Bulletin of Electrical Engineering and Informatics
Vol. 15, No. 1, February 2026, pp. 493~502
ISSN: 2302-9285, DOI: 10.11591/eei.v15i1.10161

Gift recommendation with multilabel clustering

Violitta Yesmaya, Rini Wongso
Department of Computer Science, School of Computer Science, Bina Nusantara University, Jakarta, Indonesia

Article Info

Article history:

Received Feb 26, 2025
Revised Nov 7, 2025
Accepted Dec 6, 2025

Keywords:

Decision tree
E-commerce
eXtreme gradient boosting
K-nearest neighbors
Multilabel
Random forest
Recommendation

ABSTRACT

In the rapidly evolving e-commerce landscape, personalized gift recommendation systems play a crucial role in enhancing customer satisfaction and driving sales. This study introduces a gift recommendation system using a multilabel clustering approach with the using four algorithms, aiming to provide personalized and accurate product suggestions. The proposed system compares the performance of four algorithms: K-nearest neighbors (K-NN), decision trees, random forest, and eXtreme gradient boosting (XGBoost). Through extensive model training and hyperparameter tuning, XGBoost demonstrated superior performance with a label ranking average precision score of 95% and minimal overfitting, outperforming other algorithms in accuracy and runtime. The results highlight the effectiveness of XGBoost in managing complex data and delivering precise recommendations, making it a valuable tool for improving user experience and increasing revenue in e-commerce platforms.

Corresponding Author:

Violitta Yesmaya
Department of Computer Science, School of Computer Science, Bina Nusantara University
Jakarta 11480, Indonesia
Email: vyesmaya@binus.edu

1. INTRODUCTION

In the rapidly evolving world of e-commerce, businesses constantly seek innovative ways to enhance customer experience and maintain a competitive edge. One such innovation is the implementation of a gift recommendation system, a tool that not only simplifies the shopping process but also plays a crucial role in customer retention and acquisition. Gifts are often used to recognize awards, celebrate special occasions, or acknowledge accomplishments, but selecting the right gift can be a confusing task [1]. A gift recommendation system addresses this challenge by offering personalized suggestions based on user preferences, past purchases, and browsing behaviour. By analysing this data, the system predicts which items will be interesting and useful to users, helping them make informed decisions and easily find products that are both in demand and necessary [2].

The convenience of having relevant suggestions at their fingertips enhances the shopping experience, reducing the time and effort needed to find the perfect gift. This ease of use significantly increases customer satisfaction, encouraging repeat visits and purchases. Personalization is a key driver of customer engagement in e-commerce, and a gift recommendation system leverages advanced algorithms to deliver tailored product suggestions that resonate with individual users. By understanding customer preferences, the system makes the shopping experience more enjoyable and efficient [3].

Artificial intelligence (AI) and machine learning (ML) play a crucial role in building effective recommendation systems. These technologies enable the analysis of vast amounts of data to uncover patterns and insights that drive accurate recommendations [4]. AI and ML algorithms can process user interactions, preferences, and product attributes to provide personalized and relevant suggestions. This not only enhances user experience but also helps in optimizing the system's performance. With the development of ML in the

recommendation system, it is supported by several algorithms to be able to produce the right level of accuracy and prediction. The following are several algorithms that are commonly used to develop recommendation systems:

- K-nearest neighbors (K-NN): this algorithm recommends items by finding similarities between users or products [5]. It identifies the 'k' most similar users or items and suggests products that these similar users have liked or interacted with. Advantages K-NN its works well for small to medium-sized datasets, and limitation K-NN may not perform well with very large datasets, and lack interactions.
- Decision trees: decision trees make recommendations by splitting the data into branches based on feature values, which helps in making decisions about which products to recommend based on user attributes and behaviours [6], [7]. Advantages decision trees can handle both numerical and categorical data, and for limitation easily overfit the training data, especially with deep trees, and may not perform well with very large datasets.
- Random forest: an extension of decision trees, random forest builds multiple decision trees and merges their results to improve recommendation accuracy [6]. Advantages of random forest it can handles large datasets and various features more efficiently, providing robust recommendations. Limitation of random forest slow process for train and predict with very large datasets.
- eXtreme gradient boosting (XGBoost): known for its performance and accuracy, XGBoost is an ensemble learning method that builds multiple models and combines their predictions. It is particularly effective for handling complex data, large datasets and making precise recommendations [8]. XGBoost have limitation such as can be more complex to tune compared to simpler models.

