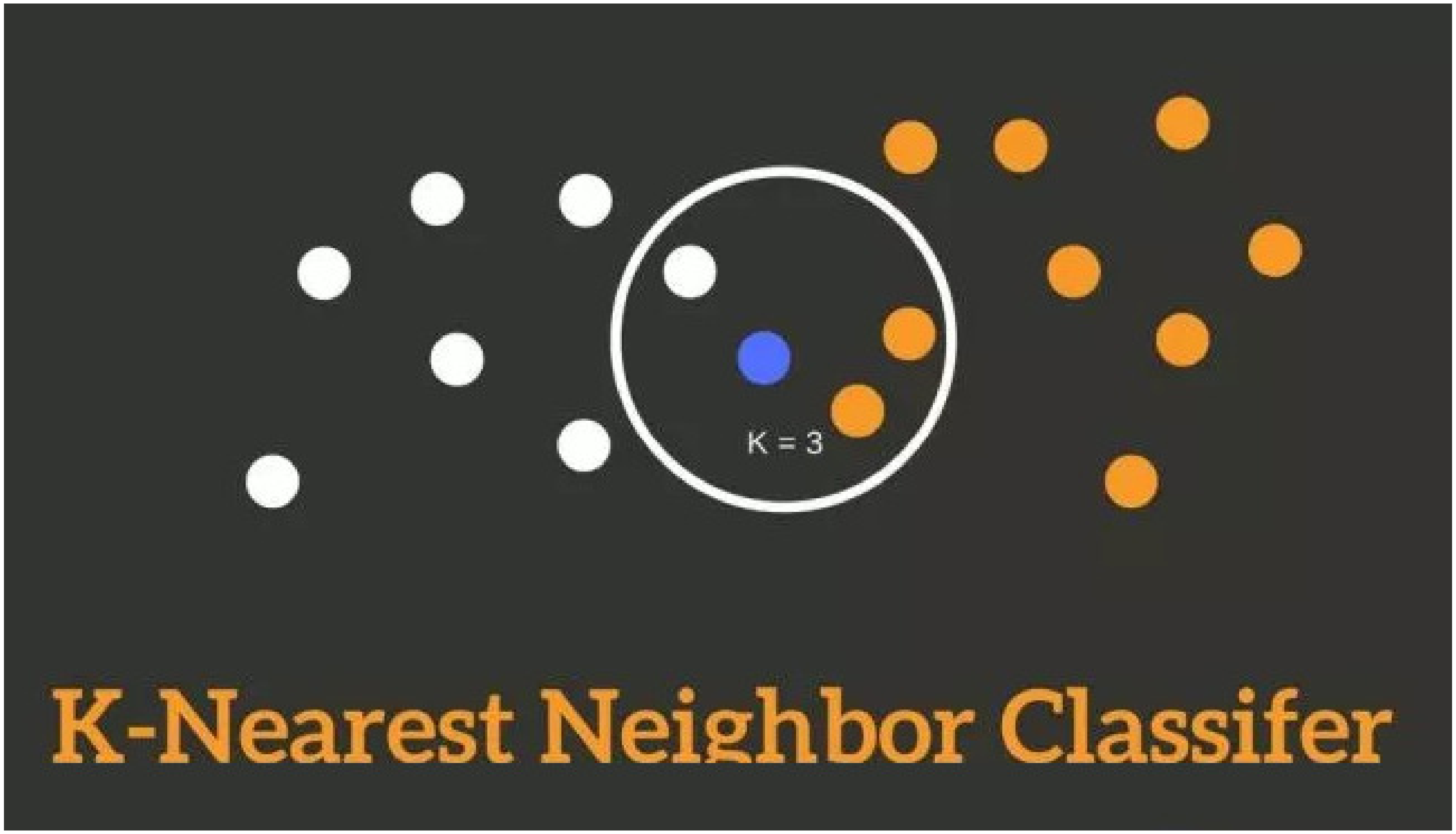
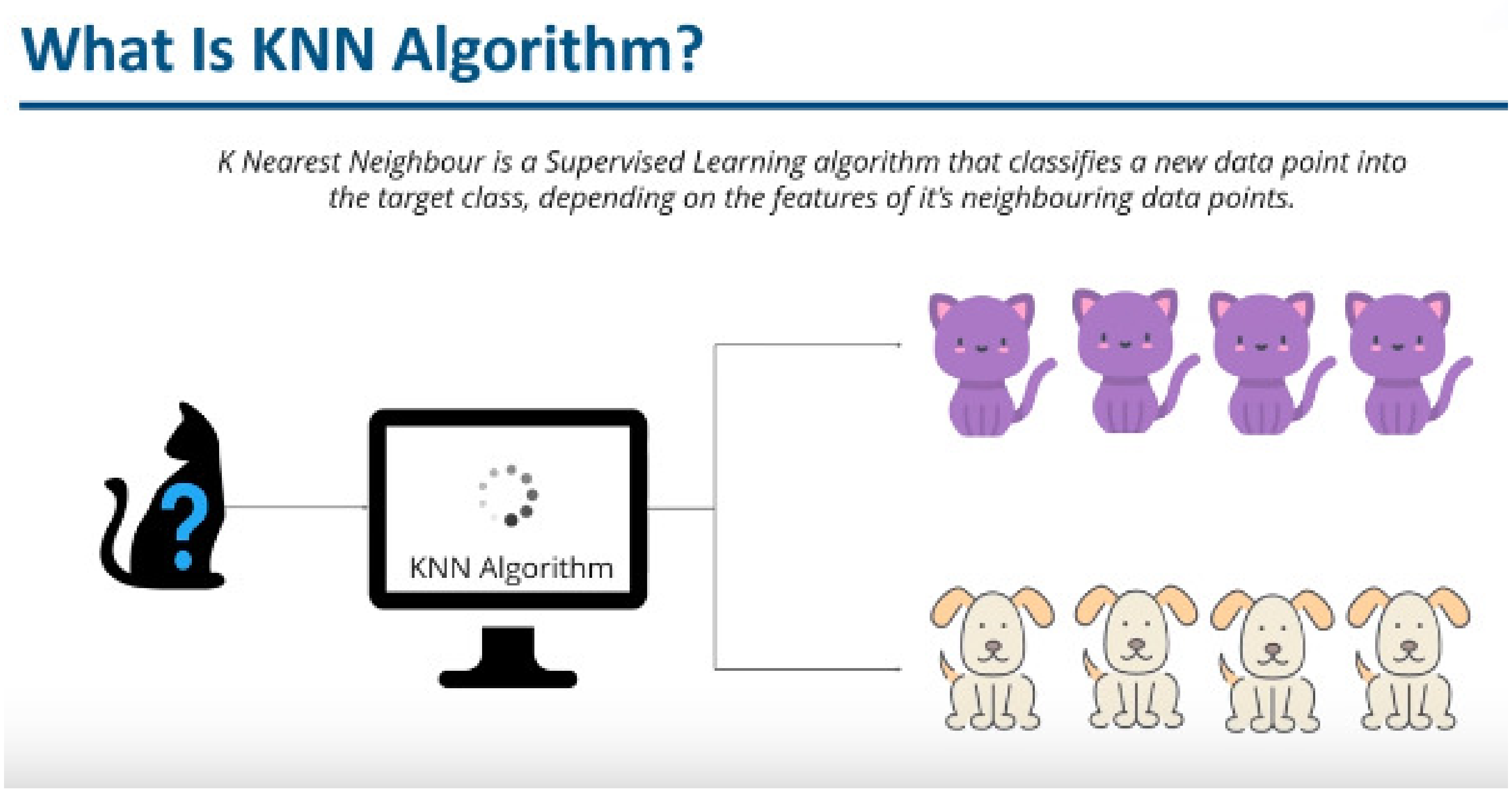

K = 3
K-Nearest Neighbor Classifer



What Is KNN Algorithm?

K Nearest Neighbour is a Supervised Learning algorithm that classifies a new data point into the target class, depending on the features of it's neighbouring data points.

Image from Edureka

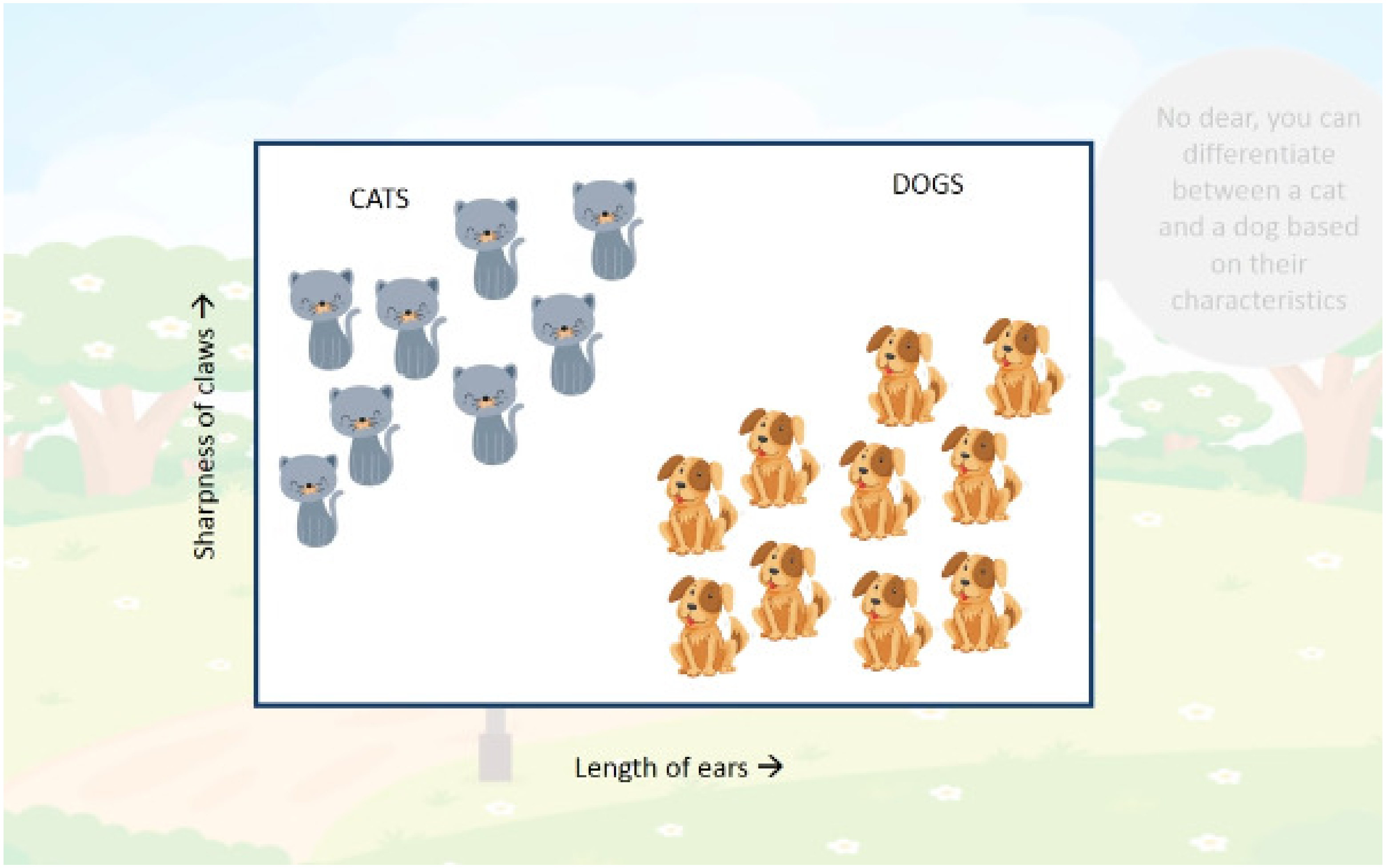
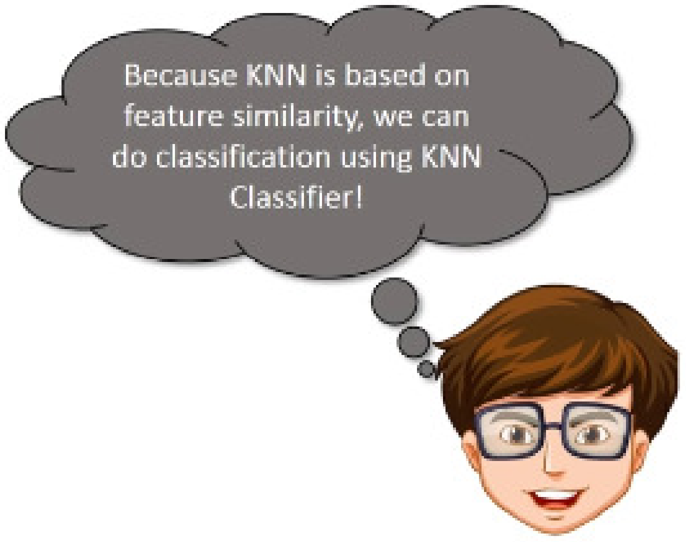

CATS
DOGS
Sharpness of claws →
Length of ears →
No dear, you can differentiate between a cat and a dog based on their characteristics

Image from Simpllea

Why KNN?


Because KNN is based on feature similarity, we can do classification using KNN Classifier!

