SYNTIA – Systems and Technologies in Artificial Intelligence Applications.
Volume 1 Issue 2 (June 2026). Page: 67-78.

Random Forest and XGBoost for Mobile-Number Validity Classification

Andi Faturrahman[1], Amriana[2*], Afiyah Rifkha Rahmika[2]

[1] *Informatics Engineering Study Program, Faculty of Engineering, Universitas Tadulako, Palu, Indonesia.*
[2] *Department of Information Technology, Faculty of Engineering, Universitas Tadulako, Palu, Indonesia.*

**Corresponding author: amriana@untad.ac.id*

Abstract: Invalid mobile numbers in registration forms cause failed notifications, and a deterministic validator catches only the cases its rules cover. This study asks whether ensemble classifiers can learn a validity rule from syntactic features, and which does it better. We built a dataset of 46,485 Indonesian numbers and labeled them by rule: prefix 08, 10–13 digits, and an allocation list for the first four digits (8.68% invalid). Random Forest and XGBoost were trained after SMOTE on 37,188 records and tested on 9,297 untouched records (807 invalid). With a binary prefix-validity flag supplied (scenario 1), both models made 53–54 errors (about 99.4% accuracy). With the third and fourth digits as numeric features instead (scenario 2), XGBoost made 9 errors (99.90%) and Random Forest 75 (99.19%); exact bounds show the difference is significant under any overlap of errors ($p < 10^{-13}$). XGBoost also trained about 36 times faster, on undocumented hardware. Because the labels come from rules that the features reproduce, the accuracies measure rule recovery, not detection of well-formed but wrong numbers; no independent ground truth or rule-based baseline was evaluated. The results support XGBoost as a learned approximation of the rule and call for validation against verified contact status.

Keywords : anomaly detection; mobile number validation; Random Forest; SMOTE; XGBoost.

Received: March 2, 2026 | Revised: April 13, 2026 | Accepted: May 11, 2026 | Published: June 30, 2026

1. Introduction

Mobile numbers collected through registration forms are used for notifications, one-time passwords, and follow-up contact, so a wrong number silently breaks these services. Invalid entries are a typical data-quality defect: the value is present but does not satisfy the rules that make it usable [1]. In a training-institution registry, typing errors, truncated numbers, placeholder strings, and invented sequences all occur. A deterministic validator based on a regular expression and a list of operator prefixes catches the cases its authors anticipated, and this raises the question of whether a learned model can add value.

Detecting records that deviate from expected behavior is the subject of anomaly detection, whose survey literature separates supervised, semi-supervised, and unsupervised settings and stresses that the meaning of "anomaly" must be defined by the application [2]. In the present case the definition is operational: a number is invalid if it violates the national numbering rules. If labels are available, the problem becomes binary classification with a strong class imbalance, since invalid numbers are a small minority. Oversampling of the minority class by synthetic interpolation, SMOTE, is the usual remedy [3], although its benefits and limits are still discussed [4], and ensemble tree classifiers such as Random Forest [5] and gradient boosting as implemented in XGBoost [6] are common choices for tabular features.

Three approaches to contact-data validation can be distinguished. Rule-based validation is exact for the rules it encodes and costs almost nothing, but it must be rewritten when the rules change. A learned classifier can approximate rules from examples and produce a score, but it is only as good as the labels. Active verification, for example by sending a message and observing delivery, is the only approach that

SYNTIA – Systems and Technologies in Artificial Intelligence Applications is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License

establishes whether a number is in use, and it is the most expensive. The present study concerns the second approach, and its main question is what a high score actually demonstrates when the labels come from the rules themselves.

Related work on telephone-number quality is mostly practical, in the form of validation libraries and regular expressions, and the machine-learning literature on data quality concentrates on detecting outliers and duplicates in larger tables [1], [2]. Studies that treat a rule-defined defect as a classification target are rarely explicit about the circularity this creates: when the label is a function of the input, high accuracy shows that the function can be approximated and says little about generalization to defects that the function does not cover. The distinction between a label that is a function of the features and a label that is an independent observation is the thread of this article.

This article reports Random Forest and XGBoost classifiers trained on 46,485 Indonesian mobile-number records described by 11 syntactic features, in two feature scenarios. We ask four questions. (RQ1) Do the two ensembles differ when a prefix-validity flag is given? (RQ2) Do they differ when the flag is withheld and the model must learn the prefix from digits? (RQ3) Which features carry the decision? (RQ4) How different are the training and prediction costs? In answering them we compute interval estimates for every proportion, exact bounds on paired significance tests from the confusion matrices, and we state explicitly what the design can and cannot show. The contributions are an evaluation with bounded significance claims, an ablation of the features, an analysis of label circularity with a list of checks that remove it, and a record of efficiency. Section 2 describes data and methods, Section 3 reports and discusses results, and Section 4 concludes.

