Published in AJSE, Vol: 25, Issue:2
Received on 7th August 2026
Revised on 8th October 2026
Published on 31st October 2026

AIUB JOURNAL OF SCIENCE AND ENGINEERING
ISSN: 1608 - 3679 (print) 2520 - 4890 (Online)
https://doi.org/10.53799/0efxvm96

Predicting Student Attrition in Competitive Programming: A Large-Scale Study Integrating Survey Insights and Global Behavioral Logs

Azuad Islam Ruhan, Golam Mostofa Naeem, Rakibul Islam Rafi, Sherin Afrin Mim, Nazira Bani Opi, Dewan Fahad Chowdhury, and Md. Rejaul Korim Sadi

Abstract—Competitive programming develops algorithmic reasoning in computer science students, yet sustained participation is fragile and many withdraw after stalled progress or thin mentorship. Dropout has been studied extensively in degree programs and online courses, but competitive programming attrition has not been examined as a behavioral phenomenon in its own right. This paper reports a two-layer study combining platform telemetry with a localized survey. We retrieved activity traces for 6,927 Codeforces accounts and reduced them to a balanced set of 1,816 users, half persistently active and half showing frustrated withdrawal, matched on peak rating so that disengagement is not confounded with skill. We additionally audit the features for temporal leakage, removing three columns whose predictive weight derives from when a user stopped rather than what they did. In parallel we surveyed students at ten universities in Bangladesh. Attrition follows an 83.7 percent fall in contest participation and a 42.7 percent drop in problems solved monthly. Students who stopped practised significantly less between contests while rating their own ability no lower than active peers, a divergence we term the Skill-Application Paradox. Random Forest reached a cross-validation F1 of 0.682 on the audited behavioral data and 0.832 on the survey layer.

Index Terms—Attrition prediction, competitive programming, educational data mining, machine learning, survey analytics

I. Introduction

Competitive programming (CP) has become one of the more reliable extra-curricular routes for building algorithmic reasoning in computer science students [1], [2]. Codeforces, LeetCode, and AtCoder all supply the same basic ingredients: immediate verdicts, timed contests, and a public leaderboard against which progress can be measured [3]. What separates this from a graded laboratory assignment is the absence of scaffolding. Problems arrive without hints, under fixed time and memory budgets, and the student has to decide on their own what to try next. Employers value exactly that ability [4].

The difficulty is staying with it. A large share of undergraduates begin with real enthusiasm and stop within months [5]. Computing education research has linked this pattern to swings in programming self-efficacy, a sense that progress has stalled, social isolation, and pressure from coursework [6], [7]. Dropout itself is well trodden ground in educational data mining: institutional models have been built on grade records, attendance, and clickstreams from massive open online courses [8], [9]. Competitive programming sits outside all of that. It happens off the transcript, runs on voluntary effort, and has attracted very little empirical attention [10].

Our motivation is practical. Programming clubs and faculty mentors have no systematic way of noticing that a student is drifting until the student has gone. If that drift could be detected while the student is still submitting, there would be room to act: a conversation, a pairing with a senior member, a structured review session after the next contest.

We therefore built a two-layer study and framed it around three questions. RQ1 asks which behavioral and academic reasons students themselves give for discontinuing competitive programming. RQ2 asks whether behavioral proxies drawn from Codeforces activity separate active users from true attrition users. RQ3 asks whether models trained on psychographic survey responses can pick out at-risk individuals among students who are still active.

The paper makes four contributions. First, we identify and test the Skill-Application Paradox, in which students who quit report at least as much theoretical confidence as students who stay while practising noticeably less. Second, we introduce two domain-specific interaction features, the Intensity Ratio and the Editorial Dependency Index. Third, we establish an empirical baseline linking global telemetry to localized survey data, with the pipeline described in enough detail to be reproduced. Fourth, we deploy an Early Warning System over currently active students as a proof of concept, scoring every student out of fold.

II. Related Work

Educational Data Mining and Student Dropout

Work on student retention has shifted over two decades. Early models predicted degree completion from demographics and first-year marks [8], [11]. Attention later moved to dynamic behavior, submission patterns and forum participation, driven by the data online courses made available [9]. Random Forest [12], XGBoost [13] and

Azuad Islam Ruhan (corresponding author) is an undergraduate researcher in the Department of Computer Science and Engineering, Metropolitan University, Sylhet, Bangladesh. E-mail: ahnafruhan7@gmail.com
Golam Mostofa Naeem is a Senior Lecturer in the Department of Computer Science and Engineering, Metropolitan University, Sylhet, Bangladesh. E-mail: gmnaeem7@gmail.com
Rakibul Islam Rafi is an undergraduate student in the Department of Computer Science and Engineering, Metropolitan University, Sylhet, Bangladesh. E-mail: rakibulislamrafi36@gmail.com
Sherin Afrin Mim is an undergraduate student in the Department of Computer Science and Engineering, Metropolitan University, Sylhet, Bangladesh. E-mail: sherinafrinmim6@gmail.com
Nazira Bani Opi is an undergraduate student in the Department of Computer Science and Engineering, Metropolitan University, Sylhet, Bangladesh. E-mail: nazirabani2@gmail.com
Dewan Fahad Chowdhury is an undergraduate student in the Department of Computer Science and Engineering, Metropolitan University, Sylhet, Bangladesh. E-mail: dewanfahadc@gmail.com
Md. Rejaul Korim Sadi is an undergraduate researcher in the Department of Computer Science and Engineering, Metropolitan University, Sylhet, Bangladesh. E-mail: rejaulkorimsadi@gmail.com

LightGBM [14] are now the default comparison points for tabular educational prediction .

Programming Platform Telemetry

Studies of online judges show that granular submission logs carry information about cognitive state: compilation error frequency, inter-submission gaps and wrong-answer rates have been treated as proxies for frustration and learning difficulty [10], [15]. Cognitive load theory gives a reason to take long struggle times seriously, since past a point repeated failure signals overload rather than productive effort [16]. Most of this work sits in structured assignments with a fixed deadline; voluntary competitive participation behaves differently, and disengagement plays out over weeks.

Psychographic and Social Factors

Motivational orientation and peer support are established predictors of persistence in computing [17], [18]. Competitive environments complicate the picture. Adoption of programming contests has been studied as a diffusion process driven by peer influence and perceived benefit [19], which implies the same social channels can work in reverse when progress stalls. Educational data mining on introductory programming has also shown that behavioral traces predict failure earlier than static background variables [20]; our design follows that logic into the voluntary setting.

The Integration Gap

We are not aware of any prior study pairing competitive programming platform logs with psychographic survey responses. Telemetry is objective but silent on why a user stopped; surveys reach the reasoning yet depend on recall [22]. We use one to compensate for the other.

