Ejemplo de datos de calidad de fabricación

Las columnas de los datos son: Feature 1 Length (Característica 1 Longitud), Feature 1 Width (Característica 1 Anchura), Feature 2 Length (Característica 2 Longitud), Feature 2 Width (Característica 2 Anchura), Label (1,0,2) Pass/Fail/Rework (Etiqueta (1,0,2) Aprobada/Suspensa/Reelaborar).

Un árbol de decisión es un conjunto de reglas simples, como "si la longitud de la Parte1 es menor que 5,45, clasifique la muestra como suspensa". Los árboles de decisión no son paramétricos porque no requieren ningún supuesto sobre la distribución de las variables.

Actualmente, se han seleccionado dos de las cuatro características para desarrollar un predictor de la calidad de la pieza. Se propone utilizar un árbol de decisión para clasificar una pieza como Aprobada, Reelaborar o Suspensa.

cargar datos (obtener atributos y etiqueta)

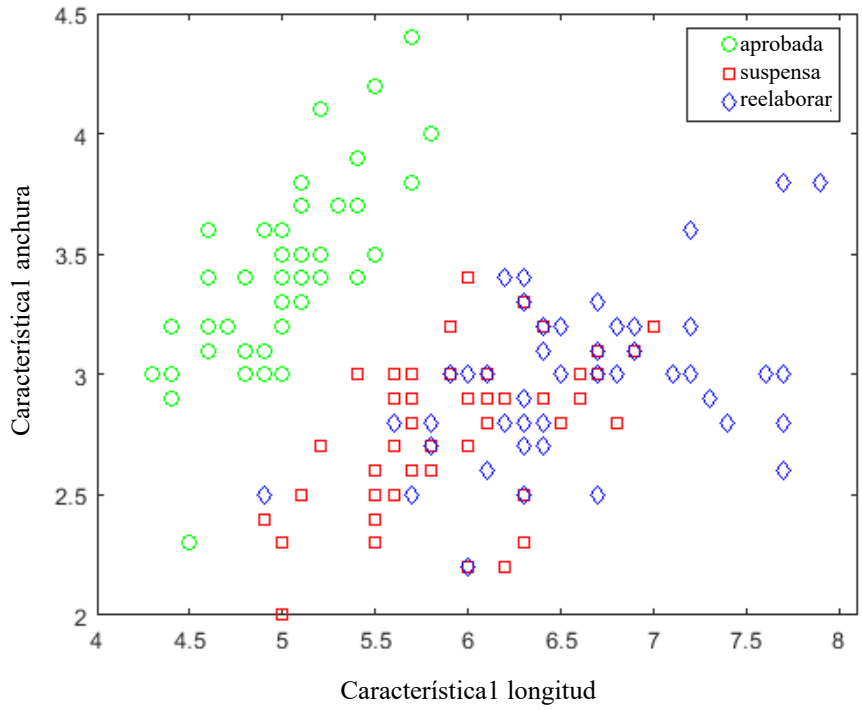
```
datain = xlsread(['manufacturingqualitydata.xlsx']);
Xin = datain(:,1:4);
Yquality = [];
for ii = 1:length(datain(:,end))
    if (datain(ii,end) == 1)
        Yquality{ii} = 'pass';
    end
    if (datain(ii,end) == 0)
        Yquality{ii} = 'fail';
    end
    if (datain(ii,end) == 2)
        Yquality{ii} = 'rework';
    end
end
Yquality = Yquality';
```

etiqueta de configuración

```
%Poda
bPrune = 1;
```

gráfica de dispersión para 2 de las características

```
figure(10)
gscatter(Xin(:,1), Xin(:,2), Yquality,'grb','osd');
xlabel('Feature1 length');
ylabel('Feature1 width');
N = size(Xin,1);
```