2. Method

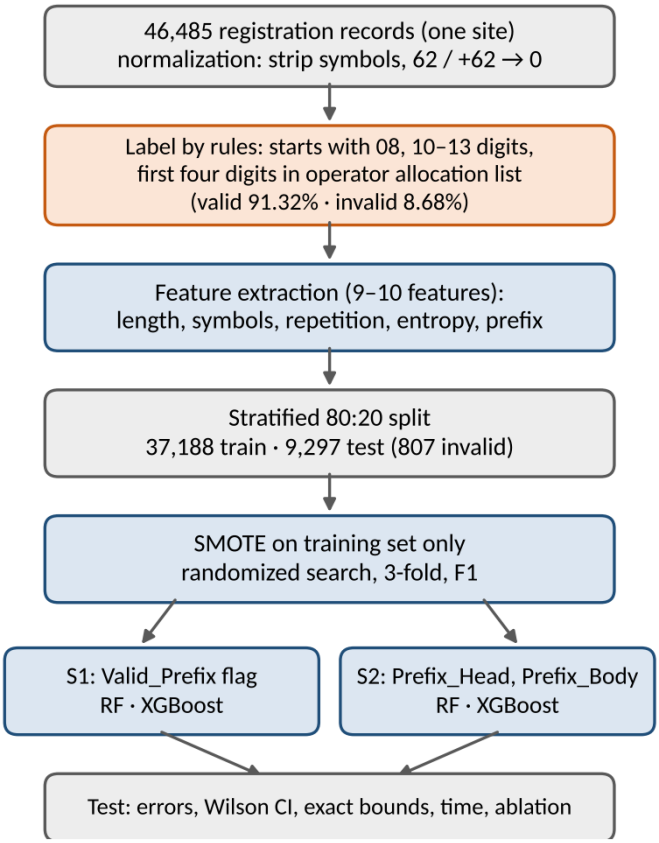


Figure 1. Research workflow.

2.1. Data and ethics

The data are 46,485 mobile-number records collected from the registration process of LPK Mitra Bandung, a training institution in Bandung, Indonesia, that partners with Kartu Prakerja, a national pre-employment program. The institution collected the records between January and December 2023 under a data-processing agreement consistent with Law No. 27 of 2022 on Personal Data Protection. A number is personal data, and the dataset is not public. This article therefore shows only synthetic examples. The institution granted permission to use the de-identified data for this study. We removed every name and identifier field before analysis, and the processed dataset is available from the corresponding author on reasonable request.

After normalization (Section 2.2), 42,452 records (91.32%) are valid and 4,033 (8.68%) are invalid. The split was stratified 80:20 with a fixed seed of 42, giving 37,188 training and 9,297 test records. The test set holds 8,490 valid and 807 invalid records. SMOTE was applied to the training set only, raising the invalid class from 3,226 to 33,962 records and giving 67,924 balanced training records (Table 1), so that the test set keeps the natural imbalance and no synthetic record reaches it [3], [7].

Table 1. Data partition and oversampling.

Set	Valid	Invalid	Total
All records	42,452	4,033	46,485
Training (before SMOTE)	33,962	3,226	37,188
Training (after SMOTE)	33,962	33,962	67,924
Test	8,490	807	9,297

2.2. Normalization and labeling

Each raw string was reduced to its digits, and a number written with the country code +62 was converted to the national form beginning with 0. The label then follows a rule with three conditions: the number begins with 08, it has between 10 and 13 digits, and its first four digits belong to the allocation list of mobile prefixes of the national regulator, which has 40 entries for six operators (nine for Telkomsel, seven for Indosat Ooredoo, six for XL, four for AXIS, five for Tri, and nine for Smartfren), published by the Ministry of Communication and Informatics in its national numbering plan regulation. With D the digit string, n its length, and $\mathcal{P}$ the prefix set,

$$y = \begin{cases} 0 \text{ (valid)} & \text{if } D_{1:2} = 08 \ \wedge \ 10 \le n \le 13 \ \wedge \ D_{1:4} \in \mathcal{P} \\ 1 \text{ (invalid)} & \text{otherwise} \end{cases} \tag{1}$$

The thesis text gives the length range as 10–13 digits in the labeling rule and 10–14 in the feature description; we use 10–13 here and flag the difference in the verification notes.

2.3. Features and scenarios

Eleven syntactic features were computed from the normalized string (Table 2). Most are simple counts. Digit dominance is the share of the most frequent digit, $d = \max_c f_c / n$, with f_c the frequency of digit c. Shannon entropy measures how evenly the digits are spread, $H = -\Sigma\, p_c \log_2 p_c$, where $p_c = f_c/n$; a repeated-digit string such as a run of one digit has entropy zero, whereas a string with ten equally frequent digits has the maximum.

$$H = -\sum_{c=0}^{9} p_c \log_2 p_c, \qquad d = \frac{\max_c f_c}{n} \tag{2}$$

Table 2. Features extracted from each record.

