"CLASIFICACION MEDIANTE KNN, RED NEURONAL Y MÁQUINAS DE VECTOR SOPORTE PARA UNA DESCRIPCIÓN RGB"

CLASIFICACIÓN DE UNA MUESTRA INDIVIDUAL

K VECINOS MÁS CERCANOS (KNN):

```
Codificacion_input = knnclassify (input, input_ent (:, 1:3) , outputs_ent, k)
```

- `input:` vector fila con los valores de R, G y B del píxel a clasificar:
- `input_ent:` datos RGB (3 columnas) de todas las muestras de entrenamiento disponibles (dispuestas en filas).
- `outputs_ent:` vector columna con la codificación de las muestras de entrenamiento.
- `k:` número de vecinos a considerar.

En versiones actuales de Matlab, kNN se diseña en dos pasos: primero se crea el modelo y después se aplica:

```
KNN_model = fitcknn(inputs_ent, outputs_ent,'NumNeighbors',k);
        Codificacion_input = predict(KNN_model,input);
```

MÁQUINA DE VECTOR SOPORTE (SVM):

Creación del modelo SVM y ajuste de sus parámetros:

```
SVMModel =  fitcsvm (inputs_ent, outputs_ent);
SVMModel = compact(SVMModel);
```

Aplicación del modelo sobre una muestra desconocida:

```
Codificacion_input = predict(SVMModel, input);
```

- `input:` vector fila con los valores de R, G y B del píxel a clasificar:
- `input_ent:` datos RGB (3 columnas) de todas las muestras de entrenamiento disponibles (dispuestas en filas).
- `outputs_ent:` vector columna con la codificación de las muestras de entrenamiento.

RED NEURONAL:

```
Codificacion_input = sim (net, input) %  net(input)
```

- `input:` vector columna con los valores de R, G y B del píxel a clasificar.
- `net:` red entrenada con los siguientes datos de entrenamiento:
    - `input_ent:` datos RGB (3 filas) de todas las muestras de entrenamiento disponibles (dispuestas en columnas).
    - `outputs_ent:` vector fila con la codificación de las muestras de entrenamiento. Puede emplearse codificación binaria, donde cada fila representaría un dígito.

CLASIFICACIÓN DE TODOS LOS PÍXELES DE LA IMAGEN

(para optimizar el funcionamiento del clasificador)

```
[N M]=size(R); % R matriz rojo. Sólo para saber dimensiones de la imagen.

KNNrgb = zeros(N,M);

SVMrgb = zeros(N,M);

NNrgb = zeros(N,M); % Inicializamos matriz Resultado


% PARA HACER EFICIENTE EL CLASIFICADOR - SOLO LO LLAMAMOS UNA VEZ CON TODOS
LOS DATOS. Recorremos por columna la matriz, y vamos poniendo la
información de cada punto ( R G B ) en filas

        input = [];

        for j=1:M

            input_temp = [R(:,j) G(:,j) B(:,j)];

            input = [input ; input_temp];

        end

        KNNrgb_vector = knnclassify(input , inputs_ent, outputs_ent,5);

        SVMrgb_vector = predict(SVMModel , input);

        NNrgb_vector = sim(net, input');

        % Notar que las muestras en input están por filas

% ESCRIBIMOS LA INFORMACION EN LAS MATRICES RESULTADO TENIENDO EN CUENTA EL
ORDEN EN QUE SE GENERARON LOS DATOS - ES DECIR, VAMOS GUARDANDO POR
COLUMNA. Del vector salida del KNN, SVM o NN vamos extrayendo bloques del
tamaño del número de filas y los vamos asignando a cada columna.

       ind =1;

       for j=1:M

            KNNrgb(:,j) =KNNrgb_vector(ind:ind+N-1);

            SVMrgb(:,j) =SVMrgb_vector(ind:ind+N-1);

            NNrgb(:,j) =NNrgb_vector(ind:ind+N-1);

            ind = ind+N;

       end
```