III. Methodology

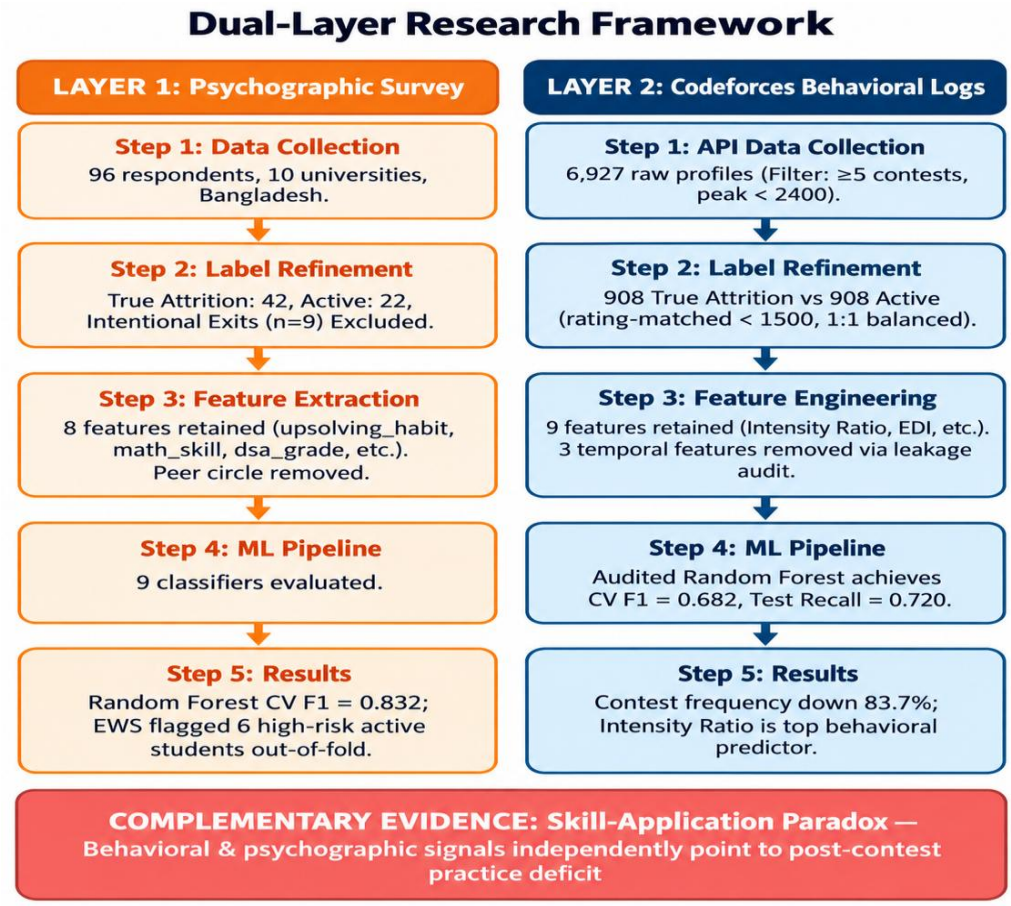


Fig. 1. Two-layer research framework. The survey pipeline and the Codeforces pipeline are processed separately and meet only at the interpretation stage.

Dataset 1: Codeforces Behavioral Logs

Submission histories, contest records, and profile metadata were pulled from the public Codeforces API for an initial pool of 6,927 accounts.

All features are computed relative to a single fixed reference date, the date on which the API crawl completed, denoted T_obs. Every account therefore contributes a snapshot taken at the same moment in calendar time, not at the same point in its own lifecycle. The recent and prior three-month windows in the Activity Trend Ratio are the three and six calendar months preceding T_obs, and the three-month inactivity criterion in the attrition label is measured backward from T_obs as well.

Two inclusion criteria were applied. Accounts were only retrieved if they had entered at least five rated contests, a criterion enforced at collection and therefore satisfied by all 6,927 raw profiles. The second, peak rating below 2400, was applied afterward and removed 35 professional-level competitors.

Inactive users fall into two very different groups. An account that had gone three consecutive months without a

submission and had never exceeded 1500, the Codeforces Specialist boundary, was labeled True Attrition, on the reasoning that a user who never climbed past Pupil is unlikely to have left because they achieved what they came for. Accounts that went quiet at 1500 or above were labeled Intentional Exits and dropped, since career transitions are the more plausible explanation there.

Our first pass compared true attrition users against a random sample of all active users. That comparison is confounded, because the attrition class is capped at 1500 by construction while the active class is not, and any classifier trained on it can separate the groups partly by inferring skill level rather than disengagement. To isolate disengagement from ability we restricted the active pool to accounts that also peaked below 1500 and drew the 1:1 sample from within that restricted pool. There is no shortage of candidates: 4,047 active accounts fall below the threshold against 908 true attrition cases. Undersampling with a fixed seed gives a balanced set of 1,816 users matched on peak rating; random undersampling is the simplest of the standard remedies for class imbalance and avoids the synthetic-sample artifacts of generative alternatives [31]. Table I gives the count at every stage.

TABLE I
Cohort Construction, Stage by Stage

Stage	Accounts
Retrieved from API (≥5 rated contests)	6,927
After peak rating < 2400	6,892
Active at T_obs	5,573
Inactive ≥3 months	1,319
True Attrition (peak < 1500)	908
Intentional Exit (excluded)	411
Matching pool: active, peak < 1500	4,047
Sampled from pool	908
Rating-matched modeling set	1,816

Dataset 2: Multi-Institutional Survey

The study was reviewed and approved by the Research Ethics Committee of the authors' institution on 17 March 2026 (approval no. REC/MU/2026/007). Participation was voluntary and anonymous. The questionnaire collected no names, student identifiers, email addresses or institutional affiliations, and responses cannot be traced to individuals. The landing page stated the purpose of the study, that responses would be used for academic research, and that submission constituted consent, and respondents could close the form at any point without submitting. No incentive was offered. The Codeforces data comes from the platform's public API and covers only information users have chosen to make public; account handles are stored solely to deduplicate records and all published statistics are aggregates.

We ran a structured survey at ten public and private universities in Bangladesh and collected 96 responses from computer science students. Respondents sorted into three groups: currently active in CP (22), previously active but stopped (51), and never started (23). The 23 who never started were set aside.

The 51 who had stopped were not a single population. Nine described a deliberate move into web development or machine learning, which is an intentional exit and not what we set out to model, so they were excluded. The remaining 42, who cited stalled ratings, burnout, absent mentorship, or academic pressure, were labeled true attrition. The modeling set is therefore 22 active and 42 true attrition students, n = 64, and every test and model reported below uses that set.

The questionnaire branches by status, so active and stopped respondents answered different item sets. Any feature drawn from one branch alone is perfectly confounded with the outcome, and a test of it recovers only which branch a respondent was routed down. We align every feature across branches before use: data structure understanding from Q14 and Q37, mentor support from Q22 and Q35, prior consideration of quitting from Q28 and Q34, break history from Q27 and Q36, and upsolving from Q11 combined with the Q33 practice-habit item. One item admits no alignment. Peer circle density was asked only of stopped respondents, and the never-started branch that carries the analogous item was excluded from modeling, so no active-branch value exists at any level. We drop the feature entirely rather than test or model a variable that encodes the label.

Statistical Testing

We tested group differences with Welch's t-test [25] for numeric features, chosen over the pooled-variance form because the groups are unequal in size, and chi-square tests for categorical ones. Effect sizes accompany every test: Cohen's d [26] for numeric comparisons and Cramer's V [27] for categorical, each with a 95 percent confidence interval, obtained for the categorical measures by the percentile bootstrap over 2,000 resamples [28].

Because we run eight tests on the same sample, the conventional 0.05 threshold no longer means what it does for a single test: across eight independent tests of features that do not differ, the chance of one landing under 0.05 by accident is roughly 34 percent. We therefore report two corrections. Bonferroni [23] divides the threshold by the number of tests, giving 0.0063, and controls the probability of any false claim across the table. Benjamini-Hochberg [24] controls the false discovery rate at 5 percent, which is more permissive while still accounting for multiplicity. A feature marked in both columns of Table II passes under either standard; one marked in neither is a directional observation, not a finding.

TABLE II
Group Differences, Active (n = 22) vs. True Attrition (n = 42)

Feature	Test	Stat.	p	Effect [95% CI]	Bonf. / BH
upsolving_habit	χ^2	23.18	<0.001	V = 0.60 [0.45, 0.77]	✓ / ✓
math_skill	t	−2.31	0.027	d = −0.65 [−1.17, −0.12]	– / –
dsa_grade	χ^2	5.96	0.114	V = 0.31 [0.13, 0.53]	– / –
mentor_support	χ^2	2.22	0.136	V = 0.19 [0.00, 0.43]	– / –

Feature	Test	Stat.	p	Effect [95% CI]	Bonf. / BH
ds_understanding	t	−1.33	0.191	d = −0.34 [−0.86, 0.18]	– / –
thought_quit	χ^2	1.48	0.224	V = 0.15 [0.00, 0.40]	– / –
prior_exp	χ^2	0.22	0.642	V = 0.06 [0.00, 0.30]	– / –
long_break	χ^2	0.11	0.745	V = 0.04 [0.00, 0.29]	– / –

One feature survives correction. Upsolving habit separates the groups at chi-square 23.18, $p < 0.001$, with $V = 0.60$, an effect whose confidence interval sits well clear of zero. Nothing else clears either threshold. Mathematical self-rating reaches nominal significance at $p = 0.027$ with a medium effect but falls under both corrections, and we describe it below as directional rather than established.

