b-it Bonn-Aachen International Center for Information Technology

Introduction to Machine Learning Tutorial 03

b-it Bonn-Aachen International Center for Information Technology

Pearson Correlation Coefficient

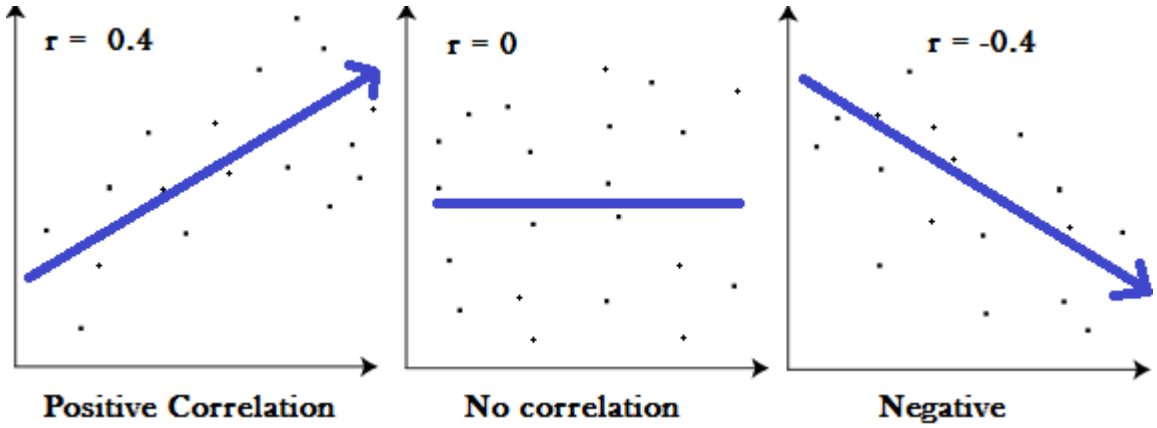
Correlation coefficients are used in statistics to measure how strong a relationship is between two variables.

There are several types of correlation coefficients. Pearson's correlation is one of them. It is (also called **Pearson's *R***) is a **correlation coefficient** commonly used in linear regression.

If you're starting out in statistics, you'll probably learn about Pearson's *R* first. In fact, when anyone refers to **the correlation coefficient**, they are usually talking about Pearson's.

In machine learning we are interested in correlations because they can tell us if a feature might be meaningful to include into our classifier or not.

Pearson Correlation Coefficient



- 1 indicates a strong positive relationship.
- -1 indicates a strong negative relationship.
- A result of zero indicates no relationship at all.

Example: Calculation of Pearson

Are the features of age (x) and glucose level (y) correlated?

Subject	Age X	Glucose Level Y
1	43	99
2	21	65
3	25	79
4	42	75
5	57	87
6	59	81

Example: Calculation of Pearson

Pearson correlation

$$r = \frac{n(\Sigma xy) - (\Sigma x)(\Sigma y)}{\sqrt{[\, n\Sigma x^2 - (\Sigma x)^2 \,]\,[\, n\Sigma y^2 - (\Sigma y)^2 \,]}}$$

b-it Bonn-Aachen International Center for Information Technology

Example: Calculation of Pearson

Are the features of age (x) and glucose level (y) correlated?

Subject	Age X	Glucose Level Y	XY	X^2	Y^2
1	43	99	4257	1849	9801
2	21	65	1365	441	4225
3	25	79	1975	625	6241
4	42	75	3150	1764	5625
5	57	87	4959	3249	7569
6	59	81	4779	3481	6561

Example: Calculation of Pearson Coeffcient

From our table:

$\Sigma x = 247$

$\Sigma y = 486$

$\Sigma xy = 20{,}485$

$\Sigma x^2 = 11{,}409$

$\Sigma y^2 = 40{,}022$

n is the sample size, in our case = 6

$$r = \frac{n(\Sigma xy) - (\Sigma x)(\Sigma y)}{\sqrt{[\, n\Sigma x^2 - (\Sigma x)^2 \,][\, n\Sigma y^2 - (\Sigma y)^2 \,]}}$$

Result:

