Grading Computer Programming Skills using Machine Learning

Shashank Srikant
Varun Aggarwal

Aspiring Minds
www.aspiringminds.com

We conduct standardized computer based assessment to judge '<u>employability</u>'

2M tested

AMCAT

GRE/SAT for Jobs

600+ companies

Assessment driven job marketplace

Automatic grading of programs– Why?

Automatic grading of programs– Why?

- Help professors and TAs save time and provide more objective feedback to learners.

Automatic grading of programs– Why?

- Help professors and TAs save time and provide more objective feedback to learners.
- Companies can recruit efficiently and provide opportunity to more applicants

Automatic grading of programs– Why?

- Help professors and TAs save time and provide more objective feedback to learners.
- Companies can recruit efficiently and provide opportunity to more applicants
- **MOOCs** - NEEDs automated open response assessments to really make it effective.

Existing solutions

Existing solutions

- **Manual evaluation:** Doesn't scale; not standardized ✗
- **Test-case based evaluation:**
  - High false-positives – hard code, inefficient code ✗
  - High false-negatives – inadvertent errors
- **Similarity metric between control flow graphs, syntax trees:**
  - Cannot be tuned to human-evaluation ✗
  - Theoretical elegance broken due to multiple correct solutions

Our approach

***Automata* – Automatic program evaluation engine**

- **Machine Learning based scoring engine**
- **Evaluation of programming best practices**
- **Asymptotic complexity evaluation**

Our approach

Automata **– Automatic program evaluation engine**

Machine Learning based scoring engine

Evaluation of programming best practices

Asymptotic complexity evaluation

A model to predict the algorithmic correctness of a program, given the control and data dependencies it possesses

Our approach

Automata **– Automatic program evaluation engine**

Machine Learning based scoring engine

Evaluation of programming best practices

Lint-styled rule-based system to detect programs not following programming best practices.

