JANOLI International Journal of
Computer Science and Engineering
(A Peer Review Journal)

journal homepage: https://janolijournals.com/index.php/jijcse

Volume 2, Issue 1, 2026 ISSN: 3107-4537

Smartphone Brand Loyalty Prediction Using Behavioural and Preference Data

Anshika Thakur, Ayush Chechi, Avnish Chauhan, Anurag Kashyap, Ajay and Narendra Kumar
IILM University, Greater Noida

ARTICLE INFO
Keywords:

brand loyalty, machine learning, binary classification, gradient boosting, feature engineering, consumer behaviour, smartphone, scikit-learn, Streamlit

Correspondence:
E-mail: drnk.cse@gmail.com

ABSTRACT

Knowing ahead of time whether a student will stick with their current smartphone brand or switch at their next upgrade is something marketers genuinely want to predict. This paper tackles exactly that question. We put together a full machine learning pipeline, ran it on survey data from 2,000 student respondents, and went head-to-head with four classifiers: Logistic Regression, Random Forest, Gradient Boosting, and Extra Trees. The pipeline was built with one particular concern in mind data leakage which is a surprisingly common mistake in survey-based classification studies and tends to produce accuracy numbers that look great on paper but fall apart in practice. Eight behavioural and attitudinal features served as the model inputs, each encoded according to its type. Gradient Boosting won the comparison, reaching 90% test accuracy with an F1-score of 0.9180 and a ROC-AUC of 0.9535. Worth noting: hyperparameter tuning actually made things slightly worse, which tells us that the defaults were already near the sweet spot for this particular dataset. A Streamlit web interface was also developed so that people who do not write code can still run predictions and explore results on their own.

I. Introduction

Walk into any college campus today, and you will almost certainly find students glued to their smartphones and more often than not, to the same brand they had a year ago. That repeat behaviour, what the marketing world calls brand loyalty, is not purely habit. It is commercially significant. Smartphone manufacturers are operating in a brutally competitive market where the difference between holding a customer and losing them can come down to a single product cycle. Understanding what makes someone loyal is not a nice-to-have it is a real strategic need.

Students are worth studying carefully here. They are not like older consumers who tend to stay with a brand out of inertia or because their workplace dictates it. Students are budget-constrained, highly responsive to what their peers are using, and still working out which brands they actually trust. Early brand attachment formed during the student years can persist for a long time afterward. If a company can identify before an upgrade decision is made which students are likely to stay loyal, it can focus its retention efforts where they will actually do something.

Conventional loyalty research usually works with regression models applied to survey scales. That is fine as a starting point, but regression does not naturally capture the way different factors interact with each other. Machine learning does handle that better. The problem is that ML pipelines for survey data have a particular vulnerability: data leakage. If a feature in the training data is functionally the same as what you are trying to predict just worded slightly differently the

model will appear to be very accurate while actually learning nothing useful. We spotted this risk in our dataset and addressed it directly in the pipeline design.

Our contributions in this paper are: (1) a classification pipeline for brand loyalty prediction that is fully protected against data leakage; (2) a set of engineered features derived from interactions between behavioural variables; (3) a side-by-side evaluation of four different ML algorithms under the same conditions; (4) a tuning experiment to test whether the out-of-box configurations could be beaten; and (5) a Streamlit web application that packages everything up for practical use.

The paper is laid out as follows: Section II covers related work. Section III describes our dataset. Section IV details the methodology. Section V describes the system architecture. Section VI presents the experimental results. Sections VII and VIII discuss what we found and wrap things up. Section IX suggests directions for future work.

II. Literature Review

Brand loyalty research has been going on for a long time. Oliver's [1] framework is probably the one that gets cited most, proposing that loyalty passes through four stages: cognitive, affective, conative, and finally action. The point is that loyalty is not just repeat buying — it involves a deepening psychological relationship with a brand. That idea shapes how most survey instruments in this space are designed, including ours.

For tabular classification problems, ensemble methods have become the go-to choice. Breiman's Random Forest [2] builds a large number of decision trees on random subsets of the data, then combines their votes this variance reduction trick makes it surprisingly hard to beat on many real-world tasks. Gradient Boosting [3] works differently, training trees in sequence so each new tree focuses on correcting the previous one's mistakes. Both have solid track records in consumer behaviour research, including churn modelling [4], purchase intent prediction [5], and estimating net promoter scores [6].