2868 / 5413.27 = 0.529809

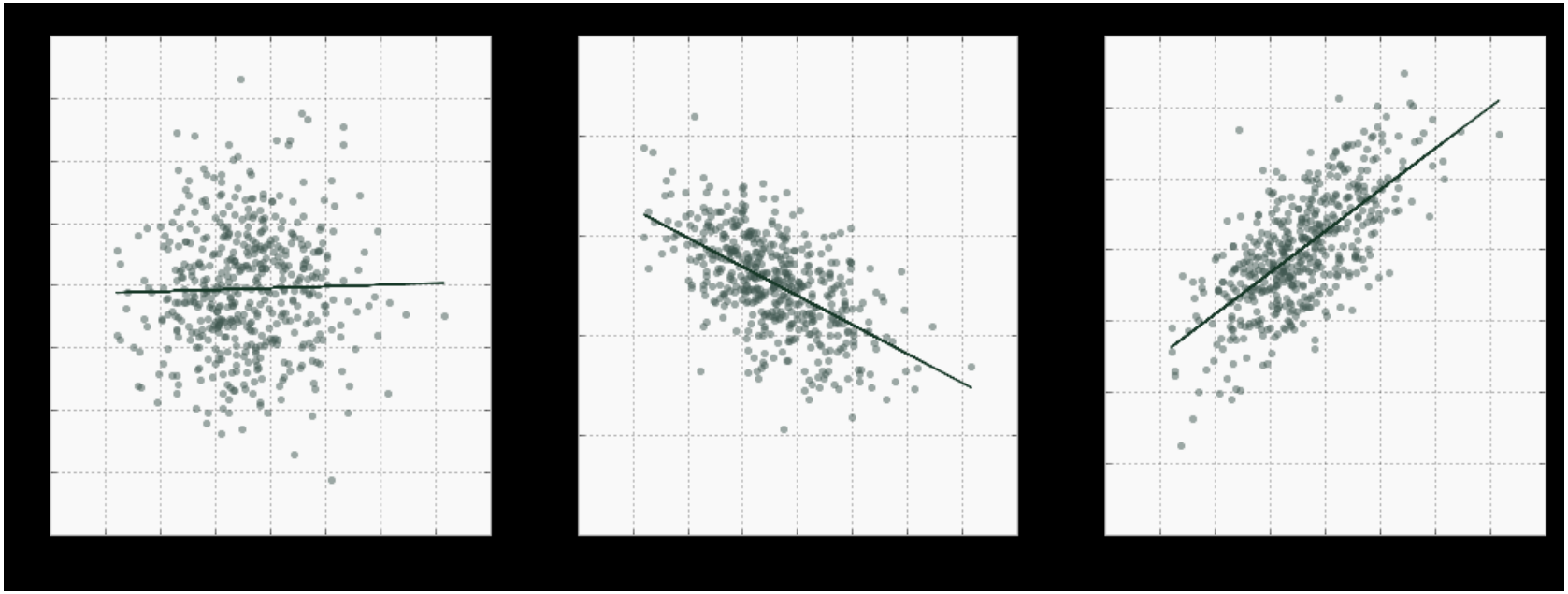
b-it Bonn-Aachen International Center for Information Technology

