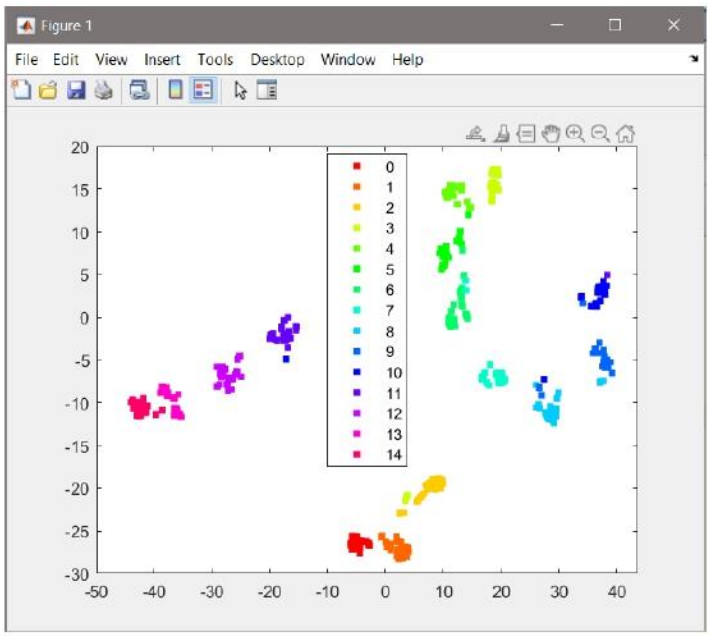
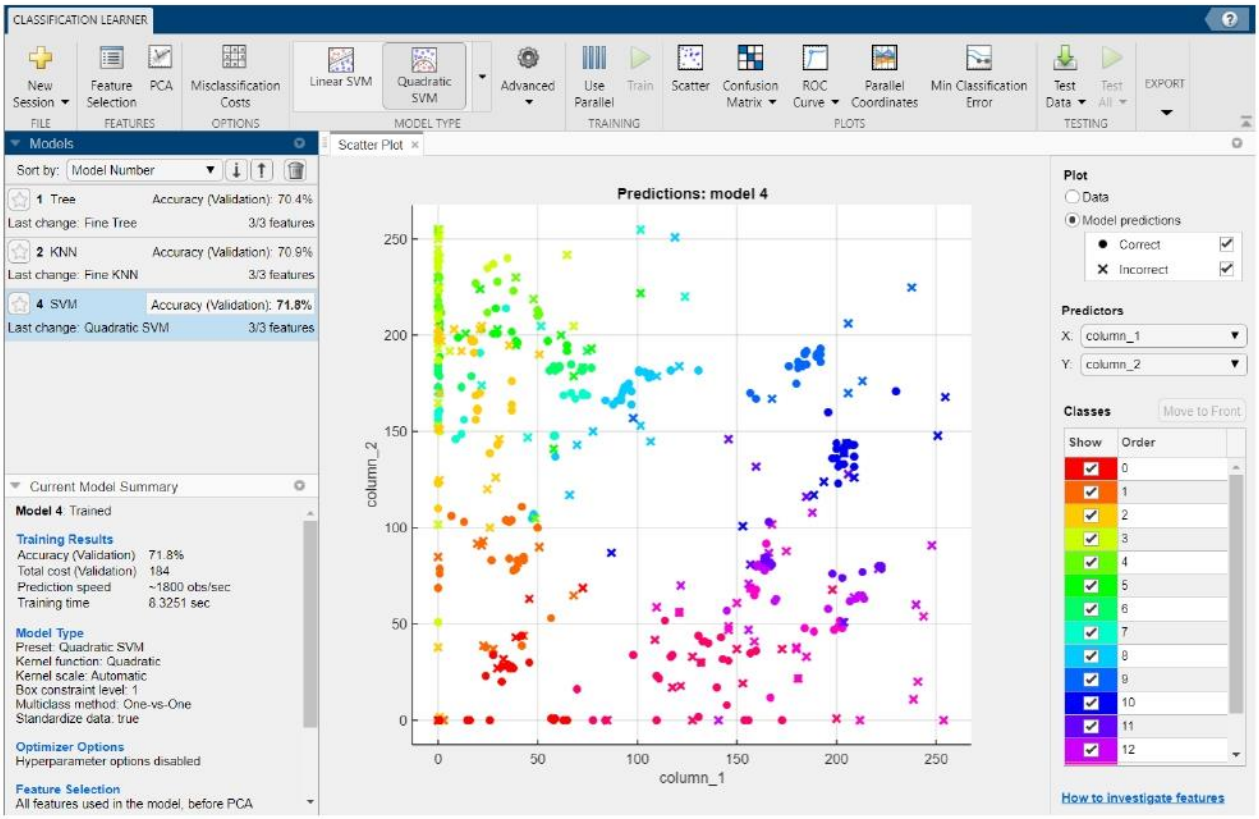