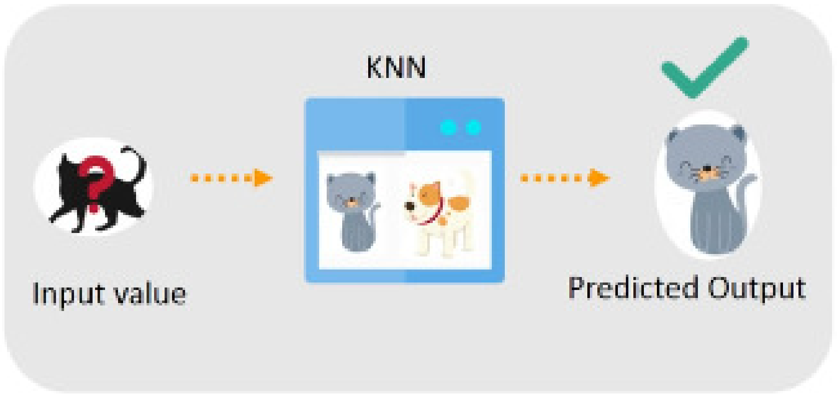
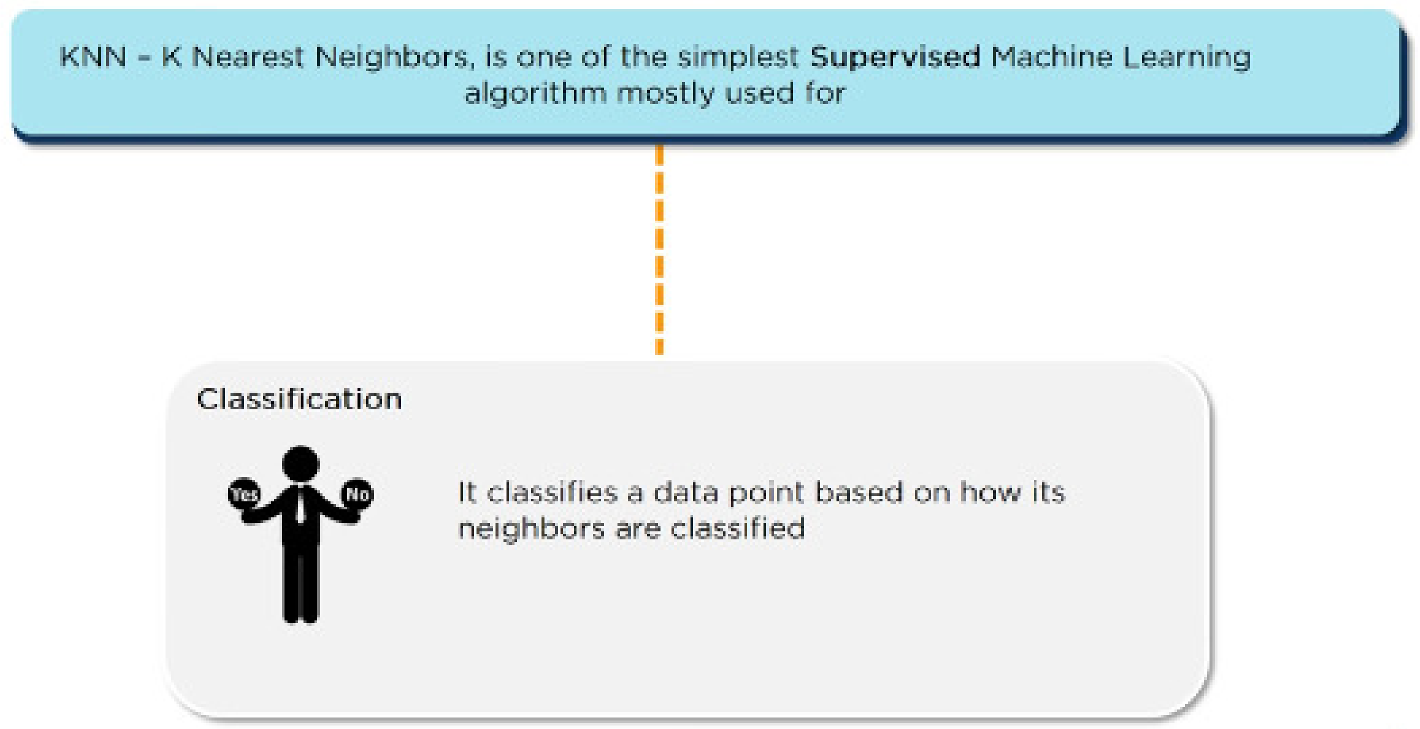


Image from Simpllea

What is KNN Algorithm?



KNN - K Nearest Neighbors, is one of the simplest **Supervised** Machine Learning algorithm mostly used for

Classification

It classifies a data point based on how its neighbors are classified

Image from Simpllea

KNN stores all available cases and classifies new cases based on a similarity measure

k in **KNN** is a parameter that refers to the number of nearest neighbors to include in the majority voting process

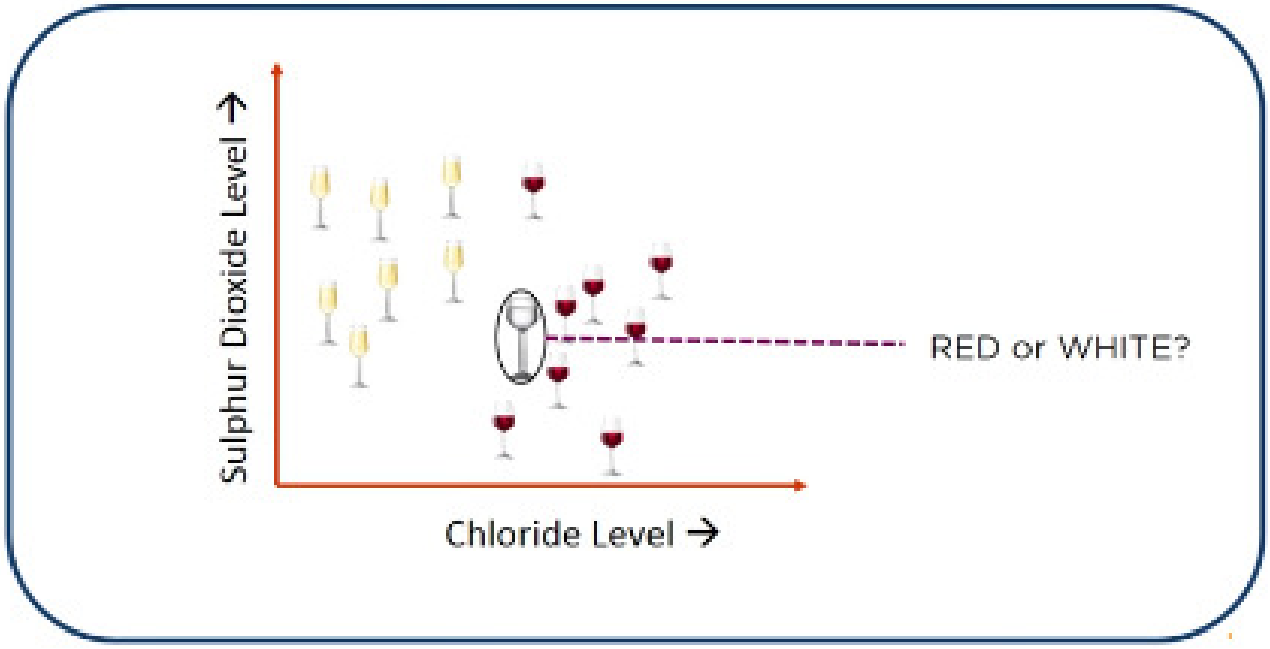
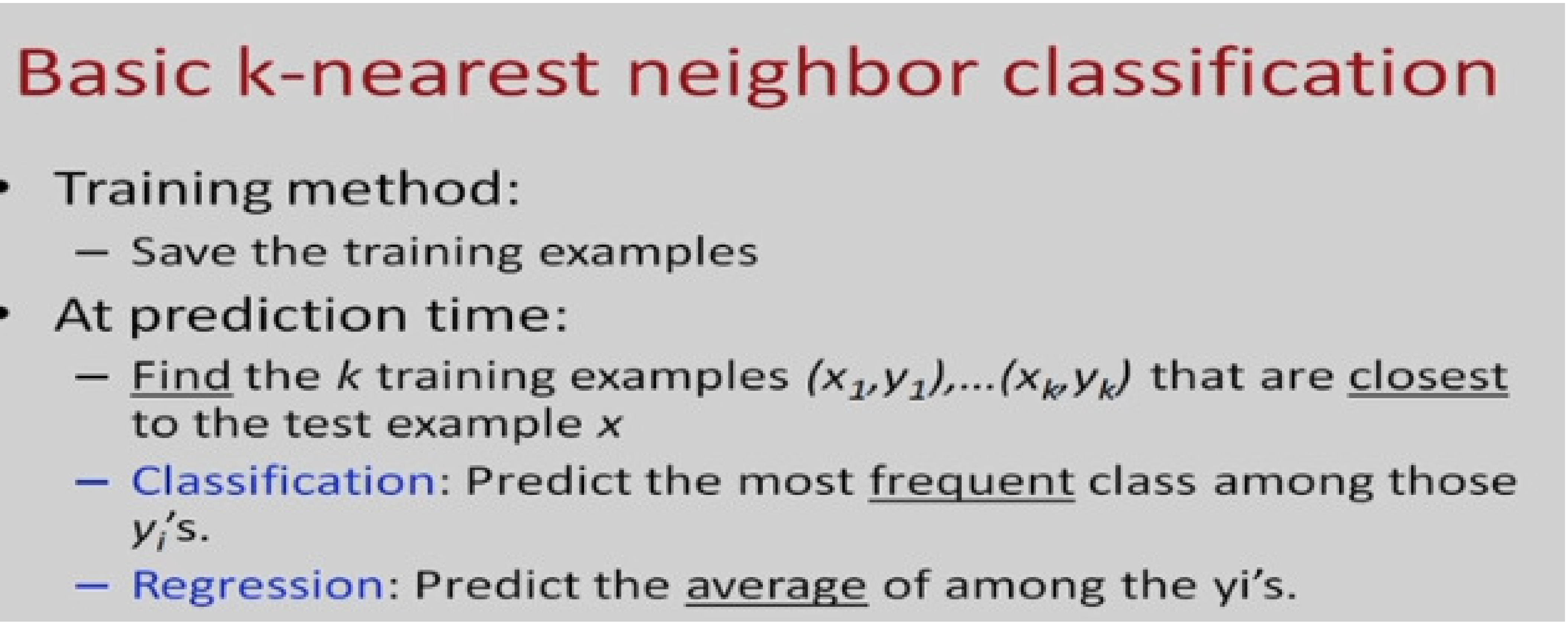


Image from Simpllea



Basic k-nearest neighbor classification

- Training method:
  - Save the training examples
- At prediction time:
  - Find the k training examples $(x_1,y_1),\ldots(x_k,y_k)$ that are closest to the test example x
  - Classification: Predict the most frequent class among those y_i's.
  - Regression: Predict the average of among the yi's.

Image from Simpllea

Features Of KNN

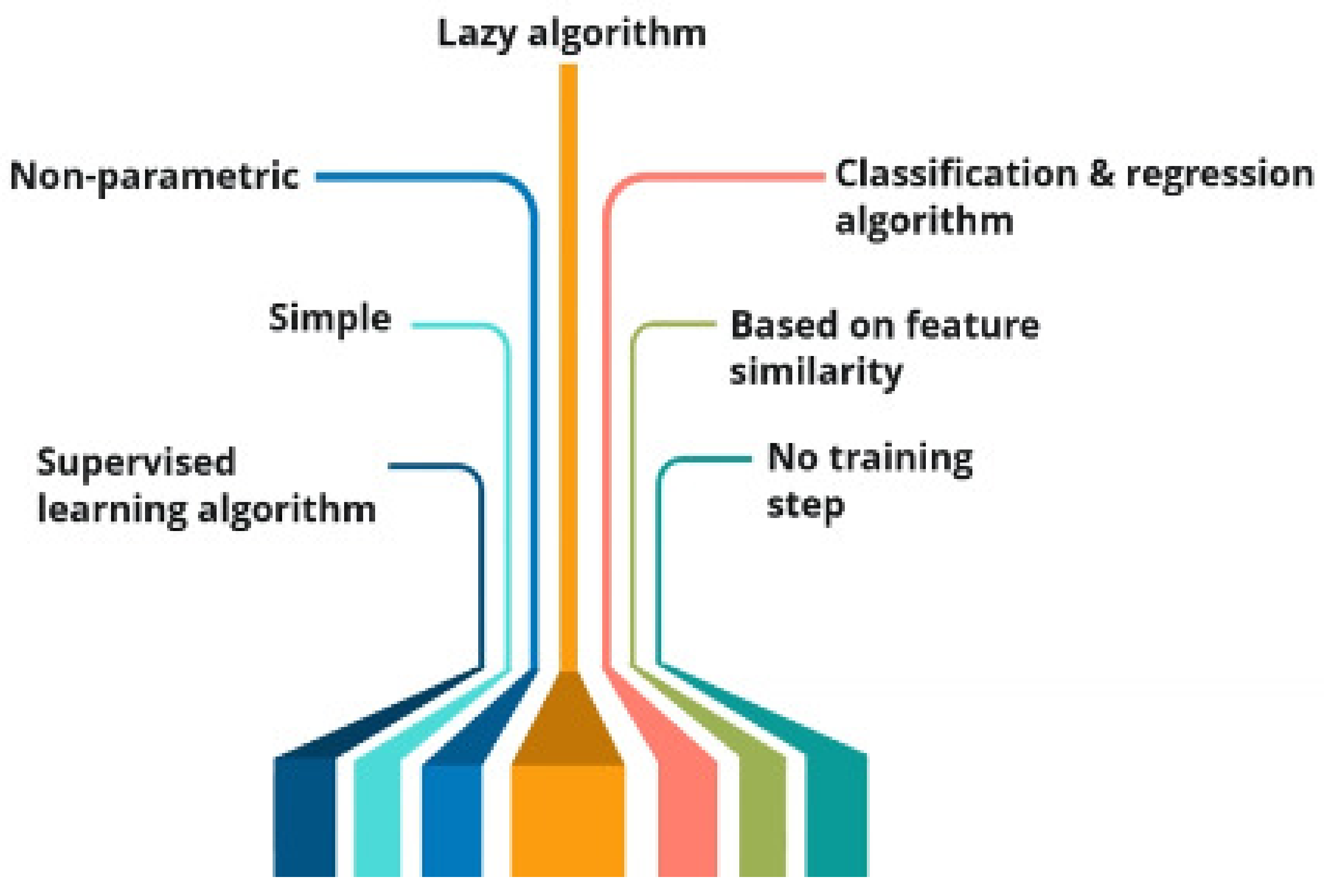
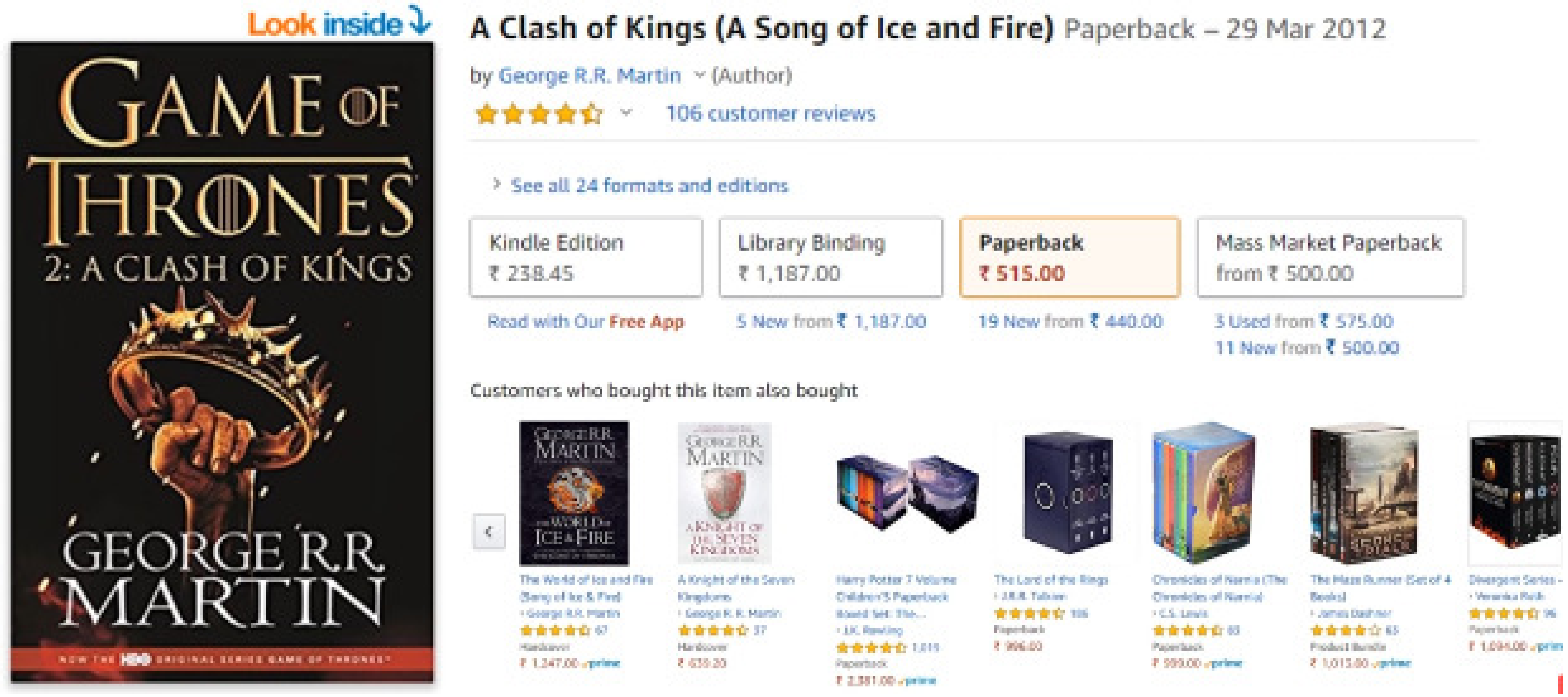

