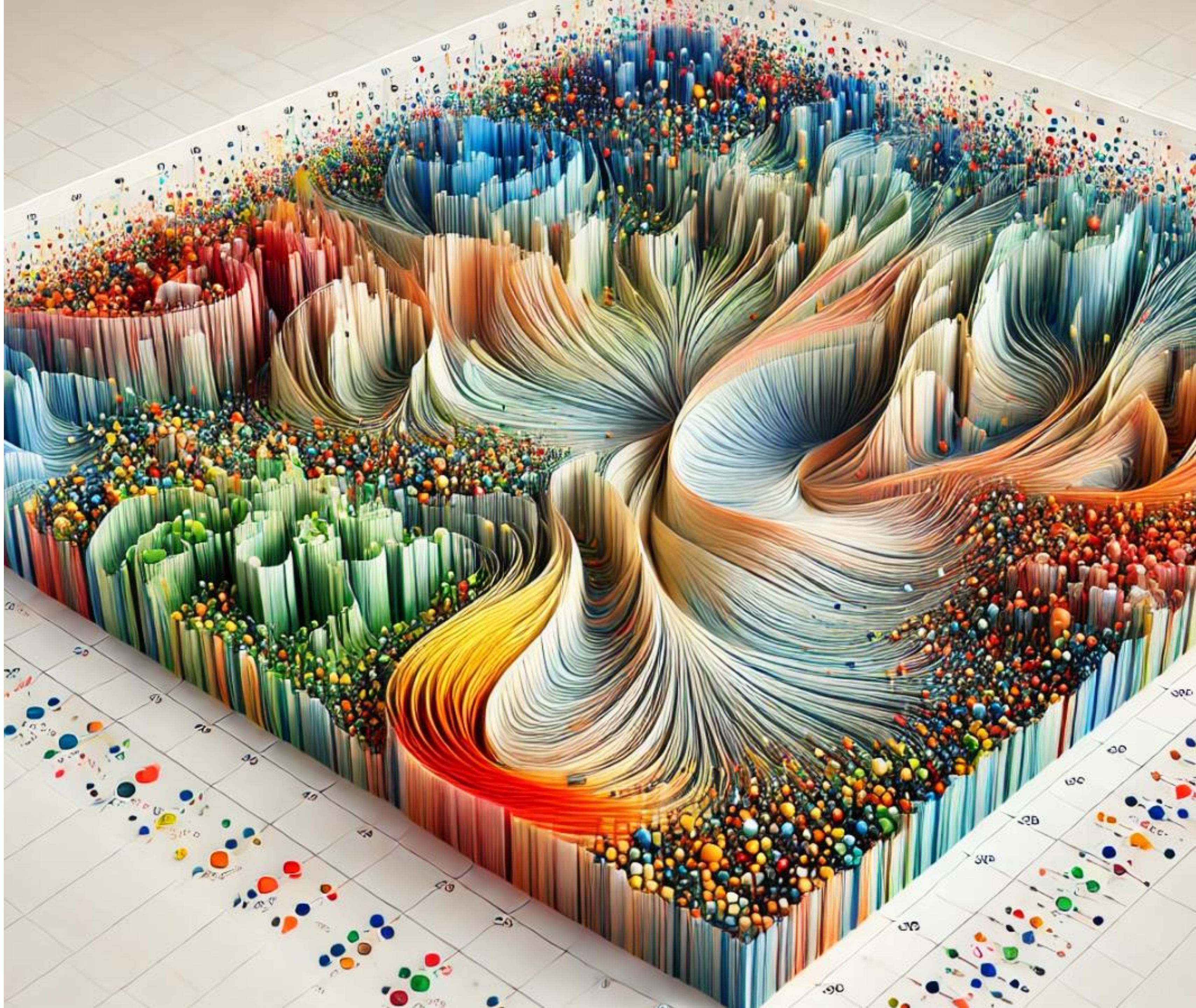
K-Nearest Neighbors

- ISE – 364 / 464
- DEPT. OF INDUSTRIAL AND SYSTEMS ENGINEERING
- GRIFFIN DEAN KENT

LEHIGH UNIVERSITY®



Overview of K-NN

K-Nearest Neighbors

Perhaps the simplest machine learning algorithm (even conceptually simpler than linear regression) is the **K-Nearest Neighbors** (K-NN) algorithm.

