K-Nearest Neighbor (KNN)

Prof. Seungchul Lee

Industrial AI Lab.

Supervised Learning

- Given training set $\left\{\left(x^{(1)}, y^{(1)}\right), \cdots, \left(x^{(m)}, y^{(m)}\right)\right\}$

- Want to find a function f_ω with learning parameter ω
  - f_ω desired to be as close as possible to y for future (x, y)
  - i.e., $f_\omega(x) \approx y$

- Define a loss function

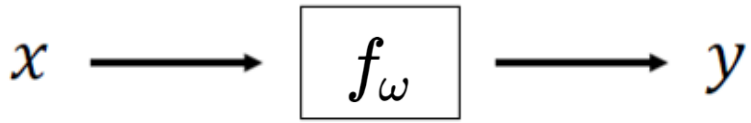
$$\ell\left(f_\omega\left(x^{(i)}\right), y^{(i)}\right)$$

- Solve the following optimization problem:

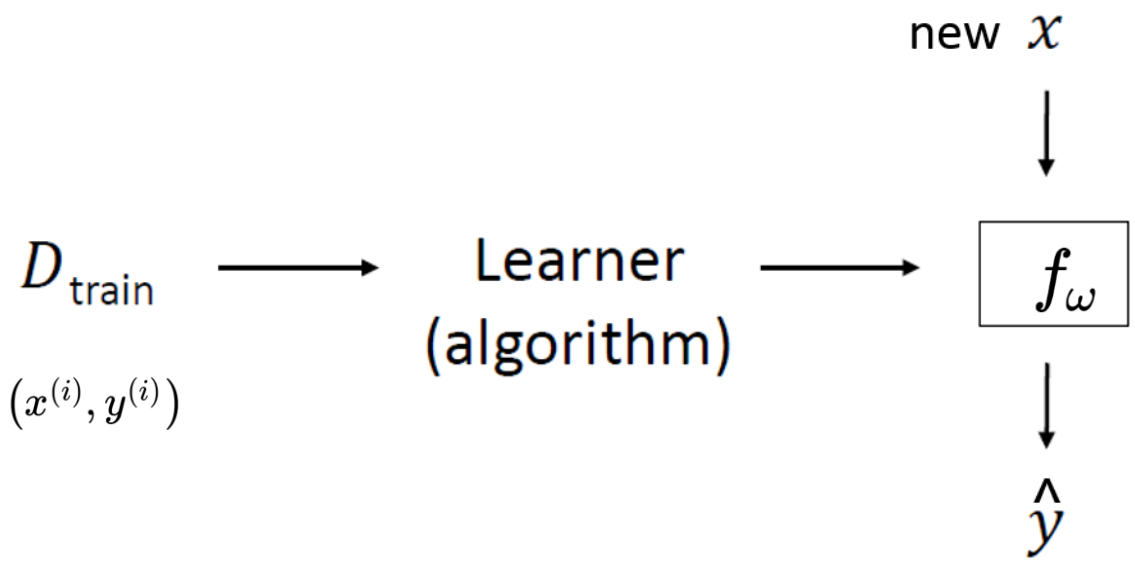
$$\begin{aligned} \text{minimize} \quad & \frac{1}{m}\sum_{i=1}^{m} \ell\left(f_\omega\left(x^{(i)}\right), y^{(i)}\right) \\ \text{subject to} \quad & \omega \in \boldsymbol{\omega} \end{aligned}$$