Clasifique con un árbol de decisión

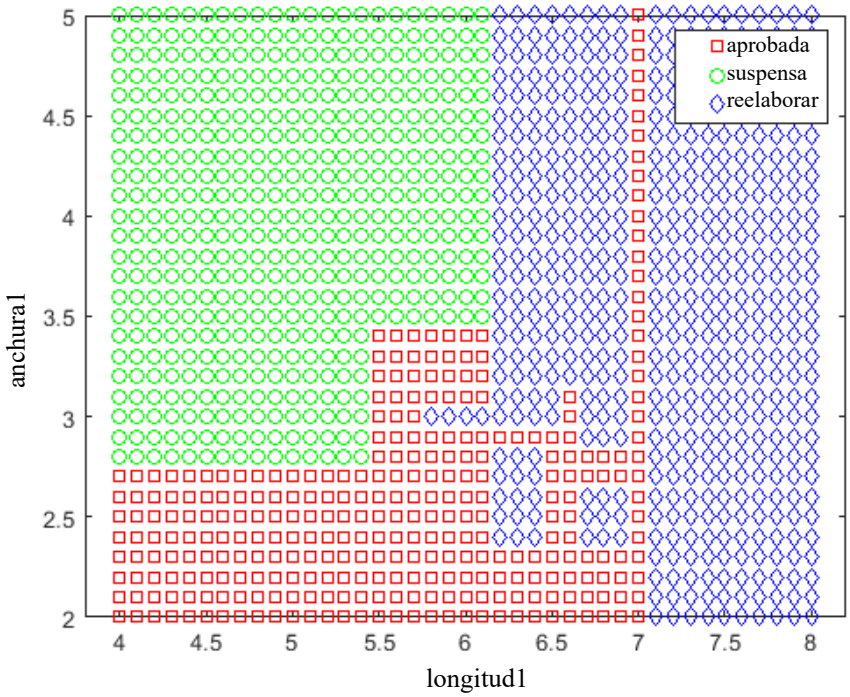
```
% Un árbol de decisión es un conjunto de reglas simples,
% como "si la longitud de la Parte1 es menor
% que 5,45, clasifique la muestra como suspensa".
% Los árboles de decisión no son paramétricos porque
% no requieren ningún supuesto sobre
% la distribución de las variables.

% la función fitctree crea un árbol de decisión.
% Cree un árbol de decisión para la medición de datos
% y vea lo bien que clasifica las piezas;
```

```
Xsub = Xin(:,1:2);
t = fitctree(Xsub, Yquality,'PredictorNames',{'Feat1W' 'Feat1L' });
```

```
% Es interesante ver cómo el método de
% árbol de decisión divide el plano.
```

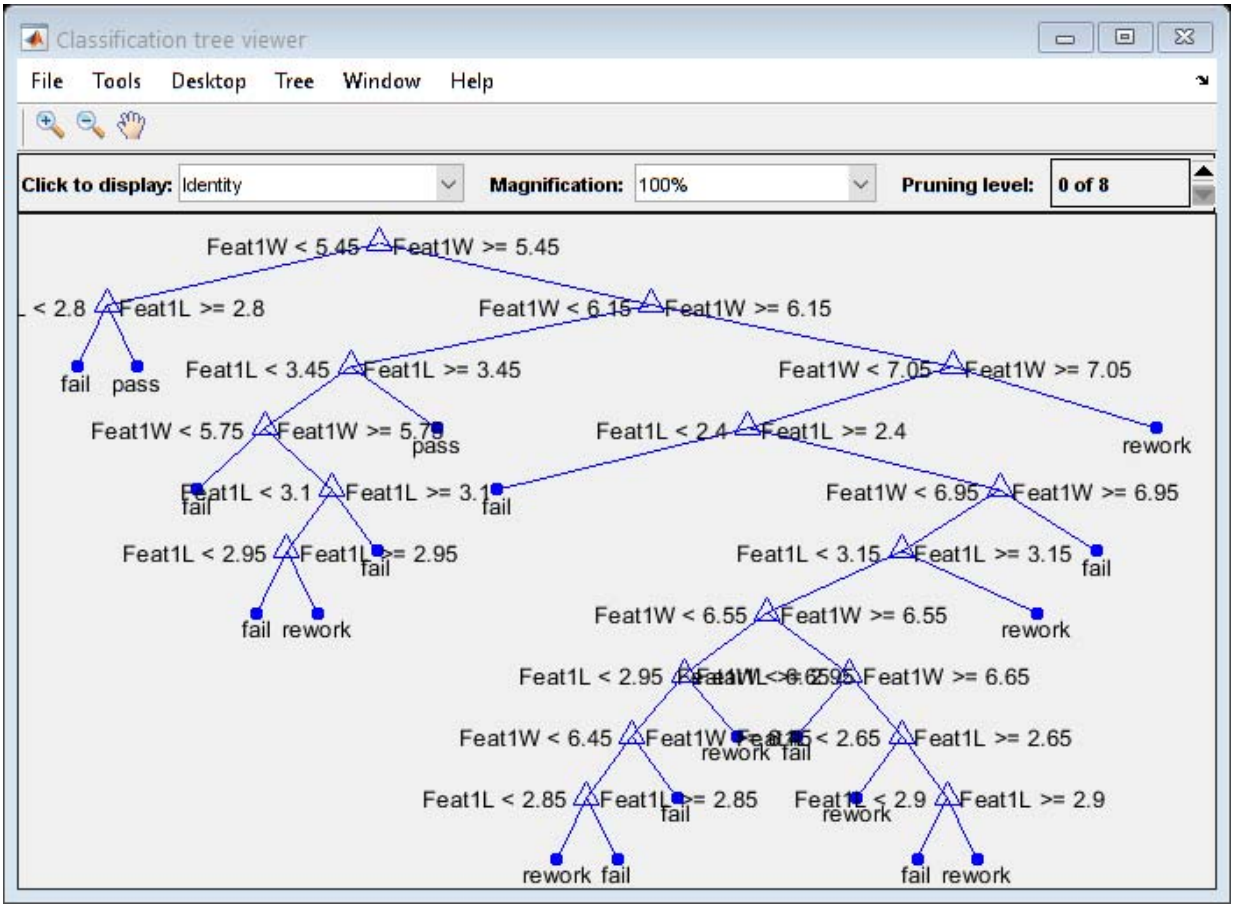
visualice las regiones asignadas a cada nivel de calidad.

```
% cree observación sobre todo el dominio
%[length1,width1] = meshgrid(4:.1:8,2:.1:4.5);
[length1,width1] = meshgrid(floor(min(Xsub(:,1))):.1:ceil(max(Xsub(:,1))),  floor(min(Xsub(:,2))):.1:ceil(max(Xsub(:,2))));
length1 = length1(:);
width1  = width1(:);

figure(20)
[groupname,node] = predict(t,[length1 width1]);
gscatter(length1,width1,groupname,'rgb','sod')
```