- The intuition behind the K-NN model is simple: the numerical value (or target class) of a datapoint is solely determined by some number K of the "nearest" datapoints (in a spatial distance sense) numerical values (or target classes). We can write the **distance function** between any two datapoints a and b as the general function $D(a, b)$.

Overview of K-NN

K-Nearest Neighbors

Perhaps the simplest machine learning algorithm (even conceptually simpler than linear regression) is the **K-Nearest Neighbors** (K-NN) algorithm.

- The intuition behind the K-NN model is simple: the numerical value (or target class) of a datapoint is solely determined by some number K of the "nearest" datapoints (in a spatial distance sense) numerical values (or target classes). We can write the **distance function** between any two datapoints a and b as the general function $D(a, b)$.
- Naturally, the most common distance metric that is used is simply the Eudlidean ℓ_2-norm distance between points
$$D(a, b) := \|a - b\|_2^2.$$
- However, any type of distance function could be used (ℓ_1-norm, ℓ_∞-norm, Hamming distance, etc.). For the rest of these slides, we will be using the Euclidean distance for simplicity and the fact that it is the most common.

Overview of K-NN

K-Nearest Neighbors

Perhaps the simplest machine learning algorithm (even conceptually simpler than linear regression) is the **K-Nearest Neighbors** (K-NN) algorithm.

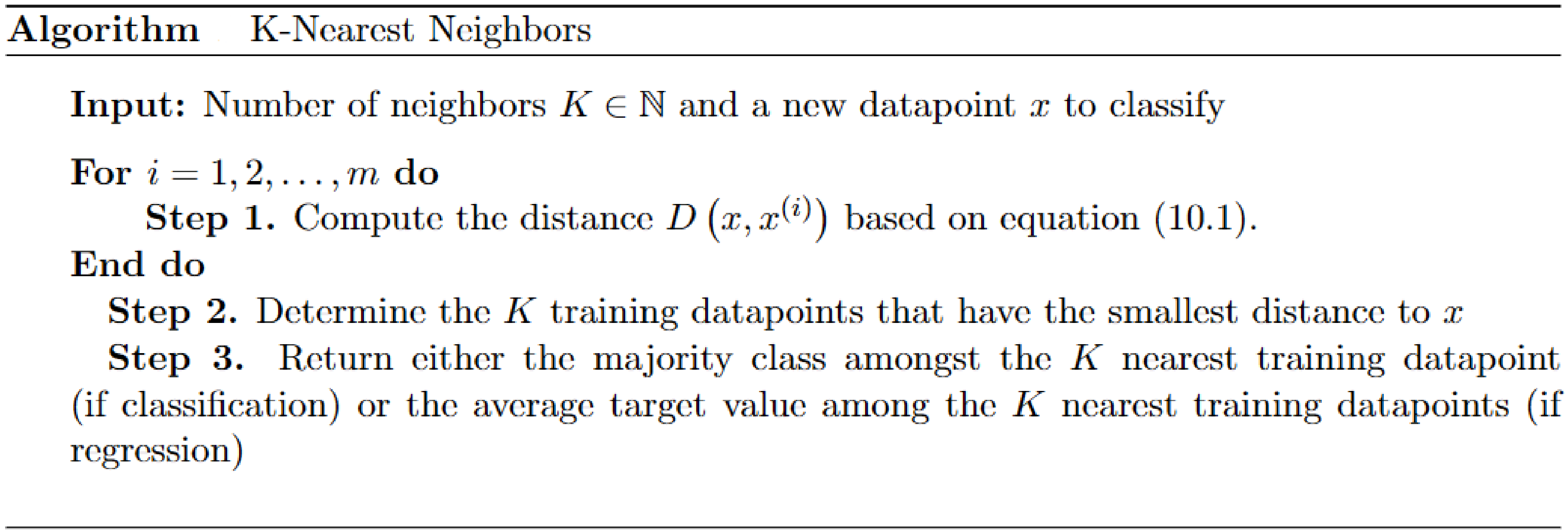
- The intuition behind the K-NN model is simple: the numerical value (or target class) of a datapoint is solely determined by some number K of the "nearest" datapoints (in a spatial distance sense) numerical values (or target classes). We can write the **distance function** between any two datapoints a and b as the general function $D(a, b)$.
- Naturally, the most common distance metric that is used is simply the Eudlidean ℓ_2-norm distance between points
$$D(a, b) \coloneqq \|a - b\|_2^2.$$
- However, any type of distance function could be used (ℓ_1-norm, ℓ_∞-norm, Hamming distance, etc.). For the rest of these slides, we will be using the Euclidean distance for simplicity and the fact that it is the most common.
- Since the K-NN model does not require "training" in the sense of determining some set of optimal parameters, it is referred to as a "**lazy learning algorithm**". However, it does require accessing every single datapoint in the dataset to classify a new point… This stands at opposition to learning algorithms which typically can be trained in a computationally stochastic setting. This option is not available for K-NN models.