Asymptotic complexity evaluation

Our approach

Automata **– Automatic program evaluation engine**

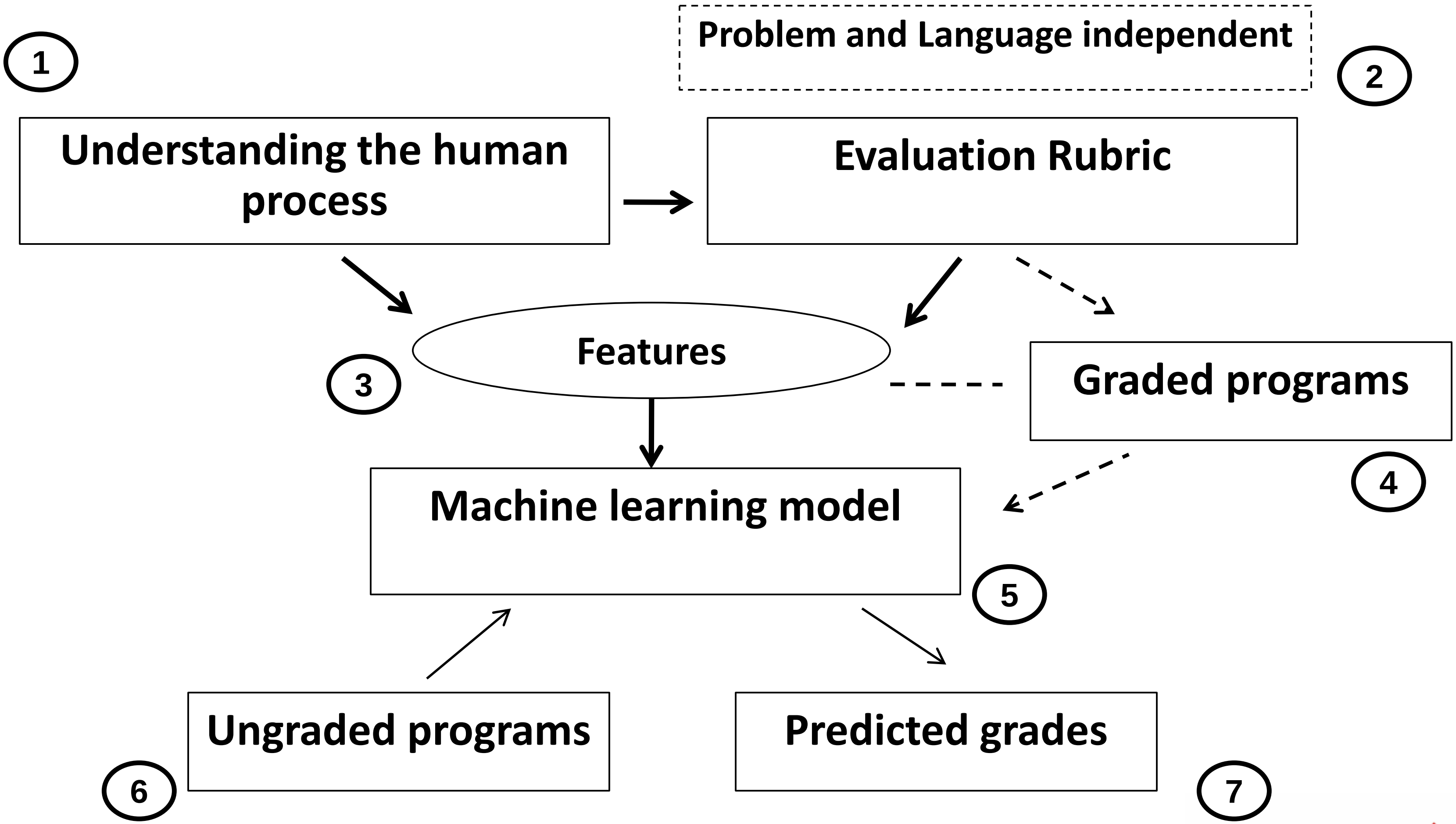
Machine Learning based scoring engine

Evaluation of programming best practices

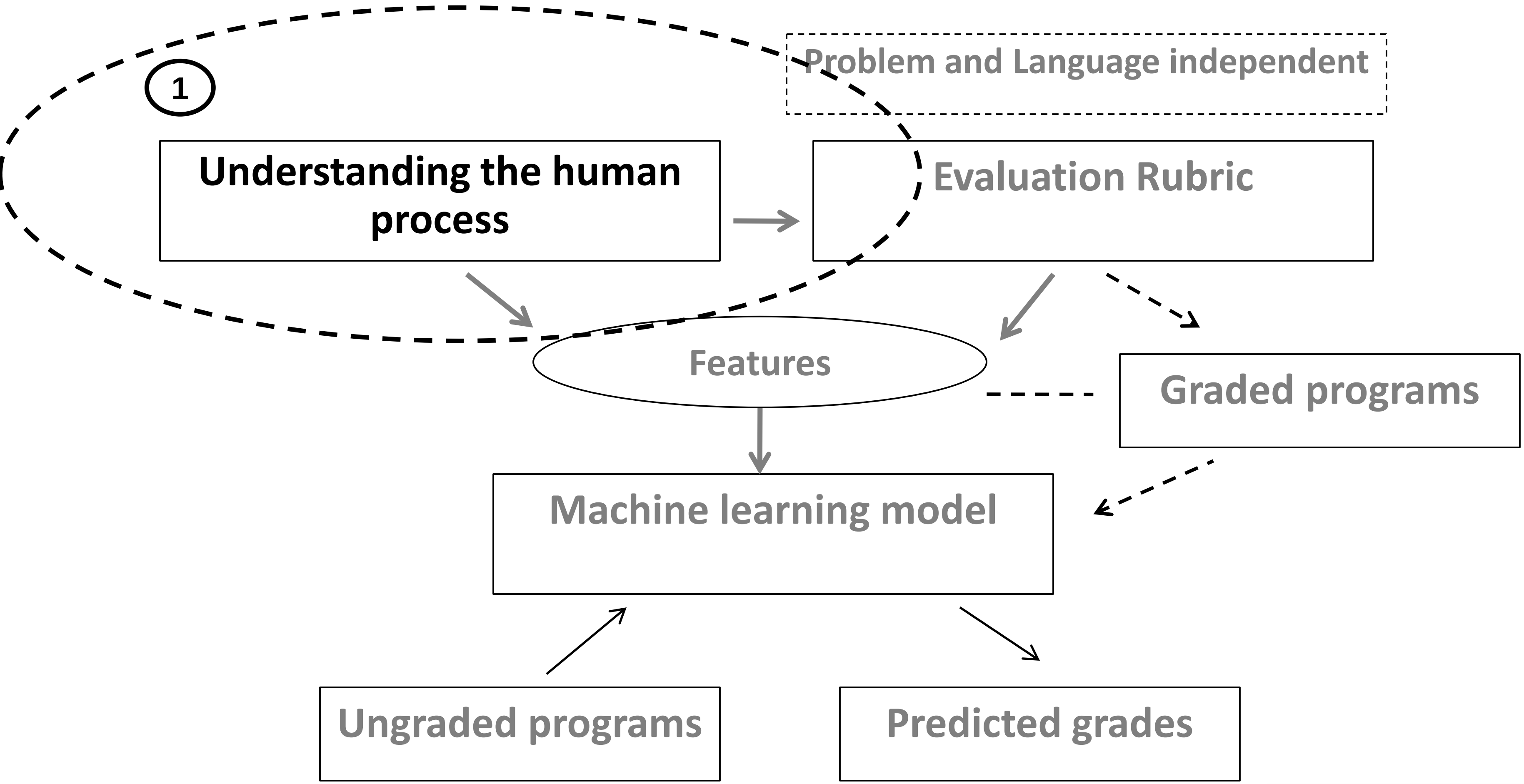
Asymptotic complexity evaluation

measures the run-time of the code for various input sizes and empirically derives the complexity

ML based scoring



Our approach



aspiringminds
Employability Quantified

What does a grader look for?