Feature Engineering

Twelve features were derived from the raw API responses. Six are direct behavioral proxies: math solve rate, data structure solve rate, upsolves per contest, mean struggle minutes, activity trend ratio, and post-drop recovery. Each was binned onto a 1 to 5 ordinal scale by percentile across the full dataset, so that users with very different activity volumes remain comparable.

Two further features are interactions we constructed for this problem. The Intensity Ratio asks how much solving a user does relative to how often they compete:

$$IR_i = P_{month,i} / (C_{month,i} + e) \quad (1)$$

where P_month is problems solved per month, C_month is contests entered per month, and e = 0.1 guards the denominator against users with no recent contest activity. A high value means the user is solving a great deal relative to how much they compete, which is the pattern we associate with self-directed practice.

The Editorial Dependency Index encodes a suspicion we had going in, that attrition-bound users upsolve often but without much independent effort:

$$EDI_i = U_i \times dT_{struggle,i} \quad (2)$$

where U_i is upsolving frequency per contest and dT_struggle is the mean time to acceptance, itself defined as the mean of T_AC minus T_first over all accepted problems. A high EDI sitting next to low contest frequency and a low Intensity Ratio describes a student who reads the solution after every round and is not otherwise building capacity.

Decay in activity is captured by the Activity Trend Ratio, the ratio of submissions in the recent three months to those in the prior three, with anything under 1.0 indicating a slowdown. Three columns were deliberately kept out: months since last submission encodes the label definition directly, peak rating appears in the label threshold, and current rating is not measured at a comparable point in time across the two groups.

From the survey, eight features were extracted: mathematics self-rating and data structure understanding, both self-rated 1 to 5; course grade in Data Structures and Algorithms; upsolving habit; mentor support; long break of fifteen or more consecutive days; prior consideration of quitting; and programming experience before university. Semester and cumulative grade average were available but excluded, as neither is plausibly connected to CP attrition except through variables already in the model.

Models and Training Protocol

Nine architectures were evaluated: logistic regression with L2 regularization, a Gini-split decision tree, Random Forest with 200 estimators, k-nearest neighbors at k = 5, an RBF-kernel support vector machine, XGBoost [13], LightGBM [14], a multi-layer perceptron with hidden layers of 64 and 32 units and ReLU activations, and a soft-voting ensemble over the first, third, fifth and sixth of these. The ensemble combines member probabilities as a weighted sum of the individual attrition probabilities, with weights tuned by cross-validation.

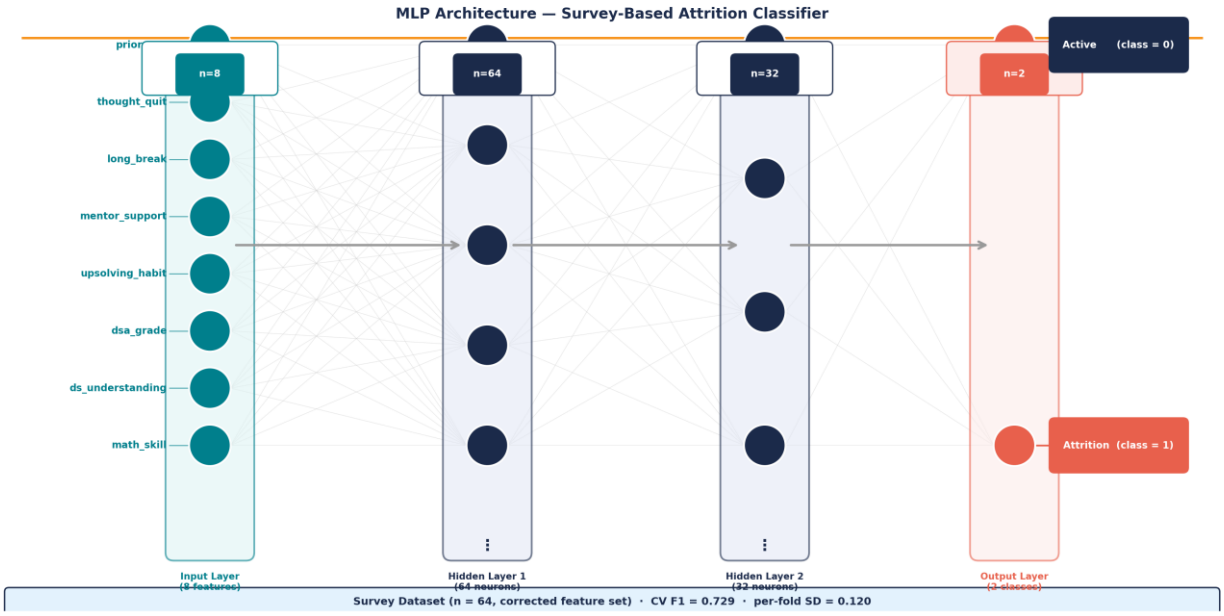


Fig. 2. Multi-layer perceptron architecture for the survey classifier. Eight psychographic inputs feed two hidden layers of 64 and 32 ReLU units, with a softmax output over the two classes.

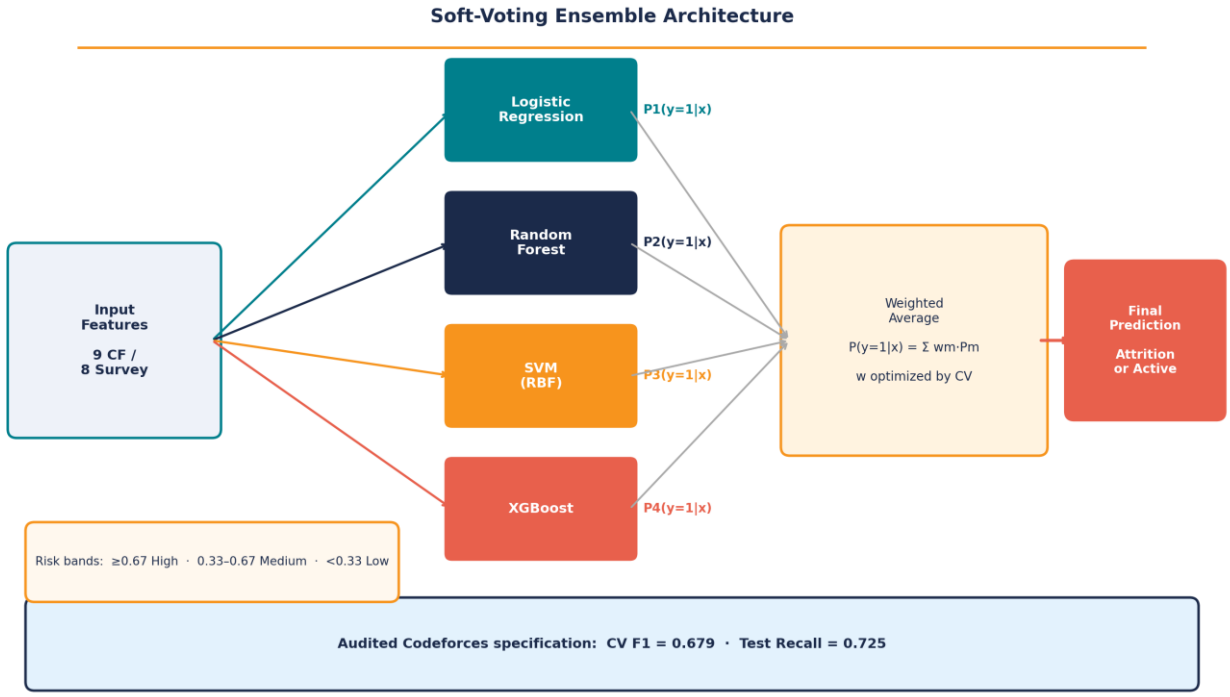


Fig. 3. Soft-voting ensemble. Four base classifiers produce independent attrition probabilities which are then averaged under cross-validated weights.