K-NN Algorithm

Given some dataset $\mathcal{D} = \left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{m}$, where $x^{(i)} \in \mathbb{R}^n$ and either $y^{(i)} \in \mathbb{R}$ or $y^{(i)} \in \mathbb{N}$ (y is either numerical or categorical). Further, assume that each feature $x^{(i)}$ is standardized.

K-NN Algorithm

Given some dataset $\mathcal{D} = \left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{m}$, where $x^{(i)} \in \mathbb{R}^n$ and either $y^{(i)} \in \mathbb{R}$ or $y^{(i)} \in \mathbb{N}$ (y is either numerical or categorical). Further, assume that each feature $x^{(i)}$ is standardized. Then, we can write the K-NN algorithm as the following.



```
**Algorithm**    K-Nearest Neighbors

    **Input:** Number of neighbors $K \in \mathbb{N}$ and a new datapoint $x$ to classify

    **For** $i = 1, 2, \ldots, m$ **do**
        **Step 1.** Compute the distance $D\left(x, x^{(i)}\right)$ based on equation (10.1).
    **End do**
      **Step 2.** Determine the $K$ training datapoints that have the smallest distance to $x$
      **Step 3.** Return either the majority class amongst the $K$ nearest training datapoint
    (if classification) or the average target value among the $K$ nearest training datapoints (if
    regression)
```

Illustration of K-NN

Say we're given a dataset with a binary target variable $y^{(i)} = \{\textbf{red dots}, \textbf{purple dots}\}$ and we want to classify new datapoints by considering the nearest $k = 3$ neighbors.

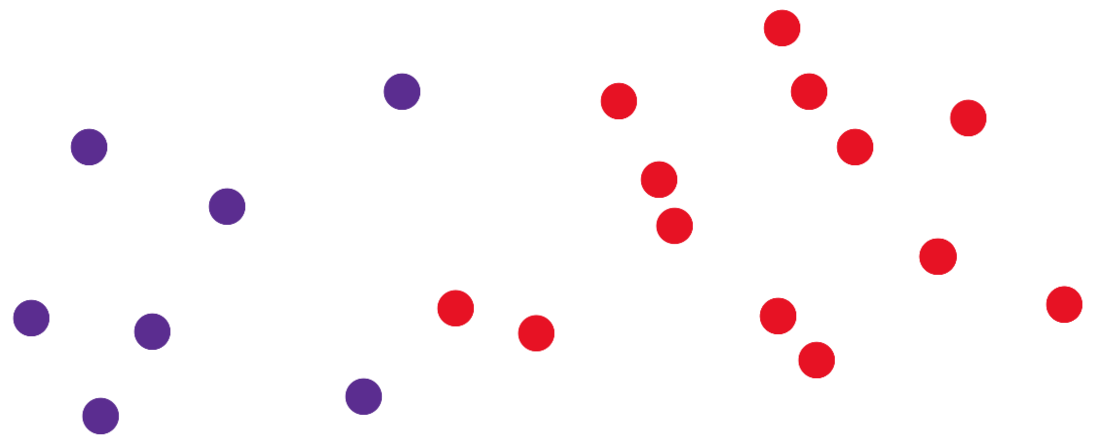


Illustration of K-NN

Say we're given a dataset with a binary target variable $y^{(i)} = \{\textbf{red dots}, \textbf{purple dots}\}$ and we want to classify new datapoints by considering the nearest $k = 3$ neighbors.

