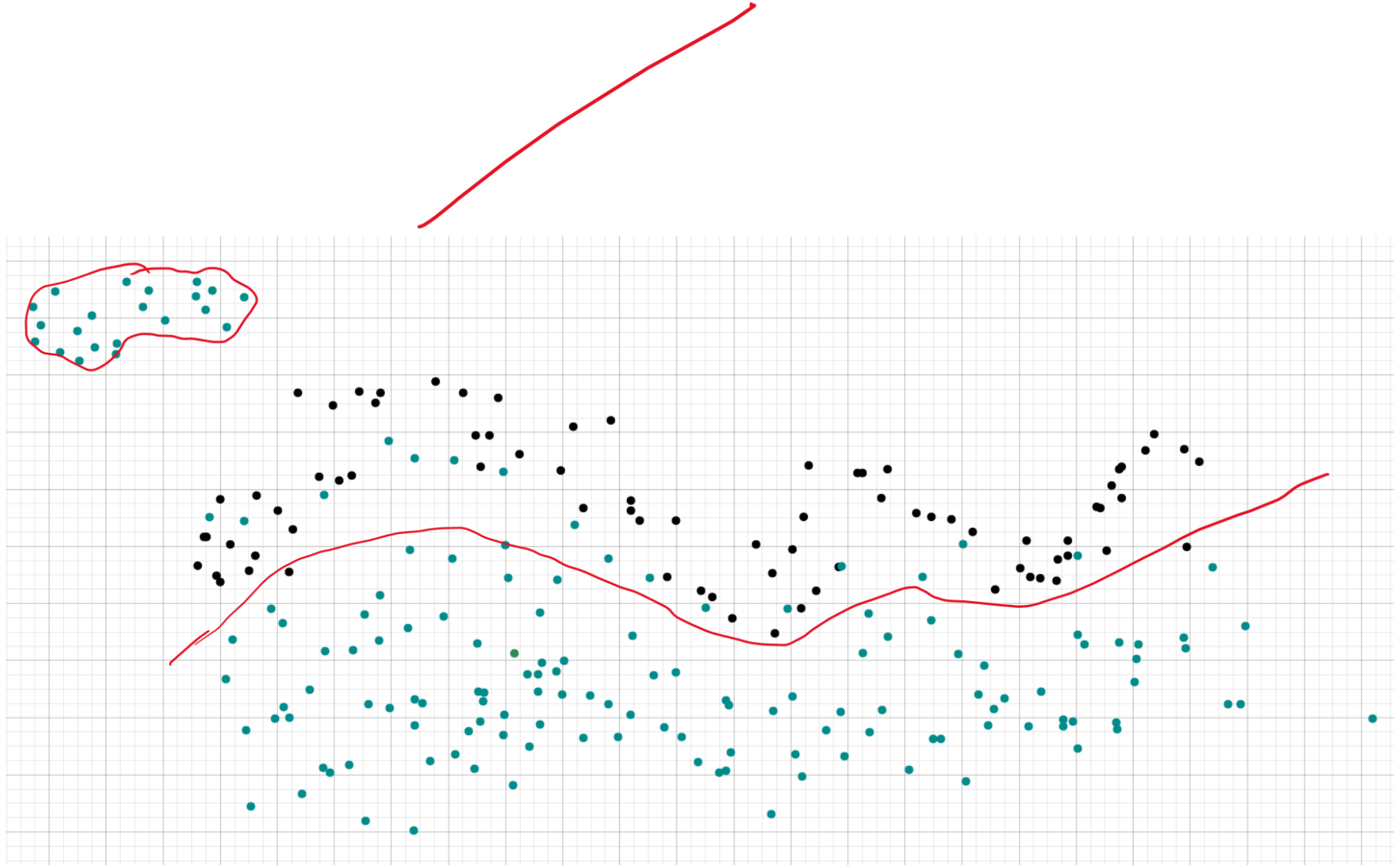
BİL 475 Örüntü Tanıma

Hafta-13:

Denetimsiz Öğrenme


Kernel SVM



$x_i = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ $y_i = +1$

$x_j = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$ $y_j = -1$

$x_j^\top \cdot x_i \cdot y_j \cdot y_i$

$[3, 4]\begin{bmatrix} 1 \\ 2 \end{bmatrix} = 11$

$11 \cdot (-1) \cdot (+1)$

$\Rightarrow -11$

Linear SVM dual formulation

$$\mathcal{L}(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i (y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1)$$

Optimality: $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \qquad \sum_{i=1}^{n} \alpha_i y_i = 0$

$$\mathcal{L}(\alpha) = \underbrace{\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_j \alpha_i y_i y_j \mathbf{x}_j^\top \mathbf{x}_i}_{\mathbf{w}^\top \mathbf{w}} - \sum_{i=1}^{n} \alpha_i y_i \underbrace{\sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j^\top \mathbf{x}_i}_{\mathbf{w}^\top} - b \underbrace{\sum_{i=1}^{n} \alpha_i y_i}_{=0} + \sum_{i=1}^{n} \alpha_i$$

$$= -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_j \alpha_i y_i y_j \mathbf{x}_j^\top \mathbf{x}_i + \sum_{i=1}^{n} \alpha_i$$

Dual linear SVM is also a quadratic program

$$\text{problem } \mathcal{D} \begin{cases} \min\limits_{\alpha \in \mathbb{R}^n} & \frac{1}{2}\alpha^\top G \alpha - \mathbf{e}^\top \alpha \\ \text{with} & \mathbf{y}^\top \alpha = 0 \\ \text{and} & 0 \le \alpha_i \qquad\qquad i = 1, n \end{cases}$$

with G a symmetric matrix $n \times n$ such that $G_{ij} = y_i y_j \mathbf{x}_j^\top \mathbf{x}_i$

$$= -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_j \alpha_i y_i y_j \mathbf{x}_j^\top \mathbf{x}_i + \sum_{i=1}^{n} \alpha_i$$

$e^{(-\sigma\|x_i - x_j\|_2^2)}$

$1 - 3 = -2$

$2 - 4 = -2$

$4 + 4 = 8$

$\Rightarrow -8$

$\Rightarrow e^{(-\sigma \cdot (-8))}$

$$\phi(\mathbf{x}) = (x_1^2, x_1x_2, x_1x_3, x_2x_1, x_2^2, x_2x_3, x_3x_1, x_3x_2, x_3^2)^T$$
$$\phi(\mathbf{y}) = (y_1^2, y_1y_2, y_1y_3, y_2y_1, y_2^2, y_2y_3, y_3y_1, y_3y_2, y_3^2)^T$$

$$\phi(\mathbf{x})^T\phi(\mathbf{y}) = \sum_{i,j=1}^{3} x_i x_j y_i y_j$$

KERNEL TRICK

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T\mathbf{y})^2$$

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$$

https://towardsdatascience.com/understanding-support-vector-machine-part-2-kernel-trick-mercers-theorem-e1e6848c6c4d

https://medium.com/@zxr.nju/what-is-the-kernel-trick-why-is-it-important-98a98db0961d

- More formally, if we have data $\mathbf{x}, \mathbf{z} \in X$ and a map $\phi : X \to \Re^N$ then

$$k(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$$

is a kernel function

$x_i = \begin{bmatrix} -1 \\ 3 \end{bmatrix}$

$x_j = \begin{bmatrix} -2 \\ -2 \end{bmatrix}$

$k(x_i, x_j) = (x_i^T \cdot x_j)^2 \Rightarrow [-1, 3] \cdot \begin{bmatrix} -2 \\ -2 \end{bmatrix} = 0 \; 2 - 6 = -4$

16

$\begin{bmatrix} 1 \\ -3 \\ -3 \\ 9 \end{bmatrix}^T \begin{bmatrix} 4 \\ -4 \\ 4 \\ 4 \end{bmatrix} \Rightarrow 4 - 12 - 12 + 36 = 16$

$\phi(x_i)^T \cdot \phi(x_j)$

$$
\begin{aligned}
e^{\frac{-1}{2\sigma^2}(x_i - x_j)^2} &= e^{\frac{-x_i^2 - x_j^2}{2\sigma^2}} \left(1 + \frac{2x_i x_j}{1!} + \frac{(2x_i x_j)^2}{2!} + ...\right) \\
&= e^{\frac{-x_i^2 - x_j^2}{2\sigma^2}} \left(1 \cdot 1 + \sqrt{\frac{2}{1!}} x_i \cdot \sqrt{\frac{2}{1!}} x_j + \sqrt{\frac{(2)^2}{2!}} (x_i)^2 \cdot \sqrt{\frac{(2)^2}{2!}} (x_j)^2 + ...\right) \\
&= \phi(x_i)^T \phi(x_j) \\
\text{where, } \phi(x) &= e^{\frac{-x^2}{2\sigma^2}} \left(1, \sqrt{\frac{2}{1!}} x, \sqrt{\frac{2^2}{2!}} x^2, ...\right)
\end{aligned}
\tag{1.25}
$$

Classifier:

$$f(\mathbf{x}) = \sum_i^N \alpha_i y_i \, \mathbf{x}_i{}^\top \mathbf{x} + b$$

$$\rightarrow f(\mathbf{x}) = \sum_i^N \alpha_i y_i \, \Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}) + b$$

Learning:

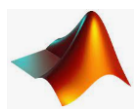
$$\max_{\alpha_i \geq 0} \sum_i \alpha_i - \frac{1}{2} \sum_{jk} \alpha_j \alpha_k y_j y_k \mathbf{x}_j{}^\top \mathbf{x}_k$$

$$\rightarrow \max_{\alpha_i \geq 0} \sum_i \alpha_i - \frac{1}{2} \sum_{jk} \alpha_j \alpha_k y_j y_k \Phi(\mathbf{x}_j)^\top \Phi(\mathbf{x}_k)$$

subject to

$$0 \leq \alpha_i \leq C \text{ for } \forall i, \text{ and } \sum_i \alpha_i y_i = 0$$

$$f(\mathbf{x}) = \sum_i^N \alpha_i y_i \, k(\mathbf{x}_i, \mathbf{x}) + b$$



Polynomial kernel

$$k(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i} \cdot \mathbf{x_j} + 1)^d$$

Polynomial kernel equation

Gaussian kernel

$$k(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right)$$

Gaussian kernel equation

$$\log\left(\frac{|x_i - x_j|^2 + \varepsilon a}{|x_i - x_j|^2 + a}\right)$$

Laplace RBF kernel

$$k(x, y) = \exp\left(-\frac{\|x - y\|}{\sigma}\right)$$

Laplace RBF kernel equation

Hyperbolic tangent kernel

$$k(\mathbf{x_i}, \mathbf{x_j}) = \tanh(\kappa \mathbf{x_i} \cdot \mathbf{x_j} + c)$$

Hyperbolic tangent kernel equation

Bessel function

$$k(x, y) = \frac{J_{v+1}(\sigma\|x - y\|)}{\|x - y\|^{-n(v+1)}}$$

Equation of Bessel function of the first kind kernel

(RBF)

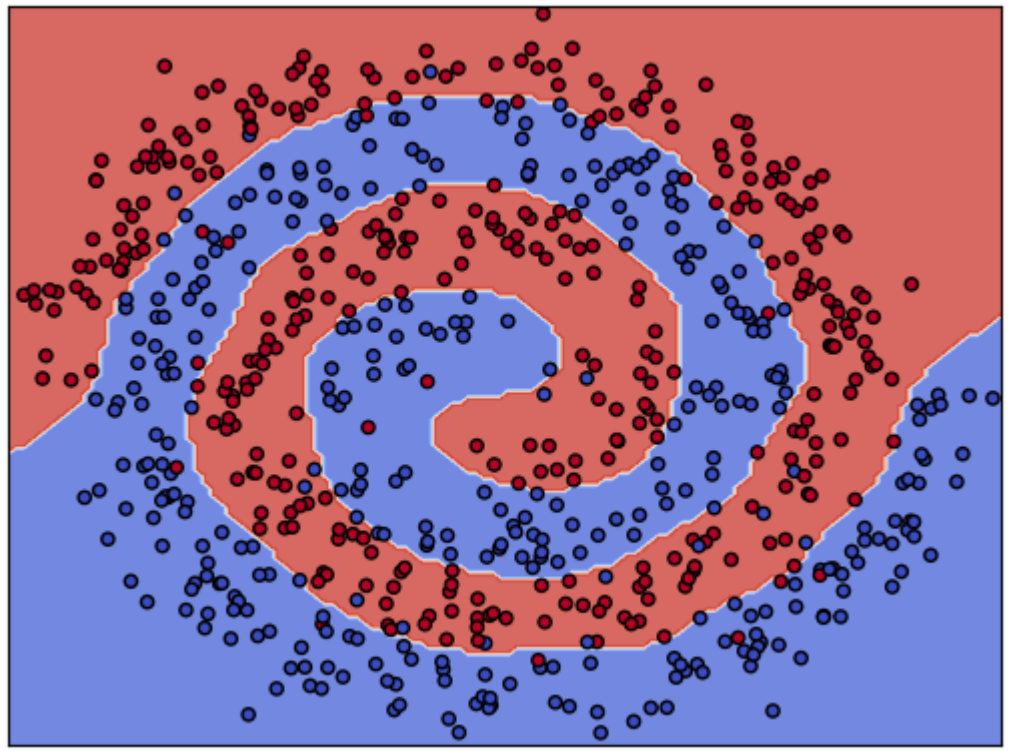
$$k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma\|\mathbf{x_i} - \mathbf{x_j}\|^2)$$