Feature	Description	S1	S2
Raw_Length	Characters in the raw string	✓	✓
Digit_Length	Digits after normalization	✓	✓
Non_Numeric	Count of non-digit characters	✓	✓
Valid_Prefix	1 if first four digits in the allocation list	✓	–
Prefix_Head	Digits 1–2 as an integer	–	✓
Prefix_Body	Digits 3–4 as an integer	–	✓
Unique_Ratio	Distinct digits divided by length	✓	✓
Max_Streak	Longest run of one digit	✓	✓
Sequential	Longest ascending or descending run	✓	✓
Digit_Dominance	Share of the most frequent digit	✓	✓
Entropy	Shannon entropy of digits	✓	✓

Two scenarios were defined. In scenario 1 (S1) the model receives the binary Valid_Prefix flag. In scenario 2 (S2) the flag is removed and replaced by the two prefix digit pairs as integers, so that the model must discover which of the 40 combinations are allocated. Valid_Prefix is computed from the very list used for labeling, so S1 hands the model one of the three conditions of Eq. (1) directly, and a perfect S1 model would reach 100% accuracy. S2 is the more informative setting for rule learning, because the model must reconstruct the prefix set from examples.

2.4. Oversampling, models, and tuning

SMOTE creates a synthetic minority record from a minority record x and one of its k nearest minority neighbours x_{nn} by interpolation [3],

$$\mathbf{x}_{\text{new}} = \mathbf{x} + \lambda\,(\mathbf{x}_{nn} - \mathbf{x}), \qquad \lambda \sim U(0,1) \tag{3}$$

so every synthetic record lies on a line segment between two real invalid records. For features that take discrete values, such as lengths and prefix integers, the interpolated values are generally not values that a real number could have, which is relevant for the interpretation of the residual errors in Section 3.1. XGBoost fits an additive ensemble of K trees by minimizing a regularized objective [6],

$$\mathcal{L} = \sum_i \ell\,(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega\,(f_k), \qquad \Omega(f) = \gamma T + 1/2\,\lambda \parallel w \parallel^2 \tag{4}$$

in which ℓ is the logistic loss, T the number of leaves of a tree, and w its leaf weights. Each new tree is fitted to the gradient of the loss at the current prediction, which lets boosting concentrate on the records that earlier trees handled badly. Random Forest, in contrast, averages independently grown trees [5]; the two ensembles therefore treat a small set of hard cases differently.

Both models were trained with scikit-learn conventions [8]. Hyperparameters were selected by randomized search with three-fold cross-validation and the F1-score of the invalid class as criterion, which samples the space instead of enumerating it [9]. Selection used the training data only, so that the test set serves for a single report [10]. The search covered 30 iterations per model. For Random Forest it selected n_estimators=300, max_depth=20, and min_samples_leaf=2; for XGBoost it selected n_estimators=300, max_depth=6, and learning_rate=0.1.

2.5. Evaluation

Accuracy, and precision, recall, and F1-score of the invalid class are reported, together with the Matthews correlation coefficient (MCC), which uses all four cells of the confusion matrix and is not inflated by a large majority class. Because a classifier that calls every record valid already attains 91.32% accuracy, accuracy alone is a weak guide. Cohen's kappa, the specificity, the negative predictive value, and the

likelihood ratios are added in Section 3.2 so that the screening value of each model can be read directly. The MCC is defined Equation 5.

$$\text{MCC} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \quad (5)$$

Proportions carry Wilson 95% intervals, which behave well for proportions near 0 or 1 [11]. Precision-based summaries are preferred to ROC curves for imbalanced problems [12]. Precision at a different prevalence π follows from the sensitivity and the false-positive rate through Bayes' rule, PPV = Se·π / (Se·π + FPR·(1 − π)).

Per-record predictions were not stored, so models cannot be compared with a standard paired test. For two models with error counts a and b, the number of records on which exactly one of them errs lies between |a − b| and a + b. The exact McNemar test [13], [14] on the discordant pairs gives a p-value for each bound, and the worst case (no shared errors) is the weakest evidence the data could support. If even that p-value is small, the difference is significant whatever the overlap. All steps used Python with scikit-learn [8] and the XGBoost library, and the split, the search, and the models share a fixed seed of 42, using scikit-learn 1.3.2 and XGBoost 2.0.3 under Python 3.10. The code is available from the corresponding author on reasonable request. Features were computed from the string of each record only, so no feature depends on other records, which excludes leakage through dataset-level statistics; duplicates across the split are a separate matter that is raised in Section 3.10. These tests are conditional on the trained models: they speak about the test set, not about the variability from retraining.

3. Results and Discussion

3.1. Scenario 1: prefix flag supplied

In S1 the two ensembles are almost identical (Table 3). Random Forest made 54 errors (40 false positives and 14 false negatives) and XGBoost 53 (40 and 13), both 99.42–99.43% accuracy. The 95% Wilson interval of the error rate is 0.45–0.76% for Random Forest and 0.44–0.74% for XGBoost, and the net difference of one record cannot be significant (exact p = 1). RQ1 is answered as follows: with the flag, the models are indistinguishable.

