Mahidol University
Faculty of Medicine Ramathibodi Hospital
Department of Clinical Epidemiology and Biostatistics

K-Nearest Neighbors

RADI608: Data Mining and Machine Learning
RADI602: Data Mining and Knowledge Discovery

Lect. Anuchate Pattanateepapon, D.Eng.

Section of Data Science for Healthcare

Department of Clinical Epidemiology and Biostatistics

Faculty of Medicine Ramathibodi Hospital, Mahidol University

Wisdom of the Land

K-Nearest Neighbors (k-NN)

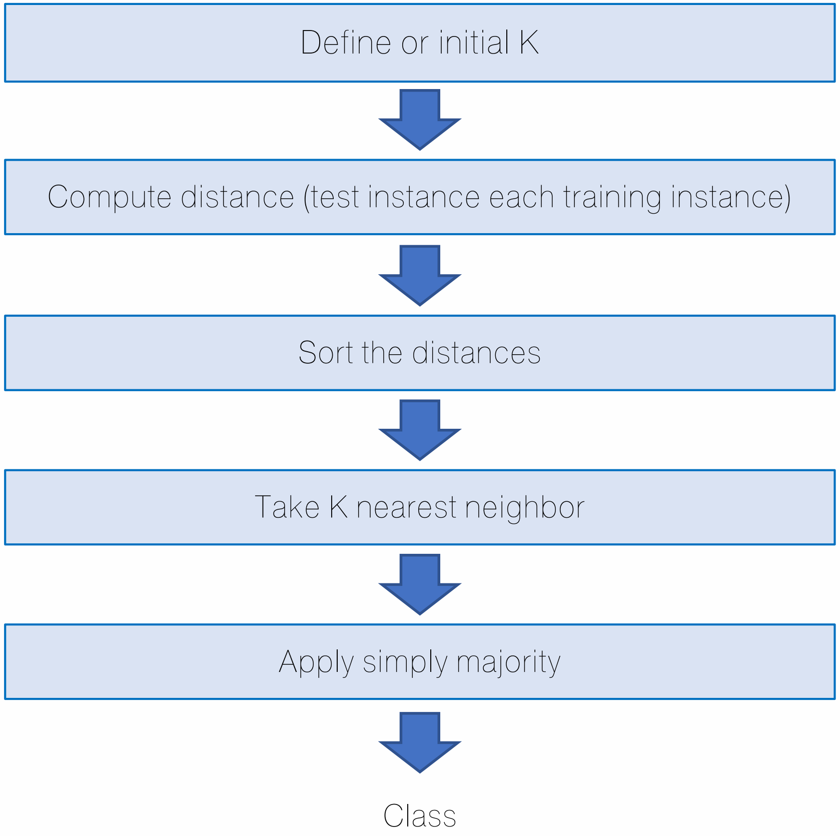
a) K-NN is a supervised training algorithm

b) No explicit training or model

c) Could perform with a classification and regression problem

d) Use the K-Nearest Neighbors of x to vote the label of x

A k-NN framework



Define or initial K

Compute distance (test instance each training instance)

