# Machine Learning

## AIML CZG565

## M6 : Instance Based Learning

BITS Pilani

Pilani Campus

Course Faculty of M.Tech Cluster

BITS – CSIS - WILP

# Machine Learning

innovate achieve lead

## Disclaimer and Acknowledgement


- These content of modules & context under topics are planned by the course owner Dr. Sugata, with grateful acknowledgement to many others who made their course materials freely available online
- We here by acknowledge all the contributors for their material and inputs.
- We have provided source information wherever necessary
- Students are requested to refer to the textbook w.r.t detailed content of the presentation deck shared over canvas
- We have reduced the slides from canvas and modified the content flow to suit the requirements of the course and for ease of class presentation

**Slide Source / Preparation / Review:**
From BITS Pilani WILP**:** Prof.Sugata, Prof.Chetana, , Prof.Monali, Prof.Rajavadhana, Prof.Seetha, Prof.Anita

External: CS109 and CS229 Stanford lecture notes, Prof. Tom Mitchell, Prof.. Burges, Prof. Andrew Moore, Dr.Andrew NG and many others who made their course materials freely available online

# Course Plan

| | |
|---|---|
| M1 | Introduction |
| M2 | Machine learning Workflow |
| M3 | Linear Models for Regression |
| M4 | Linear Models for Classification |
| M5 | Decision Tree |
| M6 | Instance Based Learning |
| M7 | Support Vector Machine |
| M8 | Bayesian Learning |
| M9 | Ensemble Learning |
| M10 | Unsupervised Learning |
| M11 | Machine Learning Model Evaluation/Comparison |

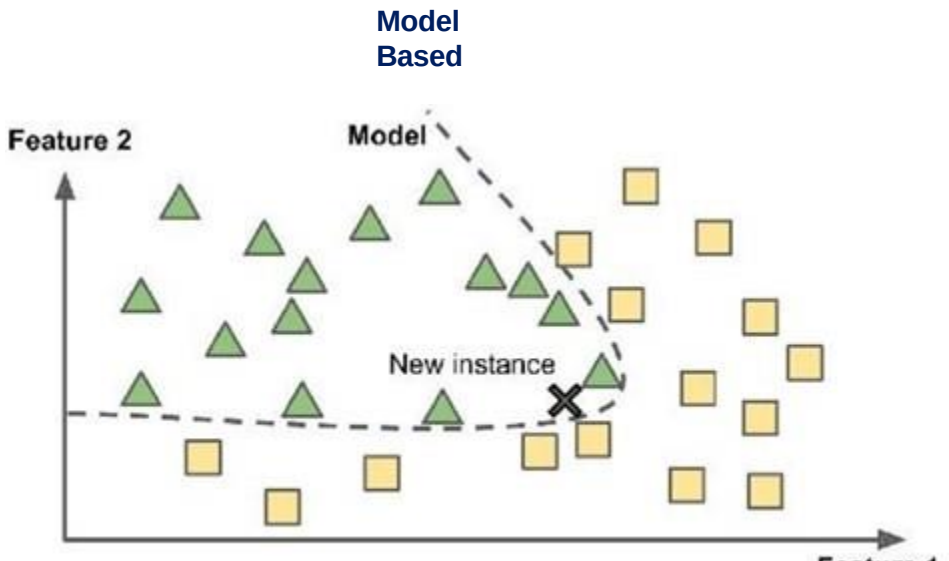
# Instance Based Learning

# Introduction to Machine Learning

## Types: Based on how & when the training data is used

- Instance Based Learning: Compare new data points to known data points
- Model Based learning : Detect patterns in the training data and build a predictive model


Lazy Learners : Less time in training but more time in predicting




Eager Learners : More time in training but Less time in predicting

# Instance Based Learning

**Model-based learning techniques**

Use the input data

$$\begin{bmatrix} x_{1,0} & x_{1,1} & \dots & x_{1,n} \\ x_{2,0} & x_{2,1} & \dots & x_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m,0} & x_{m,1} & \cdots & x_{m,n} \end{bmatrix} \text{ and } \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_m \end{bmatrix}$$

To learn a set of parameters

$$\begin{bmatrix} \theta_0 & \theta_1 & \cdots & \theta_n \end{bmatrix}$$

Which yield a **generalized** function

$$f(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

Capable of predicting values or classes on new input data

$$f(x_i) = 39$$

$$f(x_j) = 1$$

**Instance-based learning techniques**

Store the input data

$$\begin{bmatrix} x_{1,0} & x_{1,1} & \dots & x_{1,n} \\ x_{2,0} & x_{2,1} & \dots & x_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m,0} & x_{m,1} & \cdots & x_{m,n} \end{bmatrix} \text{ and } \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_m \end{bmatrix}$$

When asked to predict a new value (a query)

$$y_i = ?$$

Search for similar data points previously stored

$$\begin{bmatrix} x_{4,1} & x_{4,2} & \cdots & x_{4,n} \\ x_{9,1} & x_{9,1} & \cdots & x_{9,n} \\ x_{15,1} & x_{15,1} & \cdots & x_{15,n} \end{bmatrix} \text{ and } \begin{bmatrix} y_4 \\ y_9 \\ y_{15} \end{bmatrix}$$

And use them to generate your prediction

$$y_i = \frac{y_4 + y_9 + y_{15}}{3}$$

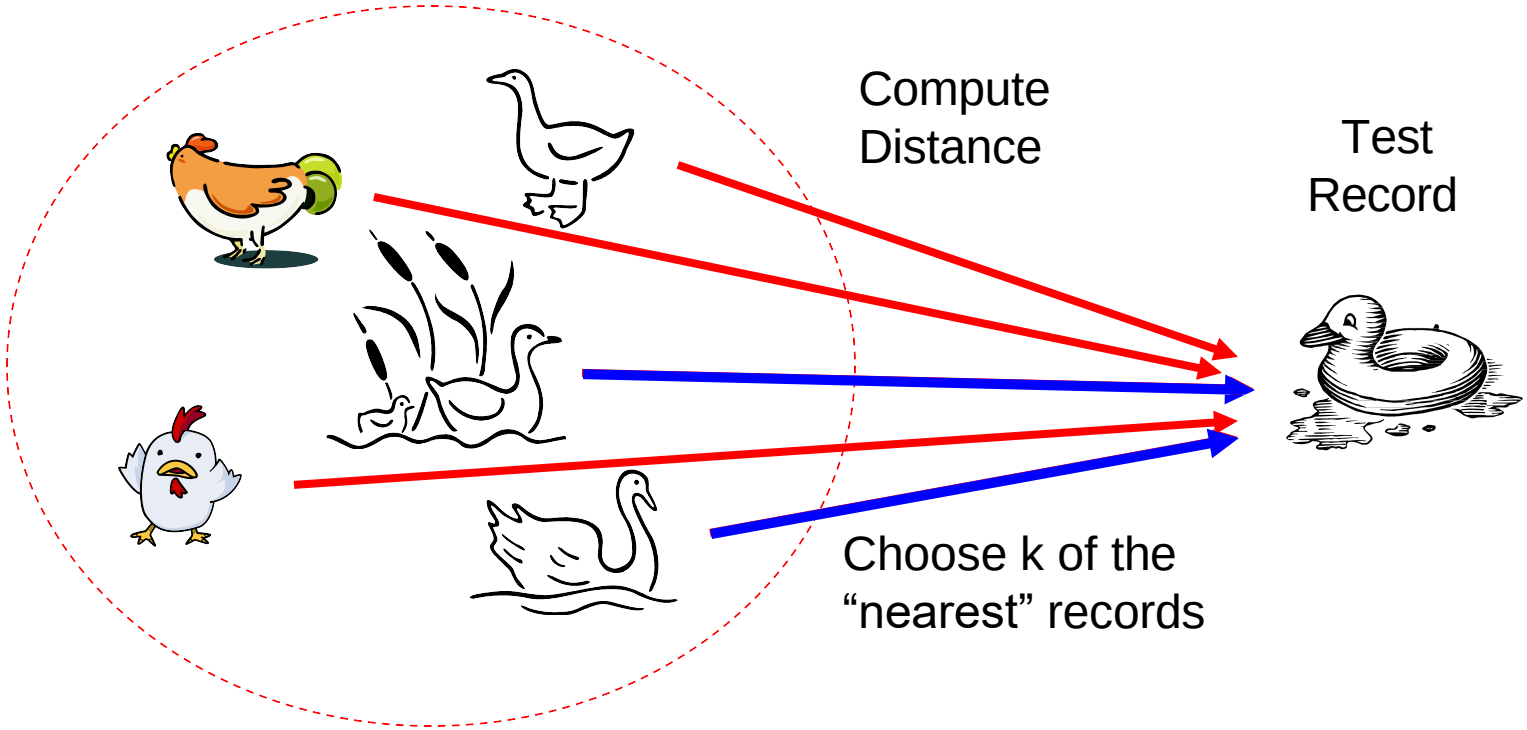
Source Credit : https://www.jeremyjordan.me/k-nearest-neighbors/

# Instance Based Learning

## Nearest Neighbor Approach

Basic idea:

- If it walks like a duck, quacks like a duck, then it's probably a duck



# k-Nearest Neighbour Classifier

- Nearest Neighbor classifier is an instance based classifier
- ‘lazy learning’, as learning is postponed until a new instance is encountered
- Constructs a local approximation to the target function, applicable in the neighborhood of new instance
- Suitable in cases where target function is complex over the entire input space, but easily describable in local approximations
- Real world applications found in recommendation systems (amazon).
- Caveat is the high cost of classification, which happens at the time of processing rather than before hand (there’s no training phase)

# Instance Based Learning

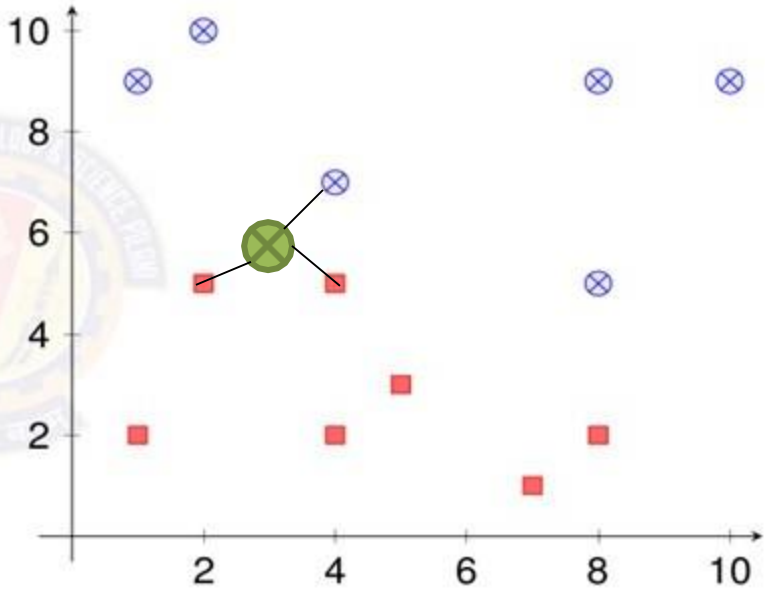
innovate achieve lead

## Nearest Neighbor Approach : Notations

K-nearest neighbor:

- Given $x_q$, take **vote** among its k nearest neighbors (if **discrete-valued target function**)
- Take **mean(or median)** of f values of k nearest neighbors (if **real-valued)** $f^*(x_q)=\Sigma_{i=1}^{k} f(x_i)/k$

| $x_1$ | $x_2$ | $y$ |
|---|---|---|
| 1 | 9 | white |
| 10 | 9 | white |
| 4 | 7 | white |
| 4 | 5 | pink |
| 5 | 3 | pink |
| 8 | 9 | white |
| 4 | 2 | pink |
| 2 | 5 | pink |
| 7 | 1 | pink |
| 2 | 10 | white |
| 8 | 5 | white |
| 1 | 2 | pink |
| 8 | 2 | pink |




Interpretation : Assume that the Instances represented as points in a Euclidean space

# k-Nearest Neighbour Classifier

**Key idea:** Just store all training examples $<x_i, f(x_i)>$

**Nearest neighbor:**

Given query instance $x_q$, first locate nearest training example $x_n$,

then estimate $f^*(x_q)=f(x_n)$

**K-nearest neighbor:**

- Given $x_q$, take vote among its k nearest neighbors (if discrete-valued target function)
- Take mean of f values of k nearest neighbors (if real-valued) $f^*(x_q)=\Sigma_{i=1}^{k} f(x_i)/k$