Table 3. Test results in scenario 1 (invalid class; 9,297 records).

Metric	Random Forest	XGBoost
TN / FP / FN / TP	8,450 / 40 / 14 / 793	8,450 / 40 / 13 / 794
Errors (error rate, 95% CI)	54 (0.58%; 0.45–0.76)	53 (0.57%; 0.44–0.74)
Accuracy	99.42%	99.43%
Precision	0.952	0.952
Recall	0.983	0.984
F1-score	0.967	0.968
MCC	0.964	0.965

The more interesting fact is that both models make about 53 errors although the feature set contains the answer to one of the three label conditions. A model that applied the three conditions exactly would make none. Forty false positives (valid numbers predicted invalid, since the "positive" class is invalid) with the same count in both models suggest a systematic cause rather than noise. Candidate explanations are that SMOTE interpolates between invalid records and so blurs the boundary near valid ones, and that the length and prefix conditions interact in a way that the 9 features express only approximately. Neither was tested, and neither can be ruled out from the data at hand.

3.2. Scenario 2: learning the prefix from digits

Removing the flag changes the picture (Table 4). XGBoost made 9 errors (4 false positives, 5 false negatives), an error rate of 0.097% (95% CI 0.05–0.18%), and an invalid-class F1-score of 0.994. Random

Forest made 75 errors (10 and 65), with an F1-score of 0.952 and a recall of 0.919; its false-negative rate is 8.05% (6.37–10.14%), meaning that about one invalid number in twelve is accepted. XGBoost in S2 is even better than either model in S1, which shows that the flag was not the limiting factor in S1.

Table 4. Test results in scenario 2 (invalid class; 9,297 records).

Metric	Random Forest	XGBoost
TN / FP / FN / TP	8,480 / 10 / 65 / 742	8,486 / 4 / 5 / 802
Errors (error rate, 95% CI)	75 (0.81%; 0.64–1.01)	9 (0.097%; 0.05–0.18)
Accuracy	99.19%	99.90%
Precision	0.987	0.995
Recall (95% CI)	0.920 (0.899–0.936)	0.994 (0.986–0.997)
F1-score	0.952	0.994
MCC	0.948	0.994
False-negative rate	8.05%	0.62% (0.27–1.44)

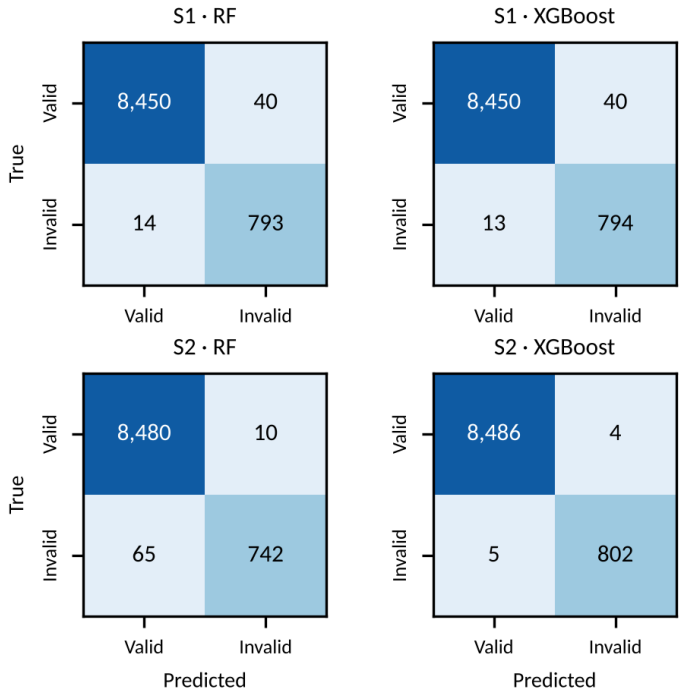


Figure 2. Confusion matrices of the four model–scenario combinations on the test set.

A plausible reason for the difference is that the prefix set is an arbitrary set of 40 values among 10,000 possible digit pairs. Boosting adds trees that correct the residual errors of earlier ones and can carve out many small regions of the two-digit space, whereas a Random Forest averages deep trees grown independently and, on a feature with only about 40 meaningful values, leaves some regions assigned by majority [5], [6]. This explanation is consistent with the recall figures but was not tested, for example by giving the forest more depth or by training a single tree. A decision tree with one branch per prefix is also the simplest way to represent Eq. (1), and it is the baseline whose absence is discussed in Section 3.6.

