Mining Software Students CVS Repositories for performance Indicators

Keir Mierle, Kevin Laven, Sam Roweis, and Greg Wilson

Presented by Keir Mierle (keir@cs.utoronto.ca)

Presentation Overview

- Goals
- CVS parsing system
- Feature Extraction
- Visual & Quantitative analysis
- Mutual Information
- Effect of work habits and code quality on grades
- Predicting grades from summary statistics

Overall Goals

- Predict student performance
- Check hypotheses like
  - “Students who start early do well”
  - “Students with many CVS transactions do well”
- Find early indicators of struggling students by predicting low grades
- Visualize students work habits
  - Useful for instructors

CVS Parsing System

- Python and MySQL
- Took the C++ RCS parser from ViewCVS and fixed it up for modern GCC
- Take two passes over every students repository and put the data into a MySQL database
- CVS is getting long in the tooth. Please switch to Subversion for the sanity of undergraduate research slaves everywhere

Feature Extraction

- DB has lots of data that is not (yet) useful.
- Need to quantify data so that we can do numerical analysis
- Convert data to summary statistics like
  - average time between CVS checkins
  - total number of lines of code a student wrote
  - total number of for loops
  - many more (166 in all)

Features from three places

1. From the database of CVS data
2. From parsing Java and Python code manually with a custom parser (**java**_ and **python**_)
3. By counting number of rule violations flagged by PMD, a style checker for Java. (**pmd**_)

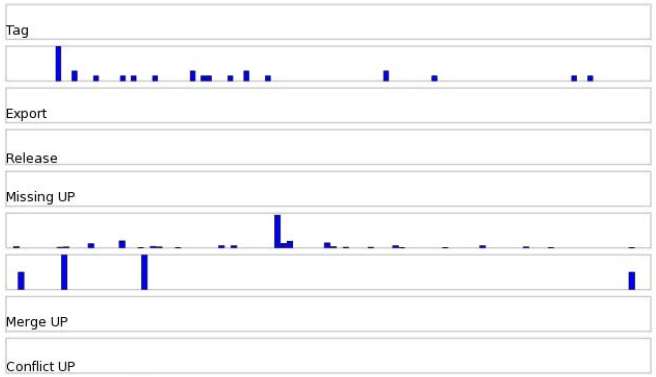
Finally, we added the grade the student received in the course.