Gaussian radial basis function (RBF)

Gaussian kernel

$$k(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right)$$

Gaussian kernel equation

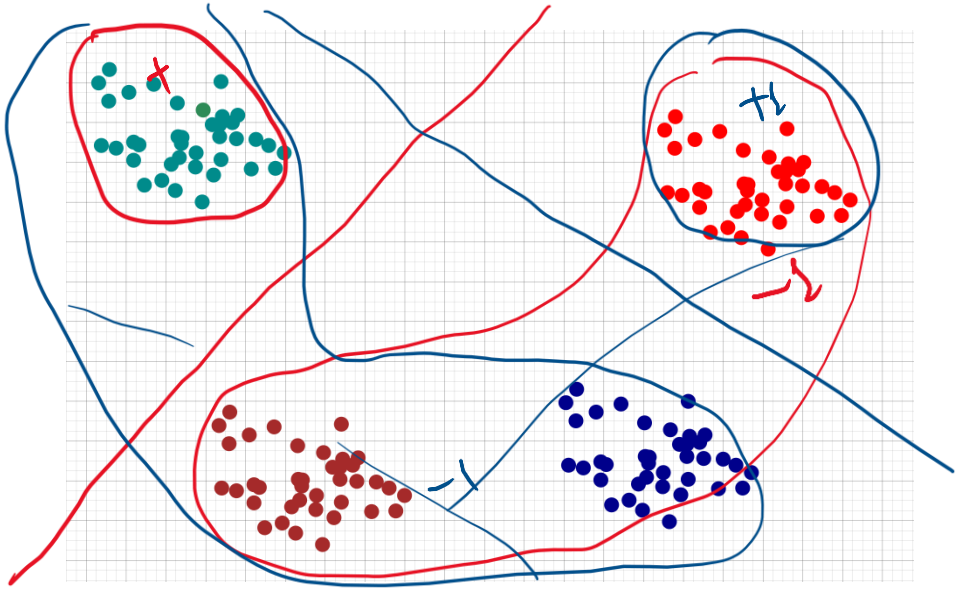


(RBF)

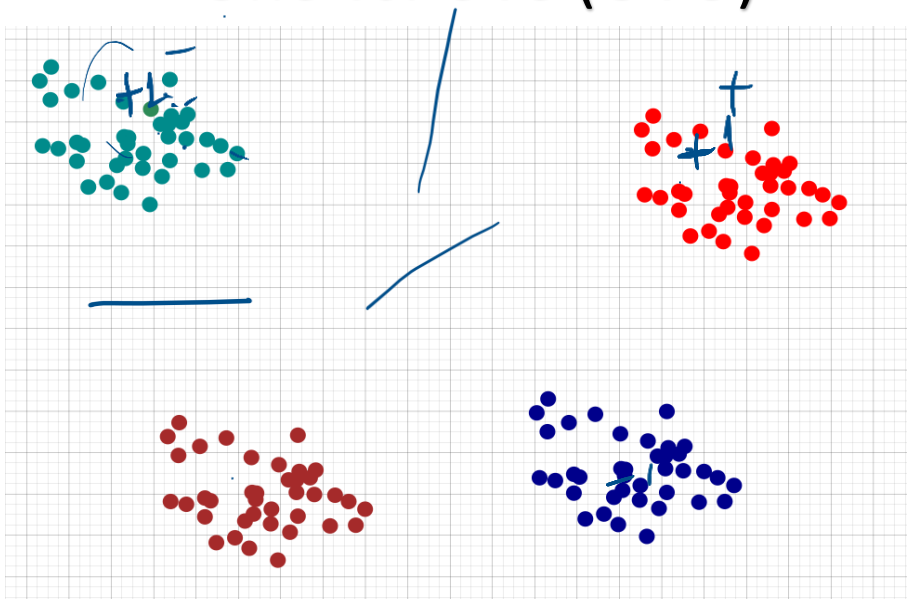
$$= \exp(-\gamma\|\mathbf{x_i} - \mathbf{x_j}\|^2)$$

radial basis function (RBF)

One vs. All (OVA)



One vs. Ove (OVO)



N_c

$$\frac{N.(N-1)}{2}$$

Parameters:

C : *float, default=1.0*

Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty.

kernel : *{'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'}, default='rbf'*

Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape `(n_samples, n_samples)`.

degree : *int, default=3*

Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.

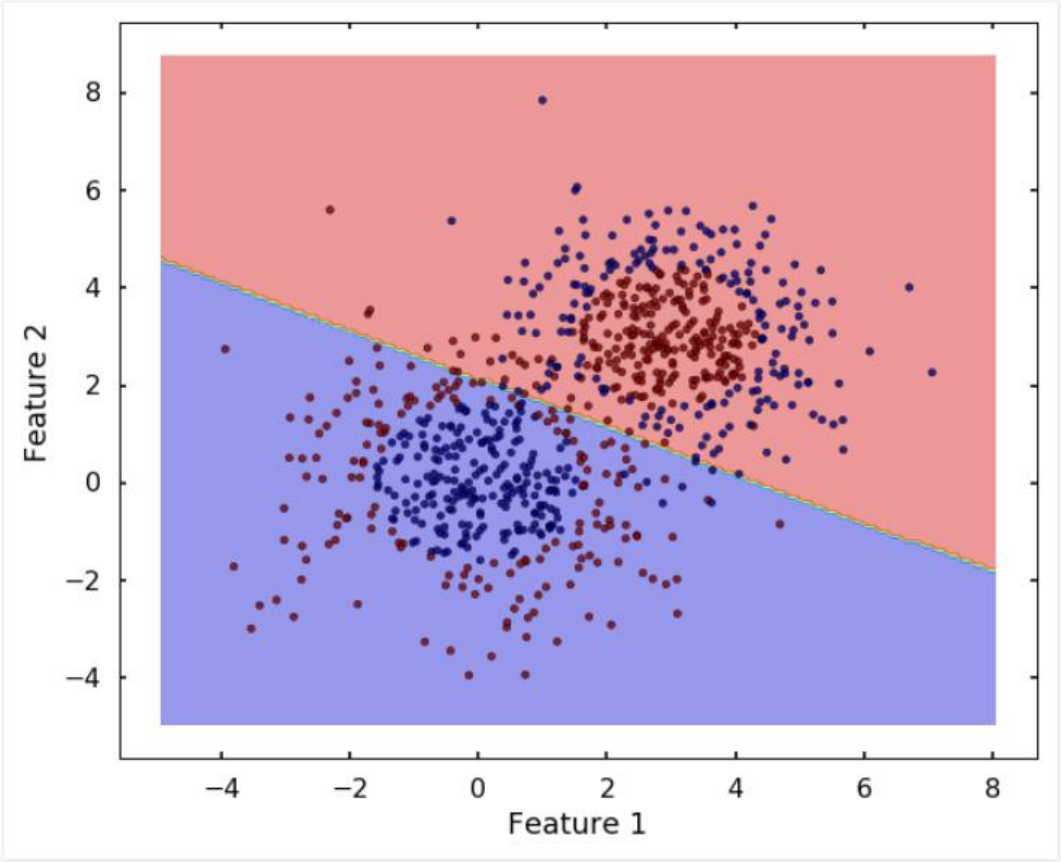
gamma : *{'scale', 'auto'} or float, default='scale'*

Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.

- if `gamma='scale'` (default) is passed then it uses 1 / (n_features * X.var()) as value of gamma,
- if 'auto', uses 1 / n_features.

decision_function_shape : *{'ovo', 'ovr'}, default='ovr'*

Whether to return a one-vs-rest ('ovr') decision function of shape (n_samples, n_classes) as all other classifiers, or the original one-vs-one ('ovo') decision function of libsvm which has shape (n_samples, n_classes * (n_classes - 1) / 2). However, one-vs-one ('ovo') is always used as multi-class strategy. The parameter is ignored for binary classification.

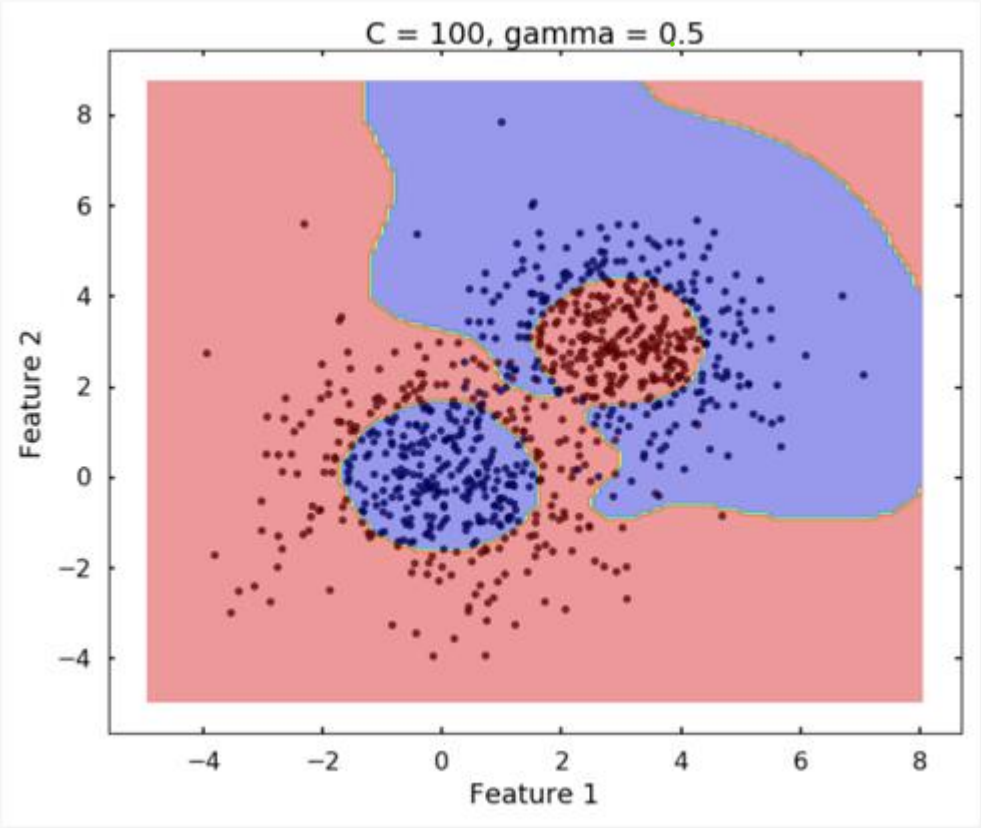



```
clf = svm.SVC(kernel='linear')
clf.fit(X,y)
```

