**Machine Learning**

Machine Learning is making the computer learn from studying data and statistics.

Machine Learning is a step into the direction of artificial intelligence (AI).

Machine Learning is a program that analyses data and learns to predict the outcome.

## Data Set

In the mind of a computer, a data set is any collection of data. It can be anything from an array to a complete database.

**Example of an array:**

[99,86,87,88,111,86,103,87,94,78,77,85,86]

**Example of a database:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Carname | Color | Age | Speed | AutoPass |
| BMW | red | 5 | 99 | Y |
| Volvo | black | 7 | 86 | Y |
| VW | gray | 8 | 87 | N |
| VW | white | 7 | 88 | Y |
| Ford | white | 2 | 111 | Y |
| VW | white | 17 | 86 | Y |
| Tesla | red | 2 | 103 | Y |
| BMW | black | 9 | 87 | Y |
| Volvo | gray | 4 | 94 | N |
| Ford | white | 11 | 78 | N |
| Toyota | gray | 12 | 77 | N |
| VW | white | 9 | 85 | N |
| Toyota | blue | 6 | 86 | Y |

By looking at the array, we can guess that the average value is probably around 80 or 90, and we are also able to determine the highest value and the lowest value, but what else can we do?

And by looking at the database we can see that the most popular color is white, and the oldest car is 17 years, but what if we could predict if a car had an AutoPass, just by looking at the other values?

That is what Machine Learning is for! Analyzing data and predicting the outcome!

In Machine Learning it is common to work with very large data sets. In this tutorial we will try to make it as easy as possible to understand the different concepts of machine learning, and we will work with small easy-to-understand data sets.

## Data Types

To analyze data, it is important to know what type of data we are dealing with.

We can split the data types into three main categories:

* **Numerical**
* **Categorical**
* **Ordinal**

**Numerical** data are numbers, and can be split into two numerical categories:

* Discrete Data  
  - numbers that are limited to integers. Example: The number of cars passing by.
* Continuous Data  
  - numbers that are of infinite value. Example: The price of an item, or the size of an item

**Categorical** data are values that cannot be measured up against each other. Example: a color value, or any yes/no values.

**Ordinal** data are like categorical data, but can be measured up against each other. Example: school grades where A is better than B and so on.

By knowing the data type of your data source, you will be able to know what technique to use when analyzing them.

# Machine Learning - Mean Median Mode

What can we learn from looking at a group of numbers?

In Machine Learning (and in mathematics) there are often three values that interests us:

* **Mean** - The average value
* **Median** - The mid point value
* **Mode** - The most common value

**Example:** We have registered the speed of 13 cars:

speed = [99,86,87,88,111,86,103,87,94,78,77,85,86]

What is the average, the middle, or the most common speed value?

## Mean

The mean value is the average value.

To calculate the mean, find the sum of all values, and divide the sum by the number of values:

(99+86+87+88+111+86+103+87+94+78+77+85+86) / 13 = 89.77

### Example

Use the NumPy mean() method to find the average speed:

import numpy  
  
speed = [99,86,87,88,111,86,103,87,94,78,77,85,86]  
  
x = numpy.mean(speed)  
  
print(x)

## Median

The median value is the value in the middle, after you have sorted all the values:

77, 78, 85, 86, 86, 86, 87, 87, 88, 94, 99, 103, 111

It is important that the numbers are sorted before you can find the median.

The NumPy module has a method for this:

### Example

Use the NumPy median() method to find the middle value:

import numpy  
  
speed = [99,86,87,88,111,86,103,87,94,78,77,85,86]  
  
x = numpy.median(speed)  
  
print(x)

If there are two numbers in the middle, divide the sum of those numbers by two.

77, 78, 85, 86, 86, 86, 87, 87, 94, 98, 99, 103  
  
(86 + 87) / 2 = 86.5

### Example

Using the NumPy module:

import numpy  
  
speed = [99,86,87,88,86,103,87,94,78,77,85,86]  
  
x = numpy.median(speed)  
  
print(x)

## Mode

The Mode value is the value that appears the most number of times:

99, 86, 87, 88, 111, 86, 103, 87, 94, 78, 77, 85, 86 = 86

### Example

Use the SciPy mode() method to find the number that appears the most:

from scipy import stats  
  
speed = [99,86,87,88,111,86,103,87,94,78,77,85,86]  
  
x = stats.mode(speed)  
  
print(x)

# Machine Learning - Standard Deviation

Standard deviation is a number that describes how spread out the values are.