The composition of the errors differs by model. In S1 the errors are dominated by false positives (40 of 53–54), in which a valid number is declared invalid; in S2 Random Forest errs mostly in the other direction (65 of 75 false negatives), accepting invalid numbers, and XGBoost errs in both directions at a very low rate. Since the false-negative errors of the forest in S2 are invalid numbers that the model treats as valid, a natural hypothesis is that they carry prefixes that are rare in the invalid class or that lie close to allocated prefixes in the two-digit space. Verifying this needs the list of misclassified records, which was not saved; keeping it would turn the error count into a diagnosis.

Derived screening metrics sharpen the comparison. All four models have specificity above 0.995, so valid numbers are rarely rejected. The difference lies in sensitivity and in the likelihood ratios: a positive prediction from XGBoost in S2 multiplies the odds of invalidity by about 2,100, against about 780 for Random Forest and about 209 in S1, and its negative likelihood ratio of 0.006 means that a "valid" prediction almost excludes an invalid number, whereas the forest's 0.081 leaves a residual risk. Kappa and MCC order the models identically, so the ranking does not depend on the summary chosen.

3.3. Statistical comparison

Table 5 gives the exact McNemar p-values for all six pairs under the worst case of no shared errors. The S2 gap between XGBoost and Random Forest has a net difference of 66 records and a worst-case p of 4×10^{-14}, so it is significant even if the two models share no errors at all, which answers RQ2: XGBoost is better. XGBoost in S2 also beats both S1 models (worst-case p about 10^{-8}). Random Forest in S2 is not distinguishable from the S1 models in the worst case (p = 0.063–0.078), so S2 did not clearly hurt the forest compared with S1, although its point estimate is worse.

Table 5. Exact McNemar test between models (worst-case overlap).

Pair	Net difference	Worst-case p
S1-RF vs S1-XGB	1	1.0
S1-RF vs S2-RF	21	0.078
S1-XGB vs S2-RF	22	0.063
S1-RF vs S2-XGB	45	6.1×10^{-9}
S1-XGB vs S2-XGB	44	1.05×10^{-8}
S2-RF vs S2-XGB	66	4.3×10^{-14}

Net difference: difference in error counts. Conditional on the trained models.

The bounds make the strongest claims safe from the missing per-record predictions, but they remain conditional on one training run each. A seed-to-seed analysis (repeated splits with the corrected resampled t-test [14]) was not done, and with only nine errors the XGBoost S2 figures could move by several records on another split. We therefore read the results as clear for the large gap in S2 and unclear for the small ones.

The precision of the estimates is limited by the number of invalid records in the test set. With 807 invalid records, the 95% interval of the S2 XGBoost recall is 98.6–99.7%, which is about 1.2 percentage points wide; narrowing it to ±0.25 points around a recall of 0.994 would need about 4,000 invalid test records by the usual normal-approximation sample-size formula for a proportion, that is, a test set several times larger at the same prevalence. The nine-error result is therefore a good estimate of a very small error rate and a poor estimate of its exact size.

3.4. Feature contribution

A backward elimination was run on XGBoost in S2, removing at each step the feature whose absence cost least (Table 6). Starting from 10 features, F1-score stays above 0.98 down to 5 features, and the first removals (Noise_Count, Prefix_Head, Unique_Ratio, Sequential) cost almost nothing. The score falls to 0.933 with 3 features and collapses to 0.553 when Prefix_Body is removed, leaving only two length features. The three features that remain at the end, Digit_Length, Raw_Length, and Prefix_Body, are the critical ones, which corresponds exactly to the three conditions of Eq. (1): length bounds and the prefix list.

Table 6. Backward elimination for XGBoost in S2.

Features	Removed	F1-score	Accuracy
10	none	0.9944	0.9990
9	Noise_Count	0.9944	0.9990
8	Prefix_Head	0.9907	0.9984
7	Unique_Ratio	0.9882	0.9980
6	Sequential	0.9894	0.9982
5	Entropy	0.9815	0.9968
4	Max_Streak	0.9827	0.9970
3	Dominance	0.9331	0.9881
2	Prefix_Body	0.5528	0.8865

Noise_Count corresponds to the non-numeric count feature of Table 2.

The ablation answers RQ3 in a way that supports the circularity argument of Section 3.6: the model keeps the features that define the label and discards those that merely describe digit patterns. Digit dominance, entropy, streaks, and sequences, which are the "anomaly" style features, add little. This is expected if the labels do not depend on them, and it should not be read as evidence that such patterns are useless for finding suspicious numbers in general. Steps are sequential, so the order is path dependent, and each point is a single run.

Figure 3 relates the learned models to the rule that generated the labels. Panel (a) draws Eq. (1) as the decision tree that a perfect classifier would implement, and panel (b) shows how the F1-score of XGBoost in S2 falls as features are removed (Table 6): the three features that survive until the end, Digit_Length, Raw_Length, and Prefix_Body, are the ones that this tree tests. Panel (a) is a schematic of the rule, not a tree learned by either model.