Personalized recommendations also help e-commerce platforms build stronger relationships with their customers. When shoppers feel that a platform understands their needs and preferences, they are more likely to return for future purchases, boosting customer loyalty and enhancing the platform's reputation as a trusted and user-friendly destination.

The implementation of a gift recommendation system can also lead to a significant increase in sales and revenue. By suggesting relevant products, the system encourages customers to explore more items and make additional purchases, resulting in higher average order values and increased sales volume. Moreover, the system's ability to highlight popular and trending products can drive impulse buying, further boosting sales. For instance, if a customer browsing for a gift sees a recommendation for a top-selling item that aligns with their needs, they are more likely to make a quick purchase, maximizing revenue through targeted marketing.

Customer retention is critical to the long-term success of any e-commerce business, and a gift recommendation system can enhance this by improving the overall shopping experience [9]. Personalized recommendations that consistently meet customer needs can increase the likelihood of repeat purchases [10]. Additionally, the system helps reduce the likelihood of abandoned carts by offering relevant product suggestions during the shopping process, leading to higher conversion rates. Satisfied customers are more likely to refer the platform to others, expanding the customer base.

In a competitive market, attracting new customers is equally important. A gift recommendation system can serve as a powerful marketing tool to draw in new shoppers by providing a personalized and seamless shopping experience. By differentiating themselves from competitors with relevant and thoughtful suggestions, e-commerce platforms can attract a broader audience, especially those looking for a hassle-free shopping experience for special occasions. Therefore, this experiment will create a gift recommendation system by selecting the best performance algorithm between K-NN, decision trees, random forest, and XGBoost.

2. METHOD

A recommendation system is designed to predict and suggest items that match a user's interests. It is commonly used by e-commerce platforms to boost product sales by recommending items that users may like or need. The system collects data from users, either directly or indirectly, to generate these recommendations [2], [11].

Direct data collection involves asking users to provide feedback on items, while indirect data collection involves observing user behaviour, such as items viewed on a website. The system then applies specific algorithms to this data to generate item recommendations, which are presented to users [11], [12].

Recommendation system typically adopt either collaborative filtering or content-based filtering. Collaborative filtering is a method that plays a significant role in the recommendation process and is the most commonly used method for designing recommendation systems. In this recommendation method, recommendations for each active user are obtained by comparing the preferences of other users who have rated products in a manner similar to the active user [13]. Essentially, collaborative filtering is based on

collecting and examining large amounts of information on behaviour, activities, or preferences, and anticipating a specific user's tastes by using similarities with other users. In the other hand, content-based filtering is a method that tries to recommend items to active users based on the degree of similarity that the user has positively rated in the past [13]. The flow of content-based filtering can be seen in the Figure 1.

Figure 1. Content-based filtering

2.1. K-nearest neighbor

K-NN is a classification method for a set of data based on learning previously classified data. K-NN is included in supervised learning, where the classification results are obtained based on the majority of the proximity of the distances of the categories in K-NN. This distance is square root of the sum squares of differences between opposite values in the vector [14]. The formulalized in (1) for the Euclidean distance.

$$euc = \sqrt{\left(\sum_{i=1}^{n}(pi - qi)^2\right)} \quad (1)$$

where pi is data training, qi is data testing, i is data variables, and n is data dimensions.

K-NN is a classification method that classifies data based on the proximity of new data to previously classified data. K-NN is a type of supervised learning, where the classification outcome is determined by the majority vote of the nearest neighbors, based on the distance between data points [14].

2.2. Decision tree

A decision tree is a method that models' data in the form of a tree structure, where incoming data traverses the tree to determine the class in which the data is classification systems based on multiple covariates for target variable [15]. Decision trees are also commonly used to explore data and uncover hidden relationships between multiple input variables and a target variable [16].

2.3. Random forest

Random forest is a method used for both classification and regression tasks. It leverages decision trees as base classifiers, which are constructed and combined to form the final model [17], [18]. The final classification result is determined by a voting process, where the majority vote from the individual decision trees is chosen as the final output of the random forest algorithm [19].

There are three key aspects of the random forest method: i) bootstrap sampling, which is used to build the individual prediction trees [20]; ii) each decision tree in the forest predicts using many classification and regression threes that using randomly selected training datasets and random subset to give a prediction [21]; and iii) the random forest combines the predictions from all the decision trees by majority vote to make a final classification [22]. An illustration of the random forest algorithm is shown in Figure 2. In random forest algorithms are settings the parameters for hyperparameters that can be tuned to optimize the model performance. Are the parameters can be figured out in Table 1.

