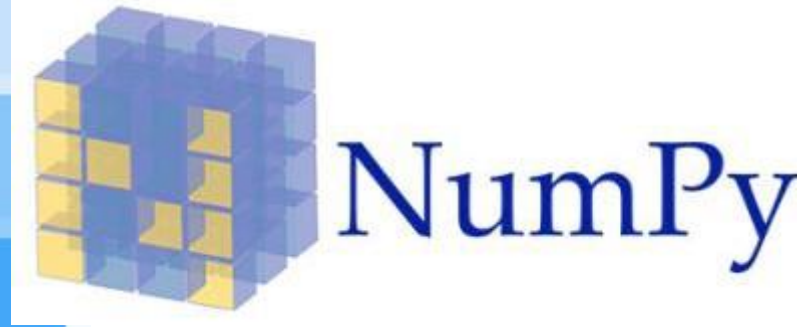
Python-Numpy

Dr. Sarwan Singh

NIELIT Chandigarh



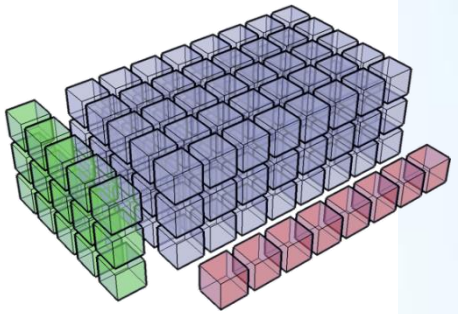
NumPy - Numerical **Python**,
library consisting of multidimensional array objects and a collection of routines for processing those arrays.

Agenda

- Mean, standard deviation, variance
- Processing data - Reading from csv files
- Sorting

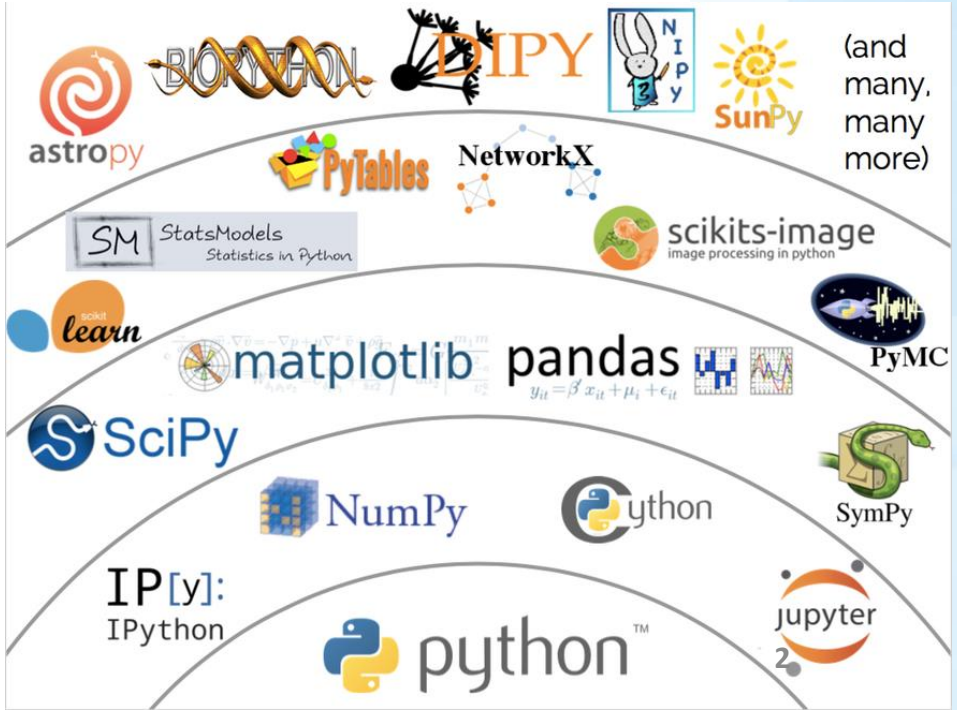
NumPy

Numerical Python



References

- Websites – google.com, statology.org, nlm.nih.gov, cuemath.com, datasciencestunt.com, medium.com, w3schools.com