Supervised Learning

- Function approximation between inputs and outputs



- Once it is learned,



K-Nearest Neighbor (KNN) Regression

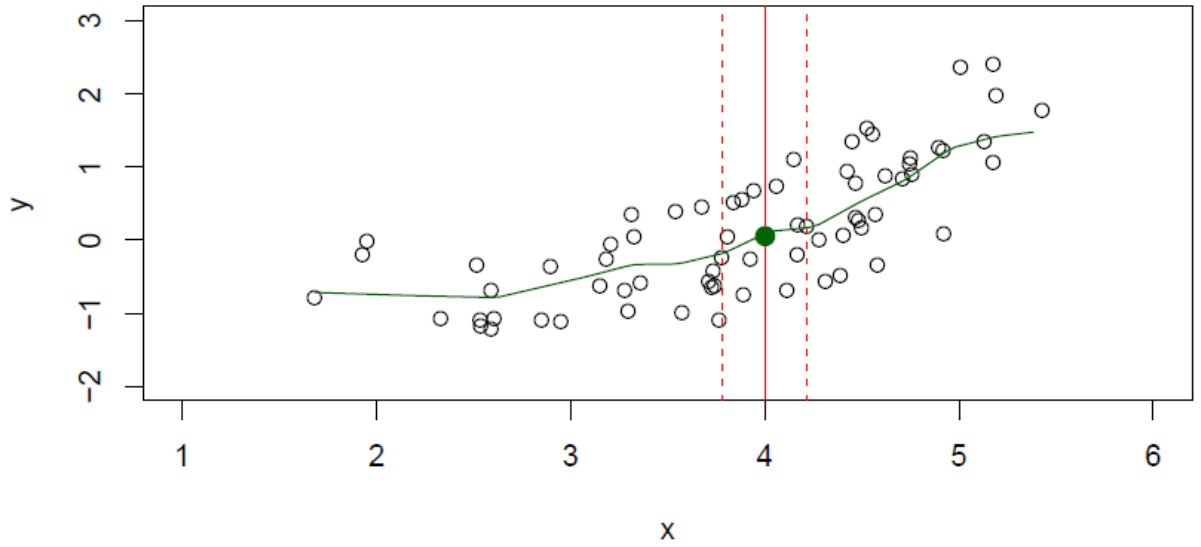
- Non-parametric method
- We write our model as

$$y = f(x) + \varepsilon$$

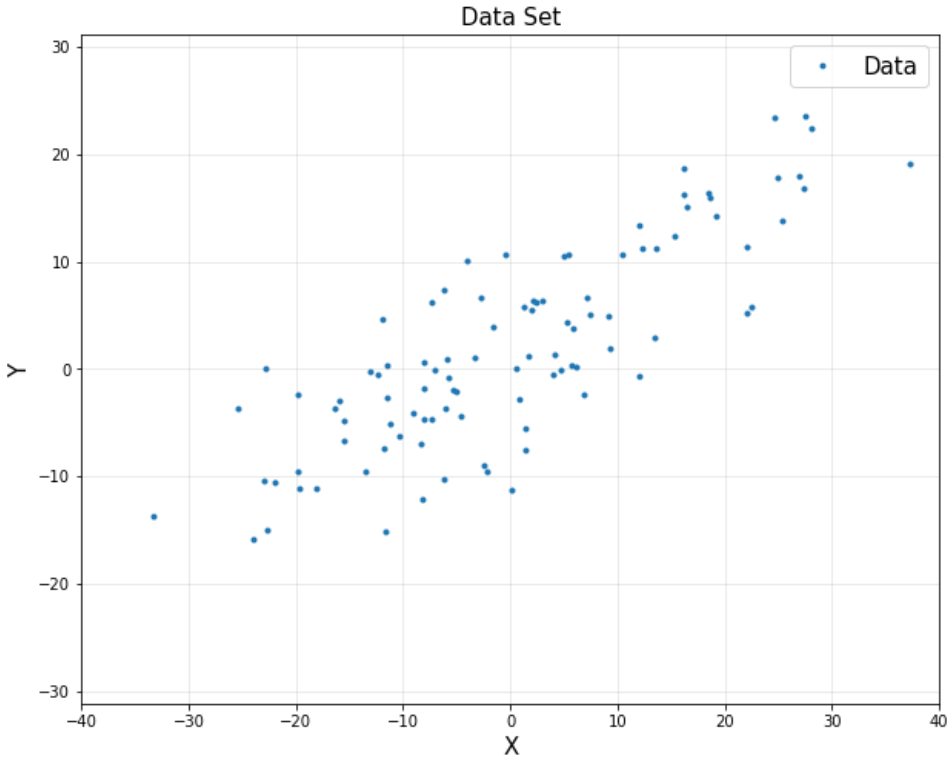
- Then, with a good f we can make predictions of y at new points x_{new}.
- One possible way so called "nearest neighbor method" is:

$$\hat{y} = \text{avg}\left(y \mid x \in \mathcal{N}(x_{\text{new}})\right)$$

where $\mathcal{N}(x)$ is some neighborhood of x

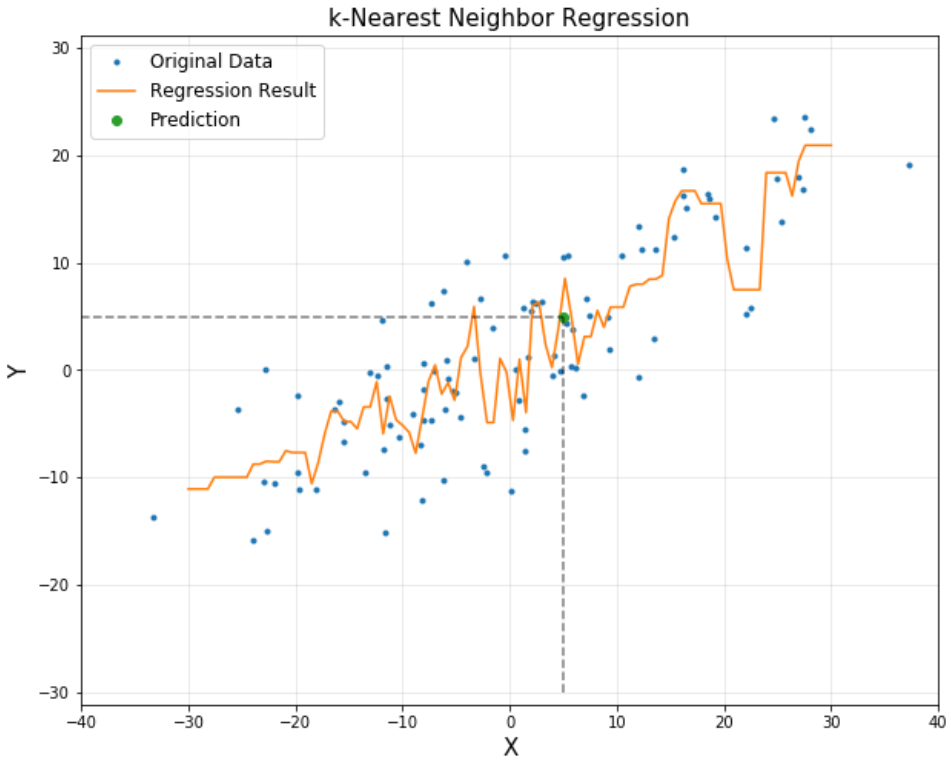


K-Nearest Neighbor (KNN) Regression



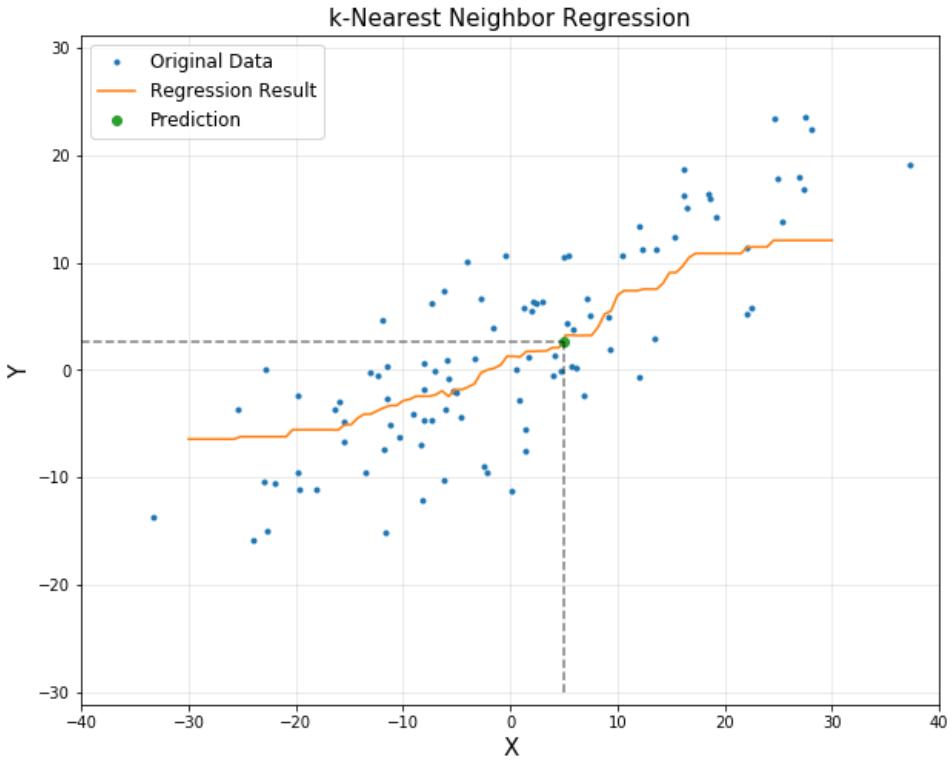
K = 3

```
from sklearn import neighbors

reg = neighbors.KNeighborsRegressor(n_neighbors = 3)
reg.fit(x, y)
```