2.4. XGBoost

XGBoost is a decision tree-based boosting algorithm. It incorporates regularization into the objective function to prevent overfitting and enhance the computation efficiency of gradient boosting machines (GBMs) [23]. XGBoost employs an ensemble learning method, which involves combining multiple base learners or classifiers to create a robust prediction model [23].

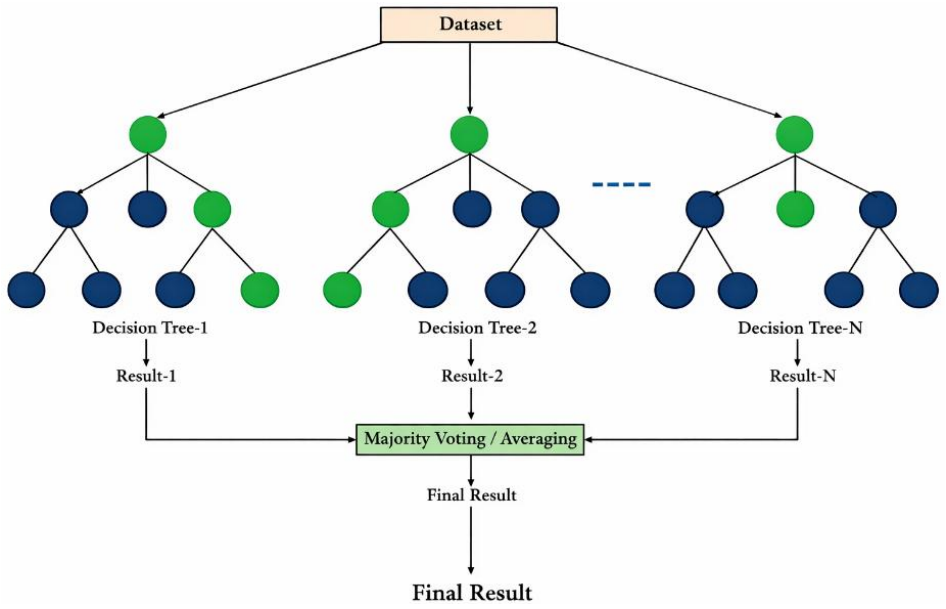

Figure 2. Random forest algorithm illustration

Table 1. Hyperparameter random forest

Parameter	Function
n_estimators	Number of trees in a random forest.
min_samples_split	The minimum number of samples is needed to split the internal node.
min_samples_leaf	The minimum number of samples required to be on a leaf node.
max_features	The number of features that are considered in determining the best division, if "auto" then the number of features is the square of the number of features.
max_depth	Maximum depth of the tree.
bootstrap	Specifies whether a bootstrap sample is used. If "False" then the entire dataset will be used to build the tree.

XGBoost is an efficient algorithm among the ML methods, because it easy to parallelism and high accuracy [24]. The performance of XGBoost is influenced by hyperparameters, which can be adjusted to optimize results. The hyperparameters for XGBoost are detailed in Table 2.

Table 2. XGBoost hyperparameters

Parameter	Function
Booster	Boosters used, by default *use* gbtree.
Colsample_bylevel	The number of sample ratios at each level. Samples will be made every reach new depth on the tree.
Colsample_bynode	Number of sample ratios for each node.
Colsample_bytree	Number of sample ratios in a column to create a new tree.
Gamma	Parameter penalty from regularization.
learning_rate	The learning rate is used to prevent overfitting. Will shrink the weight of the feature every time the boosting process is completed.
max_delta_step	Maximum delta step value allowed for each leaf output.
max_depth	The depth of a tree, the deeper it is, the more complex it is (overfitting*)*.
min_child_weight	Minimum value of the weight required by the child node.
n_estimators	Number of trees used.
n_jobs	Number of jobs running in parallel.
random_state	Controlling randomization on bootstrapping samples when building a tree.
reg_alpha	L1 regularization at weight.
reg_lambda	L2 regularization at weight.
scale_pos_weight	Controlling the balance between positive and negative weights which is useful in unbalanced classes
subsample	Number of sample ratios used by the training process.
tree_method	Algorithms used to build tree.
validate_parameters	When true, XGBoost will validate the parameter to check if the parameter is used or not.
verbosity	To display information during training.