# Instance Based Learning

## K-Nearest Neighbor Classifier: Another Example

2 votes for cat,
1 each for Buffalo, Deer, Lion

Cat wins…

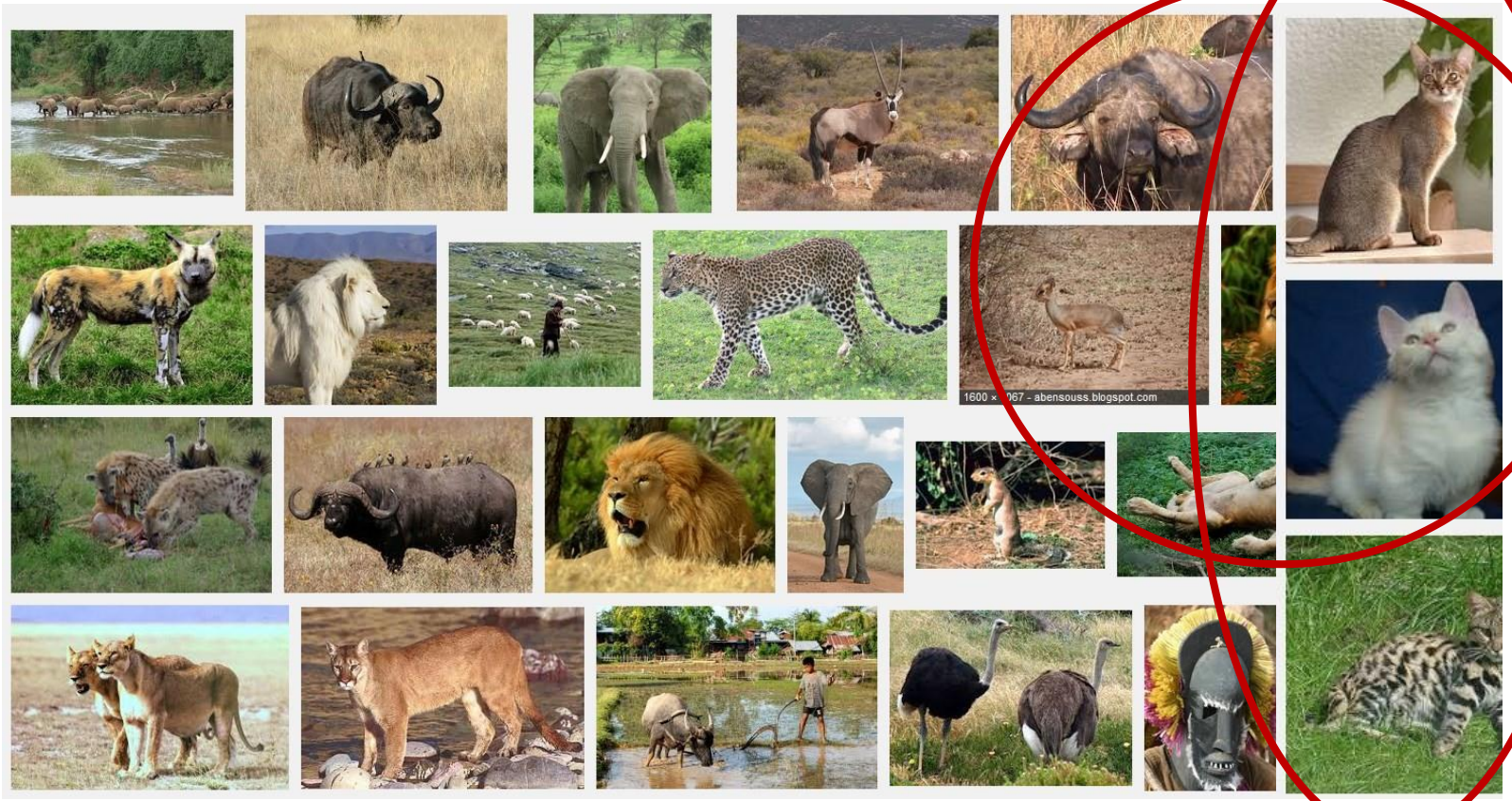
Given a query item:

k = 5 Neighborhood Region

Find k closest matches in a labeled dataset ↓

Return the most Frequent Label

k = 3 votes for “cat”



# k-Nearest Neighbour Classifier

Considers all instances as members of n-dimensional space

Nearest neighbors of an instance is determined based on Euclidean distance

Distance between two n-dimensional instances $x_i$ and $x_j$ is given by:

$$d(x_i, x_j) \equiv \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2}$$

For nearest neighbor classifier, target function can be discrete or continuous

# Instance Based Learning

innovate achieve lead

## K-Nearest Neighbor Regression: Example

| Size (feet$^2$) f1 | Number of bedrooms f2 | Number of floors f3 | Age of home (years) f4 | Price ($1000) y |
|---|---|---|---|---|
| 2104 | 5 | 1 | 45 | 460 |
| 1416 | 3 | 2 | 40 | 232 |
| 1534 | 3 | 2 | 30 | 315 |
| 852 | 2 | 1 | 36 | 178 |
| … | … | … | … | … |

Apply relevant Feature Engineering before modelling:

Eg.,
Remove Irrelevant Features
Feature Scaling
Standardization

# Instance Based Learning

innovate achieve lead

## K-Nearest Neighbor Regression: Example

| Size (feet$^2$) f1 | Number of bedrooms f2 | Number of floors f3 | Age of home (years) f4 | Price ($1000) y |
|---|---|---|---|---|
| 2104 | 5 | 1 | 45 | 460 |
| 1416 | 3 | 2 | 40 | 232 |
| 1534 | 3 | 2 | 30 | 315 |
| 852 | 2 | 1 | 36 | 178 |
| … | … | … | … | … |

Apply relevant Feature Engineering before modelling:

Eg.,
Remove Irrelevant Features
Feature Scaling
Standardization

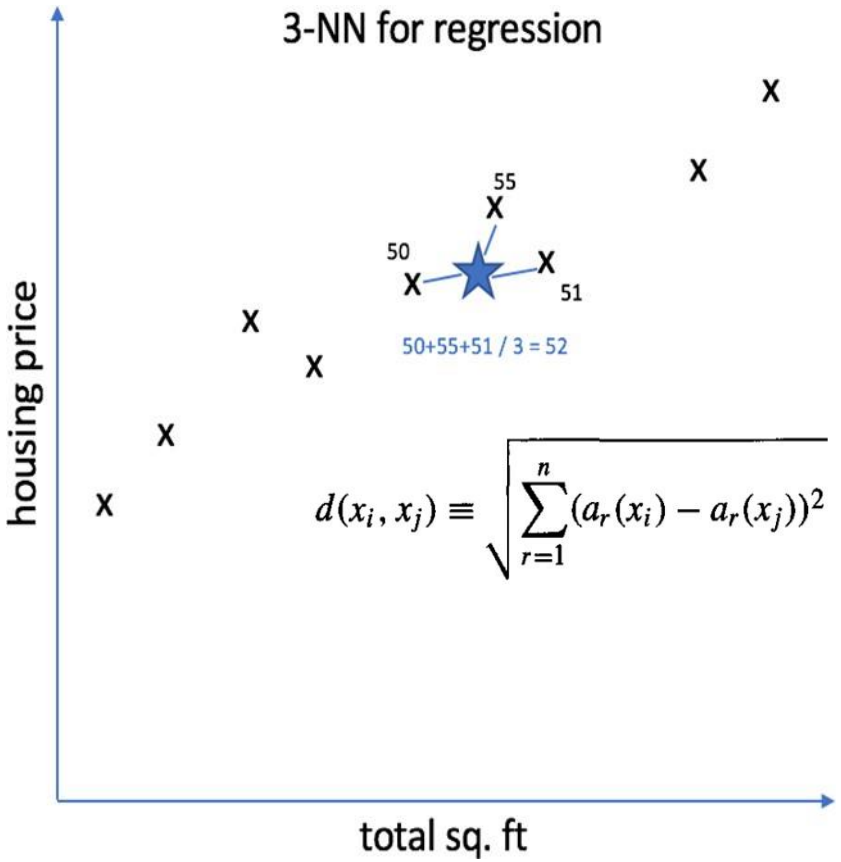
**Standardizing Numeric Data**

X: raw score to be standardized,
μ mean of the population,
σ standard deviation
The distance between the raw score and the population mean in units of the standard deviation
Transforms into negative when the raw score is below the mean, "+" when above
Where $z = \frac{x-\mu}{\sigma}$

# Instance Based Learning

## K-Nearest Neighbor Regression: Example

| Size in feet² (x) | Price ($) in 1000's (y) |
|---|---|
| 2104 | 460 |
| 1416 | 232 |
| 1534 | 315 |
| 1852 | 178 |
| … | … |
| … | … |



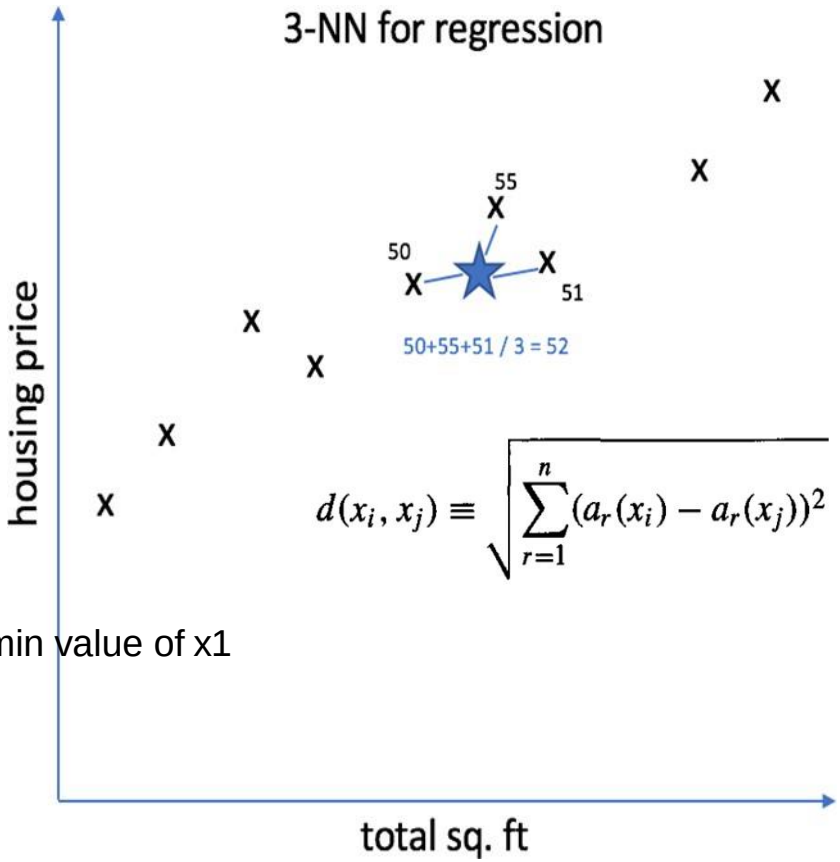

Source Credit : https://www.jeremyjordan.me/k-nearest-neighbors/

# Instance Based Learning

innovate achieve lead

## K-Nearest Neighbor Regression: Example

| Size in feet² ($x_1$) | No.of.Bed rooms ($x_2$) | Price ($) in 1000's (y) |
|---|---|---|
| 2104 | 4 | 460 |
| 1416 | 2 | 232 |
| 1534 | 3 | 315 |
| 1852 | 2 | 178 |
| … | …. | … |




Assume

min value of $x_1$

$$x_2 = \frac{\#bedrooms - 2}{5}$$

Maximum value of x1 – min value of x1

**Min-Max scaler/Normalization**

Feature scaling of features $x_i$ consists of rescaling the range of features to scale the range in [0, 1] or [−1, 1] ( Do not apply to $x_0$)