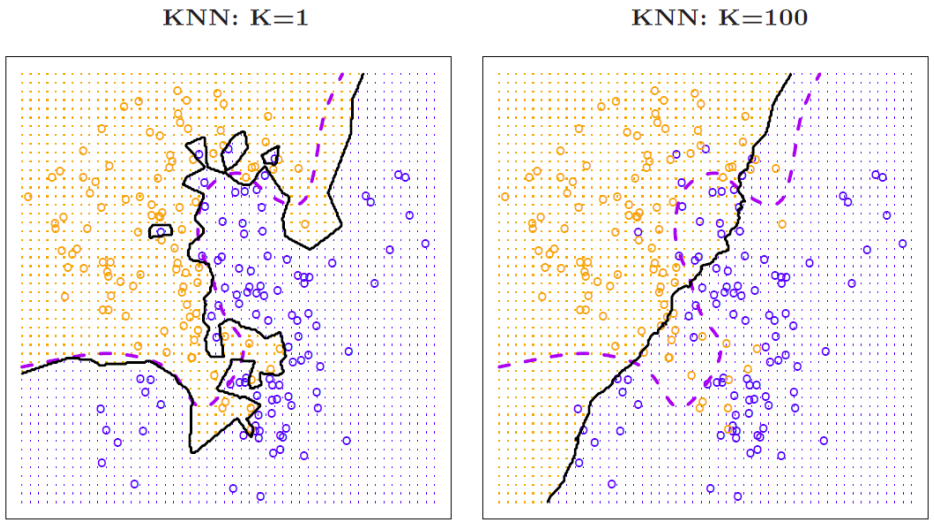
K = 31

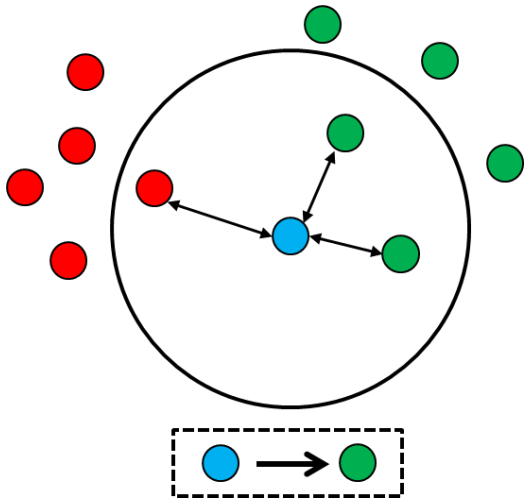
```
from sklearn import neighbors

reg = neighbors.KNeighborsRegressor(n_neighbors = 31)
reg.fit(x, y)
```



K-Nearest Neighbor (KNN) Classification

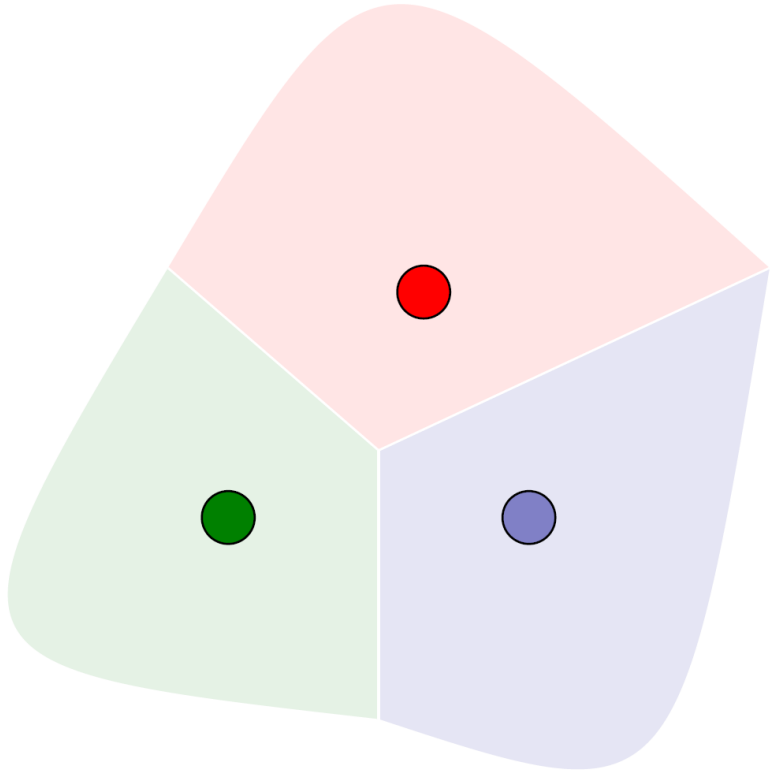
- Non-parametric method
- In k-NN classification, an object is assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small).
- If $k = 1$, then the object is simply assigned to the class of that single nearest neighbor.