All preprocessing and modeling pipelines were implemented using scikit-learn [21].Every classifier ran with balanced class weights to absorb residual imbalance, with XGBoost using a scaled positive weight for the same purpose. Preprocessing was median imputation followed by standardization on numeric columns and one-hot encoding on categorical ones. Selection and evaluation used stratified five-fold cross-validation [29] inside an 80/20 split. We treated F1 as the headline metric because the two error types are not equally costly here: failing to flag a student who is about to quit is worse than an unnecessary check-in.

IV. Results

Behavioral Trends Preceding Attrition

Active users entered 7.27 contests per month on average. True attrition users entered 1.19, a drop of 83.7 percent ($p < 0.001$, independent samples t-test). Of everything we measured, this is the clearest advance signal.

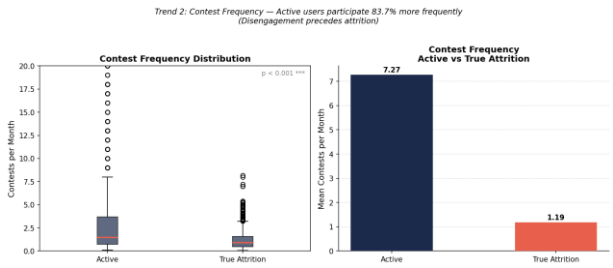


Fig. 4. Contest participation frequency, active versus true attrition ($p < 0.001$). The 83.7 percent gap is the strongest single behavioral indicator we found.

Reaching an accepted verdict took active users a mean of 2,646 minutes and true attrition users 3,059, a 15.6 percent increase. The direction matches what we expected, but the difference does not clear conventional significance ($p = 0.213$). Within-group variance is high, driven by a handful of accounts with extreme struggle times, and we report the result as inconclusive rather than supporting.

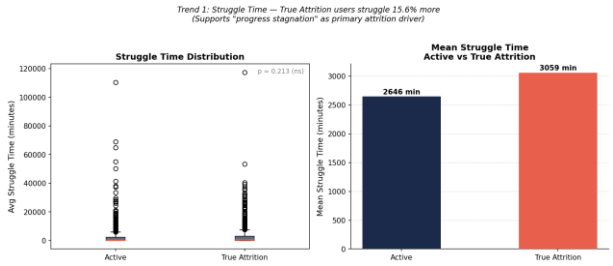


Fig. 5. Distribution and means of struggle time. The 15.6 percent increase points the right way but does not reach significance.

Attrition users underperformed on every skill-linked metric we computed: math solve rate 0.469 against 0.538, data structure solve rate 0.400 against 0.480, and problems per month 7.86 against 13.70, a gap of 42.7 percent. All three point the same way.

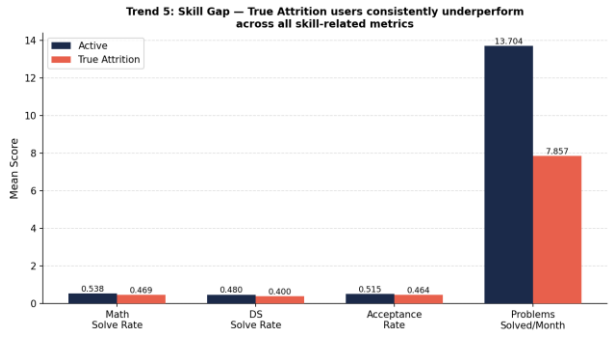


Fig. 6. Skill gap between the two cohorts. Problems solved per month shows the widest gap at 42.7 percent.

Burnout, defined as activity falling by more than half over the final three months, occurred in 45.0 percent of attrition users against 29.8 percent of active users, an incidence 51 percent higher.

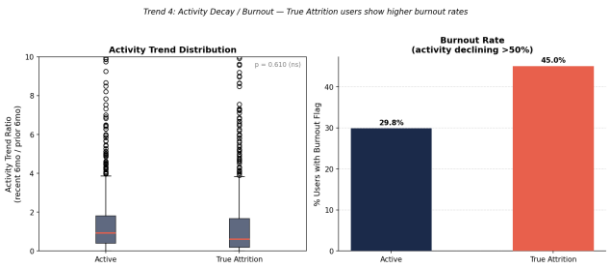


Fig. 7. Activity decay. Burnout rates run 51 percent higher in the true attrition cohort.

Upsolving is rare in both groups: 68.0 percent of active users and 67.2 percent of attrition users fall into the never band. Among those who did upsolve, though, the attrition group scored slightly higher, 2.94 against 2.81 on the 1 to 5 scale. More upsolving alongside weaker performance is odd on its face, and it is the pattern the Editorial Dependency Index was formulated to describe. Students who are struggling may be consulting editorials after each contest, which registers in the logs as upsolving activity without involving the independent work that would actually build skill. The logs do not record whether an editorial was opened, so this remains an interpretation consistent with the pattern rather than a demonstrated mechanism.

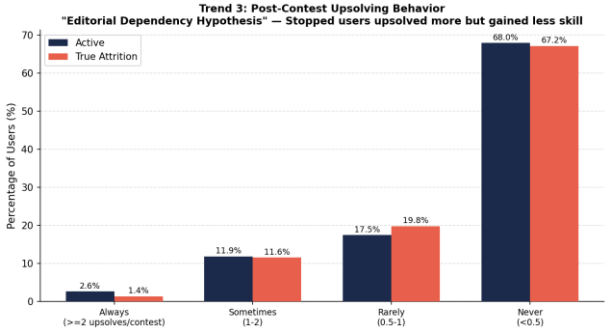


Fig. 8. Post-contest upsolving frequency. Both cohorts cluster in the never band; among those who do upsolve, attrition users score marginally higher.

The Skill-Application Paradox

The survey modeling set produced the result that surprised us most. Students who had stopped CP rated their mathematical ability at 3.93 out of 5, against 3.41 among students still active ($t = -2.31$, $p = 0.027$, $d = -0.65$). Data structure understanding followed the same direction, 3.52 against 3.09, though that comparison does not reach significance. The group that quit is the group that feels more capable.

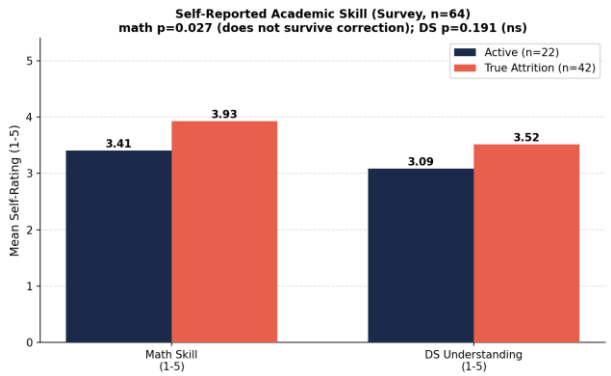


Fig. 9. Self-reported academic skill among active and true attrition respondents. The mathematics gap is nominally significant but does not survive correction for multiple comparisons.

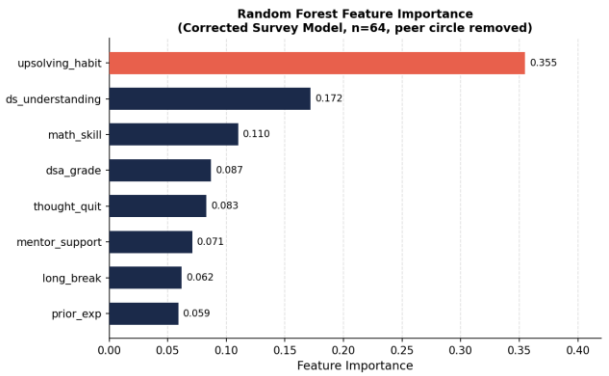


Fig. 10. Random Forest feature importance on the corrected survey model. Upsolving habit alone carries 35.5 percent of the predictive weight.

We call this the Skill-Application Paradox, and it is important to be precise about how much of it the data will bear. The behavioral half is solid. Upsolving habit separates the groups at chi-square 23.18, $p < 0.001$, with a large effect that survives both corrections. Students who stopped were markedly less likely to practise after contests, and this is the single most robust group difference in the survey.