<u>OBJECTIVE</u>

To print N lines of the pattern of integers

```
1
2  3
3  4  5
4  5  6  7
…
```

An implementation

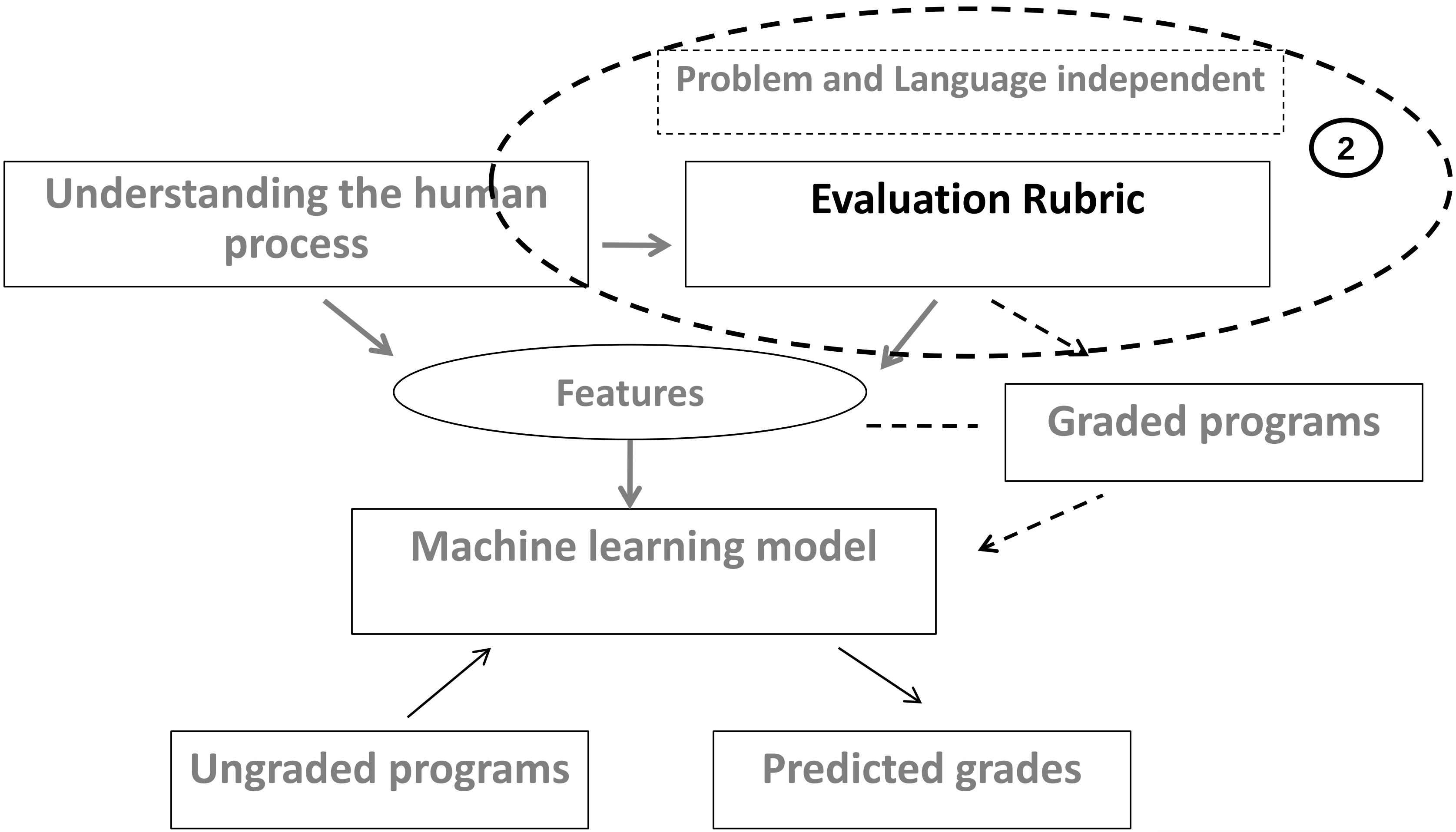
```
void print_1(int N){
  for(i =1 ; i<=N; i++){
     print newline;
     count = i;
     for(j=0; j<i; j++)
       print count;
         count++;
  }
}
```

[illegible] of the first loop?

- a variable modified in the outer loop?

- a variable used in the conditional of the outer loop?