A low standard deviation means that most of the numbers are close to the mean (average) value.

A high standard deviation means that the values are spread out over a wider range.

Example: This time we have registered the speed of 7 cars:

speed = [86,87,88,86,87,85,86]

The standard deviation is:

0.9

Meaning that most of the values are within the range of 0.9 from the mean value, which is 86.4.

Let us do the same with a selection of numbers with a wider range:

speed = [32,111,138,28,59,77,97]

The standard deviation is:

37.85

Meaning that most of the values are within the range of 37.85 from the mean value, which is 77.4.

As you can see, a higher standard deviation indicates that the values are spread out over a wider range.

The NumPy module has a method to calculate the standard deviation:

### Example

Use the NumPy std() method to find the standard deviation:

import numpy  
  
speed = [86,87,88,86,87,85,86]  
  
x = numpy.std(speed)  
  
print(x)

### Example

import numpy  
  
speed = [32,111,138,28,59,77,97]  
  
x = numpy.std(speed)  
  
print(x)

## Variance

Variance is another number that indicates how spread out the values are.

In fact, if you take the square root of the variance, you get the standard deviation!

Or the other way around, if you multiply the standard deviation by itself, you get the variance!

To calculate the variance you have to do as follows:

1. Find the mean:

(32+111+138+28+59+77+97) / 7 = 77.4

2. For each value: find the difference from the mean:

 32 - 77.4 = -45.4  
111 - 77.4 =  33.6  
138 - 77.4 =  60.6  
 28 - 77.4 = -49.4  
 59 - 77.4 = -18.4  
 77 - 77.4 = - 0.4  
 97 - 77.4 =  19.6

3. For each difference: find the square value:

(-45.4)2 = 2061.16  
 (33.6)2 = 1128.96  
 (60.6)2 = 3672.36  
(-49.4)2 = 2440.36  
(-18.4)2 =  338.56  
(- 0.4)2 =    0.16  
 (19.6)2 =  384.16

4. The variance is the average number of these squared differences:

(2061.16+1128.96+3672.36+2440.36+338.56+0.16+384.16) / 7 = 1432.2

Luckily, NumPy has a method to calculate the variance:

### Example

Use the NumPy var() method to find the variance:

import numpy  
  
speed = [32,111,138,28,59,77,97]  
  
x = numpy.var(speed)  
  
print(x)

## Standard Deviation

As we have learned, the formula to find the standard deviation is the square root of the variance:

√1432.25 = 37.85

Or, as in the example from before, use the NumPy to calculate the standard deviation:

### Example

Use the NumPy std() method to find the standard deviation:

import numpy  
  
speed = [32,111,138,28,59,77,97]  
  
x = numpy.std(speed)  
  
print(x)

## Symbols

Standard Deviation is often represented by the symbol Sigma: σ

Variance is often represented by the symbol Sigma Square: σ2

# Machine Learning - Percentiles

Percentiles are used in statistics to give you a number that describes the value that a given percent of the values are lower than.

**Example:** Let's say we have an array of the ages of all the people that lives in a street.

ages = [5,31,43,48,50,41,7,11,15,39,80,82,32,2,8,6,25,36,27,61,31]

What is the 75. percentile? The answer is 43, meaning that 75% of the people are 43 or younger.

The NumPy module has a method for finding the specified percentile:

### Example

Use the NumPy percentile() method to find the percentiles:

import numpy  
  
ages = [5,31,43,48,50,41,7,11,15,39,80,82,32,2,8,6,25,36,27,61,31]  
  
x = numpy.percentile(ages, 75)  
  
print(x)

### Example

What is the age that 90% of the people are younger than?

import numpy  
  
ages = [5,31,43,48,50,41,7,11,15,39,80,82,32,2,8,6,25,36,27,61,31]  
  
x = numpy.percentile(ages, 90)  
  
print(x)

# Machine Learning - Data Distribution

Earlier in this tutorial we have worked with very small amounts of data in our examples, just to understand the different concepts.