Processing data using Numpy

find the meaning from DataSet

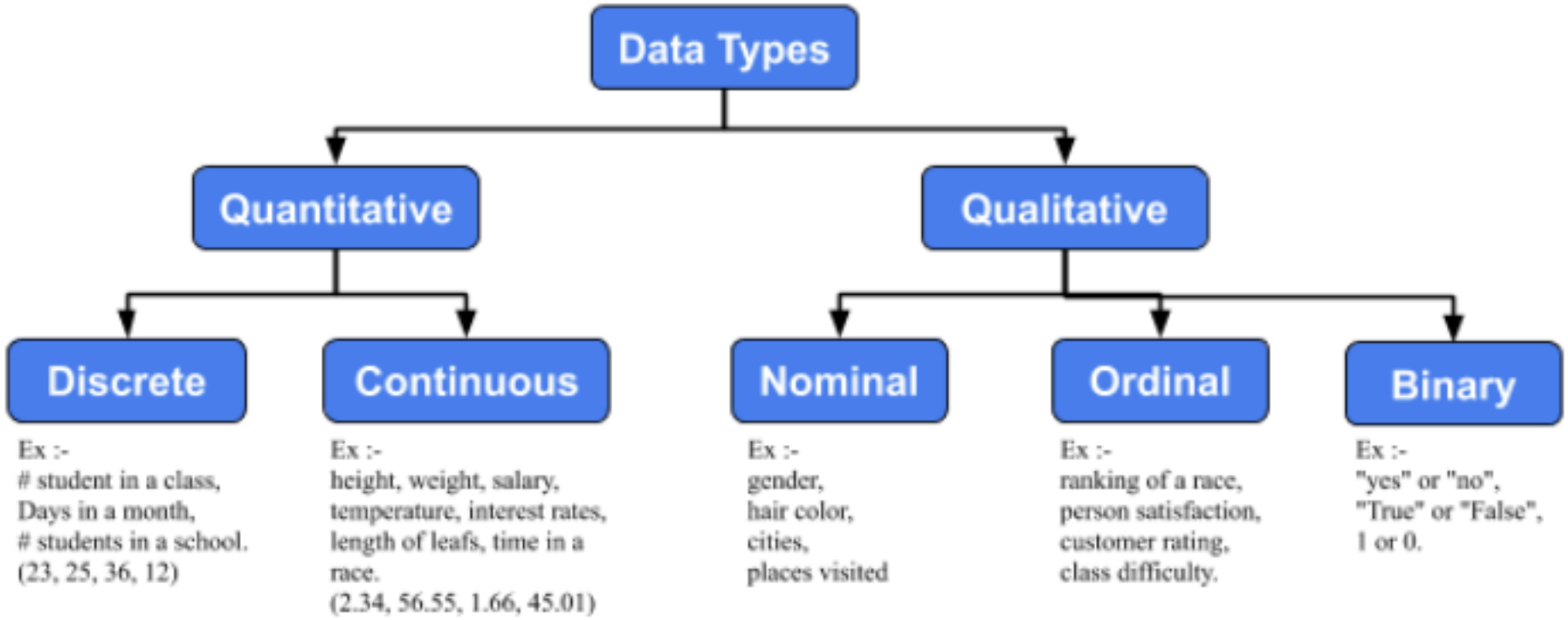
Carname	Color	Age	Speed	AutoPass
BMW	red	5	99	Y
Volvo	black	7	86	Y
VW	gray	8	87	N
VW	white	7	88	Y
Ford	white	2	111	Y
VW	white	17	86	Y
Tesla	red	2	103	Y
BMW	black	9	87	Y
Volvo	gray	4	94	N
Ford	white	11	78	N
Toyota	gray	12	77	N
VW	white	9	85	N
Toyota	blue	6	86	Y

guess

- The average value of car
- The highest value ,lowest value
- most popular color
- the oldest car is _____ years
- *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!

Data Types

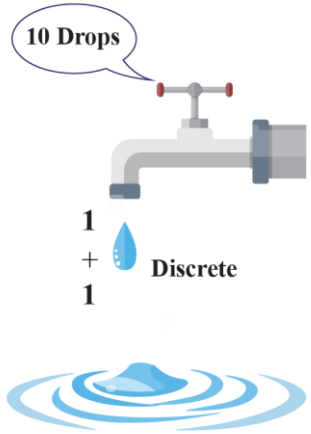
- To analyze data, it is important to understand the type of data
- The three main categories of data can be :
  - **Numerical**
  - **Categorical**
  - **Ordinal**