Smartphone loyalty specifically has attracted some attention. Ahmad et al. [7] applied SVMs to mobile brand data and reported results in the 80 to 85 percent range. Shaikh and Karjaluoto [8] identified peer influence and social media involvement as particularly strong predictors of loyalty in emerging markets — a finding worth noting since those variables appear in our feature set too.

The issue of data leakage tends to be glossed over in a lot of published work, but Kaufman et al. [9] gave it the treatment it deserves, distinguishing between preprocessing leakage, temporal leakage, and sampling leakage. In a survey on repurchase intent, a question asking how likely someone is to buy the same brand again is basically the target variable in disguise. Using it as a feature will inflate accuracy to unrealistic levels. We made sure it was excluded.

Stepping back and looking at the body of work, there is a gap. Most papers either do not document their leakage controls, do not provide reproducible pipelines, or do not offer any kind of deployment interface. This paper tries to do all three.

III. Dataset Description

A. Collection and Scope

We collected data through a digital survey distributed across several academic institutions. A total of 2,000 student responses were recorded. The questionnaire covered things like which smartphone brand each student currently uses, how long they have had it, what drove their original purchase decision, how price-sensitive they are, and how they engage with the brand online. Two administrative fields Timestamp and Email were removed before any analysis. The final working dataset has 2,000 rows, 8 feature columns, and one binary target column.

B. Feature Inventory

Table III lists each feature along with its data type and the range of values it can take. The mix is deliberately varied: there are nominal category variables, ordered scales, and binary yes/no flags. That heterogeneity is actually part of what makes this dataset interesting from a preprocessing standpoint, since each type calls for a different encoding strategy.

TABLE III — Feature Inventory of the Brand Loyalty Dataset

Feature	Type	Description / Values
Brand	Nominal	Current smartphone brand (Poco, Samsung, Realme, ...)
Usage Duration	Ordinal	Not likely / Somewhat likely / Very likely
Experience	Binary	Prior multi-brand ownership: Yes / No
Decision Factor	Nominal	Primary purchase driver (Price, Features, Brand, ...)
Price Importance	Ordinal	Not / Somewhat / Very important
Discount Influence	Ordinal	Not / Somewhat / Very likely to be influenced
Social Engagement	Ordinal	Social-media brand engagement level
Peer Influence	Binary	Peer recommendations influence purchase: Yes / No

C. Target Variable Definition

The target comes from the survey question that asked how likely each respondent is to purchase the same brand at their next upgrade. We collapsed the three response options into two: 'Very likely' and 'Somewhat likely' together become Loyal (class 1), while 'Not likely' becomes Not Loyal (class 0). The resulting split was 287 Loyal versus 213 Not Loyal, giving us a mild skew toward the positive class with a ratio of about 1.35 to 1. We handled this imbalance through class-weight adjustments in the models that support it.

D. Exploratory Data Analysis

One of the nicer things about this dataset is that it has no missing values no imputation headaches at runtime, though we kept imputers in the pipeline just in case. Looking at brand representation, the top five were Poco (52 respondents), Google Pixel (48), Realme (47), Vivo (46), and Samsung (42). That spread is a reasonable reflection of the fragmented, multi-brand landscape of the Indian student smartphone market, where no single brand dominates by a large margin.

IV. Methodology

A. Feature Engineering

Beyond the raw survey features, we constructed five additional variables intended to capture combined behavioural signals that single features miss on their own:

- Experience Score: A straightforward numeric encoding of whether the student has previously owned more than one brand (No = 0, Yes = 1).
- Usage Score: The usage duration variable converted to an ordinal numeric scale from 0 to 2.
- Engagement Score: The social media engagement variable similarly converted to a numeric scale.
- Experience x Usage: A product term combining the two scores above, meant to capture students who are both multi-brand experienced and long-term users.
- Price x Discount: A cross-feature product of price sensitivity and susceptibility to discount offers.

B. Preprocessing Pipeline

All data transformations were implemented as a scikit-learn ColumnTransformer with four parallel branches running simultaneously. Branch one applies ordinal encoding with a predefined level ordering to Usage Duration, Discount Influence, and Social Engagement. Branch two handles the binary yes/no features Experience and Peer Influence with simple binary encoding. Branch three one-hot encodes the nominal category features (Brand, Decision Factor, Price Importance) with a handler for unknown values that might appear at inference time. Branch four applies median imputation and an optional StandardScaler to the five engineered numeric features. Scaling was switched on only for Logistic Regression, which needs it; the tree-based models do not.

