CoreID JOURNAL

Vol. 1 No. 1, July 2023 (58-65)
ISSN: 2987-6990
DOI: https://doi.org/10.60005/coreid.v1i2.11

Optimizing Bank Marketing Strategies Through Analysis Using Lightgbm

Erindra Reynaldi Diaz Aditya[1], Dhian Kartika Yudha Satria[2]
[12]Information System, UPN "Veteran" Jawa Timur

Article Info

Article history:

Keywords:

Bank Marketing Campaign
Strategy Optimization
Machine Learning
LightGBM

ABSTRACT

Marketing campaigns in a bank are one of the ways for the bank to achieve its organizational goals. Optimal marketing is a crucial factor for a bank's success in attracting and retaining customers. Therefore, if a bank's marketing campaigns are carried out suboptimally, it will be challenging to achieve the goals of those campaigns. In this case study, it can be observed that the number of customers who subscribe to fixed-term deposits is lower, with a proportion of 5289 customers making deposits and 5873 customers not making deposits. This research aims to optimize bank marketing strategies by applying analysis using the LightGBM algorithm, which is a highly effective and efficient Gradient Boosting Decision Tree algorithm. This approach facilitates the design of more optimal marketing strategies. The accuracy score of the predictive model generated is 0.8584, with an F1 score of 0.8564, including 974 true negatives and 943 true positives.

Corresponding Author:

Erindra Reynaldi Diaz Aditya
Information System, UPN "Vetertan" Jawa Timur,
Pagesangan Permai No.5, Surabaya, Indonesia
Email: ereynaldi.da@gmail.com

1. INTRODUCTION

A marketing campaign, is an organized and planned action intentionally created to achieve specific objectives [1]. For example, it could aim to increase awareness of a new product or obtain feedback from customers. Banks, as one of the industries that greatly require proper marketing strategies, can benefit from accurate and efficient marketing data analysis and visualization. To achieve this, the use of LightGBM (Light Gradient Boosted Machine) data analysis technique for bank marketing can be a solution to enhance the effectiveness of marketing campaign strategies. LightGBM is a gradient boosting framework based on decision trees developed by Microsoft in 2017 [2]. In terms of CPU execution time and accuracy, LightGBM outperforms other gradient boosting methods significantly [3], [4]. Through a comparison of three gradient boosting methods, LightGBM proves to be much faster and more accurate when using the same amount of time for hyper-parameter optimization. By implementing the LightGBM technique in bank marketing data analysis, it can help banks gain deeper insights into customer preferences and behavior, enabling them to design more effective and targeted marketing campaigns. Additionally, a research study shows that LightGBM algorithm consistently outperforms XGBoost and SGB algorithms in terms of computational speed and memory usage, making LightGBM highly efficient for practical use [3], [5]. Therefore, this research holds an interesting and crucial background to be followed in the efforts to optimize marketing campaign strategies in the banking industry. The research conducted aims to perform classification using LightGBM and test the performance of the generated model using a confusion matrix and overfitting checks as a stage of implementing the LightGBM algorithm. The data to be analyzed and visualized originates from a Portuguese retail bank, collected from a previous study spanning from May 2008 to June 2013 [6]. The goal of the data analysis

resulting from this research is to create a predictive model obtained through the LightGBM analysis. The predictive model generated will be visualized through a confusion matrix, and overfitting of the model will also be checked.