Source Credit : https://www.jeremyjordan.me/k-nearest-neighbors/

# Instance Based Learning

## K-Nearest Neighbor Regression: Example

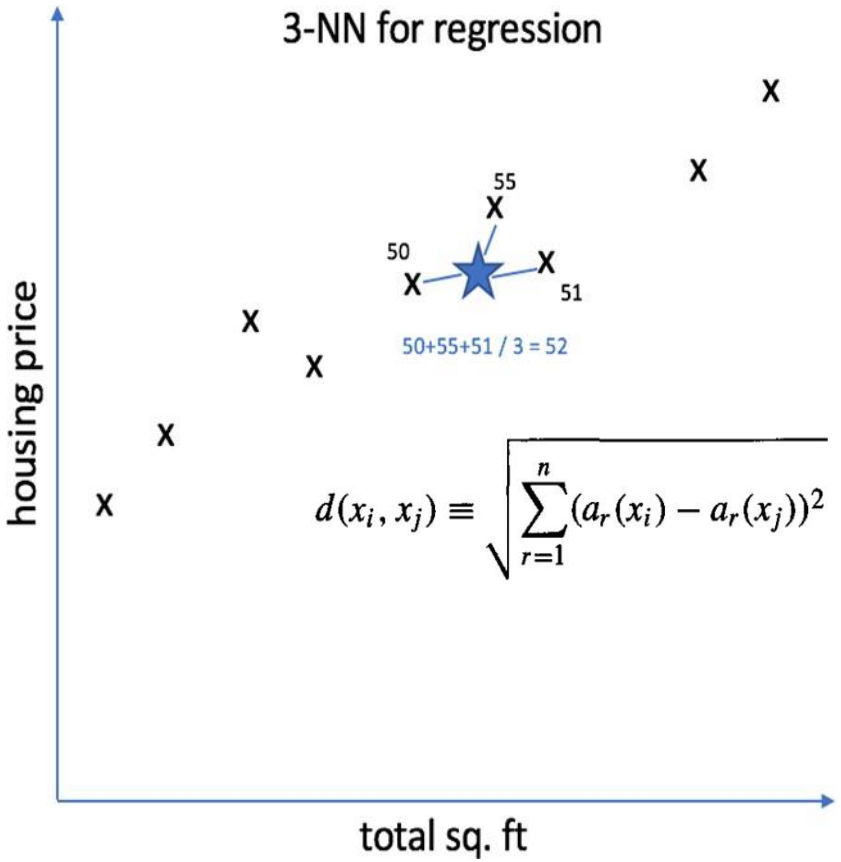
| Size in feet² ($x_1$) | No.of.Bed rooms ($x_2$) | Price ($) in 1000's (y) |
|---|---|---|
| 2104 | 4 | 460 |
| 1416 | 2 | 232 |
| 1534 | 3 | 315 |
| 1852 | 2 | 178 |
| | | … |
| | …. | … |

$$x_1 = \frac{size - 1000}{2000}$$

Average value of $x_1$

$$z = \frac{x - \mu}{\sigma}$$

The Standard Scaler assumes data is normally distributed within each feature and scales them such that the distribution centered around 0, with a standard deviation of 1




Source Credit : https://www.jeremyjordan.me/k-nearest-neighbors/

# Measures of Dis/similarity:

## Data Matrix and Dissimilarity Matrix

Data matrix

- n data points with p dimensions

$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix}$$

Dissimilarity matrix

- n data points, but registers only the distance
- A triangular matrix

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ : & : & : & & \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix}$$

# Measures of Dis/similarity:

## Nominal Attributes

Can take 2 or more states, e.g., red, yellow, blue, green (generalization of a binary attribute)

Method 1: Simple matching

- *m*: # of matches, *p*: total # of variables $d(i, j) = \frac{p - m}{p}$

Method 2: Use a large number of binary attributes

- creating a new binary attribute for each of the *M* nominal states

# Measures of Dis/similarity: Binary Attributes



- A contingency table for binary data

| Object *i* \ Object *j* | 1 | 0 | sum |
|---|---|---|---|
| 1 | $q$ | $r$ | $q+r$ |
| 0 | $s$ | $t$ | $s+t$ |
| sum | $q+s$ | $r+t$ | $p$ |

- Distance measure for symmetric binary variables:

$$d(i,j) = \frac{r+s}{q+r+s+t}$$

- Distance measure for asymmetric binary variables:

$$d(i,j) = \frac{r+s}{q+r+s}$$

- Jaccard coefficient (*similarity* measure for *asymmetric* binary variables):

$$sim_{Jaccard}(i,j) = \frac{q}{q+r+s}$$

# Measures of Dis/similarity: Binary Attributes



Example

| Name | Gender | Fever | Cough | Test-1 | Test-2 | Test-3 | Test-4 |
|---|---|---|---|---|---|---|---|
| Jack | M | Y | N | P | N | N | N |
| Mary | F | Y | N | P | N | P | N |
| Jim | M | Y | P | N | N | N | N |

- Gender is a symmetric attribute
- The remaining attributes are asymmetric binary
- Let the values Y and P be 1, and the value N is 0

$$d(jack, mary) = \frac{0+1}{2+0+1} = 0.33$$

$$d(jack, jim) = \frac{1+1}{1+1+1} = 0.67$$

$$d(jim, mary) = \frac{1+2}{1+1+2} = 0.75$$

# Measures of Dis/similarity

## Distances: Minkowski Distance

*Minkowski distance*: A popular distance measure

$$d(i, j) = \sqrt[h]{|x_{i1} - x_{j1}|^h + |x_{i2} - x_{j2}|^h + \cdots + |x_{ip} - x_{jp}|^h}$$

where $i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ and $j = (x_{j1}, x_{j2}, \ldots, x_{jp})$ are two $p$-dimensional data objects, and $h$ is the order (the distance so defined is also called L-$h$ norm)

Properties

- $d(i, j) > 0$ if $i \neq j$, and $d(i, i) = 0$ (Positive definiteness)
- $d(i, j) = d(j, i)$ (Symmetry)
- $d(i, j) \leq d(i, k) + d(k, j)$ (Triangle Inequality)

A distance that satisfies these properties is a metric

# Measures of Dis/similarity

## Distances : Special Cases of Minkowski

- $h = 1$: Manhattan (city block, $L_1$ norm) distance
  - E.g., the Hamming distance: the number of bits that are different between two binary vectors

$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \ldots + |x_{ip} - x_{jp}|$$

- $h = 2$: ($L_2$ norm) Euclidean distance

$$d(i,j) = \sqrt{(|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \ldots + |x_{ip} - x_{jp}|^2)}$$

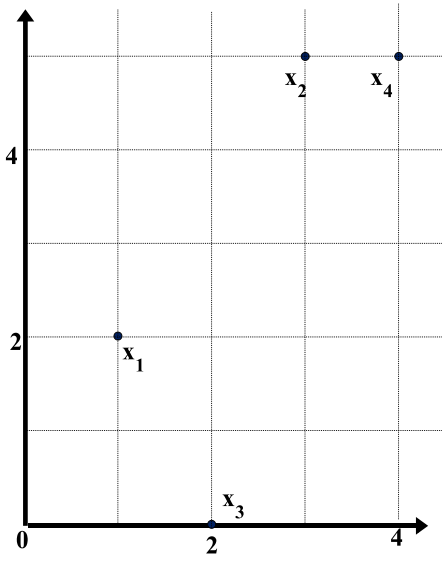
- $h \to \infty$. "supremum" ($L_{max}$ norm, $L_\infty$ norm) distance.
  - This is the maximum difference between any component (attribute) of the vectors

$$d(i,j) = \lim_{h \to \infty} \left( \sum_{f=1}^{p} |x_{if} - x_{jf}|^h \right)^{\frac{1}{h}} = \max_{f}^{p} |x_{if} - x_{jf}|$$

# Example: Minkowski Distance

(Dissimilarity Matrices)

| point | attribute 1 | attribute 2 |
|---|---|---|
| x1 | 1 | 2 |
| x2 | 3 | 5 |
| x3 | 2 | 0 |
| x4 | 4 | 5 |




**Manhattan ($L_1$)**

| L | x1 | x2 | x3 | x4 |
|---|---|---|---|---|
| x1 | 0 | | | |
| x2 | 5 | 0 | | |
| x3 | 3 | 6 | 0 | |
| x4 | 6 | 1 | 7 | 0 |

**Euclidean ($L_2$)**

| L2 | x1 | x2 | x3 | x4 |
|---|---|---|---|---|
| x1 | 0 | | | |
| x2 | 3.61 | 0 | | |
| x3 | 2.24 | 5.1 | 0 | |
| x4 | 4.24 | 1 | 5.39 | 0 |

**Supremum**

| $L_\infty$ | x1 | x2 | x3 | x4 |
|---|---|---|---|---|
| x1 | 0 | | | |
| x2 | 3 | 0 | | |
| x3 | 2 | 5 | 0 | |
| x4 | 3 | 1 | 5 | 0 |

# Standardizing Numeric Data

- Z-score:
  - X: raw score to be standardized, μ: mean of the population, σ: standard deviation
  - the distance between the raw score and the population mean in units of the standard deviation
  - negative when the raw score is below the mean, "+" when above

Where

$$z = \frac{x - \mu}{\sigma}$$

# Measures of Dis/similarity : Ordinal Variables

- An ordinal variable can be discrete or continuous
- Order is important, e.g., rank
- Can be treated like interval-scaled
  - replace $x_{if}$ by their rank $r_{if} \in \{1, \ldots, M_f\}$
  - map the range of each variable onto [0, 1] by replacing *i*-th object in the *f*-th variable by $z_{if} = \frac{r_{if} - 1}{M_f - 1}$
  - compute the dissimilarity using methods for interval-scaled variables

# Measures of Dis/similarity
# Attributes of Mixed Type: Gower distance

- A database may contain all attribute types
  - Nominal, symmetric binary, asymmetric binary, numeric, ordinal

$$d(i,j) = \frac{\sum_{c=1}^{n} \omega_c \delta_{ij}^{(c)} d_{ij}^{(c)}}{\sum_{c=1}^{n} \omega_c \delta_{ij}^{(c)}}$$

$d(i,j)$ = dissimilarity between row i and row j

$c$ = the cth column

$n$ = number of columns in the dataset

$\omega_c$ = weight of cth column $= \frac{1}{\text{nrows in dataset}}$

$$\delta_{ij}^{c} = \begin{cases} 0 & \text{if column c is missing in row i or j} \\ 0 & \text{if column c is asymmetric binary and both values in row i and j are 0} \\ 1 & \text{otherwise} \end{cases}$$

$$d_{ij}^{c}(\text{categorical}) = \begin{cases} 0 & \text{if i and j are equal in column c} \\ 1 & \text{otherwise} \end{cases}$$

$$d_{ij}^{c}(\text{continuous/ordinal}) = \frac{|\text{row i in column c - row j in column c}|}{\max(\text{column c}) - \min(\text{column c})}$$

https://healthcare.ai/clustering-non-continuous-variables/

# Example

Based on the information given in the table below, find most similar and most dissimilar persons among them. Apply min-max normalization on income to obtain [0,1] range. Consider profession and mother tongue as nominal. Consider native place as ordinal variable with ranking order of [Village, Small Town, Suburban, Metropolitan]. Give equal weight to each attribute.

| Name | Income | Profession | Mother tongue | Native Place |
|---|---|---|---|---|
| **Ram** | 70000 | Doctor | Bengali | Village |
| **Balram** | 50000 | Data Scientist | Hindi | Small Town |
| **Bharat** | 60000 | Carpenter | Hindi | Suburban |
| **Kishan** | 80000 | Doctor | Bhojpuri | Metropolitan |

# Solution

After normalizing income and quantifying native place, we get

| Name | Income | Profession | Mother tongue | Native Place |
|---|---|---|---|---|
| Ram | 0.67 | Doctor | Bengali | 1 |
| Balram | 0 | Data Scientist | Hindi | 2 |
| Bharat | 0.33 | Carpenter | Hindi | 3 |
| Kishan | 1 | Doctor | Bhojpuri | 4 |

d(Ram, Balram) = 0.67+1+1+(2-1)/(4-1)=3  d(Ram, Bharat) = 0.33+1+1+(3-1)/(4-1)=3
d(Ram, Kishan) = 0.33+0+1+(4-1)/(4-1) = 2.33  d(Balram, Bharat) = 0.33+1+0+(3-2)/(4-1)=1.67
d(Balram, Kishan) = 1+1+1+(4-2)/(4-1) = 3.67  d(Bharat, Kishan) = 0.67+1+1+(4-3)/(4-1) = 3

Most similar – Balram and Bharat;  Most dissimilar – Balram and Kishan

# Measures of Dis/similarity

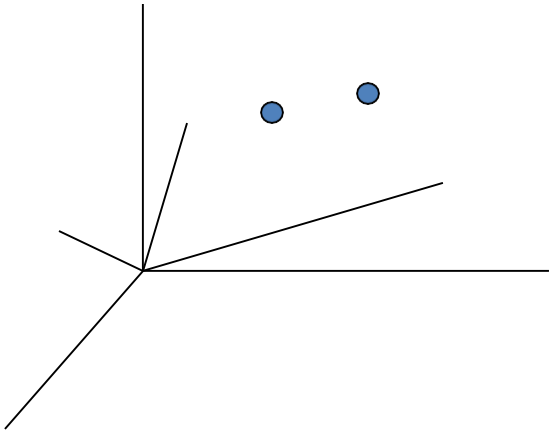
## Curse of Dimensionality

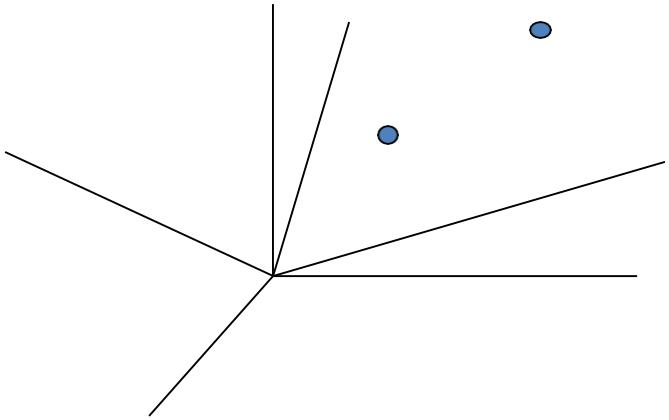
- It uses all attributes to classify instances
- If the target concept depends on only a few of the many available attributes, then the instances that are truly most "similar" may well be a large distance apart.
- ***Curse of dimensionality***: nearest neighbor is easily misled when instance space is high-dimensional

