Métricas em Machine Learning - Relatório

Bruna Costa
Faustino Sachimuco
João Meira
Virgílio Loureiro
Orientador: Rui Miguel Soares Pereira

2 de janeiro de 2024

Universidade do Minho
Escola de Ciências

1 Índice

Conteúdo

2 Introdução

Através deste trabalho pretendemos criar e otimizar um algoritmo que identifique e atribua um de três grupos a cada imagem da nossa base de dados. A base de dados consiste em eventos representados por vetores, onde foram identificados três padrões semelhantes a letras A, P e ao símbolo "+". O algoritmo de Lloyd é utilizado como um algoritmo não supervisionado para agrupar os dados em clusters, minimizando a dissimilaridade entre pontos e seus representantes (medoides). Para a escolha dos representantes, é utilizada uma métrica de dissimilaridade entre vetores. O classificador k-Nearest Neighbors (k-NN) é aplicado para classificar novos pontos com base nos k vizinhos mais próximos. Também é empregado um classificador baseado no representante mais próximo.

No seguinte capítulo explicaremos no que consiste a nossa base de dados e faremos uma pequena análise da mesma. No capítulo 4 iremos explicar mais aprofundadamente o algoritmo e as métricas usadas, no capítulo 5 falaremos dos classificadores e de como os escolher e no capítulo 6 falaremos mais dos resultados obtidos com os diferentes métodos.

Os resultados obtidos com o algoritmo de Lloyd apresentam uma eficácia consistente. O classificador k-NN apresenta resultados ligeiramente melhores em comparação com o outro classificador. Os resultados são satisfatórios, sugerindo a aplicabilidade do algoritmo de Lloyd em problemas similares.

3 Breve explicação dos dados

Pretende-se implementar um algoritmo do tipo Lloyd para um conjunto de eventos do ficheiro de texto BD1.txt que é constituído por 30 linhas com 25 atributos cada. Onde cada linha representa um evento x^i do espaço admissível de ocorrências $A = \{0, 1\}^2$.

Para que se possa construir um bom algoritmo de classificação sobre um conjunto de dados não rotulados "UML"é conveniente sempre que possível representar os dados.

Depois de representados os dados em uma Matriz 5X5, foram identificadas essencialmente três Padrões, tal como nos foi dado no número de Clusters, os Padrões identificados se assemelham a duas letras A e P e um símbolo o +.

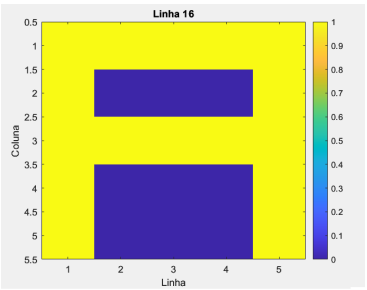

Figura 1: Evento semelhante ao A

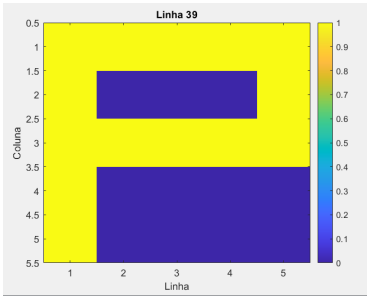

Figura 2: Evento semelhante ao P

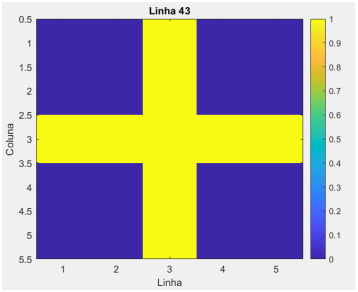

Figura 3: Evento semelhante ao +

Para teste foram disponibilizados dois ficheiros também no formato .txt, nomeadamente teste e teste2 que eram constituídos por 60 e 110 linhas respectivamente, com 26 atributos onde, os 25 primeiros representavam o evento e o último representava a classe a que pertencia este evento.