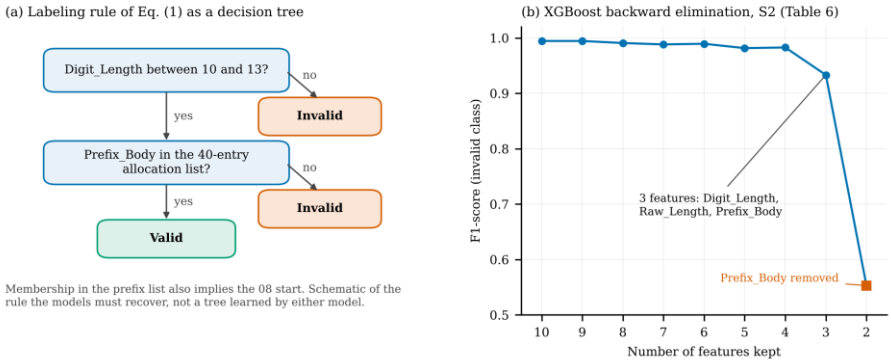


Figure 3. (a) The labeling rule of Eq. (1) as a decision tree and (b) F1-score of XGBoost in S2 as features are removed.

3.5. Efficiency

Table 7 gives the average of ten repetitions. XGBoost trained in 0.13 s (S1) and 0.11 s (S2) against 3.32 s and 4.11 s for Random Forest, about 26 and 36 times faster, and predicted the test set in 0.0033 s and 0.0024 s against 0.095 s and 0.125 s, about 29 and 52 times faster. The thesis claims "36 times" and "51 times", which correspond to the S2 ratios. Hardware, software versions, thread counts, and whether tuning time was included are not recorded, though the timing runs used the same machine as the hyperparameter search (an Intel Core i7-11700 CPU with 16 GB RAM, running scikit-learn 1.3.2 and XGBoost 2.0.3 under Python 3.10) with tuning time excluded, so the ratios are indicative and not transferable.

Table 7. Mean training and prediction time (10 repetitions; seconds).

Model	Training	Prediction (test set)
S1 Random Forest	3.3170	0.0953
S1 XGBoost	0.1268	0.0033
S2 Random Forest	4.1148	0.1247
S2 XGBoost	0.1132	0.0024

Per-record prediction time in S2: 0.0134 ms (Random Forest) and 0.00026 ms (XGBoost).

For a registration form, both predictors answer within a small fraction of a millisecond per record, so latency is not a deciding factor. The difference matters for retraining, but at 4 s it is not a practical barrier either. The efficiency finding therefore supports XGBoost mainly as a tie-breaker, not as a reason for adoption.

3.6. Label circularity and what each result supports

The central limitation is that the labels are a deterministic function of the digit string, and the features are digit-string properties. The classifier therefore learns to reproduce the labeling rule. Information from the target can enter the features in this way, a form of leakage in which a feature encodes the outcome [7]. In S1 the leak is explicit, because Valid_Prefix is one component of Eq. (1). In S2 it is implicit: Prefix_Body is a raw component, and the model's task is to memorize the allocation list from it. The accuracies measure how well the rule has been recovered. They do not measure how well the model detects numbers that satisfy the rule but are wrong, such as a typing error that lands on another allocated prefix, or a number that is allocated but unused. Table 8 states what each result does and does not support.

Table 8. What each result supports.

Result	Supports	Does not support
S2 XGBoost, 9 errors	A boosted ensemble can recover a 40-prefix rule from digits	Detection of wrong but well-formed numbers
S1 residual errors (≈53)	Models are not exact even with the flag	A cause; none was tested
Ablation	Length and prefix features carry the label	Uselessness of pattern features in general
Timing	XGBoost is faster in this run	Absolute speed on other hardware

Because a deterministic validator exists for exactly this rule, the learned model has no accuracy advantage in the present setting: a ten-line function would make zero errors on the same labels. A learned model is useful only in situations that the rule cannot express. Examples are a rule that is only partly known or changes over time, a soft score for review queues, and combination with signals that are not part of the digit string, such as repeated submission patterns. The study does not examine any of these, and the thesis itself proposes verification of WhatsApp activity as a second layer for this reason.

3.7. Imbalance, metrics, and the cost of errors

With 8.68% invalid records, accuracy overstates quality: the all-valid baseline reaches 91.32%, so the useful gain of 99.90% is the reduction of the error rate from 8.68% to 0.097%. SMOTE creates synthetic minority records by interpolation between neighbors in feature space [3]. For discrete features with a rule-defined class boundary, interpolated records may fall on the wrong side of the boundary, which is one candidate reason for the persistent S1 errors. The review of SMOTE variants also notes that oversampling is not always superior to cost-sensitive learning or to simply moving the decision threshold [4]; no such alternative was compared here.