Data Types

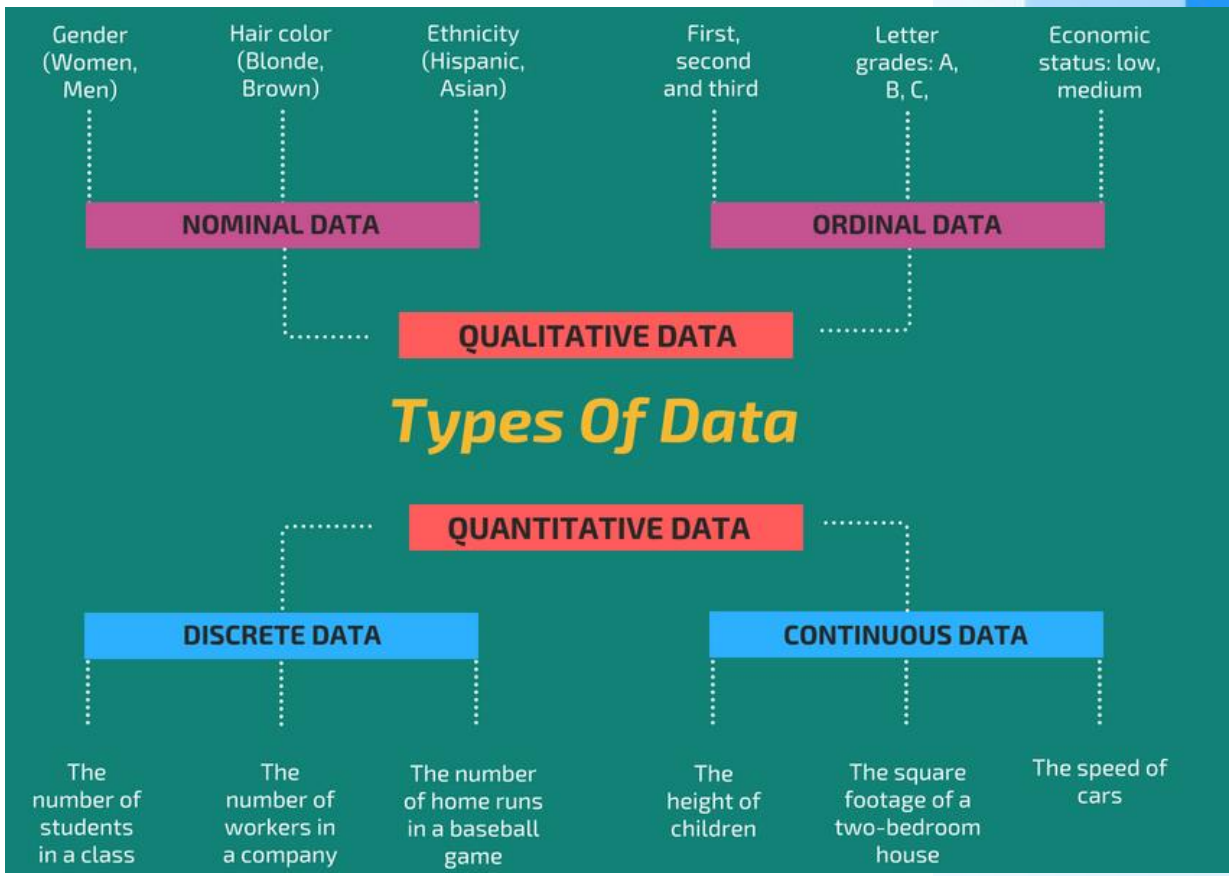
Numerical data are numbers, and can be split into two numerical categories:

- 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



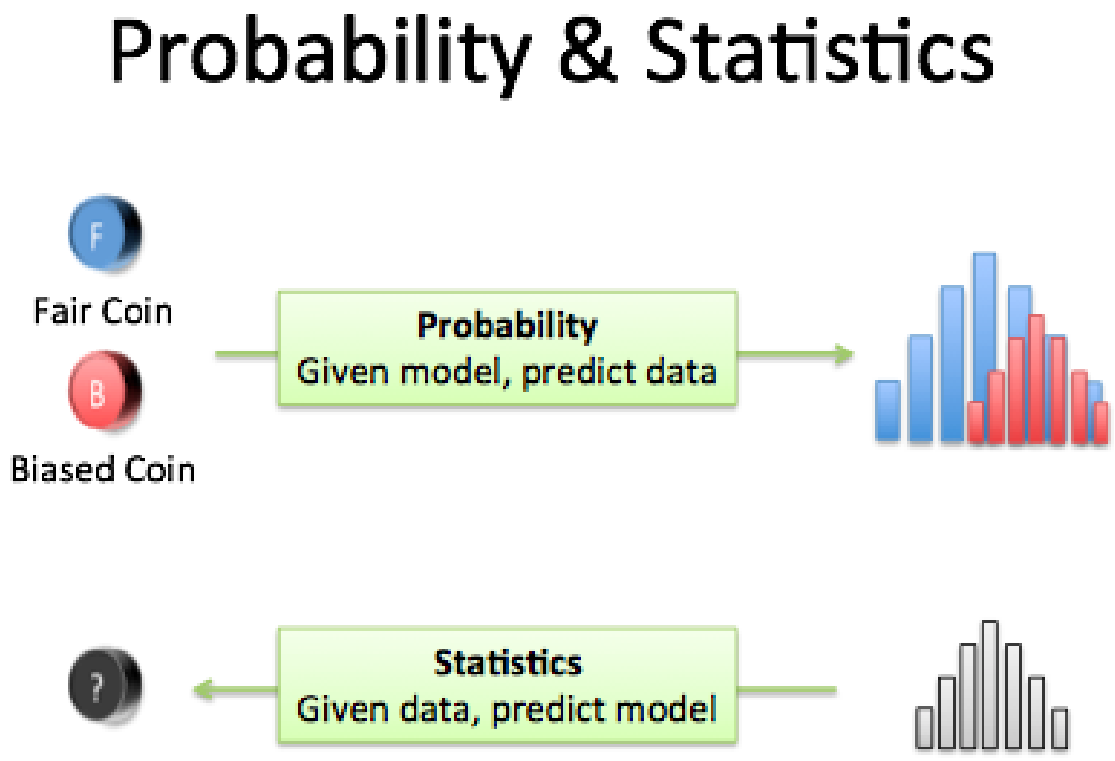
Data Types

- **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.



