

Predictive model for converting leads into repeat order customer using machine learning

Deryan Everestha Maured, Gede Putra Kusuma

Department of Computer Science, BINUS Graduate Program – Master of Computer Science, Bina Nusantara University, Jakarta, Indonesia

Article Info

Article history:

Received May 31, 2024

Revised Oct 23, 2024

Accepted Nov 19, 2024

Keywords:

DeepFM model

Gradient boosting

Machine learning

Random forest

Repeat order customers

ABSTRACT

In the competitive business landscape, customer relationship management (CRM) is pivotal for managing customer relationships. Lead generation and customer retention are critical aspects of CRM as they contribute to sustaining business growth and profitability. Also, identifying and converting leads into repeat customers is essential for optimizing revenue and minimizing promotional costs. This study focuses on developing a predictive model using machine learning techniques to convert leads into repeat order customers in conventional businesses. Leveraging data from a motorcycle distribution company in Jakarta and Tangerang, the study compares the performance of various models for predicting repeat orders. This includes individual models like DeepFM, random forest, and gradient boosting decision tree models. Additionally, it explores the effectiveness of stacking these models using logistic regression as a meta-learner. Furthermore, the study implements backward feature elimination for feature selection and hyperband for hyperparameter tuning to enhance model performance. The results indicate that Stacking model using base model default configuration stands out as the most robust, achieving the highest scores in accuracy (0.95), area under the curve receiver-operating characteristic curve (AUC-ROC) (0.67), log loss (0.19), weighted average precision (0.95), weighted average recall (0.95), and weighted average F1-score (0.92), effectively handling the imbalanced dataset.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Deryan Everestha Maured

Department of Computer Science, BINUS Graduate Program – Master of Computer Science

Bina Nusantara University

Jakarta 11480, Indonesia

Email: deryan.maured@binus.ac.id

1. INTRODUCTION

In the competitive business landscape, customer relationship management (CRM) plays a vital role in managing customer relationships, with lead generation and customer retention being key components. According to the market research report, approximately 27.8% of CRM usage is dedicated to these areas [1]. A survey conducted in the United States reveals that enhancing performance through CRM customer analytics has led to a 58% increase in customer retention or loyalty [2]. Leads represent prospective customers or prospects showing interest in a company's products or services. Managing leads within CRM systems helps companies identify and prioritize potential leads for conversion into new or repeat order customers. Retaining existing customers is crucial for long-term success, as customers who engage in repeat orders hold significant value in terms of retention and business revenue. Therefore, identifying customers

who can be converted into repeat order customers is imperative, as it reduces promotional costs and increases return on investment (ROI) [3].

The integration of machine learning techniques has emerged as a powerful tool in enhancing CRM effectiveness, particularly in optimizing lead management through the identification of potential customers [4]. Various machine learning models, including basic classifiers, ensemble models, and fusion models, have been applied to construct predictive models for repeat order customers. Benhaddou and Leray [5] developed lead scoring models using Bayesian networks, which leveraged heuristics and expert knowledge to achieve significant precision, recall, and accuracy. Nygård and Mezei [6] automated lead assessment using machine learning algorithms, highlighting random forest's superiority in controlling biases and optimizing neural network models.

In a similar vein, Espadinha-Cruz *et al.* [7] focused on enhancing lead management efficiency by comparing algorithms and leveraging ensemble regression models for improved accuracy. Ayaz [8] compared machine learning models for lead scoring, emphasizing the significance of feature selection and hyperparameter tuning in decision tree-based models. Arun [4] underscored the role of data analysis and machine learning in CRM, highlighting the importance of predictive accuracy in understanding customer behavior and optimizing prospect scores. Yang *et al.* [9] develops a dynamic, data-driven framework for predicting repeat purchases using a voting-based method with transactional data, showing that the integration of extreme gradient boosting (XGB) and light gradient boosting machine (LGBM) yields the best performance. Mortensen *et al.* [10] proposed classification models for predicting prospect success, with random forest outperforming others.

Moreover, Huang [11] compared predictive models for repeat buyer behavior, highlighting DeepFM's accuracy and LightGBM's training speed. Ensemble techniques proposed by Xu *et al.* [12] and Zhang and Wang [13] significantly enhanced model robustness and accuracy. The use of low-level interaction data [14], feature engineering [15]-[17], deep learning ensemble models [18], model distillation, and fusion techniques [19], [20], as well as vote stacking methods [21], further improved model accuracy and interpretability. Gokhale and Joshi [22] and Sangaralingam *et al.* [23] explored optimal classification algorithms for lead potential and supervised learning for customer identification, respectively, highlighting the importance of accurate modeling in CRM strategies and business decision-making.

These studies collectively showcase the diverse methodologies and approaches in lead scoring and customer prediction, emphasizing the critical role of feature engineering, model optimization, and algorithm selection in driving effective CRM strategies and business outcomes. Recent research has particularly highlighted the benefits of exploring DeepFM models, which have demonstrated superior performance in predicting repeat order customers compared to decision tree-based models like random forest and gradient boosting [11]. However, much of the prior research has focused on assessing customer potential within the e-commerce domain and has not thoroughly examined the benefits of feature selection and hyperparameter tuning for enhancing model performance, especially in scenarios involving class imbalance and optimization of predictive accuracy.

Given these findings, this study aims to compare three types of machine learning models DeepFM, random forest, and gradient boosting decision tree (GBDT). It applies feature selection through backward feature elimination and hyperparameter tuning using hyperband, while also exploring the effectiveness of stacking these models using logistic regression as a meta-learner. The goal is to enhance the accuracy of predictive models for lead conversion into repeat order customers in conventional business settings. This study adopts a case study approach focused on a company operating in the motorcycle distribution sector in the Jakarta and Tangerang regions. The company needs to manage its lead data (outbound leads) to identify those with the potential to become repeat order customers based on attributes such as demographics and consumer behavior.