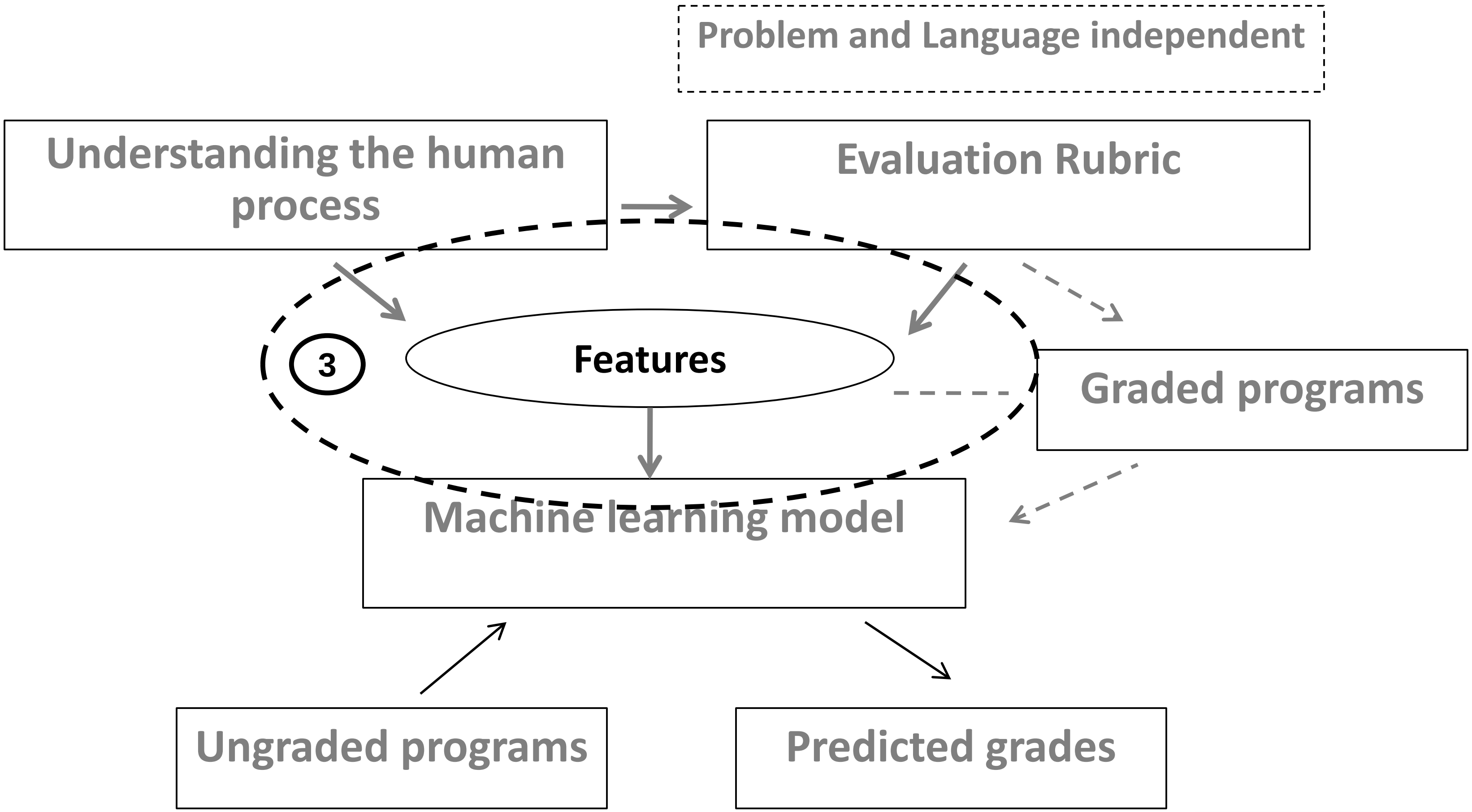
Our approach



Evaluation Rubric

Score	Interpretation
5	**Completely correct and efficient** An efficient implementation of the problem using right control structures and data-dependencies.
4	**Correct with some inadvertent errors** Correct control structures and closely matching data-dependencies. Some silly mistakes fail the code to pass test-cases.
3	**Basic program structure is consistent** Right control structures start exist with few correct data dependencies
2	**Emerging basic keywords and tokens** Appropriate keywords and tokens present, showing some understanding of the problem
1	**Gibberish code** Seemingly unrelated to problem at hand.

Our approach



aspiringminds
Employability Quantified

Grammar for expressing features

Grammar for expressing features

Simple Features

- **Keywords and Tokens - Counts :**
  - Tokens like `for, if, return, break;` function calls like `printf, strrev, strcat;` declarations like `int, char`
  - Operators like various arithmetic, logical, relational operators used
  - Character constants like `‘\0’, ‘ ’, ‘65’, ’96’`

Grammar for expressing features

Simple Features

- **Keywords and Tokens - Counts:**
  - Tokens like `for, if, return, break;` function calls like `printf, strrev, strcat;` declarations like `int, char`
  - Operators like various arithmetic, logical, relational operators used
  - Character constants like `'\0', ' ', '65', '96'`

Capturing logical constructs (Interactions)

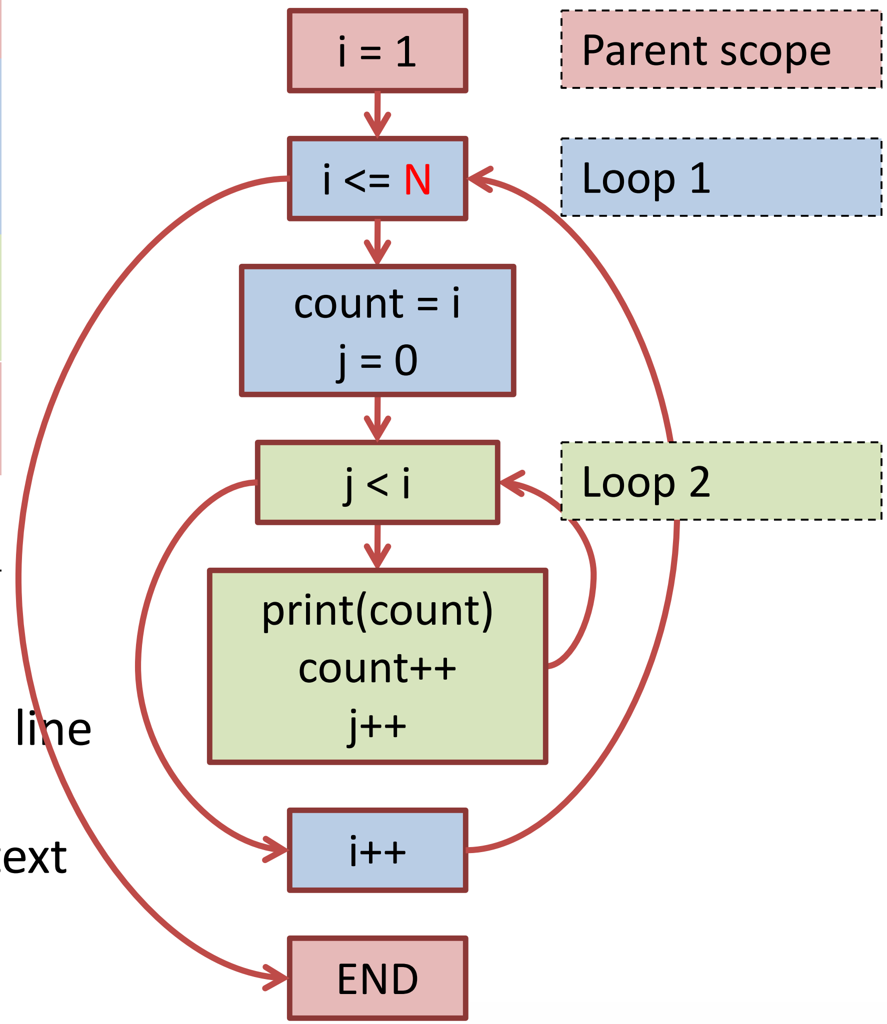
- **Control flow structure**
- **Data-dependencies**
- **Data-dependencies in context of control-flow**