Probability and Statistics

- In probability, the model is given and we need to predict the data.
- While in statistics we start with the data and predict the model.
- We look at probability and search from data distributions which closely match the data distribution that we have.
- Then we assume that the function or the model must be the same as the one we looked into in probability theory.



Mean, Median, Mode, Percentile

- **Mean** - The average value
- **Median** - The mid point value
- **Mode** - The most common value
- Percentile are used in statistics to give you a number that describes the value that a given percent of the values are lower than.
- 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]

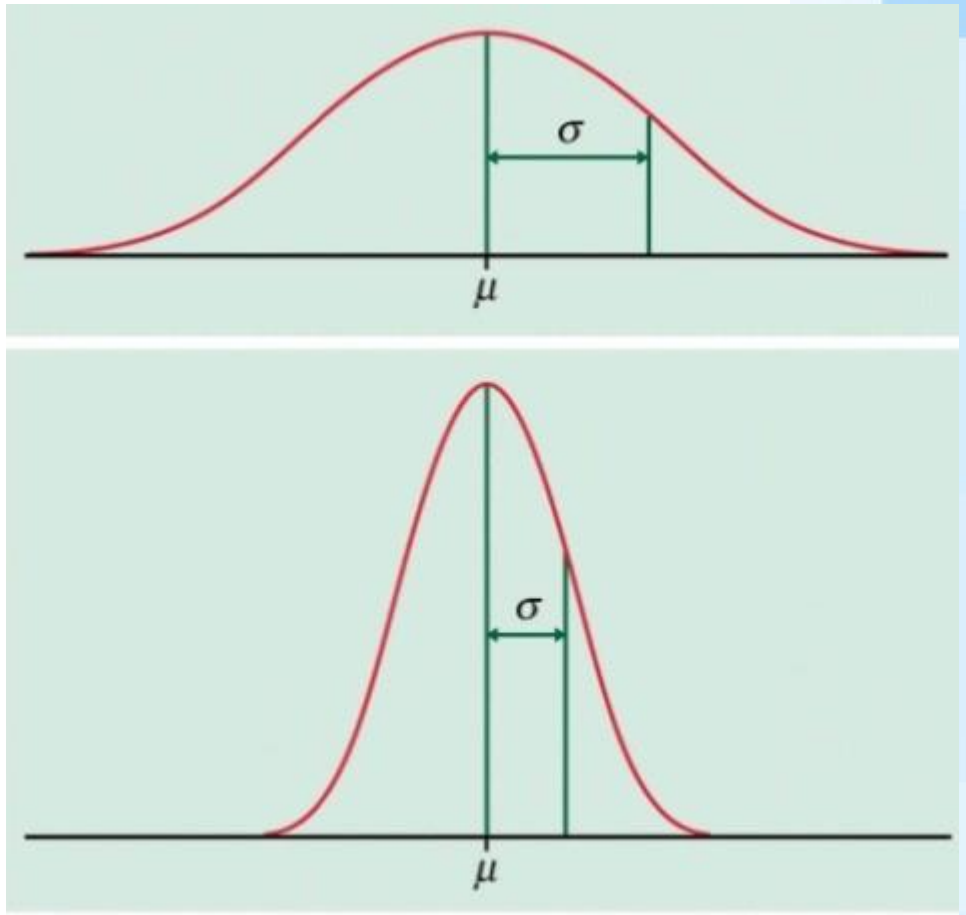
What is 75. percentile?

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

```
x = numpy.percentile (ages, 75)
print(x)
```

Standard Deviation

- A standard deviation (or σ) is **a measure of how dispersed the data is in relation to the mean**.
- Low standard deviation means data are clustered around the mean, and high standard deviation indicates data are more spread out.



Standard Deviation

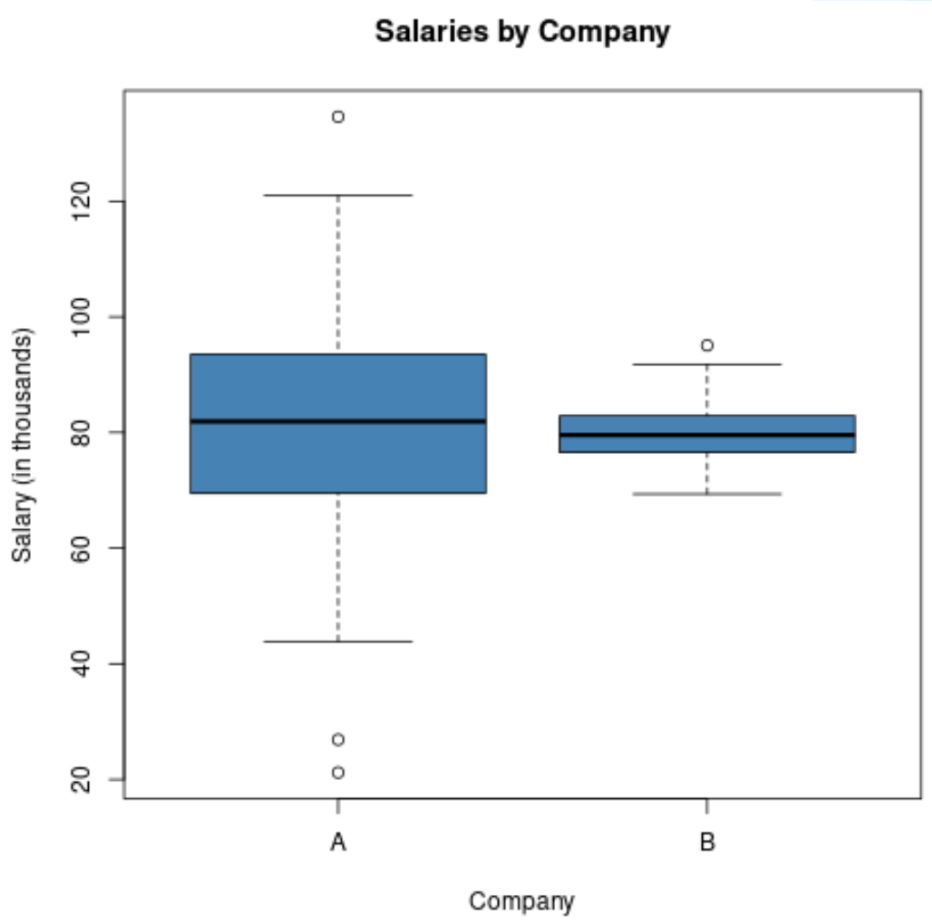
Whenever dataset is analyzed, the following metrics are the most interesting things to be searched:

- **The center of the dataset.** The most common way to measure the “center” is with the mean and the median.
- **The spread of values in the dataset.** The most common way to measure spread is with the standard deviation.

Standard Deviation -**Distribution of Salaries**

- Suppose the mean salary at company A is $80,000 and the standard deviation is $20,000. Since the standard deviation is so large, there's no guarantee that you will get paid close to $80,000 per year if you work at this company since there's such a variation in salaries.
- Conversely, suppose the mean salary at company B is also $80,000 but the standard deviation is only $4,000. Since this standard deviation is so small, you can be sure that you'll get paid close to $80,000 because there's very little variation in salaries.

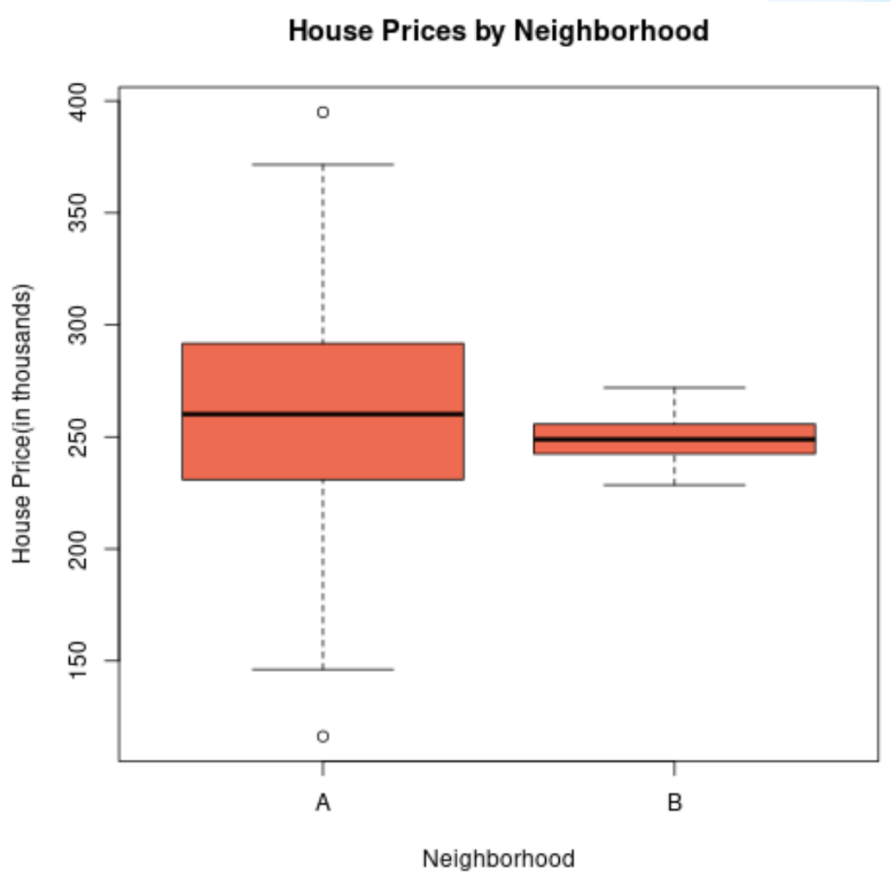
boxplot to visualize the distribution of salaries at these two companies



Standard Deviation -**Distribution of House Prices**

- The mean house price in neighborhood A is $250,000 and the standard deviation is $50,000. Since the standard deviation is quite large, this means that some of the house prices will be far greater than $250,000 and some will be far less. If you look at a given house in this neighborhood, there's no guarantee that the price will be close to the mean.
- Conversely, suppose the mean house price in neighborhood B is also $250,000 but the standard deviation is only $10,000. Since this standard deviation is fairly small, you can be sure that any given house you look at in the neighborhood is likely to be close to this price.

boxplot to visualize the distribution of house prices in these two neighborhoods



Standard Deviation

- The length of the boxplot for neighborhood A is so much greater because the standard deviation of house prices is so much higher.
- In fact, house prices range from lower than $150k to higher than $400k for neighborhood A, while prices only range from about $230k to $270k for neighborhood B.
- By simply knowing the standard deviation of house prices in each neighborhood, one can know how much variation to expect in prices in each neighborhood

Calculate Mean, Variance, Standard Deviation

Find the mean, standard deviation and variance for the data: 6, 7,10, 12, 13, 4, 8, 12.