2. RESEARCH METHOD

The conceptual framework of this research is illustrated in Figure 1. This research adopts the cross-industry standard process for data mining (CRISP-DM) framework, widely recognized for its effectiveness in tackling various challenges encountered in data mining projects within industrial environments [24]. CRISP-DM encompasses six key stages: business understanding, data understanding, data preparation, modelling, evaluation, and deployment [25]. In this research, the CRISP-DM framework is utilized up to the evaluation stage, with each stage detailed in subsections 2.1 to 2.5.

The research begins with business understanding, focusing on identifying key business objectives by analyzing data from the customer database (CDB) to understand the factors that drive repeat orders. Following this, data understanding involves collecting and exploring the data to assess its quality and uncover relevant patterns. In the data preparation stage, the data is cleaned and transformed to ensure it is ready for modeling. During modeling, algorithms such as random forest, GBDT, and DeepFM are applied to

build predictive models. Further model optimization is conducted through feature selection using backward feature elimination, hyperparameter tuning using hyperband, and stacking models using logistic regression as meta-learner to enhance performance. The models are then evaluated in the evaluation stage using performance metrics to select the most robust model. Finally, the determining the best model stage identifies the most suitable model, which is then used as the predictive model to identify leads likely to become repeat customers.

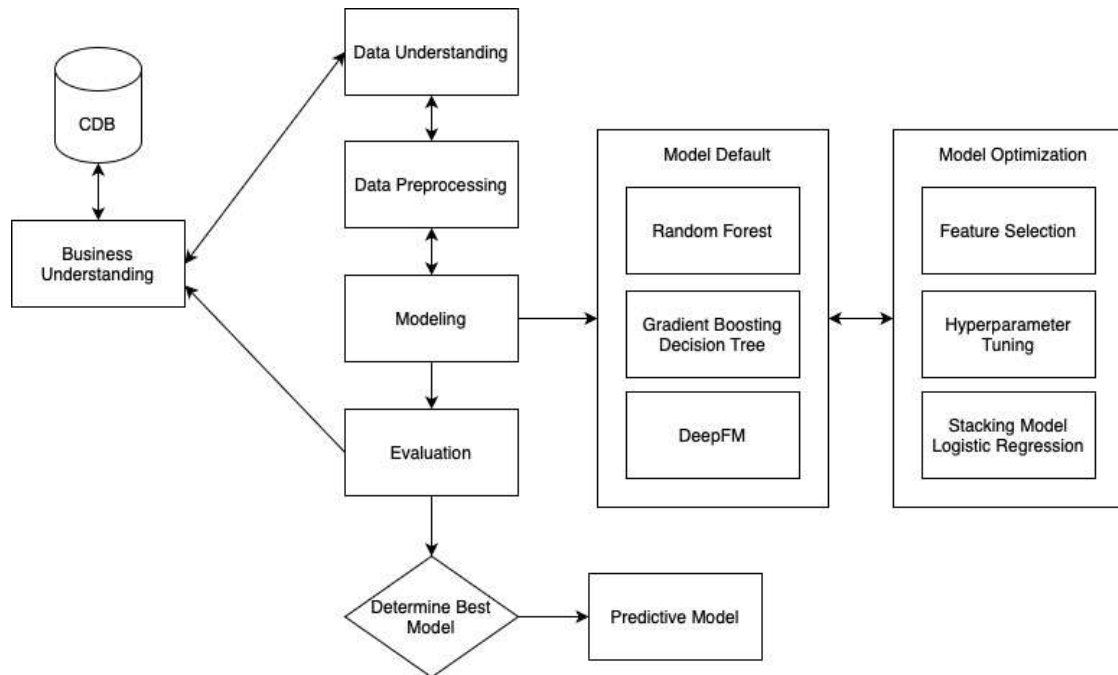


Figure 1. The conceptual framework

2.1. Business understanding

Based on the business regulations within the company at the research case study location, a repeat order consumer is defined as a customer who purchases a motorcycle more than once from the same dealer. However, to aid lead generation, particularly concerning repeat order consumers (RO), a list of customers who have purchased motorcycles more than once from different dealers will also be considered as leads for follow-up by the dealers from previous purchases. Lead generation is also based on individual customer type and customers aged over 17 years old. Hence, the labelling process for model development data is categorized as follows: i) positive class (1): repeat order consumers, meaning customers who have purchased motorcycles more than once; and ii) negative class (0): consumers who do not make repeat orders.

2.2. Data understanding

The dataset used in this research consists of the customer database (CDB), which includes invoice data and motorcycle information from the company at the case study location. This dataset covers the period from 2018 to 2022 and totals 1,474,369 records. The invoice data captures the transactions of motorcycle purchases made by customers and is stored in a table named MOHONFAKTUR, with field information detailed in Table 1.

Additionally, the motor prices table in Table 2 stores motorcycle information, including the model code (KD_MDL), original model code (KD_MDL_ASAL), motor number (NO_MTR), motor series name (NM_MTR), motor category (CUB_SPORT), and motor price (MTR_HRGJUAL).

From the combination of these two datasets, the features used in the data preparation stage consist of 16 features: 4 numerical (CICILAN, DP, JML_ANGSURAN, MTR_HRGJUAL), 8 categorical (JNS_KLM, JNS_JUAL, STS_RUMAH, TUJU_PAKAI, KODE_KERJA, CUB_SPORT, KODE_DIDIK, KELUAR_BLN), and 4 object features (TGL_MOHON, NO_DLRP, NO_KTPNPWP, TGL_LAHIR).

