

CS217: Computer Programming Language

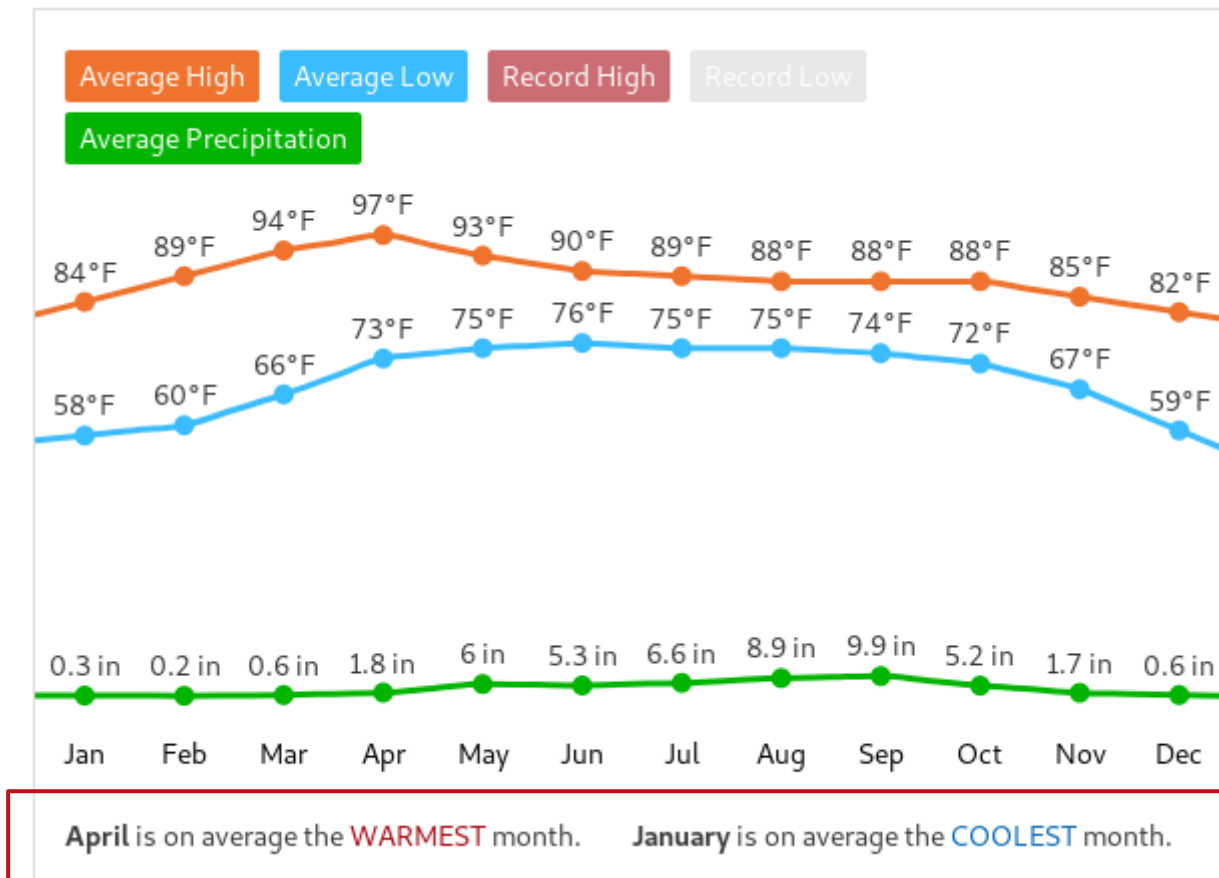
Computational Thinking and friends
Instructor: Jakramate Bootkrajang

What is Data Science ?

- The scientific process of extracting **information** or **knowledge** from data
- The art of presenting the results
- Interdisciplinary field involves
 - Mathematics
 - Statistics
 - Computer science
 - And business, finance, biology, social science ...






