The confidence half is weaker. The mathematical self-rating gap is a medium effect but nominal only at $p = 0.027$, and it does not survive correction for the eight tests we ran; the data structure gap does not reach nominal significance at all. We therefore state the paradox as one confirmed asymmetry and one consistent tendency: students who stopped practised significantly less, and rated their own ability at least as high as the active group did, with the confidence difference pointing consistently upward across both items without being individually established at this sample size.

Most of the respondents who stopped had earned an A or A+ in their Data Structures and Algorithms course, so the confidence is not baseless. Our reading is that strong coursework performance produces confidence in theoretical command, and that confidence makes the independent practice CP demands feel less necessary than it is. The gap surfaces the moment a live contest asks for a familiar concept in an unfamiliar arrangement. Rating flattens, frustration builds, and the student leaves. We offer this as the interpretation most consistent with the pattern, not as a mechanism the survey can establish.

Feature importance supports this reading. Upsolving habit dominates at 35.5 percent of the Random Forest's total weight, more than twice the next feature. Data structure understanding follows at 17.2 percent, mathematical self-rating at 11.0 percent, and course grade at 8.7 percent. The ordering places a behavioral habit above every measure of knowledge or attainment in the questionnaire.

Reasons for Discontinuation

RQ1 asks what reasons students give themselves, and we should be explicit about how thin our instrument is for answering it. Q31 offered a closed list of six options with a single selection permitted; the full instrument is reproduced in Appendix B. There were no free-text responses and no interviews. What we have is a frequency distribution over researcher-supplied categories, not a qualitative analysis in any recognised sense.

Read at that level, the distribution is still informative. Stalled progress was the most common selection at 12 of 51, followed by an unresolved other category and a shift to other technical tracks at 9 each, lack of guidance at 8, academic pressure at 7, and burnout at 6. Attrition is spread across several causes rather than dominated by one, and the leading category is consistent with the behavioral pattern in the platform data. Establishing why students leave would require open-ended responses or semi-structured interviews analysed under a stated coding procedure.

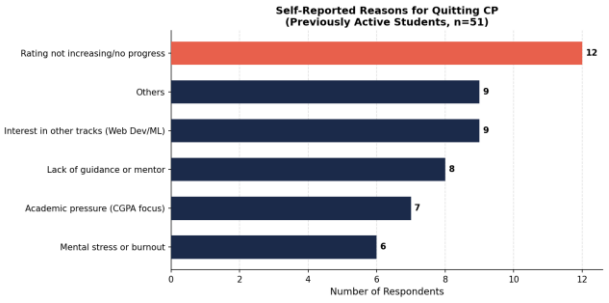


Fig. 11. Self-reported reasons for quitting. Stalled progress leads. Respondents citing a voluntary move to another track were excluded from the true attrition class.

Classifier Performance and Leakage Audit

Before reporting performance we have to deal with a problem in our own feature set. Three of the Codeforces features are not behavioral measurements in the sense the rest are. Account age records how long ago an account was created and says nothing about what the user did. Contests per month and the activity trend ratio are both normalized against a window anchored to T_obs, which means that an account already dormant for a year enters the calculation with a denominator shaped by its dormancy. All three can carry information about when a user stopped rather than the behavior preceding it, and a classifier is free to exploit that. This is leakage in the sense of Kaufman et al. [30]: information about the target enters through how the data

were assembled rather than through anything the subject did.

We therefore report three specifications. The first is the original unmatched comparison, included so the effect of the confound is visible. The second matches on peak rating but keeps all twelve features. The third matches on peak rating and drops the three temporal features, leaving only measurements of what users actually did while they were active. Table III reports all three with per-fold standard deviations.

TABLE III
Codeforces Classifier Performance. (A) Unmatched; (B) Rating-Matched; (C) Rating-Matched and Leakage-Audited

Classifier	(A) CV F1	(B) CV F1	SD	(C) CV F1	SD	Acc	F1	Rec
Logistic Regression	0.704	0.716	0.030	0.639	0.037	0.555	0.599	0.665
Decision Tree	0.621	0.655	0.028	0.568	0.043	0.558	0.561	0.566
Random Forest	0.731	0.775	0.022	0.682	0.011	0.613	0.650	0.720
KNN	0.667	0.676	0.025	0.589	0.012	0.566	0.603	0.659
SVM	0.732	0.743	0.027	0.652	0.022	0.593	0.656	0.775
XGBoost	0.726	0.755	0.018	0.652	0.027	0.643	0.665	0.709
LightGBM	0.736	0.751	0.018	0.645	0.024	0.621	0.646	0.692
MLP	0.712	0.720	0.030	0.608	0.026	0.574	0.619	0.692
Soft-Voting Ensemble	0.737	0.767	0.015	0.679	0.013	0.626	0.660	0.725

The pattern across the three columns is informative in itself. Matching on rating does not weaken the models; it strengthens them, with the soft-voting ensemble rising from 0.737 to 0.767 and Random Forest to 0.775. Whatever the original specification was picking up, it was not simply skill level. Removing the temporal features is where performance falls, to 0.679 for the ensemble and 0.682 for Random Forest. The three columns were therefore worth roughly 0.09 F1, about an eighth of the separability available in the matched specification.

We take the audited figures as the honest headline. A cross-validation F1 of 0.682 on a balanced two-class problem is a real effect, but it is more modest than the 0.775 the unaudited specification suggests, and readers should calibrate against the smaller number. The gap between the two columns quantifies how much apparent predictive power in this kind of study can come from timing artifacts rather than behavior.

Under the audited specification Random Forest leads, with test recall of 0.720, catching a little under three quarters of the true attrition cases in the held-out portion. Per-fold standard deviations stay between 0.011 and 0.043, so the ordering is stable even though the absolute level is modest. The tree ensembles hold up better than the linear and distance-based models once the temporal columns are gone, which suggests that what remains of the signal is interactive rather than additive.

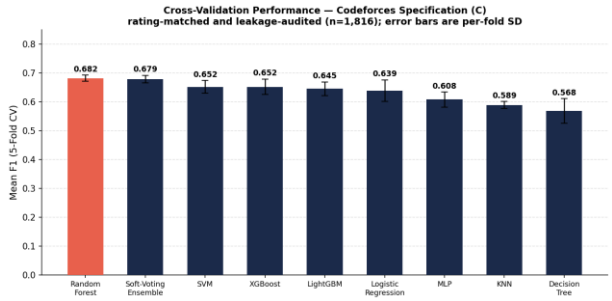


Fig. 12. Five-fold cross-validation F1 on the Codeforces dataset under the audited specification. Error bars are per-fold standard deviations.

On the survey layer, logistic regression reached 0.833 with Random Forest level at 0.832. Our earlier specification put Random Forest at 0.924, and the difference is instructive: peer circle density was available only on the stopped branch, so including it handed the model the outcome directly. Removing it costs nine points of F1, and that loss is the honest measure of how much of the original figure was structural rather than predictive. A regularized linear classifier matching a 200-tree ensemble, with a third of its per-fold variance, is what we would expect when eight features and 64 cases leave little genuine interaction structure to find.

TABLE IV
Survey Classifier Performance (n = 64, Peer Circle Removed)

Classifier	CV F1	SD
Logistic Regression	0.833	0.036
Random Forest	0.832	0.082
Soft-Voting Ensemble	0.805	0.046
KNN	0.798	0.048
Decision Tree	0.772	0.100
XGBoost	0.753	0.073
SVM	0.750	0.069
MLP	0.729	0.120
LightGBM	0.651	0.098

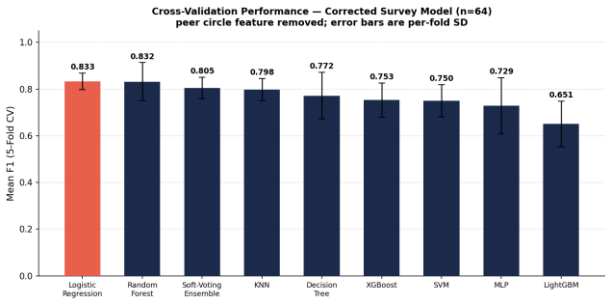


Fig. 13. Five-fold cross-validation F1 on the corrected survey dataset, after removal of the branch-confounded feature.