3. RESULTS AND DISCUSSION

Typically, reward recommendation systems predict popular and necessary products by analyzing users' purchase histories and the products they have viewed. Table 3 is an explanation of several processes and categories commonly used in e-commerce recommendation systems.

Table 3. Existing e-commerce recommendation system

Category	Information
Most in demand	The recommended items are those with the highest search frequency, determined by calculating a score that indicates the item is in high demand when the user views this section.
Our recommendations	The system recommends goods based on the user's purchase history, the products they have viewed, and items of interest determined by a gradient-based analysis.
Popular items	Using the same concept as the "Most in Demand" section, this part of the system will recommend products currently popular based on the highest search frequency. The difference is that the recommendations in this section are updated once a day and apply to all users.
Special promos	Special promo is a recommendation system based on user affinity and category affinity. User affinity refers to how similar users are to each other, while category affinity measures the similarity between different categories. The latest promotions will be highlighted at the forefront of this section.
Recommended products	The system uses an infinite recommendation approach, where items are suggested continuously. This approach incorporates various strategies, including discounts, user history, popular products, top campaign items, and new arrivals.

According to a survey of 200 respondents in Java, Indonesia, gifters prefer gift recommendations that align with the recipients' interests, needs, and desires. However, as noted in Table 1, the current recommendation system does not address these preferences. Therefore, this experiment will introduce a new recommendation system that incorporates personalization based on the recipients' interests, needs, and desires. In this research we used proposed methods with some steps in it such as preprocessing, hyperparameter tuning, model training, model testing, and the last has resulted and evaluation as shown in Figure 3.

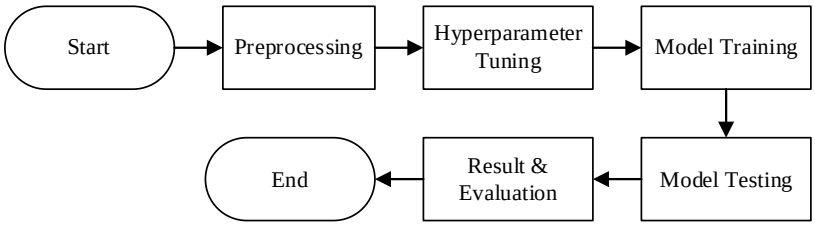

Figure 3. Proposed method

3.1. Preprocessing

Preprocessing is the steps to perform the row data before running analysis, so the data becomes clean, consistent, and learnable. It reduces noise and bias, prevents leakage, and helps model coverage and generalize. In this step involves several steps: reducing outlier categories, grouping age data that is shown in Table 4.

Table 4. Example of age data grouping

Gender	Age group	Status	Job	Interest	Category
Woman	Teens	Single	Full time employeed	Fashion, gaming, and culinary	Voucher
Woman	Teens	Single	Full time employeed	Fashion, gaming, and culinary	Tablet
Woman	Teens	Single	Full time employeed	Culinary and gadget	Voucher
Woman	Adult	Married	Unemployeed	Fashion and culinary	Home living
Man	Seniors	Married	Full time employeed	Gadget and culinary	Home living
Man	Seniors	Married	Full time employeed	Gadget and culinary	Tablet
Man	Teens	Single	Unemployeed	Gaming and gadget	Tablet
Man	Teens	Single	Unemployeed	Gaming and gadget	Voucher

Here filtering data is an example based on gender, age, status, job, fashion, gaming, culinary, gadget, voucher, home living, and tablet (as shown in Table 5). By encoding the data to fit the model's

requirements (as shown in Figure 4). The process of one hot encoding which converts gender, age, status, and occupation in the form of categorical data into numerical data which is then processed by a machine learning model. After preprocessing, the data is split into 80% for training and 20% for testing.