Table 1. Structure of MOHONFAKTUR table

Attribute group	Field	Field description	Data Type
Customer profile	KODE_DIDIK	Customer education code	Object – ordinal categorical
	KODE_KERJA	Customer job Code	Object – nominal categorical
	STS_RUMAH	Customer home status code	Object – nominal categorical
Deal behaviour	KELUAR_BLN	Range of spending money per month	Object – ordinal categorical
	TGL_MOHON	Transaction date	Object
	JNS_JUAL	Transaction method	Int64 – nominal categorical
	KD_MDL	Unique purchased motorbike model code	Object
	NO_MTR	Unique purchased motorbike code	Object
	NO_DLRP	Unique dealer code (customer purchase location)	Object
	DP	Nominal amount of down payment	Int64
	CICILAN	The nominal installment amount	Int64
	JML_ANGSURAN	Tenor amount in months	Int64
	TUJU_PAKAI	Code of the intended use of the motorbike purchased	Object – nominal categorical
Customer Identity	THN_MTR	Year motor purchased	Object
	NO_KTPNPWP	Customer unique IDE	Object
	NMI_MOHON	Customer name	Object
	NO_RGK	Unique motorbike frame number code	Object
	NO_MSN	Unique motorbike engine number code	Object
	NO_HP	Consumer telephone number	Object
	JNS_BELI	Types of consumers	Object
	TGL_LAHIR	Consumer's date of birth	Object
	GENDER	Gender	Int64 – nominal categorical

Table 2. Structure of motor prices table

Field	Field description	Data type
KD_MDL	Unique motorbike model code	Object
KD_MDL_ASAL	Unique code for original motorbike model	Object
NO_MTR	Unique motorbike code	Object
NM_MTR	Motor series name	Object
CUB_SPORT	Motorcycle category	Object – nominal categorical
MTR_HRGJUAL	Motorcycle selling price	Int64

2.3. Data preparation

The data preparation stage involved several crucial steps to ensure the quality and reliability of the dataset used for the predictive model. These steps included data integration, data transformation, handling missing values, identifying abnormal users, splitting the data, and addressing imbalanced data. Each of these processes was carefully executed to maintain the integrity of the data, ultimately leading to better model performance.

Data integration was the first step, where motorcycle purchase invoice data (Table 1) was combined with motorcycle information data (Table 2). The integration was achieved by matching the model code (KD_MDL) in the invoices with the original model code (KD_MDL_ASAL) in the motorcycle information data. This step ensured that all relevant data was linked and could be used cohesively for further analysis.

Next, the data transformation process was undertaken to convert and modify the data into a format more suitable for analysis and modeling. This involved converting certain fields, such as TGL_MOHON and TGL_LAHIR, from object types to datetime, standardizing other object types to integers, and replacing specific string values with numeric ones in fields like KODE_DIDIK, KODE_KERJA, STS_RUMAH, TUJU_PAKAI, and JML_ANGSURAN. Additionally, new variables were created, including purchasing behavior attributes (e.g., buying cycle, recency, total purchases, total amount spent, and average spending per purchase) and age categories, while categorical variables such as JNS_KLM, JNS_JUAL, STS_RUMAH, TUJU_PAKAI, KODE_KERJA, KATEGORI_USIA, and CUB_SPORT were transformed into one-hot encoding for better model interpretation. Moreover, numeric variables were transformed using logarithmic and squared transformations to enhance feature richness, such as CICILAN, DP, JML_ANGSURAN, JML_ANGSURAN_MEAN, mtr_hrgjual, BUYING_CYCLE, RECENCY, AVG_TOTAL_SPENT, JML_PEMBELIAN.

Handling missing values was a critical step to ensure that the dataset was complete and accurate. Missing values can significantly disrupt analysis and modeling, so null or NaN values in specific fields like KODE_DIDIK and RECENCY were replaced with 0. This replacement prevented potential errors in the modeling process and ensured that all records were fully utilized.

Identifying abnormal users involved cleaning the data to remove potentially invalid or mistyped entries, particularly in the NO_KTPNPWP field. Records with incorrect or suspicious NO_KTPNPWP values were deleted, and data was filtered to include only customers aged 17 and above. This step was crucial

to maintaining the accuracy and consistency of the dataset, which in turn ensured more reliable analysis and modeling outcomes.

The data was then split into training and testing sets to facilitate model training and evaluation. The dataset was divided with 80% allocated to training and 20% to testing, and the training set was further split using 4-fold cross-validation. This approach helped prevent overfitting and ensured that the model's predictions would be consistent and generalizable.

Finally, the issue of imbalanced data was addressed using SMOTEENN, a combination of synthetic minority over-sampling technique (SMOTE) and edited nearest neighbors (ENN) [26]. This method helped to balance the distribution of classes by generating synthetic samples for the minority class and refining the dataset by removing noisy samples from the majority class. SMOTEENN is highly effective in handling imbalanced data and significantly enhances model accuracy [27]. In this research, SMOTE was configured with `sampling_strategy='auto'` to balance the minority class with the majority class, and `k_neighbors=5` to determine the number of nearest neighbors for synthesizing new samples. ENN was applied with `sampling_strategy='all'` to remove noise from all classes, ensuring a cleaner dataset. To ensure consistent results, SMOTE and ENN set the seed for the random number generator using `random_state=42`. Applying SMOTEENN to the training data resulted in significant changes in the class distribution. Before resampling, the dataset contained 864,039 samples, with the majority class (non-repeat order) having 815,870 samples and the minority class (repeat order) only 48,169 samples. After applying SMOTEENN, the total number of samples was reduced to 715,697, with the majority class reduced to 480,975 samples and the minority class increased to 234,722 samples. Although SMOTEENN did not achieve perfect balance, the more equal distribution improved the model's ability to accurately predict the minority class. After completing all these data preparation steps, the dataset was refined to include 78 final features and a total of 1,080,049 records. These features, which are essential for the modeling process, are detailed in Table 3. This prepared dataset provided a strong foundation for building a reliable and effective predictive model.