4 O algoritmo usado

O algoritmo de Loyd é caracterizado por ser um algoritmo não-supervisionado, isto é, um algoritmo que não requer informações das classes associadas às instâncias a serem agrupadas. É um algoritmo de agrupamento bastante popular em várias áreas de conhecimento e tem sido usado em um vasto número de aplicações computacionais, busca descobrir grupos naturais em um conjunto de instâncias, por meio de técnicas estatísticas, realizando comparações de múltiplos atributos.

O principal objetivo do algoritmo de Lloyd é particionar um conjunto de dados em k clusters de forma a minimizar a dissimilaridade ou distância entre os pontos de dados e seus representantes, que podem ser centróides (K-Means) ou medoides (K-Medoids). A dissimilaridade é frequentemente medida usando uma métrica de distância, como a distância euclidiana ou de Manhatan.

4.1 Métricas Usadas

A métrica enunciada foi a dissemelhança entre dois vetores **x** e **y** definida como:

$$\mathrm{d}(\mathbf{x}, \mathbf{y}) = \frac{1}{25} \sum_{i=1}^{25} (d_i(x, y))$$

$$d_i(x, y) = \begin{cases} 1 & \text{se } x_i = y_i \\ 0 & \text{se } x_i \neq y_i \end{cases}$$

O representante de cada cluster é do tipo medóide, pelo que deve-se escolher o elemento do cluster que minimiza a função custo, de m ser representante de C:

$$E(m; C) = \textstyle\sum_{x \in C} d(m, x)$$

O algoritmo usado (do tipo Lloyd) e métricas usadas.

5 O classificador

5.1 k-Nearest Neighbors

O algoritmo de k-Nearest Neighbors(k-NN) é um método de aprendizagem supervisionado desenvolvido pela primeira vez por Evelyn Fix e Joseph Hodges em 1951. Quando encarregado de classificar um novo ponto de dados ou prever um valor alvo, o algoritmo identifica os k vizinhos mais próximos(usando a métrica que definimos) à instância dada com base. No nosso caso o algoritmo atribui o rótulo de classe para a nova instância com base na classe maioritária (no caso de empate a função que usamos escolhe o valor de classe maioritária menor).

Para escolher o melhor valor de K nós definimos um loop que testa todos os valores de k e escolhe o que obteve melhor acuracy.

5.2 Representante mais próximo

Utilizamos também o classificador baseado no representante mais próximo da entrada de teste. O classificador atribui a cada entrada de teste o valor da classe a que pertence o representante que lhe estará mais proximo.(a definição exata depende da métrica usada)

6 Resultados Obtidos

6.1 Métrica dada com Classificador de Representante mais Próximo

Depois de implementado o classificador, usando o algoritmo de Lloyd, foi implementado o arquivo teste.m, por forma a que, caso surjam novos eventos depois obter a partição do domínio seja possível saber a que classe (cluster) deveria pertencer sem voltar a usar o algoritmo de clusterização. O algoritmo foi executado nos arquivos teste2.txt e teste.txt, e os resultados foram analisados.

6.1.1 Resultados com Base de Dados teste.txt

Os resultados para o arquivo teste.txt foram muito bons, indicando uma eficácia consistente do algoritmo, falhando apenas na classificação de 3 elementos. Algumas estatísticas específicas incluem:

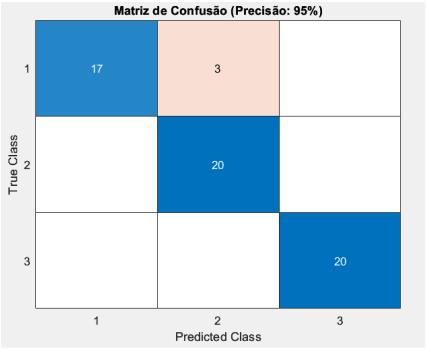

Figura 4: Resultado dos testes sobre os dados de teste em teste.txt

6.1.2 Resultados com Base de Dados teste2.txt

Os resultados obtidos para o arquivo teste2.txt indicam um desempenho positivo, embora ligeiramente inferior ao observado para o arquivo teste.txt. Novamente, o algoritmo apresentou falhas apenas na correta distinção entre os clusters 1 e 2. Algumas estatísticas relevantes incluem:

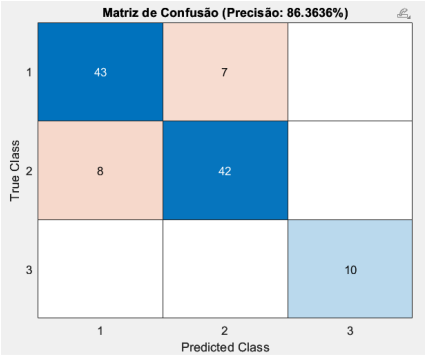

Figura 5: Resultado dos testes sobre os dados de teste em teste2.txt

6.2 Métrica dada com Classificador KNN

Foi então implementado este outro classificador. O algoritmo foi executado nos arquivos teste2.txt e teste.txt, e os resultados foram analisados.

6.2.1 Resultados com Base de Dados teste.txt

Os resultados deste classificador com esta base de dados são muito bons e melhores do que com o classificador usado anteriormente.

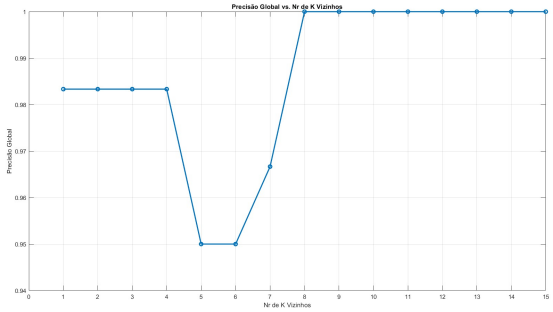

Figura 6: Accuracy vs Nr^{o} de K vizinhos

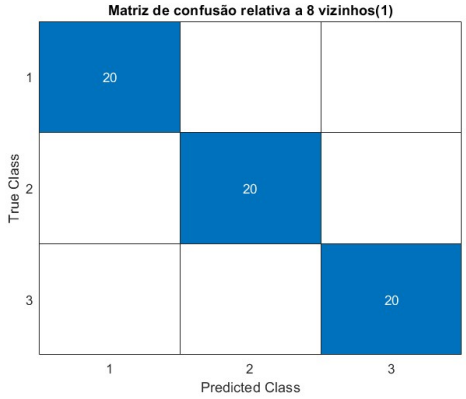

Figura 7: Resultado da classificação sobre os dados teste.txt com o melhor parametro deste classificador nesta base de dados

6.2.2 Resultados com Base de Dados teste2.txt

Os resultados deste classificador com esta base de dados são ligeiramente melhores que com o outro classificador que estudamos, verificando-se que também neste classificador o algoritmo falha essencialmente apenas por vezes a distinguir a classe 1 e a 2.

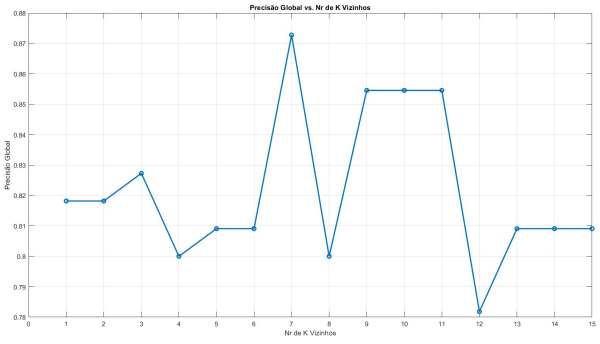

Figura 8: Accuracy vs Nr^{o} de K vizinhos

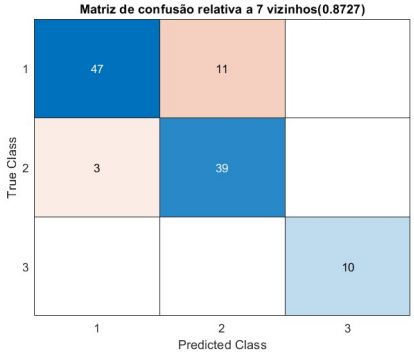

Figura 9: Resultado de teste2.txt com o melhor parametro deste classificador nesta base de dados

Macro-average F1 Score: 0.84587.