Otra manera de visualizar el árbol de decisión es dibujando un diagrama de la regla de decisión y las asignaciones de clase.

```
%figure(30)
view(t,'Mode','graph');

%Este árbol de aspecto agrupado utiliza una serie de reglas... DESCRIBA CÓMO
%FUNCIONA
```



Calcule el error de resustitución y el error de validación cruzada.

```
% Las observaciones con etiquetas de clase conocidas suelen llamarse datos
% de entrenamiento. El error de resustitución es el error de clasificación errónea (la proporción
% de observaciones mal clasificadas) del conjunto de entrenamiento.

% Normalmente, la gente se interesa más por el error de test (también conocido como
% error de generalización), que es el errorde predicción esperado de un
% conjunto independiente. De hecho, es probable que el error de resustitución
% estime a la baja el error de test.

% En caso de no tener otro conjunto de datos etiquetados, puede simular uno
% realizando una validación cruzada. Una validación cruzada estratificada de 10 iteraciones es una
% opción popular para estimar el error de test en algoritmos
% de clasificación. Divide aleatoriamente el conjunto de entrenamiento en 10 subconjuntos de
% iteraciones. Cada subconjunto tiene aproximadamente el mismo tamaño y las mismas proporciones de
% clase que en el set de entrenamiento. Elimine un subconjunto, entrene el
% modelo de clasificación utilizando los otros nueve subconjuntos, y utilice el modelo
% entrenado para clasificar al subconjunto eliminado. Podría repetir esto eliminando
% los 10 subconjuntos, uno cada vez.

% Ya que la validación cruzada divide los datos al azar, su resultado depende de
% la semilla al azar inicial.

% utilice cvpartition para generar 10 subconjuntos de iteraciones estratificadas.

cp = cvpartition(Yquality, 'KFold' ,10);
```

Calcule el error de resustitución y de validación cruzada para el árbol de decisión.