aspiringminds
Employability Quantified

TARGET PROGRAM

```
void print(int N){
for(i =1 ; i<=N; i++){
    print newline;
     count = i;
    for(j=0; j<i; j++)
      print count; count++;
    }
}
```

CONTROL FLOW GRAPH



CONTROL FEATURES – COUNTS

Control-context of tokens

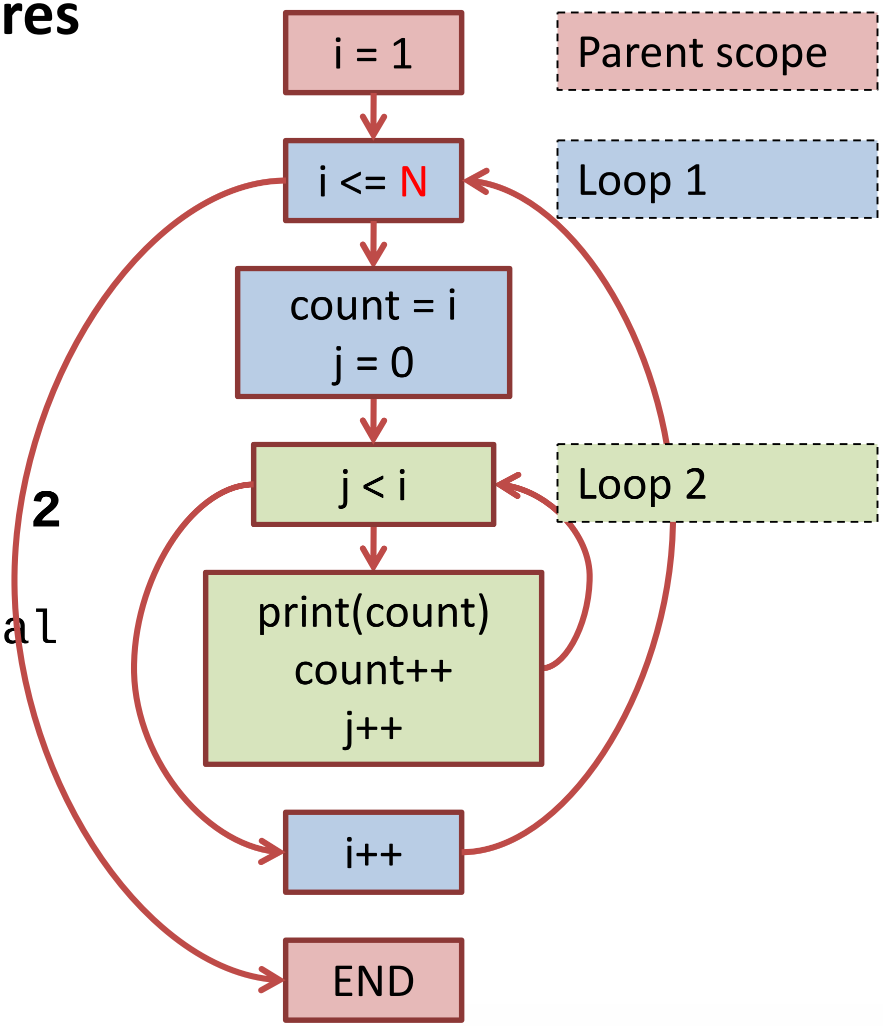
- Do a recursive traversal of tokens on each line
- For each token, tag it with its control context information

What do these control context features look like?