  **Solution 1:** weigh the attributes differently (use cross-validation to determine the weights)

  **Solution 2:** eliminate the least relevant attributes (again, use cross-validation to determine which attributes to eliminate)

Stretching the axes in the Euclidean space, shortening the axes that correspond to less relevant attributes, and lengthening the axes that correspond to more relevant attributes





# Measures of Dis/similarity

## Curse of Dimensionality

Imagine instances described by 20 attributes but only a few are relevant to target function

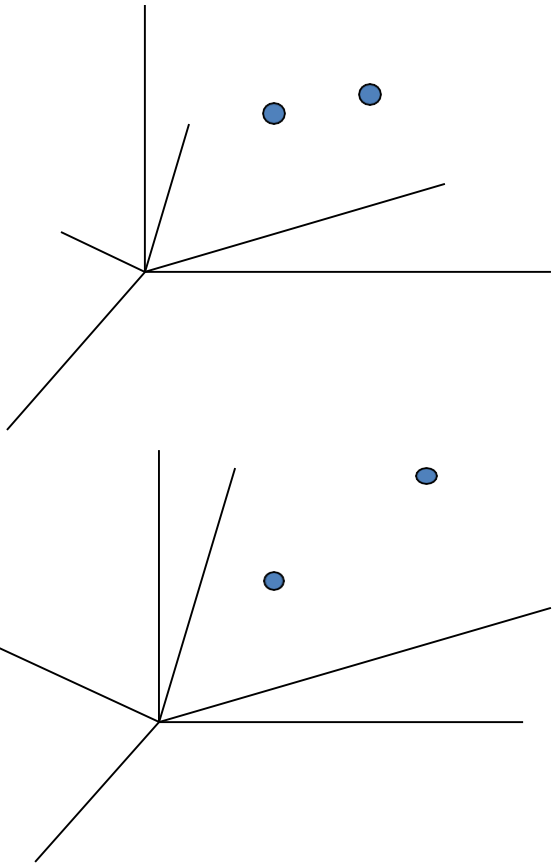
*Curse of dimensionality*: nearest neighbor is easily misled when instance space is high-dimensional

One approach:

- Stretch j-th axis by weight $z_j$, where $z_1,\ldots,z_n$ chosen to minimize prediction error
- Use cross-validation to automatically choose weights $z_1,\ldots,z_n$

Note setting $z_j$ to zero eliminates this dimension all together (feature subset selection)

Stretching the axes in the Euclidean space, shortening the axes that correspond to less relevant attributes, and lengthening the axes that correspond to more relevant attributes



# Instance Based Learning

innovate achieve lead

## K-Nearest Neighbor : Algorithm

### Approximating a discrete-valued function

$$f : \Re^n \rightarrow V.$$

Training algorithm:

- For each training example $\langle x, f(x) \rangle$, add the example to the list *training_examples*

Classification algorithm:

- Given a query instance $x_q$ to be classified,
  - Let $x_1 \ldots x_k$ denote the $k$ instances from *training_examples* that are nearest to $x_q$
  - Return

$$\hat{f}(x_q) \leftarrow \underset{v \in V}{\operatorname{argmax}} \sum_{i=1}^{k} \delta(v, f(x_i))$$

where $\delta(a, b) = 1$ if $a = b$ and where $\delta(a, b) = 0$ otherwise.

### Approximate a real-valued target function

$$f : \Re^n \rightarrow \Re$$

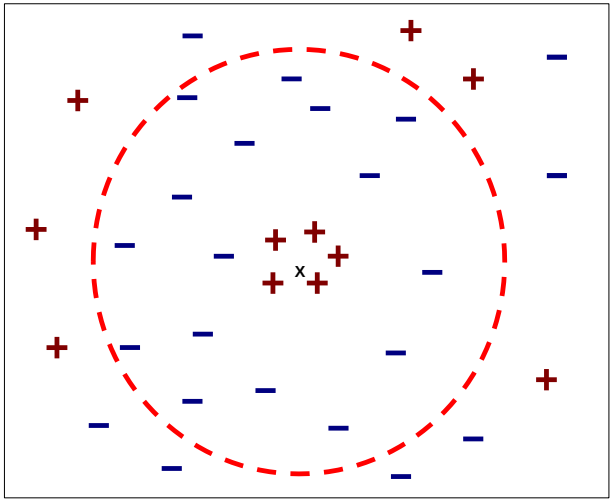
$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

# Instance Based Learning

## K-Nearest Neighbor : Hyper parameter “K”

Choosing the value of k:

**Rule of thumb:**
K = sqrt(N)
N: number of training points



# Instance Based Learning

innovate achieve lead

## K-Nearest Neighbor : Hyper parameter "K"

Choosing the value of k:

- If k is too small, sensitive to noise points
- If k is too large, neighborhood may include points from other classes

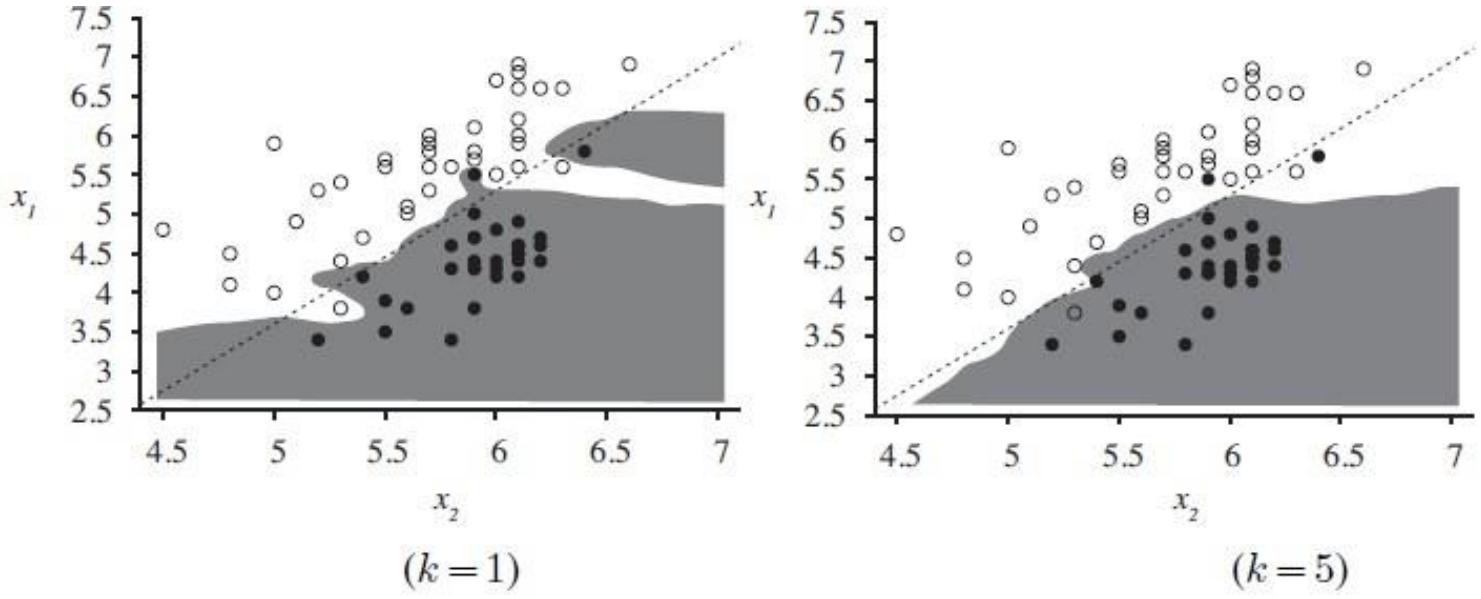


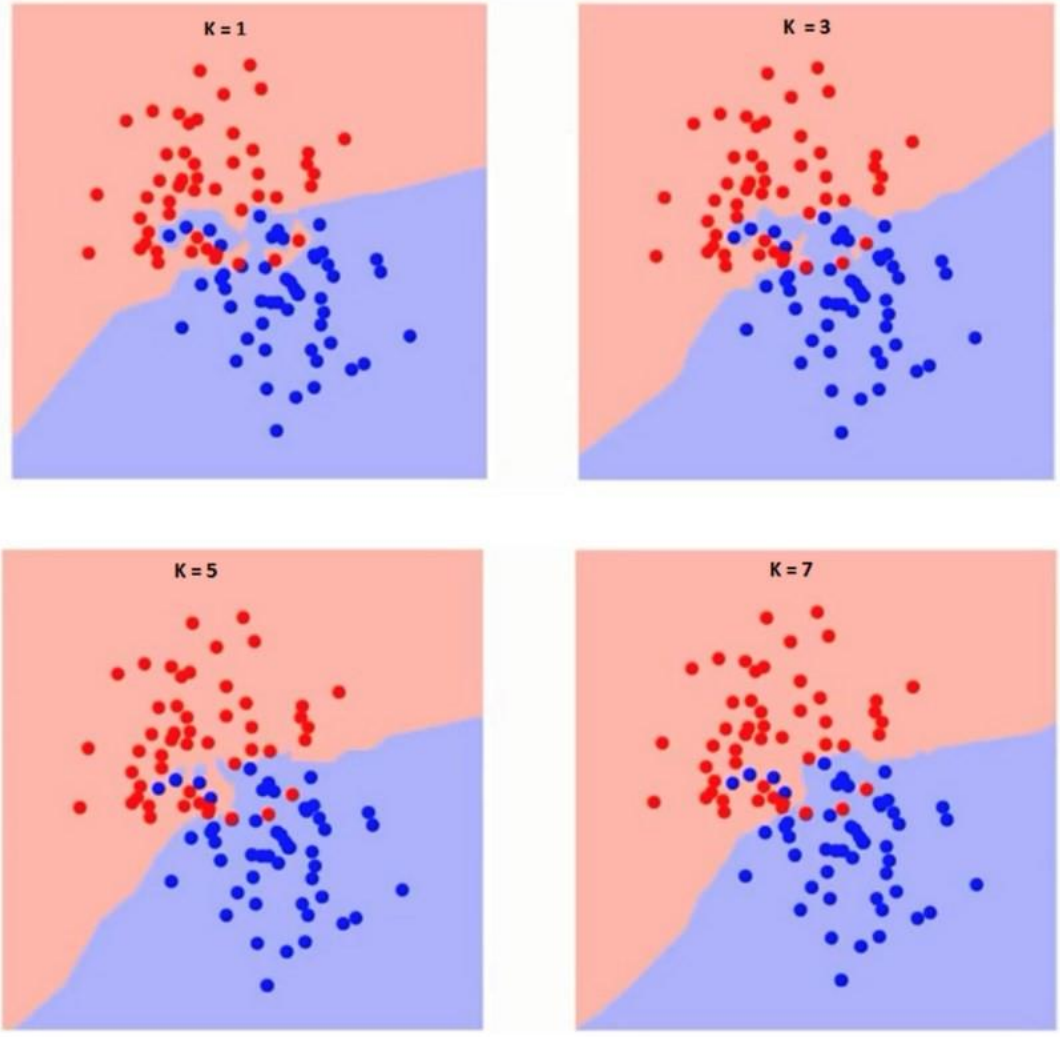

**Figure 18.26** (a) A $k$-nearest-neighbor model showing the extent of the explosion class for the data in Figure 18.15, with $k=1$. Overfitting is apparent. (b) With $k=5$, the overfitting problem goes away for this data set.

