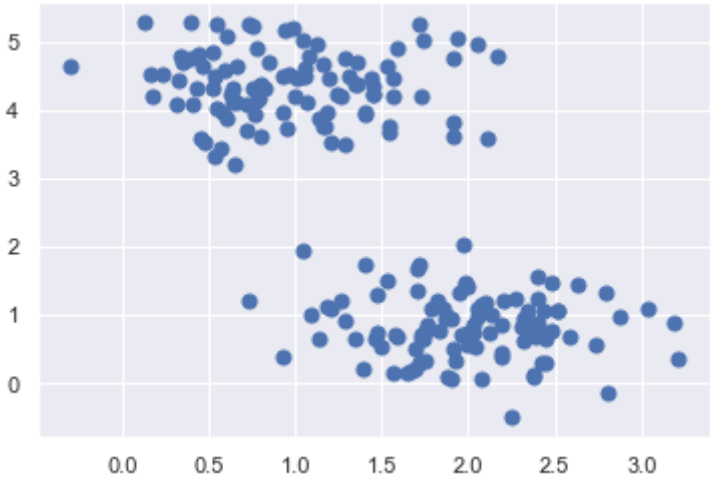
In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```

Consider a small "city" of people.

- Each point represents a person
- Friendships are formed entirely based on how close they live to each other

Could you put these people into communities?

In [2]:
```
from sklearn.datasets.samples_generator import make_blobs
X, y_true = make_blobs(n_samples=200, centers=2,
                       cluster_std=0.50, random_state=0)
plt.scatter(X[:, 0], X[:, 1], s=50);
```



How would you tell a program to do what you did visually?

Remember how the computer "sees" these points

In [3]:
```
# Print first 15 points
print(X[:15, :])
```

```
[[2.43859911 1.07581007]
 [1.85554301 1.0826916 ]
 [2.58952222 0.67097076]
 [1.73654901 0.69902775]
 [1.74265969 5.03846671]
 [0.64003985 4.12401075]
 [1.04829186 5.03092408]
 [0.5323772  3.31338909]
 [1.98882723 0.74876822]
 [0.16117091 4.53517846]
 [1.7571105  0.87138001]
 [1.28486901 0.92929466]
 [1.16448284 3.75408693]
 [0.3498724  4.69253251]
 [2.10413001 1.1891405 ]]
```

Brief aside on multi-dimensional numpy arrays

- Shape
- Indexing
- Boolean indexing
- Slicing

In [4]:
```
# Shape
print('The shape of the array is %s' % (str(X.shape)))
print('The number of samples is %d' % X.shape[0])
print('The number of dimensions is %d' % X.shape[1])
```

```
The shape of the array is (200, 2)
The number of samples is 200
The number of dimensions is 2
```

In [5]:
```
# Indexing
i = 4
j = 1
print('The %d-th dimension of sample %d is %g' % (j+1, i+1, X[i, j]))

temp = -10*np.arange(10) # numbers 0-9
print('Selecting 1st 3rd and 8th part of array')
print(temp[[1,3,8]])
```

```
The 2-th dimension of sample 5 is 5.03847
Selecting 1st 3rd and 8th part of array
[-10 -30 -80]
```

In [6]:
```
# Boolean indexing
temp = -10*np.arange(10) # numbers 0-9
selection = np.zeros(temp.shape[0], dtype=bool)
print('Boolean array')
print(selection)
selection[1] = True
selection[3] = True
selection[8] = True
# Or equivalently selection[[1,3,8]] = True

print('Selecting 1st 3rd and 8th part of array via boolean array')
print(temp[selection])
```

```
Boolean array
[False False False False False False False False False False]
Selecting 1st 3rd and 8th part of array via boolean array
[-10 -30 -80]
```

In [7]:
```
# Slicing
print('The first 10 samples with all dimensions')
print(X[:10, :])
```

```
The first 10 samples with all dimensions
[[2.43859911 1.07581007]
 [1.85554301 1.0826916 ]
 [2.58952222 0.67097076]
 [1.73654901 0.69902775]
 [1.74265969 5.03846671]
 [0.64003985 4.12401075]
 [1.04829186 5.03092408]
 [0.5323772  3.31338909]
 [1.98882723 0.74876822]
 [0.16117091 4.53517846]]
```

In [8]:
```
print('The %d-th sample is:' % (i+1))
print(str(X[i, :]))
```

```
The 5-th sample is:
[1.74265969 5.03846671]
```

In [9]:
```
print('The first 10 samples for the dimension %d' % (j+1))
print(X[:10, j])
```

```
The first 10 samples for the dimension 2
[1.07581007 1.0826916  0.67097076 0.69902775 5.03846671 4.12401075
 5.03092408 3.31338909 0.74876822 4.53517846]
```

In [10]:
```
print('The last 10 samples for the dimension %d' % (j+1))
print(X[-10:, j])
```

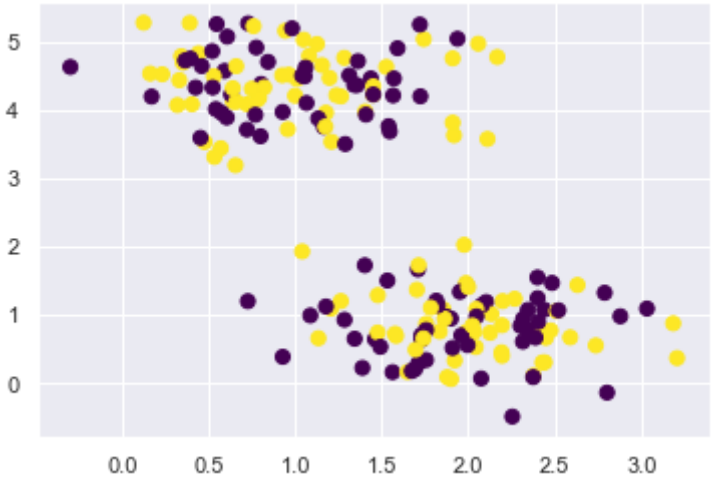
```
The last 10 samples for the dimension 2
[1.10568868 3.97204818 1.12313089 0.84858847 1.46821459 1.24503823
 4.2012082  3.57660449 4.22810872 4.75420057]
```

How do we formalize what we did visually?

- Let's assume for now that we know there are exactly *two* communities
- How can we assign each person to a community?
- Naive idea: Randomly assign points to each community

In [11]:
```
from sklearn.utils import check_random_state
def get_random_assignment(random_state=None):
    rng = check_random_state(random_state)
    y = rng.randint(2, size=X.shape[0])
    return y
y_rand = get_random_assignment(random_state=0)
plt.scatter(X[:, 0], X[:, 1], c=y_rand, s=50, cmap='viridis')
```

Out[11]:
```
<matplotlib.collections.PathCollection at 0x1a182624e0>
```



This clustering "looks" quite bad.

How can we formalize whether a particular assignment is good or bad?

- One intuition: People in a communities will be as close to each other as possible.
- Take average distance between each person in a community to every other person in the **same** community.
- Sum over all communities.

(Derive on board)