Sort the distances

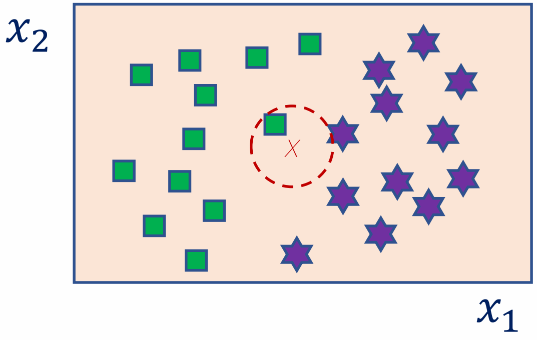
Take K nearest neighbor

Apply simply majority

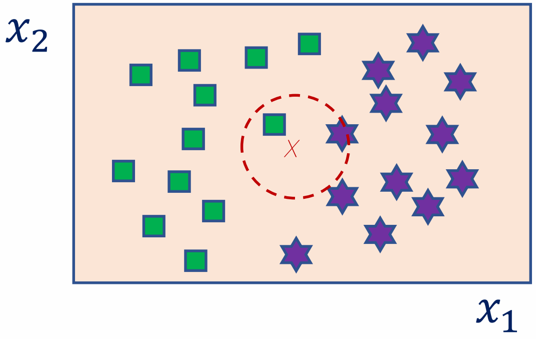
Class

A majority vote in K-Nearest Neighbors

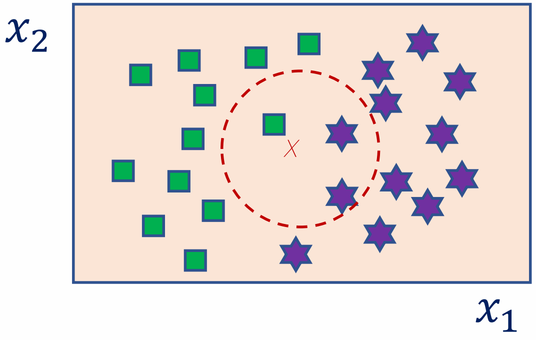
k-NN is classify by using the majority vote of the k closest training points (or predict the class of the query point, using distance-weighted voting)



1-nearest neighbor



2-nearest neighbor



3-nearest neighbor

Weighted voting and type of data

k-NN is classify by using the majority vote of the k closest training points (or predict the class of the query point, using weighted voting)

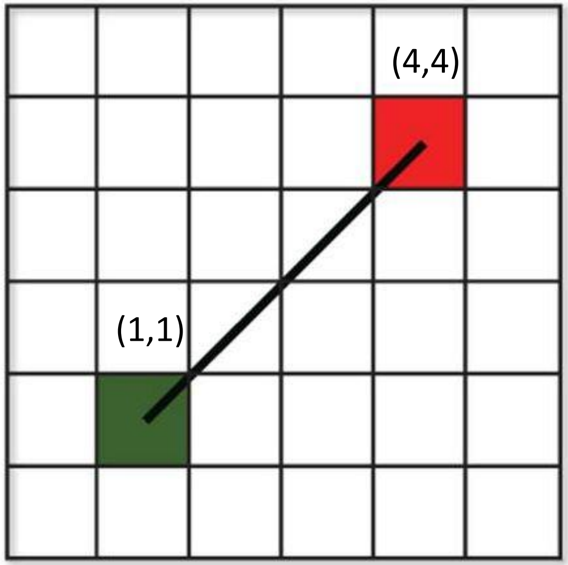
Numerical	apply Euclidean distance, Minkowski Distance, Manhattan Distance
Categorical	apply Cosine Distance
Two binary data strings	Hamming Distance

Euclidean distance

This distance is the most widely used one as it is the default metric that SKlearn library of Python uses for K-Nearest Neighbour. It is a measure of the true straight line distance between two points in Euclidean space.

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

$$= \sqrt{(4-1)^2 + (4-1)^2} = 4.24$$



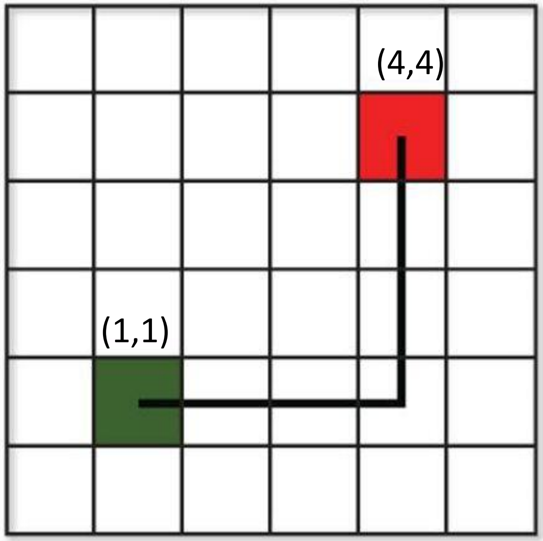
Euclidean Distance

Manhattan Distance

The Manhattan distance, also called the Taxicab distance or the City Block distance, calculates the distance between two real-valued vectors.

$$d = \sum_{i=1}^{n} |x_i - y_i|$$

d = |4-1| + |4-1| = 6



Manhattan Distance

Minkowski Distance

Minkowski Distance – It is a metric intended for real-valued vector spaces. We can calculate Minkowski distance only in a normed vector space, which means in a space where distances can be represented as a vector that has a length and the lengths cannot be negative. It is the generalized form of Euclidean and Manhattan Distance.