The practical meaning of the errors depends on cost. A false negative lets an invalid number enter the registry, which costs a failed notification later. A false positive rejects a valid number, which costs a user a second attempt or a manual check. In S2 XGBoost produced 5 false negatives and 4 false positives among 9,297 records; scaled to 100,000 registrations that would be about 54 and 43. Because the invalid prevalence in a live registry may be lower than 8.68%, the positive predictive value would fall with prevalence, which

makes reporting at expected prevalence advisable. At a prevalence of 1%, the precision of 0.995 observed here (which rests on 4 false positives) would drop to about 0.955 for XGBoost and 0.887 for Random Forest in S2, and to 0.678 in S1, since false alarms come from the much larger valid population (Fig. 4).

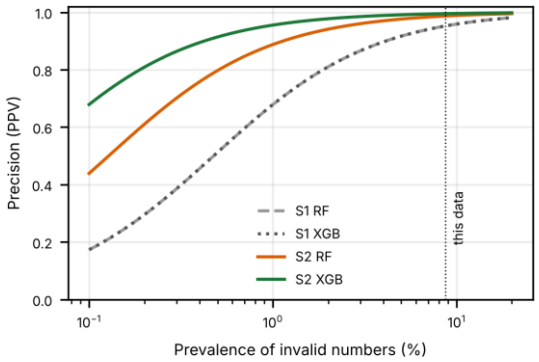


Figure 4. Precision of the invalid-class prediction as a function of prevalence.

Figure 4 shows that the ranking of the models is preserved at every prevalence, but the loss of precision at low prevalence is steep for the S1 models, whose 40 false positives are spread over the large valid class. A registry whose invalid rate is near 1% would flag roughly one valid number for every two invalid numbers under S1, against about one in twenty under S2 XGBoost. Any threshold for sending records to manual review should be set with the prevalence of the target registry in mind, not with the 8.68% of this dataset.

3.8. Probability output

The XGBoost probabilities in S2 are concentrated at both ends: valid records are scored at most 0.02 and invalid ones above 0.98, with almost no mid-range overlap. This indicates that the decision boundary is sharp, as expected for a deterministic rule. It should not be read as calibrated confidence, since tree ensembles trained on rebalanced data usually produce uncalibrated scores [15], [16], and SMOTE shifts the base rate the model has seen. A calibration curve on the untouched test set would be needed before the score is used as a probability.

A last practical point concerns the data themselves. A mobile number is personal data, and a validator needs only its syntactic properties. The 11 features here are computed from the digit string and do not identify a person, so a deployed model could receive the features alone, or hash the number after feature extraction, which reduces exposure. Training data that contain real numbers should be stored with restricted access, minimized, and used under a documented legal basis, and the model should be evaluated on synthetic or consented data when it is shared. This is consistent with treating data quality as a managed process, not a one-off cleaning step [1].

3.9. When is learning justified?

The comparison with a deterministic validator suggests a way to decide when a learned model is worth its cost. Learning is justified when the target concept cannot be written down, when it changes faster than rules can be maintained, or when a graded score is needed. It is not justified when the concept is a short, stable rule, as in the present labels. In the latter case the machine-learning result is best read as a proof of feasibility: a boosted ensemble can find a 40-element set of prefixes among 10,000 possibilities from about 37,000 examples with an error rate near 0.1%, which suggests that the approach could be applied to concepts whose rules are unknown.

The sample size argument gives a second reason for caution. The result rests on 4,033 invalid records, of which 3,226 trained the model; if the allocation list contained prefixes that occur rarely among invalid numbers, a rare prefix could be absent from the training data or appear too seldom for the model to learn

its status. A classifier would then be least reliable exactly on the rare prefixes, which the rule handles without difficulty. Reporting errors by prefix would show whether the nine errors of XGBoost are concentrated there, and an exhaustive test on all 10,000 four-digit prefixes against the rule would measure the learned prefix set directly, a check that costs nothing because the rule can label every prefix.

3.10. Threats to validity and checks

Several threats limit the conclusions (Table 9). The data come from one institution and one period, so regional or temporal variation is not covered. Labels are rule-derived, not verified. No simple baseline was evaluated. One split and one run per model were used. The hardware for the timing is unknown. Each threat has a concrete check that could be run on the same data.

Table 9. Threats to validity and planned checks.

Threat	Check
Labels equal the features' rule	Verify a random sample by delivery test; train on independent labels
No rule or tree baseline	Add a regular-expression validator and a single decision tree
SMOTE effect unknown	Repeat without SMOTE and with class weights
One split, one run	Repeated stratified splits; corrected resampled t-test
Probabilities uncalibrated	Calibration curve and isotonic or Platt scaling
Single source	Test on numbers from another institution or period
Duplicate or near-duplicate numbers across the split	Deduplicate before splitting; group split by number
Timing not reproducible	Report hardware, versions, and threads

A deployment protocol would turn the circularity problem into a data-collection mechanism: the rule handles what it can handle exactly, and the learned model is evaluated on the cases where the rule is silent, with delivery outcomes as labels. Only then can its accuracy be compared with that of the rule on equal terms.