They are counts of occurrences of various sub structures like -

- `loop(variable assigned) :` **6**
- `nested loop(variable assigned):` **2**
- `loop(expression with a relational operator):` **2**
- `Loop(expression with a <= operator):` **1**

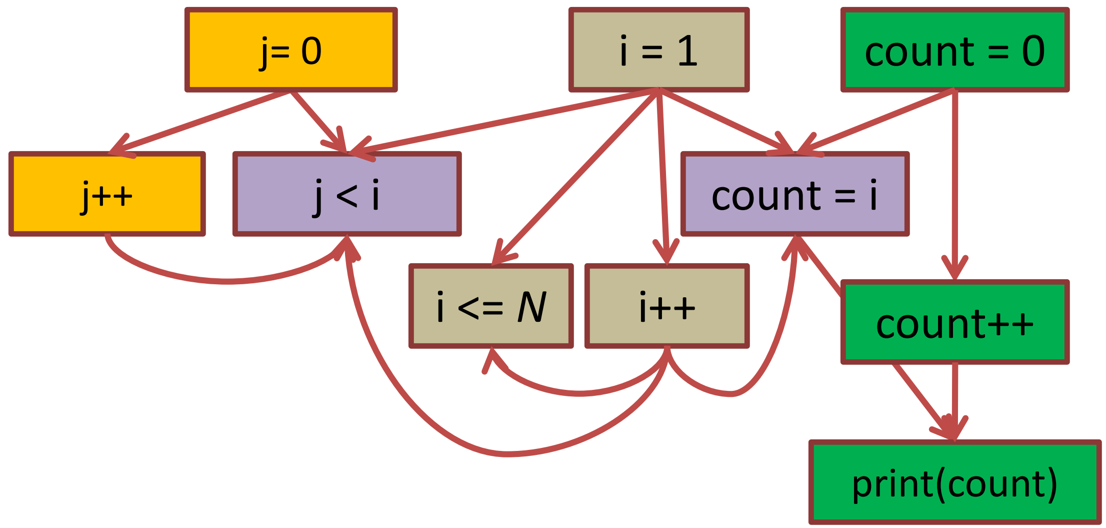
CONTROL FLOW GRAPH



TARGET PROGRAM

```
void print(int N){
for(i =1 ; i<=N; i++){
    print newline;
     count = i;
    for(j=0; j<i; j++)
      print count; count++;
    }
}
```

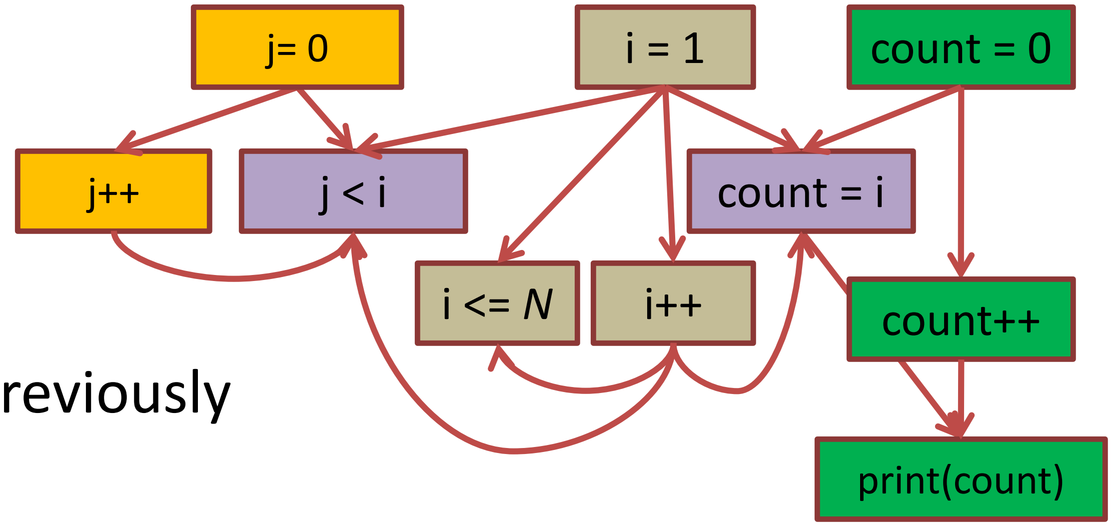
DATA DEPENDENCY GRAPH



Data dependency features

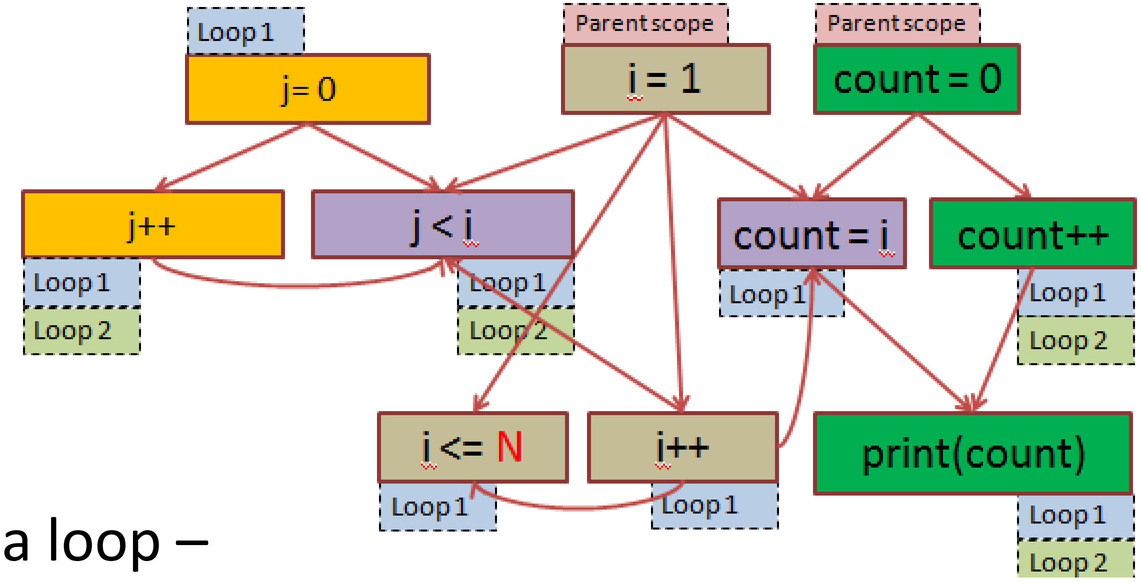
- Count the occurrences of usages and definitions of the variables coming up in the program
- For e.g.
- `i++` → `j < i` : **var (`i`) related to var (`j`) – previously incremented**