# Instance Based Learning

innovate achieve lead

## K-Nearest Neighbor : Hyper parameter “K” : Elbow Method

- Compute sum of squares error (SSE) or any other error function for varying values of K (1 to a reasonable X) and plot against K
- In the plot, the elbow (see pic) gives the value of K beyond which the error function plot almost flattens
- As K approaches the total number of instances in the set, error function drops down to ‘0’

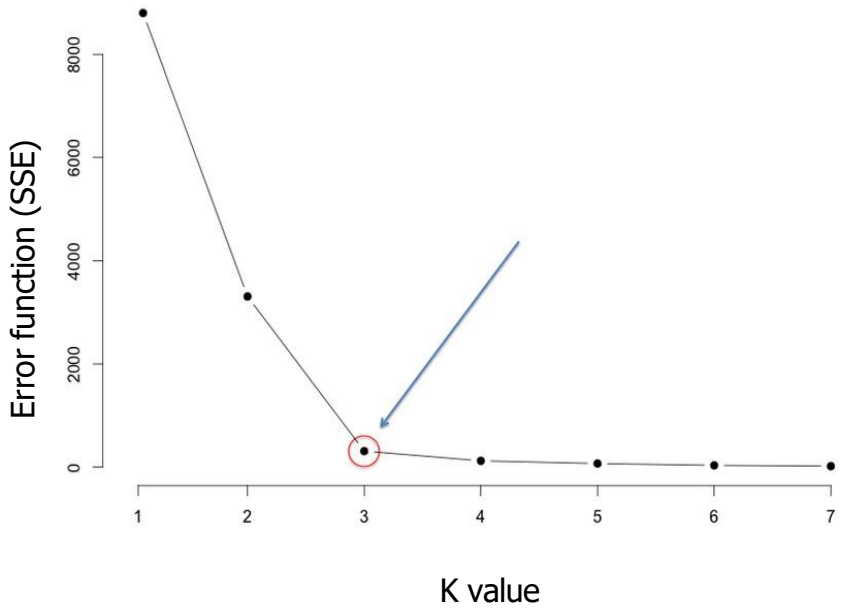


# Instance Based Learning

innovate achieve lead

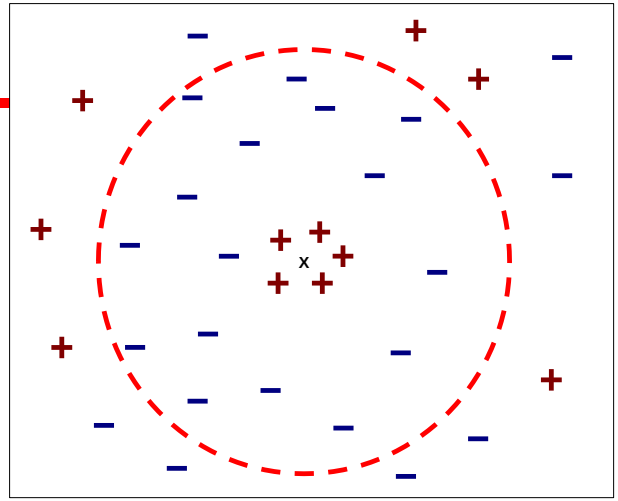
## K-Nearest Neighbor : Hyper parameter "K" : Elbow Method

- Compute sum of squares error (SSE) or any other error function for varying values of K (1 to a reasonable X) and plot against K
- In the plot, the elbow (see pic) gives the value of K beyond which the error function plot almost flattens
- As K approaches the total number of instances in the set, error function drops down to '0' , but beyond optimal K, the model becomes too generic



# Instance Based Learning

## K-Nearest Neighbor : Variation




- Locally Weighted K-NN algorithm or Distance Weighted K-NN algorithm

contribution of each of the $k$ nearest neighbors is weighted accorded to their distance to $x_q$

- **discrete-valued target functions**

$$\hat{f}(x_q) \leftarrow \underset{v \in V}{argmax} \sum_{i=1}^{k} w_i \delta(v, f(x_i))$$

where $w_i \equiv \frac{1}{d(x_q, x_i)^2}$ and $\hat{f}(x_q) = f(x_i)$ if $x_q = x_i$

- **continuous-valued target function:**

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$$

Kernel Functions

$w_i = K(d(x_q, x_i))$

$K(d(x_q, x_i)) = 1/ d(x_q, x_i)^2$

$K(d(x_q, x_i)) = 1/(d_0 + d(x_q, x_i))^2$

$K(d(x_q, x_i)) = \exp(-(d(x_q, x_i)/\sigma_0)^2)$

$d(x_q, x_i)$ is the distance between $x_q$ and $x_i$

# Problem Type – 1

Consider the following training set in 2-dimensional Euclidean space. What is the class of the point (1, 1) if 7NN classifier is considered? If the value of K is reduced whether the class will change? (Consider K=3 and K=5). What should be the final class if the above 3 values of K are considered?

| Point | Coordinate | Class | Distance from (1,1) |
|---|---|---|---|
| X1 | (-1, 1) | Negative | **2** |
| X2 | (0, 1) | Positive | **1** |
| X3 | (0, 2) | Negative | **1.414** |
| X4 | (1, -1) | Negative | **2** |
| X5 | (1, 0) | Positive | **1** |
| X6 | (1, 2) | Positive | **1** |
| X7 | (2, 2) | Negative | **1.41** |
| X8 | (2, 3) | Positive | **2.236** |

| Point | Distance | Class |
|---|---|---|
| **X2** | **1** | Positive |
| **X5** | **1** | Positive |
| **X6** | **1** | Positive |
| **X7** | **1.41** | **Negative** |
| **X3** | **1.414** | **Negative** |
| X1 | 2 | Negative |
| X4 | **2** | Negative |

# Problem Type – 2

Based on the information given in the table below, find the customer type of $x_q$=C4 using K-NN. In given example consider all the instances and use locally weighted K-NN algorithm with kernel $K(d(x_q,x_i)) = 1/ d(x_q,x_i)^2$.

*Apply min-max normalization on income to obtain [0,1] range. Consider profession and Region as nominal. Consider :Locality as ordinal variable with ranking order of [Village, Small Town, Suburban, Metropolitan]. Give equal weight to each attribute.*

| Customer | Income | Profession | Region | Locality | Category |
|---|---|---|---|---|---|
| **C0** | 60000 | Doctor | Hindi | Village | **L1** |
| **C1** | 70000 | Doctor | Bengali | Village | **L2** |
| **C2** | 60000 | Carpenter | Hindi | Suburban | **L2** |
| **C3** | 80000 | Doctor | Bhojpuri | Metropolitan | **L2** |
| **C5** | 80000 | Data Scientist | Hindi | Small Town | **L1** |
| **C4** | **50000** | **Data Scientist** | **Hindi** | **Small Town** | |

# Problem Type – 2

Based on the information given in the table below, find the customer type of $x_q$=C4 using K-NN. In given example consider all the instances and use locally weighted K-NN algorithm with kernel $K(d(x_q,x_i)) = 1/ d(x_q,x_i)^2$.

*Apply min-max normalization on income to obtain [0,1] range. Consider profession and Region as nominal. Consider :Locality as ordinal variable with ranking order of [Village, Small Town, Suburban, Metropolitan]. Give equal weight to each attribute.*

| Customer | Income | Profession | Region | Scaled Locality | Category | Distance | Weight |
|---|---|---|---|---|---|---|---|
| **C0** | 0.33 | Doctor | Hindi | 1 → 0 | **L1** | **0.97** | **1.06** |
| **C1** | 0.67 | Doctor | Bengali | 1 → 0 | **L2** | **1.75** | **0.33** |
| **C2** | 0.33 | Carpenter | Hindi | 3 → 0.67 | **L2** | **0.97** | **1.06** |
| **C3** | 1 | Doctor | Bhojpuri | 4 → 1 | **L2** | **2.2** | **0.21** |
| **C5** | 1 | Data Scientist | Hindi | 2 → 0.33 | **L1** | **1** | **1** |
| **C4** | **0** | **Data Scientist** | **Hindi** | **2 → 0.33** | | **0 (NA)** | |

d(C4, C2)

= distance w.r.t to numerical attributes + distance w.r.t Nominal attributes

= Euclidean distance on (Income, Scaled Locality ) + Categorical distance on attributes (Profession, Region)

$= \sqrt{(\text{Income}_{c4} - \text{Income}_{c2})^2 + (\text{Locality}_{c4} - \text{Locaility}_{c2})^2} + (No.\,of.\,Categorical\,features - No.\,of.\,Matches\,between\,C4\,\&\,C2)/(No.\,of.\,Categorical\,Features)$

$= \sqrt{(0 - 0.33)^2 + (0.33 - 0.67)^2} + \frac{2-1}{2}$

= 0.47 + 0.5 = 0.97

# Problem Type – 2

Based on the information given in the table below, find the customer type of $x_q$=C4 using K-NN. In given example consider all the instances and use locally weighted K-NN algorithm with kernel $K(d(x_q,x_i)) = 1/ d(x_q,x_i)^2$.

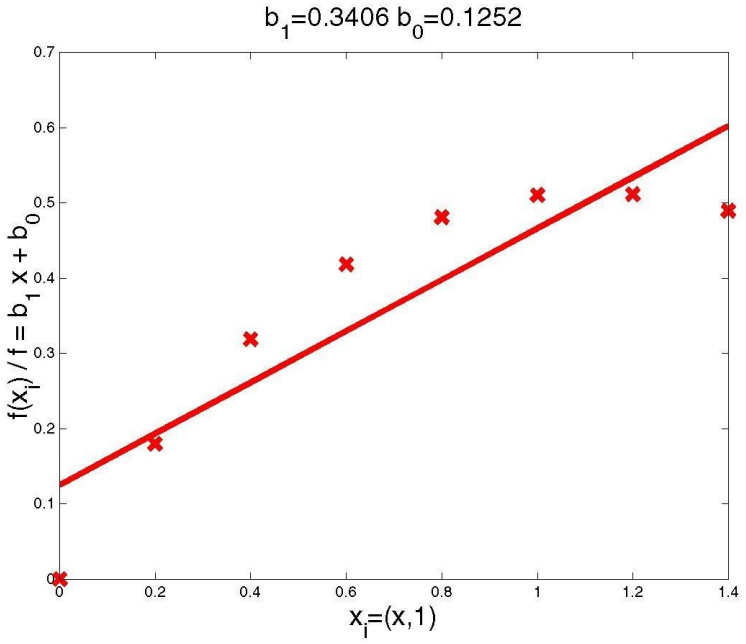
*Apply min-max normalization on income to obtain [0,1] range. Consider profession and Region as nominal. Consider :Locality as ordinal variable with ranking order of [Village, Small Town, Suburban, Metropolitan]. Give equal weight to each attribute.*

| Customer | Income | Profession | Region | Scaled Locality | Category | Distance | Weight |
|---|---|---|---|---|---|---|---|
| **C0** | 0.33 | Doctor | Hindi | 1 → 0 | **L1** | **0.97** | **1.06** |
| **C1** | 0.67 | Doctor | Bengali | 1 → 0 | **L2** | **1.75** | **0.33** |
| **C2** | 0.33 | Carpenter | Hindi | 3 → 0.67 | **L2** | **0.97** | **1.06** |
| **C3** | 1 | Doctor | Bhojpuri | 4 → 1 | **L2** | **2.2** | **0.21** |
| **C5** | 1 | Data Scientist | Hindi | 2 → 0.33 | **L1** | **1** | **1** |
| **C4** | **0** | **Data Scientist** | **Hindi** | **2 → 0.33** | | **0 (NA)** | |

| Point | Kernel / Weight | Class |
|---|---|---|
| C5 | 1 | L1 |
| C0 | 1.06 | L1 |
| C2 | 1.06 | L2 |
| C1 | 0.33 | L2 |
| C3 | **0.21** | L2 |

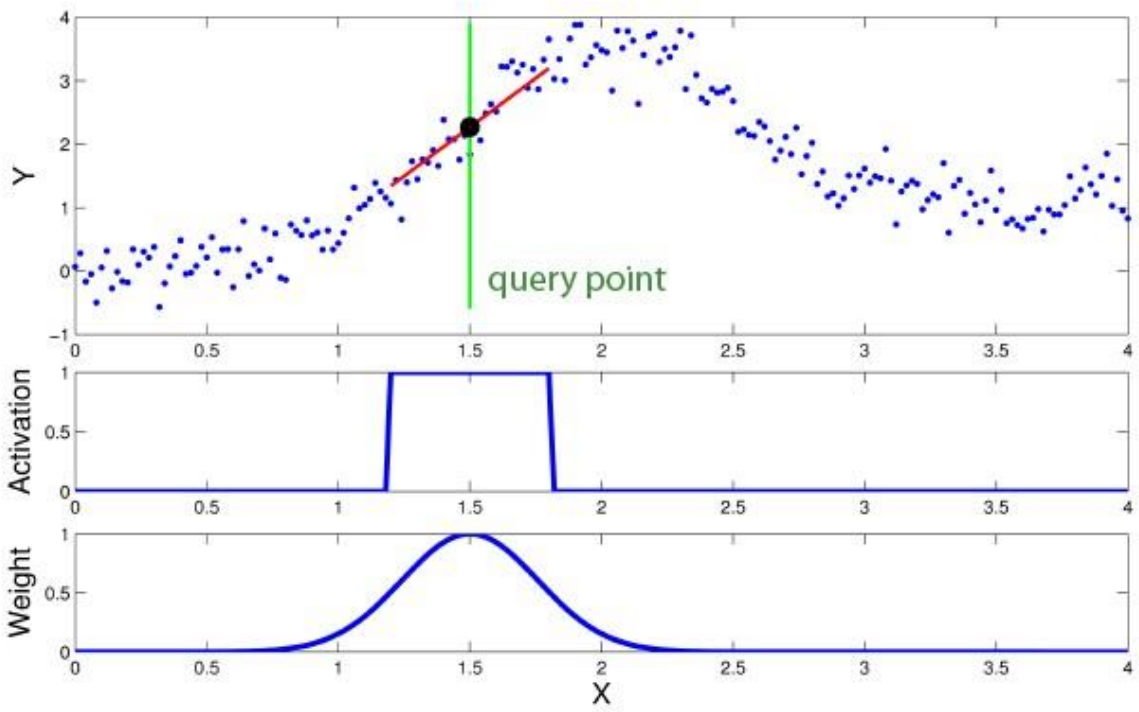
**Inference** : Without weighted KNN, L2 is the majority voted class.
But with weighted distances, L1 class instances are more similar to C4