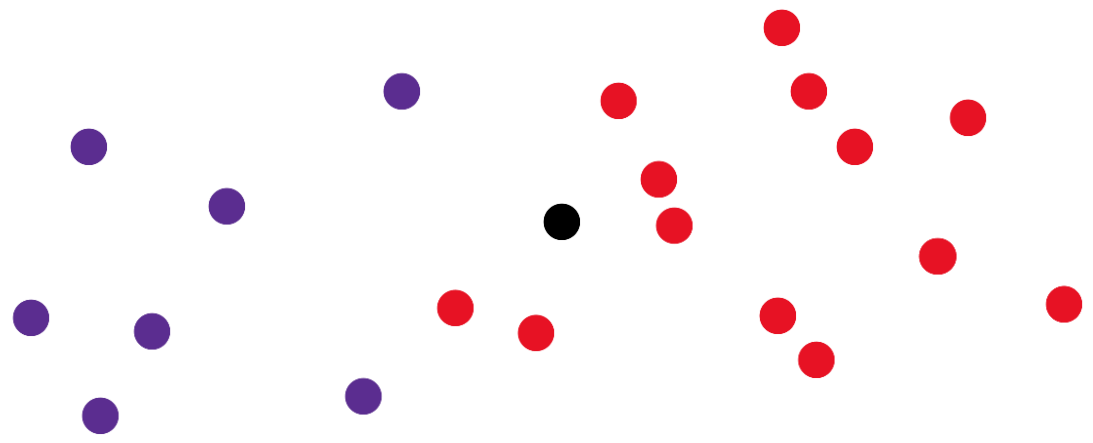


Illustration of K-NN

Say we're given a dataset with a binary target variable $y^{(i)} = \{\textbf{red dots}, \textbf{purple dots}\}$ and we want to classify new datapoints by considering the nearest $k = 3$ neighbors.

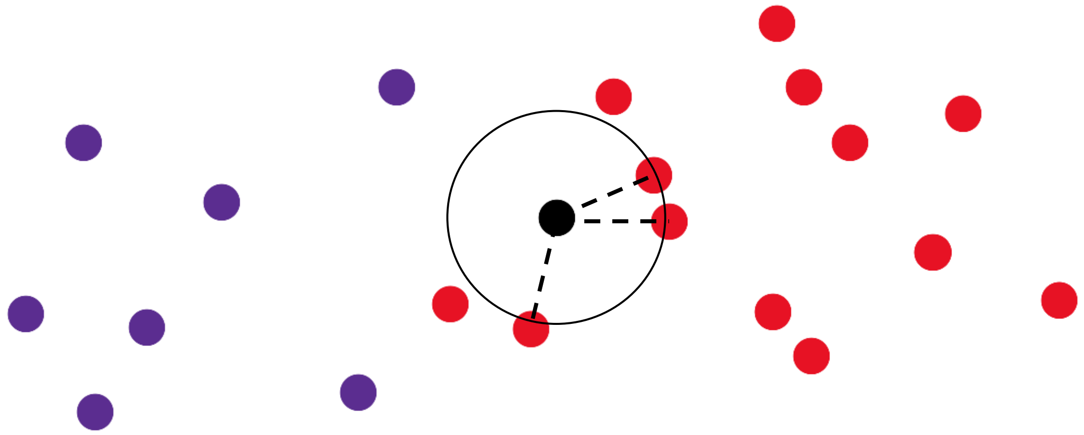


Illustration of K-NN

Say we're given a dataset with a binary target variable $y^{(i)} = \{\textbf{red dots}, \textbf{purple dots}\}$ and we want to classify new datapoints by considering the nearest $k = 3$ neighbors.