Solution:

Given data: 6, 7,10, 12, 13, 4, 8, 12

Finding Mean:

Mean = Sum of observations / Total number of observations.

Mean = (6+7+10+12+13+4+8+12)/8

Mean = 72/8

Mean = 9.

x_i
6
7
10
12
13
4
8
12

Calculate Mean, Variance, Standard Deviation

data: 6, 7,10, 12, 13, 4, 8, 12.

Mean : 9

Finding Variance: 74/8 = 9.25

Variance =

$$\frac{\sum(x_i-\bar{x})^2}{n}$$

Here, n=8

$$\sum(x_i-\bar{x})^2$$

x_i	$x_i-\bar{x}$	$(x_i-\bar{x})^2$
6	6 – 9 = -3	9
7	7 – 9 = -2	4
10	10 – 9 = 1	1
12	12 – 9 = 3	9
13	13 – 9 = 4	16
4	4 – 9 = -5	25
8	8 – 9 = -1	1
12	12 – 9 = 3	9
$\sum(x_i-\bar{x})^2$		74

Calculate Mean, Variance, Standard Deviation

data: 6, 7,10, 12, 13, 4, 8, 12.

Mean: 9

Variance: 9.25

variance is the square of standard deviation

standard deviation = √variance

Standard deviation = √(9.25) = 3.041

x_i	$x_i - \bar{x}$	$(x_i - \bar{x})^2$
6	6 − 9 = -3	9
7	7 − 9 = -2	4
10	10 − 9 = 1	1
12	12 − 9 = 3	9
13	13 − 9 = 4	16
4	4 − 9 = -5	25
8	8 − 9 = -1	1
12	12 − 9 = 3	9
$\sum(x_i - \bar{x})^2$		74

Variance vs Standard Deviation

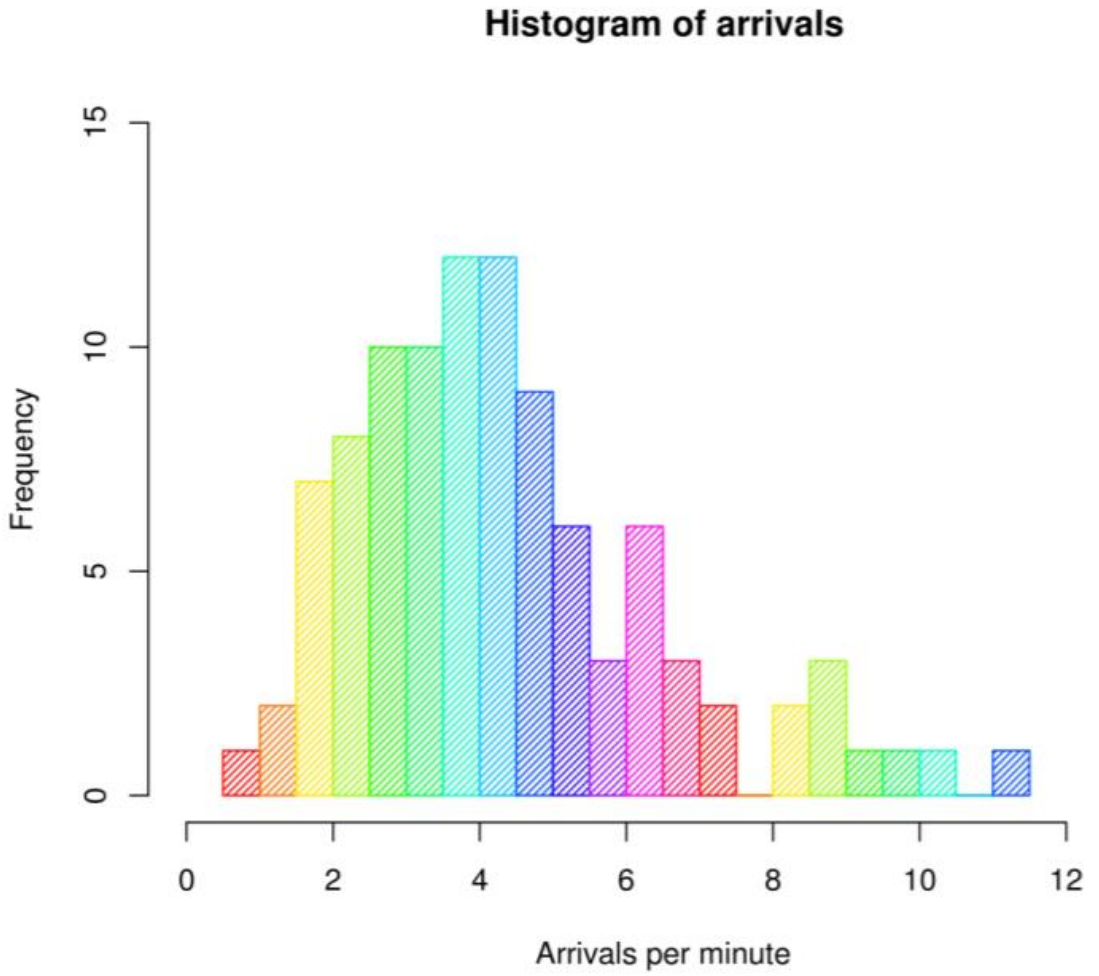
- Both measures exhibit variability in distribution
- Variance is a measure of how data points vary from the mean, whereas standard deviation is the measure of the distribution of statistical data.
- The basic difference between variance and the standard deviation is in their units.
- The standard deviation is represented in the same units as the mean of data, while the variance is represented in squared units.
- Variance is equal to the average squared deviations from the mean, while standard deviation is the number's square root. Also, the standard deviation is a square root of variance.