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape=None, degree=3, gamma='auto', kernel='linear',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

```
plot_desicion_boundary(X, y, clf)
```

http://qingkaikong.blogspot.com/2016/12/machine-learning-8-support-vector.html

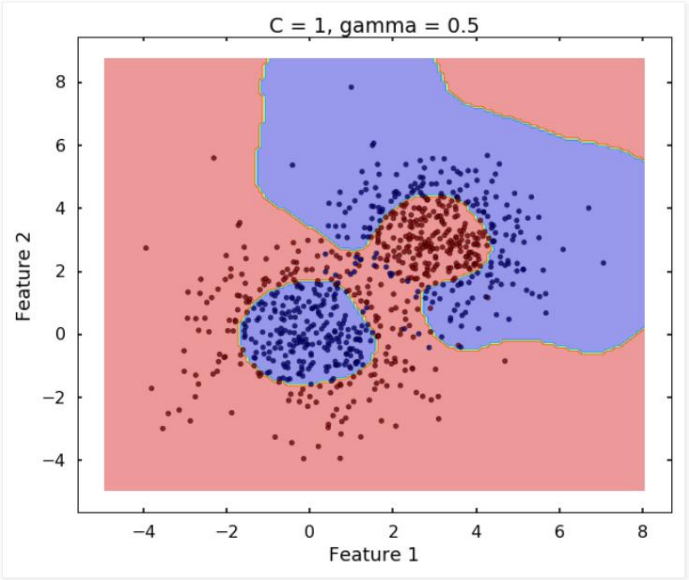


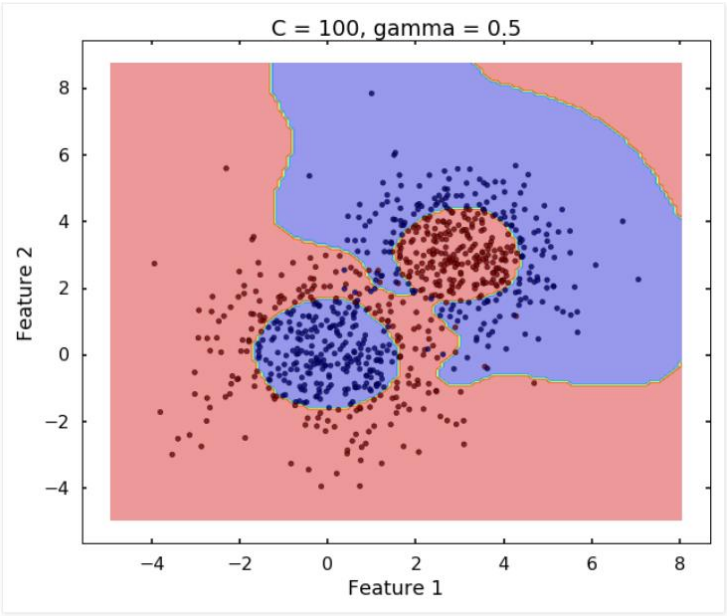

```
clf = svm.SVC(kernel='rbf')
clf.fit(X,y)
```

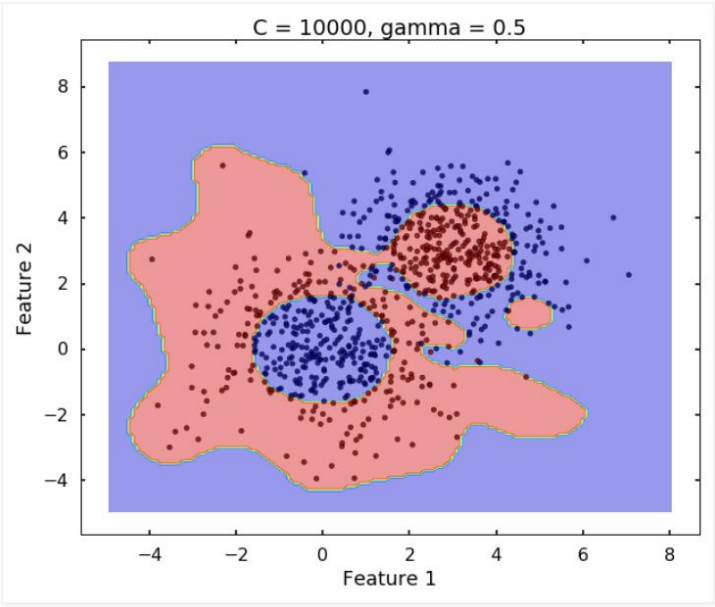
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape=None, degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

```
plot_desicion_boundary(X, y, clf)
```

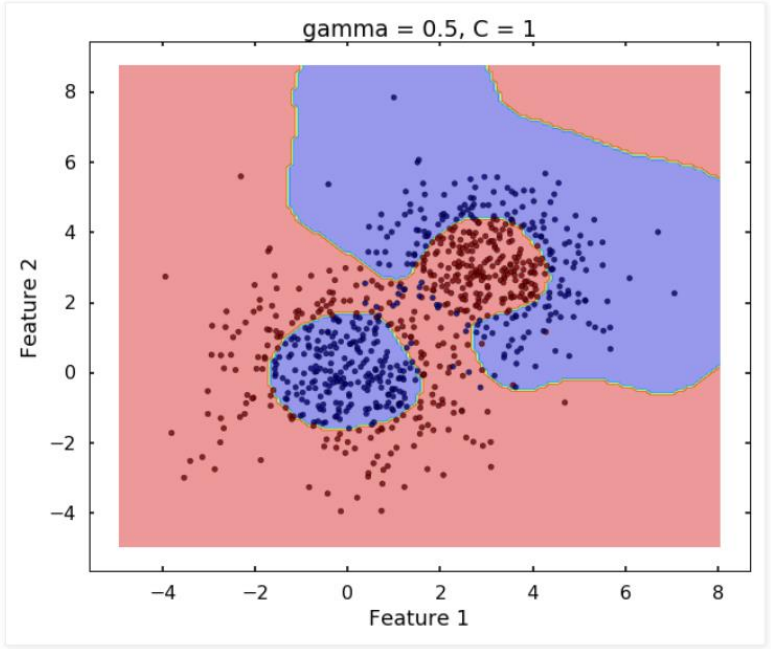
http://qingkaikong.blogspot.com/2016/12/machine-learning-8-support-vector.html