Example: Pearson Coeffcients

Pearson = 0.03

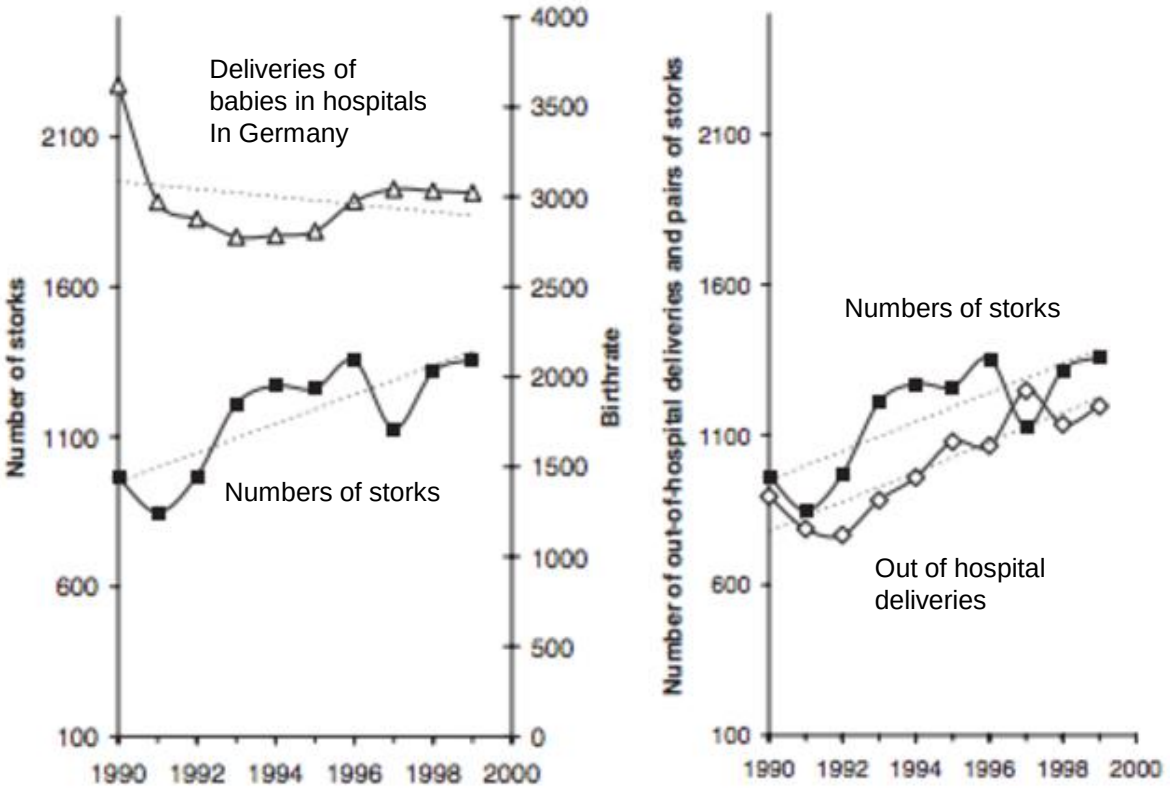
Pearson = -0.58

Pearson = 0.69



Correlation and Causation

Correlation and causation are not the same: If this were the case, then the adjacent figure would „prove“ that babies are delivered by storks.



Overview of Pearson Coefficients

R value =	
+.70 or higher	Very strong positive relationship
+.40 to +.69	Strong positive relationship
+.30 to +.39	Moderate positive relationship
+.20 to +.29	weak positive relationship
+.01 to +.19	No or negligible relationship
0	No relationship [zero correlation]
-.01 to -.19	No or negligible relationship
-.20 to -.29	weak negative relationship
-.30 to -.39	Moderate negative relationship
-.40 to -.69	Strong negative relationship
-.70 or higher	Very strong negative relationship

Spearman Correlation

Spearman's rank correlation coefficient can be defined as a special case of Pearson applied to **ranked (sorted) variables**.

Unlike Pearson, Spearman's correlation is **not restricted to linear** relationships and is the **non-parametric version** of the Pearson coefficient.

Instead, it measures **monotonic association** (only strictly increasing or decreasing, but not mixed) between two variables and relies on the rank order of values.

In other words, rather than **comparing** means and variances, Spearman's coefficient looks at the **relative order of values for each variable**.

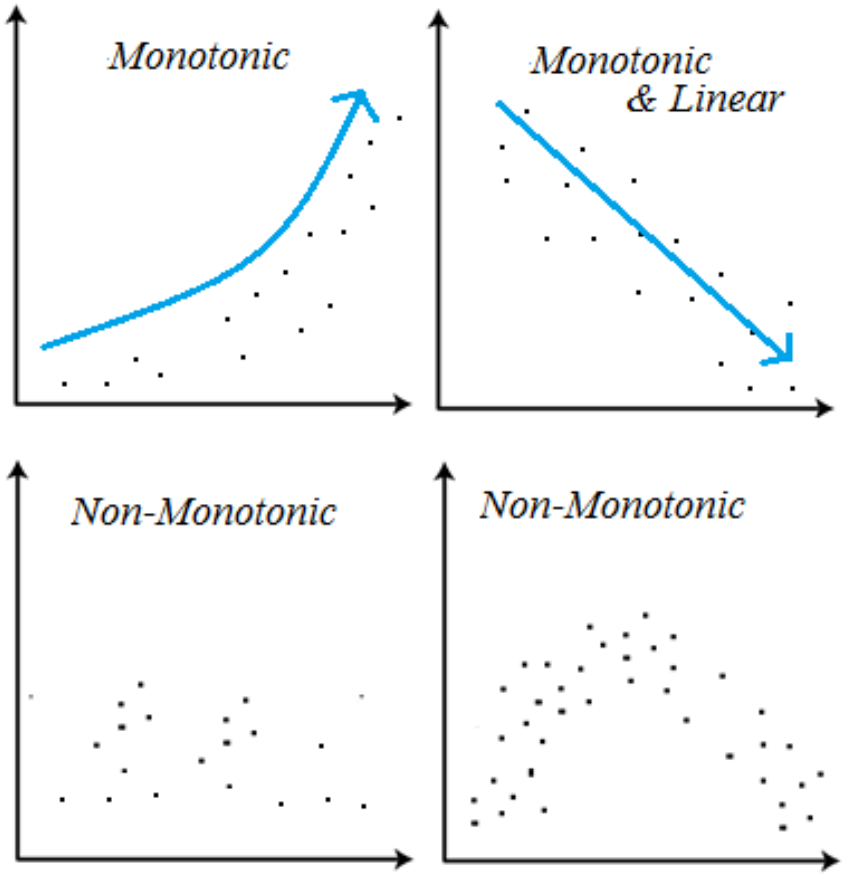
This makes it appropriate to use with both **continuous** and **discrete data.**