Table 3. The features used as input for modeling process

Numerical feature	Categorical feature
CICILAN, DP, JML_ANGSURAN,	JNS_KLM_0, JNS_KLM_1, JNS_KLM_2, JNS_JUAL_1,
JML_ANGSURAN_MEAN, mtr_hrgjual,	JNS_JUAL_2, STS_RUMAH_0, STS_RUMAH_1,
BUYING_CYCLE, RECENCY, TOTAL_SPENT,	STS_RUMAH_2, STS_RUMAH_3, TUJU_PAKAI_0,
JML_PEMBELIAN, CICILAN_DP_ratio,	TUJU_PAKAI_1, TUJU_PAKAI_2, TUJU_PAKAI_3,
JML_ANGSURAN_TOTAL_SPENT_ratio,	TUJU_PAKAI_4, TUJU_PAKAI_5, TUJU_PAKAI_6,
RECENCY_BUYING_CYCLE_ratio, CICILAN_log,	TUJU_PAKAI_7, KODE_KERJA_0, KODE_KERJA_1,
DP_log, JML_ANGSURAN_log,	KODE_KERJA_2, KODE_KERJA_3, KODE_KERJA_4,
JML_ANGSURAN_MEAN_log, mtr_hrgjual_log,	KODE_KERJA_5, KODE_KERJA_6, KODE_KERJA_7,
BUYING_CYCLE_log, RECENCY_log,	KODE_KERJA_8, KODE_KERJA_9, KODE_KERJA_10,
TOTAL_SPENT_log, JML_PEMBELIAN_log,	KODE_KERJA_11, KODE_KERJA_12, KODE_KERJA_13,
CICILAN_squared, DP_squared,	KODE_KERJA_14, KODE_KERJA_15, KODE_KERJA_16,
JML_ANGSURAN_squared,	CUB_SPORT_1, CUB_SPORT_2, CUB_SPORT_3,
JML_ANGSURAN_MEAN_squared, mtr_hrgjual_squared,	KATEGORI_USIA_1, KATEGORI_USIA_2,
BUYING_CYCLE_squared, RECENCY_squared,	KATEGORI_USIA_3, KATEGORI_USIA_4,
TOTAL_SPENT_squared, JML_PEMBELIAN_squared,	KATEGORI_USIA_5, KATEGORI_USIA_6,
TOTAL, LABEL	KATEGORI_USIA_7, KODE_DIDIK, KELUAR_BLN

2.4. Modeling

The development of the predictive model involved three machine learning models: random forest, GBDT, and DeepFM. The dataset was split into 80% for training and 20% for testing. Initially, these models were trained with default parameters, followed by optimization to enhance performance. Cross-validation was performed using k -fold ($k=4$) on the training data. For each fold, the training data was split into k subsets, and SMOTEENN was applied to balance the data within the current fold. The models were then trained on the balanced $k-1$ subsets and validated on the remaining subset. Performance metrics were recorded for each fold, and the best-performing fold was identified based on these metrics. The model was retrained using this best-performing fold and subsequently used to make predictions on the test data, ensuring robustness. Additionally, optimization included feature selection through backward feature elimination and hyperparameter tuning using hyperband to identify the best features and parameters for each model. Following the individual model training, a stacking process was implemented, with logistic regression serving as the meta-learner to further enhance predictive accuracy.

2.4.1. Model random forest

Random forest is an ensemble learning method used for classification and regression tasks. Its operation involves building multiple decision trees during the training phase and combining their predictions

to make the final prediction. Each decision tree is constructed using a random subset of the training data and random subset of features, introducing diversity and reducing overfitting [8]. In this research, random forest model was constructed using the RandomForestClassifier library with default parameters: max_depth=5, n_jobs=-1, oob_score=True, and random_state=42.

2.4.2. Model gradient boosting decision tree

GBDT is an ensemble model used for regression and classification tasks. This model constructs multiple weak learners (usually decision trees) sequentially, where each tree attempts to correct the errors made by the preceding one. This process continues until a predetermined number of learners is created or until certain stopping criteria are met. This is achieved by minimizing the loss function (method parameter) with each base model added [14]. In this research, GBDT model was constructed using the GradientBoostingClassifier library with default parameters: max_depth=5, n_estimators=100, oob_score=True, and random_state=42.

2.4.3. Model DeepFM

DeepFM is a hybrid model that effectively captures low-order and high-order feature interactions by combining the linear component of factorization machines (FM) and deep neural network (DNN) [11]. The DeepFM model structure consist of several components: the linear component (sparse feature and dense embedding), the factorization machine layer component, the deep neural network component (hidden layer). The linear component deals with categorical features, encoding them into both sparse feature space and dense embedding vectors. It captures linear interactions by summing sparse feature values and calculating dot products of dense embeddings. The FM layer focuses on pairwise interactions, computing the inner product of feature embeddings to capture interactions between different features. The DNN component captures high-order interactions through multiple layers of neural networks, learning complex patterns and feature representations. It processes dense embeddings and predicts the target variable using non-linear transformations.

2.4.4. Model optimization

Model optimization in this research involved several techniques, starting with feature selection using backward feature elimination. In this method, a significance level (SL) of 0.05 was set, corresponding to a 95% confidence level. The p-value was calculated for each feature, and any feature with a p-value greater than the significance level was removed. If a feature's p-value did not exceed the significance level, the elimination process was halted. This iterative approach continued until no additional features met the removal criteria, ultimately reducing the number of features from 78 to 59, as shown in Table 4.

Table 4. Final feature after feature selection

Numerical feature	Categorical feature
CICILAN, DP, JML_ANGSURAN,	JNS_KLM_1, JNS_KLM_2, JNS_JUAL_1,
JML_ANGSURAN_MEAN, mtr_hrgjual,	STS_RUMAH_1, STS_RUMAH_2, TUJU_PAKAI_1,
BUYING_CYCLE, RECENCY, TOTAL_SPENT,	TUJU_PAKAI_2, TUJU_PAKAI_3, TUJU_PAKAI_4,
CICILAN_log, DP_log, JML_ANGSURAN_log,	TUJU_PAKAI_5, TUJU_PAKAI_6, TUJU_PAKAI_7,
JML_ANGSURAN_MEAN_log,	KODE_KERJA_0, KODE_KERJA_1, KODE_KERJA_2,
BUYING_CYCLE_log, RECENCY_log,	KODE_KERJA_5, KODE_KERJA_6, KODE_KERJA_7,
TOTAL_SPENT_log, CICILAN_squared,	KODE_KERJA_8, KODE_KERJA_11, KODE_KERJA_12,
DP_squared, JML_ANGSURAN_squared,	KODE_KERJA_13, KODE_KERJA_14,
JML_ANGSURAN_MEAN_squared,	KODE_KERJA_15, KODE_KERJA_16, CUB_SPORT_2,
mtr_hrgjual_squared, BUYING_CYCLE_squared,	CUB_SPORT_3, KATEGORI_USIA_1,
RECENCY_squared, TOTAL_SPENT_squared,	KATEGORI_USIA_3, KATEGORI_USIA_4,
TOTAL	KATEGORI_USIA_5, KATEGORI_USIA_7,
	KODE_DIDIK, KELUAR_BLN

Hyperparameter tuning was also carried out using hyperband, an efficient algorithm that dynamically allocates resources to the most promising hyperparameter configurations, significantly reducing computational costs compared to traditional grid search methods [28]. In this research, hyperband was employed to identify the best parameters for the three models, using several parameter options listed in Table 5, with a hyperband configuration set at max_iteration=27 and eta=3 for each model. This tuning process was applied to models utilizing both the full set of features and the reduced set, with the optimal parameters detailed in Table 6.