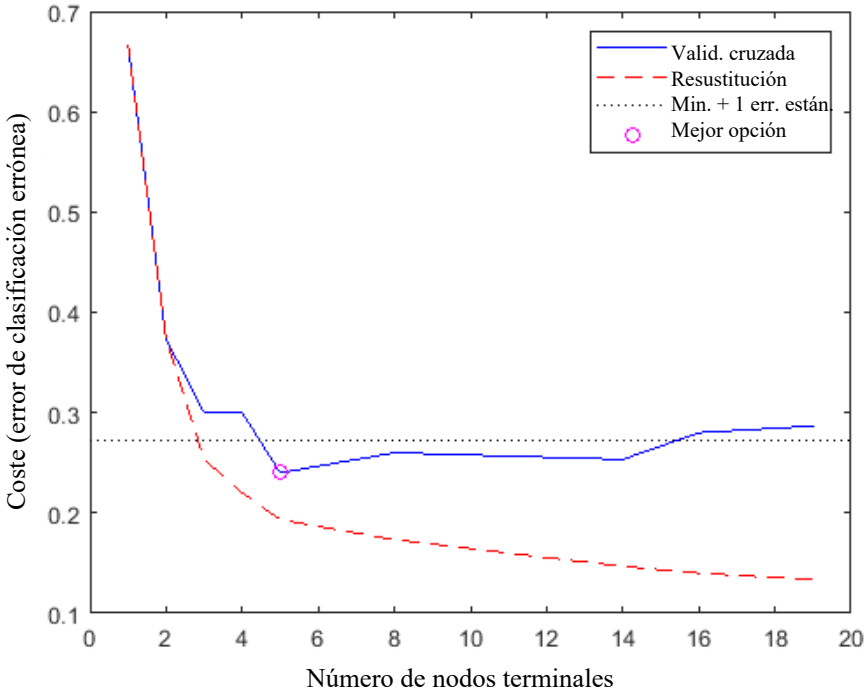
```
% resubLoss --- Error de clasificación por resustitución
dtResubErr = resubLoss(t)

% cvloss --- Error de clasificación por validación cruzada
```

```
cvt = crossval(t,'CVPartition',cp);
dtCVErr = kfoldLoss(cvt)

% Para el algoritmo de árbol de decisión, el cálculo de error de validación cruzada es
% normalmente bastante mayor que el error de resustitución. Esto muestra
% que el árbol generado se sobreajusta (overfits) al conjunto de entrenamiento. Es decir,
este
% es un árbol que clasifica bien el conjunto de entrenamiento original, pero la
% estructura del árbol es sensible a este conjunto en particular, por lo que
% es posible que su desempeño con nuevos datos empeore. A menudo,
% se puede encontrar un árbol más simple que funcione mejor con datos nuevos
% que un árbol más complejo.
```

```
dtResubErr =

    0.1333


dtCVErr =

    0.3000
```

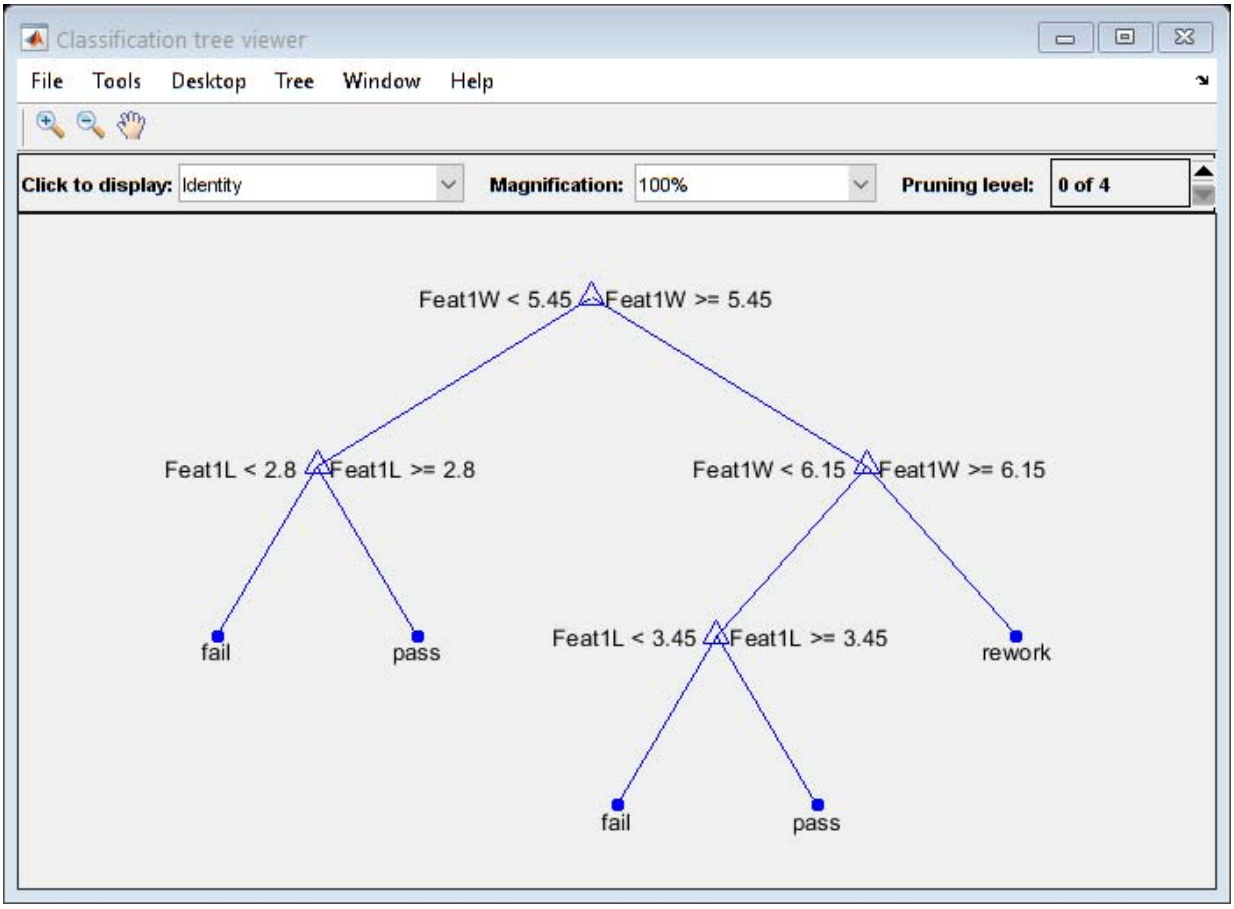
Pruebe a podar el árbol. Primero, calcule el error de resustitución para varios subconjuntos del árbol original. Luego, calcule el error de validación cruzada para estos "subárboles". Una gráfica muestra que el error de resustitución es excesivamente optimista. Siempre disminuye, mientras que el tamaño del árbol aumenta; pero después de cierto punto, aumentar el tamaño del árbol aumenta la tasa de error de validación cruzada.