b-it Bonn-Aachen International Center for Information Technology

Monotonic

Monotonic relationships are where:

- One variable increases and the other increases. Or,
- One variable decreases and the other decreases.



Spearman Correlation

How is it calculated?

$$r_s = \frac{cov(rg_X, rg_Y)}{\sigma_{rg_X}, \sigma_{rg_Y}}$$

Whereas:

r_s	Spearman coefficient
cov	covariance of the rank variables
rg_X	rank of X
rg_Y	rank of Y
σ	standard deviation of ranks variables

Spearman Correlation

It can only be calculated with the popular formula below if all n are distinct integers it can be computed as:

$$r_s = 1 - \frac{6 \sum d_i^2}{n\,(n^2 - 1)}$$

Whereas:

d_i distance between the two ranks of each observation
$d_i = rg(X_i) - rg(Y_i)$

n number of observations or sample size

Calculation of Spearman Correlation

IQ (X_i)	Hours spent watching TV (Y_i)
86	0
97	20
99	28
100	27
101	50
103	29
106	7
110	17
112	6
113	12

b-it Bonn-Aachen International Center for Information Technology

Calculation of Spearman Correlation

IQ (X_i)	Hours spent watching TV (Y_i)	Rank (x_i)	Rank (y_i)	d_i	d_i^2
86	0	1	1	0	0
97	20	2	6	-4	16
99	28	3	8	-5	25
100	27	4	7	-3	9
101	50	5	10	-5	25
103	29	6	9	-3	9
106	7	7	3	4	16
110	17	8	5	3	9
112	6	9	2	7	49
113	12	10	4	6	36

Calculation of Spearman Correlation

$$r_s = 1 - \frac{\mathrm{n} \sum d_i^2}{n\,(n^2 - 1)}$$

$$r_s = 1 - \frac{6\ x\ 194}{10\ (10^2 - 1)}$$

$r_s = 0{,}18 \rightarrow$ week correlation

Calculation of Spearman Correlation

The Spearman correlation coefficient ranges between 1 and -1:

1	perfect positive correlation
0	no correlation
-1	perfect negative correlation

Spearman Correlation Coefficients

r_s value	Interpretation
0.00 -0.19	Very week
0.20 – 0.39	week
0.40 – 0.59	Moderate
0.60 – 0.79	Strong
0.80 – 1.0	Very strong