Types of knowledge [1]

- Descriptive statistics



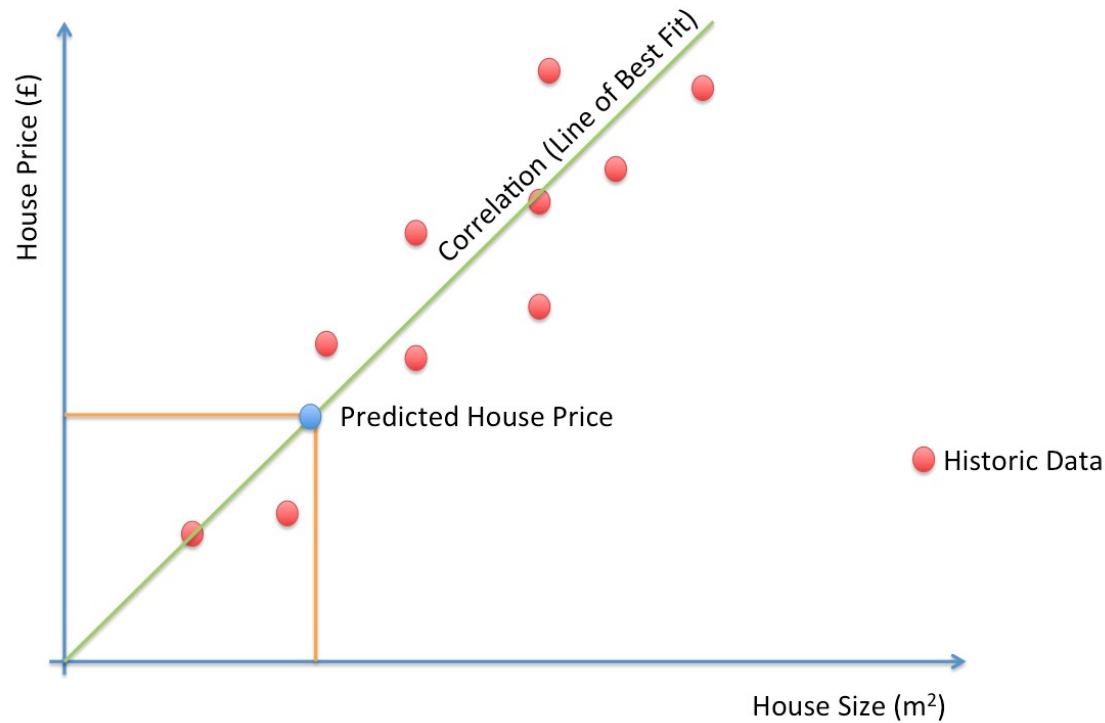
Types of knowledge [2]

- Forecasting model

13  88° 63°	14   87° 65°	15   86° 66°	16   84° 66°	17   83° 64°	18   84° 63°	19   83° 61°
20   84° 61°	21   84° 61°	22  Avg 85° 58°	23  Avg 85° 58°	24  Avg 85° 58°	25  Avg 85° 58°	26  Avg 85° 58°
27  Avg 85° 58°	28  Avg 85° 58°	29  Avg 86° 58°	30  Avg 86° 58°	31  Avg 86° 59°	1  Avg 84° 58°	2  Avg 84° 58°

Types of knowledge [3]

- Regressor



Real world example



ประมาณราคาบ้านหรือคอนโดของคุณ ด้วย Bestimate



Bestimate คืออะไร ?

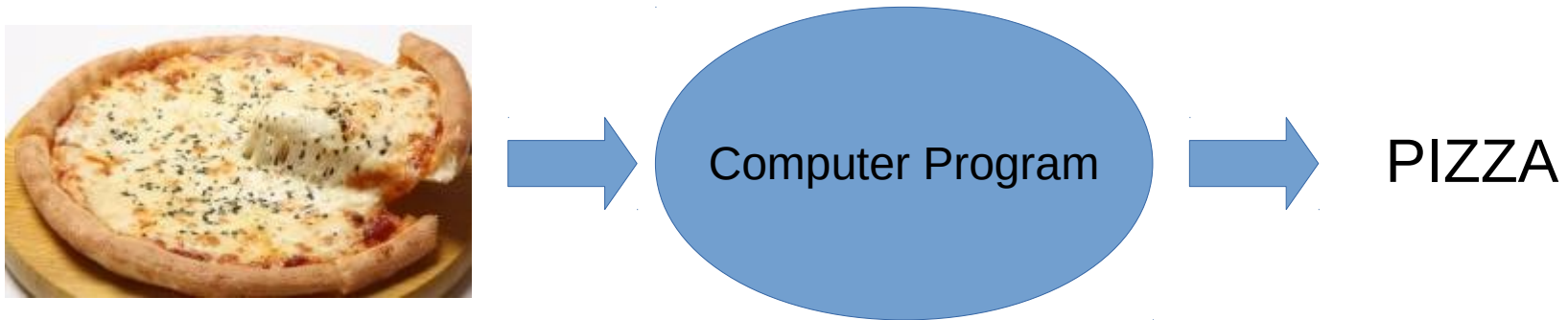
Bestimate (อ่านว่า "เบส-ตี-เมท") คือ โปรแกรมประมาณมูลค่าอสังหาริมทรัพย์ ซึ่งได้รับการพัฒนาโดย บริษัทบาเนีย (ประเทศไทย) จำกัด โดยให้บริการบนเว็บไซต์ Baania.com ทั้งนี้การประมาณมูลค่าดังกล่าว มิใช่การประเมินราคา แต่เป็นการประมาณมูลค่าโดยใช้ข้อมูลอสังหาริมทรัพย์จากการเก็บรวบรวมโดยทีมงาน Baania และข้อมูลทุกข้อมูมิจากแหล่งข้อมูลที่ได้มาตรฐานและมีความน่าเชื่อถือสูง