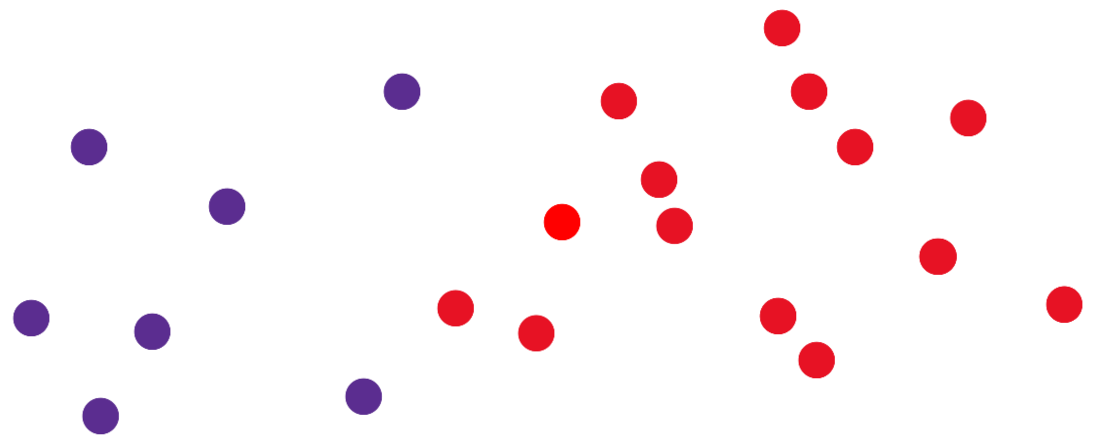


Illustration of K-NN

Say we're given a dataset with a binary target variable $y^{(i)} = \{\textbf{red dots}, \textbf{purple dots}\}$ and we want to classify new datapoints by considering the nearest $k = 3$ neighbors.

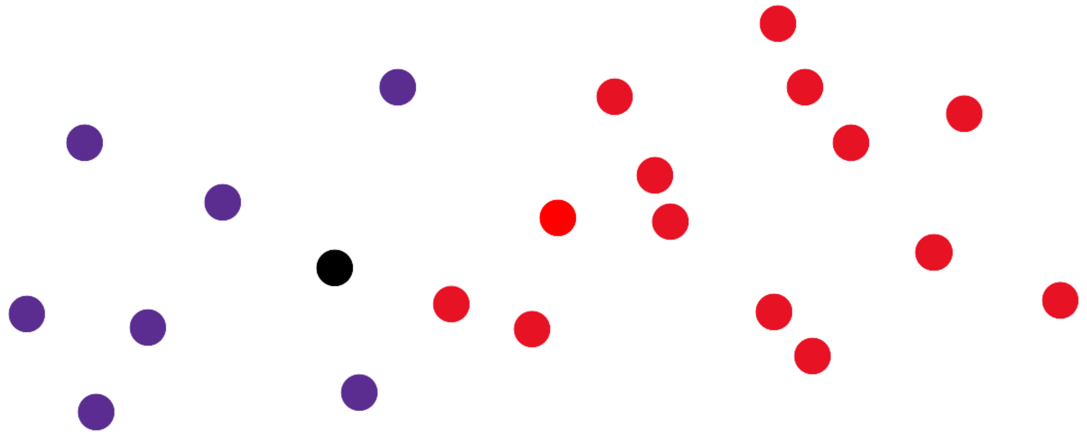


Illustration of K-NN

Say we're given a dataset with a binary target variable $y^{(i)} = \{\textbf{red dots}, \textbf{purple dots}\}$ and we want to classify new datapoints by considering the nearest $k = 3$ neighbors.