K-Nearest Neighbor (KNN) Classification

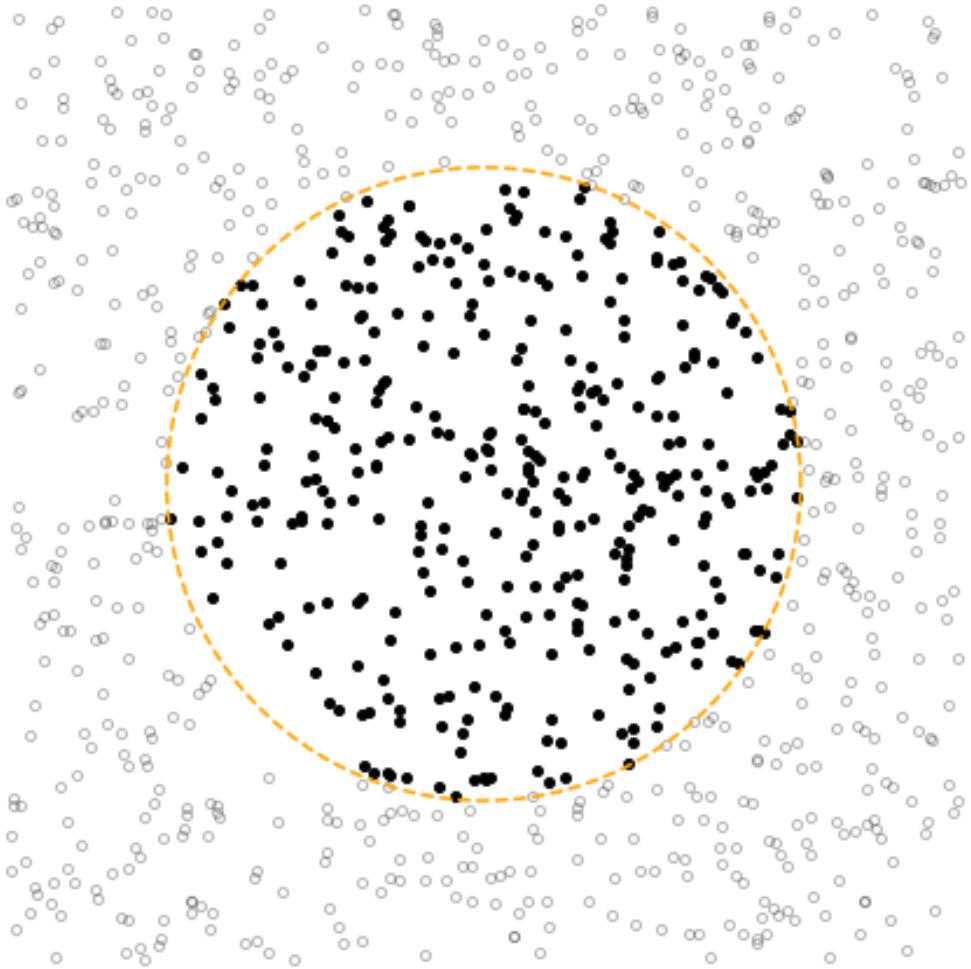
- K = 1



K = 1

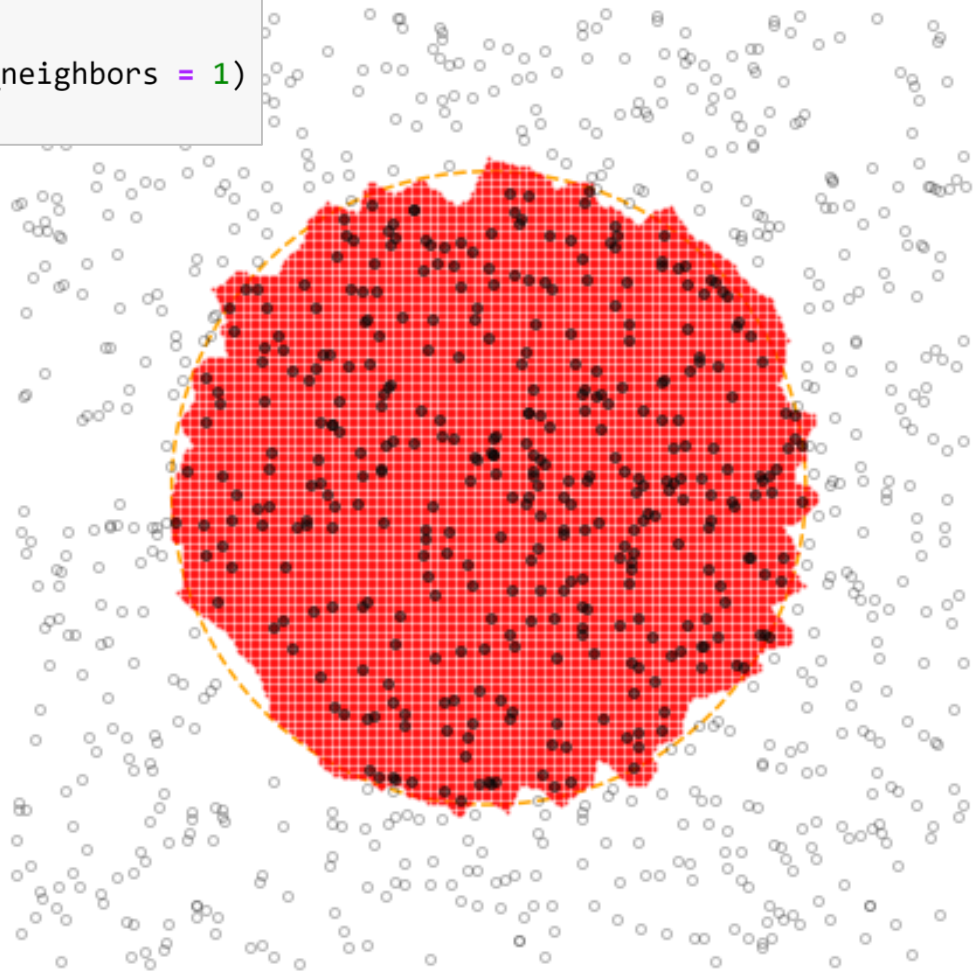
Source: Dr. Francois Fleuret at EPFL

Overfitting Example