เริ่มการประมาณราคา ฟรี!

ค้นหาบ้านและคอนโด **ตามจังหวัด**

Types of knowledge [4]

- Classification rule



Text	Sentiment
"Don't stay here if you can avoid it. Everything smells like old cigarettes."	Negative
"Friendly service. Superior room! Loved the high ceiling."	Positive
...	...

Types of knowledge [5]

- Association rule
 - Given huge transactional data
 - Find items that are frequently occur together

<i>Customers</i>	<i>Transactions</i>
1	<i>milk, bread</i>
2	<i>bread, butter</i>
3	<i>beer</i>
4	<i>milk, bread, butter</i>
5	<i>bread</i>
6	<i>milk, bread, butter</i>



IF
people buy MILK

THEN

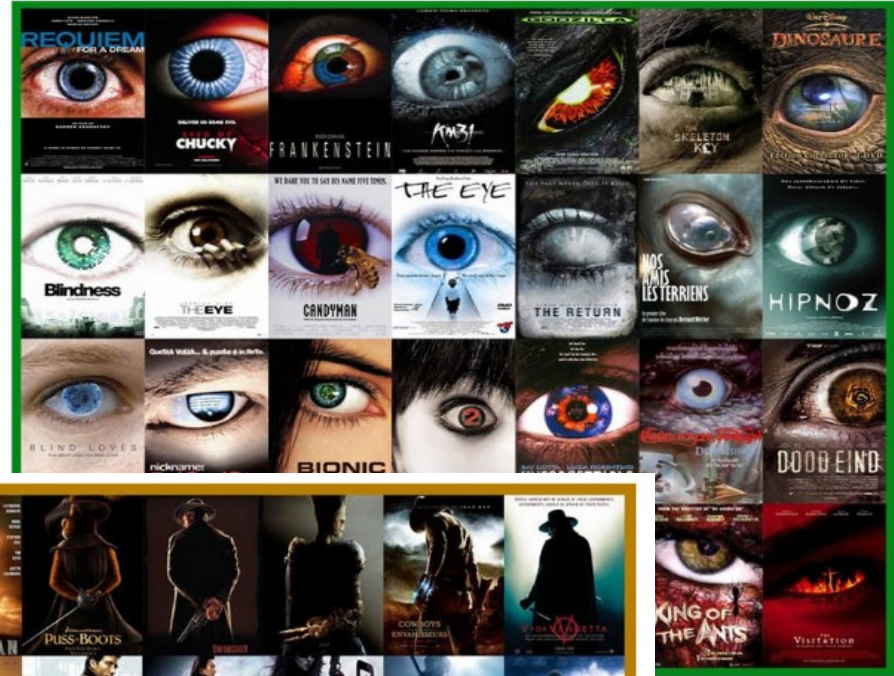
they are likely to buy BREAD

Types of knowledge [6]