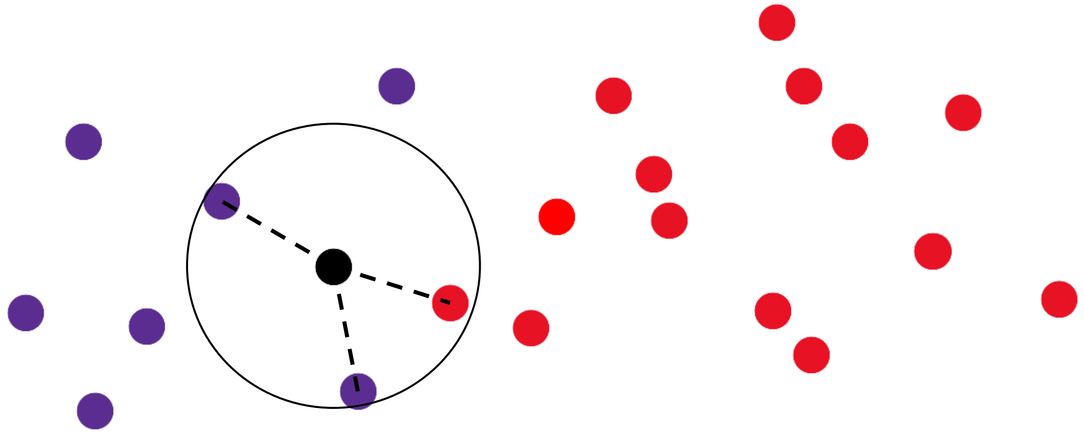
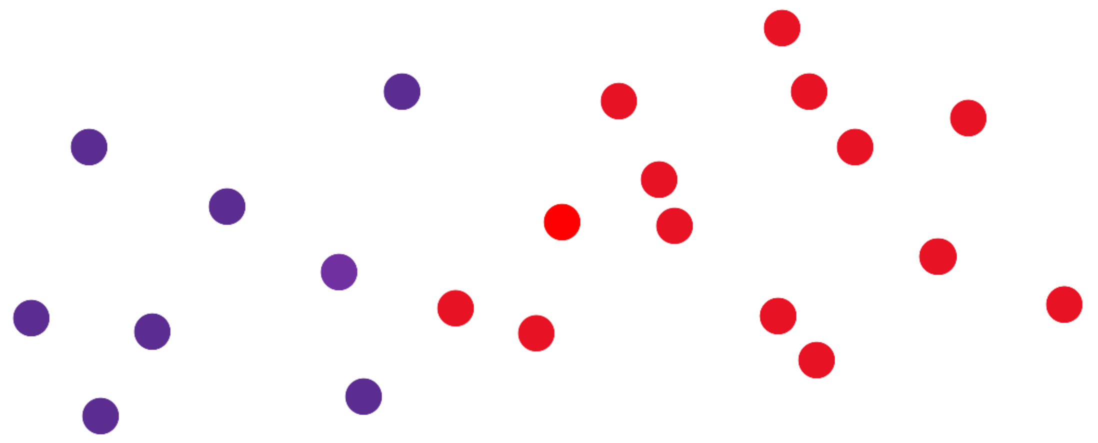


Illustration of K-NN

Say we're given a dataset with a binary target variable $y^{(i)} = \{\textbf{red dots}, \textbf{purple dots}\}$ and we want to classify new datapoints by considering the nearest $k = 3$ neighbors.