DATA DEPENDENCY GRAPH



Example features

- Assignment(variable) – previously incremented : 1
- Relational(variable) – previously incremented : 2
- Print(variable) – previously incremented : 1



Example features

- Assignment(variable) in a loop – previously incremented in a loop: 1

- Relational(variable) in a nested loop – previously incremented in a loop: 1

- Print(variable) in a nested loop – previously incremented in a nested loop: 1

Feature Grammar Summary

- Keywords
- Keywords in control-context
- Data dependencies
- Data dependencies in control context

Feature Grammar Summary

- Keywords
- Keywords in control-context
- Data dependencies
- Data dependencies in control context

To mimic human intuition, features are derived from DDGs and CFGs

They are able to distinguish between different rubric levels

Understanding the human process → Evaluation Rubric

Features

AUTOMATA – Our enterprise program evaluation software

- Online compiler, editing option available, each problem has a suite of test-cases it is tested against.
- Test case suite checks for basic cases and pathological conditions in the code.
- Each test contains two programming problems. Involves both, freshman level and advanced level problems.
- The system generates a reports scores an functionality score (ML), time complexity score and program maintainability/readability score.
- The test works with a web-cam in an autoproctored environment.

Automata – A sample report

Problem Statement 1

Sort an array partially into ascending and remaining into descending order.
input: *arr*-> integer array, *k*->index till which the array is in ascending order.
Output: Resulting array.

Note: A more detailed problem statement is shown to the candidates.

Problem summary

Candidate Source Code

Final Code Submitted

```
 1. // IMPORT LIBRARY PACKAGES NEEDED BY YOUR PROGRAM
 2. // SOME CLASSES WITHIN A PACKAGE MAY BE RESTRICTED
 3. // DEFINE ANY CLASS AND METHOD NEEDED
 4. //CLASS BEGINS, THIS CLASS IS REQUIRED
 5. public class ArraySort
 6.  {
 7.  //METHOD SIGNATURE BEGINS, THIS METHOD IS REQUIRED
 8.    public static int[] findArrSort(int[] arr, int k)
 9.    {
10.      // Sort first K elements of arr in ascending and
   remaining in descending order
11.      // Return the sorted array
12.    // INSERT YOUR CODE HERE
16.      {
17.          for(int i=0;i<k;i++)
18.          {
19.              if(arr[h] < arr[i])
20.              {
21.                  temp=arr[h];
22.                  arr[h]=arr[i];
23.                  arr[i]=temp;
24.              }
25.
26.          }
27.      }
28.      for(int g=k;g<a;g++)
29.      {
30.          for(int j=k;j<a;j++)
31.          {
32.              if(arr[g]>arr[j])
33.              {
34.                  temp=arr[g];
35.                  arr[g]=arr[j];
36.                  arr[j]=temp;
37.              }
38.          }
39.      }
40.    return arr;
```

Candidate's source code

Results

Code Execution Summary

Code Compilation	:	Pass
Compiler Warnings Generated	:	No
Test Cases Passed	:	5/5

Warnings Generated

None

Test Case Execution Results(Cases Passed/ Total Cases)

Basic 3/3
They demonstrate the primary logic of the problem. They encompass situati... seen on an average and do not reveal situations which need extra checks/ha... the logic.

Advanced 1/1
They contain pathological input conditions which would attempt to break codes which have incorrect / semi-correct implementations of the correct logic or incorrect / semi-correct formulation of the logic.

Edge 1/1
They specifically confirm whether the code runs successfully on the extreme ends of the domain of inputs.

Total **5 / 5**

Test case pass/fail information
Machine learning score

Structural Vulnerabilities and Errors

Readability
Line No 8,13: Variables are given very short names.

Performance
Line No 13: Local variable 'a' could be declared final

Feedback on programming best practices

Average-Case Time Complexity Detected

$O(N^2)$

This problem can be ideally solved in **O(N)** time

*N represents the number of elements in the input array

Asymptotic complexity of the candidate's solution

Experiment - Objectives

- Do our features predicting control-flow and data-dependency information add value over simple count-based features? If so, by how much?
- Do features derived from keywords, control-structures and data-dependencies add value over the information provided by test-cases?
- How accurately can a machine learning approach based on our novel feature set predict grades as compared to grades given by human assessors?

Experiment - Details

PROBEM 1 – *Encrypt* - Add numbers to each character based on its position in a string

PROBLEM 2 - *Alt Sort* - Sort a given list of numbers and return the alternating elements

PROBLEM 3 - *Find Digit* - Given two numbers - a multi-digit number and a digit, find the number of times the digit appears in the number

PROBLEM 4 - *List Primes* - List out all the prime numbers less than a given number

PROBLEM 5 - *Print Spiral* - Print N lines of a spiraling pattern of digits.