Table 5. Data filtering examples

Gender	Age group	Status	Job	Fashion	Gaming	Culinary	Gadget	Voucher	Home living	Tablet
Woman	Teens	Single	Full time employeed	1	1	1	0	1	0	1
Woman	Adult	Married	Unemployeed	1	0	0	0	0	1	0
Man	Seniors	Married	Full time employeed	0	0	1	1	0	1	1
Man	Teens	Single	Unemployeed	0	1	0	1	1	0	1

Woman	Man	Teens	Adult	Seniors	Single	Married	Full time employeed	Unemployeed	Fashion	Gaming	Culinary	Gadget	Voucher	Home living	Tablet
1	0	1	0	0	1	0	1	0	1	1	1	0	1	0	1
1	0	0	1	0	0	1	0	1	1	0	0	0	0	1	0
0	1	0	0	1	0	1	1	0	0	0	1	1	0	1	1
0	1	1	0	0	1	0	0	1	0	1	0	1	1	0	1

Figure 4. Example of data after one hot encoding

3.2. Hyperparameter tuning

Hyperparameters are parameters set before the learning process of a model begins. These parameters can be adjusted and have a direct impact on the model's performance. Optimizing hyperparameters is a crucial step in improving the accuracy and effectiveness of ML models [25]. Hyperparameter tuning is a process used to determine the best-performing parameters for each model [26]. Once the optimal hyperparameters are identified, the models undergo the training and testing process again with these parameters, resulting in improved algorithm performance. In this context, a multi-label classification model is used for the reward recommendation system. The algorithms involved include K-NN, decision tree, random forest, and XGBoost. Before training, the models go through a hyperparameter tuning process to obtain the best-performing hyperparameters.

3.3. Model training

The training data, consisting of 194 types of users, will be inserted into models that have already undergone the hyperparameter tuning process. These models will learn from the prepared training data to provide accurate predictions that align with the characteristics of the prize recipients.

3.4. Model testing

In the model testing phase, the models will be evaluated by comparing the actual data with the prediction results on the testing data. The prediction results are probability scores for 100 product categories, indicating how well each category matches the user's type. These scores range from zero to one, with higher scores representing a better fit for the user type. At the testing stage, the metrics do not directly indicate user preference for product categories; instead, they reflect the purchase behavior of different user types. This stage involves comparing the purchase predictions from the reward recommendation system with actual purchase data. The testing involved 49 user types, targeting 100 product categories with the highest number of purchases.

3.5. Result and evaluation

The evaluation will be conducted using several metrics: run time, training accuracy, test accuracy, Hamming loss, and the label ranking average precision score. These metrics will help assess the model's performance by accurately predicting product categories that match the data of the prize recipients.

Overfitting occurs when a model is more complex than necessary [27]. This condition means that the model becomes overly specific to the training data, resulting in poor performance during testing. When overfitting happens, the model loses its ability to generalize, leading to inaccurate predictions on new data [28]. The label ranking average precision score is used to evaluate the ranking of predictions generated by the model. This metric is specifically designed for assessing multi-label ranking problems [29]. Meanwhile, accuracy refers to the proportion of correctly predicted data points, and it is calculated using (2) [30]:

$$Accuracy = \frac{TP+TN}{TP+FN+FP+TN} \quad (2)$$

where: True positives (TP)=the correct number of positive objects classified;
True negatives (TN)=the correct number of classified negative objects;
False positives (FP)=number of negative objects classified as positive;
False negatives (FN)=number of positive objects classified as negative.

Hamming loss is a commonly used evaluation metric for multi-label data testing, as it calculates the error rate of the classification results [31]. It measures how many class labels are misclassified. A perfect performance is indicated by a Hamming loss value of zero; the smaller the Hamming loss value, the better the system's performance [32]. The formula for calculating Hamming loss is shown in (3):

$$Hamming\ loss = \frac{1}{m}\sum_{i=1}^{m}\frac{|Yi\Delta Y_i^p|}{Q} \quad (3)$$

where m=a lot of data is classified,
Q=number of class labels,
$|Yi\Delta Y_i^p|$=the number of classification errors that occur.

Experiment is conducted by dividing data set into 80% of training data and 20% of testing data, and the result can be seen in Table 6.

Table 6. Comparison results before hyperparameter

	XGBoost	Random forest	Decision tree	K-NN
Time	00:07.7	00:12.8	00:00.4	00:01.1
Train accuracy	0.79	0.95	0.95	0.74
Test accuracy	0.78	0.78	0.73	0.78
Hamming loss	0.04	0.04	0.05	0.04
Label ranking average precision score	0.94	0.94	0.89	0.92

From the analysis of the test results that have been carried out, the XGBoost model has the best performance where the average ranking accuracy label score reaches 0.94 and the Hamming loss score reaches 0.04. It can also be seen that the accuracy score in the training and testing of the XGBoost model is not very different. This explains that the XGBoost model is not overfitting while in the random forest model the model is overfitting. Hyperparameter tuning of the XGBoost model identifies optimal parameters to improve performance and mitigate potential overfitting.