There are a few conditions that the distance metric must satisfy:

Non-negativity: d(x, y) >= 0

Identity: d(x, y) = 0 if and only if x == y

Symmetry: d(x, y) = d(y, x)

Triangle Inequality: d(x, y) + d(y, z) >= d(x, z)

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$$

The p value in the formula can be manipulated to give us different distances like:

p = 1, when p is set to 1 we get Manhattan distance

p = 2, when p is set to 2 we get Euclidean distance

Cosine Distance

Cosine Distance – This distance metric is used mainly to calculate similarity between two vectors. It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in the same direction. It is often used to measure document similarity in text analysis. When used with kNN this distance gives us a new perspective to a business problem and lets us find some hidden information in the data which we didn’t see using the above two distance matrices.

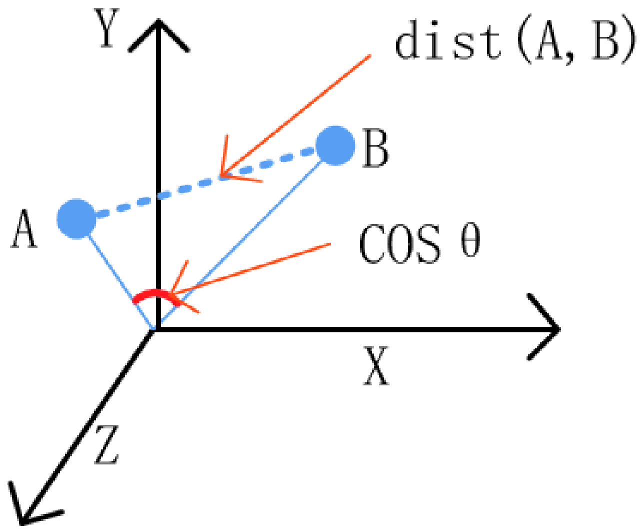
$$\cos \theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \cdot \|\vec{b}\|}$$

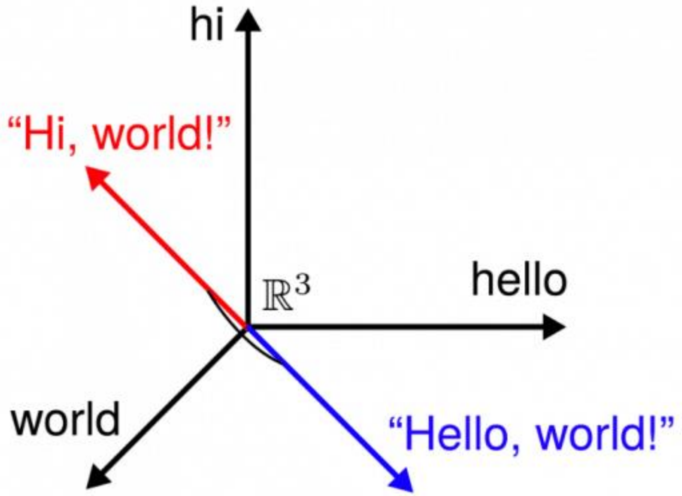
Using this distance we get values between 0 and 1, where 0 means the vectors are 100% similar to each other and 1 means they are not similar at all.

Cosine Similarity

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

0 means the vectors are 100% similar to each other
1 means they are not similar at all





Hamming Distance

Hamming Distance - Hamming distance is a metric for comparing two binary data strings. While comparing two binary strings of equal length, Hamming distance is the number of bit positions in which the two bits are different. The Hamming distance method looks at the whole data and finds when data points are similar and dissimilar one to one. The Hamming distance gives the result of how many attributes were different.

Hamming Distance

Suppose we have two strings “ABCDE” and “AGDDF” of same length and we want to find the hamming distance between these. We will go letter by letter in each string and see if they are similar or not like first letters of both strings are similar, then second is not similar and so on.