The hyperparameter tuning results revealed distinct strategies depending on feature selection. For the random forest model, the feature-selected version opted for a deeper tree (8 vs. 5) and the Gini criterion,

while the all-features model adopted more conservative settings. In the GBDT, feature selection resulted in fewer estimators (15 vs. 45) but deeper trees and stricter splits, indicating a more efficient learning process. The DeepFM model, when using feature selection, employed a smaller batch size (64 vs. 256) and a simpler architecture but a higher learning rate (0.7356 vs. 0.04129), suggesting faster adaptation with fewer, more significant features. Overall, models with feature selection were more aggressive and optimized for a smaller feature set, whereas models utilizing all features took a more cautious approach to prevent overfitting.

Lastly, a stacking model approach was utilized in three variations. The first approach used base models (random forest, GBDT, and DeepFM) with default configurations. The second approach used base model that applied hyperparameter tuning using all features, and the third approach used base model that applied hyperparameter tuning with feature selection. This ensemble method aimed to leverage the strengths of each base model and improve overall prediction accuracy, particularly for predicting repeat order customers.

Table 5. List of parameters to find best parameter

Model	List parameter to find best parameter
RF	'criterion': hp.choice('c', ('gini', 'entropy')), 'bootstrap': hp.choice('b', (True, False)), 'class_weight': hp.choice('cw', ('balanced', 'balanced_subsample', None)), 'max_depth': hp.quniform('md', 2, 10, 1), 'max_features': hp.choice('mf', ('sqrt', 'log2', None)), 'min_samples_split': hp.quniform('msp', 2, 20, 1), 'min_samples_leaf': hp.quniform('msl', 1, 10, 1)
GBDT	'learning_rate': hp.uniform('lr', 0.01, 0.2), 'subsample': hp.uniform('ss', 0.8, 1.0), 'max_depth': hp.quniform('md', 2, 10, 1), 'max_features': hp.choice('mf', ('sqrt', 'log2', None)), 'min_samples_leaf': hp.quniform('mss', 1, 10, 1), 'min_samples_split': hp.quniform('mss', 2, 20, 1)
DeepFM	'optimizer': hp.choice('o', ['rmsprop', 'adagrad', 'adamax']), 'learning_rate': hp.loguniform('learning_rate', -5, 0), 'batch_size': hp.choice('batch_size', [64, 128, 256]), 'l2_reg_linear': hp.loguniform('l2_reg_linear', -10, -5), 'l2_reg_embedding': hp.loguniform('l2_reg_embedding', -10, -5), 'l2_reg_dnn': hp.loguniform('l2_reg_dnn', -10, -5), 'dnn_dropout': hp.uniform('dnn_dropout', 0, 0.5), 'dnn_activation': hp.choice('dnn_activation', ['relu', 'sigmoid', 'tanh']), 'seed': hp.choice('seed', [0, 1024]), 'dnn_hidden_units': hp.choice('dnn_hidden_units', [(256, 128, 64), (64, 32, 16), (30, 20, 10)])

Table 6. The optimal parameter for each model

Model	Best parameter
RF	'n_estimators': 5, 'bootstrap': False, 'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 5, 'max_features': 'log2', 'min_samples_leaf': 5, 'min_samples_split': 15
RF (Feature Selection)	'n_estimators': 5, 'bootstrap': False, 'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 8, 'max_features': 'log2', 'min_samples_leaf': 8, 'min_samples_split': 6
GBDT	'n_estimators': 45, 'learning_rate': 0.18861818786755138, 'max_depth': 4, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 7, 'subsample': 0.9025409533959227
GBDT (Feature Selection)	'n_estimators': 15, 'learning_rate': 0.19080522396266575, 'max_depth': 9, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 4, 'subsample': 0.9442618269109583
DeepFM	'iterations': 9, 'batch_size': 256, 'dnn_activation': 'relu', 'dnn_dropout': 0.09398487307992287, 'dnn_hidden_units': (256, 128, 64), 'l2_reg_dnn': 0.00015427748646839538, 'l2_reg_embedding': 0.0006685422033088367, 'l2_reg_linear': 0.0005985209123729851, 'learning_rate': 0.041293725390164925, 'optimizer': 'adamax', 'seed': 0
DeepFM (Feature Selection)	'iterations': 9, 'batch_size': 64, 'dnn_activation': 'relu', 'dnn_dropout': 0.043312236659931225, 'dnn_hidden_units': (30, 20, 10), 'l2_reg_dnn': 7.363305306789893e-05, 'l2_reg_embedding': 0.0014565149402621209, 'l2_reg_linear': 0.0012858385266040664, 'learning_rate': 0.7355648719703358, 'optimizer': 'adamax', 'seed': 1024

2.5. Evaluation

The performance evaluation in cases of class imbalance in this research will focus on several key metrics: accuracy, AUC-ROC, log loss, weighted average precision, weighted average recall, and weighted average F1-score. These metrics are crucial for providing a comprehensive and fair assessment of the model's ability to handle class imbalance. By evaluating these metrics, we can determine how well the model differentiates between majority and minority classes and ensure that the predictions are not biased towards the majority class. Accuracy measures how successful a classification model is at making correct predictions. For binary classification, accuracy can be calculated using (1).