Lazy algorithm
Non-parametric
Classification & regression algorithm
Simple
Based on feature similarity
Supervised learning algorithm
No training step

Image from Edu

Book Recommendation Using KNN



Look inside

A Clash of Kings (A Song of Ice and Fire) Paperback – 29 Mar 2012

by George R.R. Martin (Author)

106 customer reviews

See all 24 formats and editions

Kindle Edition	Library Binding	Paperback	Mass Market Paperback
₹ 238.45	₹ 1,187.00	₹ 515.00	from ₹ 500.00
Read with Our Free App	5 New from ₹ 1,187.00	19 New from ₹ 440.00	3 Used from ₹ 575.00 11 New from ₹ 500.00

Customers who bought this item also bought

The World of Ice and Fire (Song of Ice & Fire) › George R.R. Martin 67 Hardcover ₹ 1,247.00 prime

A Knight of the Seven Kingdoms › George R. R. Martin 37 Hardcover ₹ 639.20

Harry Potter 7 Volume Children's Paperback Boxed Set: The... › J.K. Rowling 1,019 Paperback ₹ 2,381.00 prime

The Lord of the Rings › J.R.R. Tolkien 106 Paperback ₹ 996.00

Chronicles of Narnia (The Chronicles of Narnia) › C.S. Lewis 83 Paperback ₹ 599.00 prime

The Maze Runner (Set of 4 Books) › James Dashner 63 Product Bundle ₹ 1,013.00 prime

Divergent Series - › Veronica Roth 96 Paperback ₹ 1,094.00 prime

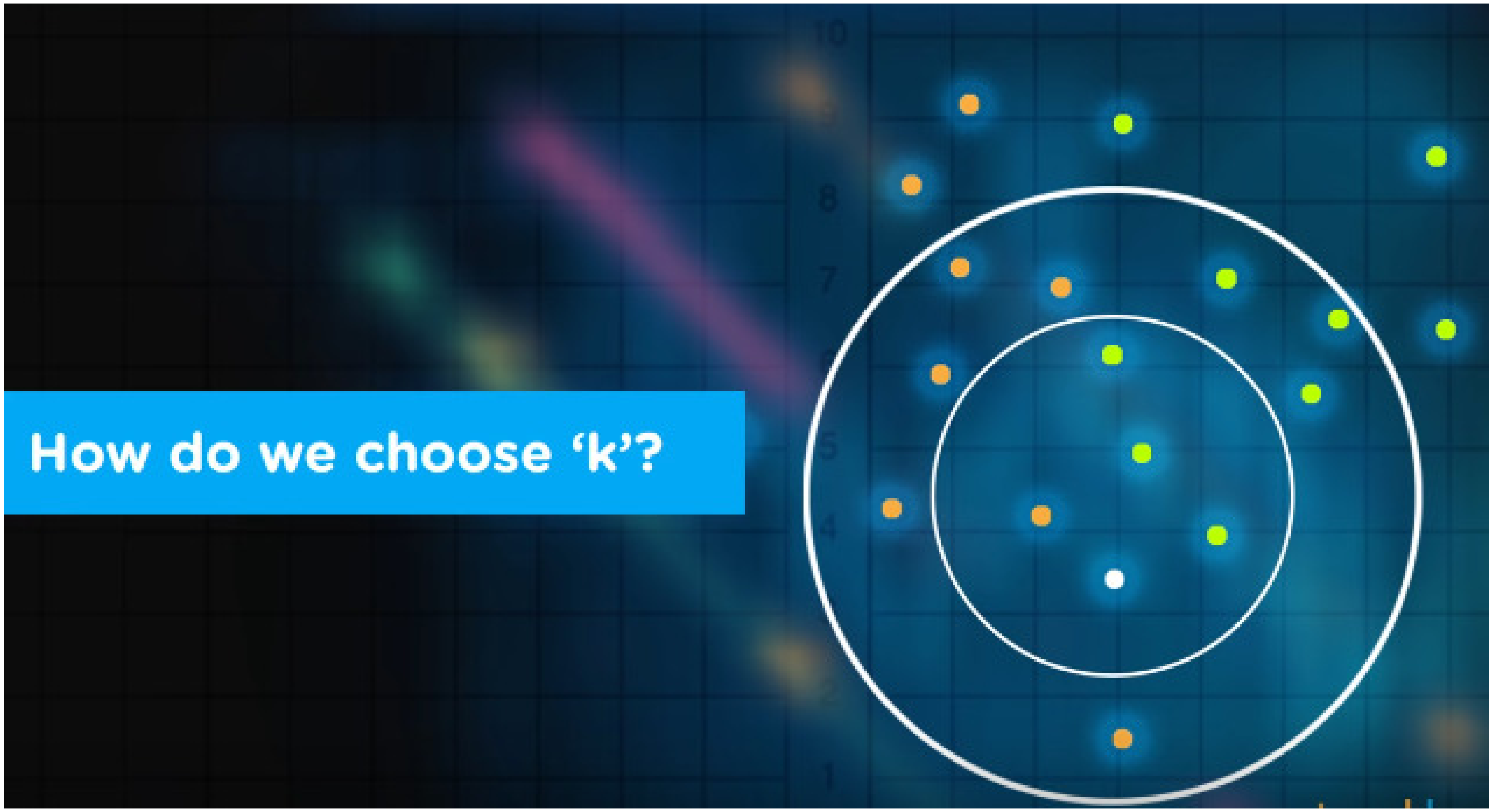
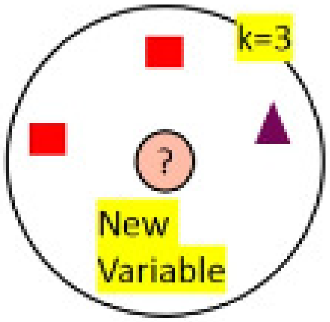


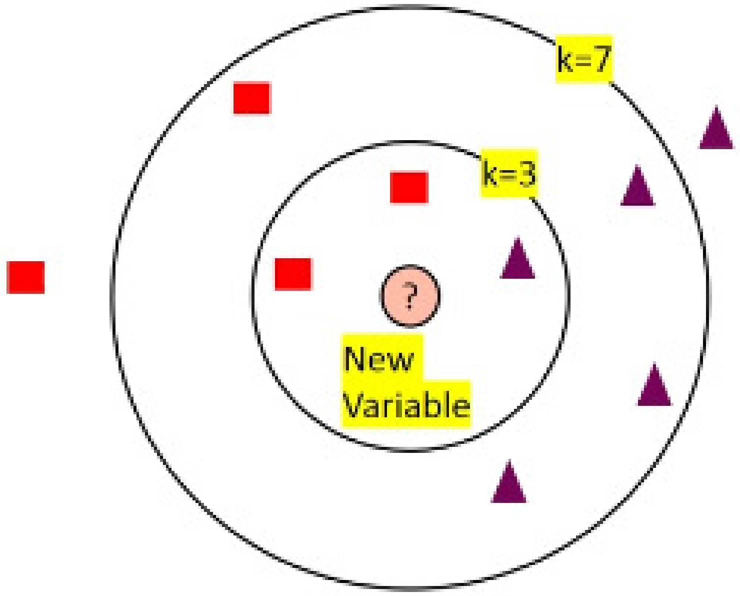
Image from Simpllea

How do we choose the factor ‘k’?

KNN Algorithm is based on feature similarity: Choosing the right value of *k* is a process called parameter tuning, and is important for better accuracy



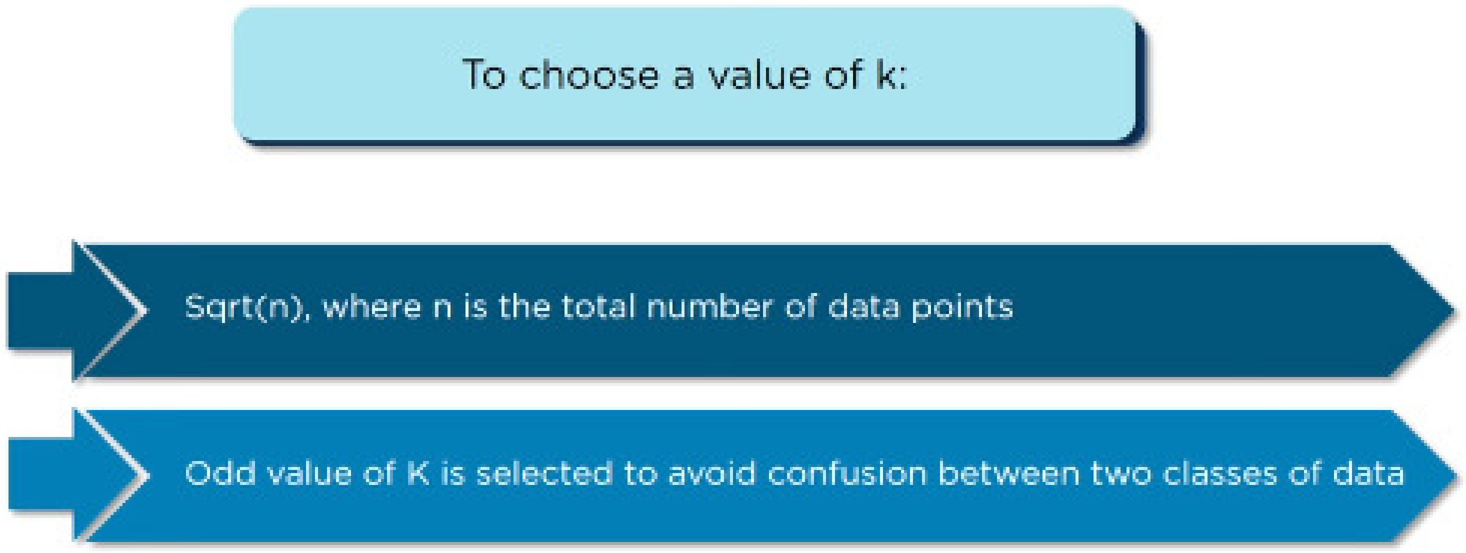
So at k=3, we can classify ‘?’ as 🟥



But at k=7, we classify ‘?’ as ▲

Image from Simplea

How do we choose the factor ‘k’?



To choose a value of k:

- Sqrt(n), where n is the total number of data points
- Odd value of K is selected to avoid confusion between two classes of data