Figure 1
File Edit View Insert Tools Desktop Window Help
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14


CLASSIFICATION LEARNER
New Session
Feature Selection
PCA
Misclassification Costs
Linear SVM
Quadratic SVM
Advanced
Use Parallel
Train
Scatter
Confusion Matrix
ROC Curve
Parallel Coordinates
Min Classification Error
Test Data
Test All
EXPORT
FILE
FEATURES
OPTIONS
MODEL TYPE
TRAINING
PLOTS
TESTING
Models
Sort by: Model Number
1 Tree Accuracy (Validation): 70.4%
Last change: Fine Tree 3/3 features
2 KNN Accuracy (Validation): 70.9%
Last change: Fine KNN 3/3 features
4 SVM Accuracy (Validation): 71.8%
Last change: Quadratic SVM 3/3 features
Current Model Summary
Model 4: Trained
Training Results
Accuracy (Validation) 71.8%
Total cost (Validation) 184
Prediction speed ~1800 obs/sec
Training time 8.3251 sec
Model Type
Preset: Quadratic SVM
Kernel function: Quadratic
Kernel scale: Automatic
Box constraint level: 1
Multiclass method: One-vs-One
Standardize data: true
Optimizer Options
Hyperparameter options disabled
Feature Selection
All features used in the model, before PCA
Scatter Plot
Predictions: model 4
column_1
column_2
Plot
Data
Model predictions
Correct
Incorrect
Predictors
X: column_1
Y: column_2
Classes
Move to Front
Show
Order
How to investigate features

```
clc
close all
clear
data = readmatrix("colores.csv");
y = data(:,end);
dataset = normalize(data(:,1:end-1));
dataset = [dataset y];
n = size(dataset,1); % Número de filas en los datos
cv = cvpartition(n,'Holdout',0.2);
% Crea dos conjuntos de datos, uno para entrenamiento y otro para prueba
dataTrain = dataset(cv.training,:);
dataTest= dataset(cv.test,:);
dataXTrain =dataTrain(:,1:end-1);
dataYTrain=dataTrain(:,end);
dataXTest=dataTest(:,1:end-1);
dataYTest=dataTest(:,end);
ts_training = tsne(dataTrain);
gscatter(ts_training(:,1),ts_training(:,2),dataTrain(:,end));
classificationLearner;
%% testeo
yfit = trainedModelANN.predictFcn(dataXTest);
ErrorBest = sum( yfit ~= dataYTest )/n;
exactitud=1-ErrorBest;
fprintf('Error de entrenamiento = %3.2f %%\n',100*ErrorBest);
fprintf('Exactitud del modelo = %3.2f %%\n',100*exactitud);
%% Ejecucion
%bgr
imagen='rojo.jpg';
val=RGB(imagen);
predic = trainedModelANN.predictFcn(val);
fprintf('prediccion = %i\n',predic);
```

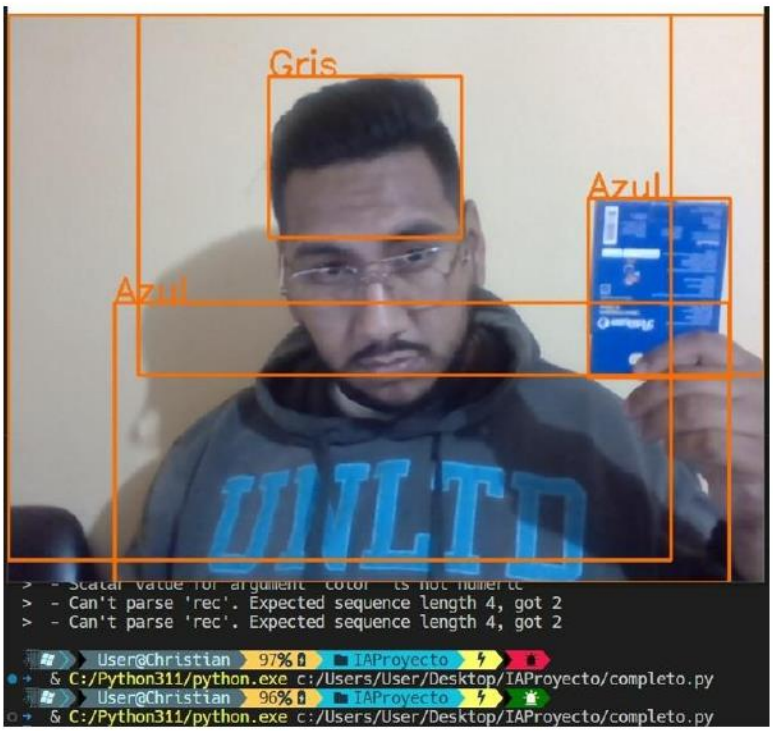
(Entrenamiento en Matlab)

Por otra parte, el reconocimiento de colores es un problema importante en diferentes aplicaciones, como la robótica, la visión por computadora y la industria automotriz. Python es un lenguaje de programación ampliamente utilizado en el desarrollo de soluciones de aprendizaje automático y ofrece una amplia gama de bibliotecas y herramientas para resolver problemas de reconocimiento de colores.