ABCDE and AGDDF

When we are done doing this we will see that only two letters marked in red were similar and three were dissimilar in the strings. Hence, the Hamming Distance here will be 3. Note that larger the Hamming Distance between two strings, more dissimilar will be those strings (and vice versa).

k-NN in a classification problem

- No explicitly decision boundaries computation
- The boundaries between distinct classes form a subset of the Voronoi diagram of the training data
- Each line segment is equidistant to neighboring points

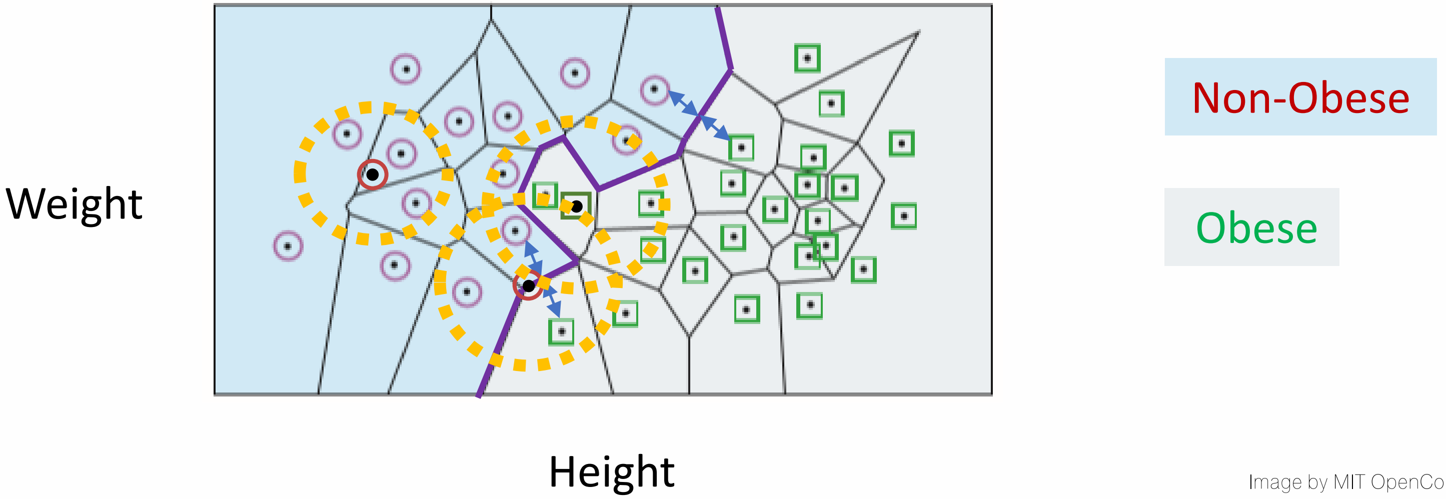
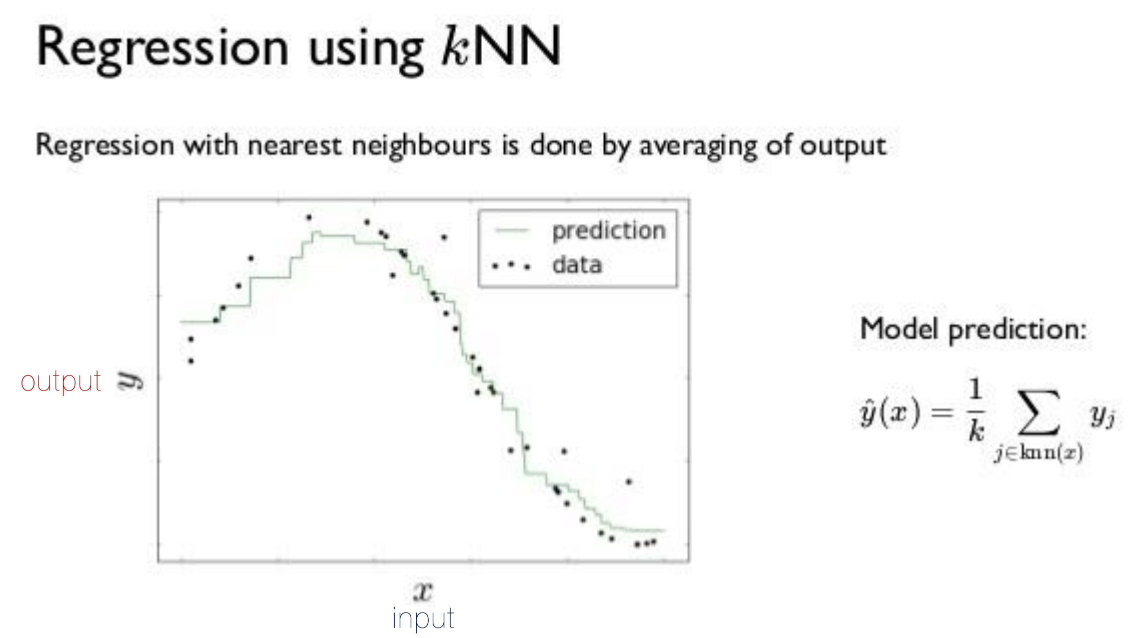