# Locally Weighted Regression



# Locally Weighted Regression

- Locally – Function approximated based on data near query point
- Weighted – Contribution by each training example is weighted by its distance from query point
- Regression- Approximates real-valued target function
- ***Residual*** is the error in approximating the target function. $\hat{f}(x) - f(x)$
- ***Kernel function*** is the function of distance that is used to determine the weight of each training example. In other words, the kernel function is the function ***K*** such that $w_i = K(d(x_i, x_q))$




Source Credit : https://www.ias.informatik.tu-darmstadt.de/uploads/Teaching/AutonomousLearningSystems/Englert_ALS_2012.pdf

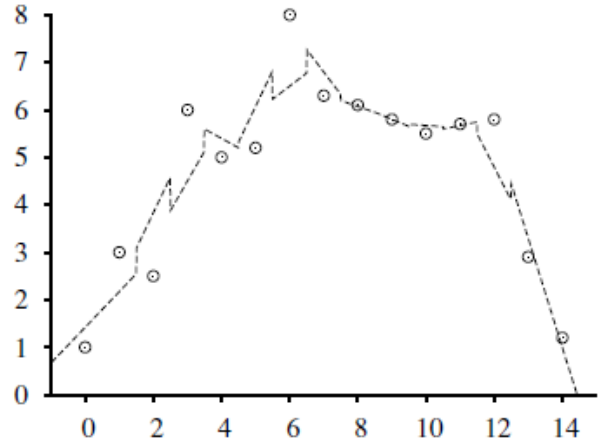
# Locally Weighted Regression

- The nearest-neighbor approaches can be thought of as approximating the target function f ***(x)*** at the single query point $\boldsymbol{x} = \boldsymbol{x_q}$

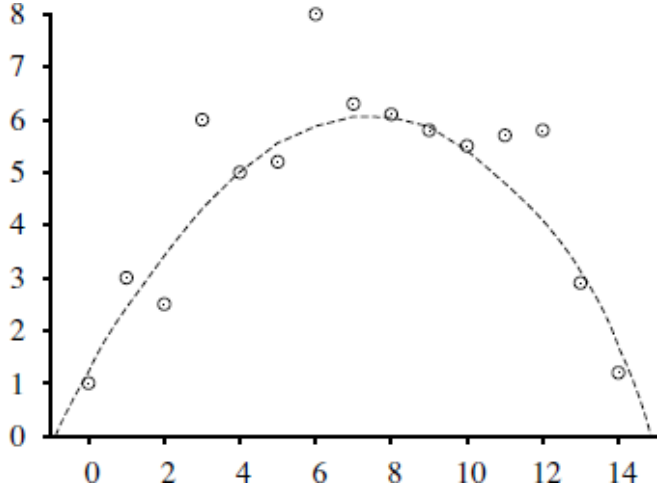
- Locally weighted regression is a generalization of this approach. It constructs an explicit approximation to f over a local region surrounding $\boldsymbol{x_q}$

- Approximate the target function in the neighborhood surrounding x, using a linear function, a quadratic function, a multilayer neural network, or some other functional form.

3-nearest-neighbors linear regression




Locally weighted regression with a quadratic kernel



# Locally weighted linear regression

- target function is approximated using a **linear function**

$$\hat{f}(x) = w_0 + w_1 a_1(x) + ... + w_n a_n(x)$$

- methods like **gradient descent** can be used to calculate the coefficients $w_0, w_1, ..., w_n$ to minimize the error in fitting such linear functions

- ANNs require a global approximation to the target function
- here, just a local approximation is needed

$\Rightarrow$ the error function has to be redefined

# Locally weighted linear regression

- possibilities to redefine the error criterion $E$
  1. Minimize the squared error over just the $k$ nearest neighbors

$$E_1(x_q) \equiv \frac{1}{2} \sum_{x \in k \text{ nearest neigbors}} (f(x) - \hat{f}(x))^2$$

  2. Minimize the squared error over the entire set $D$, while weighting the error of each training example by some decreasing function $K$ of its distance from $x_q$

$$E_2(x_q) \equiv \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2 \cdot K(d(x_q, x))$$

  3. Combine 1 and 2

$$E_3(x_q) \equiv \frac{1}{2} \sum_{x \in k \text{ nearest neighbors}} (f(x) - \hat{f}(x))^2 \cdot K(d(x_q, x))$$

# Locally weighted linear regression

- For a given query point $\mathbf{x}_q$ we solve the following weighted regression problem using gradient descent training rule:

$$\Delta w_j = \eta \sum_{x \in \ k \ nearest \ nbrs \ of \ x_q} K(d(x_q, x)) \ (f(x) - \hat{f}(x)) \ a_j(x)$$

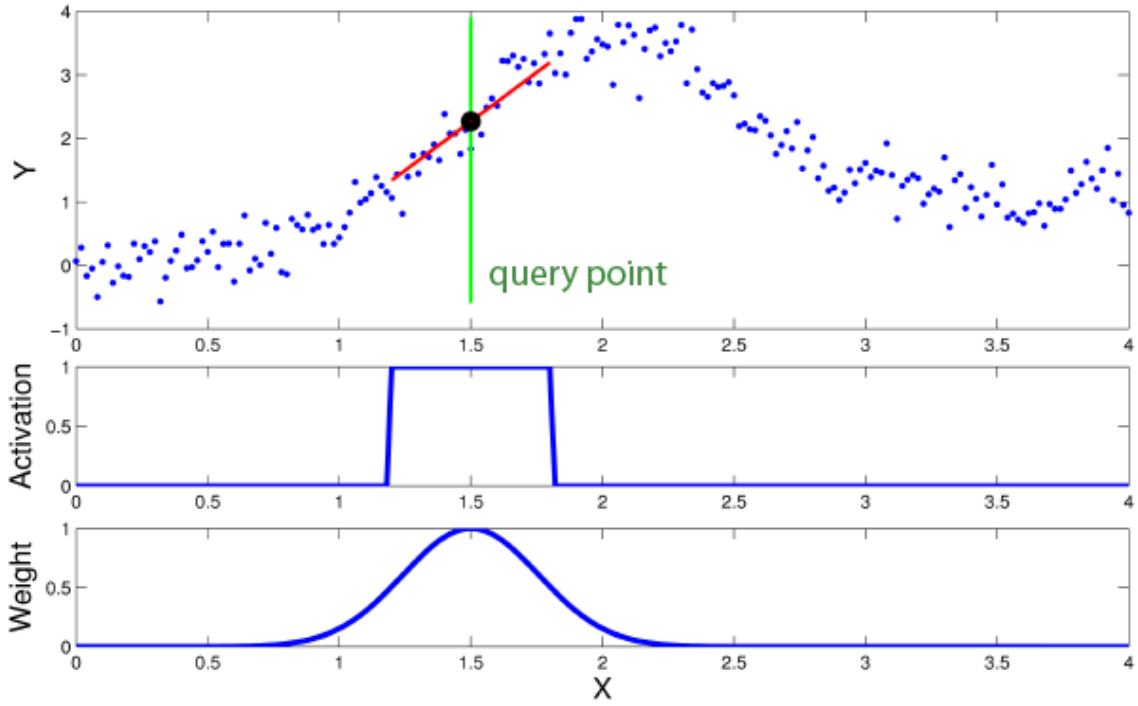
- Note that we need to solve a new regression problem for *every* query point—that's what it means to be *local*.
- In ordinary linear regression, we solved the regression problem once, globally, and then used the same $h_w$ for any query point.

$$w_i \leftarrow w_i + \Delta w_i$$

Note: + sign in weight update rule as in $\Delta w_i$ (actual-prediction ) is used.

If we use (prediction – actual) then instead of '+', '-' sign will be used.

# Example



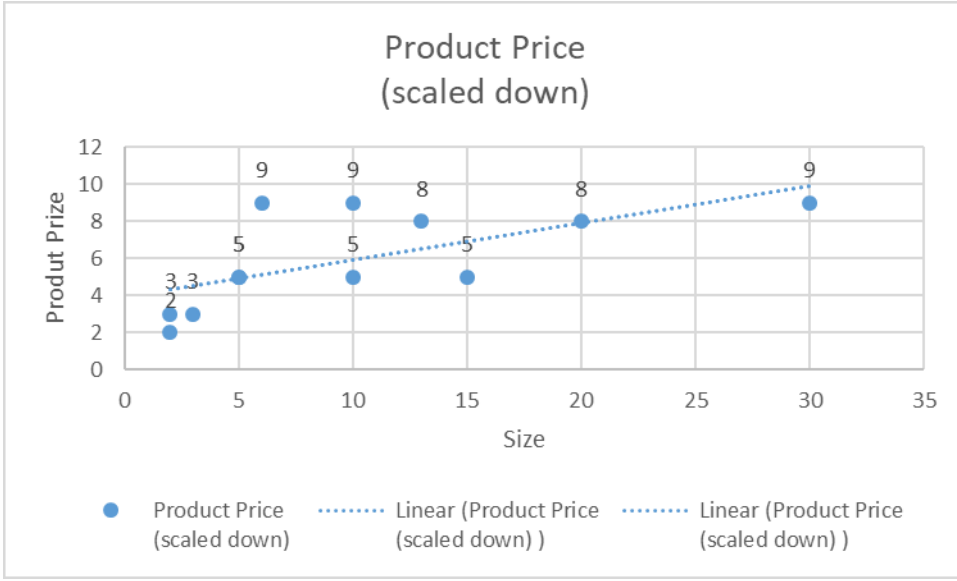

- The upper graphic the set of data points (x,y) (blue dots), query point (green line), local linear model (red line) and prediction (black dot).
- The graphic in the middle shows the activation area of the model.
- The corresponding weighting kernel (receptive field) is shown in the bottom graphic.

Source Credit : https://www.ias.informatik.tu-darmstadt.de/uploads/Teaching/AutonomousLearningSystems/Englert_ALS_2012.pdf

# Problem Type - 3

Consider the following training set in 2-dimensional Euclidean space. What is the prize of the product with size 7 if 5-NN is considered? Use the kernel of the form $K(d(x_q,x_i)) = \exp( - d(x_q,x_i)^2 / 2b^2)$ where b=1. Use the linear regression of the form $y = 3.5 + 0.8\ x$ and the learning rate of 0.2 to apply the gradient descent and update the weights at the end of first iteration

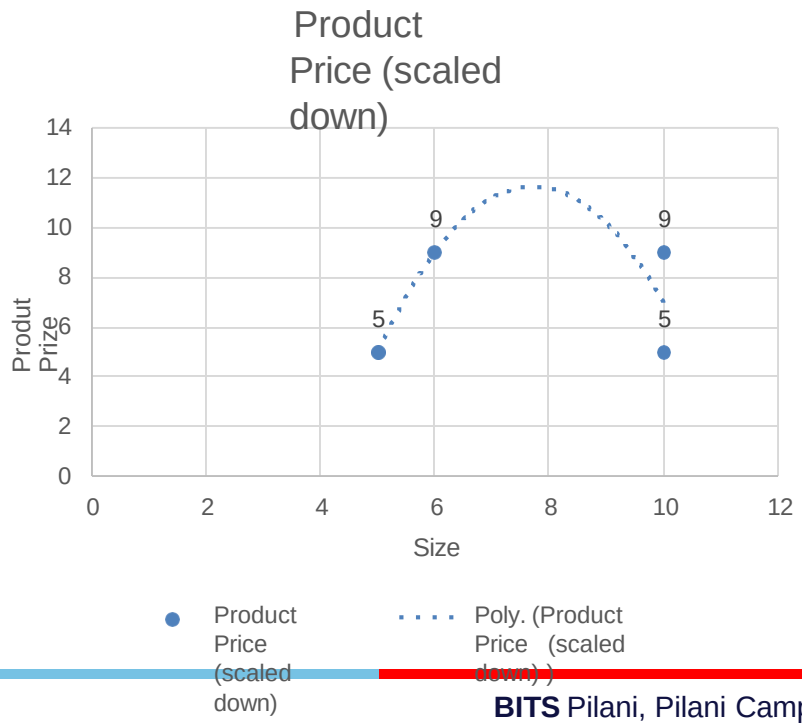
| Size | Product Price (scaled down) |
|---|---|
| 10 | 5 |
| 5 | 5 |
| 2 | 2 |
| 3 | 3 |
| 5 | 5 |
| 15 | 5 |
| 30 | 9 |
| 20 | 8 |
| 6 | 9 |
| 2 | 3 |
| 10 | 9 |
| 13 | 8 |



