Algorithm Design
An algorithm can be written out in pseudo code

Algorithm 2: Bernoulli Trials Attribute Selection

Input: Classifier $L$
Attributes $A$

Output: Attribute subset $A^s$

1. $r_{\min} = 0$, $r_{\max} = |A|$, $A^s = A$, $success = 0$, $fail = 0$
2. Compute tables: $T_{sum}(n, r)$ for $r_{\min} \leq r < r_{\max}$
3. $K_{opt}(n, r)$ for $r_{\min} \leq r < r_{\max}$
4. threshold $= \frac{1}{n} (\sum_{i=1}^{n} \epsilon(L, A))$

$\implies$ Start in states 1 or 2

6. while $r \leq |A^s|$ do

7. $k = K_{opt}(|A^s|, r)$

8. $A^k = \text{Select k attributes at random from } A^s \text{ with } \epsilon(A^s_i) < \text{threshold}$

9. $e = \epsilon(L, A^s)$

10. $e' = \epsilon(L, A^s - A^k)$

11. if $e' \leq e$ then

12. $A^s = A^s - A^k$

13. $success++$

14. $fail = 0$

else

16. $fail++$

17. $success = 0$

18. if $fail \geq N^{-}(A, A^s, k)$ then

19. $\implies$ Failed more than expected

20. if $(r_{\min} + 1 = r_{\max})$ then

21. $\implies$ Recover in case of state 1 or 3

22. $r_{\min} = 0$

23. $r_{\max} = |A^s|$
Then turned into source code

```javascript
for (var $docid in results) {
    var doc = results[$docid];
    var authorName = $(doc).find("author").text();
    // $("#arxiv-data").append(authorName);
    // alert(authorName);
    var authorAff = $(doc).find("author").text();

    var paperTitle = $(doc).find("title").text();
    var paperURL = $(doc).find("id").text();

    var paperAbstract = $(doc).find("summary").text();
    var paperUpdateTime = $(doc).find("update").text();

    // filter out metadata
    if (paperTitle.indexOf("ArXiv") === -1)
        codeAddress(authorName, authorAff, paperTitle, paperUpdateTime)
}
```
Which is then compiled to machine code:

```
080481c <main>:
  080481c: 55  push  %ebp
  080481d: 89 e5  mov  %esp,%ebp
  080481f: 83 e4 f0  and  $0xffffffff0,%esp
  0804822: 83 ec 50  sub  $0x50,%esp
  0804825: 8d 44 24 24  lea  0x24(%esp),%eax
  0804829: 89 04 24  mov  %eax,(%esp)
  080482c: e8 2b fa ff ff  call  804885c <pipe@plt>
  0804831: c7 44 24 08 00 00 00  movl  $0x0,0x8(%esp)
  0804838: 00  movl  $0x2,0x4(%esp)
  0804839: c7 44 24 04 02 00 00  movl  $0x13,%esp
  080483e: e8 4f f9 ff ff  call  804879c <socket@plt>
  0804844: 89 44 24 2c  mov  %eax,0x2c(%esp)
  080484d: c7 44 24 04 00 00 00  movl  $0x0,0x4(%esp)
  0804851: 00  movl  $0x804921a,%esp
  0804852: e8 67 f8 ff ff  call  80486cc <open@plt>
  0804855: 89 44 24 30  mov  %eax,0x30(%esp)
  0804858: 8b 44 24 24  mov  0x24(%esp),%eax
  080485b: 85 c0  test  %eax,%eax
  080485d: 78 18  js  8048e89 <main+0x6d>
  080485f: 8b 44 24 28  mov  0x28(%esp),%eax
  0804863: 85 c0  test  %eax,%eax
  0804865: 78 10  js  8048e89 <main+0x6d>
  0804867: 8b 44 24 2c  mov  0x2c(%esp),%eax
  080486a: 85 c0  test  %eax,%eax
  080486c: 78 08  js  8048e89 <main+0x6d>
  080486e: 8b 44 24 30  mov  0x30(%esp),%eax
  0804871: 85 c0  test  %eax,%eax
  0804873: 79 16  js  8048e89 <main+0x6d>
  0804875: c7 44 24 24 92 04 08  movl  $0x8049224,%esp
  080487c: e8 a7 f9 ff ff  call  804883c <puts@plt>
```
Problem Solving