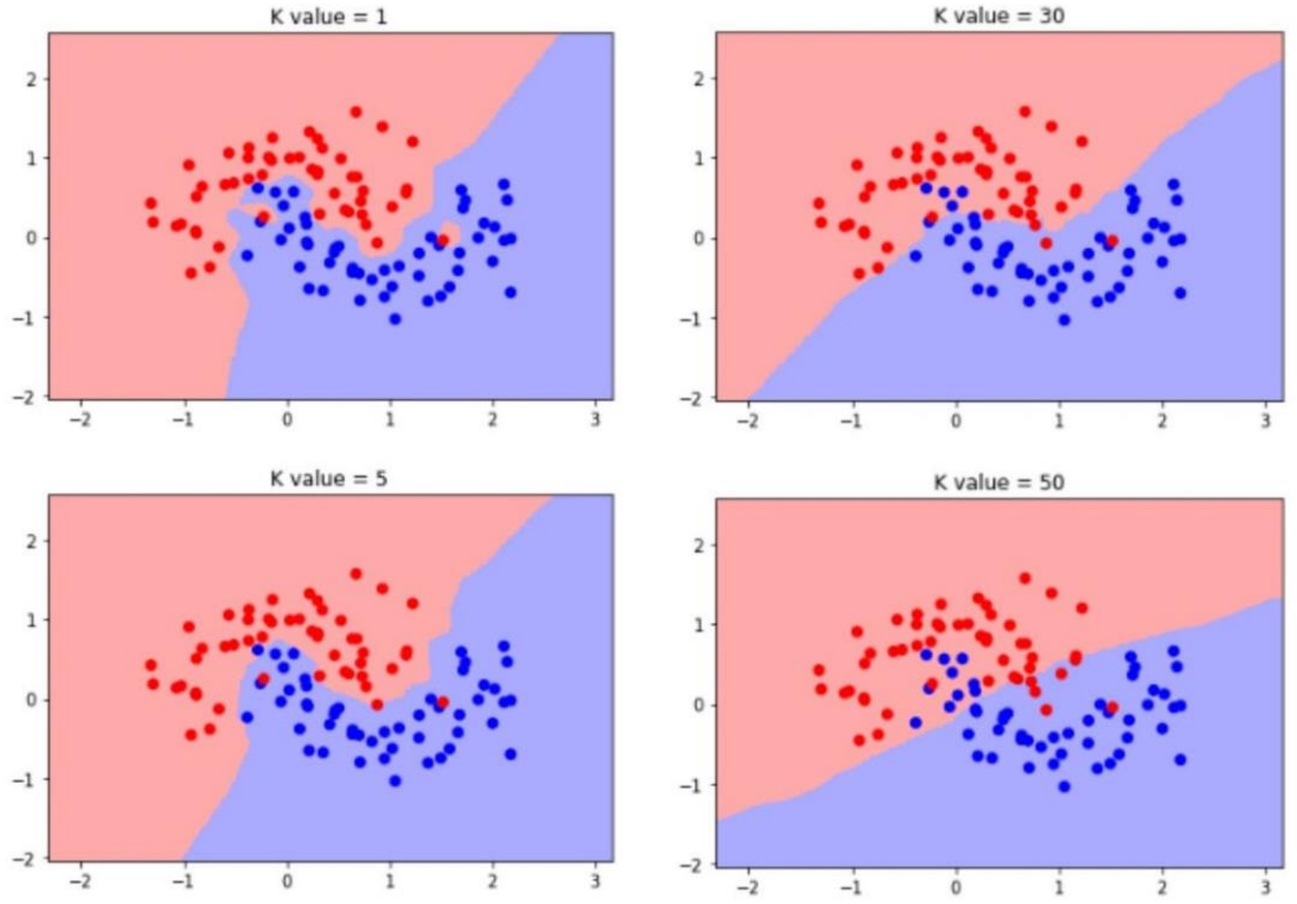


Influence of K on the Decision Boundary

- Intuitively, the complexity of a K-NN model can directly be controlled by the number of neighbors K to consider. As such, K-NN models can identify highly nonlinear relationships among datapoints.

- Specifically, for smaller choices of K, the model will yield a more complex decision boundary.

- Conversely, for larger choices of K, the model will yield a more generalized boundary.



Tradeoffs of K-NN

Pros

- Simple, intuitive, and explainable.
- Can be applied to both regression and classification problems.
- Can learn highly nonlinear decision boundaries / patterns.
- Most useful when features have a strong geographic relationship or are strongly related in a purely spatial sense.

Cons

- Cost prohibitive to classify new datapoints when using very large datasets (since the pair-wise distances must be computed for all m datapoints).
- Requires all features to be on the same scale, i.e., standardized (this is not so much a con as it is a simple fact).
- K-NN models are often not able to fully capture more complex patterns that are not purely spatial. As such, one can almost always obtain better results with more advanced models.