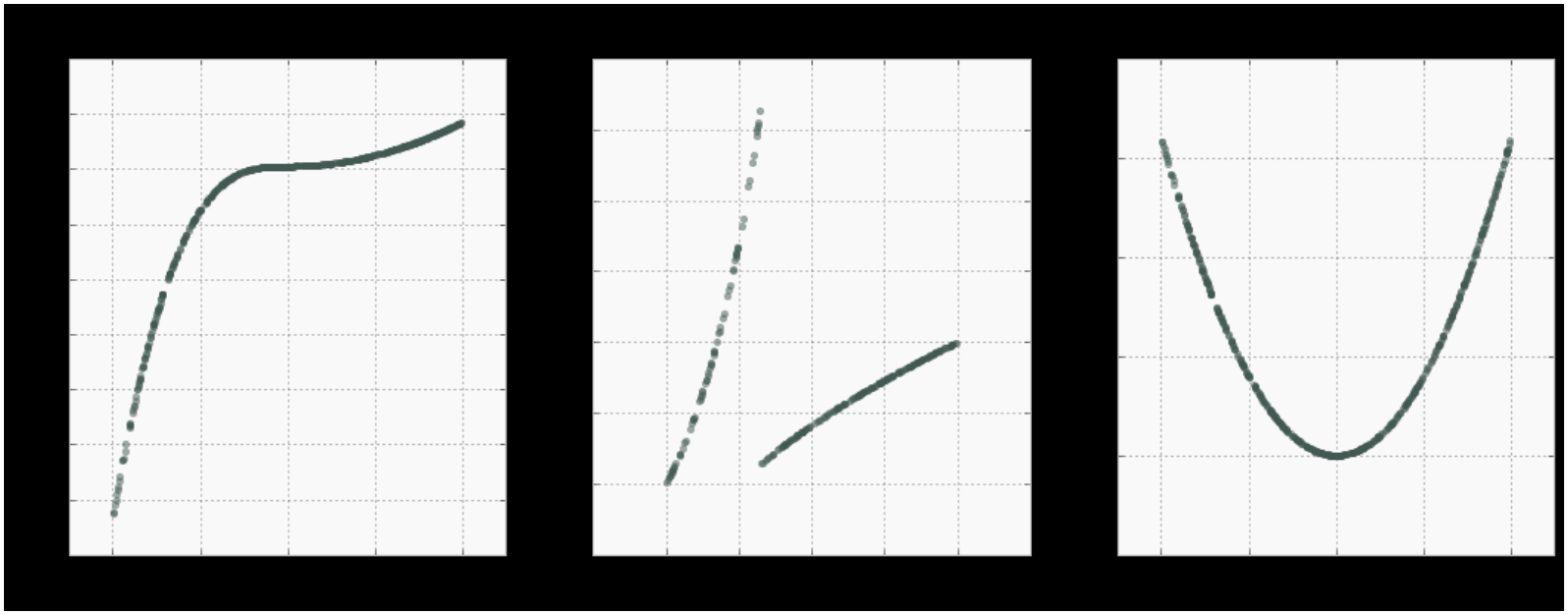
Comparision Spearman and Pearson Correlation

The best way to understand Pearson and Spearman correllation is to look at an example.

Pearson = 0.8
Spearman = 1.0

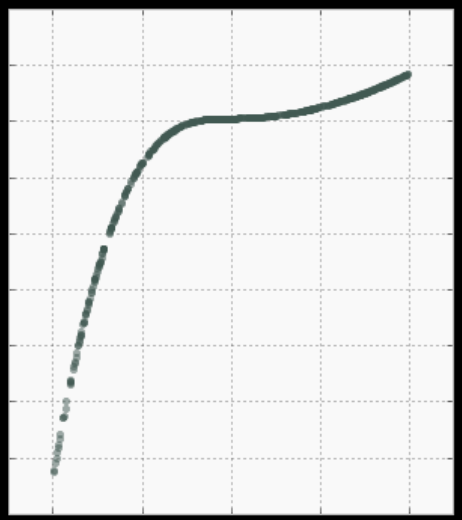
Pearson = 0.01
Spearman = 0.25

Pearson = 0.02
Spearman = 0.01



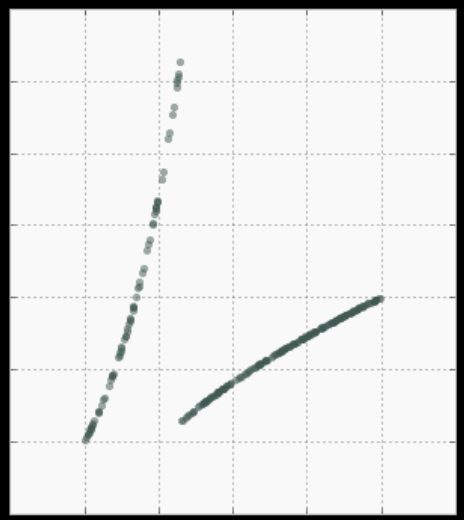
Spearman and Pearson Correlation

Pearson = 0.8
Spearman = 1.0



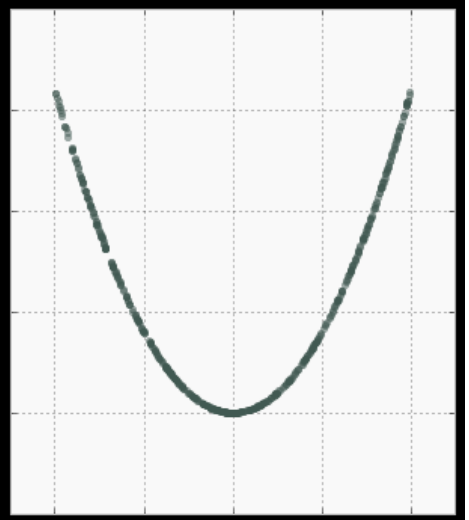
Both work well, as data mostly linear and monotonic

Pearson = 0.01
Spearman = 0.25



Data non-monotonic, both fail, Spearman les than Pearson.