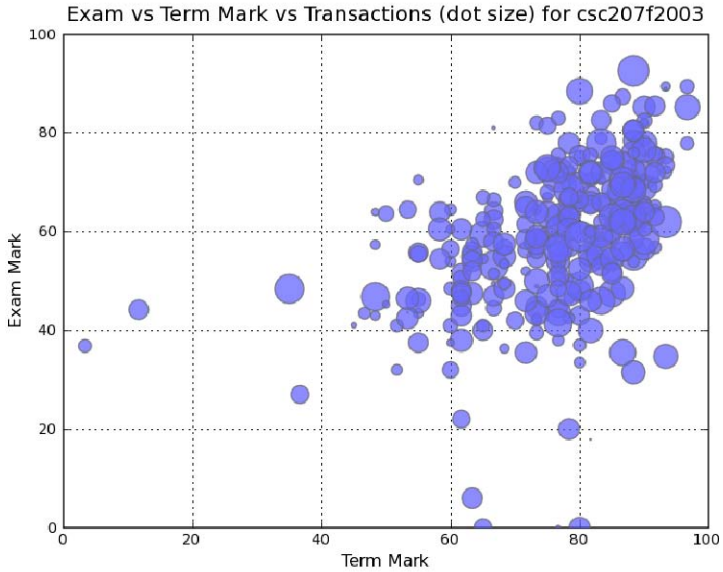
Cryptic subset of features

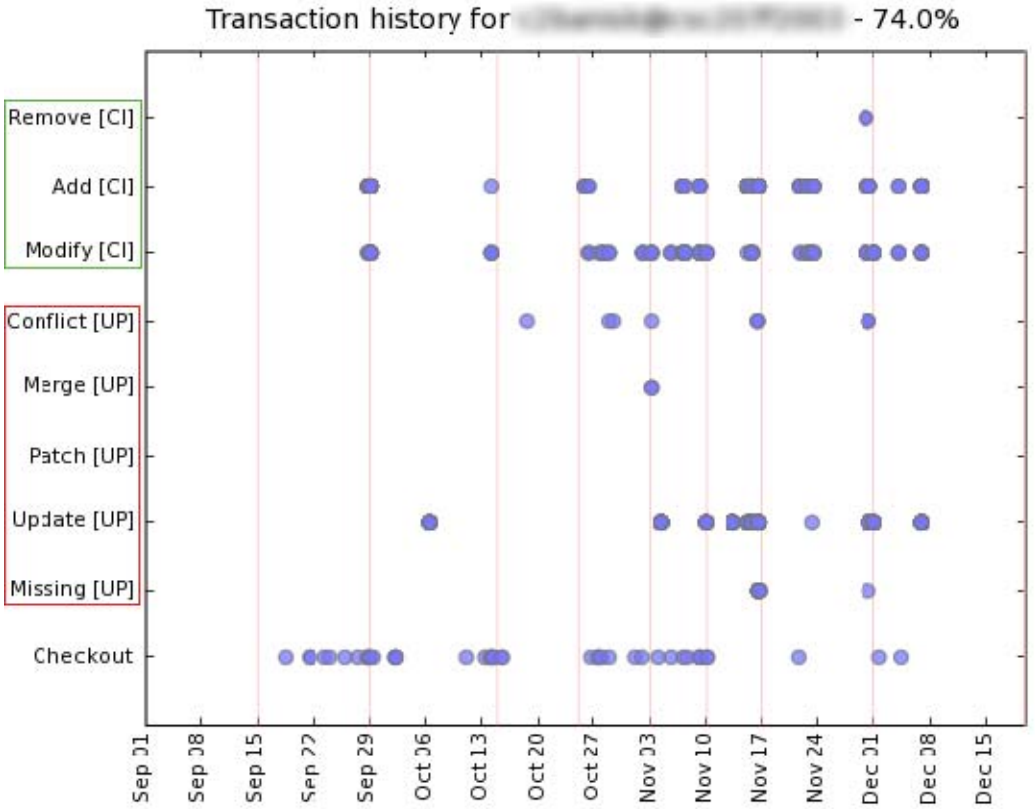
totaldifflength
python_tokencount_COMMENT
python_tokencount_STRING
python_tokencount_OP
python_source_comment_length
totalcomments
leading_tabs
four_spaces
python_tokencount_KW_def
python_tokencount_KW_while
python_terminal_tokens
avgrevsperfile
comment_nospace_nocaps
comma_space
java_count_public
numcommits
blank_lines
python_tokencount_KW_for
mixed_tabs_spaces
python_tokencount_ENDMARKER
python_tokencount_NL
totalfiles
comma_nospace
comment_space_caps
constant_subscript
mixed_spaces_tabs
comment_space_nocaps
pmd_assertions_need_message_rule
numtrans_type_M
python_tokencount_KW_except
python_tokencount_KW_self
python_tokencount_NAME
pmd_avoid_duplicate_literals
numtrans
comment_nospace_caps
java_error_parsing
python_tokencount_KW_in
pmd_only_one_return
java_source_files
python_tokencount_KW_try
pmd_assignment_in_operand_rule
python_tokencount_KW_if
numtrans_type_A
java_class_files_in_repo
python_tokencount_KW_class
remoteops
python_tokencount_NEWLINE
python_tokencount_INDENT
python_tokencount_DEDENT
pmd_bean_members_should_serialize
avgwaittime
python_tokencount_KW___name__
tabs
python_tokencount_KW_import
pmd_short_variable
pmd_excessive_method_length
pmd_empty_catch_block
pmd_long_variable
localops
pmd_method_naming_conventions
pmd_parse_error
python_tokencount_NUMBER
python_tokencount_KW_elif
numtrans_type_U
java_count_static
pmd_string_instantiation
pmd_at_least_one_constructor
java_count_String
pmd_loose_coupling_rule
java_count_void
java_count_main
pmd_avoid_reassigning_parameters_rule
avgfilespercommit
pmd_cyclomatic_complexity_rule
numtrans_type_W
numtrans_type_P
pmd_empty_if_stmt
numtrans_type_C
numtrans_type_G
pmd_final_field_could_be_static
java_count_out
java_count_println
java_count_while_statement
numtrans_type_O
pmd_if_stmts_must_use_braces
pmd_unnecessary_constructor_rule
python_tokencount_KW_range
python_tokencount_KW_assert
pmd_unused_imports
pmd_for_loops_must_use_braces_rule
pmd_unused_local_variable
pmd_signature_declare_throws_exception
python_tokencount_KW_or
java_count_protected