- Cluster analysis

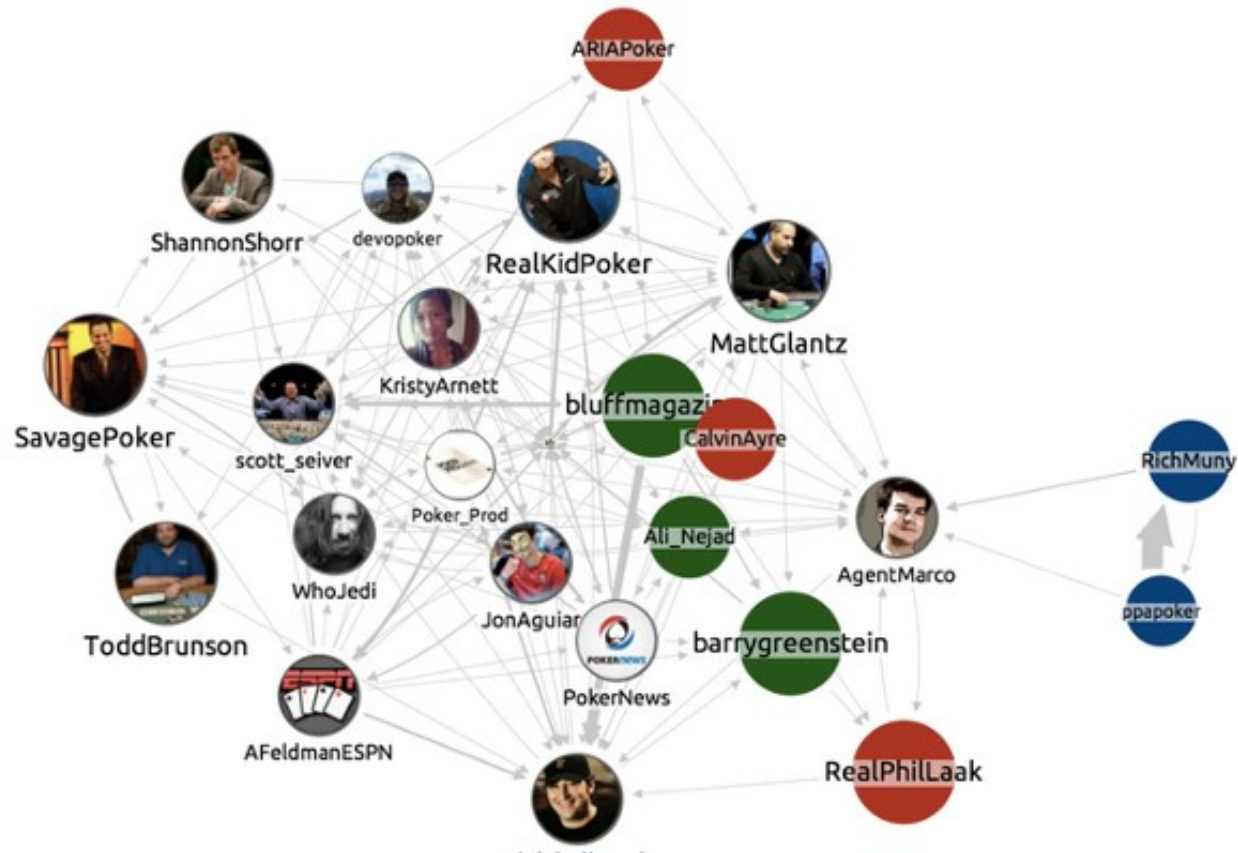


Credit <http://christophecourtois.blogspot.com/>



Types of knowledge [6]

- Structure analysis






To do data science you need

- 1) background knowledge of the problem
- 2) mathematical or statistical skill/ideas to solve the problem at the abstraction level
- 3) programming skill to implement the ideas (efficiently)
 - what we'll practice in this class -



Programming for Data Science

- To program is
 - to tell a computer what to do using a programming language
- Programming for data science is no different from programming for XYZ
 - The basics are all the same
- Also true for programming language
 - Learning the second language takes less effort



To become a capable programmer

- You need
 - A computational mindset
 - Or computational thinking
 - A lot of practices
 - Writing code
 - Reading code



Computational Thinking

- To formulate real-life problem as computational problem
- To think about possible solutions
- To express solutions in ways that a computer could execute
 - Express clearly and accurately
 - A set of steps to solve something is known as **an algorithm**



Computational problems

- Computational problems can be categorised into
 - Calculation problems
 - Decision problems
 - Search/Sort problems
 - Counting problems
 - Optimisation problems