Image by MIT OpenCourseWare

k-NN in a regression problem

the value for the testing data becomes the (weighted) average of the values of the K neighbors: 1-D k-NN regression



Regression using kNN

Regression with nearest neighbours is done by averaging of output

Model prediction:

$$\hat{y}(x) = \frac{1}{k} \sum_{j \in \text{knn}(x)} y_j$$

Image by https://www.slideshare.net/arogozhnikov/machine-learning-in-science-and-industry-day-1

Summary

a) k-NN can deal with complex and arbitrary decision boundaries

b) Despite its simplicity, researchers have shown that the classification accuracy of k-NN can be quite strong and in many cases as accurate as those elaborated method

c) k-NN is slow at the classification time

d) k-NN does not produce an understandable model

k-NN in Python

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

k-NN in Python

```
X = [[0], [1], [2], [3]]
y = [0, 0, 1, 1]
from sklearn.neighbors import KNeighborsClassifier
neigh = KNeighborsClassifier(n_neighbors=3)
neigh.fit(X, y)

print(neigh.predict([[1.1]]))
print(neigh.predict_proba([[1.1]]))

print(neigh.predict([[2.9]]))
print(neigh.predict_proba([[2.9]]))
```

```
[0]
```

```
[[0.66666667 0.33333333]]
```

```
[1]
```

```
[[0.33333333 0.6666666 ]]
```

k-NN in Python

```
samples = [[0., 0., 0.], [0., .5, 0.], [1., 1., .5]]
from sklearn.neighbors import NearestNeighbors
neigh = NearestNeighbors(n_neighbors=1)
neigh.fit(samples)

print(neigh.kneighbors([[1., 1., 1.]]))
```

we construct a Nearest Neighbors class from an array representing our data set and ask who's the closest point to [1,1,1]

```
(array([[0.5]]), array([[2]], dtype=int64))
```

means that the element is at distance 0.5 and is the third element of samples (indexes start at 0)

k-NN in Python

```
print(__doc__)


import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn import neighbors, datasets


n_neighbors = 15


# import some data to play with
iris = datasets.load_iris()
```

Set n = 15

k-NN in Python

```
# we only take the first two features. We could avoid this ugly
# slicing by using a two-dim dataset
X = iris.data[:, :2]
y = iris.target

h = .02  # step size in the mesh

# Create color maps
cmap_light = ListedColormap(['orange', 'cyan', 'cornflowerblue'])
cmap_bold = ListedColormap(['darkorange', 'c', 'darkblue'])
```

k-NN in Python

```
for weights in ['uniform', 'distance']:
    # we create an instance of Neighbours Classifier and fit the data.
    clf = neighbors.KNeighborsClassifier(n_neighbors, weights=weights)
    clf.fit(X, y)

    # Plot the decision boundary. For that, we will assign a color to each
    # point in the mesh [x_min, x_max]x[y_min, y_max].
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                         np.arange(y_min, y_max, h))
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
```

Return coordinate matrices from coordinate vectors

the array creation routines based on numerical ranges

Translates slice objects to concatenation along the second axis.