What the Branch-Alignment Correction Changed

Because the correction described above altered several headline numbers, we state its effect plainly. Two survey features were originally drawn from the stopped branch alone. Break history was taken from Q36 without pairing it to the active-branch equivalent Q27, and peer circle density from Q39, for which no active-branch item exists at all. In both cases every active respondent was missing and every stopped respondent was present, so a chi-square test recovered the branching structure of the questionnaire

rather than any difference in behavior. The resulting statistics, 73.00 and 60.46, are artifacts of that separation; the first equals the sample size exactly, which is the signature of the problem.

Correcting them gives two different outcomes. Break history becomes testable once Q27 and Q36 are combined, and the difference is then small and not significant: 73 percent of the attrition group reported a long break against 59 percent of the active group, chi-square 0.11, $p = 0.745$. The direction matches our hypothesis; the magnitude does not support it at this sample size. Peer circle density cannot be repaired, since the item was never put to active respondents, so we drop it from both the tests and the model.

Feature Importance on the Behavioral Layer

Account tenure ranks first at 14.3 percent in the unaudited model. In an earlier draft we read this as evidence of persistence, on the reasoning that a long-tenured account still submitting has demonstrated commitment. That interpretation does not hold. Account age is fixed at registration and updated by the calendar rather than by anything the user does, so it cannot be a behavioral measurement, and its predictive weight is better explained by the fact that attrition accounts in our sample were created more recently than active ones. Contest frequency sits second at 12.4 percent and carries a related problem, since its denominator is anchored to T_obs rather than to the span over which the user was active.

Both columns were removed in the audited specification for these reasons. What survives the removal is the more interesting result. The Intensity Ratio ranks third at 11.3 percent in the unaudited model and becomes the top-ranked feature once the temporal columns are gone, ahead of the raw problem counts from which it is derived. That is the evidence we were looking for when we constructed it: the balance between independent practice and contest participation carries information that neither component conveys on its own, and unlike tenure or contest frequency it is computed entirely from what a user did rather than from when they did it.

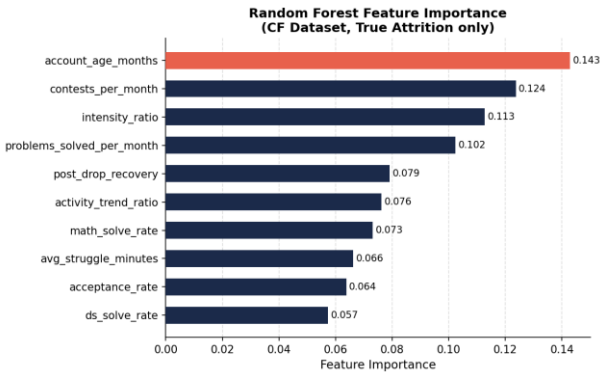


Fig. 14. Random Forest feature importance on the unaudited Codeforces feature set. Account tenure and contest frequency lead but are both temporally confounded and were dropped in the audited specification.

Early Warning System Deployment

We then ran the survey-trained Random Forest pipeline over the 22 students in our cohort who were still active, as a proof-of-concept Early Warning System. The model returns a continuous risk score, bucketed into high at or above 67 percent, medium between 33 and 67, and low below 33.

Scores are computed out of fold. Each student is scored by a model fitted on the four folds that exclude them, so no student contributes to the model that assesses them; scoring subjects with a model that has already seen them yields an optimistically biased estimate of how the system would behave on new students [29]. Our earlier specification scored all 22 with a model trained on the full cohort, which made every score an in-sample fit and inflated the apparent separation.

Six students landed in high risk, ten in medium, and six in low. Out-of-fold scoring shifts students out of the low band and into medium relative to the in-sample version, which is the expected direction: a model that has not seen a student is less confident that the student resembles the active class.

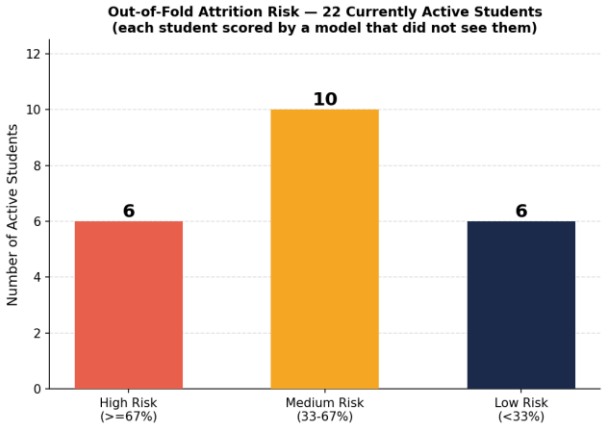


Fig. 15. Out-of-fold risk distribution across 22 active students: six high, ten medium, six low. No longitudinal validation exists yet.

The six high-risk profiles share a shape. Four of the six held an A or A+ in their algorithms course. All six reported no institutional mentor. None of the six had ever seriously considered quitting. Upsolving was irregular in every case. Scores ran from 68 to 97 percent. That is the Skill-Application Paradox signature appearing in students who have not yet left: strong attainment, no mentor, no intention of stopping, and a practice habit that has already lapsed. We would not read these flags as predictions. They indicate who might benefit from someone reaching out before the pattern completes itself.

V. Discussion

What the Two Layers Say Together

We described this work in an earlier draft as a framework in which the two layers converge. That word claims more than the design supports, and we withdraw it. The layers share no respondents, no features and no labels. The Codeforces cohort is a global sample of anonymous accounts labeled by an inactivity proxy; the survey cohort is 64 students at Bangladeshi universities labeled by self-report. Nothing links an individual in one to an individual in the other, and no statistic computed on one constrains a statistic computed on the other.

What we can claim is weaker and worth stating precisely. Two independent studies, on disjoint populations with disjoint instruments, return findings that point the

same way. Platform data shows disengagement arriving after contest participation falls. Survey data shows the students who stopped practising least consistently between contests, with upsolving the one feature that separates the groups under correction. That both point to practice frequency between contests is a replication of direction, not a joint result, and its evidential value lies exactly in the independence of the two measurements: a bias in the Codeforces inactivity proxy has no route to the survey, and recall bias among respondents who stopped has no route to the platform logs.

Practical Implications

If a university programming club wanted to act on one thing from this study, we would suggest upsolving discipline. It outranks academic performance in our models and a club can influence it directly. Running more contests does not address it; structured post-contest review sessions, mentor pairings, and checking in on students returning from a long break would.

Limitations

Several things constrain what we can claim. The survey modeling set is small at 64 and confined to Bangladeshi universities, so generalization is untested. The Codeforces labels rest on an inactivity proxy that cannot separate a permanent departure from a semester spent on coursework. The survey model's corrected F1 still likely overstates what it would achieve elsewhere.

There is also a bias we cannot design away. Features such as upsolving habit and prior consideration of quitting were reported by students who had already stopped, which means we are asking people to recall their past behavior from inside their current disengagement. If that colors their answers systematically, the separation we observe between the cohorts on self-reported measures is inflated to an unknown degree.

Finally, the Early Warning System run over 22 students is a demonstration and nothing more. We have no follow-up data confirming whether the six flagged students subsequently disengaged or their performance declined. Both datasets are observational throughout, so nothing reported here supports a causal claim.

VI. Conclusion

This paper set out to study attrition in competitive programming using two independent sources: Codeforces behavioral logs after filtering, rating matching and balancing, and a survey across ten universities in Bangladesh. The behavioral layer confirmed that attrition follows an 83.7 percent collapse in contest participation and consistent underperformance across skill-linked metrics, and these differences persist when the two cohorts are held to the same peak-rating ceiling.

The Skill-Application Paradox held up asymmetrically. Its behavioral half is firm: upsolving habit separates the cohorts at $p < 0.001$ with a large effect, surviving correction for eight tests, and it is the dominant feature in the survey model at 35.5 percent. Its confidence half is directional only, with the mathematics gap nominal at $p = 0.027$ and not surviving correction.