6.3 Exemplo de Classificação Errada

Mostramos aqui um exemplo de uma classificação atribuída erroneamente usando esta métrica dada, base de dados 2 e classificador de representante mais próximo:

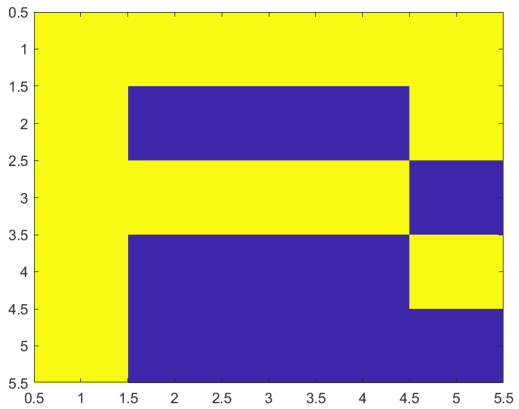

Figura 10: Imagem Nº52 da Base de dados teste2.txt

Esta imagem está classificada na Base de dados como pertencente ao cluster 2 (ou seja o padrão que se assemelha com a letra P). No entanto o nosso classificador atribui-lhe o cluster 1(ou seja o padrão que se assemelha com a letra A). Isto aconteceu porque apesar de visualmente esta imagem se assemelhar bastante com a letra P vejamos que pela definição da métrica usado, a imagem é igualmente dissimilar a ambos os representantes no entanto a função usada no nosso algoritmo escolhe a primeira aparição dessa dissemelhança mínima para o caso de empate entre dissemelhanças de representantes.

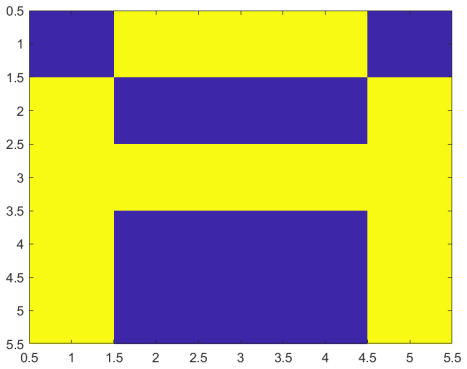

Figura 11: Representante da Classe 1 desta métrica e classificador especificamente

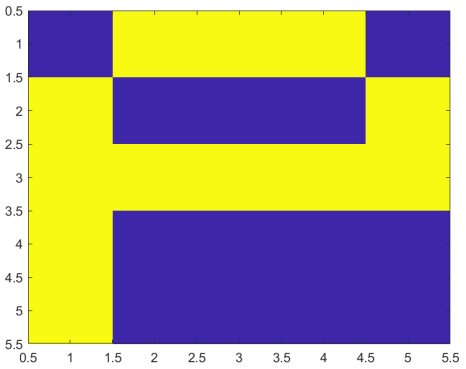

Figura 12: Representante da Classe 2 desta métrica e classificador especificamente

7 Conclusão

Em conclusão, os resultados obtidos com o algoritmo de Loyd foram muito satisfatórios para ambos os conjuntos de dados analisados. A eficiência do algoritmo sugere sua aplicabilidade em problemas similares.