$$Accuracy = \frac{True\ Positives + True\ Negatives}{Total\ Instances} \quad (1)$$

AUC-ROC measures the classification model's performance in distinguishing between positive and negative classes. The ROC curve is a graph that illustrates the relationship between the true positive rate (Recall) and false positive rate at various prediction thresholds. AUC-ROC calculates the area under the ROC curve. The AUC-ROC value ranges from 0 to 1, where a value closer to 1 indicates that the model is better at distinguishing between positive and negative classes, the formula is defined as (2).

$$AUC - ROC = \frac{\sum_{ins_i \in \text{positive class}} rank_{ins_i} - \frac{M*(M+1)}{2}}{M*N} \quad (2)$$

Where $rank_{ins_i}$ is the serial number of the sample, M is the number of positive samples, and N is the number of negative samples. Log loss measures the likelihood estimation of the prediction probability, the formula is defined as (3).

$$L_{\text{Log_loss}} = -\frac{1}{N} \sum_{i=1}^N (y_i \log(p_i) + (1 - y_i) \log(1 - p_i)) \quad (3)$$

Precision measures the level of accuracy between the predicted results provided by the model and the actual available data, precision is defined as (4).

$$Precision = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positive}} \quad (4)$$

The recall metric evaluates how well a model can recognize all accurately identified data samples in the true class within the dataset, recall is defined as (5).

$$Recall = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negative}} \quad (5)$$

The F1-score gives a comprehensive assessment of the model's performance by incorporating both precision and recall into a single value, the formula is defined as (6).

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (6)$$

The weighted average provides a more accurate assessment of the model's performance in cases of imbalanced data. By appropriately weighting each class based on its actual distribution, the weighted average allows us to evaluate the model's performance by considering the relative importance of each class. Weighted average precision (7), recall (8), and F1-score (9) consider class imbalance by assigning weights to the precision, recall, and F1-score of each class based on the proportion of that class in the dataset.

$$Weighted Precision = \sum_{i=1}^n \left(\frac{\text{Number of samples in Class } i}{\text{Total samples}} \right) \times Precision \text{ Class } i \quad (7)$$

$$Weighted Recall = \sum_{i=1}^n \left(\frac{\text{Number of samples in Class } i}{\text{Total samples}} \right) \times Recall \text{ Class } i \quad (8)$$

$$Weighted F1score = \sum_{i=1}^n \left(\frac{\text{Number of samples in Class } i}{\text{Total samples}} \right) \times F1score \text{ Class } i \quad (9)$$

3. RESULTS AND DISCUSSION

After completing all stages, the models were evaluated on the testing data. To enrich the analysis of model performance, this research also compared the results with a baseline model using logistic regression. This comparison aimed to assess whether the added complexity of more advanced models provided significant advantages over the simpler baseline model. The evaluation metrics for the three models whether using default settings, hyperparameter tuning using all feature, hyperparameter tuning using feature selection, or the baseline model are detailed in Table 7.

For the random forest model, feature selection and hyperparameter tuning led to substantial improvements. The default configuration yielded an accuracy of 0.61 and an AUC-ROC of 0.66, suggesting moderate classification capability. Hyperparameter tuning using all features slightly improved accuracy to 0.62, though AUC-ROC decreased to 0.65. However, after feature selection, accuracy increased significantly to 0.76, and the weighted average F1-score rose to 0.82, indicating enhanced performance in handling class imbalance and generating more accurate predictions. Additionally, the log loss metric decreased from 0.49 in

the default setting to 0.41 after feature selection and tuning, reflecting an improvement in the model's ability to produce well-calibrated probability estimates.

Table 7. The comparison of performance measurement results from all models

Model		Accuracy	AUC-ROC	Weighted avg precision	Weighted avg recall	Weighted avg F1-score	Log loss
Random forest	Default	0.61	0.66	0.92	0.61	0.71	0.49
	All feature and hyperparameter tuning	0.62	0.65	0.92	0.62	0.72	0.51
	Feature selection and hyperparameter tuning	0.76	0.66	0.91	0.76	0.82	0.41
Gradient boosting decision tree	Default	0.95	0.66	0.95	0.95	0.93	0.21
	All feature and hyperparameter tuning	0.95	0.66	0.95	0.95	0.93	0.21
	Feature selection and hyperparameter tuning	0.94	0.66	0.92	0.94	0.92	0.21
DeepFM	Default	0.60	0.46	0.89	0.60	0.71	6.31
	All feature and hyperparameter tuning	0.60	0.46	0.89	0.60	0.71	6.30
	Feature selection and hyperparameter tuning	0.60	0.46	0.89	0.60	0.71	6.31
Stacking – logistic regression	Default	0.95	0.67	0.95	0.95	0.93	0.19
	All feature and hyperparameter tuning	0.95	0.66	0.95	0.95	0.93	0.19
	Feature selection and hyperparameter tuning	0.95	0.67	0.95	0.95	0.92	0.20
Model baseline – logistic regression		0.85	0.52	0.89	0.85	0.87	0.68

The GBDT model showed consistent performance across different configurations. The default model had an accuracy of 0.95 and an AUC-ROC of 0.66, with a weighted average F1-score of 0.93. Hyperparameter tuning using all features did not significantly alter these metrics, indicating that GBDT model was already well-optimized. After feature selection, the weighted average F1-score slightly decreased to 0.92, while accuracy and AUC-ROC remained stable, reflecting robustness to feature changes. The log loss for this model remained low at 0.21 across all configurations, indicating high confidence in its predictions.

In contrast, the DeepFM model struggled in all scenarios, which contradicts Huang's [11] findings in predicting repeat order customers compared to decision tree-based models like random forest and GBDT. Whether using default settings, hyperparameter tuning using all feature, or hyperparameter tuning using feature selection, accuracy stayed at 0.60, and AUC-ROC remained low at 0.46. The consistent weighted average F1-score of 0.71 and high log loss values, ranging from 6.30 to 6.31, indicated significant challenges in dealing with class imbalance, suggesting that DeepFM may not be ideal for such tasks.