Pearson = 0.02
Spearman = 0.01



This graph is monotonic but not linear. Both coefficients fail as there is a parabolic relationship.

Corellation Conclusion

This correlation example shows very clearly why it is usually very useful to **visualize data** before choosing an algorithm.

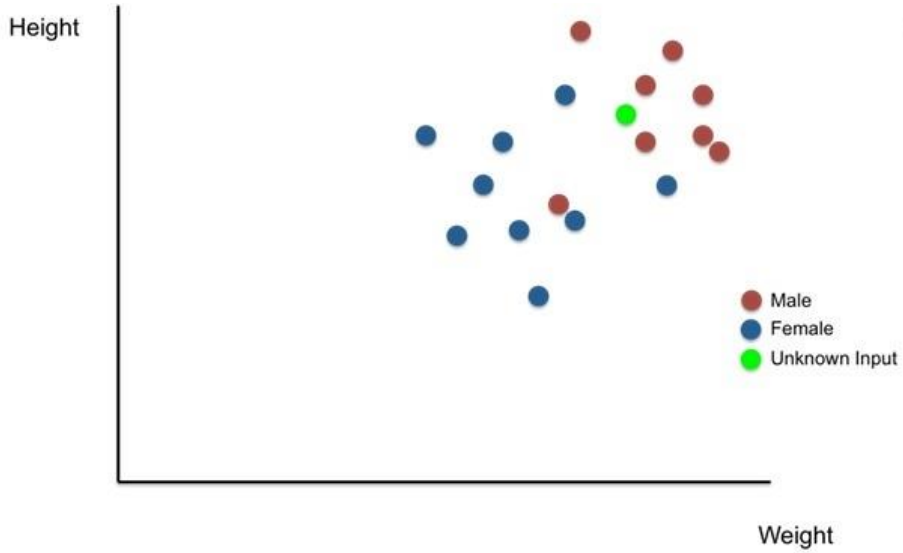
It is therefore always a good idea to use visualization techniques and multiple statistical data summaries to get a better pictures of **how your variables relate to each other.**

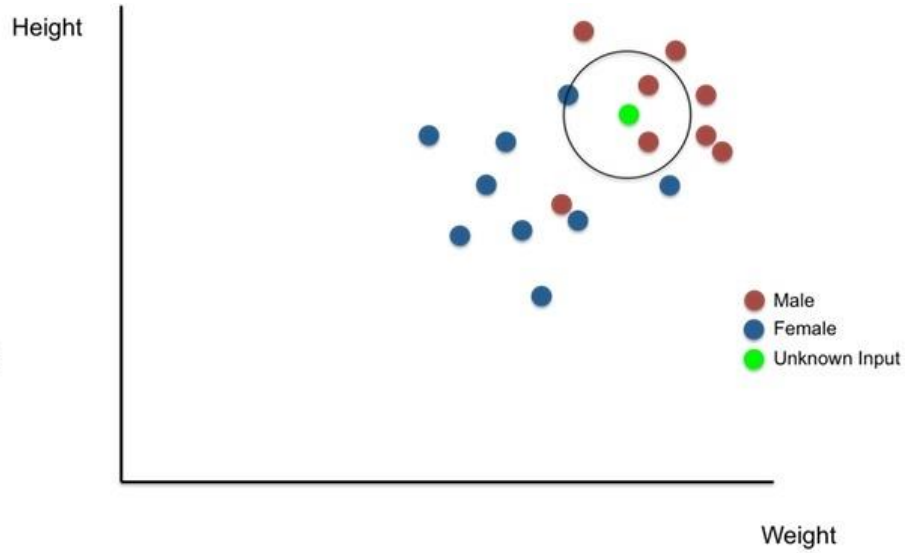
K-Nearest Neighbor

How does K-Nearest Neighbor (k-NN) algorithm work?

Let’s look at an example!