Referências

[1] Marcelo Kuchar, Matte (2020). *Impacto do Uso da Desigualdade Triangular para Acelerar o Algoritmo k-Means.*

[2] Cover, Thomas M.; Hart, Peter E. (1967). *Nearest neighbor pattern classification.*

8 Código

```
clear all
clc

%%% PASSO 1:Abrir os dados do arquivo
data=fopen('BD1.txt','r');
 I=25;

%%% PASSO 2:Definir o número de clusters(k = 3)(seria interessante mudar k)
K = 3;

%%% PASSO 3:Ler os dados do arquivo e criar uma matriz com eles
i=1;
while ~feof(data)
    xx=fscanf(data,'%f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f
    for j=1:I
        x(i,j)=xx(j); %x(i,j) é coordenada j do evento
    end
    i=i+1;
end
fclose(data);
x; %%% matriz criada
N=i-1; %numero de imagens

                  %%%%%%%%%%%%% PLOT DE DADOS %%%%%%%%%%%
%[m, n] = size(x);
%    % Loop para exibir gráficos para cada linha
 %   for i = 1:m
  %      % Criar uma nova figura para cada linha
   %     figure;
    %    % Reshape da linha atual para formar uma matriz 5x5
     %   vetorsimbolo=x(i, :);
      %  imagemSimbolo=vetorsimbolo(end:-1:1);
       % matriz_5x5 = reshape(imagemSimbolo, 5, 5);
%
        % Exibir a matriz usando a função imagesc de maneira a que as
        % imagens fiquem viradas de pé
 %       imagesc(transpose(flipud(matriz_5x5)));
  %  end

               %%%%%%%%%% ALGORITMO DE LLOYD %%%%%%%%%%%%%

% Gerar 3 Numeros aleatorios entre 1 e N
%antigos_representante = randperm(N, 3);
antigos_representante = [9 12 23];
```

```
% Exibe os números gerados
disp(antigos_representante)

CP=1;
it=0;
while  CP>0.0001 && it<10
 P=calcula_particao(antigos_representante,x,K);   % calculo uma partição usando os

 novos_representantes = calcula_representantes(P,x,N,K);  %calculo representantes u
 CP=diferenca_representantes(antigos_representante, novos_representantes) %medida e
 antigos_representante = novos_representantes;
 it=it+1;
end

length(novos_representantes)
novos_representantes

%%%%%%%%%%%%%%% Teste 1 %%%%%%%%%%%%%%%%%%%

i=1;
teste=fopen('teste.txt','r');
while ~feof(teste)
    linhasTeste1=fscanf(teste,'%f\n',26);
    entradasimagem1 = linhasTeste1(1:25);
    ClustersDado1 = linhasTeste1(26);
    if i==101
        entradasimagem1
        ClustersDado1
    end
    for j=1:25
        M1(i,j)=entradasimagem1(j); %x(i,j) é coordenada j do evento
    end
    v(1,i)=ClustersDado1(1);
    i=i+1;
end
fclose(teste);

%%%%%%%%%%%%%%%%% Classificação Teste 1 %%%%%%%%%%%%
ListaPercentagensteste1=[]; %iniciar listas para comparar diferencentes k vizinhos

for U=1:15;
KnnTeste1=knn_predict(x,P,M1,U);

%%%%%%%%%%%%%%%% MATRIZ DE CONFUSÃO Teste 1 %%%%%%%%%%%%%%%%
```

```
 matriz_confusao1 = zeros(K, K);
for i = 1:length(M1)
    classe_teste = v(i);
    classe_prevista = KnnTeste1(i);
%cada vez que um destes casos acontece adicionamos 1 na entrada desse caso na matri

if classe_teste == 1
        if classe_prevista == 1
            matriz_confusao1(1, 1) = matriz_confusao1(1, 1) + 1;
        elseif classe_prevista == 2
            matriz_confusao1(1, 2) = matriz_confusao1(1, 2) + 1;
        elseif classe_prevista == 3
            matriz_confusao1(1, 3) = matriz_confusao1(1, 3) + 1;
        end
    elseif classe_teste == 2
        if classe_prevista == 1
            matriz_confusao1(2, 1) = matriz_confusao1(2, 1) + 1;
        elseif classe_prevista == 2
            matriz_confusao1(2, 2) = matriz_confusao1(2, 2) + 1;