Image from Simpllea

From Hastie, Tibshirani, Friedman 2001

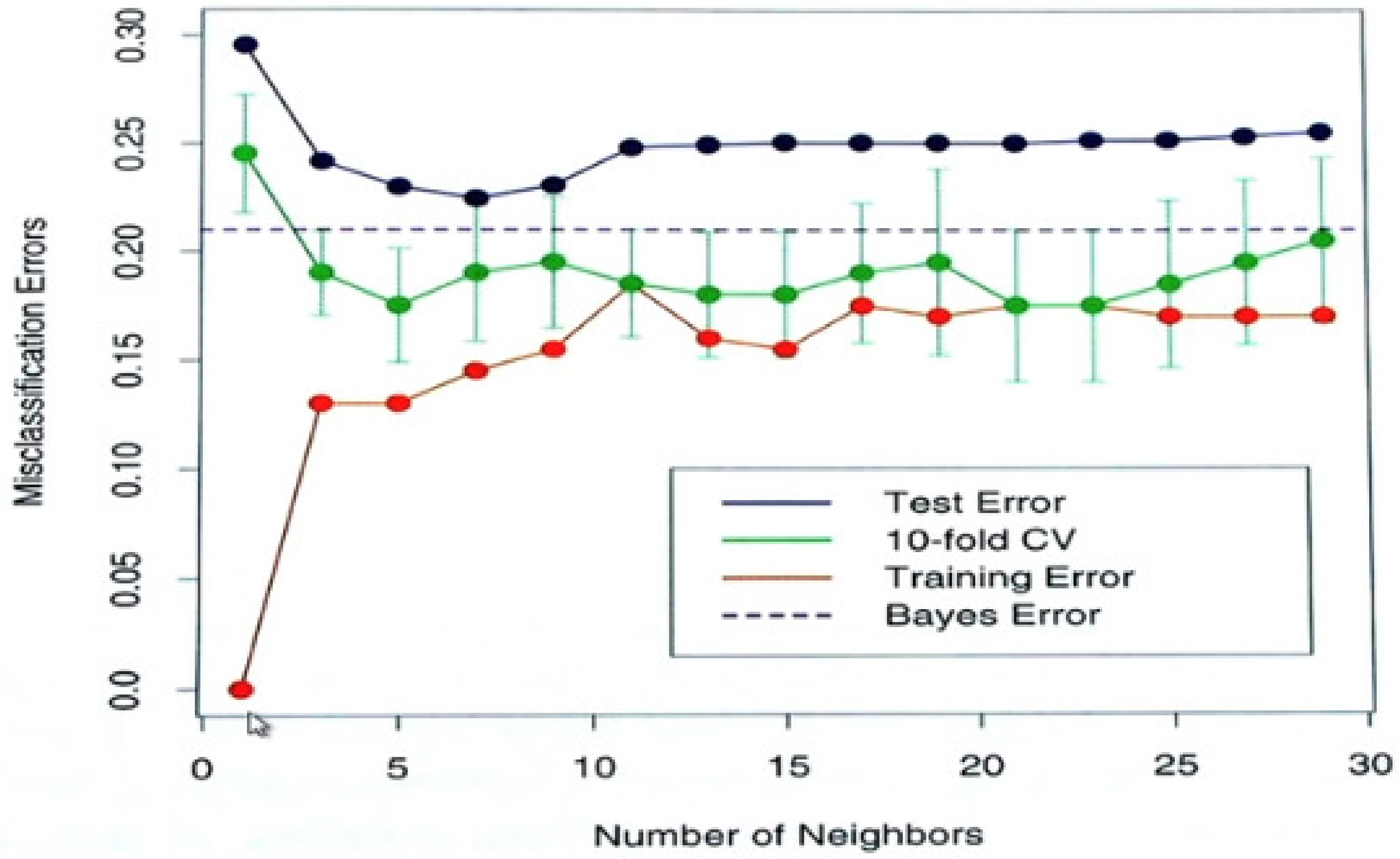
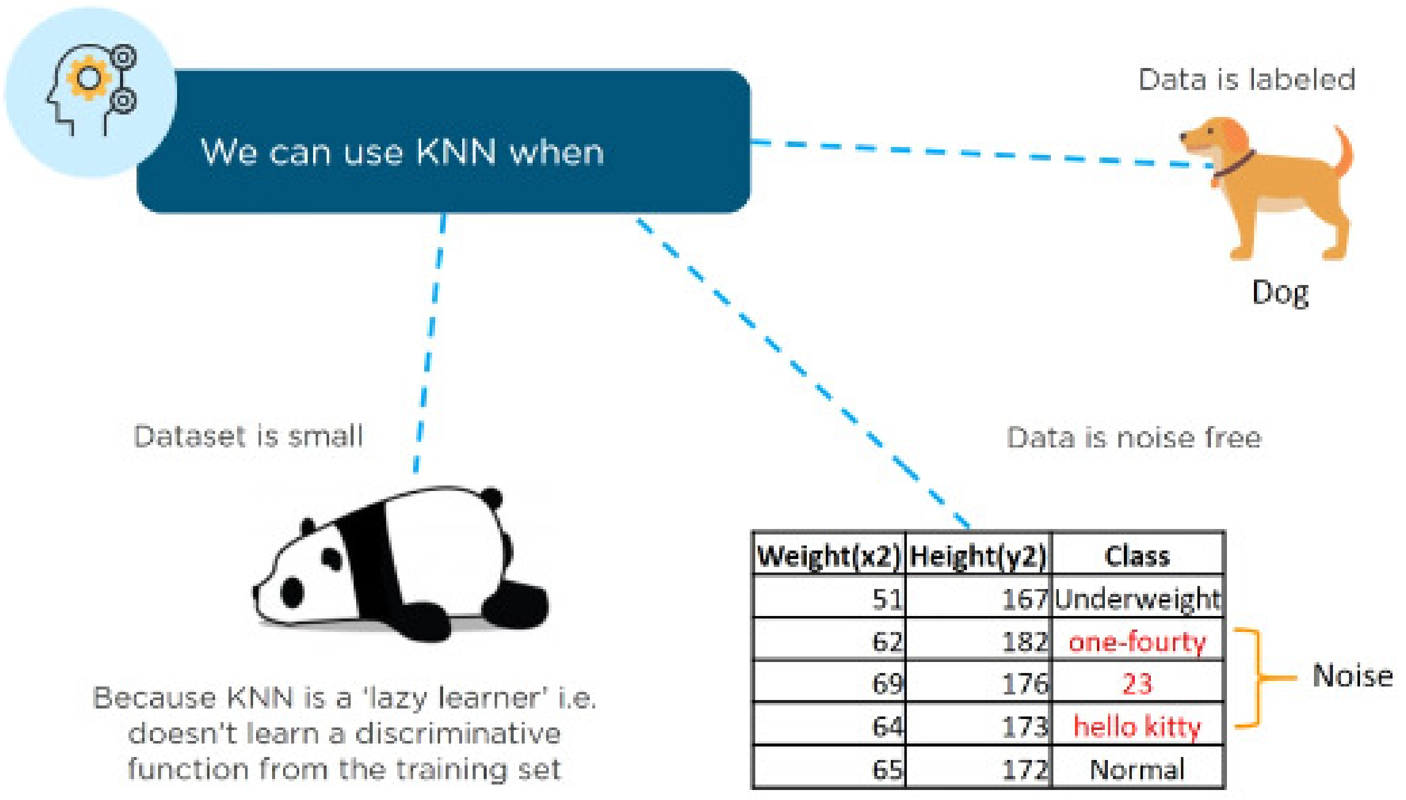


Image from Simplea

When do we use KNN Algorithm?



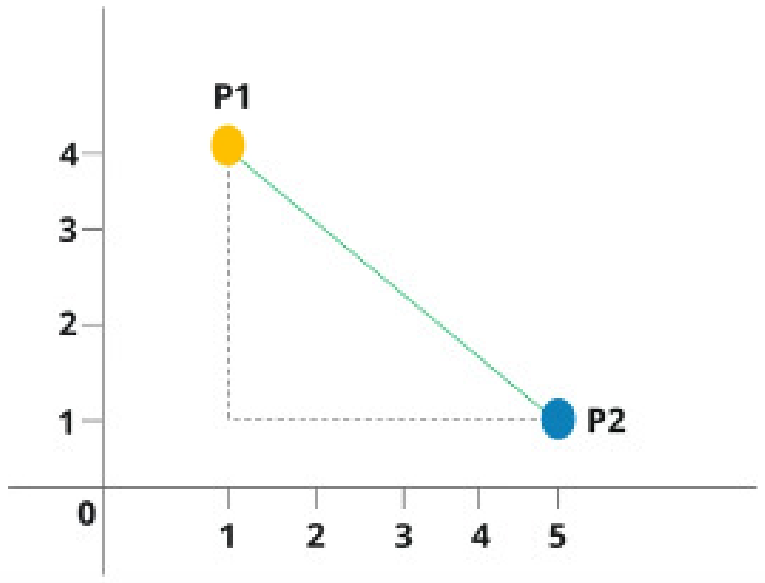
Because KNN is a 'lazy learner' i.e. doesn't learn a discriminative function from the training set

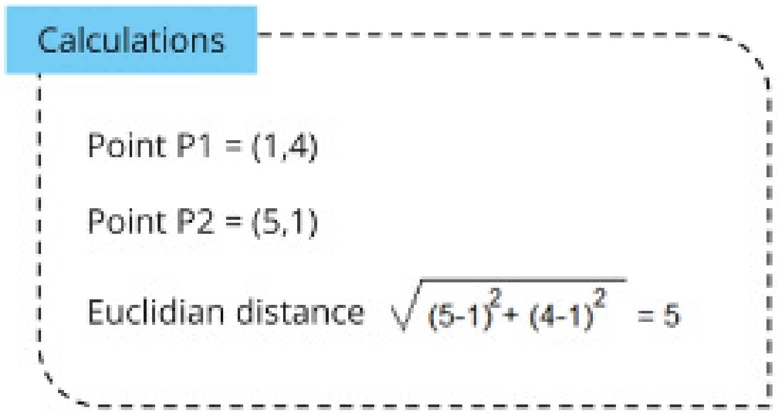
Weight(x2)	Height(y2)	Class
51	167	Underweight
62	182	one-fourty
69	176	23
64	173	hello kitty
65	172	Normal

Image from Simpllea

How Does KNN Algorithm Work?

Euclidian Distance




Calculations

Point P1 = (1,4)

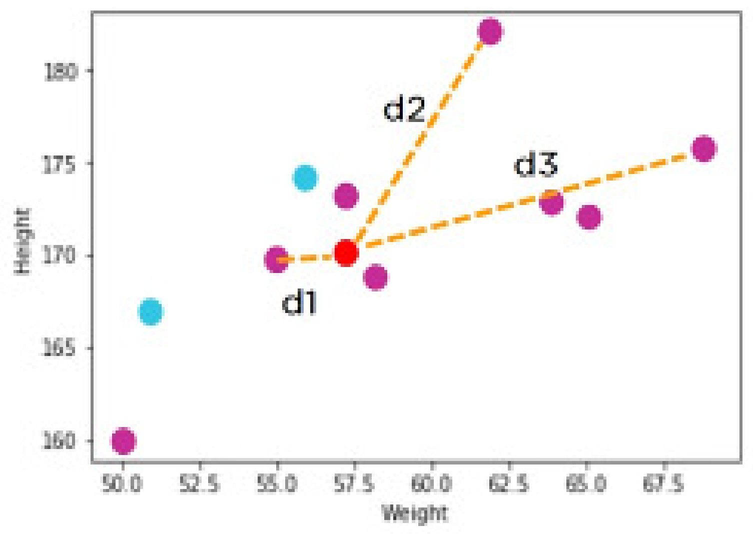
Point P2 = (5,1)

Euclidian distance $\sqrt{(5-1)^2 + (4-1)^2} = 5$

Image from Edureka

How Does KNN Algorithm Work?

Let's calculate it to understand clearly:



● Unknown data point

dist(d1)= √(170-167)² + (57-51)² -= 6.7

dist(d2)= √(170-182)² + (57-62)² -= 13

dist(d3)= √(170-176)² + (57-69)² -= 13.4

Similarly, we will calculate Euclidean distance of unknown data point from all the points in the dataset

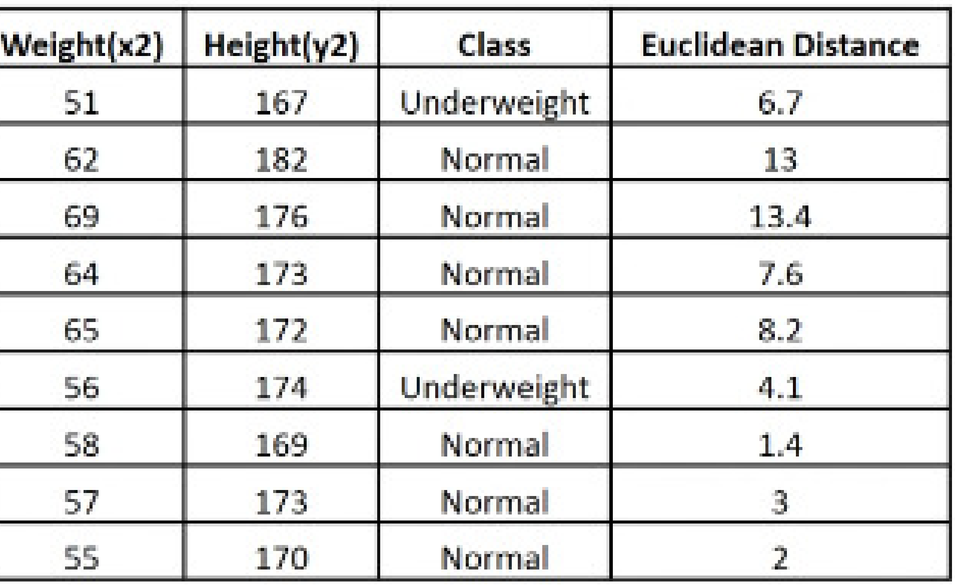
Image from Edureka

How Does KNN Algorithm Work?

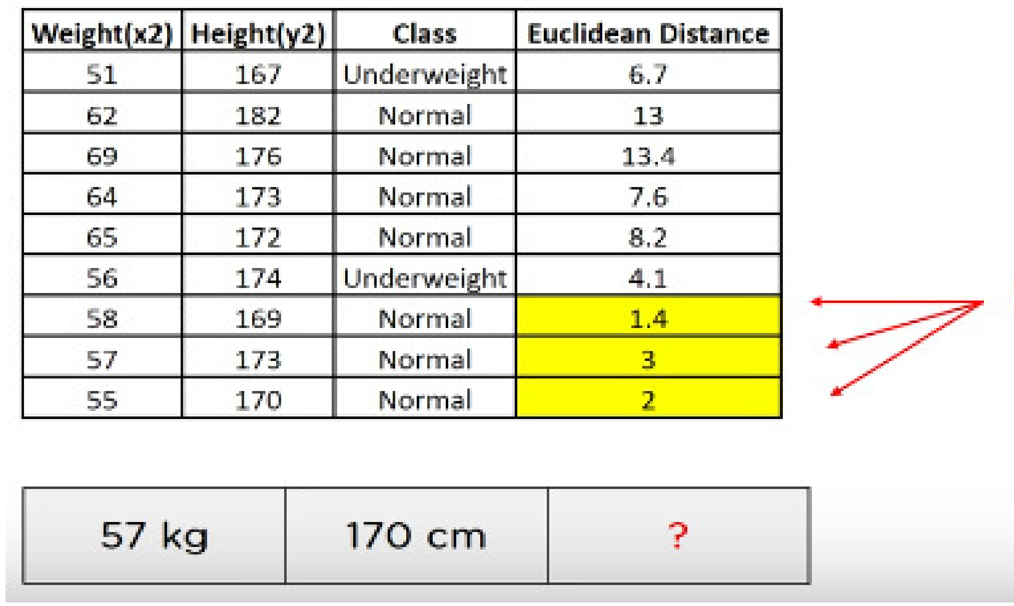
Hence, we have calculated the Euclidean distance of unknown data point from all the points as shown:

Where (x1, y1) = (57, 170) whose class we have to classify

Now, lets calculate the nearest neighbor at



Weight(x2)	Height(y2)	Class	Euclidean Distance
51	167	Underweight	6.7
62	182	Normal	13
69	176	Normal	13.4
64	173	Normal	7.6
65	172	Normal	8.2
56	174	Underweight	4.1
58	169	Normal	1.4
57	173	Normal	3
55	170	Normal	2