In the real world, the data sets are much bigger, but it can be difficult to gather real world data, at least at an early stage of a project.

### How Can we Get Big Data Sets?

To create big data sets for testing, we use the Python module NumPy, which comes with a number of methods to create random data sets, of any size.

### Example

Create an array containing 250 random floats between 0 and 5:

import numpy  
  
x = numpy.random.uniform(0.0, 5.0, 250)  
  
print(x)

## Histogram

To visualize the data set we can draw a histogram with the data we collected.

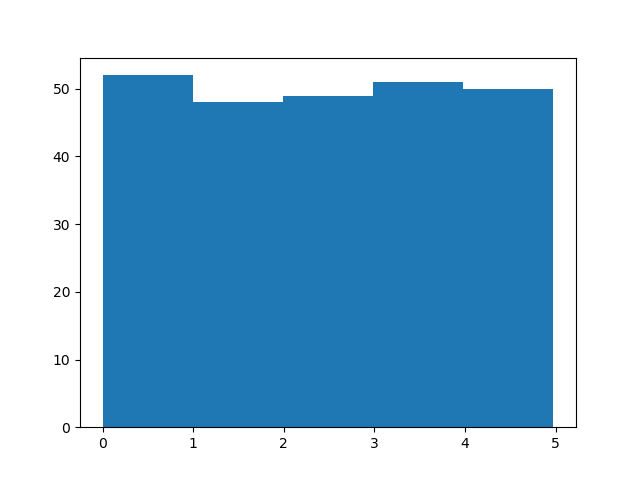
We will use the Python module Matplotlib to draw a histogram.

### Example

Draw a histogram:

import numpy  
import matplotlib.pyplot as plt  
  
x = numpy.random.uniform(0.0, 5.0, 250)  
  
plt.hist(x, 5)  
plt.show()

### Result:



### Histogram Explained

We use the array from the example above to draw a histogram with 5 bars.

The first bar represents how many values in the array are between 0 and 1.

The second bar represents how many values are between 1 and 2.

Etc.

Which gives us this result:

* 52 values are between 0 and 1
* 48 values are between 1 and 2
* 49 values are between 2 and 3
* 51 values are between 3 and 4
* 50 values are between 4 and 5

**Note:** The array values are random numbers and will not show the exact same result on your computer.

## Big Data Distributions

An array containing 250 values is not considered very big, but now you know how to create a random set of values, and by changing the parameters, you can create the data set as big as you want.

### Example

Create an array with 100000 random numbers, and display them using a histogram with 100 bars:

import numpy  
import matplotlib.pyplot as plt  
  
x = numpy.random.uniform(0.0, 5.0, 100000)  
  
plt.hist(x, 100)  
plt.show()

# Machine Learning - Normal Data Distribution

In the previous chapter we learned how to create a completely random array, of a given size, and between two given values.

In this chapter we will learn how to create an array where the values are concentrated around a given value.

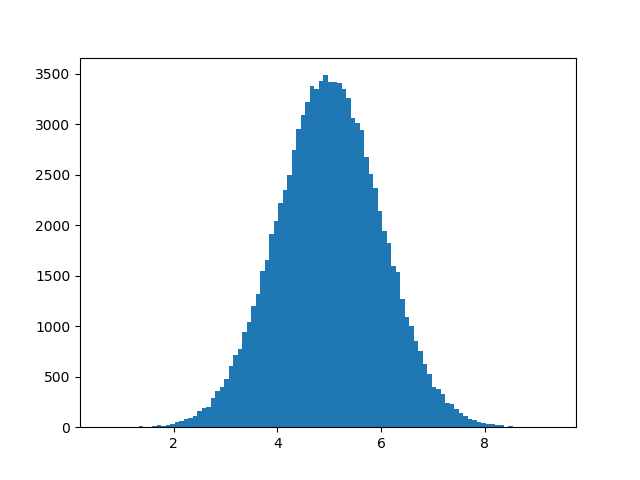
In probability theory this kind of data distribution is known as the normal data distribution, or the Gaussian data distribution, after the mathematician Carl Friedrich Gauss who came up with the formula of this data distribution.