The stacking model using logistic regression as the meta-learner consistently outperformed other models, aligning with earlier studies by [12], [15], [19], [21], which demonstrated that ensemble methods typically offer superior predictive performance. The default stacking model achieved an accuracy of 0.95 and an AUC-ROC of 0.67. While hyperparameter tuning with all features slightly decreased the AUC-ROC to 0.66, accuracy and the weighted average F1-score remained high at 0.95 and 0.93, respectively. After feature selection, the AUC-ROC improved back to 0.67, though the weighted average F1-score slightly decreased to 0.92. This highlights the stacking model's strength in managing class imbalance and delivering accurate, balanced predictions. Notably, the log loss of the stacking model was the lowest across all models, reaching 0.19, which indicates superior reliability in its probability predictions.

Compared to the baseline logistic regression, which had an accuracy of 0.85 and an AUC-ROC of 0.52, the advanced models demonstrated significant improvements. The baseline's weighted average F1-score of 0.87 and log loss of 0.68 further underscore the benefits of using more complex models like random forest, GBDT, and especially stacking with logistic regression, which delivered superior performance across various metrics.

Overall, the stacking model with logistic regression as the meta-learner delivered the best performance, consistently achieving high accuracy, weighted average F1-scores, and low log loss across all configurations. While the random forest model showed significant improvement after feature selection and hyperparameter tuning, it still fell short compared to the GBDT and stacking models. The DeepFM model consistently underperformed across all configurations. Although the baseline model was less powerful, it provided a useful benchmark, highlighting the performance gains offered by more advanced models.

The application of backward feature elimination and hyperband was particularly effective in enhancing the random forest model, showing that these methods can improve model performance by optimizing feature selection and configuration. However, for the GBDT, performance remained robust and stable, even with minimal adjustments. Unfortunately, these methods did not significantly enhance the performance of the DeepFM model. Stacking methods across the models did not substantially increase accuracy but provided consistent and stable performance.

4. CONCLUSION

This research conducted experiments to develop predictive models for converting leads into repeat order customers, using random forest, GBDT, and DeepFM. Model optimization was performed through feature selection using backward feature elimination and hyperparameter tuning with hyperband. These processes were particularly important given the imbalanced nature of the dataset, as they aimed to enhance the models' ability to handle class imbalance effectively. Additionally, the exploration of stacking models was carried out, with logistic regression as the meta-learner. The performance of these models on the test data was also compared with a baseline model using logistic regression to enrich the analysis of model performance. The results indicate that the Stacking model using the default base model configuration stands out as the most robust, achieving the highest scores in accuracy (0.95), AUC-ROC (0.67), log loss (0.19), weighted average precision (0.95), weighted average recall (0.95), and weighted average F1-score (0.93), effectively handling the imbalanced dataset.

To address the data imbalance, SMOTEENN was applied, but the challenge persisted, particularly affecting the DeepFM model's performance. Given this underperformance, future research should explore alternative models like DeepForest, DeepGBM, and other neural networks, which may be better suited for predicting repeat order customers. Additionally, further studies could investigate more nuanced data-level preprocessing methods, such as ADASYN or SMOTETomek, and hybrid approaches that integrate over-sampling, under-sampling, and ensemble techniques. While methods like backward feature elimination and hyperband offered some improvements, especially in handling class imbalance, their impact was limited. Thus, future research could benefit from techniques like recursive feature elimination, Lasso, Bayesian optimization, or AutoML to enhance model accuracy and better address class imbalance.

The stacking approach with logistic regression as the meta-learner did not yield significant performance improvements, suggesting that exploring different stacking combinations and meta-learners could be beneficial. Future studies should focus on refining these techniques and applying them to practical CRM scenarios to better manage and increase repeat customer orders. By addressing these areas, future research can contribute to more accurate predictive models and offer valuable insights for businesses conventional aiming to optimize their CRM strategies and boost customer retention.





REFERENCES

- [1] "Market Research Report 2022," *Fortune Business Insight*, 2023. <https://www.fortunebusinessinsights.com/customer-relationship-management-crm-market-103418> (accessed Jul. 19, 2023).
- [2] Reichheld F F, "The one number you need to grow," *Harvard Business Review*, vol. 81, no. 12, pp. 46–54, 2004.
- [3] B. Zhao, A. Takasu, R. Yahyapour, and X. Fu, "Loyal Consumers or one-time deal hunters: repeat buyer prediction for E-commerce," in *2019 International Conference on Data Mining Workshops (ICDMW)*, Nov. 2019, pp. 1080–1087, doi: 10.1109/ICDMW.2019.00158.
- [4] V. Arun, "Machine learning techniques for customer relationship management," *International Journal of Creative Research Thoughts (IJCRT)*, vol. 9, no. 6, pp. 753–763, 2021.
- [5] Y. Benhaddou and P. Leray, "Customer relationship management and small data - application of bayesian network elicitation techniques for building a lead scoring model," in *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, Oct. 2017, pp. 251–255, doi: 10.1109/AICCSA.2017.51.
- [6] R. Nygård and J. Mezei, "Automating lead scoring with machine learning: an experimental study," in *Proceedings of the Annual Hawaii International Conference on System Sciences*, 2020, vol. 2020-January, pp. 1439–1448, doi: 10.24251/hicss.2020.177.
- [7] P. Espadinha-Cruz, A. Fernandes, and A. Grilo, "Lead management optimization using data mining: A case in the telecommunications sector," *Computers & Industrial Engineering*, vol. 154, p. 107122, Apr. 2021, doi: 10.1016/j.cie.2021.107122.
- [8] S. B. Ayaz, "Lead Scoring with machine learning," University of Applied Sciences, 2023.
- [9] L. Yang, J. Wu, X. Niu, and L. Shi, "Towards purchase prediction: a voting-based method leveraging transactional information," in *2022 5th International Conference on Data Science and Information Technology (DSIT)*, Jul. 2022, pp. 1–5, doi: 10.1109/DSIT55514.2022.9943898.
- [10] S. Mortensen, M. Christison, B. Li, A. Zhu, and R. Venkatesan, "Predicting and defining B2B sales success with machine learning," in *2019 Systems and Information Engineering Design Symposium (SIEDS)*, Apr. 2019, pp. 1–5, doi: 10.1109/SIEDS.2019.8735638.
- [11] S. Huang, "AI-based repeat buyers prediction system using deep learning," in *2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP)*, Apr. 2021, pp. 800–806, doi: 10.1109/ICSP51882.2021.9408760.
- [12] D. Xu, W. Yang, and L. Ma, "Repurchase prediction based on ensemble learning," in *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCOM/IOP/SCI)*, Oct. 2018, pp. 1317–1322, doi: 10.1109/SmartWorld.2018.00229.