The results obtained are good enough, to improve the performance of the ML model, the author performs hyperparameter tuning on each algorithm to obtain the parameters that have the best performance. After obtaining the best parameters, then the algorithm comparison is again compared whose results can be seen in Table 7.

Table 7. Comparison results after hyperparameter tuning

	XGBoost	Random forest	Decision tree	K-NN
Time	00:07.4	00:09.6	00:00.4	00:01.1
Train accuracy	0.74	0.77	0.95	0.74
Test accuracy	0.78	0.78	0.73	0.78
Hamming loss	0.04	0.04	0.05	0.05
Label ranking average precision score	0.95	0.93	0.89	0.92

Here about model performance comparison, from this graph shows that the best result is from XGBoost which is better than K-NN, decision tree, and random forest on the test data. The performance was evaluated using accuracy, precision, recall, F1-score, and AUC-ROC.

After performing hyperparameter tuning, the XGBoost model still has the best performance with a ranking average precision score of 0.95 and a Hamming loss of 0.04. This is depicted from model performance comparison as shown in Figure 5. The results are quite good results in the product recommendation system considering that the tests carried out show purchases in the product category.

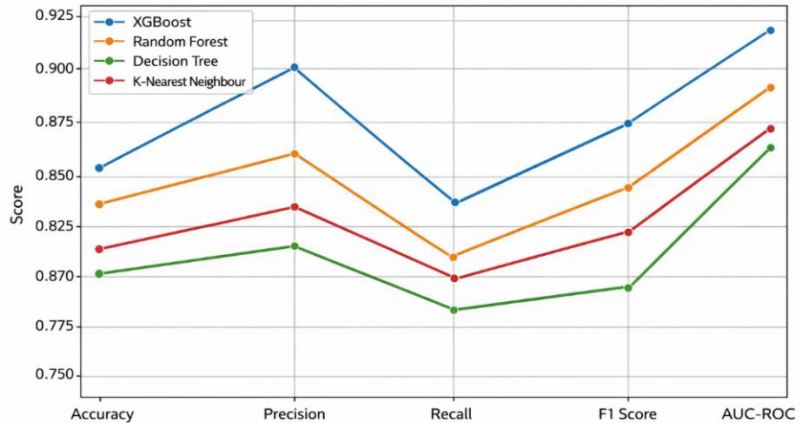

Figure 5. Model performance comparison

4. CONCLUSION

In this study, several algorithm tests were conducted to identify the most suitable algorithm for the reward recommendation system. With 49 types of users, the model was developed using multi-label classification, and various algorithms—K-NN, decision tree, random forest, and XGBoost—were compared after undergoing hyperparameter tuning.

The hyperparameter tuning process was used to optimize each model's parameters to achieve the best performance. Among the algorithms tested, XGBoost emerged as the most suitable for the reward recommendation system, achieving a label ranking average precision score of 95%, a Hamming loss of 0.04, and a processing time of 7.4 seconds, with no overfitting observed. The reward recommendation system successfully meets the company's needs and aligns with expectations by accurately displaying product categories that match the characteristics of the prize recipients.

FUNDING INFORMATION

The publication of this paper is financially supported by BINUS University, Indonesia without specific grant.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Violitta Yesmaya	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	
Rini Wongso		✓	✓			✓	✓	✓	✓	✓	✓			✓

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

The authors declare there are no conflicts of interest regarding this journal.

DATA AVAILABILITY

The data availability statement is a valuable link between a paper's results and the supporting evidence. Derived data supporting the findings of this study are available from the corresponding author [vy] on request.

REFERENCES

[1] Y. Fu, D. B. Dose, and R. Dimitriu, "Gift giving in the age of AI: The role of social closeness in using AI gift recommendation tools," *Psychology & Marketing*, vol. 41, no. 10, 2024, doi: 10.1002/mar.22050.

[2] P. Tomar, P. Arora, A. Goel, and D. Saini, "Social Profile Based Gift Recommendation System," *International Journal of Computer Science and Information Technologies*, vol. 5, no. 3, pp. 3670-3673, 2014.

[3] M. Ylilehto, H. Komulainen, and P. Ulkuniemi, "The critical factors shaping customer shopping experiences with innovative technologies," *Baltic Journal of Management*, vol. 16, no. 5, pp. 661–680, 2021, doi: 10.1108/BJM-02-2021-0049.