On the modeling side, Random Forest performed best on both layers: 0.682 on the audited behavioral specification and 0.832 on the survey layer, where regularized logistic regression matched it. We want to be direct about the first of those numbers. Retaining the three temporally confounded features would have let us report 0.775, and the decision to audit them out costs about an eighth of the apparent separability. We think the smaller number is the one that means something. The Intensity Ratio becomes the highest-ranked behavioral feature once the audit is applied.

We offer these results as an exploratory baseline rather than a finished system. The work we most want to do next is longitudinal: larger cohorts followed across universities and academic years, repository activity and club attendance folded into the feature set, and a prospective trial of the Early Warning System with a partner programming club. A second priority is qualitative. RQ1 asks why students leave, and a closed six-option item cannot answer that; semi-structured interviews with a subset of the cohort that stopped, coded under a stated procedure with inter-coder reliability reported, would replace the frequency distribution we offer here with something that genuinely addresses the question.

Appendix A: Complete Feature Specification

Table V lists every feature entering either model, its definition, its source, and whether it is retained in the specification we report. Exclusion reasons are keyed as follows. (a) fixed at registration and therefore not a behavioral measurement; (b) denominator or window anchored to the observation date rather than to the user's active span; (c) directly encodes the label definition; (d) sets the label threshold; (e) not time-comparable across cohorts; (f) no active-branch item exists, so the feature encodes group membership; (g) not causally linked to attrition except through variables already included.

TABLE V
Complete Feature Specification for Both Layers

Feature	Definition	Source	Kept
CODEFORCES LAYER			
math_solve_rate	Share of attempted math-tagged problems accepted	API submissions	yes
ds_solve_rate	Share of attempted data-structure problems accepted	API submissions	yes
upsolves_per_contest	Problems solved after a contest closes, per contest	API submissions	yes
avg_struggle_minutes	Mean time from first attempt to acceptance	API submissions	yes
post_drop_recovery	Submissions in the 30 days after a rating fall	API rating history	yes
problems_solved_per_month	Accepted problems divided by account tenure	API submissions	yes
acceptance_rate	Accepted submissions over all submissions	API submissions	yes
intensity_ratio	P_month / (C_month + 0.1), Eq. (1)	engineered	yes
editorial_dependency_index	U x dT_struggle, Eq. (2)	engineered	yes
account_age_months	Months since account registration	API profile	no (a)

Feature	Definition	Source	Kept
contests_per_month	Contests entered divided by tenure	API contest history	no (b)
activity_trend_ratio	Recent 3 months over prior 3 months	engineered	no (b)
months_since_last_submission	Months of inactivity	API submissions	no (c)
max_rating	Peak rating attained	API profile	no (d)
current_rating	Most recent rating	API profile	no (e)
SURVEY LAYER			
math_skill	Self-rated mathematics ability, 1 to 5	Q5, both branches	yes
ds_understanding	Self-rated data-structure understanding, 1 to 5	Q14 / Q37	yes
dsa_grade	Grade in the algorithms course	Q4, both branches	yes
prior_exp	Programming experience before university	Q3, both branches	yes
mentor_support	Access to a mentor or senior guide	Q22 / Q35	yes
long_break	Any break of 15+ consecutive days	Q27 / Q36	yes
thought_quit	Ever seriously considered quitting	Q28 / Q34	yes
upsolving_habit	Post-contest practice frequency, four levels	Q11 / Q33, mapped	yes
friend_circle_cp	Peers in the respondent's circle doing CP	Q39, stopped only	no (f)
semester, CGPA	Academic progress measures	Q1, Q2	no (g)

APPENDIX B: SURVEY INSTRUMENT

The questionnaire opens with six items put to every respondent, then branches on Q6. Items marked with an asterisk enter the feature set. Response options are given where the item is closed.

Common Items (All Respondents)

Q1 current semester. Q2 current CGPA. Q3* programming experience before university (yes / no). Q4* grade in the Data Structures and Algorithms course (A+ or A / B+ or B / C or below / not yet taken). Q5* self-rated mathematics ability (1 to 5). Q6 current CP status (currently active / previously active but stopped / never started), which determines the branch.

Active Branch (Q7 to Q29, n = 22)

Q7 hours per week on CP. Q8 platforms used. Q9 current rating. Q10 contests entered per month. Q11* upsolving after contests (always / sometimes / rarely / never). Q12 new problems solved per month. Q13 time on a problem before opening the editorial. Q14* understanding of standard data structures (1 to 5). Q15 comfort with dynamic programming and graph problems. Q16 language used. Q17 mental state when rating falls. Q18 frequency of burnout. Q19 demotivation from comparing ratings. Q20 motivation for doing CP. Q21 whether fear of failure deters contest entry. Q22* access to a mentor or senior (yes / no). Q23 quality of club support. Q24 internet or hardware constraints. Q25 effect of academic pressure. Q26 family pressure toward grades. Q27* any break of 15 or more consecutive days in the last three months (yes / no). Q28* ever seriously considered quitting (yes / no). Q29 expectation of still being active in six months.

Stopped Branch (Q30 to Q39, n = 51)

Q30 duration of practice before stopping. Q31 main reason for stopping, a single selection from: rating not increasing or no progress; lack of guidance or mentor; academic pressure with CGPA focus; mental stress or burnout; interest in other tracks such as web development or machine learning; others. Q32 highest rating before stopping. Q33* practice habit in the final month (regular / became irregular / stopped completely). Q34* ever seriously considered quitting (yes / no). Q35* had mentor or senior support (yes / no). Q36* any break of 15 or more consecutive days during the CP journey (yes / no). Q37* understanding of standard data structures at the point of stopping (1 to 5). Q38 whether stopping was foreseeable six months earlier. Q39 peers in the friend circle doing CP.

Never-Started Branch (Q40 to Q49, n = 23)

Excluded from all modeling and all tests reported here.

Branch Alignment

Q11 and Q33 are mapped to a common four-level scale: regular maps to sometimes, became irregular to rarely, and stopped completely to never. Q14 pairs with Q37, Q22 with Q35, Q27 with Q36, and Q28 with Q34. Q31 is used for label refinement, with interest in other tracks classifying an intentional exit. Q39 has no active-branch counterpart and is therefore excluded from both the statistical tests and the model.

ACKNOWLEDGMENT

The authors thank the students from the ten universities in Bangladesh who took the time to answer the survey, and the Research Ethics Committee of Metropolitan University for reviewing the study. We are also grateful to the two anonymous reviewers, whose detailed comments led to real corrections in this version of the paper.

AUTHOR CONTRIBUTIONS

Azuad Islam Ruhan led the study from start to finish. He fetched the whole Codeforces dataset on his own, carried out most of the student survey data collection, built the processing and modelling pipelines, ran the experiments and the early-warning analysis, wrote the first draft of the manuscript, prepared the revision, and handled all correspondence with the journal.

Golam Mostofa Naeem supervised the work. He helped shape the research idea and the methodology, guided the study from start to finish, and reviewed and revised the manuscript for technical accuracy and clarity.

Rakibul Islam Rafi helped with collecting the student survey responses and reviewed the manuscript.

Sherin Afrin Mim provided general support during the study and reviewed the manuscript.

Nazira Bani Opi assisted with supporting tasks during the study and reviewed the manuscript.

Dewan Fahad Chowdhury provided limited support during the study and reviewed the manuscript.

Md. Rejaul Korim Sadi contributed to the conceptualization and design of the research framework, guided the data modelling, and led the processing and analysis of the Codeforces behavioral logs. He also helped compile the survey results and carried out a major structural revision of the final manuscript.

All authors read and approved the final manuscript.

AI Tool Usage Acknowledgement

Generative AI tools were used only in a supporting role while preparing this work. ChatGPT (OpenAI) and Grammarly were used for language editing and for tightening the wording of some passages, and ChatGPT was also used while refining the research topic and structuring the initial proposal. Google Scholar was used to find and check literature. The research design, data collection, analysis, results and conclusions are the authors' own. All AI-assisted text was read and revised by the authors, who take full responsibility for the accuracy and originality of the work.