2. METHOD

2.1. Research Methodology

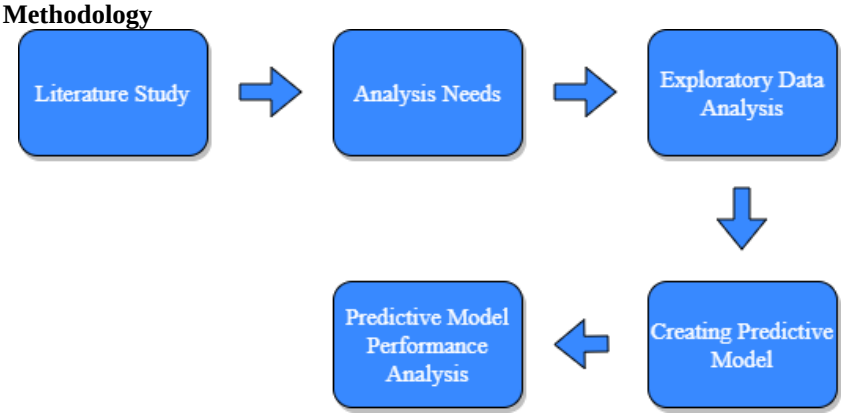

Figure 1. Research Methodology

The conducted research will follow the flow and stages as depicted in the above figure, starting with conducting a literature study to gather data, reference materials, read and record information, as well as manage research materials. Additionally, the literature study will search for relevant theories related to the existing issues [7]. The next stage involves the analysis of requirements for all data that needs to be used to achieve the research objectives. In the exploratory data analysis stage, data exploration and analysis will be performed by seeking correlations between various features available in the data and the deposits made in the bank, as well as the distribution of the data. To facilitate the exploration and analysis process, visualization is essential. This is done to ease the identification of patterns and trends in the data, which will be used to create a machine-learning model. It also involves label encoding using Label Encoder and dividing the data into train and test sets [8]. In the predictive model creation stage, hyperparameter tuning will be conducted to determine the best parameters using Grid Search and use them to create the predictive model. The final stage involves analyzing the performance of the predictive model using a confusion matrix and checking for overfitting.

2.2. Previous Research

The previous research used as a reference in this study is literature titled "LightGBM: A Highly Efficient Gradient Boosting Decision Tree", which aims to showcase the advantages of LightGBM over other machine learning algorithms, such as Gradient Boosting Decision Tree (GDBT) algorithms like XGBoost and SGB. The results of the experiments conducted in this literature are highly consistent with the theory and demonstrate that the LightGBM algorithm significantly outperforms XGBoost and SGB in terms of computational speed and memory usage, making LightGBM highly efficient for practical use [5].

In another study comparing the XGBoost and LightGBM algorithms in handling class imbalance, which is a challenge affecting the performance of many classification algorithms, causing poor and suboptimal classification, it was concluded that LightGBM achieved an overall score of 80.41%, outperforming XGBoost, which scored 74.64% [9].

LightGBM offered the best performance in terms of balanced accuracy for both internal validation and external test sets, compared to four widely used algorithms, DNN, RF, SVC, and XGBoost. In addition to the excellent predictive performance, LightGBM is also faster and more scalable in model development [10] .

2.3. Exploratory Data Analysis (EDA)

Exploratory data analysis (EDA) is an approach to examine what can be communicated by data to us without going through formal modeling or hypothesis testing stages [11]. EDA is the initial step in understanding data by performing visualizations using available analytical tools in data processing software [12], he language used in the research that involves EDA utilizes Python. The community around the tools and libraries available in Python makes it highly appealing for use in the fields of data science, machine learning, and scientific computing [13].

In the process of labeling several categorical features within the dataset, the values of categories are transformed into integers to allow processing by machine learning algorithms that require numeric values as input. The label encoder technique is chosen for labeling in this research because it is highly effective in converting categorical features into numerical values [14]. Moreover, the label encoder is more flexible than other label encoding techniques. By using a label encoder, the author can determine which attributes need to

be encoded into new attributes and which attributes do not need to be encoded into new attributes when performing encoding on the existing attributes.

2.5. LightGBM

LightGBM is a GBDT (Gradient Boosting Decision Tree) algorithm typically used for classification, ranking, and regression tasks. Additionally, LightGBM supports more efficient parallel model training [15], [16] [17]. GBDT algorithms have been quite successful in winning machine learning and data mining competitions [18], [19]. While some decision tree algorithms use level-wise tree growth in constructing decision trees, LightGBM adopts leaf-wise tree growth, resulting in more complex but effective decision trees with higher accuracy. Data splitting is based on a new sampling technique called Gradient-Based One Side Sampling [20].