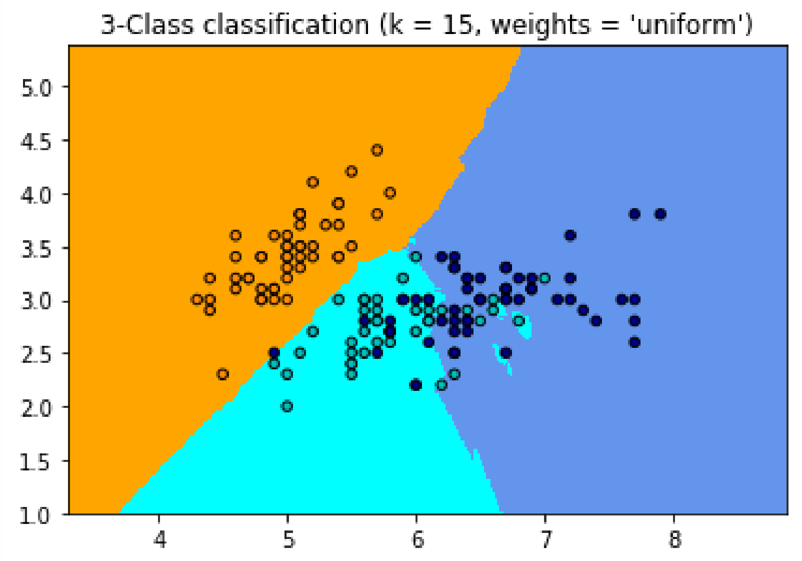
k-NN in Python

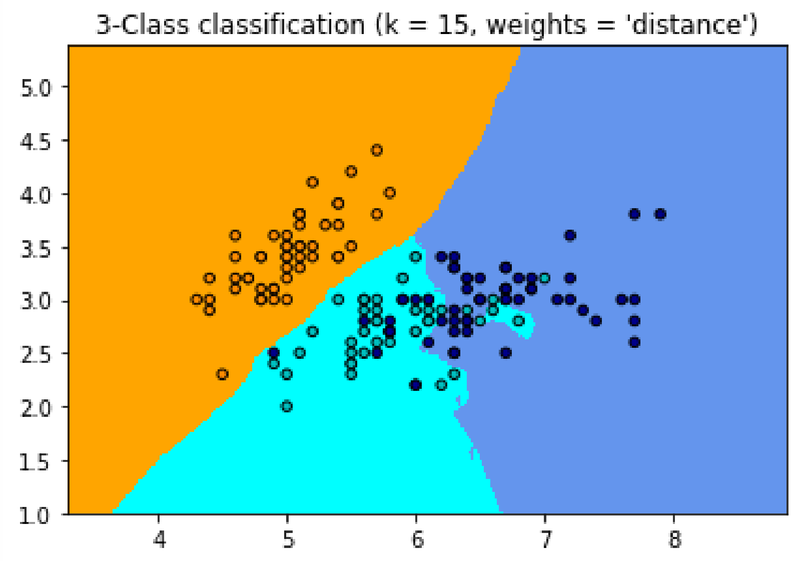
```
    # Put the result into a color plot
    Z = Z.reshape(xx.shape)
    plt.figure()
    plt.pcolormesh(xx, yy, Z, cmap=cmap_light)

    # Plot also the training points
    plt.scatter(X[:, 0], X[:, 1], c=y, cmap=cmap_bold,
          edgecolor='k', s=20)
    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())
    plt.title("3-Class classification (k = %i, weights = '%s')"
         % (n_neighbors, weights))

plt.show()
```

k-NN in Python





predict the class of the query point, using uniform-weighted voting: Treat of each point weight the same regardless of the distance.

predict the class of the query point, using distance-weighted voting: The closer is the more weight.

Pros and Cons of k-NN

Pros:

1. k-NN algorithm is very simple to understand and equally easy to implement.
2. k-NN has no assumptions.
3. No training step.
4. k-NN has one hyper parameter.
5. Can be used both for Classification and Regression
6. Variety of distance criteria to be choose. Such as,
    - Euclidean distance
    - Manhattan distance
    - Etc.

Pros and Cons of k-NN

Cons:

1. k-NN slow algorithm.
2. k-NN works well with small number of input variables
3. Optimal number of neighbors.
4. Outlier sensitivity.
5. k-NN inherently has no capability of dealing with missing value problem.
6. k-NN doesn't perform well on imbalanced data. For class A and B, if the majority of the training data is labeled as A, then the model will ultimately give a lot of preference to A.
7. k-NN needs homogeneous features. If we decide using Euclidean or Manhattan distances, it is completely necessary that features have the same scale.

Assignment:

kNN - due on 11 November, 2022 (10 points)

a. (5 points)

- From colon.csv
- Sampling (choose your seed) a training set and a testing set = 80:20
- Perform k-NN algorithm to predict which patient (sample) have a cancer by setting k = [1, 2, 4, 6, 8, 10, 12] and weights = distance, and using a testing set from previous step.

b. (5 points)

- From your training set and test set in a
- Perform k-NN algorithm to predict which patient (sample) have a cancer by setting k = [1, 2, 4, 6, 8, 10, 12] and weights = uniform, and using a testing set from previous step.
- Compare the performance of the model from item a & b, and add your comments

References

1. Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician. 46 (3): 175–185. doi:10.1080/00031305.1992.10475879.