Example in context to AI/ML

- capture the data of arrival of flights per minute at an airport over a period of time.
- plot a histogram.
- From the histogram, find out the most common, most frequent, less frequent number, etc.

Variance measures how "spread-out" the data is.

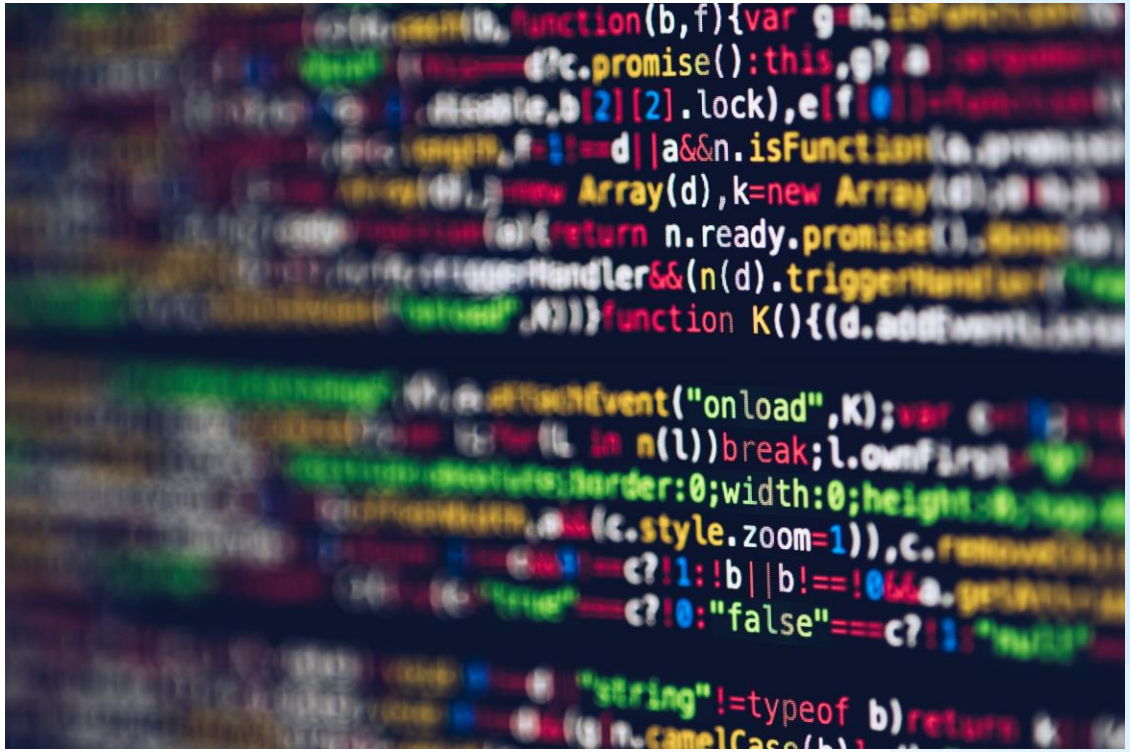
Variance ($\sigma 2$) is simply the average of the squared differences from the mean.



Example in context to AI/ML

Standard Deviation

- usually used as a way to identify outliers.
- Data points that lie more than one standard deviation from the mean can be considered unusual.