VII. References

[1] A. Laaksonen, Guide to Competitive Programming: Learning and Improving Algorithms Through Contests. Springer, 2017.

[2] S. S. Skiena and M. A. Revilla, Programming Challenges: The Programming Contest Training Manual. Springer, 2003.

[3] S. Combéfis, G. Beresnevičius, and V. Dagienė, "Learning programming through games and contests: Overview, characterisation and discussion," Olympiads in Informatics, vol. 10, pp. 39–60, 2016.

[4] S. Halim and F. Halim, Competitive Programming 3: The New Lower Bound of Programming Contests. Lulu, 2013.

[5] L. Malmi, A. Korhonen, and R. Saikkonen, "Experiences in automatic assessment on mass courses and issues for designing virtual courses," in Proc. ITiCSE, 2005, pp. 233–237.

[6] M. Guzdial, "Learner-centered design of computing education," Synthesis Lectures on Human-Centered Informatics, vol. 8, no. 6, pp. 1–165, 2015.

[7] A. Robins, J. Rountree, and N. Rountree, "Learning and teaching programming: A review and discussion," Comput. Sci. Educ., vol. 13, no. 2, pp. 137–172, 2003.

[8] C. Romero and S. Ventura, "Educational data mining: A review of the state of the art," IEEE Trans. Syst. Man Cybern. C, vol. 40, no. 6, pp. 601–618, 2010.

[9] S. B. Kotsiantis, "Supervised machine learning: A review of classification techniques," Informatica, vol. 31, no. 3, pp. 249–268, 2007.

[10] P. Ihantola et al., "Educational data mining and learning analytics in programming: Systematic literature review and calls for action," in Proc. ITiCSE Working Group Reports, 2015, pp. 41–63.

[11] R. S. J. d. Baker and K. Yacef, "The state of educational data mining in 2009: A review and future visions," J. Educ. Data Min., vol. 1, no. 1, pp. 3–17, 2009.

[12] L. Breiman, "Random forests," Mach. Learn., vol. 45, no. 1, pp. 5–32, 2001.

[13] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. 22nd ACM SIGKDD, 2016, pp. 785–794.

[14] G. Ke et al., "LightGBM: A highly efficient gradient boosting decision tree," in Advances Neural Inf. Process. Syst., vol. 30, 2017, pp. 3146–3154.

[15] C. Watson, F. W. B. Li, and J. L. Godwin, "Predicting performance in an introductory programming course by logging and analyzing student programming behavior," in Proc. IEEE 13th Int. Conf. Advanced Learning Technologies (ICALT), 2013, pp. 319–323.

[16] J. Sweller, "Cognitive load during problem solving: Effects on learning," Cogn. Sci., vol. 12, no. 2, pp. 257–285, 1988.

[17] E. L. Deci and R. M. Ryan, Intrinsic Motivation and Self-Determination in Human Behavior. Springer, 1985.

[18] A. Bandura, "Self-efficacy: Toward a unifying theory of behavioral change," Psychol. Rev., vol. 84, no. 2, pp. 191–215, 1977.

[19] R. Raman, H. Vachharajani, and K. Achuthan, "Students motivation for adopting programming contests: Innovation-diffusion perspective," Educ. Inf. Technol., vol. 23, no. 5, pp. 1919–1932, 2018.

[20] E. B. Costa, B. Fonseca, M. A. Santana, F. F. de Araújo, and J. Rego, "Evaluating the effectiveness of educational data mining techniques for early prediction of students' academic failure in introductory programming courses," Comput. Hum. Behav., vol. 73, pp. 247–256, 2017.

[21] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.

[22] P. Prinsloo and S. Slade, "Student vulnerability, agency, and learning analytics: An exploration," J. Learn. Anal., vol. 3, no. 1, pp. 159–182, 2016.

[23] O. J. Dunn, "Multiple comparisons among means," J. Amer. Statist. Assoc., vol. 56, no. 293, pp. 52–64, 1961.

[24] Y. Benjamini and Y. Hochberg, "Controlling the false discovery rate: A practical and powerful approach to multiple testing," J. Roy. Statist. Soc. B, vol. 57, no. 1, pp. 289–300, 1995.

[25] B. L. Welch, "The generalization of Student's problem when several different population variances are involved," Biometrika, vol. 34, no. 1–2, pp. 28–35, 1947.

[26] J. Cohen, Statistical Power Analysis for the Behavioral Sciences, 2nd ed. Hillsdale, NJ: Lawrence Erlbaum, 1988.

[27] H. Cramér, Mathematical Methods of Statistics. Princeton, NJ: Princeton Univ. Press, 1946.

[28] B. Efron and R. J. Tibshirani, An Introduction to the Bootstrap. New York: Chapman and Hall, 1993.

[29] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in Proc. 14th Int. Joint Conf. Artificial Intelligence (IJCAI), 1995, pp. 1137–1143.

[30] S. Kaufman, S. Rosset, C. Perlich, and O. Stitelman, "Leakage in data mining: Formulation, detection, and avoidance," ACM Trans. Knowl. Discov. Data, vol. 6, no. 4, art. 15, pp. 1–21, 2012.

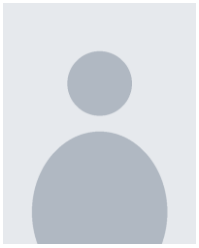
[31] H. He and E. A. Garcia, "Learning from imbalanced data," IEEE Trans. Knowl. Data Eng., vol. 21, no. 9, pp. 1263–1284, 2009.

Author Biographies

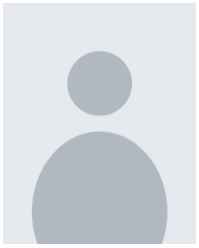
Azuad Islam Ruhan is an undergraduate researcher in the Department of Computer Science and Engineering at Metropolitan University, Sylhet, Bangladesh. His research focuses on machine learning for behavioral prediction, natural language processing for Bangla, and educational technology for rural learners. An active competitive programmer with more than 1,000 solved problems, he is currently working on his undergraduate thesis, which draws on Cortical Labs' work on biological computing. He plans to pursue graduate studies in artificial intelligence and machine learning.

Golam Mostofa Naeem is a Senior Lecturer in the Department of Computer Science and Engineering at Metropolitan University, Bangladesh. His research interests are machine learning, natural language processing, explainable artificial intelligence, and deep learning, with a focus on AI methods that are interpretable and reliable enough to be used on real problems. Alongside his research, he teaches and works with students on academic projects.

Rakibul Islam Rafi is a final-year undergraduate student in the Department of Computer Science and Engineering at Metropolitan University, Sylhet, Bangladesh. His interests are artificial intelligence, computer vision, machine learning and deep learning, particularly image analysis and automated defect detection. He wants to build practical AI solutions to real-world problems and to take up research in computer vision and intelligent systems.



Sherin Afrin Mim is a final-year undergraduate student in Computer Science and Engineering at Metropolitan University, Bangladesh. Her research interests include natural language processing, machine learning, artificial intelligence and computational linguistics. She is also interested in compiler construction, algorithm design, and data-driven approaches to real-world computational problems.



Nazira Bani Opi is an undergraduate student in the Department of Computer Science and Engineering at Metropolitan University, Bangladesh. Her research interest is machine learning and how it is applied in computer science. She is keen to learn about new technologies and about how artificial intelligence can be used to solve real-world problems, and she hopes to build her research skills and continue working in machine learning.

Dewan Fahad Chowdhury is an undergraduate student of Computer Science and Engineering at Metropolitan University, Sylhet, Bangladesh. His research interests include machine learning, artificial intelligence, natural language processing and computer vision, particularly model evaluation, benchmarking and real-world data analytics. He has worked on academic projects in applied machine learning and software systems, and aims to build computational solutions that are scalable and easy to interpret.

Md. Rejaul Korim Sadi is an undergraduate researcher in the Department of Computer Science and Engineering at Metropolitan University, Sylhet. His research interests are natural language processing, machine learning and trustworthy AI, with a particular focus on the structural causes of hallucination in large language models. His work tries to pair theory with controlled empirical measurement.