# Problem Type - 3

Consider the following training set in 2-dimensional Euclidean space. What is the prize of the product with size 7 if 5-NN is considered? Use the kernel of the form $K(d(x_q,x_i)) = \exp( - d(x_q,x_i)^2 / 2b^2)$ where b=1. Use the linear regression of the form $y = 3.5 + 0.8\ x$ and the learning rate of 0.2 to apply the gradient descent and update the weights at the end of first iteration

| Size | Product Price (scaled down) |
|---|---|
| 10 | 5 |
| 5 | 5 |
| 2 | 2 |
| 3 | 3 |
| 5 | 5 |
| 15 | 5 |
| 30 | 9 |
| 20 | 8 |
| 6 | 9 |
| 2 | 3 |
| 10 | 9 |
| 13 | 8 |



# Problem Type - 3

Use either of below formula :

$$W_{j-new} = W_{j-old} + \alpha * \sum_{i=1}^{K\ Neighbors} K(d(xq,xi))( f(x_i) - f\hat{}(x_i) )a_j(x_i)$$

$$W_{j-new} = W_{j-old} - \alpha * \sum_{i=1}^{K\ Neighbors} K(d(xq,xi))(f\hat{}(x_i) - f(x_i))a_j(x_i)$$

| Size | Product Price (scaled down) | d(xq, xi) | $K(d(x_q,x_i))$ |
|---|---|---|---|
| **10** | 5 | 3 | 0.01 |
| **5** | 5 | 2 | 0.14 |
| **2** | 2 | 5 | 0 |
| **3** | 3 | 4 | 0 |
| **5** | 5 | 2 | 0.14 |
| **15** | 5 | 8 | 0 |
| **30** | 9 | 23 | 0 |
| **20** | 8 | 13 | 0 |
| **6** | 9 | 1 | 0.61 |
| **2** | 3 | 5 | 0 |
| **10** | 9 | 3 | 0.01 |
| **13** | **8** | 6 | 0 |

Apply gradient descent where the gradient is weighed by the kernel output.

Answer :

Updated weights after 1st iteration

w0 = 3.43, w1= 0.43

| Price Prediction | |
|---|---|
| 6.459515 | After 1st iteration of GD |
| 9.1 | Before GD |

# When to Consider Nearest Neighbors

- Suitable for Low dimensional datasets
- Instances map to points in RN
- Less than 20 attributes per instance
- Lots of training data
- Local information can yield highly adaptive behavior
- Lots of training data (distance-weighted KNN)
  - Training is very fast
- Learn complex target functions
  - Do not lose information
- Noisy training data (distance-weighted KNN)
  - by taking the weighted average of the k neighbors nearest to the query point, it can smooth out the impact of isolated noisy training examples.

# Nearest-Neighbor Classifiers: Challenges

- The value of *k*, the number of nearest neighbors to retrieve
- Choice of Distance Metric to compute distance between records
- Expensive, Slow at query time
  - To determine the nearest neighbour of a query point *q*, must compute the distance to all *N* training examples
    - + Pre-sort training examples into fast data structures (kd-trees)
    - + Remove redundant data (condensing)
- Storage Requirements
  - Must store all training data **P**
    - + Remove redundant data (condensing)

# Additional Exercise for Students

| Size | Rating (1-10) | Product Price (scaled down) |
|---|---|---|
| **10** | 5 | **5** |
| **5** | 2 | **5** |
| **2** | 7 | **2** |
| **3** | 6 | **3** |
| **5** | 4 | **5** |
| **15** | 8 | **5** |
| **30** | 4 | **9** |
| **20** | 8 | **8** |
| **6** | 10 | **9** |
| **2** | 2 | **3** |
| **10** | 4 | **9** |
| **13** | 6 | **8** |

1. Consider the following training set in 2-dimensional Euclidean space. What is the prize of the product with size 12 if 7-NN is considered? Use the kernel of the form $K(d(x_q,x_i)) = 1/ d(x_q,x_i)^2$. Use the linear regression of the form $y = 2 + 1.5\ x$ and the learning rate of 0.6 to apply the gradient descent and update the weights at the end of first iteration. Use Manhattan Distance for the distance calculation.
2. Try to scale all the features with min-max of [0-5] and redo the modelling under previous question. Observe to Interpret the influence of the scaling
3. Find the MSE of the K-NNs from the results of both above parts separately & state your observations
4. If the Product with features (13,6) and (30,4) are outliers then interpret the effect of the same in the K-NN modelling, before and after removing them with numerical justication.

# Lab : Practise :

## Take any BITS virtual Lab dataset and experiment on below cases

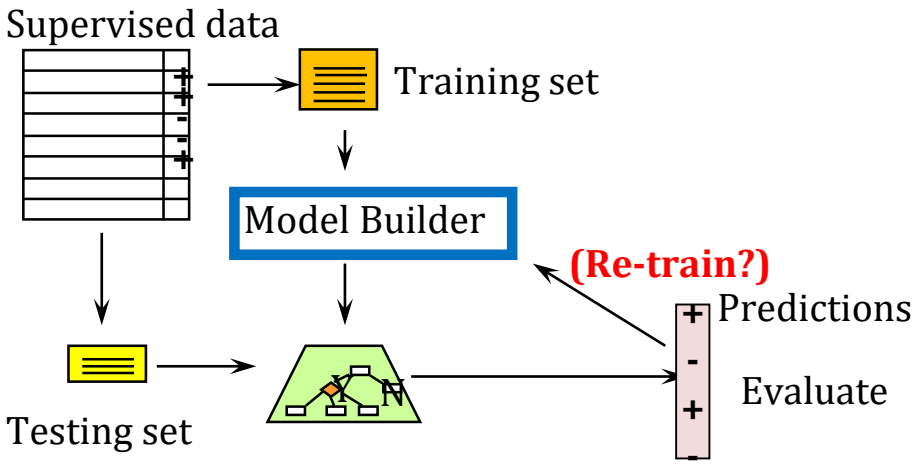
- **Tune KNN**. Try different values of *k* to see if you can improve the performance of the algorithm.
- **Use weighted KNN** and check the performance
- **Regression**. Apply it to a regression predictive modelling problem (e.g. predict a numerical value)
- **More Distance Measures**. Implement other distance measures such as Hamming distance, Manhattan distance and Minkowski distance.
- **Data Preparation**. Distance measures are strongly affected by the scale of the input data. Experiment with and without standardization and other data preparation methods in order to improve results.

# Evaluating Classifier Performance

# Model selection and assessment

- Model selection: estimating the performance of different models in order to choose the best one.
- Model assessment: having chosen a final model, estimating its prediction error (generalization error) on new data.
- To estimate generalization error (measure the quality of inductive bias), we need data unseen during training. We split the data as
  - Training set : used to fit the models
  - Validation set : used to estimate prediction error for model selection, tune hyperparameters, feature selection
  - Test (publication) set : is used for assessment of the generalization error of the final chosen model

# Hyper parameter tuning using test set




Problem 1: Hesitating between two models (e.g linear model and a polynomial model) ?
Solution 1: train both and compare how well they generalize using the test set: linear model generalizes better but looks like overfitting So, apply some regularization
Problem 2: how to choose the value of regularization hyper parameter?
Solution 2: train 100 different models using 100 different values for this hyper parameter. Choose best hyper parameter value with the lowest generalization error (5%). Great!!!: launch into production

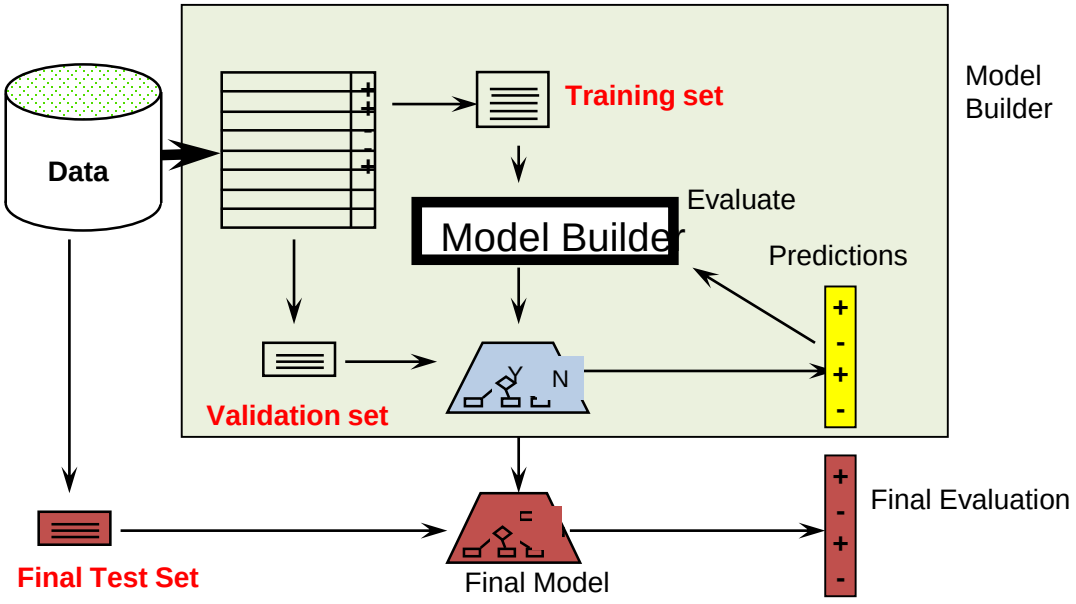
Problem 3: Error now is increased to 5% errors. Why?
Answer 3: Generalization error was measured multiple times on the test set, So model and hyper parameters are adapted to produce the best model for that particular set, unlikely to perform as well on new data.

**Learning : The test data can't be used for parameter tuning!**

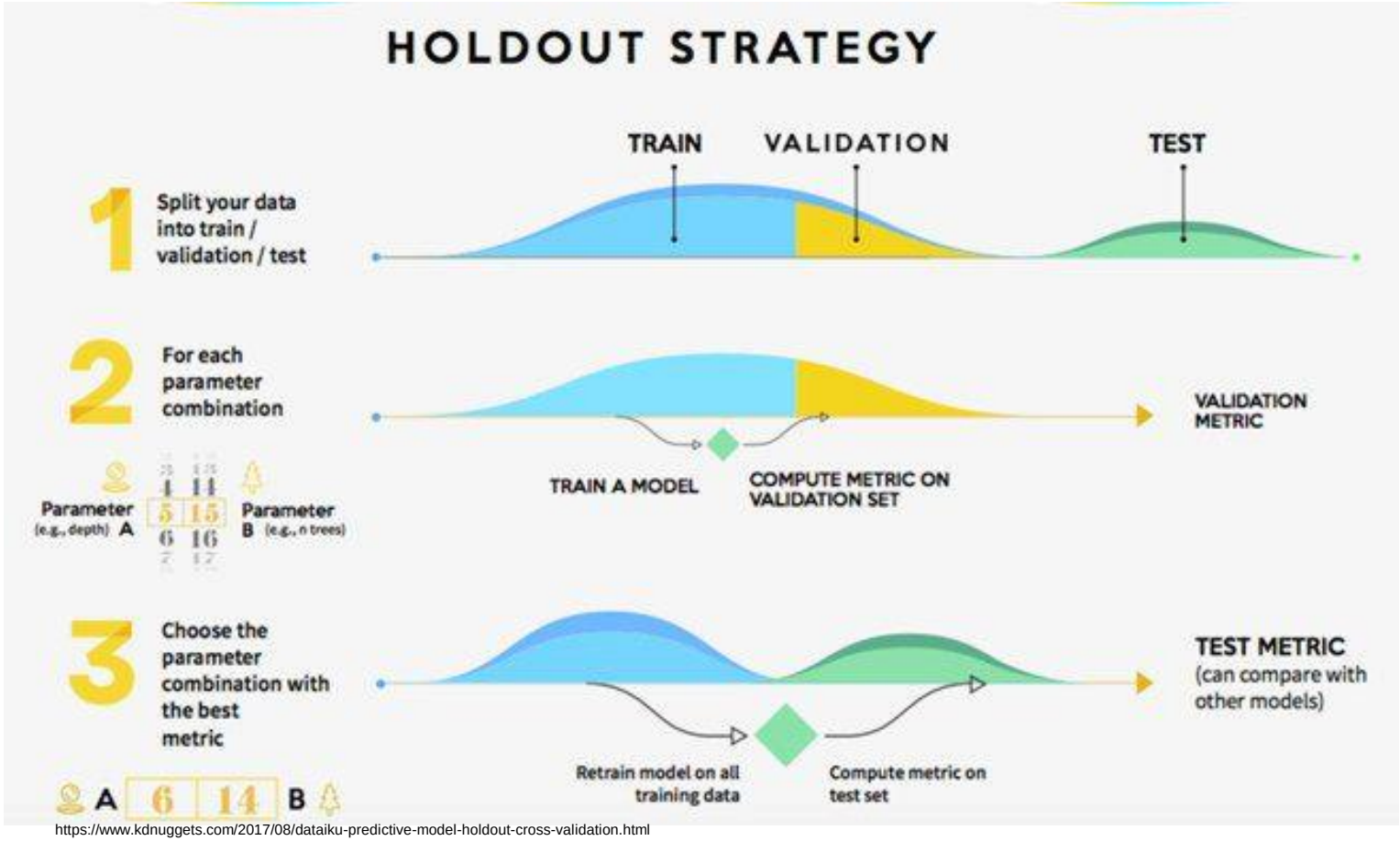
# Train, Validation, Test split

***Holdout validation (holdout or development, or dev set)***

1. Train multiple models with various hyper parameters on the reduced training set (i.e., the full training set minus the validation set)
2. Select the model that performs best on the validation set
3. After this holdout validation process, you train the best model on the full training set (including the validation set), to get the final model.
4. Lastly, evaluate this final model on the test set to get an estimate of the generalization error