PROBLEM	Prob 1	Prob 2	Prob 3	Prob 4	Prob 5
SAMPLE SIZE	106	84	235	280	294

Experiment - Learning algorithms used

- Linear Regression with ridge regularization
- SVM
- Random Forests

- Neighborhood approach (Mimics single class classification)

Experiment - Objectives

- **Do our features predicting control-flow and data-dependency information add value over simple count-based features? If so, by how much?**
- **Do features derived from keywords, control-structures and data-dependencies add value over the information provided by test-cases?**
- **How accurately can a machine learning approach based on our novel feature set predict grades as compared to grades given by human assessors?**

Experiment - Results

PROBLEM	Type of feature	# of features	Cross-val correl	Train correl	Validation correl
1	All, w/o testcase	35	0.57	0.72	**0.56**
	Basic	60	0.62	0.87	**0.41**
2	All, w/o testcase	80	0.81	0.99	**0.80**
	Basic	26	0.59	0.72	**0.67**
3	All, w/o testcase	190	0.87	0.97	**0.90**
	Basic	26	0.74	0.89	**0.74**
4	All, w/o testcase	134	0.85	0.91	**0.82**
	Basic	35	0.83	0.88	**0.69**
5	All, w/o testcase	166	0.66	0.81	**0.64**
	Basic	40	0.61	0.78	**0.61**

Control and Data dependency features add around 0.15 correlation points above bag-of-words information

Experiment - Objectives

- Do our features predicting control-flow and data-dependency information add value over simple count-based features? If so, by how much? ✓
- Do features derived from keywords, control-structures and data-dependencies add value over the information provided by test-cases?
- How accurately can a machine learning approach based on our novel feature set predict grades as compared to grades given by human assessors?

Experiment - Results

PROBLEM	# of features	Cross-val correl	Train correl	Validation correl	Test Case correl
1	80	0.61	0.85	**0.79**	**0.54**
2	68	0.77	0.93	**0.91**	**0.80**
3	193	0.91	0.98	**0.90**	**0.64**
4	66	0.90	0.94	**0.90**	**0.80**
5	87	0.81	0.92	**0.84**	**0.84**

Validation correlation > 0.79

Matches inter-rater correlation between two human raters

Experiment - Results

Neighborhood approach (Mimics single class)

- Absolute mean distance of programs from model programs (score 4 and 5)
- Mean of 25% minimum distances from model programs as score
- Score 4 and Score 5 codes chosen as train set.
- Use threshold for type 1/type 2 error: Set 1: Scores 1,2 and 3; Set 2: Scores 4 and 5

PROBLEM	All features		Basic features	
	Mean	Min25	Mean	Min25
1	0.65	**0.65**	0.59	**0.63**
2	0.80	**0.83**	0.68	**0.69**
3	0.72	**0.80**	0.56	**0.67**
4	0.76	**0.78**	0.65	**0.66**
5	0.58	**0.58**	0.49	**0.51**

Features - Insights

- Analyze the most contributing features in a problem's model
- It could help discover important logic elements in the program, thereby helping in providing feedback to candidates
- It could help improve feature engineering
- Features for *FindDigit* problem analyzed. Given a multi-digit number and a digit, one has to find the number of times the digit appears in the number

Features - Insights

- The most contributing feature for *FindDigit* problem -

$$\begin{array}{c} \text{Dep@Var:1,Op:!=,Const:1\#input:m\_LOOPc} \\ \uparrow \\ \text{Var:1,Op:/,Const:1\#input:m\_LOOPb} \end{array}$$

```
int findDigit(int N, int digit){
   ...
   LOOP (N != <constant value>){
      ...
      N = N / <constant value>
      ...
   }
   ...
}
```

→

```
int findDigit(int N, int digit){
   ...
   while(N != 0){
      d = N%10;
      if(d == digit)
         ...
      N = N / 10;
   }
}
```

Conclusion

- We propose the first machine learning based approach to automatically grade programs
- An innovative feature grammar is proposed which matches human intuition of grading programs.
- Models built for sample problems show promising results.
- We propose machine learning techniques to lower the need of human-graded data to build models.
- We have a working system which can be used by companies and universities… Try it!

Future work

- This is a beginning point in automatic program assessment!
- Better ML techniques on problems with more data points + unsupervised feature clustering
- Bigger picture – a framework to use machine learning in the assessment of any open-response problem
- To reduce the requirement of sample programs needed to be evaluated by experts – improvements by one-class classification techniques
- Is this a beginning point for an automatic programming TA?

We are happy to engage with folks
- who want to use the platform in their class
- who want to use our data sets/features for fun stuff!

We have 200,000+ code samples!
2M+ assessment data + employment outcomes

Thank you

varun@aspiringminds.com

shashank.srikant@aspiringminds.com

aspiringminds
Employability Quantified

Research team

Experiment - Details

Sample sizes of problem set -

PROBLEM	Prob 1	Prob 2	Prob 3	Prob 4	Prob 5
SAMPLE SIZE	106	84	235	280	294

Number of features selected -

FEATURE TYPE	Prob 1	Prob 2	Prob 3	Prob 4	Prob 5
All features	80	79	193	66	87
All features w/o test cases	35	80	190	134	166

Sample feature generated -

%:arith_op:m_LOOPb_IFb_IFb
A modulus operator appears inside the body of a nested-conditional which in turn is present in a loop