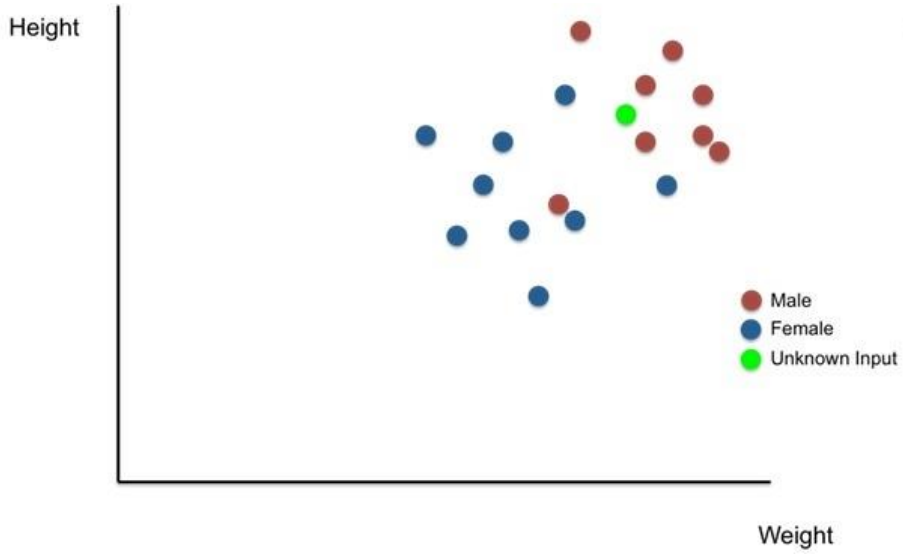
K-Nearest Neighbor

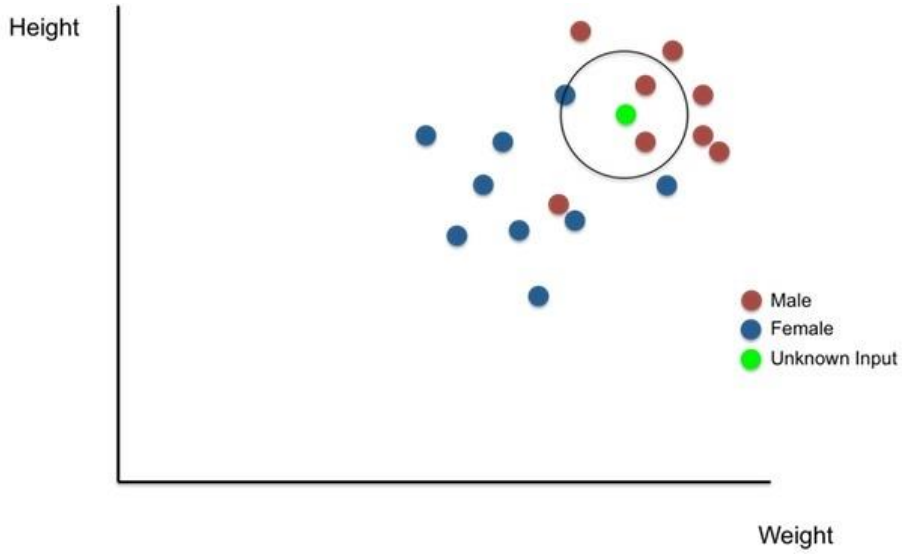




For example, suppose a k-NN algorithm was given an input of data points of specific men and women's **weight** and **height**, as plotted above. To **determine the gender of an unknown input** (green point), *k*-NN can look at the **nearest *k* neighbors** (suppose k= 3) and will determine that the input's gender is male. This method is a very simple and logical way of marking unknown inputs, with a high rate of success.

K-Nearest Neighbor





In our use case, k-NN uses the **Euclidian distance** to determine the distances between the closest **data points**.