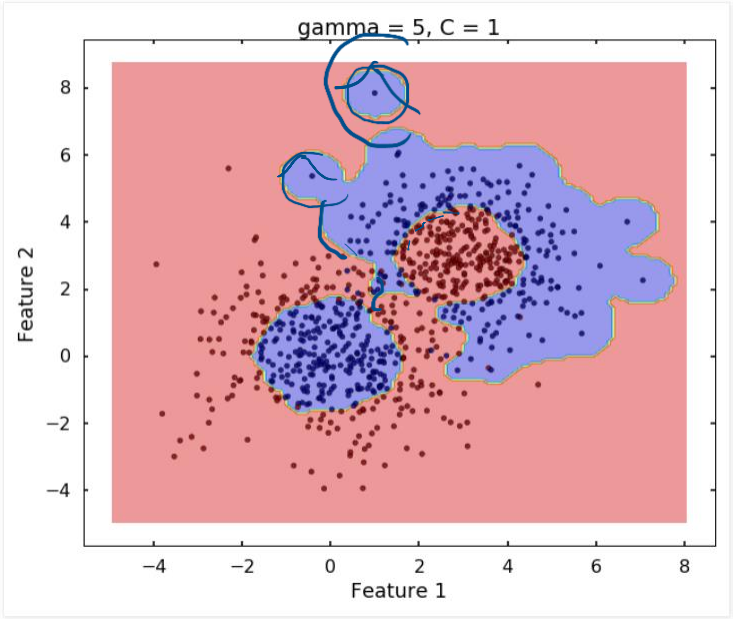


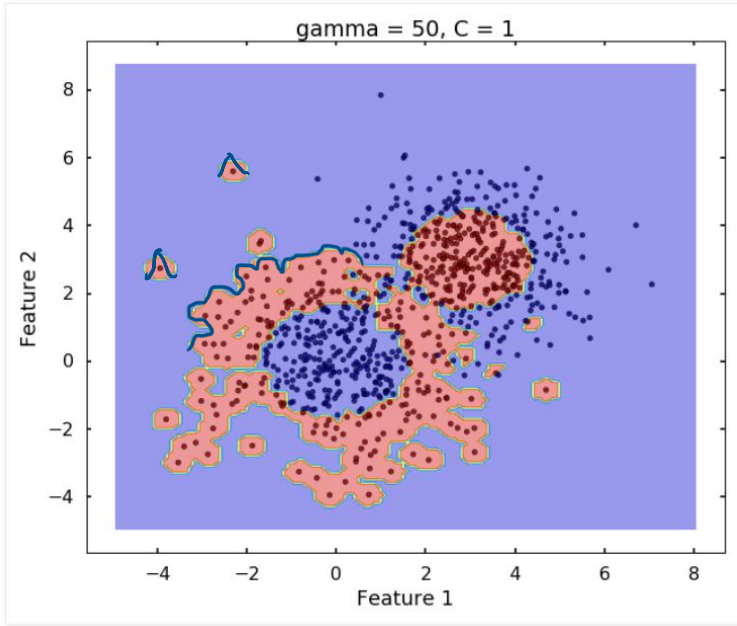


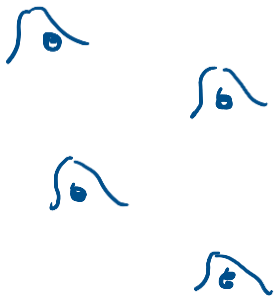


http://qingkaikong.blogspot.com/2016/12/machine-learning-8-support-vector.html

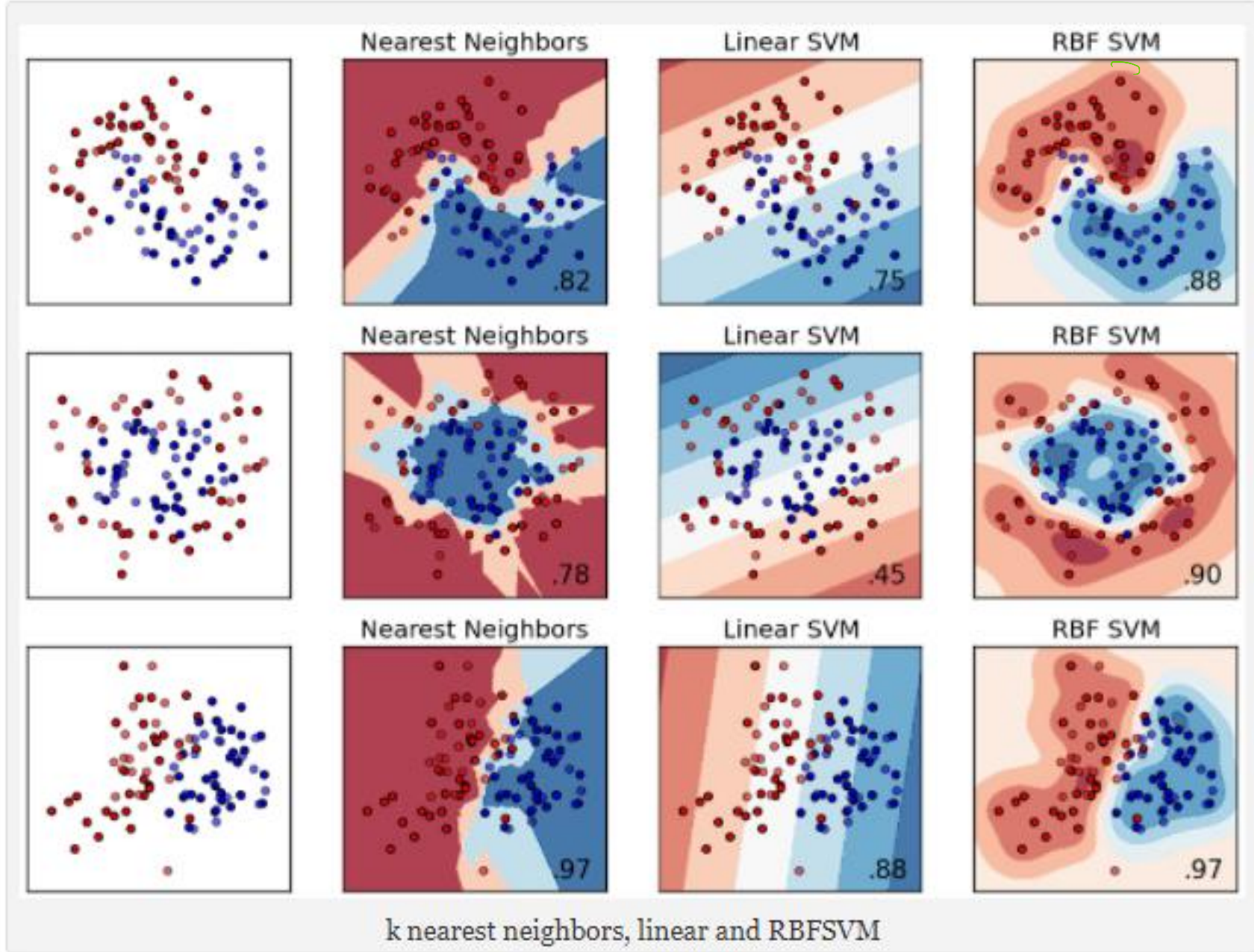






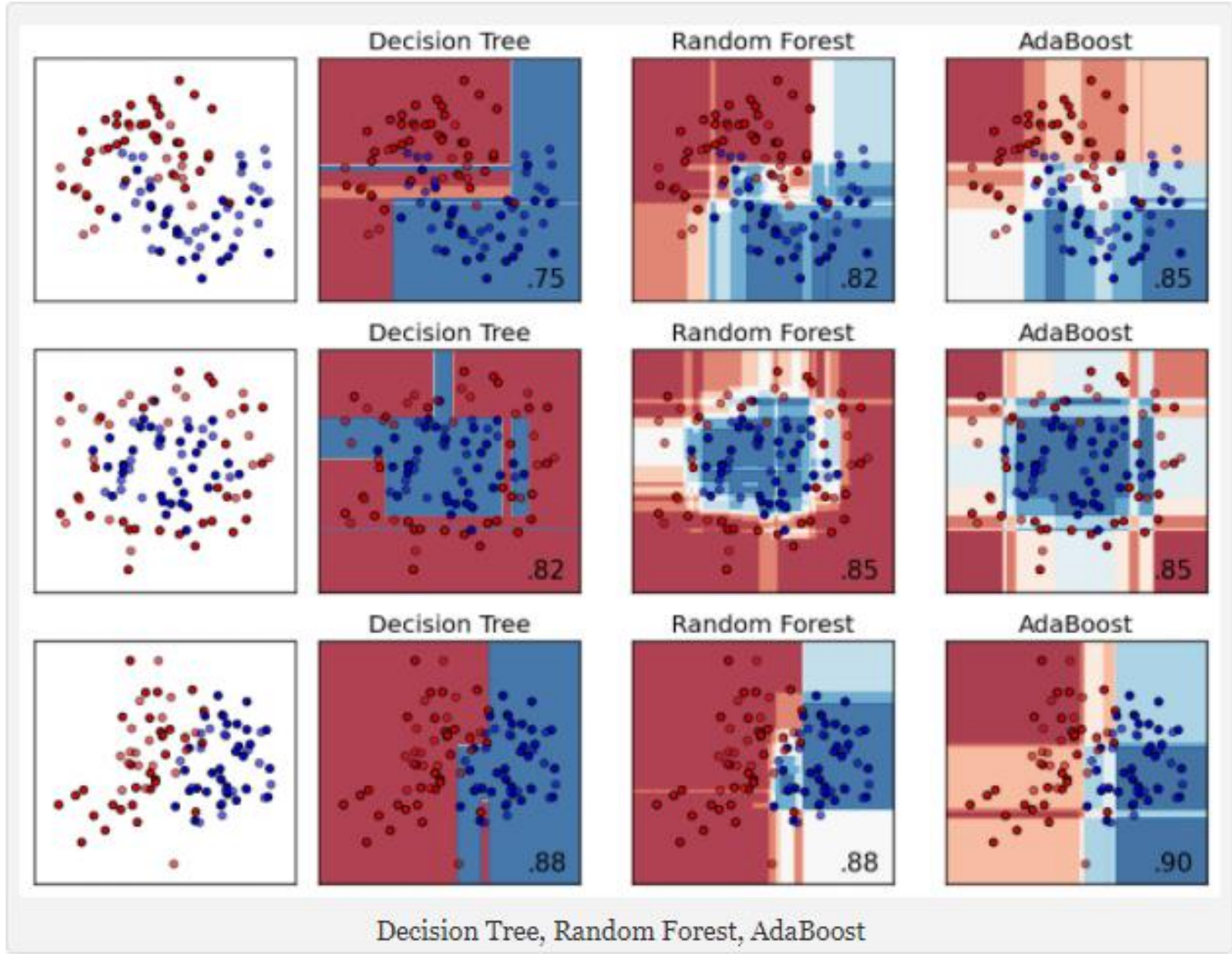


http://qingkaikong.blogspot.com/2016/12/machine-learning-8-support-vector.html



k nearest neighbors, linear and RBFSVM

https://martin-thoma.com/comparing-classifiers/



https://martin-thoma.com/comparing-classifiers/

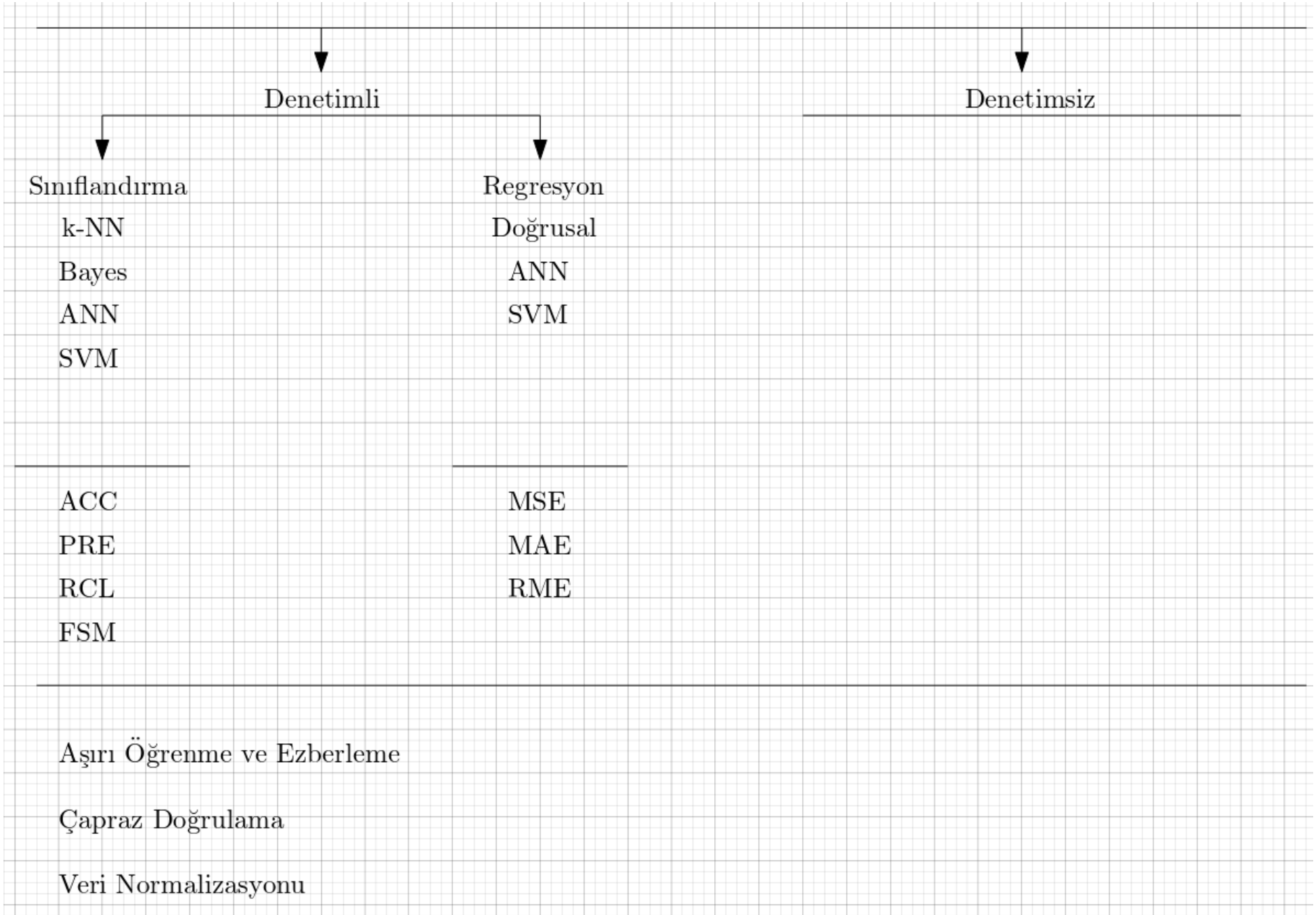
- Büyük veri
- Homojen olmayan veriler
- Gürültülü veri seti
- Boyut Örnek İkilemi
- Hesaplama yükü

- Veri Normalizasyonu

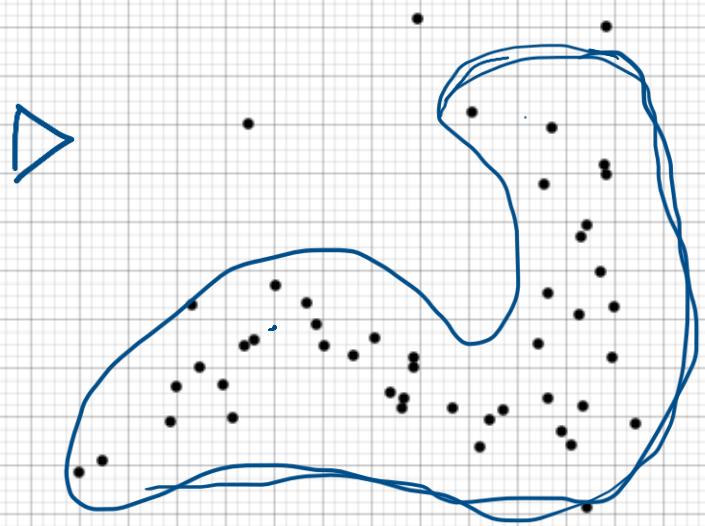
```
X = [-2 , 80 , 0.1
     3 , 120 , 0.5
     6 55 , -0.8
     9 , 74 , -0.3];


mean_vect = mean(X,1);

X_zm = X - mean_vect;

std_vect = std(X_zm , 1);


X_zs = X_zm./std_vect
```

```
%% Min Max Normalization

feat_max = max(X_zm,[],1);
feat_min = min(X_zm,[],1);

X_zm./(feat_max-feat_min)
```

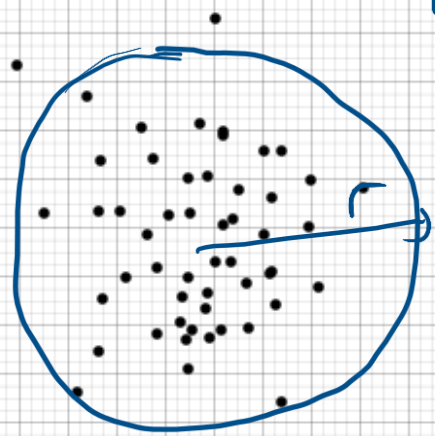
```
%%  Max Normalization

feat_absmax = max( abs(X_zm),[],1);

X_zm./feat_absmax
```


Denetimli
Denetimsiz
Sınıflandırma
k-NN
Bayes
ANN
SVM
Regresyon
Doğrusal
ANN
SVM
ACC
PRE
RCL
FSM
MSE
MAE
RME
Aşırı Öğrenme ve Ezberleme
Çapraz Doğrulama
Veri Normalizasyonu