[4] X. Zhao and P. Keikhosrokiani, "Sales prediction and product recommendation model through user behavior analytics," *Computers, Materials and Continua*, vol. 70, no. 2, pp. 3855–3874, 2022, doi: 10.32604/cmc.2022.019750.

[5] B. Hssina, A. Grota, and M. Erritali, "Recommendation system using the k-nearest neighbors and singular value decomposition algorithms," *International Journal of Electrical and Computer Engineering*, vol. 11, no. 6, pp. 5541–5548, 2021, doi: 10.11591/ijece.v11i6.pp5541-5548.

[6] A. S. Salsabila, C. A. Sari, and E. H. Rachmawanto, "Classification of Movie Recommendation on Netflix Using Random Forest Algorithm," *Advance Sustainable Science, Engineering and Technology (ASSET)*, vol. 6, no. 3, pp. 1–8, 2024, doi: 10.26877/asset.v6i3.676.

[7] I. Portugal, P. Alencar, and D. Cowan, "The use of machine learning algorithms in recommender systems: A systematic review," *Expert Systems with Applications*, vol. 97, no. 2017, pp. 205–227, 2018, doi: 10.1016/j.eswa.2017.12.020.

[8] A. M. Purnamasari, "E-Commerce Product Recommendations using XGBoost with User Clusters and Clickstream," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 9, no. 4, pp. 5942–5948, 2020, doi: 10.30534/ijatcse/2020/258942020.

[9] B. Artha, I. Zahara, Bahri, and N. P. Sari, "Customer Retention: A Literature Review," *Social Science Studies*, vol. 2, no. 1, pp. 030–045, 2022, doi: 10.47153/sss21.2952022.

[10] P. Song and Y. Liu, "An xgboost algorithm for predicting purchasing behaviour on e-commerce platforms," *Tehnicki Vjesnik*, vol. 27, no. 5, pp. 1467–1471, 2020, doi: 10.17559/TV-20200808113807.

[11] V. U. Parikh and P. Shah, "E-commerce Recommendation System using Association Rule Mining and Clustering," *Research Journal*, vol. 4, pp. 148–155, 2015.

[12] P. Bellini, L. A. I. Palesi, P. Nesi, and G. Pantaleo, "Multi Clustering Recommendation System for Fashion Retail," *Multimedia Tools and Applications*, vol. 82, no. 7, pp. 9989–10016, 2023, doi: 10.1007/s11042-021-11837-5.

[13] P. B.Thorat, R. M. Goudar, and S. Barve, "Survey on Collaborative Filtering, Content-based Filtering and Hybrid Recommendation System," *International Journal of Computer Applications*, vol. 110, no. 4, pp. 31–36, 2015, doi: 10.5120/19308-0760.

[14] R. K. Halder, M. N. Uddin, M. A. Uddin, S. Aryal, and A. Khraisat, "Enhancing K-nearest neighbor algorithm: a comprehensive review and performance analysis of modifications," *Journal of Big Data*, vol. 11, no. 1, Dec. 2024, doi: 10.1186/s40537-024-00973-y.

[15] Y. Y. Song and Y. Lu, "Decision tree methods: applications for classification and prediction," *Shanghai Arch Psychiatry*, vol. 27, no. 2, pp. 130–135, 2015, doi: 10.11919/j.issn.1002-0829.215044.

[16] O. F.Y, A. J.E.T, A. O, H. J. O, O. O, and A. J, "Supervised Machine Learning Algorithms: Classification and Comparison," *International Journal of Computer Trends and Technology*, vol. 48, no. 3, pp. 128–138, 2017, doi: 10.14445/22312803/ijctt-v48p126.

[17] A. Primajaya and B. N. Sari, "Random Forest Algorithm for Prediction of Precipitation," *Indonesian Journal of Artificial Intelligence and Data Mining*, vol. 1, no. 1, p. 27, 2018, doi: 10.24014/ijaidm.v1i1.4903.

[18] G. Khanvilkar and D. Vora, "Product recommendation using sentiment analysis of reviews: A random forest approach," *International Journal of Engineering and Advanced Technology (IJEAT)*, vol. 8, no. 2, pp. 146–152, 2019.

[19] A. Paul, D. P. Mukherjee, P. Das, A. Gangopadhyay, A. R. Chintha, and S. Kundu, "Improved Random Forest for Classification," *IEEE Transactions on Image Processing*, vol. 27, no. 8, pp. 4012–4024, 2018, doi: 10.1109/TIP.2018.2834830.