        elseif classe_prevista == 3
            matriz_confusao1(2, 3) = matriz_confusao1(2, 3) + 1;
        end
    elseif classe_teste == 3
        if classe_prevista == 1
            matriz_confusao1(3, 1) = matriz_confusao1(3, 1) + 1;
        elseif classe_prevista == 2
            matriz_confusao1(3, 2) = matriz_confusao1(3, 2) + 1;
        elseif classe_prevista == 3
            matriz_confusao1(3, 3) = matriz_confusao1(3, 3) + 1;
        end
    end
end
%disp(matriz_confusao1);
% Calcular a precisão global (accuracy)
precisao_global1 = sum(diag(matriz_confusao1)) / sum(matriz_confusao1(:));
disp(['Precisão Global do teste 1 com ' num2str(U) ' vizinhos proximos é: ' num2str
ListaPercentagensteste1(end + 1) = precisao_global1;
end

Knnbest1=knn_predict(x,P,M1,8);
confusion_chart=confusionchart(Knnbest1,v);
confusion_chart.Title = 'Matriz de confusão relativa a 8 vizinhos(1)';
drawnow;

keyboard;
```

```
kvizinhos = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15];
plot(kvizinhos, ListaPercentagensteste1, 'o-', 'LineWidth', 2);

% Add labels and title
xlabel('Nr de K Vizinhos');
ylabel('Precisão Global');
title('Precisão Global vs. Nr de K Vizinhos');
grid on;
keyboard;
 %%%%%%%%%%%%%%  Teste 2 %%%%%%%%%%%%%%%%%%%

i=1;
teste2=fopen('teste2.txt','r');
while ~feof(teste2)
    linhasTeste2=fscanf(teste2,'%f\n',26);
    entradasimagem = linhasTeste2(1:25);
    ClustersDado = linhasTeste2(26);
    if i==101
        entradasimagem;
        ClustersDado;
    end
    for j=1:25
        M2(i,j)=entradasimagem(j); %x(i,j) é coordenada j do evento
    end
    v(1,i)=ClustersDado(1);
    i=i+1;
end
fclose(teste2);

%%%%%%%%%%%%%%%%%% Classificação Teste 2 %%%%%%%%%%%%
ListaPercentagens=[]; %iniciar listas para comparar diferencentes k vizinhos
for T=1:15;
KNNteste2=knn_predict(x,P,M2,T);


%%%%%%%%%%%%%%%%%% MATRIZ DE CONFUSÃO Teste 2 %%%%%%%%%%%%%%%%%
 matriz_confusao2 = zeros(K, K);
for i = 1:length(M2)
    classe_teste = v(i);
    classe_prevista = KNNteste2(i);
%cada vez que um destes casos acontece adicionamos 1 na entrada desse caso na matri

if classe_teste == 1
        if classe_prevista == 1
            matriz_confusao2(1, 1) = matriz_confusao2(1, 1) + 1;
```

```
        elseif classe_prevista == 2
            matriz_confusao2(1, 2) = matriz_confusao2(1, 2) + 1;
        elseif classe_prevista == 3
            matriz_confusao2(1, 3) = matriz_confusao2(1, 3) + 1;
        end
    elseif classe_teste == 2
        if classe_prevista == 1
            matriz_confusao2(2, 1) = matriz_confusao2(2, 1) + 1;
        elseif classe_prevista == 2
            matriz_confusao2(2, 2) = matriz_confusao2(2, 2) + 1;
        elseif classe_prevista == 3
            matriz_confusao2(2, 3) = matriz_confusao2(2, 3) + 1;
        end
    elseif classe_teste == 3
        if classe_prevista == 1
            matriz_confusao2(3, 1) = matriz_confusao2(3, 1) + 1;
        elseif classe_prevista == 2
            matriz_confusao2(3, 2) = matriz_confusao2(3, 2) + 1;
        elseif classe_prevista == 3
            matriz_confusao2(3, 3) = matriz_confusao2(3, 3) + 1;
        end
    end
   end
%disp(matriz_confusao2);
%%%%%%%%%%%%%%%%%% F1 Score %%%%%%%%%%%%%%%%%
precision = zeros(3, 1);
recall = zeros(3, 1);
f1_score = zeros(3, 1);

for i = 1:3
    true_positive = matriz_confusao2(i, i);
    false_positive = sum(matriz_confusao2(:, i)) - true_positive;
