Question 1

(a)

Gini Index: Wind

In [2]:
```
parent_gini_wind = 1-((8/14)**2+(6/14)**2)
left_child_gini_weakwind = 1-((6/7)**2+(1/7)**2)
right_child_gini_strongwind = 1-((2/7)**2+(5/7)**2)
```

In [3]:
```
gini_index_gain_wind = parent_gini_wind-(7/14*left_child_gini_weakwind+
print("Gini Index for strong wind is", right_child_gini_strongwind,
     "\nGini Index for weak wind is", left_child_gini_weakwind,
     "\nGini Index gain is", gini_index_gain_wind)
```

```
Gini Index for strong wind is 0.40816326530612246
Gini Index for weak wind is 0.24489795918367352
Gini Index gain is 0.16326530612244905
```

Gini Index: Humidity

In [4]:
```
parent_gini_humidity = 1-((8/14)**2+(6/14)**2)
left_child_gini_high = 1-((4/9)**2+(5/9)**2)
right_child_gini_normal = 1-((4/5)**2+(1/5)**2)
```

In [5]:
```
gini_index_gain_humidity = parent_gini_humidity-(9/14*left_child_gini_h
print("Gini Index for high humidity is", left_child_gini_high,
     "\nGini Index for normal humidity is", right_child_gini_normal,
     "\nGini Index gain is", gini_index_gain_humidity)
```

```
Gini Index for high humidity is 0.49382716049382713
Gini Index for normal humidity is 0.31999999999999984
Gini Index gain is 0.05804988662131538
```

(b)

Wind feature will provide better Gini Index Gain because it has **higher** value (the higher Gini Index the better split).

(c)

Entropy: Wind

In [6]:
```python
import math
```

In [7]:
```python
parent_entropy = -8/14*math.log2(8/14)-6/14*math.log2(6/14)
left_child_entroy_weakwind = -6/7*math.log2(6/7)-1/7*math.log2(1/7)
right_child_entroy_strongwind = -2/7*math.log2(2/7)-5/7*math.log2(5/7)
```

In [8]:
```python
entropy_wind = parent_entropy-(7/14*left_child_entroy_weakwind+7/14*rig
print("Entropy for strong wind is", right_child_entroy_strongwind,
     "\nEntropy for weak wind is", left_child_entroy_weakwind,
     "\Information gain is", entropy_wind)
```

```
Entropy for strong wind is 0.863120568566631
Entropy for weak wind is 0.5916727785823275 \Information gain is 0.257831
4624597723
```

Entropy: Humidity

In [9]:
```python
#parent_entropy = -8/14*math.log2(8/14)-6/14*math.log2(6/14)
left_child_entroy_high = -4/9*math.log2(4/9)-5/9*math.log2(5/9)
right_child_entroy_normal = -4/5*math.log2(4/5)-1/5*math.log2(1/5)
```

In [10]:
```python
entropy_humidity = parent_entropy-(9/14*left_child_entroy_high+5/14*rig
print("Entropy for high humidity is", left_child_entroy_high,
     "\nEntropy for normal humidity is", right_child_entroy_normal,
     "\nInformation gain is", entropy_humidity)
```

```
Entropy for high humidity is 0.9910760598382222
Entropy for normal humidity is 0.7219280948873623
Information gain is 0.09027634939276485
```

(d)

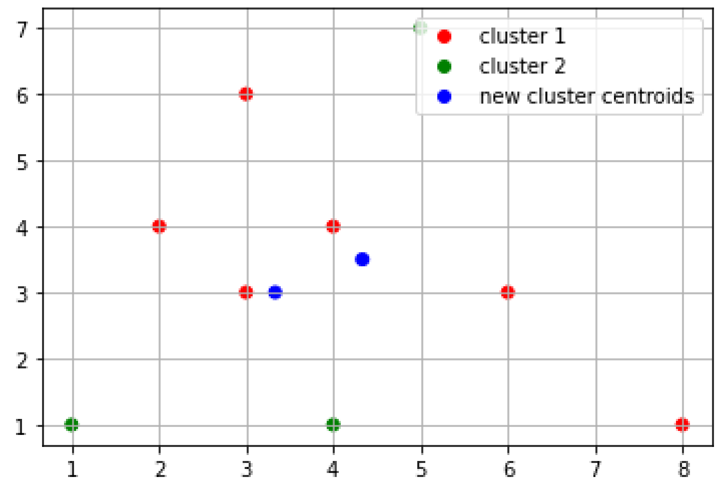
Wind feature will provide better Information Gain because it has **higher** entropy value (the higher the better means we gain more information).

Question 2

In [11]:
```
import matplotlib.pyplot as plt
import matplotlib.colors
```

In [19]:
```
x1 = [1,2,4,6,5,8,4,3,3]
x2 = [1,4,1,3,7,1,4,6,3]
centroid1=[2,2]
centroid2=[5,5]

def kmeans(x,y):

    labels = []
    legends = ['cluster 1', 'cluster 2', 'new centroid']

    for i in range(len(x)):
        d1 = (x[i]-centroid1[0])^2 + (y[i]-centroid1[1])^2
        d2 = (x[i]-centroid2[0])^2 + (y[i]-centroid2[1])^2

        if math.exp(d1) / (math.exp(d1)+math.exp(d2)) > 0.5:
            labels.append(0)
        else:
            labels. append(1)


    for i in range(2):
        new_c1 = 0
        new_c2 = 0

        for j in range(len(x)):
            if labels[j] == i:
                new_c1 += x[j]
                new_c2 += y[j]

        new_c1 = new_c1 / labels.count(i)
        new_c2 = new_c2 / labels.count(i)

        x.append(new_c1)
        y.append(new_c2)
        labels.append(2)
        print("new clusters centroids:(", round(new_c1,1), round(new_c2
        #print(labels)

    cmap = matplotlib.colors.ListedColormap(['r','g','b'])
    legends = ['cluster 1', 'cluster 2', 'new cluster centroids']
    scatter = plt.scatter(x1,x2, c=labels, cmap=cmap)
    plt.grid()
    plt.legend(handles = scatter.legend_elements()[0], labels=legends,
    plt.show()
kmeans(x1,x2)
```

```
new clusters centroids:( 4.3 3.5 )
new clusters centroids:( 3.3 3.0 )
```