Reading data from csv file

- Using – genfromtxt , csv.reader

	A	B	C
	order	name	height(cm)
1	1	George Washington	189
2	2	John Adams	170
3	3	Thomas Jefferson	189
4	4	James Madison	163
5	5	James Monroe	183
6	6	John Quincy Adams	171
7	7	Andrew Jackson	185
8	8	Martin Van Buren	168
9	9	William Henry Harrison	173
10	10	John Tyler	183
11	11	James K. Polk	173
12	12	Zachary Taylor	173
13	13	Millard Fillmore	175
14	14	Franklin Pierce	178
15	15	James Buchanan	183
16	16	Abraham Lincoln	193
17	17	Andrew Johnson	178
18	18	Ulysses S. Grant	173
19	19	Rutherford B. Hayes	174
20	20	James A. Garfield	183
21	21	Chester A. Arthur	183

```
from numpy import genfromtxt
my_data = genfromtxt('jupyter-demo/president_heights.csv', delimiter=',',skip_header=1)
heights = np.array(my_data[:,2])
print(heights)
```

```
[ 189.  170.  189.  163.  183.  171.  185.  168.  173.  183.  173.  173.
  175.  178.  183.  193.  178.  173.  174.  183.  183.  168.  170.  178.
  182.  180.  183.  178.  182.  188.  175.  179.  183.  193.  182.  183.
  177.  185.  188.  188.  182.  185.]
```

```
import csv
with open('jupyter-demo/president_heights.csv', 'r') as f:
    datalist=list (csv.reader(f, delimiter=','))

print(datalist[:5])
```

```
[['order', 'name', 'height(cm)'], ['1', 'George Washington', '189'], ['2', 'John Adams', '170'], ['3',
'Thomas Jefferson', '189'], ['4', 'James Madison', '163']]
```

Genfromtxt vs csv.reader

```
from numpy import genfromtxt
genfromtxt(fname = dest_file, dtype = (<whatever options>))
```

versus

```
import csv
import numpy as np
with open(dest_file,'r') as dest_f:
    data_iter = csv.reader(dest_f,
                           delimiter = delimiter,
                           quotechar = '"')
    data = [data for data in data_iter]
data_array = np.asarray(data, dtype = <whatever options>)
```

on 4.6 million rows with about 70 columns and found that the numpy path took 2 min 16s and the csv-list comprehension method took 13s.

exercise

Calculate following using data from presidents_heights.csv

- Mean height
- Standard deviation
- Minimum height
- Maximum height
- 25th percentile
- Median
- 75th percentile

Using seattle2014.csv file :

- extract rainfall inches
- Max rainfall

Sort, Search & Counting Functions

- Various sorting functions are available in NumPy having different sorting algorithms.
- Every algorithm is characterized by the speed of execution, worst case performance, the workspace required and the stability.

kind	speed	worst case	work space	stable
‘quicksort’	1	O(n^2)	0	no
‘mergesort’	2	O(n*log(n))	~n/2	yes
‘heapsort’	3	O(n*log(n))	0	no

Sorting

- numpy.sort (array , axis, kind, order)
  - array- to be sorted
  - axis- axis of array to be sorted. If none, the array is flattened, sorting on the last axis
  - kind - Default is quicksort
  - order - If the array contains fields, the order of fields to be sorted

```
A
```

```
array([[0, 1, 2],
       [3, 4, 3],
       [6, 7, 8],
       [9, 8, 9]])
```

```
A.sort()   #sort array in place
```

```
A
```

```
array([[0, 1, 2],
       [3, 3, 4],
       [6, 7, 8],
       [8, 9, 9]])
```

```
np.sort(A) #sort and create copy
```

```
array([[0, 1, 2],
       [3, 3, 4],
       [6, 7, 8],
       [8, 9, 9]])
```

Sorting

- np.sort(a, order = 'name')

```
import numpy as np
data = np.zeros(4, dtype={'names':('name', 'age', 'weight'), 'formats':('U10', 'i4', 'f8')})
print(data.dtype)
```

```
[('name', '<U10'), ('age', '<i4'), ('weight', '<f8')]
```

```
data['name'] =  ['Sumit', 'Baljeet', 'Akbar', 'Neeru']
data['age'] = [23,67,35,44]
data['weight'] = [23.8,67.8,35.8,44.8]
```

```
data[0]
```

```
('Sumit', 23,  23.8)
```

```
data[-1]['name']
```

```
'Neeru'
```

```
data[data['age'] < 36]['name']
```

```
array(['Sumit', 'Akbar'],
      dtype='<U10')
```

```
np.sort(data, order = 'name')
```

```
array([('Akbar', 35,  35.8), ('Baljeet', 67,  67.8), ('Neeru', 44,  44.8),
       ('Sumit', 23,  23.8)],
      dtype=[('name', '<U10'), ('age', '<i4'), ('weight', '<f8')])
```

```
np.sort(data, order = 'name')
```

```
array([('Akbar', 35,  35.8), ('Baljeet', 67,  67.8), ('Neeru', 44,
       ('Sumit', 23,  23.8)],
      dtype=[('name', '<U10'), ('age', '<i4'), ('weight', '<f8')])
```

```
# use != or negate the condition using ~
data[~(data['name']=='Baljeet')]['age']
```

```
array([23, 35, 44])
```

```
data[(data['name']=='Baljeet')]['age']
```

```
array([67])
```

Jupyter Notebook Link

- https://colab.research.google.com/drive/1UVDRN2LLHqtiVVBhpV08FN1adSM-brir?usp=sharing

Github Repository Link

- https://github.com/sarwansingh/Python/tree/master/ORD