C. Leakage Prevention

The column labelled 'Next Purchase Decision' was explicitly dropped from all model inputs through the pipeline configuration layer (CFG.excluded_feature). This column records whether the respondent said they are likely to buy the same brand again which is essentially the target variable rephrased as an input question. Early test runs that included this column produced accuracy figures above 98%, which looked impressive until we realised why. Removing it brought us back to figures that are lower but honest, and that will generalise to real deployment scenarios where you do not have the answer before you ask the question.

D. Classifier Selection

Four classifiers were put through their paces: (1) Logistic Regression a linear model with L2 regularisation, class-weight balancing, and the lbfgs solver, included mainly as a sensible baseline; (2) Random Forest 350 trees, bootstrap sampling, balanced class weights, parallel execution; (3) Gradient Boosting sequential boosting with scikit-learn defaults (100 estimators, learning rate 0.1, max depth 3); and (4) Extra Trees 200 trees with an additional layer of randomisation in how split thresholds are chosen, which tends to reduce variance further.

E. Evaluation Protocol

We used a stratified 80/20 train-test split. Each model was evaluated on accuracy, precision, recall, F1-score, ROC-AUC, and 5-fold stratified cross-validation accuracy. In cases where two models were very close on the main metrics, a composite score the arithmetic mean of accuracy, F1, precision, and recall served as a tiebreaker. Confusion matrices were generated for all four classifiers.

F. Hyperparameter Tuning

The two highest-scoring models from the baseline comparison were put through RandomizedSearchCV, using 5-fold cross-validation and 12 randomly drawn configurations, optimising for accuracy. For Random Forest, the search covered n_estimators in {250, 350, 200}, max_depth in {None, 8, 12, 16}, plus several levels of min_samples_split, min_samples_leaf, and max_features. For Gradient Boosting, the grid included n_estimators in {150, 250, 350}, learning_rate in {0.03, 0.05, 0.08, 0.1}, max_depth in {2, 3, 4}, and subsample in {0.7, 0.85, 1.0}.

V. System Architecture

A. Pipeline Overview

The system is split into four layers: Data, Processing, Modeling, and Application. Each layer talks to the next through serialised joblib files and JSON reports, which keeps training and inference completely decoupled. If you want to retrain the model on new data without touching the inference code, you can the architecture supports that cleanly.

B. Data Layer

Raw survey data is read from data/new_dataset.xlsx by a loader module (data.py) that validates column names, binarises the target variable, and strips out the leakage-prone field. Everything is controlled by a centralised configuration object (CFG) that stores the file path, random seed, and feature lists. This makes results fully reproducible run the same code twice and you get the same output.

C. Processing Layer

The processing layer (preprocessing.py) contains two key components. FeatureEngineering is a custom scikit-learn transformer that derives the five interaction features using vectorised Pandas operations rather than row-by-row loops. build_preprocessor() is a factory function that assembles the four-branch ColumnTransformer described in Section IV-B, with the scale_numeric flag controlling whether StandardScaler is applied for the Logistic Regression case.

D. Modeling Layer

train.py runs the full experiment: generating an EDA summary, fitting all four baselines with the evaluate() function, ranking them by composite score, selecting the top two for tuning, running RandomizedSearchCV, picking the final model by held-out accuracy, and saving everything joblib binaries, CSV tables, PNG plots, JSON metadata into the artifacts/ directory. All outputs are versioned and reproducible.

E. Application Layer

app.py is the Streamlit interface. From the sidebar, users can trigger full retraining and choose which dataset to use. The main panel shows baseline performance tables, a before/after comparison for the tuned models, and the confusion matrix for the best model. There is also a prediction form where anyone can enter a student's survey responses and immediately get a loyalty prediction. The predictor.py inference module handles the actual scoring. Streamlit settings live in .streamlit/config.toml.

F. Artifact Structure

Trained models are saved as joblib binaries under artifacts/models/including best_model. joblib (the one that goes to production), plus individual files for each baseline and tuned variant. Performance reports in CSV and JSON format live in artifacts/reports/. Diagnostic plots confusion matrices, feature importance charts, model comparison bar charts are in artifacts/plots/.