Question 3

(a)

It will not affect decision boundary too much because the decision boundary in SVM is determined by the support vectors, which are the data points **closest to the margin.**

(b)

Hard Margin SVM:

Hard margin SVM aims to find a decision boundary that **completely separates the classes without allowing any misclassified points.** If the data is not linearly separable or there are outliers, hard margin SVM may fail to find a feasible solution.

Soft Margin SVM:

Soft margin SVM relaxes the strict requirement of perfect separation by **allowing some misclassified or overlapping points.**

The C parameter controls the trade-off between allowing misclassifications and maintaining a wider margin. A smaller C value allows for more misclassifications and a wider margin (soft margin SVM), while a larger C value enforces stricter classification and a narrower margin (hard margin SVM).

(c)

5 support vector

Hard margin SVM will completely separates two classes. After removing circled +, the closest + points to "-" class will become 3 instead of 1. Also add 2 - class support vectors, we will have 5 support vectors in total.

Question 4

logistic regression:

$$\ln(\frac{P(Y=1|X)}{1-P(Y=1|X)}) = \beta_0 + \beta_1 X$$

Multinomial case have K classes: for $i \in 1, 2..., K-1$

$$\ln(\frac{P(Y=i|X)}{P(Y=K|X)}) = \beta_{0,i} + \beta_{1,i} X$$

$$P(Y=i|X) = e^{\beta_{0,i}+\beta_{1,i}X} P(Y=K|X)(1)$$

Due to: $\sum_{i=1}^{K} P(Y=i|X) = 1$

$$\sum_{i=1}^{K} P(Y=K|X) * e^{\beta_{0,i}+\beta_{1,i}X} = 1$$

$$P(Y=K|X) \sum_{i=1}^{K} e^{\beta_{0,i}+\beta_{1,i}X} = 1$$

When k=i, $\ln(\frac{P(i=K|X)}{P(K|X)}) = \ln(1)$, also $e^{\ln(1)} = 1$, so

$$P(Y=K|X) = \frac{1}{1+\sum_{i=1}^{K-1} e^{\beta_{0,i}+\beta_{1,i}X}}(2)$$

replace (2) into (1):

$$P(Y=i|X) = \frac{e^{\beta_{0,i}+\beta_{1,i}X}}{1+\sum_{i=1}^{K-1} e^{\beta_{0,i}+\beta_{1,i}X}}(i \in 1, 2..., K-1)$$

For $1 \leq i \geq K$:

$$\ln P(Y=i|X) = \beta_{0,i} + \beta_{1,i} X - \ln Z$$

Because normalization term with Z in it has probabilities sum up to 1:

$$P(Y=i|X) = \frac{e^{\beta_{0,i}+\beta_{1,i}X}}{\sum_{i=1}^{K} e^{\beta_{0,i}+\beta_{1,i}X}} 1 \leq i \leq K$$

(b)

$$P(Y=1|X) = \frac{e^{-0.2+0.06*5}}{e^{-0.2+0.06*5}+e^{0.2+0.04*5}+e^{0.3+0.5*5}} = 0.058$$

$$P(Y=2|X) = \frac{e^{0.2+0.04*5}}{e^{-0.2+0.06*5}+e^{0.2+0.04*5}+e^{0.3+0.5*5}} = 0.078$$

$$P(Y=3|X) = \frac{e^{0.3+0.5*5}}{e^{-0.2+0.06*5}+e^{0.2+0.04*5}+e^{0.3+0.5*5}} = 0.864$$

So, test point will be assigned to **class 3**

Question 5

(a)

False:

For regression trees, we pick the feature and split point that **minimizes** the mean squared error (MSE).

(b)

False: K-mean algorithm is sensitive to the initial placement of centroids and can converge to different solutions depending on the initial configuration.

Different initializations can result in different final cluster assignments and centroid locations. To mitigate this sensitivity, K-means is often run multiple times with different random initializations, and the solution with the lowest objective function value is chosen as the final result. This approach helps reduce the impact of random initialization on the stability and quality of the clustering solution.

(c)

True: The objective of agglomerative clustering is to minimize the overall distortion.

(d)

False:

In soft margin SVM, the constant λ, the regularization parameter or penalty parameter, controls the trade-off between maximizing the margin and allowing some training points to violate the margin or be misclassified.

A larger value of λ imposes a stronger penalty for violating the margin or misclassifying points, leading to a more strict classification. This can result in a **narrower margin** as the algorithm tries to minimize the errors and enforce a stricter separation between classes.

(e)

True:

The purpose of using a random forest of shallow decision trees learned on bootstrapped samples is indeed to avoid overfitting. Random forests are an ensemble learning method that combines multiple decision trees to make predictions. By constructing a forest of trees and training each tree on a different bootstrapped sample of the data, the individual trees are encouraged to have low correlation with each other.