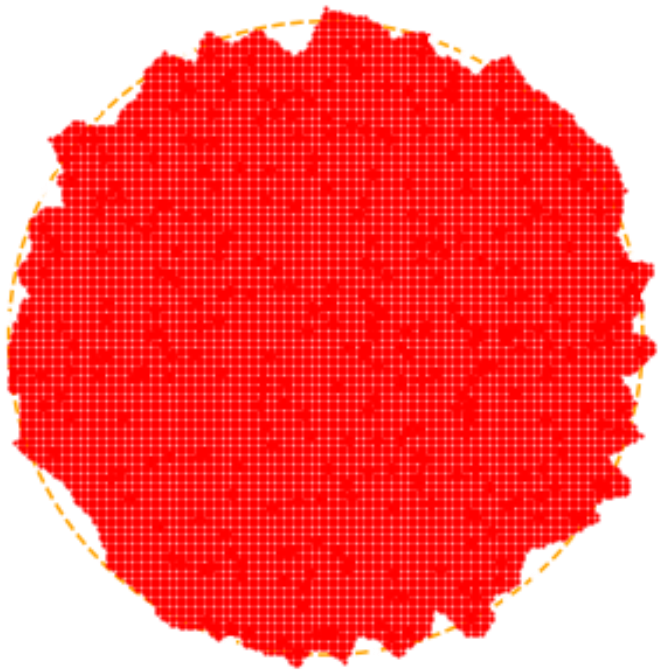


K = 1

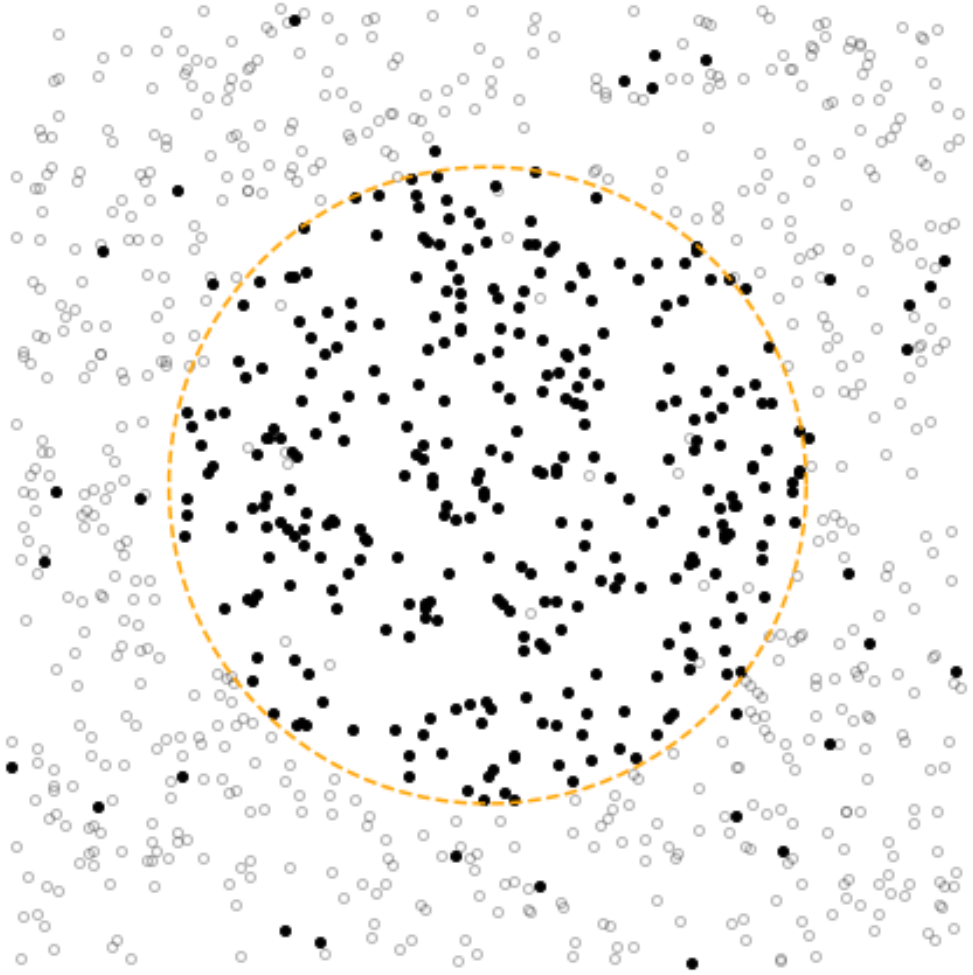
```
from sklearn import neighbors

clf = neighbors.KNeighborsClassifier(n_neighbors = 1)
clf.fit(X, np.ravel(y))
```