Calculation problems

- The easiest to program
- Usually involves evaluating a mathematical expression

$$p(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

with $\mu = 0, \sigma = 1,$

Example solution

```
1 import math # import some mathematical functions
2
3 x = 2      # define x
4 mu = 0     # define mu
5 sigma = 1  # define sigma
6
7 llh = 1/math.sqrt(2*3.14*sigma**2) * math.exp(-(x-mu)**2 / (2*sigma**2))
8
9 print(llh) # display the result
```

0.054004657278425724

- one needs to be careful when working with mathematical formulas
- Computer cannot represent all values on the real line.
- Techniques for computing maths expression with minimal errors are studied in the field of **numerical methods**



Decision problem

- Problem that requires decision making
- Usually the output will either be **Yes** or **No**
- May exist on its own or is part of more complicated problem

Example

- A python function that decides whether to buy a stock market or not

```
1 def buyOrNot(price):      # defining a function
2     if price < 50:        # actual decision making process
3         return "Buy"
4     else:
5         return "Do not buy"
```

```
1 todayPrice = 60
2
3 answer = buyOrNot(todayPrice)
4
5 print(answer)
```

Do not buy



Search/Sort problem

- Searching the problem of finding the solution (often there is only one solution) in a solution space efficiently
- Sorting is the problem of re-arranging the items in our collection of items into some order.



Example of searching

- What is the largest number in set $\{2, 9, 10, 40, 25, 30, 99, 86\}$?
- How would you solve this ?

Possible solution

```
1 numbers = [2,9,10,40,25,30,99,86]      # array
2 largest = -1    # start with something small
3
4 for num in numbers:    # loop to see each number in the array
5     if num > largest:  # if we found the new largest
6         largest = num | # store it
7
8 print(largest)
```

99

Example of sorting

```
In [27]: 1 numbers = [2,9,10,40,25,30,99,86] # array  
        2  
        3 sorted(numbers) # someone was kindly enough to write sorting function for us
```

Out[27]: [2, 9, 10, 25, 30, 40, 86, 99]

Counting problem

- To count the number of solutions in the solution space, efficiently
- For example one might ask you to count how many even number in a set
 $\{2, 9, 10, 40, 25, 30, 99, 86\}$?
- How would you solve this ?

Possible solution

```
1 numbers = [2, 9, 10, 40, 25, 30, 99, 86]      # array
2 evenNumber = 0
3
4 for num in numbers:
5     if num % 2 == 0:
6         evenNumber = evenNumber + 1
7
8 print(evenNumber)
```

5



Optimisation problem

- Concerns with finding the best possible solution from many possible solutions
- Example
 - Linear programming
 - (stock) portfolio management problem



Problem solving techniques

[1]

- Abstraction
 - solving the problem in a model of the system before applying it to the real system
- Analogy
 - using a solution that solves an analogous problem

Problem solving techniques

[2]

- Divide and Conquer
 - breaking down a large, complex problem into smaller, solvable problems
 - Solve the small problems
 - Combine the solutions to get the solution to the original problem
- Proof
 - try to prove that the problem cannot be solved
 - Or show that the problem is equivalent to solving something easier

Problem solving techniques

[3]

- Research
 - employing existing ideas or adapting existing solutions
- Reduction
 - Transform problem into something we know how to solve
- Trial-and-error (not recommended)
 - Time consuming but worth trying if everything above fails



Problem solving with programming

- First, you have to understand the problem.
- After understanding, then make a plan.
- Carry out the plan.
- Look back on your work. How could it be better?



Step 1: Understand the Problem

- Make sure you
 - Fully understand the problem
 - Can describe the problem in your own words
 - Know what are the inputs
 - Know what are the final output




Step 2: Devise a Plan

- Using aforementioned problem solving techniques
 - Divide-and-conquer, analogy, reduction, etc...
- Immediate concern: to get correct solution
- Future concerns (after more CS courses)
 - Security, efficiency, speed, reliability

Step 3: Carry out the Plan

- Translate your solution into code
- Simulate test cases
 - A **test case** is a correct input-output pair which is used to test the correctness of the solution
- Test your solution (program)



Step 4: Examine and Review

- Review/discuss your solution with others
- Keep the solution in good place for later usage (analogy-typed problem solving)