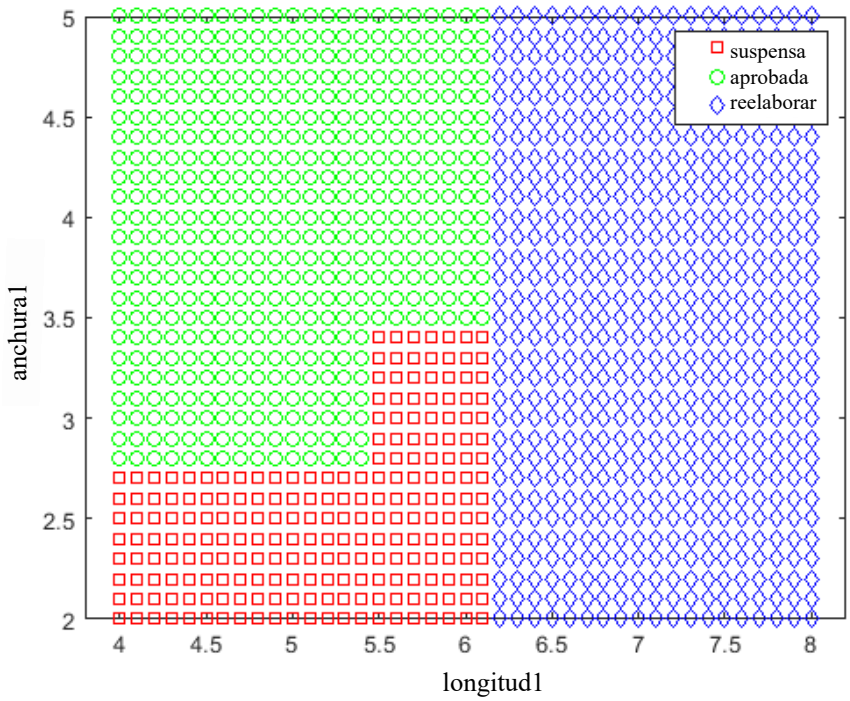
```
figure(40)
resubcost = resubLoss(t,'Subtrees','all');
[cost,secost,ntermnodes,bestlevel] = cvloss(t,'Subtrees','all');
plot(ntermnodes,cost,'b-', ntermnodes,resubcost,'r--')
figure(gcf);
xlabel('Number of terminal nodes');
ylabel('Cost (misclassification error)')
legend('Cross-validation','Resubstitution')

%
% ¿Qué árbol deberíamos escoger? Una regla sencilla sería escoger el árbol
% con el menor error de validación cruzada. Aunque esto pueda ser satisfactorio,
% quizás prefiera utilizar un árbol más sencillo si es prácticamente igual de bueno que uno
% más complejo. Para este ejemplo, escoja el árbol más sencillo que se encuentre dentro del
% error estándar del mínimo. Esa es la regla predeterminada que utiliza el método
% de error de validación cruzada (cvloss)de árbol de clasificación.

% Puede plasmar esto en una gráfica calculando un valor límite que sea igual
% al coste mínimo más un error estándar. El "mejor" nivel calculado por
% el método cvloss es el árbol más pequeño por debajo de ese límite.(Observe que
% bestlevel=0 se corresponde con el árbol sin podar, así que debe añadir 1 para usarlo
% como índice en los resultados de vector de cvloss).

[mincost,minloc] = min(cost);
cutoff = mincost + secost(minloc);
hold on
plot([0 20], [cutoff cutoff], 'k:')
plot(ntermnodes(bestlevel+1), cost(bestlevel+1), 'mo')
legend('Cross-validation','Resubstitution','Min + 1 std. err.','Best choice')
hold off
```




```
if(bPrune)

% Finalmente, puede examinar el "mejor" árbol podado y calcular el
% error de clasificación errónea estimado para el mismo.

    pt = prune(t,'Level',bestlevel);
    view(pt,'Mode','graph')
    cost(bestlevel+1)

   % visualice las regiones asignadas a cada nivel de Calidad para el árbol podado

    figure(50)
    [groupname,node] = predict(pt,[length1 width1]);
    gscatter(length1,width1,groupname,'rgb','sod')

end
```

```
ans =

    0.2400
```