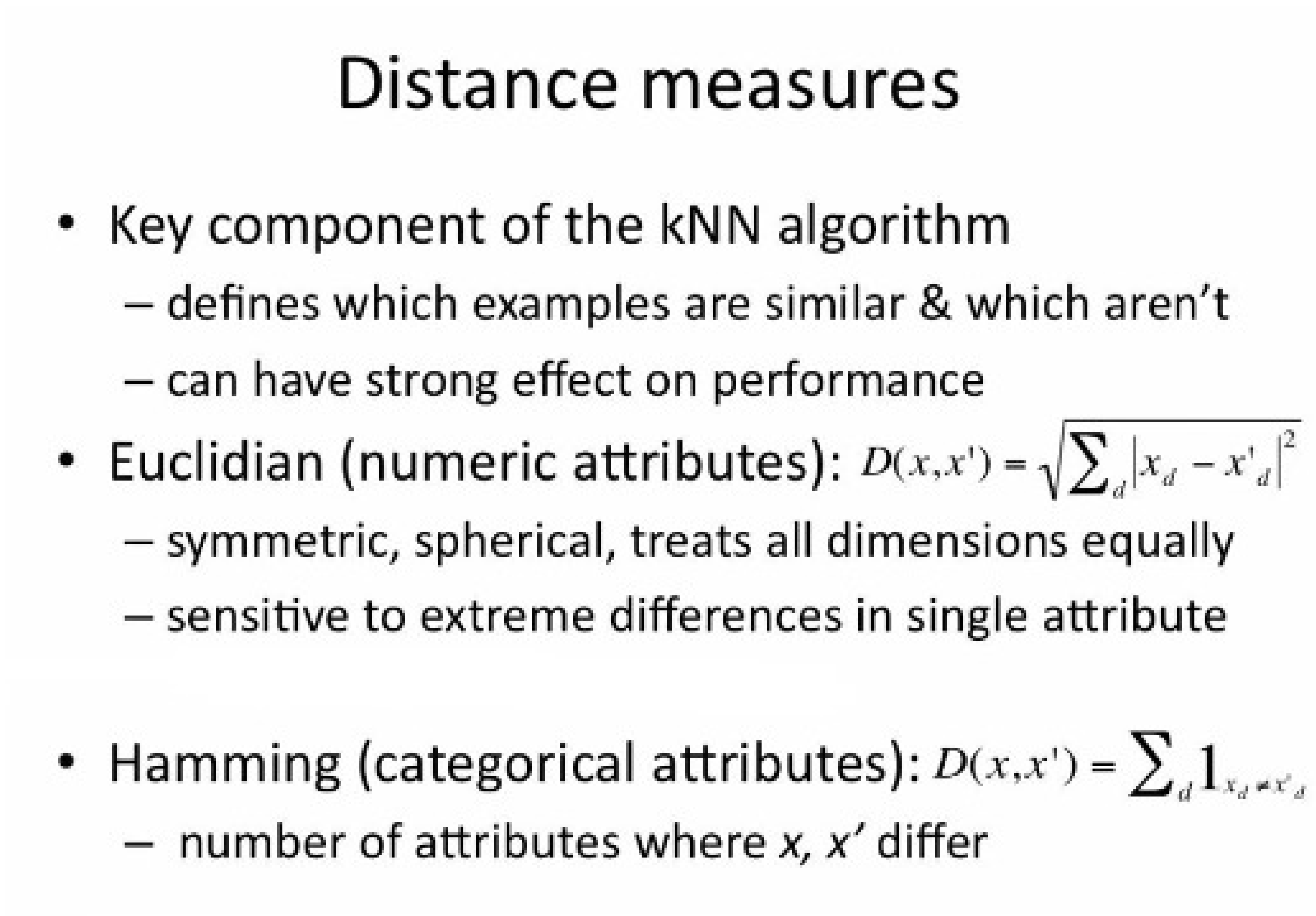


Weight(x2)	Height(y2)	Class	Euclidean Distance
51	167	Underweight	6.7
62	182	Normal	13
69	176	Normal	13.4
64	173	Normal	7.6
65	172	Normal	8.2
56	174	Underweight	4.1
58	169	Normal	1.4
57	173	Normal	3
55	170	Normal	2

57 kg	170 cm	?

So, majority neighbors are pointing towards '*Normal*'

Hence, as per KNN algorithm the class of (57, 170) should be 'Normal'

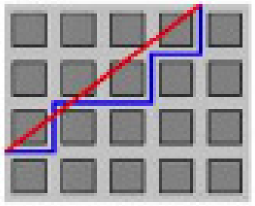


Distance measures

- Key component of the kNN algorithm
  - defines which examples are similar & which aren't
  - can have strong effect on performance
- Euclidian (numeric attributes): $D(x,x') = \sqrt{\sum_d |x_d - x'_d|^2}$
  - symmetric, spherical, treats all dimensions equally
  - sensitive to extreme differences in single attribute
- Hamming (categorical attributes): $D(x,x') = \sum_d 1_{x_d \neq x'_d}$
  - number of attributes where *x*, *x'* differ

Manhattan Distance between (x_1, y_1) and (x_2, y_2) = $|x_1 - x_2| + |y_1 - y_2|$

- Minkowski distance (p-norm): $D(x,x') = \sqrt[p]{\sum_d |x_d - x'_d|^p}$
  - p=2: Euclidian
  - p=1: Manhattan
  - p=0: Hamming
  - p=∞: $\max_d |x_d - x'_d|$. **chebyshev distance**



Weighted Euclidean distance

$$dist(x_i, x_j) = \sqrt{w_1(x^{i1} - x^{j1})^2 + w_2(x^{i2} - x^{j2})^2 + \ldots + w_r(x^{ir} - x^{jr})^2}$$

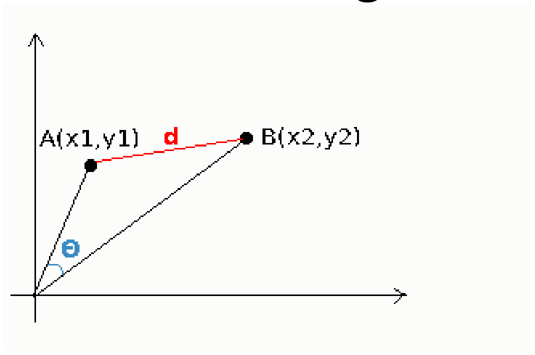
Chebychev distance: one wants to define two data points as "different" if they are different on any one of the attributes.

$$dist(x_i, x_j) = \max(|x_{i1} - x_{j1}|, |x_{i2} - x_{j2}|, \ldots, |x_{ir} - x_{jr}|)$$

The Cosine Similarity

The cosine similarity between two vectors (or two documents on the Vector Space) is a measure that calculates the cosine of the angle between them.

This metric is a measurement of orientation and not magnitude.



$$\vec{a} \cdot \vec{b} = \|\vec{a}\|\|\vec{b}\| \cos$$

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|\|\vec{b}\|}$$

When to Use Cosine?

Cosine similarity is generally used as a metric for measuring distance when the magnitude of the vectors does not matter. This happens for example when working with text data represented by word counts. We could assume that when a word (e.g. science) occurs more frequent in document 1 than it does in document 2, that document 1 is more related to the topic of science. However, it could also be the case that we are working with documents of uneven lengths (Wikipedia articles for example). Then, science probably occurred more in document 1 just because it was way longer than document 2. Cosine similarity corrects for this.

Best data practice for KNN

1. **Rescale Data**: KNN performs much better if all of the data has the same scale. Normalizing your data to the range [0, 1] is a good idea. It may also be a good idea to standardize your data if it has a Gaussian distribution.

2. **Address Missing Data**: Missing data will mean that the distance between samples can not be calculated. These samples could be excluded or the missing values could be imputed.

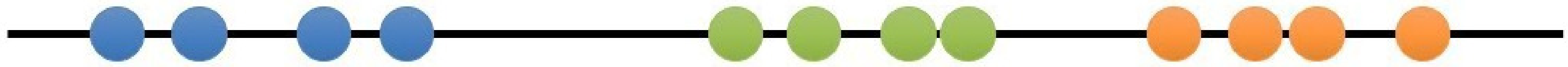
3. **Lower Dimensionality**: KNN is suited for lower dimensional data. You can try it on high dimensional data (hundreds or thousands of input variables) but be aware that it may not perform as well as other techniques. KNN can benefit from feature selection that reduces the dimensionality of the input feature space.

Pros of K Nearest Neighbors

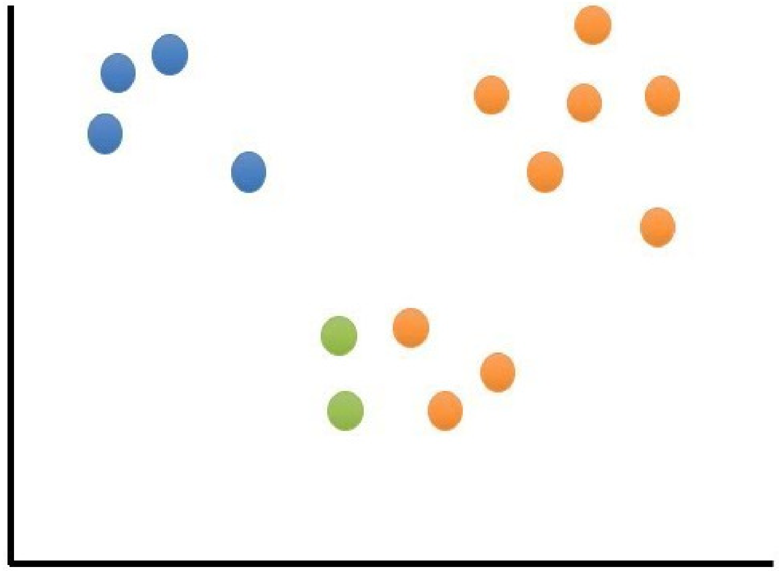
- Simple algorithm and hence easy to interpret the prediction
- Non parametric, so makes no assumption about the underlying data pattern
- used for both classification and Regression
- Training step is much faster for nearest neighbor compared to other machine learning algorithms

Cons of K Nearest Neighbors

•KNN is computationally expensive as it searches the nearest neighbors for the new point at the prediction stage
•High memory requirement as KNN has to store all the data points
•Prediction stage is very costly
•Sensitive to outliers, accuracy is impacted by noise or irrelevant data.



K-Means Clustering



Clustering is a method of unsupervised learning

where the data we want to describe is not labeled.

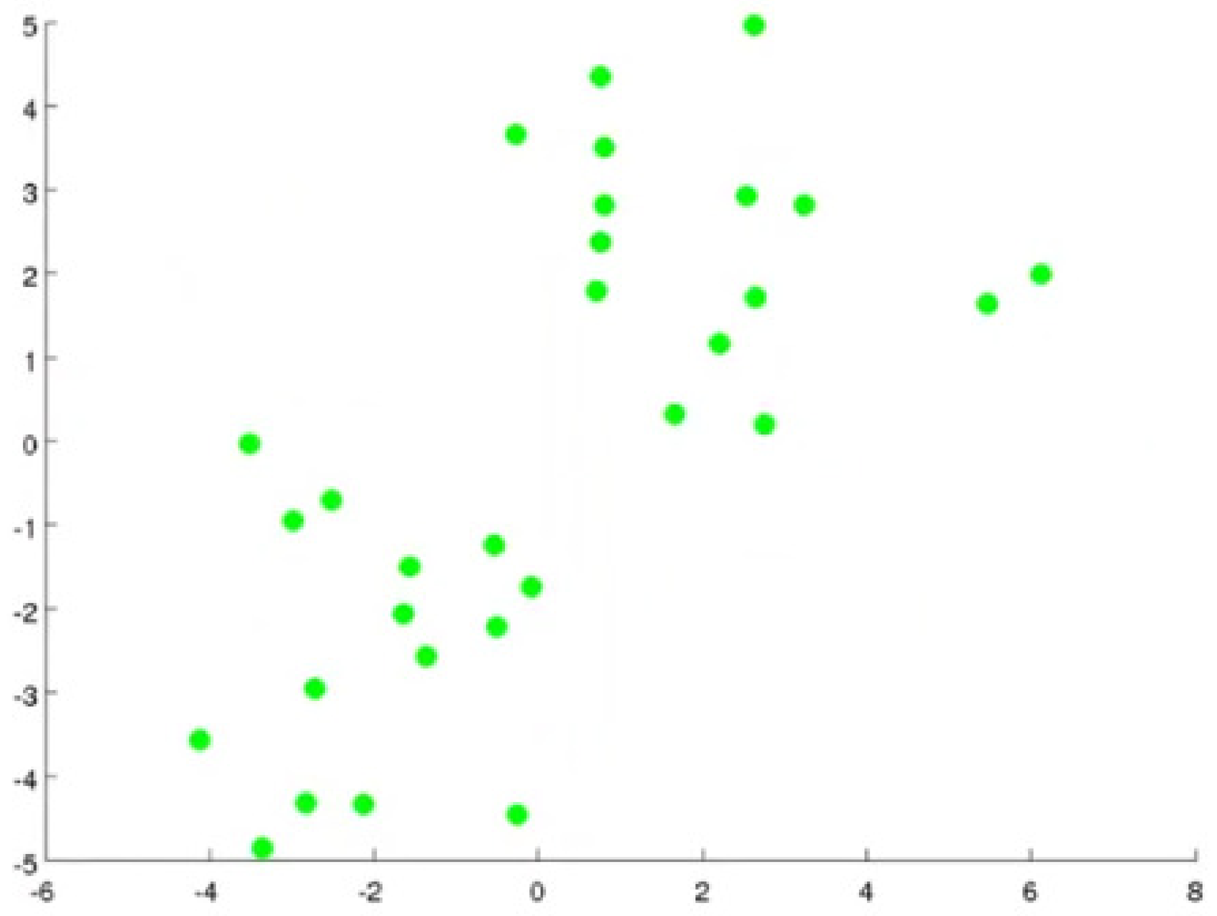
Where user does not give us much information of what is the expected output.

✓**Grouping** of data points.
✓Given a set of data points, clustering algorithm **classify each data point into a specific group.**
✓Data points that are in the **same group should have similar properties** and/or features.
✓Data points in **different groups should have highly dissimilar properties** and/or features.