- [13] W. Zhang and M. Wang, "An improved deep forest model for prediction of e-commerce consumers' repurchase behavior," *PLOS ONE*, vol. 16, no. 9, p. e0255906, Sep. 2021, doi: 10.1371/journal.pone.0255906.
- [14] E. Kuric, A. Puskas, P. Demcak, and D. Mensatorisova, "Effect of low-level interaction data in repeat purchase prediction task," *International Journal of Human-Computer Interaction*, vol. 40, no. 10, pp. 2515–2533, May 2024, doi: 10.1080/10447318.2023.2175973.
- [15] M. Zhang, J. Lu, N. Ma, T. C. E. Cheng, and G. Hua, "A feature engineering and ensemble learning based approach for repeated buyers prediction," *International Journal Of Computers Communications & Control*, vol. 17, no. 6, Dec. 2022, doi: 10.15837/ijccc.2022.6.4988.
- [16] Y. Li and C. You, "Brand loyalty measurement model based on machine learning clustering algorithm," *Journal of Physics: Conference Series*, vol. 1982, no. 1, p. 012089, Jul. 2021, doi: 10.1088/1742-6596/1982/1/012089.
- [17] A. H. Kazmi, G. Shroff, and P. Agarwal, "Generic framework to predict repeat behavior of customers using their transaction history," in *2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, Oct. 2016, pp. 449–452, doi: 10.1109/WI.2016.0072.
- [18] M. R. Machado, S. Karray, and I. T. de Sousa, "LightGBM: an effective decision tree gradient boosting method to predict customer loyalty in the finance industry," in *2019 14th International Conference on Computer Science & Education (ICCSE)*, Aug. 2019, pp. 1111–1116, doi: 10.1109/ICCSE.2019.8845529.
- [19] Y. Shen, X. Xu, and J. Cao, "Reconciling predictive and interpretable performance in repeat buyer prediction via model distillation and heterogeneous classifiers fusion," *Neural Computing and Applications*, vol. 32, no. 13, pp. 9495–9508, Jul. 2020, doi: 10.1007/s00521-019-04462-9.
- [20] X. Zhai, P. Shi, L. Xu, Y. Wang, and X. Chen, "Prediction model of user purchase behavior based on machine learning," in *2020 IEEE International Conference on Mechatronics and Automation (ICMA)*, Oct. 2020, pp. 1483–1487, doi: 10.1109/ICMA49215.2020.9233677.
- [21] H. Zhang and J. Dong, "Prediction of repeat customers on E-commerce platform based on blockchain," *Wireless Communications and Mobile Computing*, vol. 2020, pp. 1–15, Aug. 2020, doi: 10.1155/2020/8841437.
- [22] P. Gokhale and P. Joshi, "A binary classification approach to lead identification and qualification," in *Smart Trends in Information Technology and Computer Communications: Second International Conference, SmartCom 2017*, 2018, pp. 279–291, doi: 10.1007/978-981-13-1423-0_30.
- [23] K. Sangaralingam, N. Verma, A. Ravi, S. W. Bae, and A. Datta, "High value customer acquisition & retention modelling – a scalable data mashup approach," in *2019 IEEE International Conference on Big Data (Big Data)*, Dec. 2019, pp. 1907–1916, doi: 10.1109/BigData47090.2019.9006106.
- [24] G. Mariscal, Ó. Marbán, and C. Fernández, "A survey of data mining and knowledge discovery process models and methodologies," *The Knowledge Engineering Review*, vol. 25, no. 2, pp. 137–166, Jun. 2010, doi: 10.1017/S0269888910000032.
- [25] P. Chapman *et al.*, "The CRISP-DM user guide," in *4th CRISP-DM SIG Workshop in Brussels in March*, 1999, p. 14.
- [26] A. Fernández, S. García, M. Galar, R. C. Prati, B. Krawczyk, and F. Herrera, "Data level preprocessing methods," in *Learning from Imbalanced Data Sets*, Cham: Springer International Publishing, 2018, pp. 79–121.
- [27] M. Lin, X. Zhu, T. Hua, X. Tang, G. Tu, and X. Chen, "Detection of ionospheric scintillation based on XGBoost model improved by SMOTE-ENN technique," *Remote Sensing*, vol. 13, no. 13, p. 2577, Jul. 2021, doi: 10.3390/rs13132577.
- [28] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, A. Talwalkar, A. Rostamizadeh, and A. Talwalkar, "Hyperband: a novel bandit-based approach to hyperparameter optimization," *Journal of Machine Learning Research*, vol. 18, no. 185, pp. 1–52, 2016.

BIOGRAPHIES OF AUTHORS



Deryan Everestha Maured     received her first degree from Bina Nusantara University, Information System, Jakarta, in 2019. She is currently a master's degree student at Bina Nusantara University, Master of Computer Science, Jakarta. Her main research interests focus on natural language processing, machine learning, data science, text mining, and recommendation system. She can be contacted at email: deryan.maured@binus.ac.id.



Gede Putra Kusuma     received Ph.D. degree in Electrical and Electronic Engineering from Nanyang Technological University (NTU), Singapore, in 2013. He is currently working as a Lecturer and Head of Department of Master of Computer Science, Bina Nusantara University, Indonesia. Before joining Bina Nusantara University, he was working as a Research Scientist in I2R – A*STAR, Singapore. His research interests include computer vision, deep learning, face recognition, appearance-based object recognition, gamification of learning, and indoor positioning system. He can be contacted at email: inegara@binus.edu.