### Example

A typical normal data distribution:

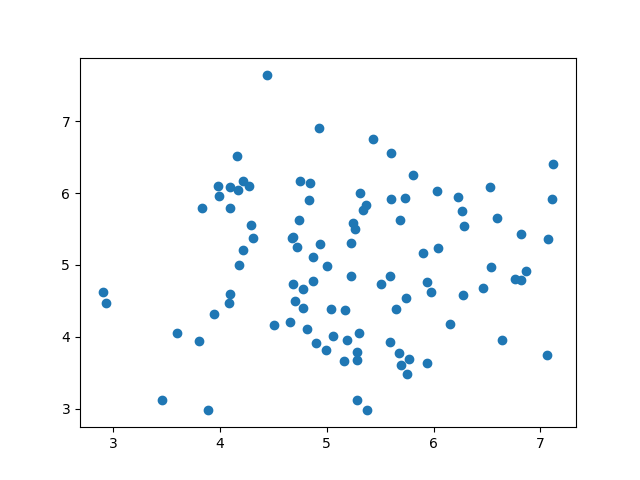
import numpy  
import matplotlib.pyplot as plt  
  
x = numpy.random.normal(5.0, 1.0, 100000)  
  
plt.hist(x, 100)  
plt.show()

### Result:



# Machine Learning - Scatter Plot

A scatter plot is a diagram where each value in the data set is represented by a dot.



The Matplotlib module has a method for drawing scatter plots, it needs two arrays of the same length, one for the values of the x-axis, and one for the values of the y-axis:

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]

y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

The x array represents the age of each car.

The y array represents the speed of each car.

### Example

Use the scatter() method to draw a scatter plot diagram:

import matplotlib.pyplot as plt  
  
x = [5,7,8,7,2,17,2,9,4,11,12,9,6]  
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]  
  
plt.scatter(x, y)  
plt.show()

### Result:



### Scatter Plot Explained

The x-axis represents ages, and the y-axis represents speeds.

What we can read from the diagram is that the two fastest cars were both 2 years old, and the slowest car was 12 years old.

**Note:** It seems that the newer the car, the faster it drives, but that could be a coincidence, after all we only registered 13 cars.

## Random Data Distributions

In Machine Learning the data sets can contain thousands-, or even millions, of values.

You might not have real world data when you are testing an algorithm, you might have to use randomly generated values.

As we have learned in the previous chapter, the NumPy module can help us with that!

Let us create two arrays that are both filled with 1000 random numbers from a normal data distribution.

The first array will have the mean set to 5.0 with a standard deviation of 1.0.

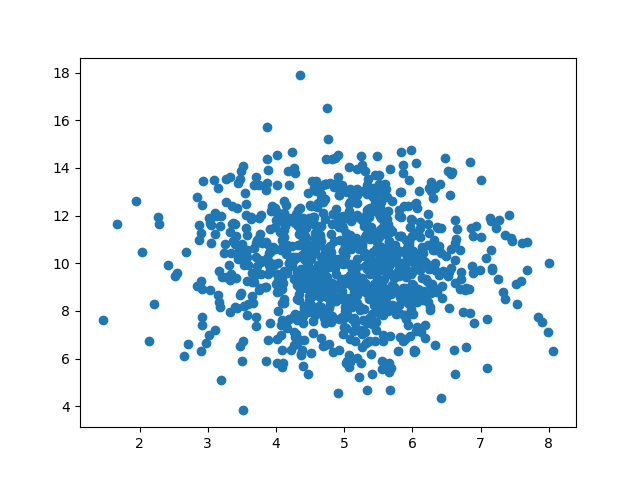
The second array will have the mean set to 10.0 with a standard deviation of 2.0:

### Example

A scatter plot with 1000 dots:

import numpy  
import matplotlib.pyplot as plt  
  
x = numpy.random.normal(5.0, 1.0, 1000)  
y = numpy.random.normal(10.0, 2.0, 1000)  
  
plt.scatter(x, y)  
plt.show()

### Result:



### Scatter Plot Explained

We can see that the dots are concentrated around the value 5 on the x-axis, and 10 on the y-axis.

We can also see that the spread is wider on the y-axis than on the x-axis.