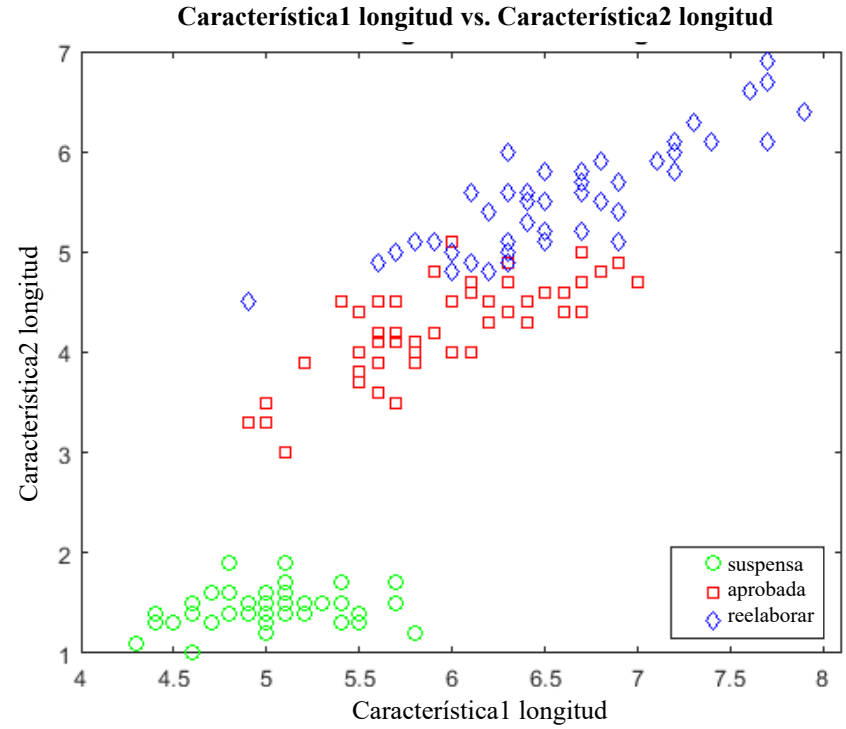


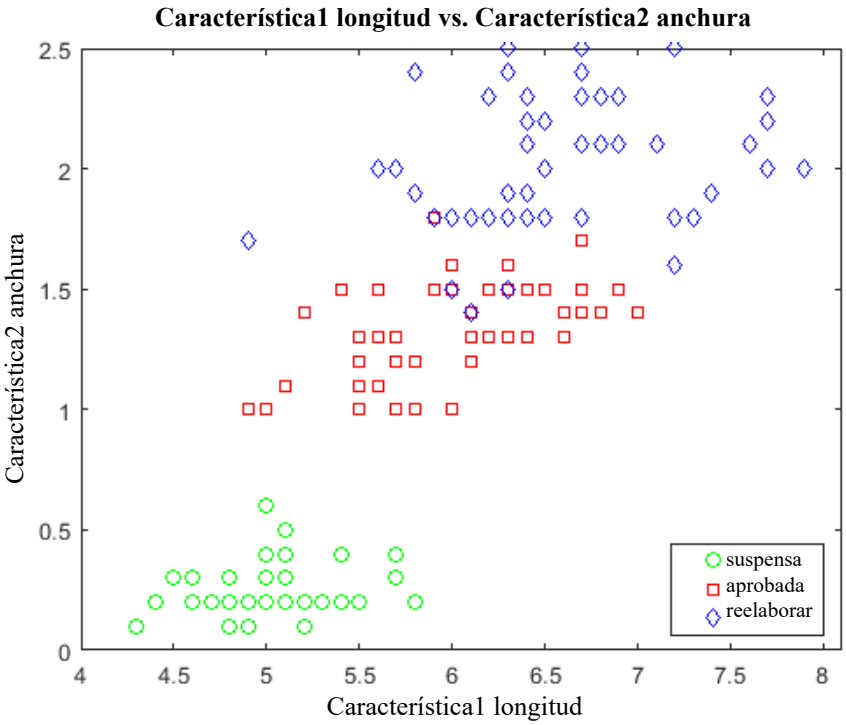


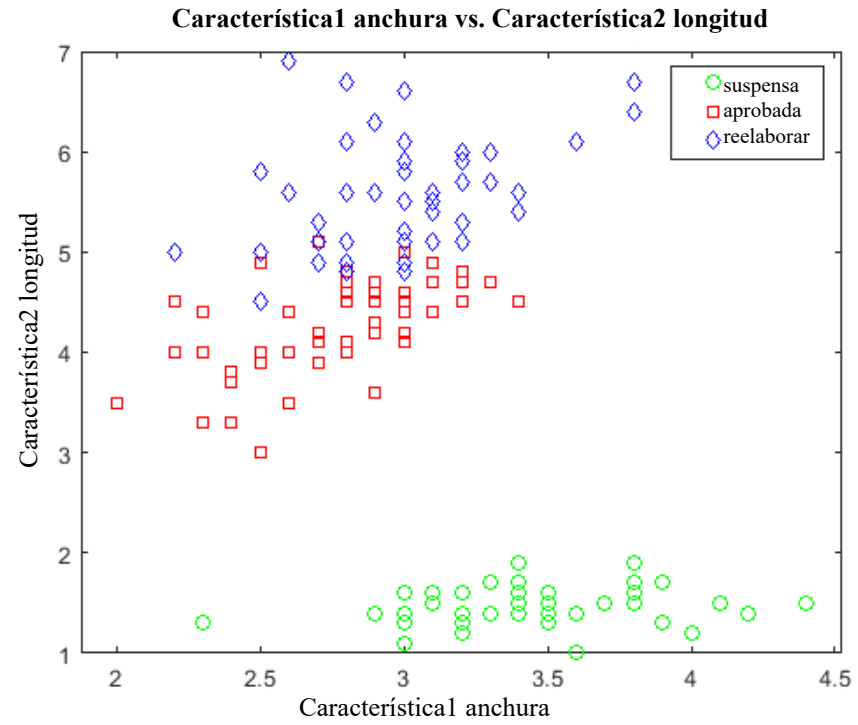
El análisis de datos que se muestra arriba no es ideal.
Utilizando las otras medidas de característica