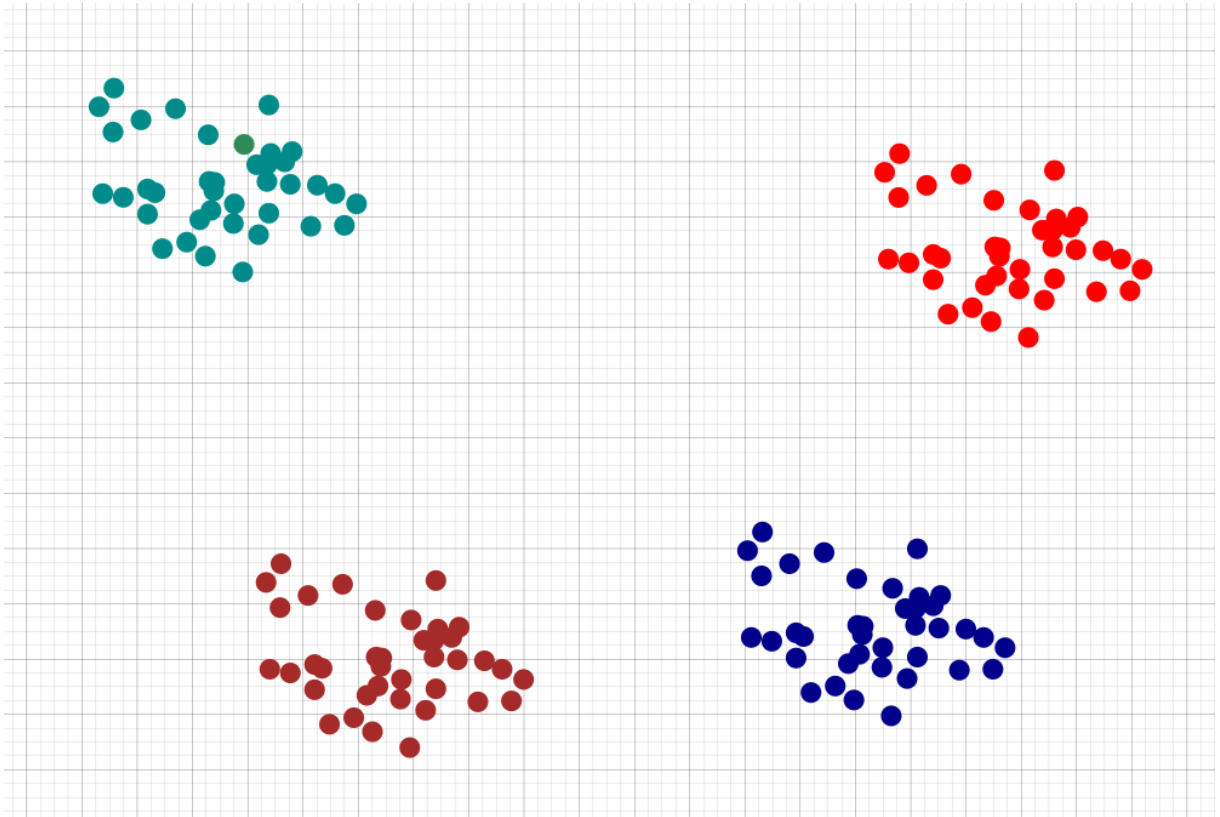
Verinin kendi uzayındaki dağılımı

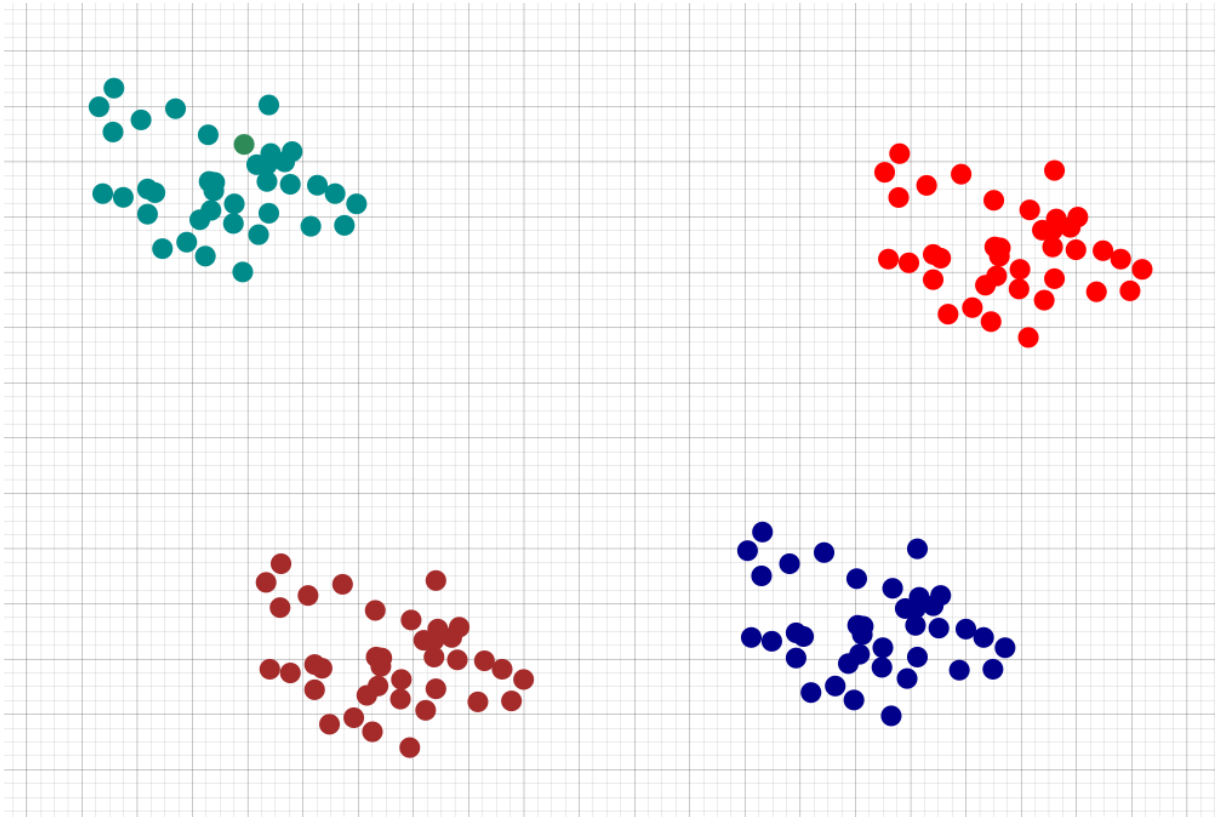


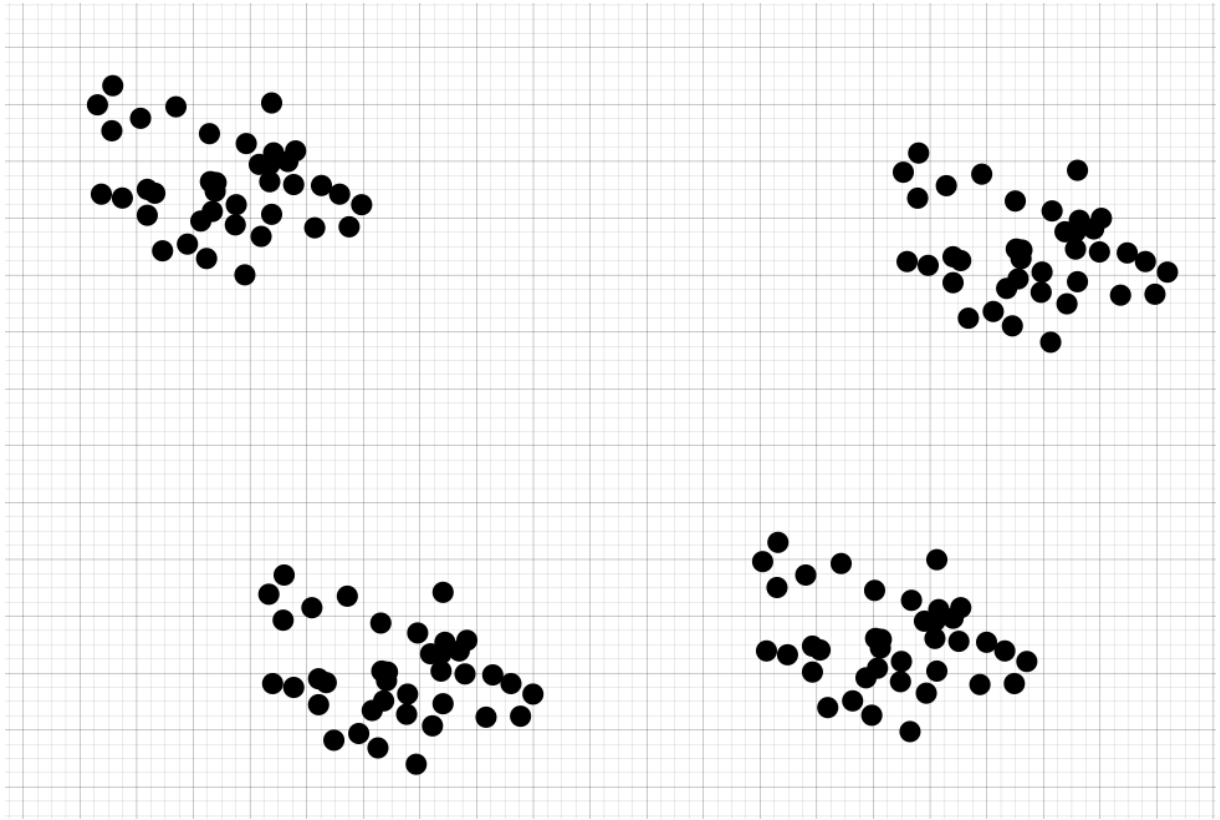
Kernel uzayı



Denetimsiz Öğrenme

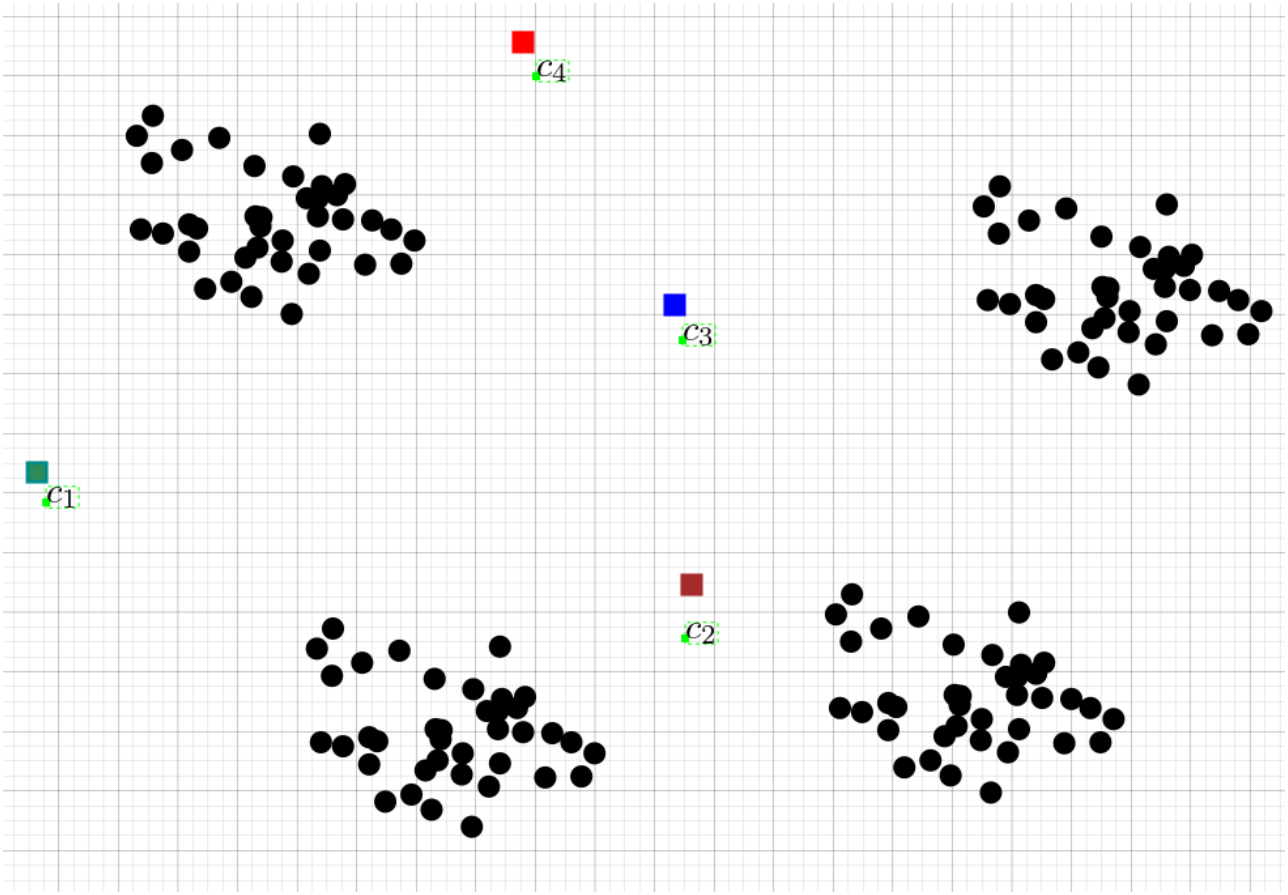






Ön ***Bilgi***:

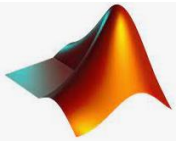
$$\boldsymbol{S} = \{S_1, S_2, S_3, S_4\}$$