K-Nearest Neighbor - Properties

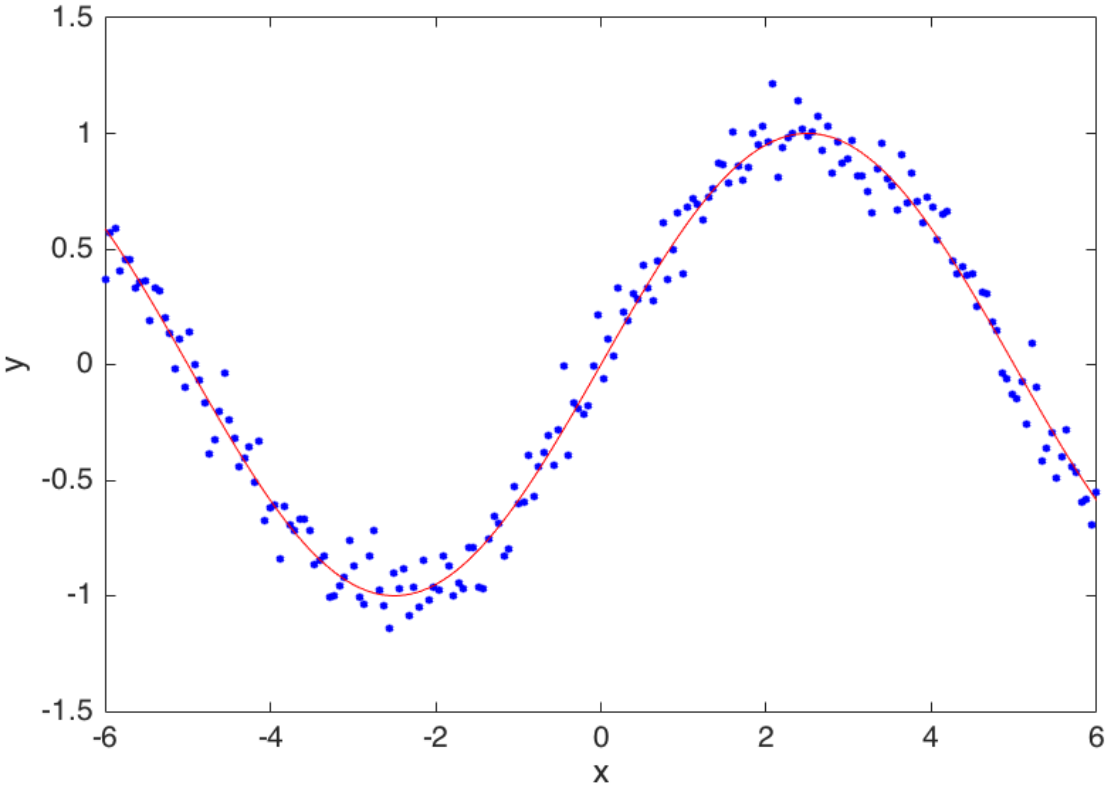
Properties :

- Non-parametric (no assumptions about the probability distribution of the input.)

b-it Bonn-Aachen International Center for Information Technology

K-Nearest Neighbor - Non-Parametric

In the case, the input data has the distribution of a sinus curve, k-NN will look **locally,** at the e. g. five nearest neighbors, to „determine“ or better to „vote“ the input point.



K-Nearest Neighbor - Properties

Properties :

- Non-parametric (no assumptions about the probability distribution of the input.)
- More robust (as it is non parametric)
- Lazy learning (learning method that generalizes data in the testing phase, rather than during the training phase)
- Can quickly adapt to changes

K-Nearest Neighbor - Pros and Cons

Pros:

• ***Very easy to understand and implement***. A *k*-NN implementation does not require much code and can be a quick and simple way to begin machine learning datasets.

• ***Does not assume any probability distributions on the input data***. This can come in handy for inputs where the probability distribution is unknown and is therefore robust.

• ***Can quickly respond to changes in input***. *k*-NN employs lazy learning, which generalizes during testing--this allows it to change during real-time use.

K-Nearest Neighbor - Pros and Cons

Cons:

• ***Sensitive to localized data.*** Since *k*-NN gets all of its information from the input's neighbors, localized anomalies affect outcomes significantly, rather than for an algorithm that uses a generalized view of the data.

• ***Computation time.*** Lazy learning requires that most of *k*-NN's computation be done during testing, rather than during training. This can be an issue for large datasets.

• ***Normalization.*** If one type of category occurs much more than another, classifying an input will be more biased towards that one category (since it is more likely to be neighbors with the input). This can be mitigated by applying a lower weight to more common categories and a higher weight to less common categories; however, this can still cause errors near decision boundaries

Parameter Selection

Parameter selection is performed for most machine learning algorithms, including *k*-NN.

To determine the **number of neighbors (a hyperparameter)** to consider when running the algorithm (*k*), a common method involves choosing the optimal *k* for a validation set (the one that reduces the percentage of errors) and using that for the test set.

In general, a higher *k* reduces noise and localized anomalies but allows for more error near decision boundaries, and vice versa.

More Tasks for Project 1

1. Change the number of neighbors (the k) in your program such as

k = 3
k = 4
k = 7

How does the accuracy change? Does it change at all?

2. Check the respective performance of the algorithm using accuracy, precision and sensitivity (or recall).

More Tasks for Project 1

3. Check for each change accuracy, precision and sensitivity (= recall).

```
# Code for precision and sensitivity (recall):

from sklearn.metrics import precision_score, recall_score

precision = precision_score (labels_test, pred)
recall = recall_score(labels_test, pred)

print('Precision: {:.2f}'.format(precision))
print ('Recall: {:.2f}'.format(recall))
```

More Tasks for Project 1

4. Which performance test is more indicative in this example? Explain by giving examples from the tasks!