Visual and Quantitative Analysis

- Created lots of displays to visualize workflow
- Instructors discovered things they didn't expect







Mutual Information

$$MI(f) = \sum_{y=0,1}\sum_{k=1}^{K} p(y)p(f \in f_k \mid y)\log_2 \frac{p(f \in f_k \mid y)}{p(f \in f_k)}$$

$$MI(f) = \frac{1}{2}\sum_{y=0,1}\sum_{k=1}^{K} p(f \in f_k \mid y)\log_2 \frac{p(f \in f_k \mid y)}{p(f \in f_k)}$$

- A better measure of correlation than the correlation coefficient
- Captures higher-order relations
- Intuitively, the MI between two random variables is “how many bits of information do I know about X if I already know the value of Y?”
- In our case, *p(y)* = 0.5 because we divided the class up into thirds, and only considered the top and bottom thirds (which have equal size)

Effect of Work Habits on Grades

- Theory: Students who use CVS do well
- Reality: The data suggests how a student uses CVS has no affect on final grade
  - None of the CVS-related features had enough mutual information to be considered even close to significant (at the $p = 0.05$ level)
  - Starting early or very late has no effect
  - Submitting final assignment with 10 seconds to spare or 2 days to spare has no effect

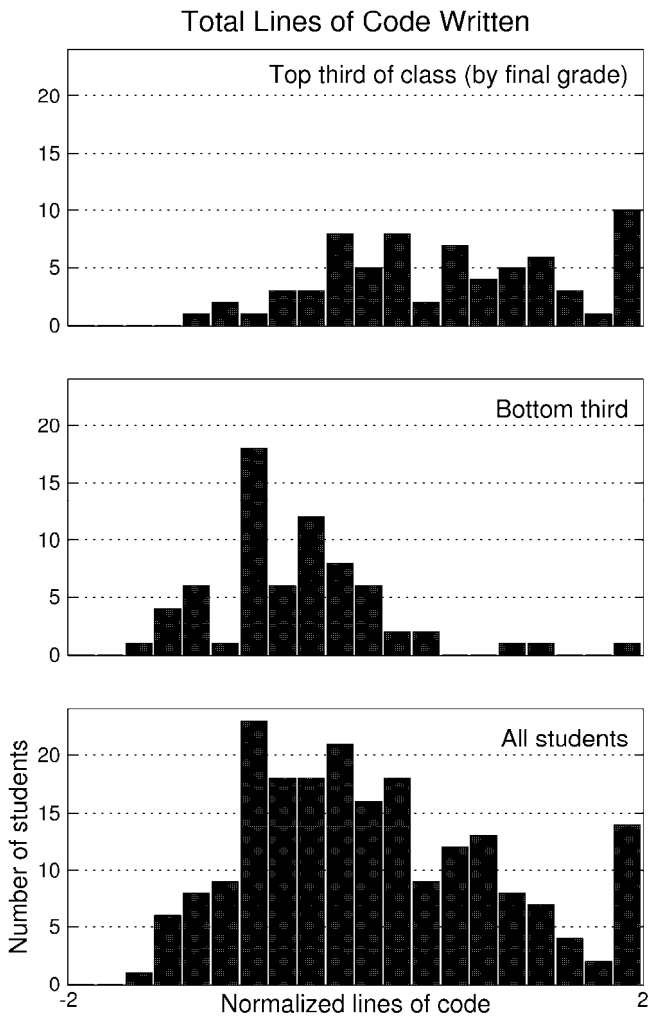
Effects of Code Quality on Grades

- Strongest indicator of performance:

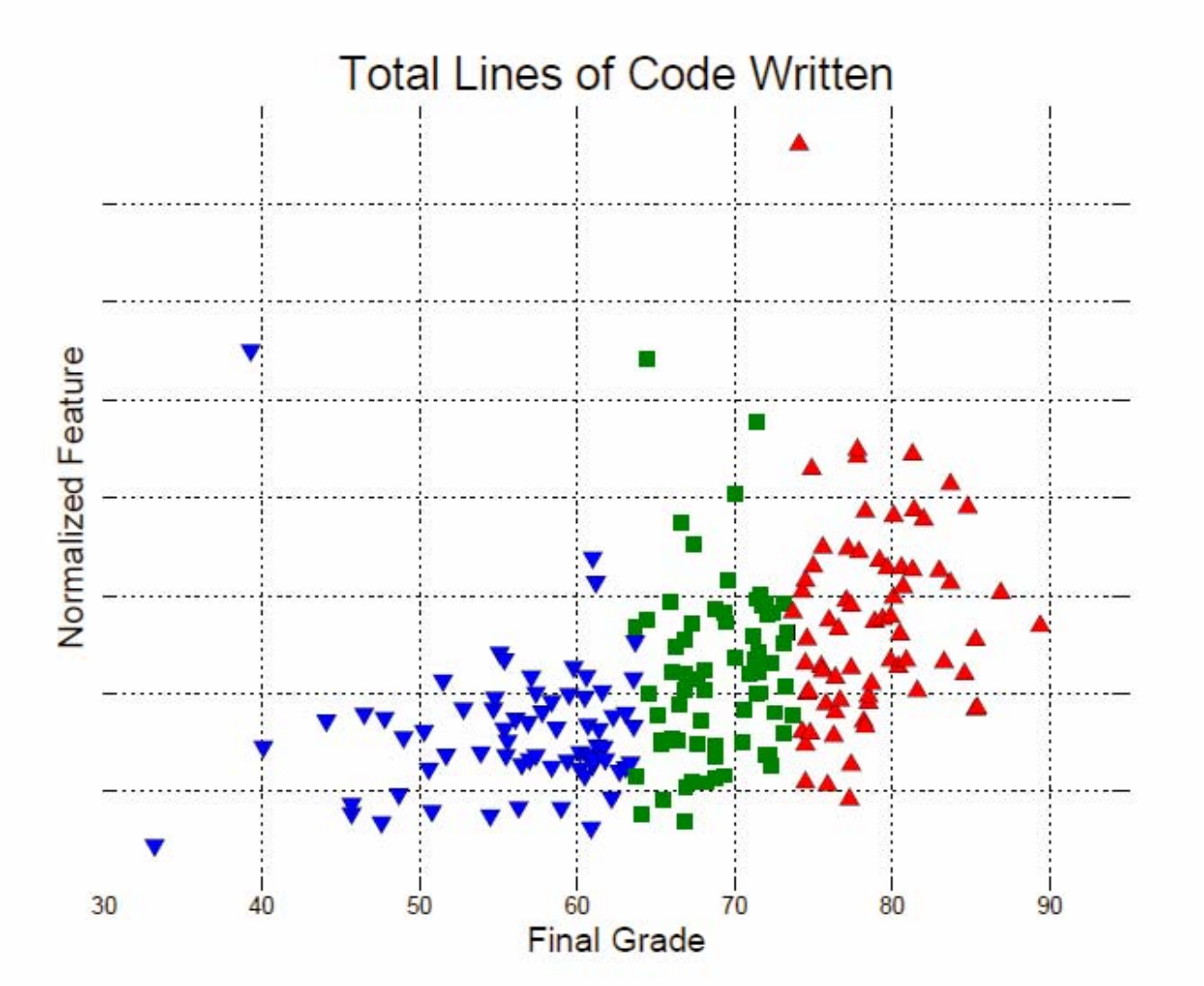
  Total lines of code written

- Only one feature besides code length was a significant indicator of performance:

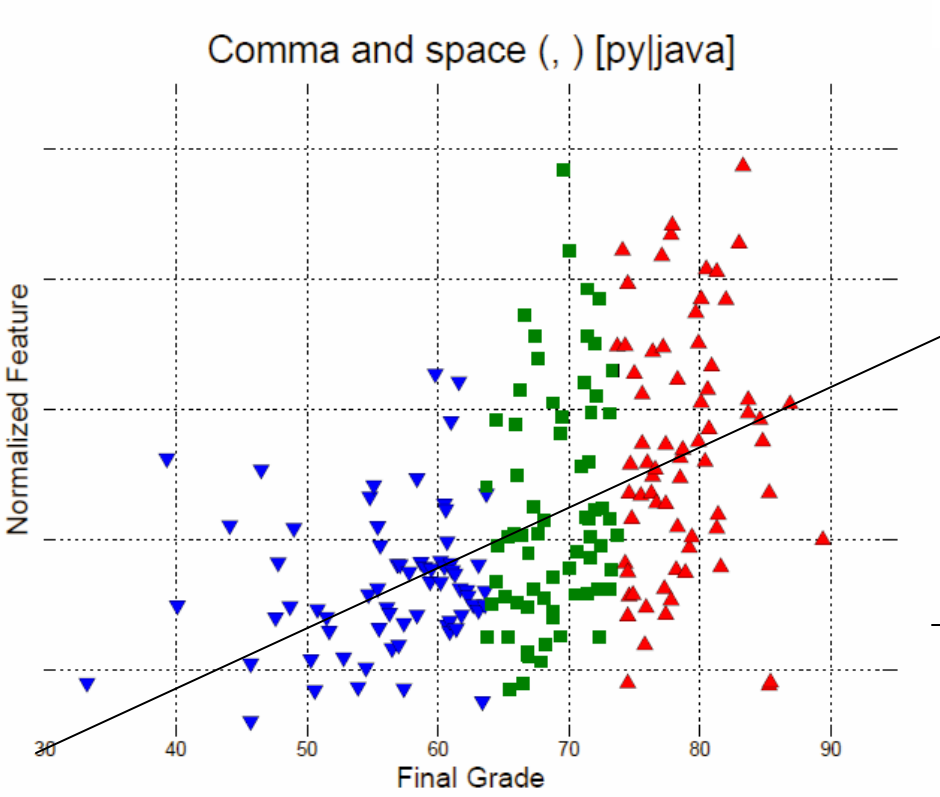
  How many times a space followed a comma. i.e. foo(a,b,c) vs foo(a, b, c)