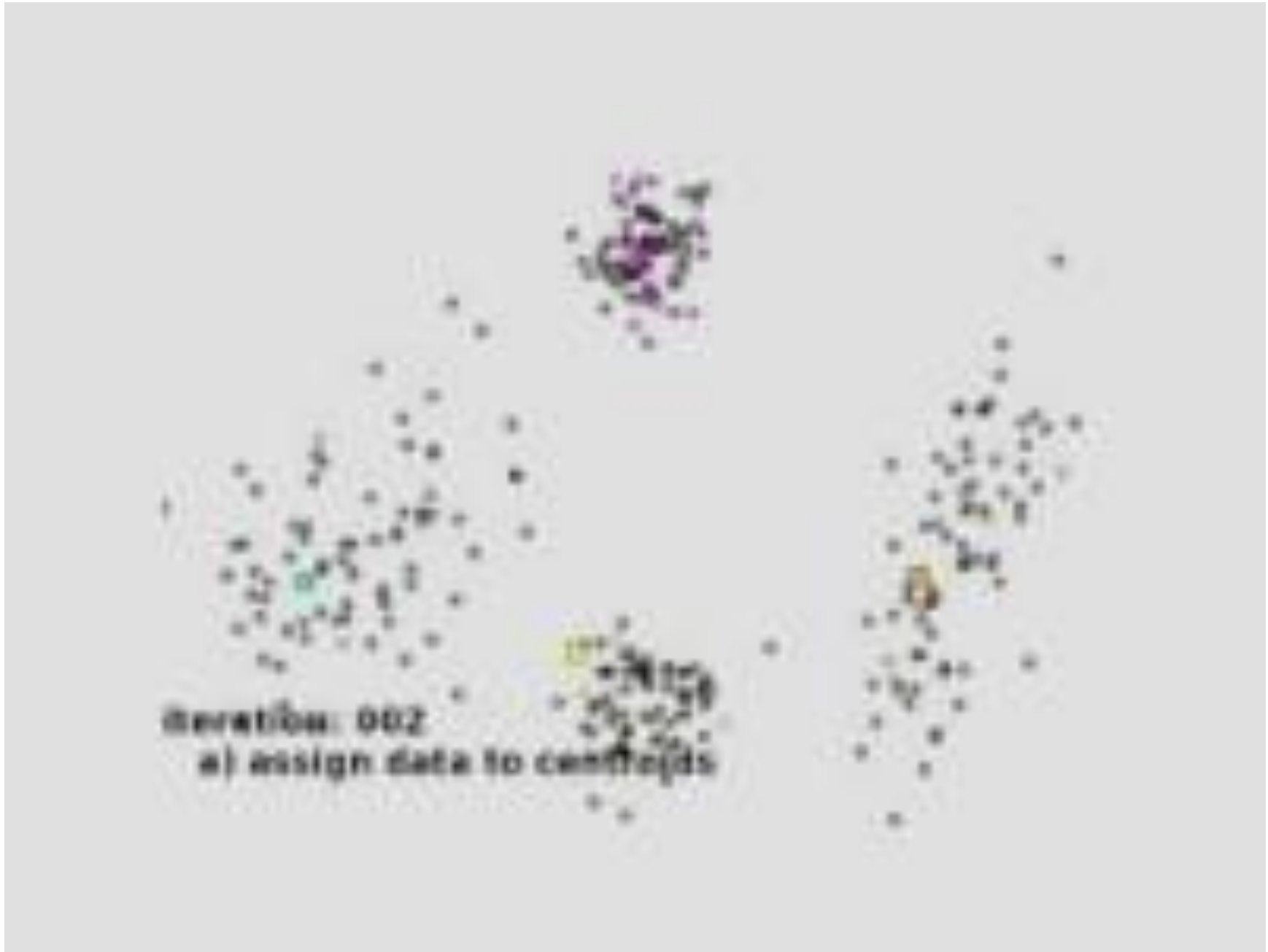


Ön $\boldsymbol{Bilgi}$: $\boldsymbol{S} = \{S_1, S_2, S_3, S_4\}$

- **Atama Adımı** Veri kümesindeki her nokta **c** merkezleri ile ilişkilendirilir.

- **Güncelleme Adımı** Atanmış veriler üzerinden yeni sınıf merkezleri üretilir.

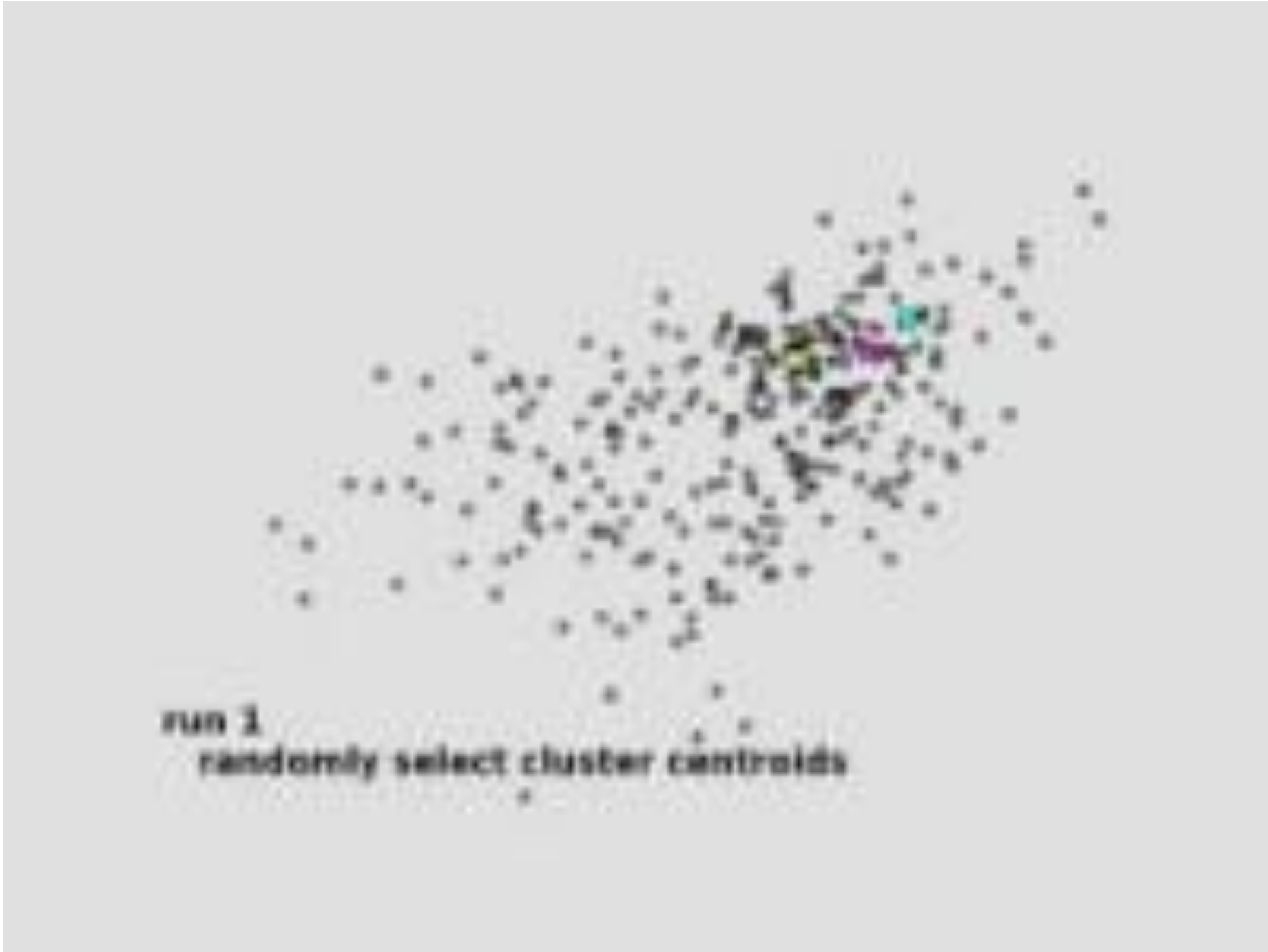


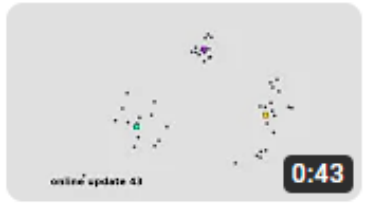

Iteration: 002
a) assign data to centroids

x1	x2
-2	1
8	6
-3	-2
10	5
3	4
-3	-4
12	3

-15 , -15	

15 , 15	

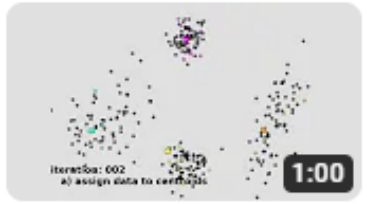

run 1
randomly select cluster centroids



online k-means clustering

bitLectures

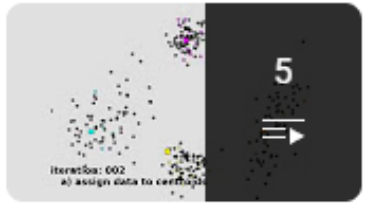
4,2 B görüntüleme • 7 yıl önce



k-means clustering

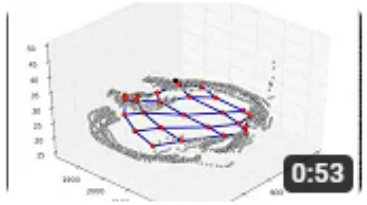
bitLectures

40 B görüntüleme • 8 yıl önce



k-means clustering

bitLectures



self organizing map (grid topology)