Clustering

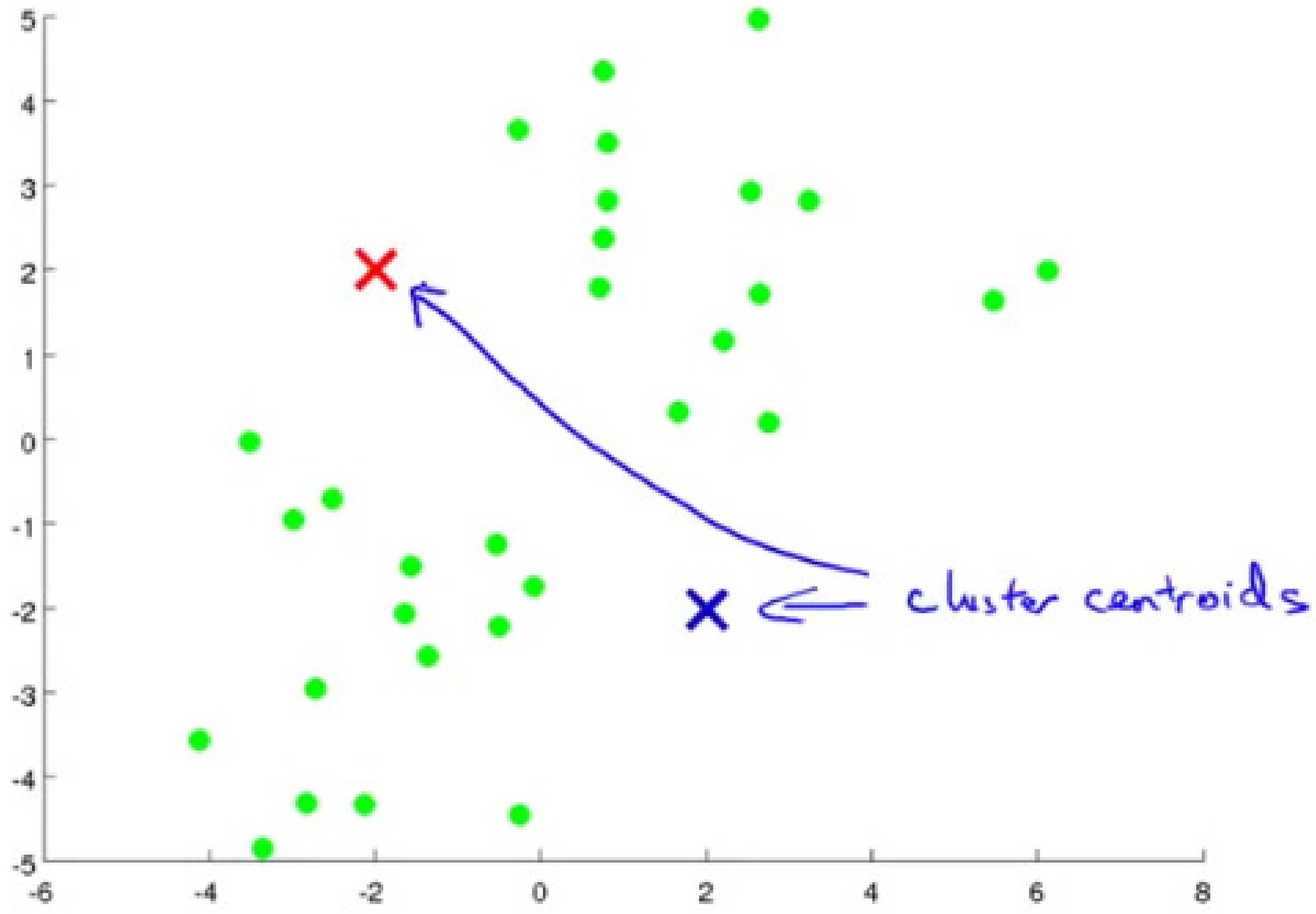
K-means algorithm

Image from AndrewN



Given all the unlabeled data points

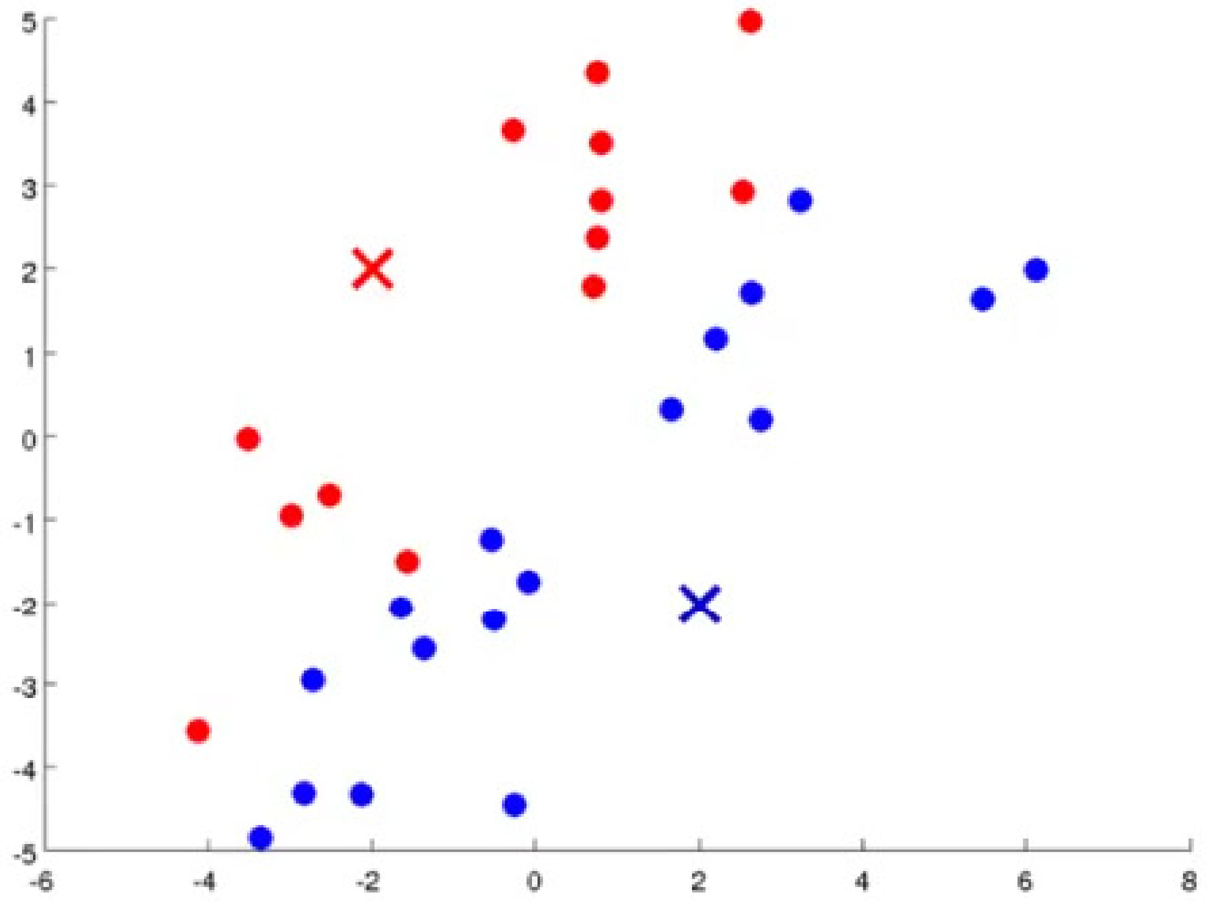
Image from AndrewN



➢First select a number of classes/groups for clustering.

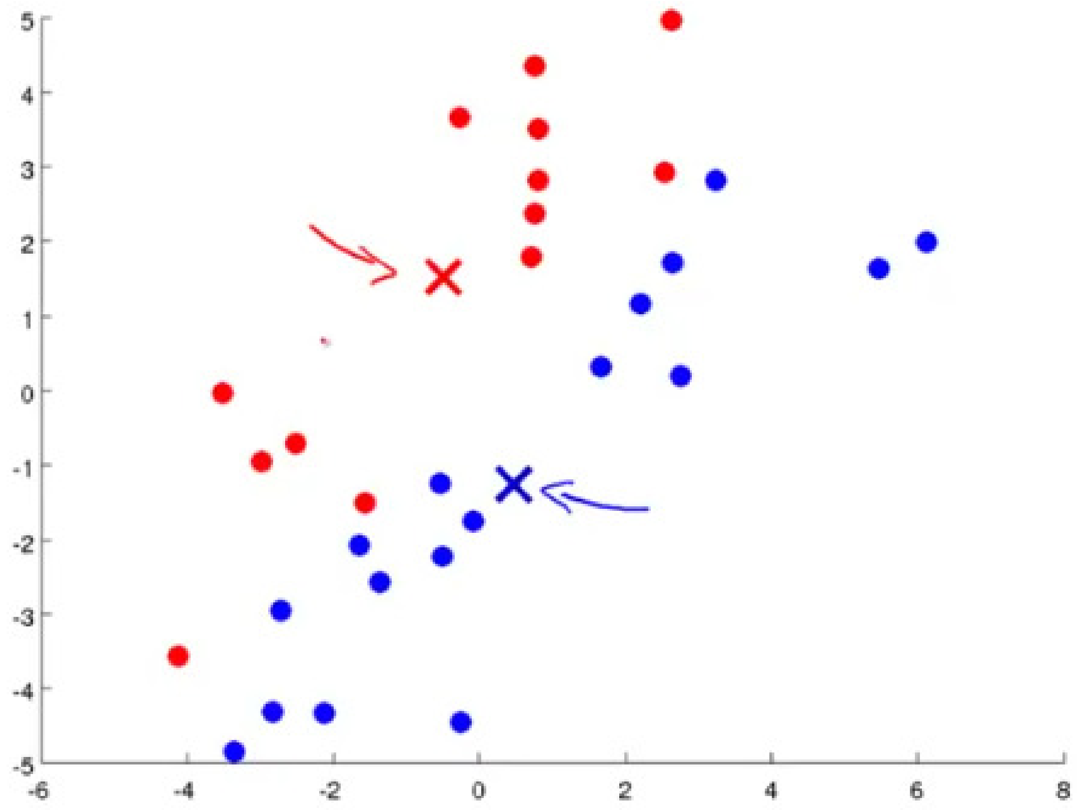
➢Randomly initialize their respective center points called cluster centroids.

Image from AndrewN



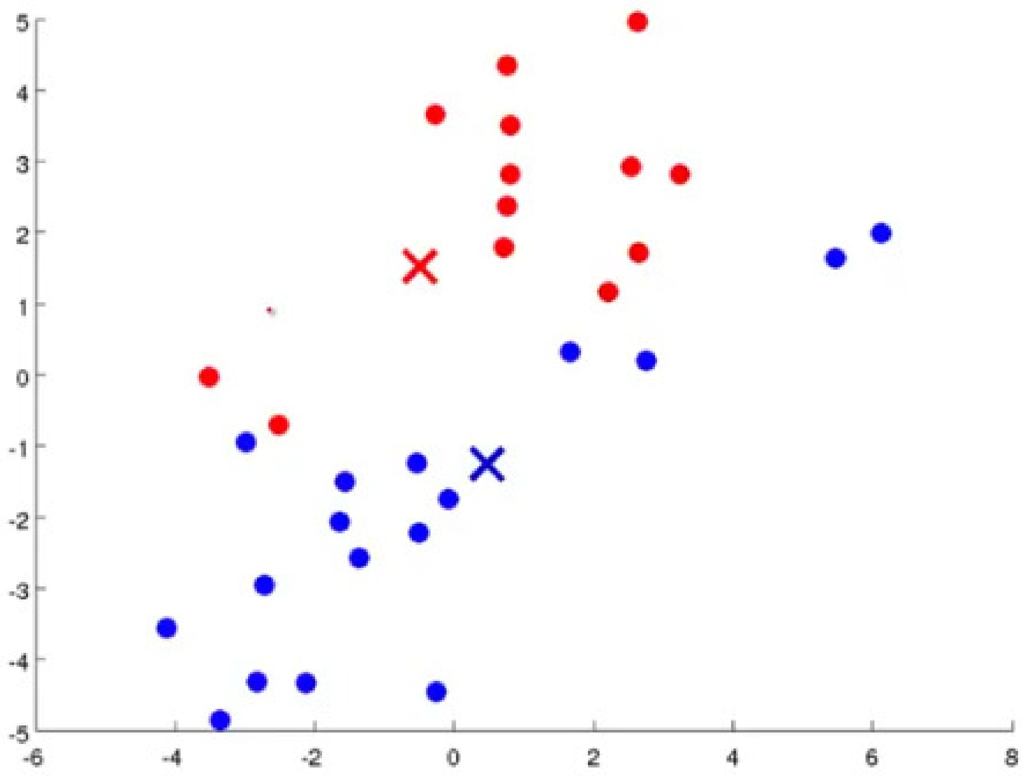
Each data point is classified by computing the distance between that point and each centroid and then classifying the point to be in the group whose center is closest to it.

Image from AndrewN

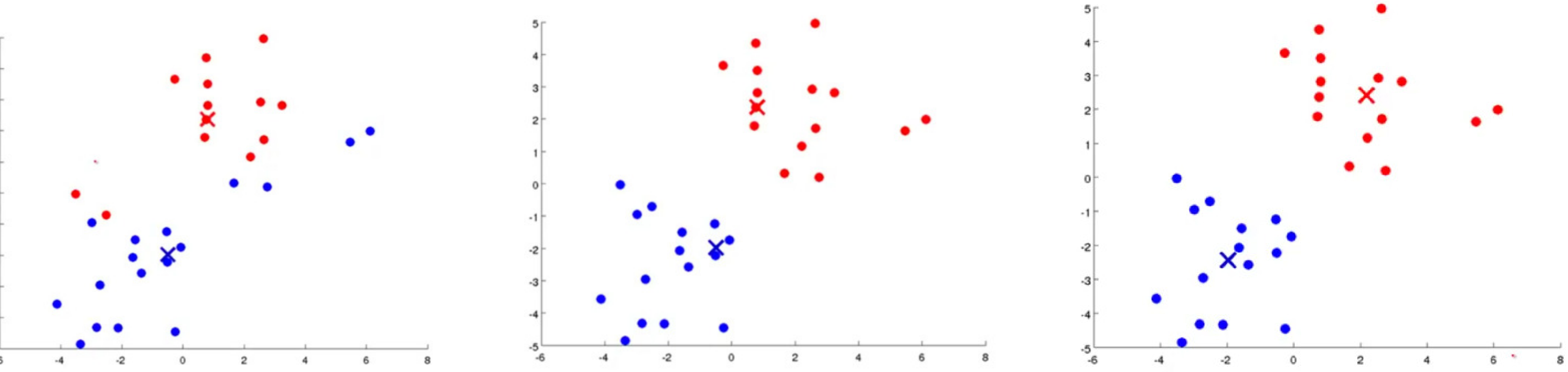


Based on these classified points, re-compute the group center by taking the mean of al the data point vectors in the group and shift the centroids.