en lugar de, o además de, las medidas de Característica1 puede generar una mejor clasificación.

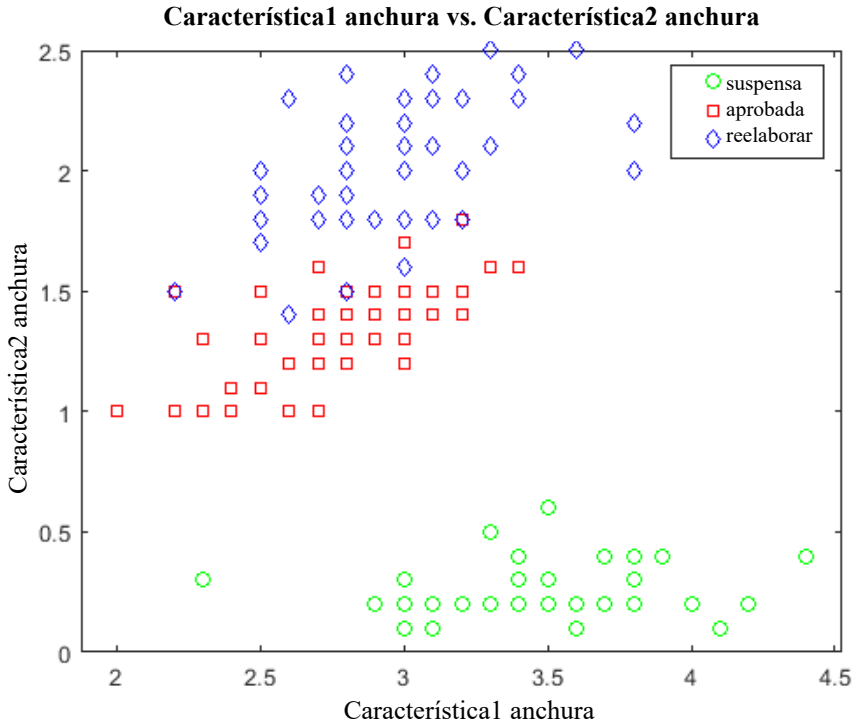
```
if(1)
    figure(100)
    gscatter(Xin(:,1), Xin(:,3), Yquality,'grb','osd');
```