[20] Z. Tang, Z. Mei, W. Liu, and Y. Xia, "Identification of the key factors affecting Chinese carbon intensity and their historical trends using random forest algorithm," *Journal of Geographical Sciences*, vol. 30, no. 5, pp. 743–756, May 2020, doi: 10.1007/s11442-020-1753-4.

[21] J. L. Speiser, M. E. Miller, J. Tooze, and E. Ip, "A comparison of random forest variable selection methods for classification prediction modeling," *Expert Systems with Applications*, vol. 134, pp. 93-101, 2019, doi: 10.1016/j.eswa.2019.05.028.

[22] A. Revathi, R. Kaladevi, K. Ramana, R. H. Jhaveri, M. R. Kumar, and M. S. P. Kumar, "Early Detection of Cognitive Decline Using Machine Learning Algorithm and Cognitive Ability Test," *Security and Communication Networks*, vol. 2022, 2022, doi: 10.1155/2022/4190023.

[23] L. Torlay, M. Perrone-Bertolotti, E. Thomas, and M. Baciu, "Machine learning–XGBoost analysis of language networks to classify patients with epilepsy," *Brain Informatics*, vol. 4, no. 3, pp. 159–169, 2017, doi: 10.1007/s40708-017-0065-7.

[24] M. Chen, Q. Liu, S. Chen, Y. Liu, C. H. Zhang, and R. Liu, "XGBoost-Based Algorithm Interpretation and Application on Post-Fault Transient Stability Status Prediction of Power System," *IEEE Access*, vol. 7, pp. 13149–13158, 2019, doi: 10.1109/ACCESS.2019.2893448.

[25] L. Li *et al.*, "A System for Massively Parallel Hyperparameter Tuning," *Proceedings of machine learning and systems*, pp. 230–246, 2020.

[26] K. Tyagi, C. Rane, R. Sriram, and M. Manry, "Unsupervised learning," *Artificial Intelligence and Machine Learning for EDGE Computing*, pp. 33-52, 2022, doi: 10.1016/B978-0-12-824054-0.00012-5.

[27] R. Roelofs *et al.*, "A meta-analysis of overfitting in machine learning," *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, 2019.

[28] L. Rice, E. Wong, and J. Z. Kolter, "Overfitting in adversarially robust deep learning," in *37th International Conference on Machine Learning, ICML 2020*, 2020, vol. PartF16814, pp. 8049–8074.

[29] C. Zhang, N. Du, W. Fan, Y. Li, C. T. Lu, and P. S. Yu, "Bringing semantic structures to user intent detection in online medical queries," *2017 IEEE International Conference on Big Data (Big Data)*, Boston, MA, USA, 2017, pp. 1019-1026, doi: 10.1109/BigData.2017.8258025.

[30] M. O. Sanjaya, S. Bukhori, and M. `A. Furqon, "Virtual Assistant for Thesis Technical Guide Using Artificial Neural Network," *Indonesian Journal of Artificial Intelligence and Data Mining*, vol. 6, no. 2, p. 188, 2023, doi: 10.24014/ijaidm.v6i2.23473.

[31] M. Serhan *et al.*, "Total iron measurement in human serum with a smartphone," *AIChE Annual Meeting, Conference Proceedings*, 2019.

[32] L. Zhou, X. Zheng, D. Yang, Y. Wang, X. Bai, and X. Ye, "Application of multi-label classification models for the diagnosis of diabetic complications," *BMC Medical Informatics and Decision Making*, vol. 21, no. 1, pp. 1–10, 2021, doi: 10.1186/s12911-021-01525-7.

BIOGRAPHIES OF AUTHORS

Violitta Yesmaya received the Computer Science degree in School of Computer Science Bina Nusantara University (BINUS) in 2011, and master's degree in Magister Technique Information from Bina Nusantara University (BINUS) in 2013. She start works in BINUS University as lecturer since 2012 in Department of Computer Science. Her research interests are software engineering, computing interaction, and deep learning modeling. She can be contacted at email: vyesmaya@binus.edu.

Rini Wongso received the Computer Science bachelor's (2009) and master's (2014) degree of Computer Science from Bina Nusantara University. She starts becoming Faculty Member of Computer Science in Bina Nusantara University in 2012. Her research interests are software engineering and artificial intelligence fields. She can be contacted at email: rwongso@binus.ac.id.