● Algorithm: set of unambiguous instructions to solve a problem
  ○ Breaking down a problem into a set of sub-problems
  ○ Example: Toast some bread

● Without instructions – computers cannot do anything at all!
Algorithm design

- Analysis and specification
  - Analyze: Understand/define the problem
  - Specify: Specify particulars

- Algorithm development phase
  - Develop: Logical sequence of steps
  - Test: Follow outline, test cases

- Implementation phase
  - Code: The steps into a programming language
  - Test: Debug

- Maintenance phase
  - Use the program
  - Maintain: Correct errors, meet changing requirements
An example:

*Making a perfect piece of toast*

What do we need:

- a loaf of bread
- knife
- toaster
- plate
- butter
- cutting board
An example:

Making a perfect piece of toast

pseudo code
1. Move a *loaf of bread* on a *cutting board*
2. Cut a slice of bread with a *knife*
3. Move the slice of bread to the *toaster*
4. Turn *toaster* on
5. Wait for the *toaster* to finish
6. Move the toasted bread on a *plate*
7. Spread *butter* on the toast with *knife*
An example:

**Making a perfect piece of toast**

<table>
<thead>
<tr>
<th>Variables:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A = loaf of bread</td>
</tr>
<tr>
<td>K = knife</td>
</tr>
<tr>
<td>T = toaster</td>
</tr>
<tr>
<td>P = plate</td>
</tr>
<tr>
<td>B = butter</td>
</tr>
<tr>
<td>CB = cutting board</td>
</tr>
<tr>
<td>S = slice of bread</td>
</tr>
</tbody>
</table>

**pseudo code**

1. move A to CB
2. cut S with K
3. move S to T
4. IF( NOT ON(T))
5. turn on T
6. WHILE( NOT TOASTED(S))
7. wait for T
8. move S to P
9. spread B on S with K
Basic concepts

- **Instructions** – simple and unambiguous
- **Variables** – input and temporary
- **Subprocedures** – smaller tasks
- **Looping**: FOR each variable, WHILE
  - Act of repeating tasks
- **Conditional statements**: IF ELSE
  - Selectively execute instructions
Basic concepts

IF
Execute some statements based on the truth on a statement
Basic concepts

IF

if ( 5 > 0 ) {

print "the statement is true!"

}

Basic concepts

IF

if ( 5 > 9 ) {
    print "the statement is true!"
}

}
Basic concepts

IF

a = 5
if ( a > 9 ) {

print "the statement is true!"

}


Basic concepts

IF

a = 5
if ( a < 9 ) {

print "the statement is true!"

}
Basic concepts

**WHILE**

Execute statements while a condition is true
Basic concepts

**WHILE**

while ( true ) {

print "in the while loop!"

}

```
Basic concepts

**WHILE**

```java
a = 3
while ( a > 1 ) {
    print "in the while loop!"
}
```
Basic concepts

**WHILE**

```plaintext
a = 3
while ( a > 1 ) {
    print "in the while loop!"
    a--
}
```
Basic concepts

FOR
Execute statements while a condition is true as well as increment and initialize variables
Basic concepts

FOR LOOP

for ( var i=0; i < 10 ; i++ ) {
    print "loop" + i
}
Basic concepts

loop 0
loop 1
loop 2
.
.
.
loop 9
Basic concepts

**IF ELSE**

Execute some statements if a condition is true otherwise execute different statements.
Basic concepts

**IF ELSE**

\[
a = 5
\]
\[
\text{if ( } a < 9 \text{ ) }
\]
\[
\text{print "the statement is true!"}
\]
\[
\text{else}
\]
\[
\text{print "the statement is false!"
}\]
Basic concepts

IF
FOR
WHILE
IF ELSE