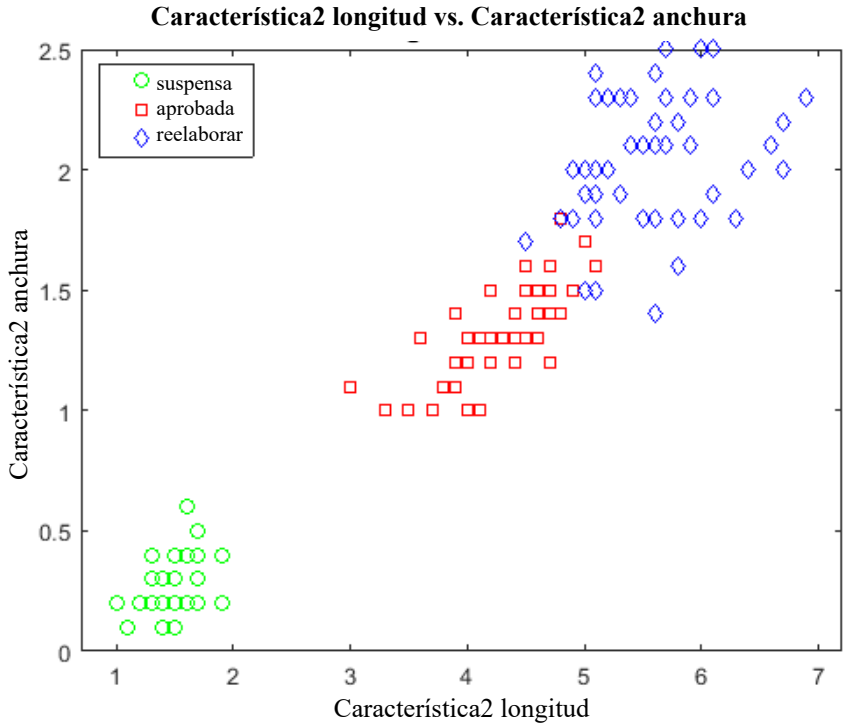
```
    xlabel('Feature1 length');
    ylabel('Feature2 length');
    N = size(Xin,1);
    title('Feature1 length vs Feature2 length')

    figure(110)
    gscatter(Xin(:,1), Xin(:,4), Yquality,'grb','osd');
    xlabel('Feature1 length');
    ylabel('Feature2 width');
    N = size(Xin,1);
    title('Feature1 length vs Feature2 width')

    figure(120)
    gscatter(Xin(:,2), Xin(:,3), Yquality,'grb','osd');
    xlabel('Feature1 width');
    ylabel('Feature2 length');
    N = size(Xin,1);
    title('Feature1 width vs Feature2 length')

    figure(130)
    gscatter(Xin(:,2), Xin(:,4), Yquality,'grb','osd');
    xlabel('Feature1 width');
    ylabel('Feature2 width');
    N = size(Xin,1);
    title('Feature1 width vs Feature2 width')

    figure(140)
    gscatter(Xin(:,3), Xin(:,4), Yquality,'grb','osd');
    xlabel('Feature2 length');
    ylabel('Feature2 width');
    N = size(Xin,1);
    title('Feature2 length vs Feature2 width')

end

% Además, puede que diferentes algoritmos de clasificación funcionen mejor.

% Por lo general, es importante realizar el análisis en otros
% conjuntos de datos y comparar distintos algoritmos.

% "TreeBagger" (bolsa de árboles de decisión) realiza agregación de bootstrap
para un conjunto de árboles
% de decisiones...
```


Característica1 longitud vs. Característica2 longitud
Característica2 longitud
Característica1 longitud
suspensa
aprobada
reelaborar


Característica1 longitud vs. Característica2 anchura
Característica2 anchura
Característica1 longitud
suspensa
aprobada
reelaborar


Característica1 anchura vs. Característica2 longitud
suspensa
aprobada
reelaborar
Característica2 longitud
Característica1 anchura
1
2
3
4
5
6
7
2
2.5
3
3.5
4
4.5


Característica1 anchura vs. Característica2 anchura
suspensa
aprobada
reelaborar
Característica2 anchura
Característica1 anchura
0
0.5
1
1.5
2
2.5
2
2.5
3
3.5
4
4.5



Published with MATLAB® R2018b