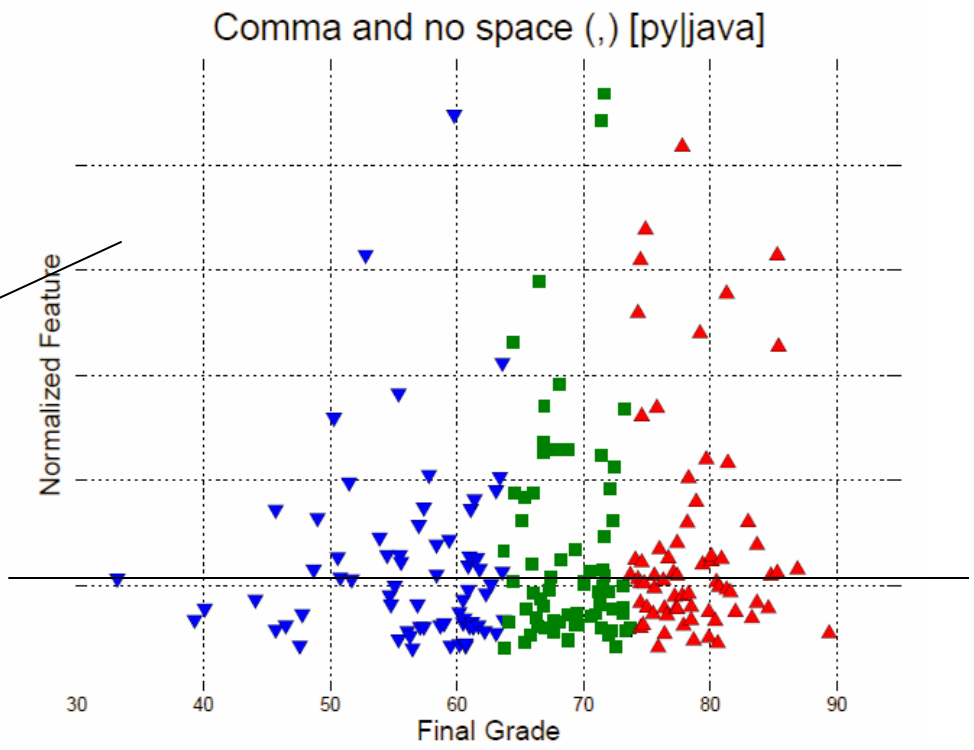
Total lines of code written



Space follows comma



Notice linear trend



No trend here!

Predicting Performance

- Represent each student as a vector of 166 features
- Normalize each feature as if they were Gaussians, i.e. subtract mean and divide by standard deviation (of that feature)
- Mutual information suggests this is not going to work well

Predicting Performance

- Logistic Regression
  - A simple hyperplane classifier
  - With leave-one-out testing, 30% error
- Naïve Bayes
  - (Naïvely) assumes features are independent
  - With leave-one-out testing, 24% error
- K-nearest-neighbour
  - With leave-one-out testing, also 24% error
- Prediction not very useful

Conclusion

- Didn’t find what we expected
- We’re integrating some of the displays developed for this research into the Argon project
  - http://www.third-bit.com/trac/argon
- Python CVS parser is available
  - www.cs.utoronto.ca/~keir/slurp-1.0.0.tar.gz
  - Other code too specific to release. Email me if you’re interested anyway
- If you find indicative features, let us know!