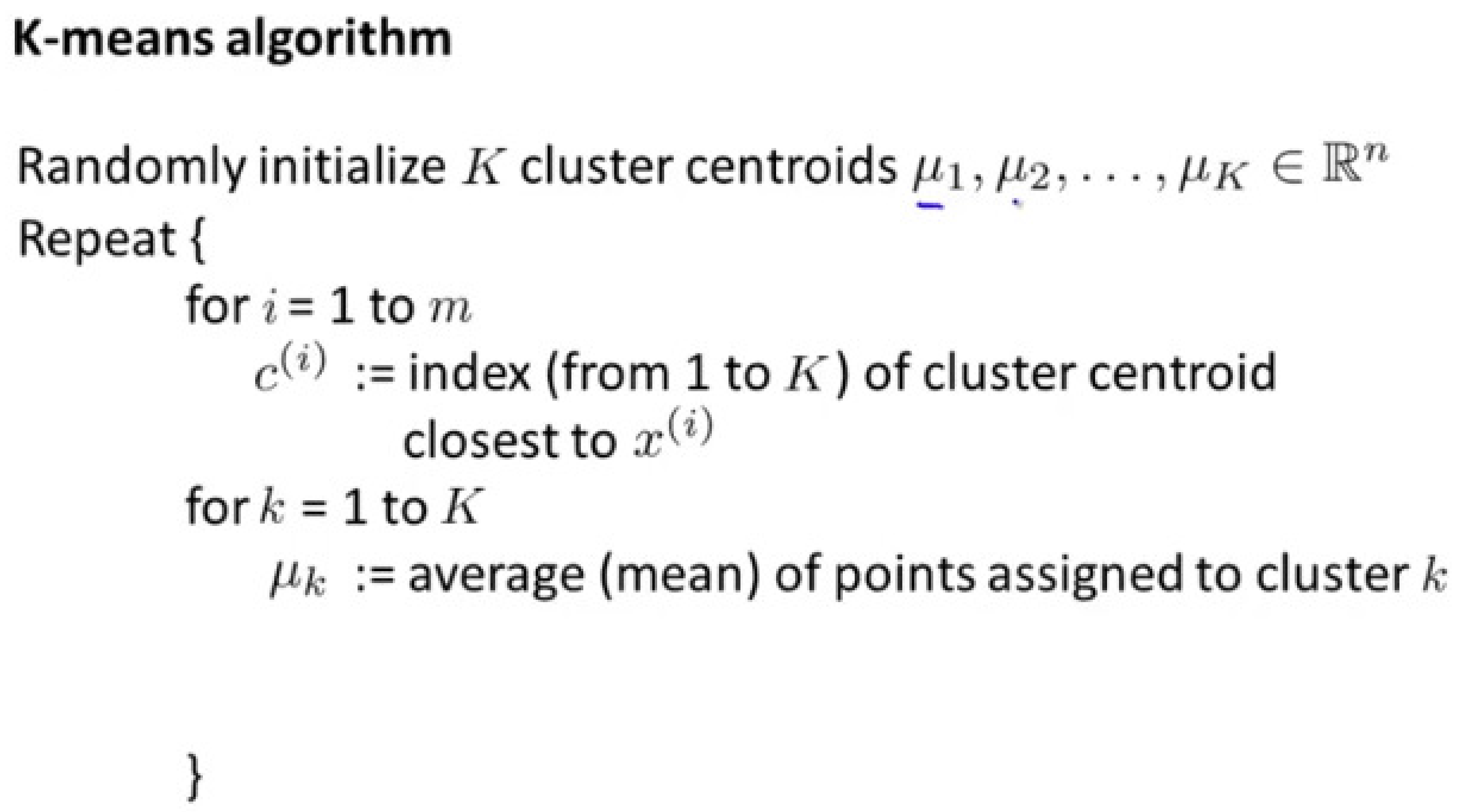
Image from AndrewN



Now again calculate the distance of each data point vectors from the new position of centroids and assign them to any of the centroids on the basis of the minimum distanc



Repeat these steps for a set number of iterations or until the group centers don't change much between iterations.



K-means algorithm

```
Randomly initialize $K$ cluster centroids $\mu_1, \mu_2, \ldots, \mu_K \in \mathbb{R}^n$
Repeat {
        for $i$ = 1 to $m$
            $c^{(i)}$ := index (from 1 to $K$) of cluster centroid
                    closest to $x^{(i)}$
        for $k$ = 1 to $K$
            $\mu_k$ := average (mean) of points assigned to cluster $k$


        }
```

Image from AndrewN

K-means optimization objective

$\rightarrow$ $c^{(i)}$ = index of cluster (1,2,…,K) to which example $x^{(i)}$ is currently assigned

$\rightarrow$ μ_k = cluster centroid $\underline{k}$ ($\mu_k \in \mathbb{R}^n$) K $k \in \{1, 2, \ldots, K\}$

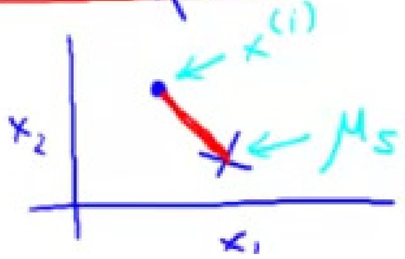
$\mu_{c^{(i)}}$ = cluster centroid of cluster to which example $x^{(i)}$ has been assigned $x^{(i)} \rightarrow \underline{5}$ $\underline{c^{(i)} = 5}$ $\underline{\mu_{c^{(i)}}} = \mu_5$

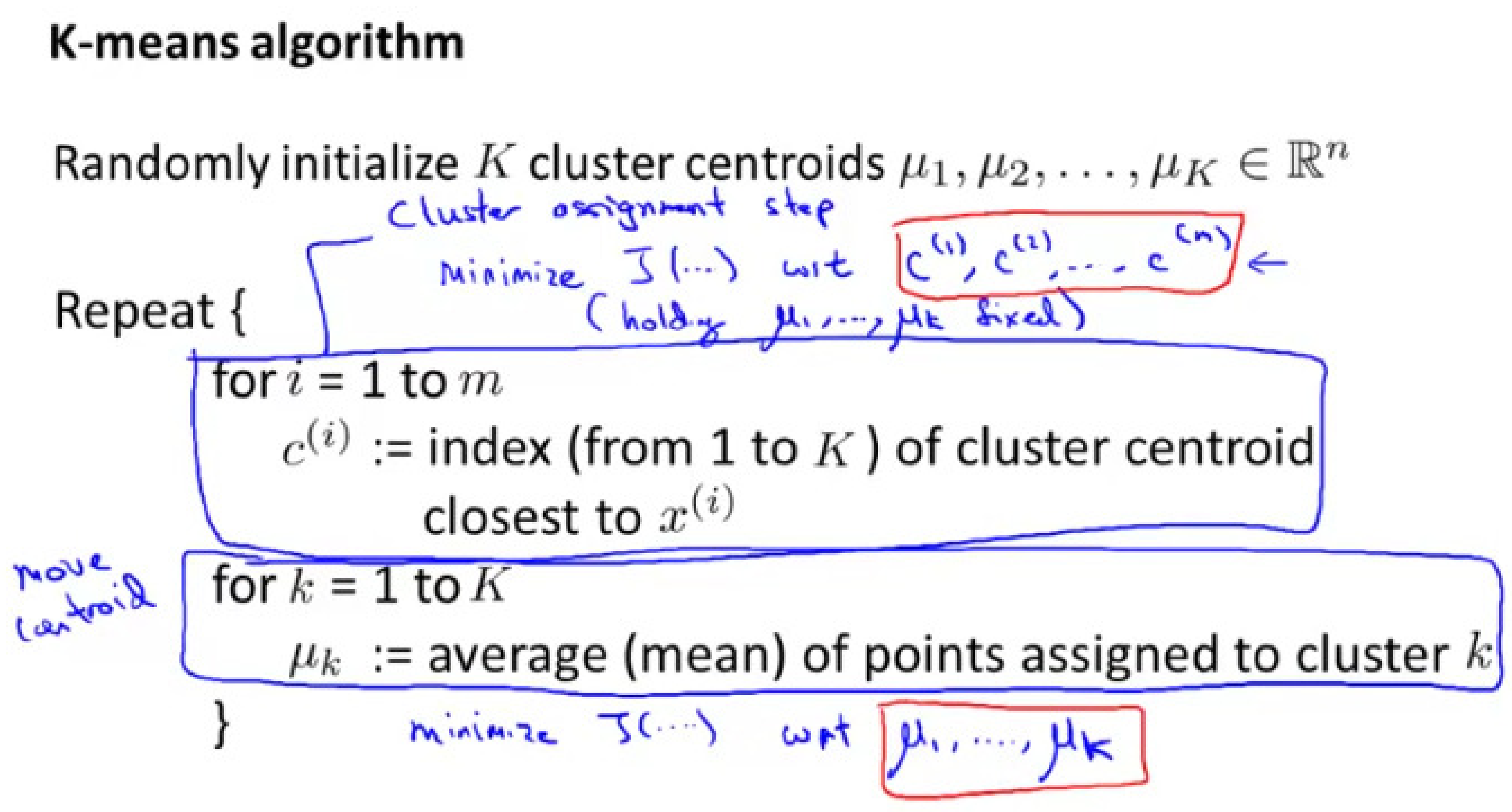
Optimization objective:

$$\rightarrow \underline{J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K)} = \frac{1}{m} \sum_{i=1}^{m} \boxed{||\underset{\uparrow}{x^{(i)}} - \underset{\uparrow}{\mu_{c^{(i)}}}||^2} \leftarrow$$

$$\min_{\substack{\rightarrow c^{(1)}, \ldots, c^{(m)}, \\ \rightarrow \mu_1, \ldots, \mu_K}} J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K)$$

Distortio





K-means algorithm

Randomly initialize K cluster centroids $\mu_1, \mu_2, \dots, \mu_K \in \mathbb{R}^n$

Cluster assignment step

minimize $J(\dots)$ wrt $\boxed{c^{(1)}, c^{(2)}, \dots, c^{(m)}}$ $\leftarrow$

(holding $\mu_1, \dots, \mu_K$ fixed)

```
Repeat {
    for $i$ = 1 to $m$
        $c^{(i)}$ := index (from 1 to $K$) of cluster centroid
                closest to $x^{(i)}$
    for $k$ = 1 to $K$
        $\mu_k$ := average (mean) of points assigned to cluster $k$
}
```

move centroid

minimize $J(\dots)$ wrt $\boxed{\mu_1, \dots, \mu_K}$

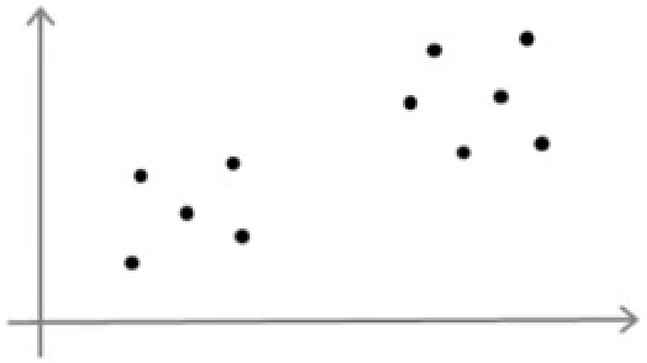
Image from AndrewN

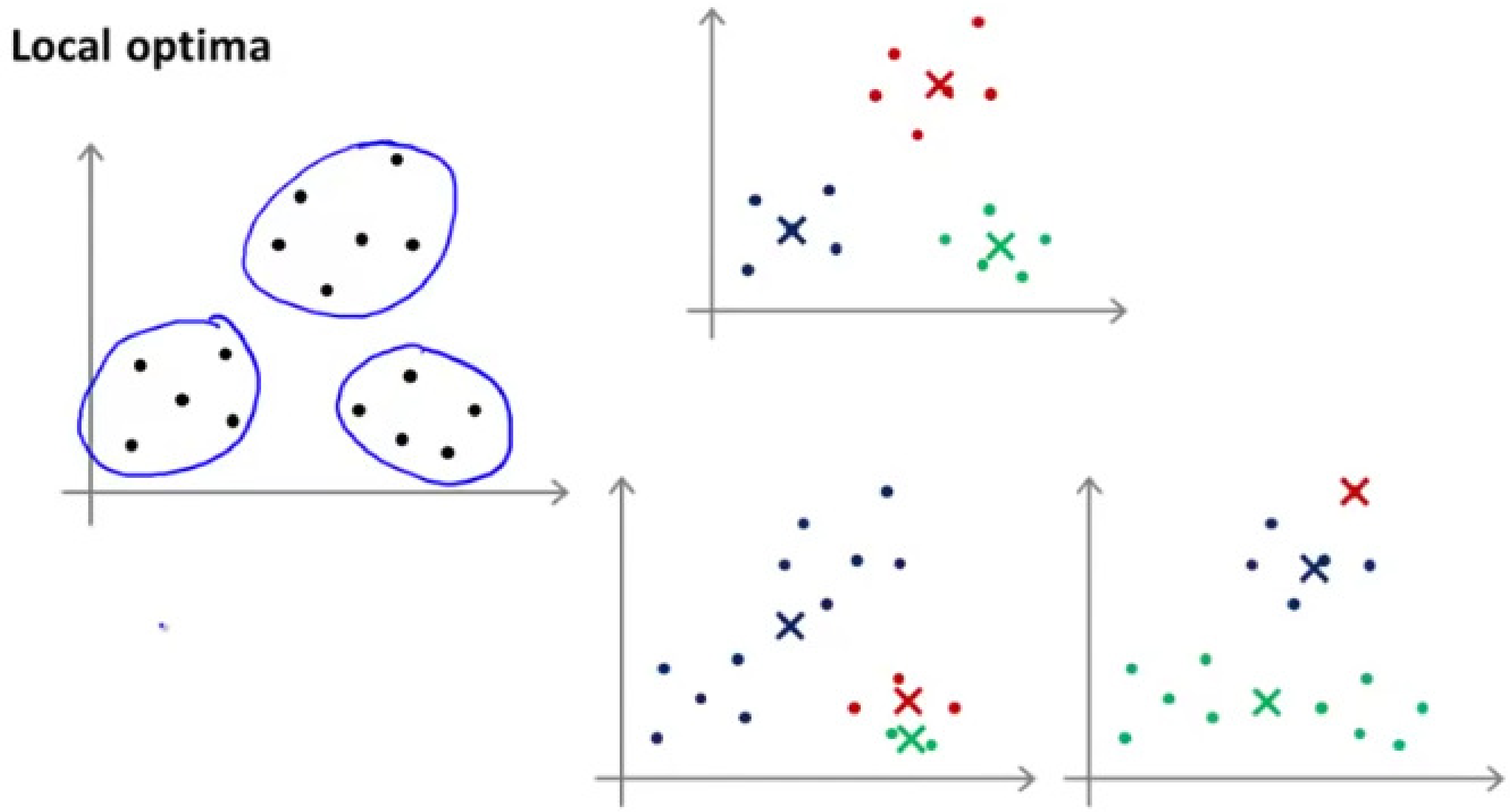
Random initialization

Should have $K < m$

Randomly pick K training examples.