3.11. Reporting recommendations

The analysis suggests items that a study of rule-labelled data quality should report so that readers can interpret the accuracy. The first is the provenance of the labels, because the same score has a different meaning for rule-derived and independently verified labels. The second is the ceiling of each scenario: a feature set that contains a label condition has a theoretical accuracy of 100%, so the distance from the ceiling, here 53–54 errors in S1, is the informative quantity and not the accuracy itself. The third is a trivial baseline, namely the all-valid classifier at 91.32% and a rule or single-tree model, since a score is meaningful only relative to what a simple method achieves. The fourth is stored per-record predictions, which allow paired tests and error analysis, and the fifth is an interval for every proportion. These items cost little and would have removed most of the uncertainty identified in Section 3.10.

Finally, the two scenarios illustrate a more general design lesson. Engineered features that encode a decision step, like the prefix flag, make the model look strong while hiding the part of the problem that is hard, and removing them (S2) exposes differences between algorithms that the engineered feature masked: the models were tied in S1 and separated by a factor of eight in errors in S2. A study that reports only the engineered-feature setting would have concluded that the choice of ensemble does not matter, whereas the harder setting shows that it does. This is a reason to evaluate models under the least-informed feature set that still represents the task.

4. Conclusion

On 46,485 Indonesian mobile-number records, XGBoost recovered a rule-defined notion of validity from syntactic features with 9 errors among 9,297 test records (99.90%) when the prefix had to be learned from digits, whereas Random Forest made 75 errors and missed 8% of invalid numbers. Exact bounds show that this gap is significant under any overlap of errors, while the two models are indistinguishable when a

prefix-validity flag is supplied. Ablation shows that length and prefix features carry the decision, and XGBoost trains about 36 times and predicts about 50 times faster in the recorded run. Since the labels are generated by the same rule that the features express, the results demonstrate rule recovery and not the detection of wrong but well-formed numbers, and a deterministic validator would be exact on these labels. The next step is validation against independent ground truth, such as message delivery, together with a rule baseline, a decision tree, a no-SMOTE comparison, and repeated splits.

AI Disclosure Statement

The authors used Generative AI and AI-Assisted Technologies throughout the preparation of this manuscript: to draft and revise portions of the text, to write and run the scripts behind the experiments and statistical analyses, to generate the figures included in the revision, and to format the document from the authors' own research data and source code. The authors verified the scripts and their outputs, reviewed the resulting text, cross-checked every reference against its original source, and accept full responsibility for the content of this publication.

Author Contributions Statement

Andi Faturrahman: Conceptualization, Methodology, Software, Investigation, Formal Analysis, Data Curation, Visualization, Writing – Original Draft.
Amriana: Supervision, Validation, Writing – Review and Editing.
Afiyah Rifkha Rahmika: Validation, Writing – Review and Editing.

References

[1] L. L. Pipino, Y. W. Lee, and R. Y. Wang, "Data quality assessment," *Commun. ACM*, vol. 45, no. 4, pp. 211–218, Apr. 2002, doi: 10.1145/505248.506010.

[2] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, Jul. 2009, doi: 10.1145/1541880.1541882.

[3] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, 2002, doi: 10.1613/jair.953.

[4] A. Fernández, S. García, F. Herrera, and N. V. Chawla, "SMOTE for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary," *J. Artif. Intell. Res.*, vol. 61, pp. 863–905, 2018, doi: 10.1613/jair.1.11192.

[5] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001, doi: 10.1023/A:1010933404324.

[6] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, San Francisco, CA, USA, 2016, pp. 785–794, doi: 10.1145/2939672.2939785.

[7] 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, pp. 1–21, Dec. 2012, doi: 10.1145/2382577.2382579.

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

[9] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, 2012.

[10] G. C. Cawley and N. L. C. Talbot, "On over-fitting in model selection and subsequent selection bias in performance evaluation," *J. Mach. Learn. Res.*, vol. 11, pp. 2079–2107, 2010.

[11] L. D. Brown, T. T. Cai, and A. DasGupta, "Interval estimation for a binomial proportion," *Stat. Sci.*, vol. 16, no. 2, pp. 101–133, May 2001, doi: 10.1214/ss/1009213286.

[12] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets," *PLoS ONE*, vol. 10, no. 3, Art. no. e0118432, Mar. 2015, doi: 10.1371/journal.pone.0118432.

[13] Q. McNemar, "Note on the sampling error of the difference between correlated proportions or percentages," *Psychometrika*, vol. 12, no. 2, pp. 153–157, Jun. 1947, doi: 10.1007/BF02295996.

[14] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural Comput.*, vol. 10, no. 7, pp. 1895–1923, Oct. 1998, doi: 10.1162/089976698300017197.

[15] A. Niculescu-Mizil and R. Caruana, "Predicting good probabilities with supervised learning," in *Proc. 22nd Int. Conf. Mach. Learn. (ICML)*, 2005, pp. 625–632, doi: 10.1145/1102351.1102430.

[16] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, 2017, arXiv:1706.04599.