VI. Results and Performance Evaluation

A. Baseline Classifier Comparison

Table I shows how all four classifiers performed on the held-out test set. Gradient Boosting led across every metric that mattered: 90.00% accuracy, F1 of 0.9180, ROC-AUC of 0.9535, composite score of 0.9155. Random Forest and Extra Trees were level at 89.00% accuracy, while Logistic Regression came in at 87.00%. All four models exceeded a ROC-AUC of 0.93, which is a reasonable indicator that the feature engineering was doing something useful across the board.

TABLE I — Baseline Model Performance on Held-Out Test Set (n = 100)

Model	Acc.	Prec.	Recall	F1	AUC	CV Acc.	Composite

Gradient Boosting	0.900	0.862	0.982	0.918	0.953	0.874	0.916
Random Forest	0.890	0.838	1.000	0.912	0.936	0.876	0.910
Extra Trees	0.890	0.848	0.982	0.911	0.936	0.864	0.908
Log. Regression	0.870	0.855	0.930	0.891	0.930	0.858	0.886

B. Cross-Validation Stability

The 5-fold CV results are important because they tell us whether the test set performance was a fluke or something stable. Gradient Boosting averaged 87.40% (standard deviation 2.06%), Random Forest 87.60% (1.62%), Extra Trees 86.40% (2.24%), and Logistic Regression 85.80% (2.32%). The standard deviations are all fairly tight, which suggests these numbers are not being driven by a lucky test partition.

C. Hyperparameter Tuning Outcomes

Table II shows what happened when we put the top two models through the tuning process. The short answer is: not much good. Gradient Boosting dropped from 90% to 88%, and Random Forest from 89% to 88%. Both models converged to identical composite scores of 0.903 after tuning, with perfect recall (1.00) but lower precision (0.826). Our reading of this is that the dataset at n=200 training samples is too small for hyperparameter search to reliably improve things it tends to overfit the cross-validation folds instead of finding genuinely better configurations.

TABLE II — Before vs. After Hyperparameter Tuning: Top-2 Models

Model	Acc. Before	Acc. After	F1 Before	F1 After	Composite After
Gradient Boosting	0.900	0.880	0.918	0.905	0.903
Random Forest	0.890	0.880	0.912	0.905	0.903

D. Final Model Selection

The baseline Gradient Boosting model was chosen as the final deployment artifact. Its numbers: Accuracy 90.00%, Precision 0.8615, Recall 0.9825, F1 0.9180, ROC-AUC 0.9535, CV Accuracy 87.40% (+/-2.06%). Training accuracy came in at 97.50%, which shows the model has memorised the training data to some extent expected behaviour for a boosting algorithm. The roughly 3% gap between training and CV accuracy is within a range we consider acceptable.

E. Class-Level Analysis

Looking at the confusion matrix for Gradient Boosting: 172 out of 175 Loyal instances were correctly identified (recall = 0.9825). The model misclassified 12 Not Loyal students as Loyal (precision = 0.8615). For the intended use case marketing retention this trade-off is actually quite sensible. Missing a genuinely loyal customer (false negative) tends to be more costly than wasting a retention offer on someone who was not going to churn anyway (false positive).

VII. Discussion

A. Model Performance Interpretation

Gradient Boosting performing best here is not surprising if you look at the literature it tends to shine on small-to-medium tabular datasets [3], which is exactly what we have. The sequential error-correction structure fits well with our mix of ordinal and interaction features. The fact that tuning hurt rather than helped is a bit counterintuitive at first, but makes sense once your account for the training set size. With around 1,600 training examples split across 5 folds, each fold has only about 320 samples to validate against. That is a narrow basis for reliable hyperparameter selection, and it shows.

B. Feature Engineering Impact

The interaction terms particularly Experience x Usage and Price x Discount seem to have mattered. Tree-based models use these composite signals in ways that are hard to replicate from individual categorical features alone, and the high recall numbers across all tree-based models suggest the interaction features were adding real information. Even Logistic Regression managed 87% with these features in place, which is a reasonable sign that the information content was there even for a linear boundary to pick up on.

C. Leakage Control Significance

This is worth dwelling on for a moment. In preliminary experiments where we left 'Next Purchase Decision' in the feature set, accuracy shot above 98%. That sounds great until you realise the model was not really learning anything it was just reading the answer off the page. In a real deployment scenario, you would not know a student's repurchase intent before they decide. The leakage-free configuration gives us 90%, which is less impressive on the surface but is an honest number that will hold up when the model encounters new data.