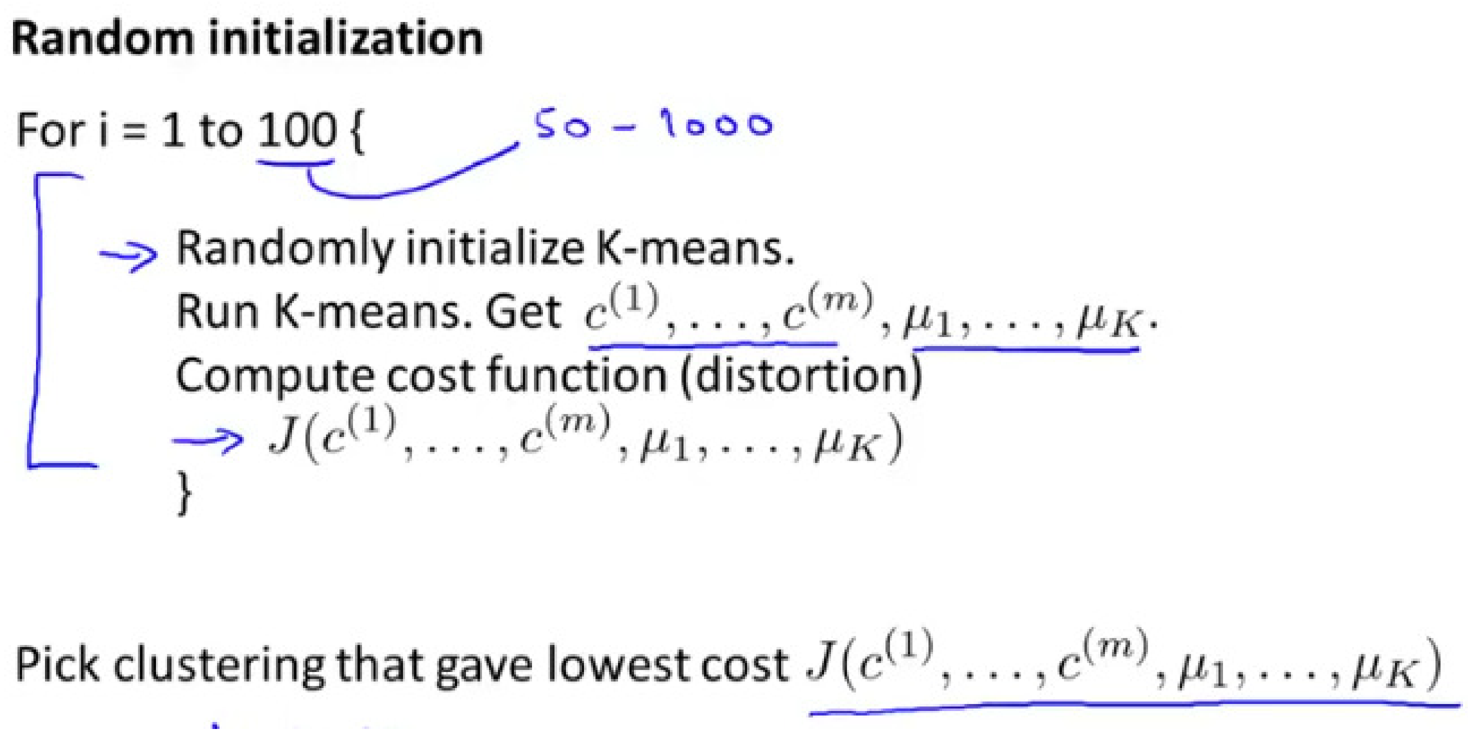
Set $\mu_1, \ldots, \mu_K$ equal to these K examples.





Clustering gets affected by the random initialization of the centroids.

Image from AndrewN



Random initialization

```
For i = 1 to $\underline{100}$ {        50 - 1000
    → Randomly initialize K-means.
      Run K-means. Get $\underline{c^{(1)},\dots,c^{(m)}},\underline{\mu_1,\dots,\mu_K}$.
      Compute cost function (distortion)
      → $J(c^{(1)},\dots,c^{(m)},\mu_1,\dots,\mu_K)$
}
```

Pick clustering that gave lowest cost $\underline{J(c^{(1)},\dots,c^{(m)},\mu_1,\dots,\mu_K)}$

If 2=<K <= 10 random initialization has greater impact on clustering but if k>=10 random initialization has very less impact on clustering and most probably very first random initialization of centroids will give pretty good result..

Image from AndrewN

What is the right value of K?

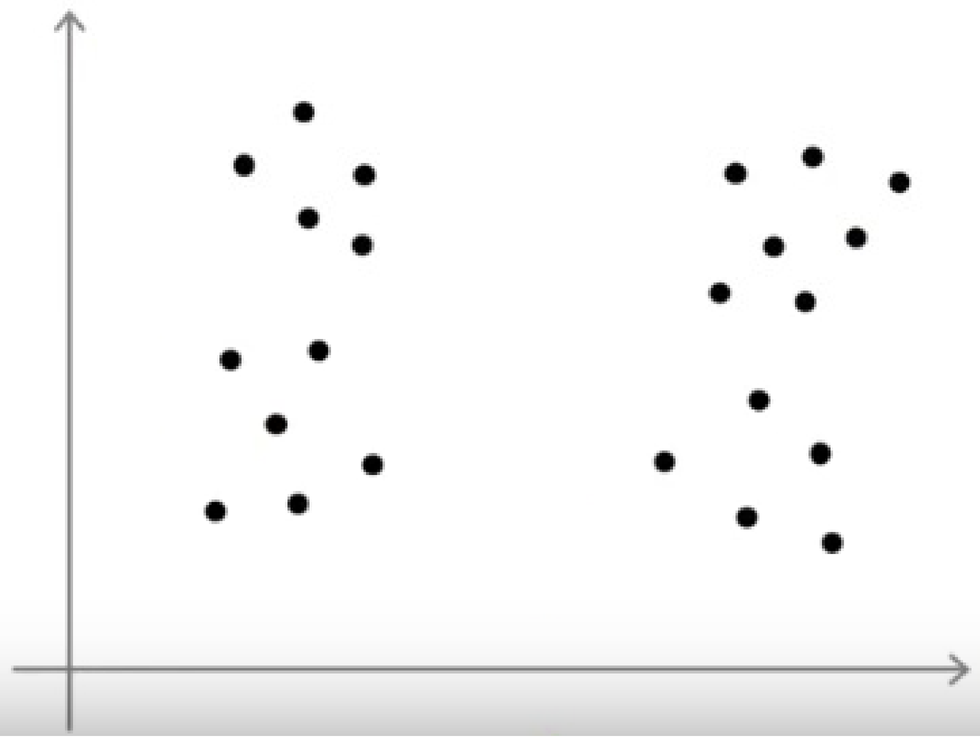
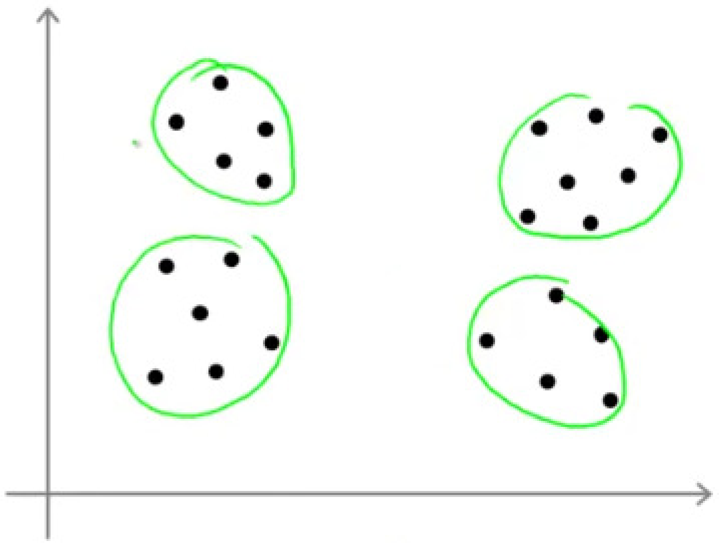


Image from AndrewN

What is the right value of K?



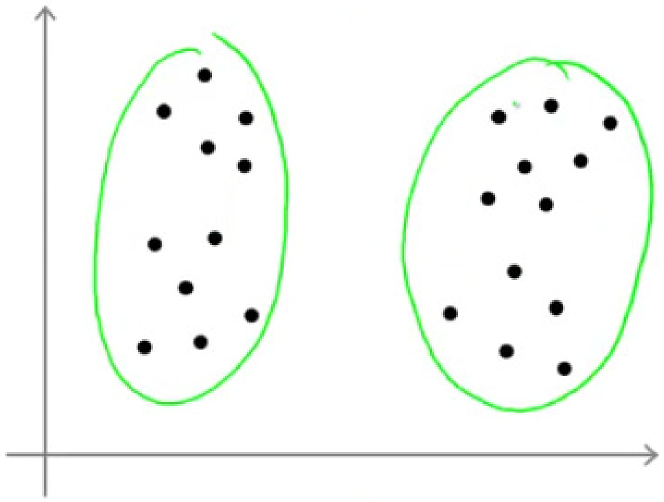
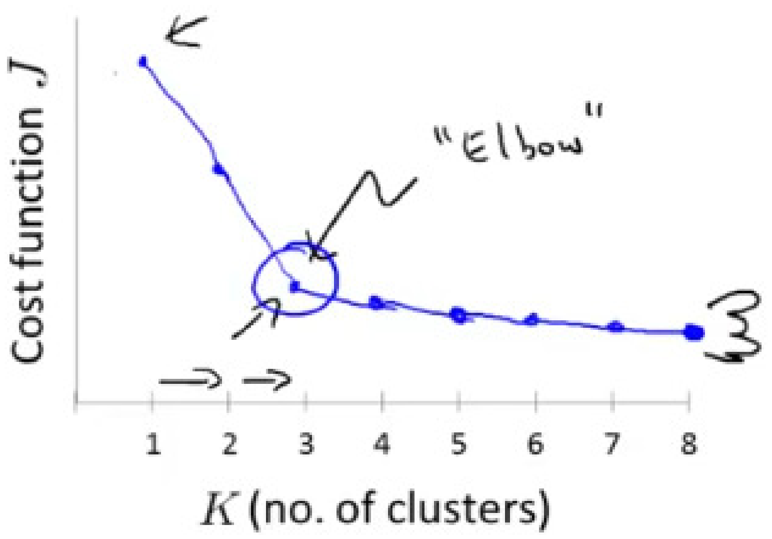
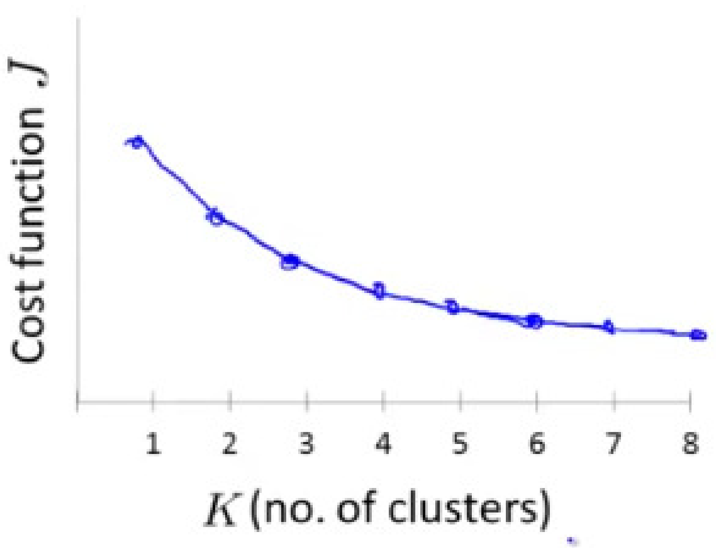


Image from AndrewN

Choosing the value of K

Elbow method:





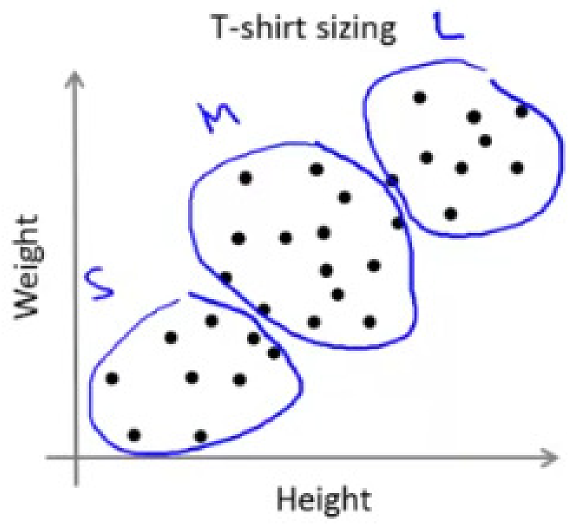
Choosing the value of K

Sometimes, you're running K-means to get clusters to use for some later/downstream purpose. Evaluate K-means based on a metric for how well it performs for that later purpose.

K=3 S, M, L

K=5 XS, S, M, L, XL

E.g.



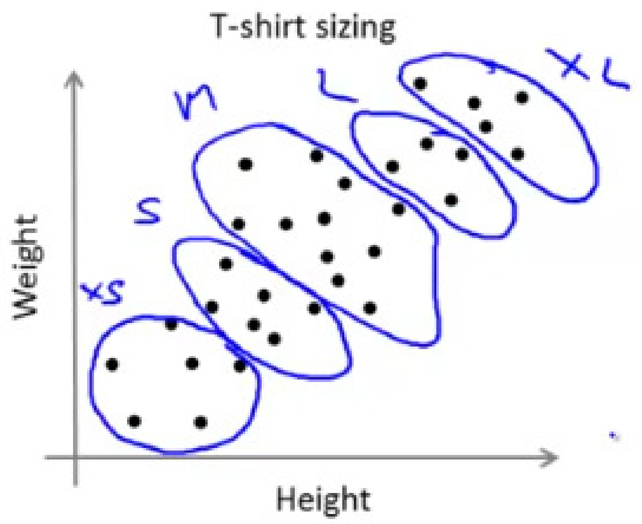


Image from AndrewN

Top 5 Clustering Algorithms

1. **K-Means Clustering**
2. **Mean-Shift Clustering**
3. **Density-Based Spatial Clustering of Applications with Noise (DBSCAN)**
4. **Expectation–Maximization (EM) Clustering using Gaussian Mixture Models (GMM)**
5. **Agglomerative Hierarchical Clustering**

https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68

Image from AndrewN