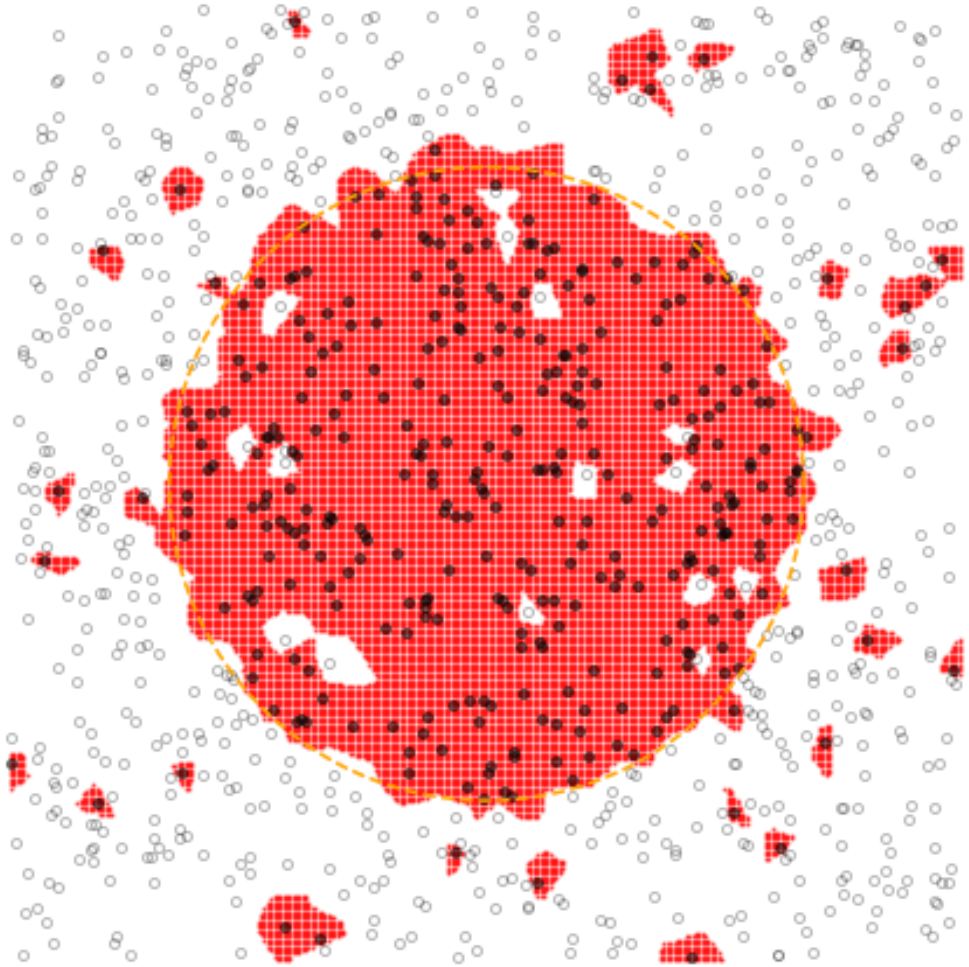
K = 1



Outliers

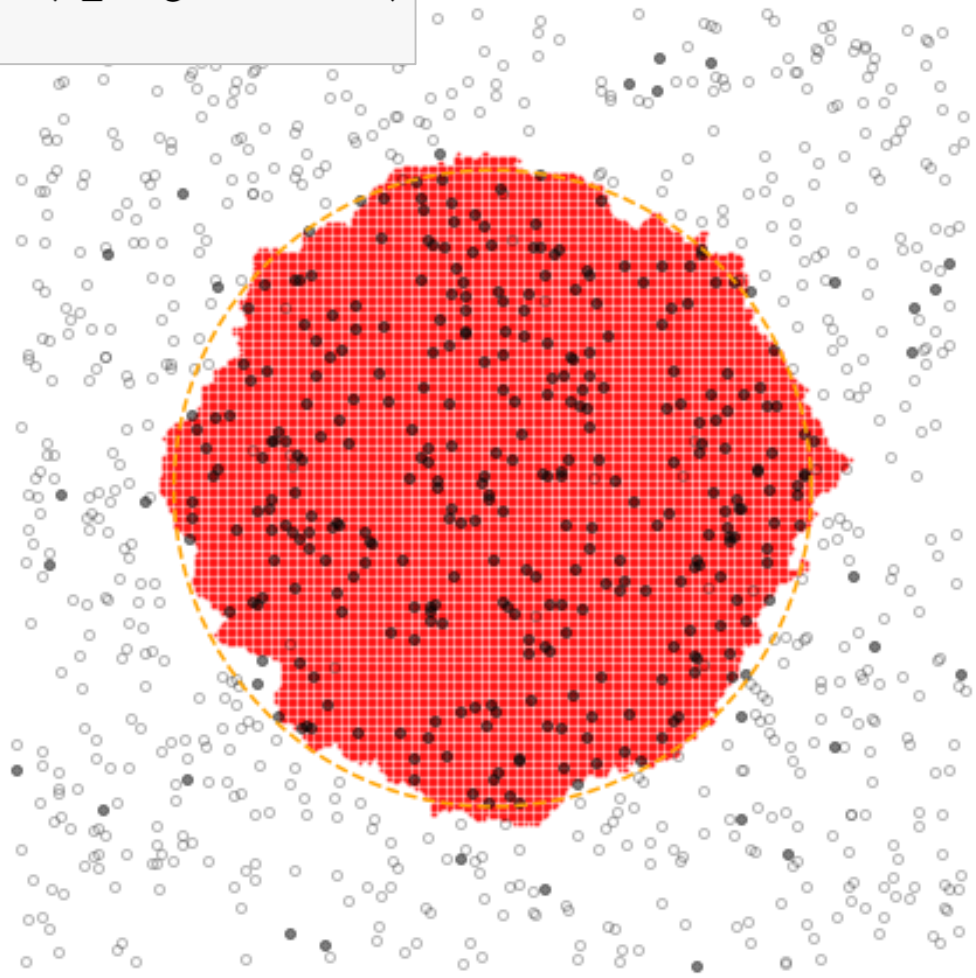


K = 1 → Too Noisy



K = 11→ Become Smooth

```
clf = neighbors.KNeighborsClassifier(n_neighbors = 11)
clf.fit(X, np.ravel(y))
```



K = 21→ Become Smoother

```
clf = neighbors.KNeighborsClassifier(n_neighbors = 21)
clf.fit(X, np.ravel(y))
```