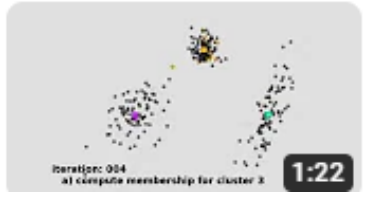
bitLectures

3,5 B görüntüleme • 8 yıl önce

Hartigan's algorithm for k-means clustering

bitLectures

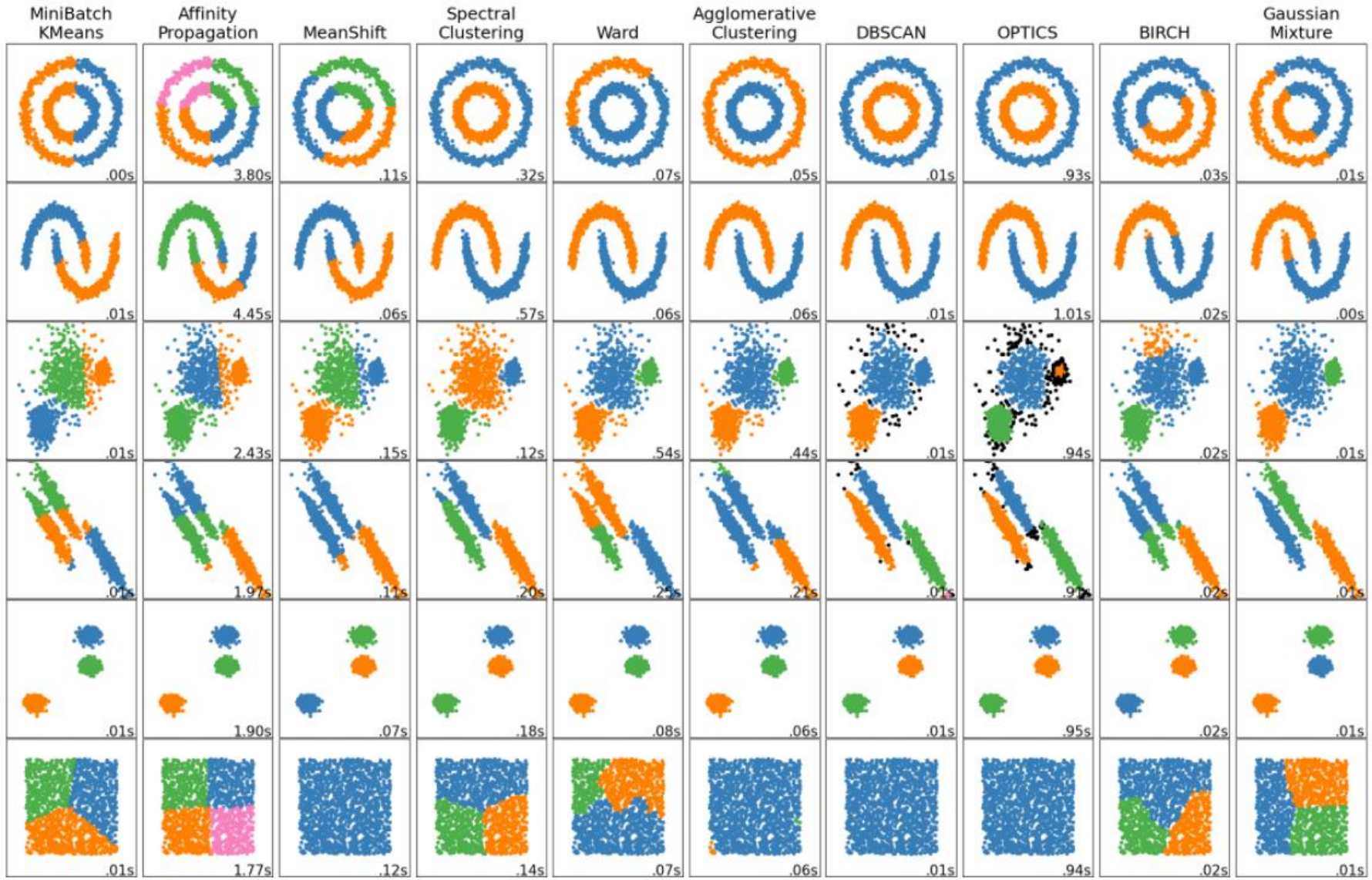
2,4 B görüntüleme • 6 yıl önce



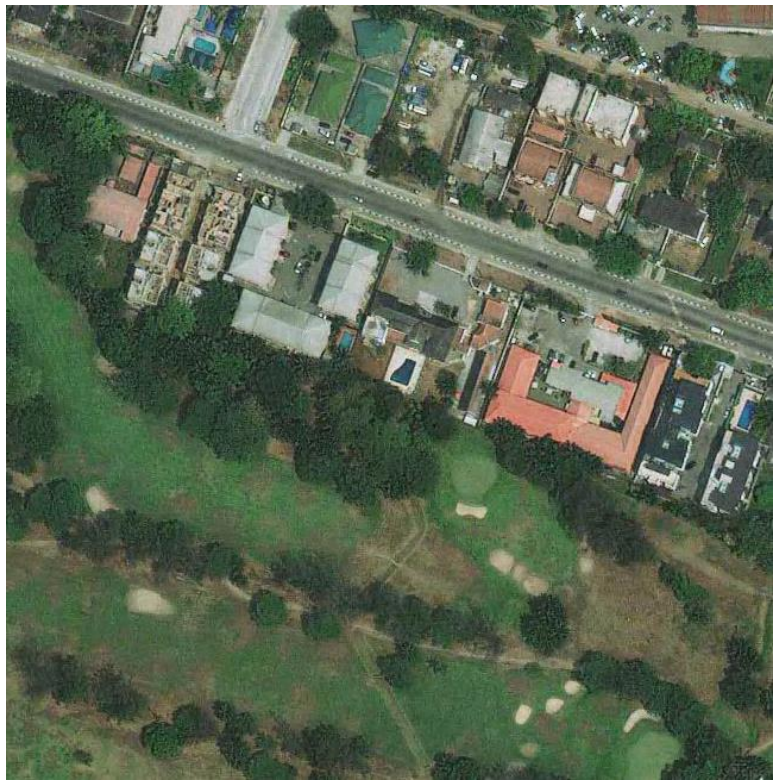
soft k-means clustering

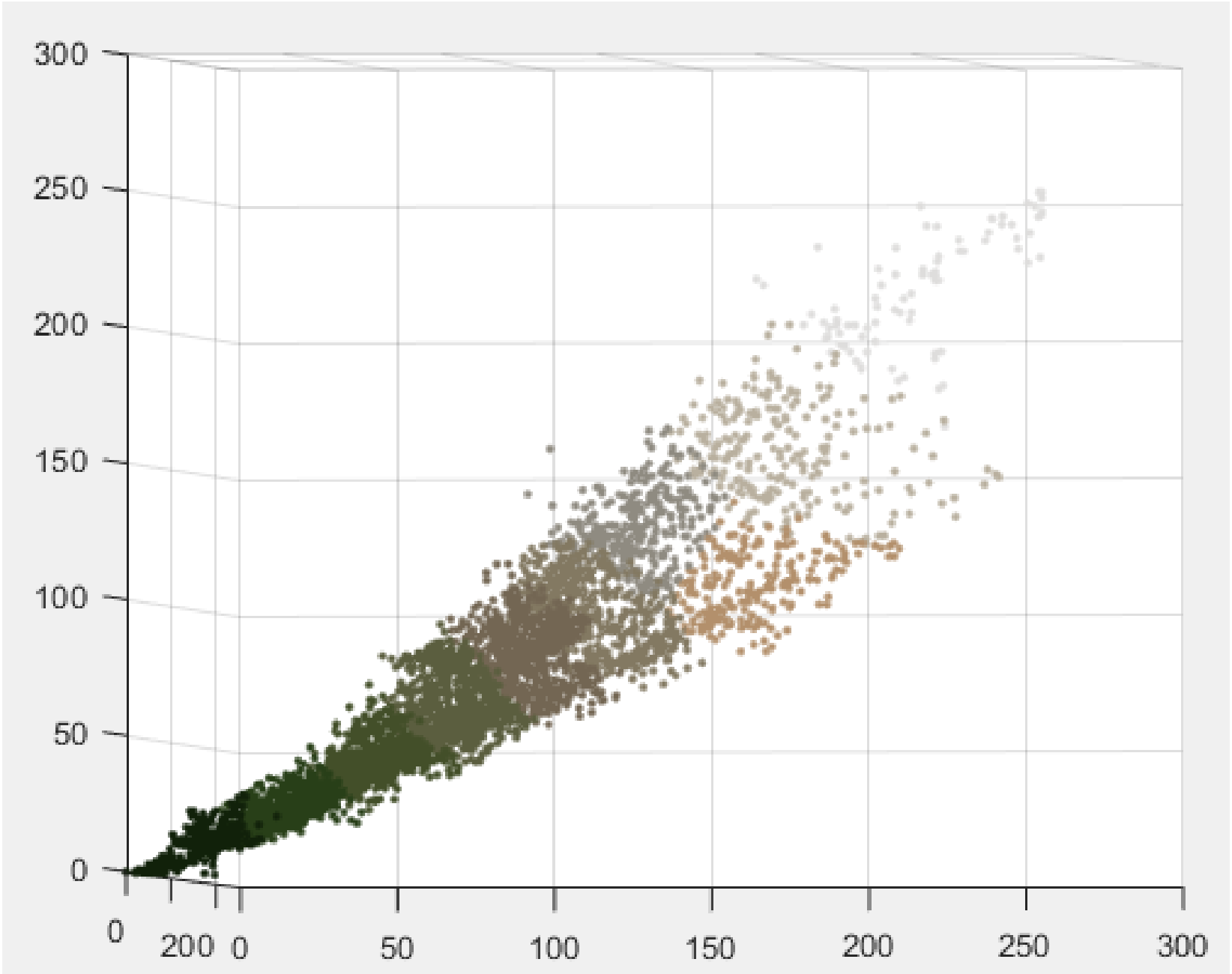
bitLectures

3,2 B görüntüleme • 7 yıl önce



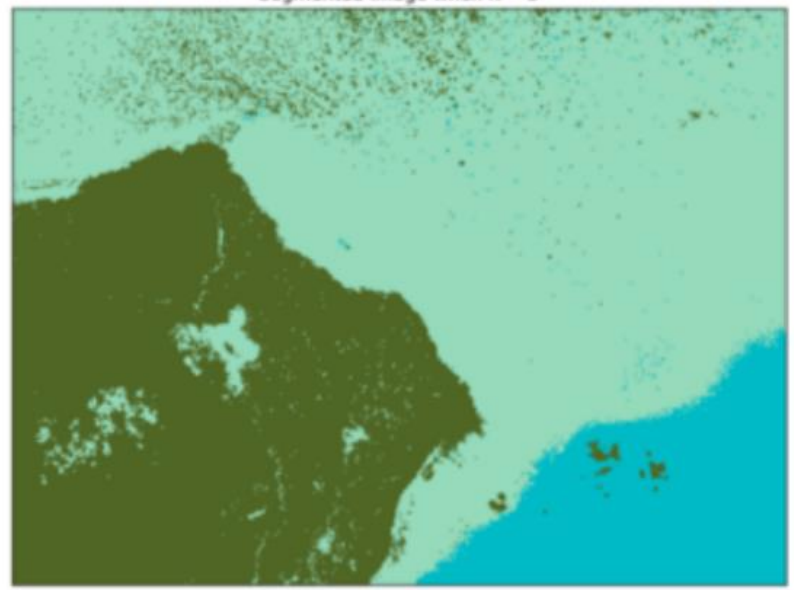
A comparison of the clustering algorithms in scikit-learn




300
250
200
150
100
50
0
0
200
0
50
100
150
200
250
300

Original Image

Segmented Image when K = 3



Original Image

Segmented Image when K = 6

Original Image

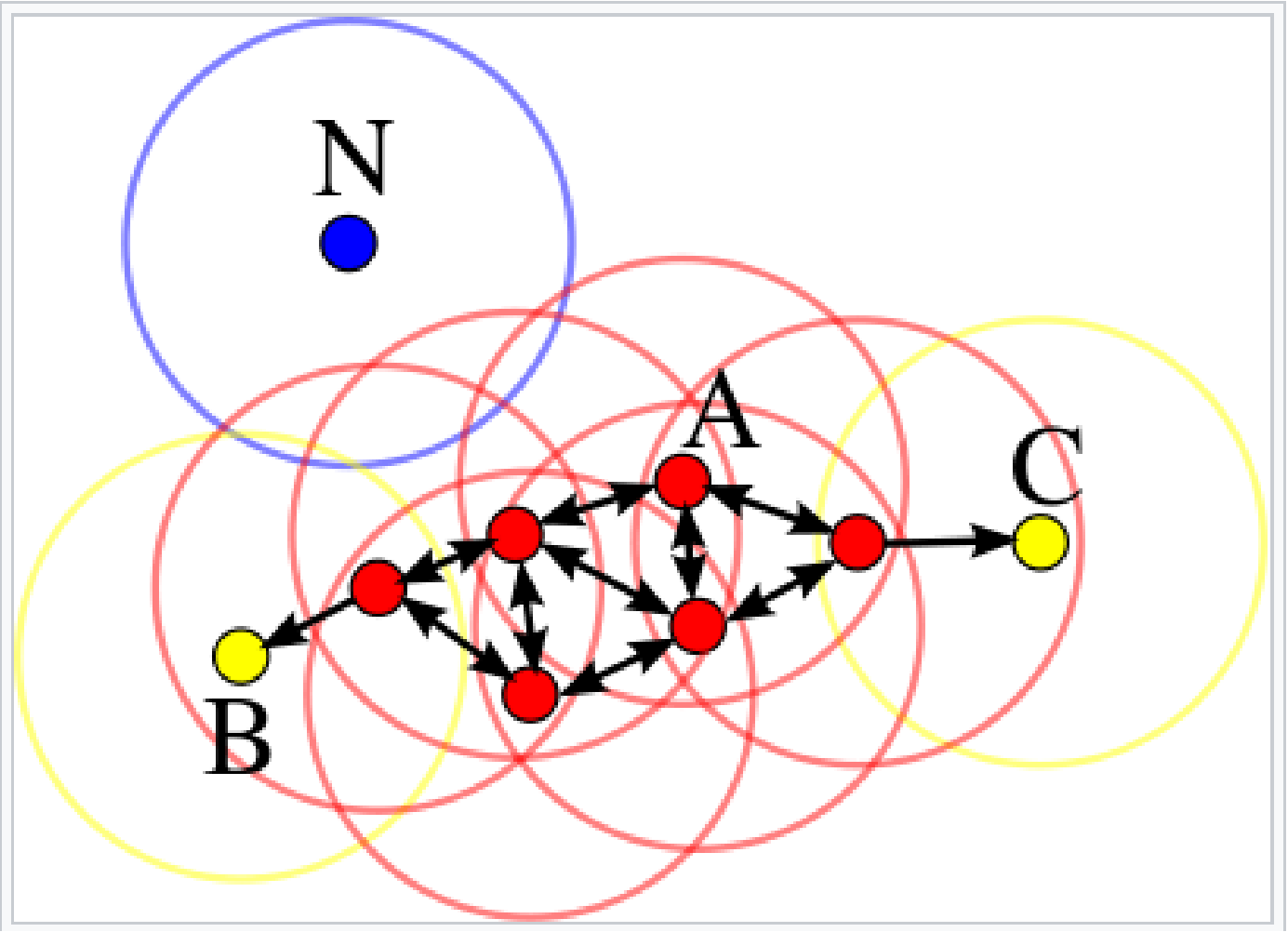
Compressed Image with 30 colors

https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a

K Sayısı?