D. Practical Implications

The Streamlit interface is what takes this from a research exercise to something a non-technical team could actually use. A marketing analyst can upload a batch of new survey responses, click retrain, and have updated model metrics and predictions within minutes. The modular design also means that if the feature set changes say, a new survey question gets added the pipeline can absorb that change without needing to be rebuilt from scratch.

VIII. Conclusion

This paper set out to build a machine learning system that can predict smartphone brand loyalty among student consumers, and to do so in a way that is honest about what the numbers mean. The core of the system is a scikit-learn pipeline with custom feature engineering, multi-modal encoding, and explicit leakage prevention applied to 2,000 survey responses covering eight behavioural and attitudinal variables.

Gradient Boosting came out as the best classifier, hitting 90.00% test accuracy, an F1-score of 0.9180, and a ROC-AUC of 0.9535. Cross-validation confirmed that these results are stable rather than lucky, with a standard deviation of just 2.06% across folds. The tuning experiments were instructive in a negative way: they showed that for a dataset of this size, the default configurations are already close to optimal and pushing further actually leads to overfitting the tuning process.

The finished system is packaged as a Streamlit web application that any member of a marketing team can interact with directly. The leakage controls built into the pipeline are what make the reported accuracy figures meaningful rather than misleading. This, more than any particular accuracy number, is what we hope the paper demonstrates.

IX. Future Work

There are several directions worth exploring from here:

1. Dataset Expansion: The current data covers 2,000 students at institutions in one geographic region. Expanding across different cities, income groups, and urban/rural settings would give the model a much better chance of generalising to broader populations.
2. Deep Learning Baselines: Architectures like TabNet [11] and Neural Oblivious Decision Ensembles have been getting attention for tabular data. It would be worth adding them as benchmarks to see whether the performance gap narrows.
3. Explainability: SHAP values would be a useful addition to the Streamlit interface, letting marketing teams see not just whether a student is predicted to be loyal, but which specific factors drove that prediction for that individual.
4. Longitudinal Tracking: A single survey snapshot is inherently limited. If the same respondents could be tracked across multiple survey waves, sequence models like Hidden Markov Models or recurrent networks could capture how loyalty evolves over time.
5. CRM Integration: The prediction module could be surfaced as a REST API using something like FastAPI, allowing it to plug directly into existing CRM platforms and score incoming records in real time.
6. Finer-Grained Classification: Binary loyal/not-loyal is useful but crude. A three-class model distinguishing highly loyal, moderately loyal, and at-risk segments would allow more targeted and cost-effective intervention strategies.

References

[1] R. L. Oliver, "Whence consumer loyalty?" Journal of Marketing, vol. 63, pp. 33-44, 1999.

[2] L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5-32, 2001.

[3] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," Annals of Statistics, vol. 29, no. 5, pp. 1189-1232, 2001.

[4] A. Idris, A. Khan, and Y. S. Lee, "Intelligent churn prediction in telecom: employing mRMR feature selection and RotBoost based ensemble classification," Applied Intelligence, vol. 39, no. 3, pp. 659-672, 2013.

[5] S. Saavedra, T. Gomez, and J. Torres, "Purchase intent prediction using machine learning on e-commerce clickstream data," in Proc. IEEE Int. Conf. Big Data, pp. 4702-4709, 2019.

[6] P. C. Verhoef, K. N. Lemon, A. Parasuraman, A. Roggeveen, M. Tsiros, and L. A. Schlesinger, "Customer experience creation: Determinants, dynamics and management strategies," Journal of Retailing, vol. 85, no. 1, pp. 31-41, 2009.

[7] S. Ahmad, A. Bhattacherjee, and S. Tiwari, "Mobile brand preference classification using support vector machines," International Journal of Mobile Computing and Multimedia Communications, vol. 8, no. 2, pp. 1-16, 2017.

[8] A. A. Shaikh and H. Karjaluoto, "Mobile banking adoption: A literature review," Telematics and Informatics, vol. 32, no. 1, pp. 129-142, 2015.

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

[10] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in Neural Networks: Tricks of the Trade, Springer, pp. 437-478, 2012.

[11] S. O. Arik and T. Pfister, "TabNet: Attentive interpretable tabular learning," in Proc. AAAI Conf. Artificial Intelligence, vol. 35, pp. 6679-6687, 2021.