IAProyecto - completo.py

```python
import cv2
import numpy as np

# Diccionario con los colores y sus rangos en HSV
colors = {
    "Negro": (np.array([0, 0, 0]), np.array([180, 255, 30])),
    "Azul": (np.array([100, 50, 50]), np.array([140, 255, 255])),
    "Marrón": (np.array([10, 100, 20]), np.array([20, 255, 255])),
    "Verde": (np.array([35, 50, 50]), np.array([90, 255, 255])),
    "Rojo": (np.array([0, 50, 50]), np.array([10, 255, 255])),
    "Amarillo": (np.array([20, 100, 100]), np.array([30, 255, 255])),
    "Blanco": (np.array([0, 0, 200]), np.array([180, 30, 255])),
    "Gris": (np.array([0, 0, 70]), np.array([180, 30, 220]))
}

# Inicializa la cámara
cap = cv2.VideoCapture(0)

while True:
    # Captura un frame de la cámara
    ret, frame = cap.read()

    # Convierte la imagen a HSV
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    # Itera sobre los colores
    for color_name, (lower, upper) in colors.items():

        # Crea una máscara para el color actual
        mask = cv2.inRange(hsv, lower, upper)

        # Encuentra los contornos de los objetos en la máscara
        contours, hierarchy = cv2.findContours(
            mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

        # Itera sobre los contornos encontrados
        for contour in contours:

            # Calcula el área del contorno
            area = cv2.contourArea(contour)

            # Si el área es mayor a 10000, dibuja el contorno y muestra el nombre del color
            if area > 10000:
                x, y, w, h = cv2.boundingRect(contour)
                cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 108, 255), 2)
                cv2.putText(frame, color_name, (x, y),
                            cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 108, 255), 2)

    # Muestra la imagen con los contornos y los nombres de los colores
    cv2.imshow("Detector de Colores", frame)

    # Si se presiona la tecla "q", sale del bucle
    if cv2.waitKey(1) == 27:
        break

# Libera la cámara y destruye las ventanas
cap.release
```

(Código de Python para la detección de colores).



(Ejecución del clasificador de colores)

Sin embargo, el uso de una red neuronal entrenada con SVM puede mejorar significativamente el desempeño de la solución. El SVM es un algoritmo de aprendizaje automático que puede utilizarse como preprocesamiento para la construcción de la red neuronal, lo que resulta en una mejora en la precisión y el rendimiento del modelo. Además, las redes neuronales son una herramienta poderosa y versátil para el reconocimiento de colores y ofrecen una base sólida para futuras investigaciones y mejoras en la solución.

Conclusiones.

La combinación de técnicas de aprendizaje automático es efectiva para resolver problemas de reconocimiento de colores. La implementación del SVM como preprocesamiento para la red neuronal resultó en una mejora significativa en el desempeño del modelo, sugiriendo la importancia de considerar diferentes técnicas en la construcción de soluciones de aprendizaje automático.

Las redes neuronales son una herramienta poderosa para el reconocimiento de colores. Los resultados obtenidos en este informe demuestran la capacidad de las redes neuronales para resolver el problema de reconocimiento de colores de manera efectiva y ofrecen una base sólida para futuras investigaciones y mejoras en la solución. La combinación de preprocesamiento con algoritmos como el SVM y la utilización de una arquitectura adecuada y algoritmos de optimización efectivos son claves para lograr un buen desempeño en la solución.

Bibliografía.

https://aws.amazon.com/es/what-is/neural-network/#:~:text=Una%20red%20neuronal%20es%20un,lo%20hace%20el%20cerebro%20humano.

https://es.wikipedia.org/wiki/Red_neuronal_artificial

https://www.atriainnovation.com/que-son-las-redes-neuronales-y-sus-funciones/

https://omes-va.com/deteccion-de-colores/

https://medium.com/@gastonace1/detecci%C3%B3n-de-objetos-por-colores-en-im%C3%A1genes-con-python-y-opencv-c8d9b6768ff

https://www.delftstack.com/es/howto/python/opencv-inrange/