    false_negative = sum(matriz_confusao2(i, :)) - true_positive;

    % Calculate precision, recall, and F1 for each class
    precision(i) = true_positive / (true_positive + false_positive);
    recall(i) = true_positive / (true_positive + false_negative);

    % Avoid division by zero for precision or recall equal to zero
    if precision(i) + recall(i) == 0
        f1_score(i) = 0;
    else
        f1_score(i) = 2 * (precision(i) * recall(i)) / (precision(i) + recall(i));
    end
end
```

```
% Calculate macro-average F1 score
macro_average_f1 = mean(f1_score);
disp(['Macro-average F1 Score: ', num2str(macro_average_f1)]);

% Calcular a precisão global (accuracy)
precisao_global2 = sum(diag(matriz_confusao2)) / sum(matriz_confusao2(:));
disp(['Precisão Global do teste 2 com ' num2str(T) ' vizinhos proximos é: ' num2str
ListaPercentagens(end + 1) = precisao_global2;
end

Knnbest2=knn_predict(x,P,M2,7);
confusion_chart=confusionchart(Knnbest2,v);
confusion_chart.Title = 'Matriz de confusão relativa a 7 vizinhos(0.8727)';
drawnow;
keyboard;

%%%%%%%%%%%%%%%%%%%%%%% Representação visual das iteraçoes vs accuracy de
%%%%%%%%%%%%%%%%%%%%%%% k vizinhos

kvizinhos = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15];
plot(kvizinhos, ListaPercentagens, 'o-', 'LineWidth', 2);

% Add labels and title
xlabel('Nr de K Vizinhos');
ylabel('Precisão Global');
title('Precisão Global vs. Nr de K Vizinhos');
grid on;

%%%%%representante mais proximo%%%%%
for i=1:length(M2)
    d1=calcula_dissemelhanca(x(novos_representantes(1), :),M2(i,:));
    d2=calcula_dissemelhanca(x(novos_representantes(2), :),M2(i,:));
    d3=calcula_dissemelhanca(x(novos_representantes(3), :),M2(i,:));
    d=[d1 d2 d3];
    [mn,id]=min(d);
    Q(i)=id(1);
end

%%%%%%%%%%%%%%%%Representar um caso em que o Algoritmo preveu mal %%%
% index da primeira classificaçao errada do modelo

misclassified_index = find(v ~= Q,1);
if ~isempty(misclassified_index)
misclassified_vector = M2(misclassified_index, :);
```

```
% Reshape da linha atual para formar uma matriz 5x5
        misclassified_vector=M2(52, :);
        imagemSimbolo=misclassified_vector(end:-1:1);
        matriz_5x5 = reshape(imagemSimbolo, 5, 5);
        % Exibir a matriz usando a função imagesc de maneira a que as
        % imagens fiquem viradas de pé
        imagesc(transpose(flipud(matriz_5x5)));
        v(misclassified_index)
        Q(misclassified_index)

end
keyboard;

%%%%%%%%%%%%%%%% Função KNN MODELO %%%%%%%%%%%%

function predictedLabels = knn_predict(trainData, labelsTreino, testData, k)
    testsize = size(testData, 1);
    numTrainSamples = size(trainData, 1);
    predictedLabels = zeros(testsize, 1);

    for i = 1:testsize
        % Initialize distances for each test sample
        distances = zeros(numTrainSamples, 1);

        for j = 1:numTrainSamples
            % Calcula dissemelhança (usando a métrica especifica) entre a data de t
            distances(j) = sum(testData(i, :) ~= trainData(j, :));
        end

        % ordenar as distancias e retirar as k menores
        [~, indices] = sort(distances);
        kNearestIndices = indices(1:k);

        %extrair o nome/numero do cluster
        kNearestLabels = labelsTreino(kNearestIndices);
        % fazer previsão baseado na moda dos K mais proximos vizinhos
        predictedLabels(i) = mode(kNearestLabels);
    end
end
```