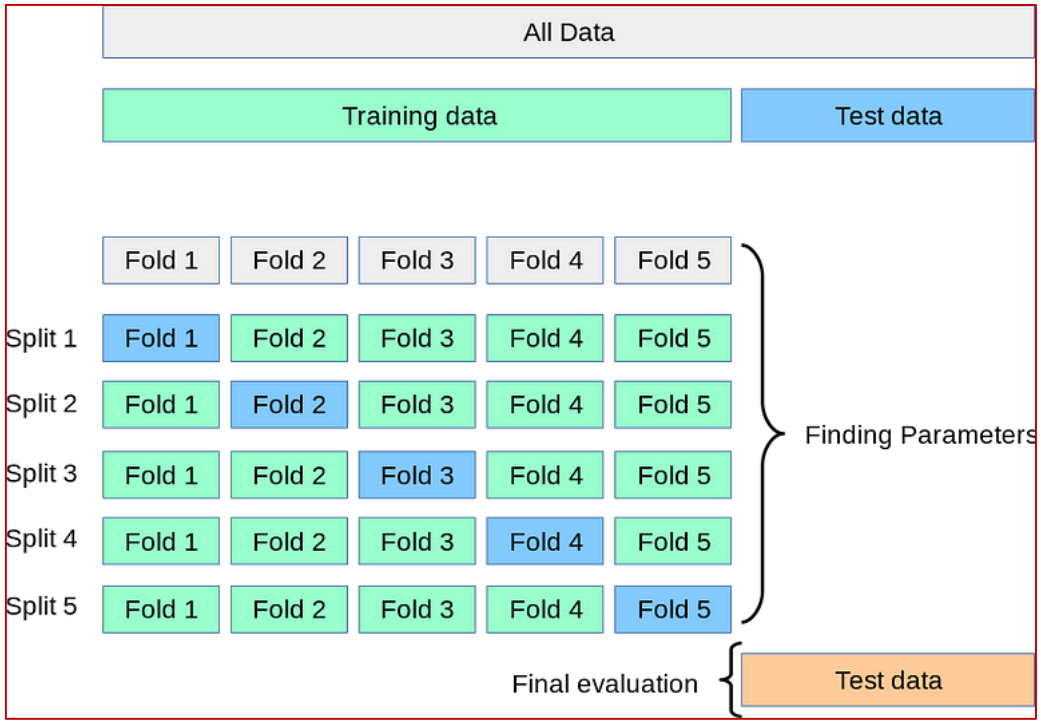
# Train-Test-Validation




https://www.kdnuggets.com/2017/08/dataiku-predictive-model-holdout-cross-validation.html

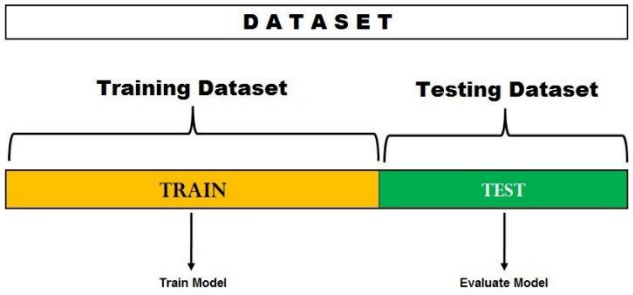
# Making most out of the data

- We don't want to touch the test set until we are ready to launch a model we are confident about
- If the validation set is too small, then model evaluations will be imprecise
  - Larger the test data the more accurate the error estimate
- If the validation set is too large, then the remaining training set will be much smaller than the full training set.
  - Larger the training data the better the classifier
- **Solution: cross validation**

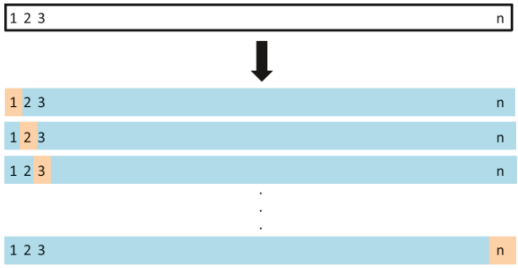


# Cross-Validation techniques

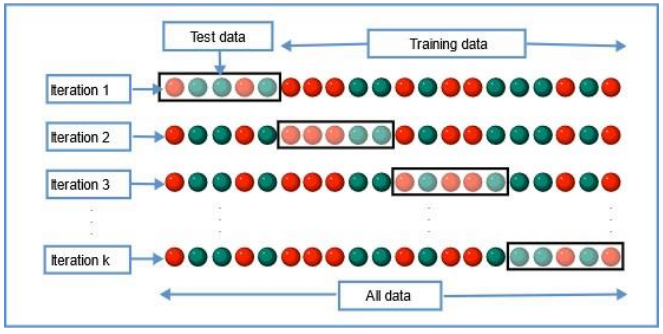
**Hold Out method**



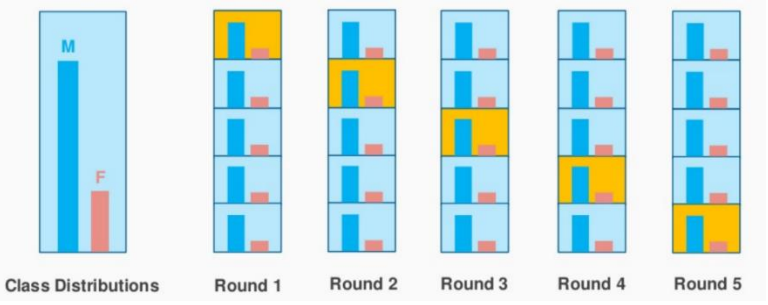

**Leave One Out Cross-Validation**




**K-Fold Cross-Validation**

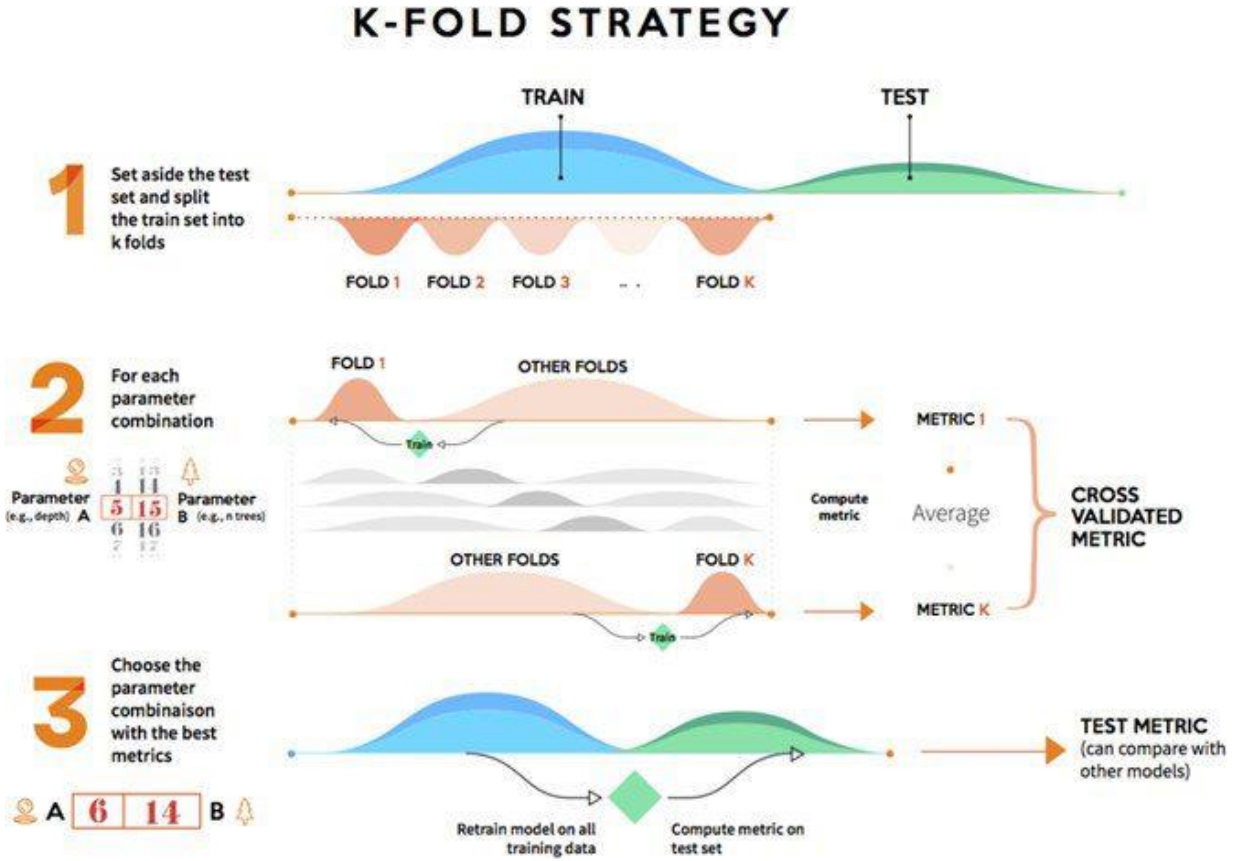



**Stratified K-Fold Cross-Validation**




https://www.analyticsvidhya.com/blog/2021/05/4-ways-to-evaluate-your-machine-learning-model-cross-validation-techniques-with-python-code/

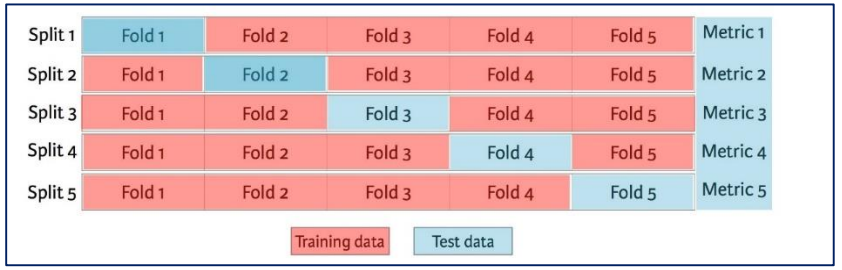
# Cross-Validation for hyper parameter tuning




https://www.kdnuggets.com/2017/08/dataiku-predictive-model-holdout-cross-validation.html

# Estimating generalization error : cross validation set Vs test set

- Training set performance is not a good indicator of generalization error
  - A more complex model overfits, a less complex one underfits
  - Which model do I select?
- Test set
  - Typically 80%, 20%
  - Wastes valuable labeled data
- Cross validation
  - In Sklearn : cross_val_score() (default is accuracy)
  - cross_val_score will automatically split data into train and test
  - fit the model on train data
  - score on test data
  - For 10 folds, 10 scores will be returned in scores variable. Then take average of that




```
model=DecisionTreeClassifier()
scores = cross_val_score(model, X, y, cv=10)
avg_score = np.mean(score_array)
print(avg_score)
```

# References


- T. Mitchell, “Machine Learning”, chapter 8, “Instance-Based Learning”
- “Locally Weighted Learning”, Christopher Atkeson, Andrew Moore, Stefan Schaal
- R. Duda et al., “Pattern recognition”, chapter 4 “Non-Parametric Techniques”

# Thank you !

**Required Reading for completed session :**

T1 - Chapter # 8 (Tom M. Mitchell, Machine Learning)

## Next Session Plan :

Module 7 : Support vector machines

Important Note for SVM : Below Math Pre-requisite (Yet to be discussed in MFML parallel course) :

Since the math basics will be covered in detail in math course, ML course is intended to discuss the application & interpretation of the model

- Lagrange Multipliers
- KKT – Conditions
- Primal – Dual for SVM
- Kernel for Non-Linear SVM