Density−**B**ased **S**patial **C**lustering of **A**pplications with **N**oise $\boldsymbol{DBSCAN}$

Density−**B**ased **S**patial **C**lustering of **A**pplications with **N**oise $\boldsymbol{DBSCAN}$

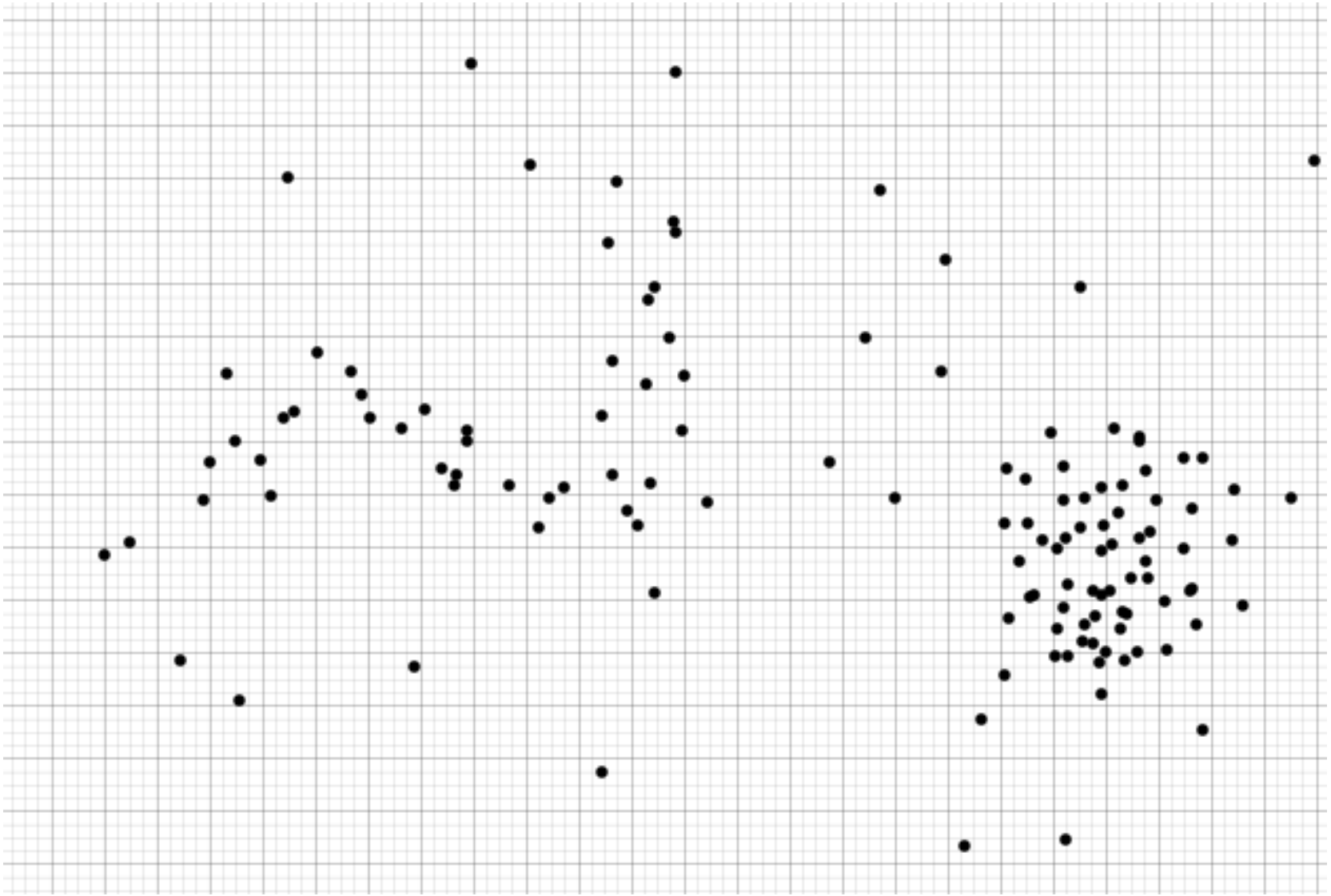


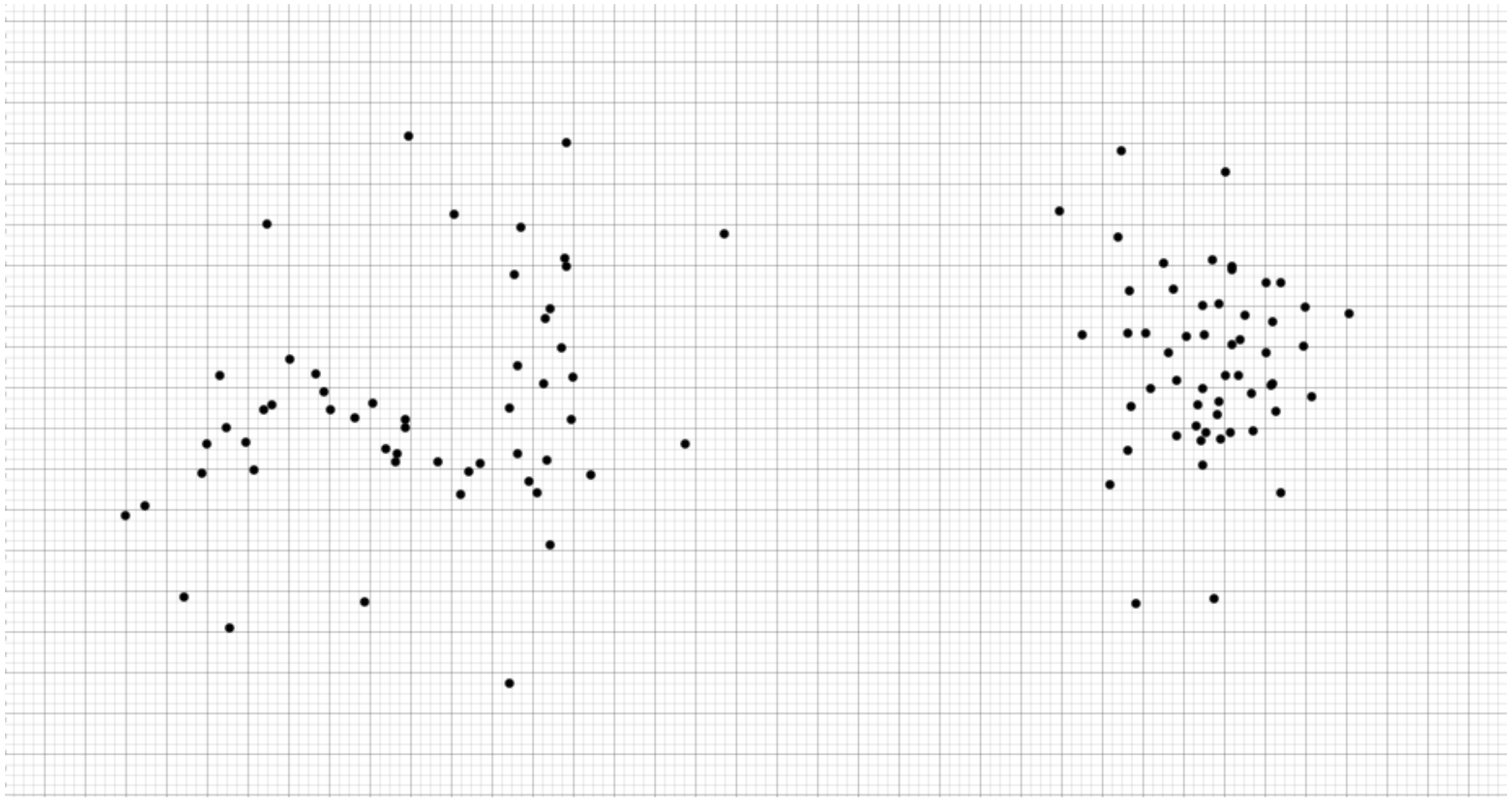
$\boldsymbol{Amaç}$:

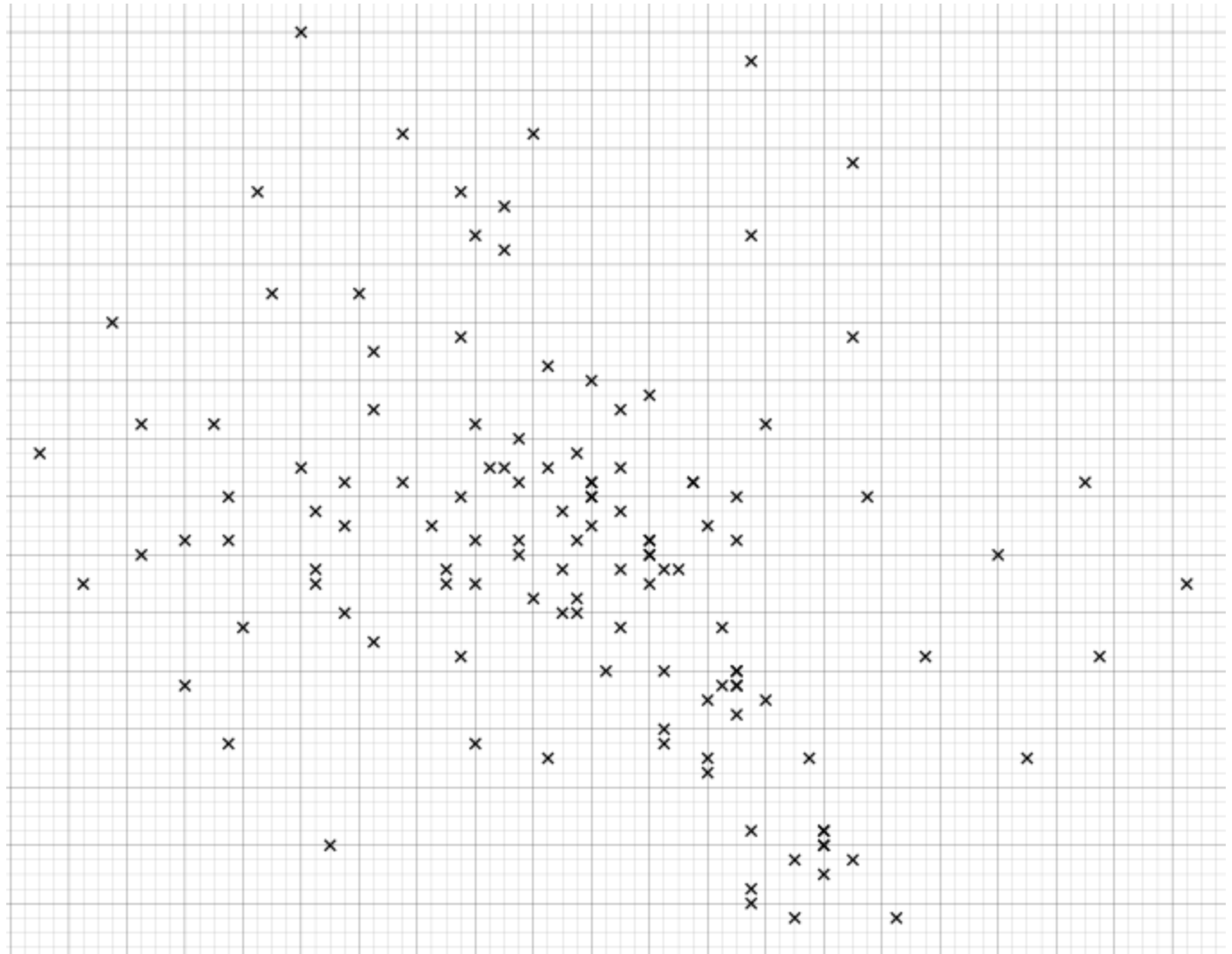
Ön $\boldsymbol{Bilgi}$:

https://en.wikipedia.org/wiki/DBSCAN

Density−**B**ased **S**patial **C**lustering of **A**pplications with **N**oise $\boldsymbol{DBSCAN}$







Density−**B**ased **S**patial **C**lustering of **A**pplications with **N**oise $\boldsymbol{DBSCAN}$