2.6. Performance Analysis

Performance analysis is conducted using a confusion matrix. A confusion matrix is a table used to evaluate the performance of a model [21]–[24]. It is essential for performance analysis as it provides a table-based assessment to examine the values of each class [25].

$$Precision = \frac{TP}{TP+FP} \quad (1)$$

$$Recall = \frac{TP}{TP+FN} \quad (2)$$

$$F1 - Score = \frac{2 \times Precision \times Recall}{Precision+Recall} \quad (3)$$

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (4)$$

3. RESULTS AND DISCUSSION

3.1. Analysis Needs

The required data originates from a previous study, specifically data obtained from a Portuguese retail bank, comprising information about bank clients [6].

3.2. Exploratory Data Analysis

In this stage, the analysis begins with data import, which involves a file in Comma Separated Values (CSV) format containing 11,162 data entries with 17 attributes. Upon examining the data, no null values are observed.

```
data.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	deposit
0	59	admin.	married	secondary	no	2343	yes	no	unknown	5	may	1042	1	-1	0	unknown	yes
1	56	admin.	married	secondary	no	45	no	no	unknown	5	may	1467	1	-1	0	unknown	yes
2	41	technician	married	secondary	no	1270	yes	no	unknown	5	may	1389	1	-1	0	unknown	yes
3	55	services	married	secondary	no	2476	yes	no	unknown	5	may	579	1	-1	0	unknown	yes
4	54	admin.	married	tertiary	no	184	no	no	unknown	5	may	673	2	-1	0	unknown	yes

Figure 2. Data sample

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11162 entries, 0 to 11161
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   age        11162 non-null  int64
 1   job        11162 non-null  object
 2   marital    11162 non-null  object
 3   education  11162 non-null  object
 4   default    11162 non-null  object
 5   balance    11162 non-null  int64
 6   housing    11162 non-null  object
 7   loan       11162 non-null  object
 8   contact    11162 non-null  object
 9   day        11162 non-null  int64
 10  month      11162 non-null  object
 11  duration   11162 non-null  int64
 12  campaign   11162 non-null  int64
 13  pdays      11162 non-null  int64
 14  previous   11162 non-null  int64
 15  poutcome   11162 non-null  object
 16  deposit    11162 non-null  object
dtypes: int64(7), object(10)
memory usage: 1.4+ MB
```

Figure 3. Data information

In the dataset, it is evident that some attributes still contain non-numeric data types. These attributes with non-numeric data are referred to as object data types, meaning they have values that are not in numerical form. In the process of data processing using machine learning algorithms, the data must be in numerical form. Therefore, data with non-numeric values need to be encoded. Attribute encoding will be performed using Label Encoder. By using Label Encoder, we can select specific attributes to be encoded from the original attributes, without the need to encode all attributes in the dataset. This allows us to encode only the necessary attributes, providing more flexibility during the data processing.

```
data1.head(2)
```

	age	default	balance	housing	loan	day	duration	campaign	pdays	previous	...	month_jun	month_mar	month_may	month_nov	month_oct	month_sep	poutcome_failure	poutcome_other	poutcome_success	poutcome_unknown
0	59	0	2343	1	0	5	1042	1	-1	0	...	0	0	1	0	0	0	0	0	0	1
1	56	0	45	0	0	5	1467	1	-1	0	...	0	0	1	0	0	0	0	0	0	1

2 rows × 49 columns

Figure 4 Data sample after encoded

The attributes that have been transformed from categorical to numerical values include job divided into several new attributes such as job_admin, job_blue-collar, job_entrepreneur, job_housemaid, job_management, job_retired, job_self-employed, job_service, job_student, job_technician, job_unemployed, and job_unknown. Marital: Divided into marital_divorced, marital_married, and marital_single. Education divided into education_primary, education_secondary, education_tertiary, and education_unknown. Contact divided into contact_cellular, contact_telephone, and contact_unknown. Month divided into month_jan, month_feb, month_mar, month_apr, month_may, month_jun, month_jul, month_aug, month_sep, month_oct, month_nov, and month_dec. Poutcome divided into poutcome_failure, poutcome_other, poutcome_success, and poutcome_unknown. Furthermore, some other attributes with binary values "yes" and "no" such as default, housing, loan, and deposit have been encoded as "1" for yes and "0" for no.

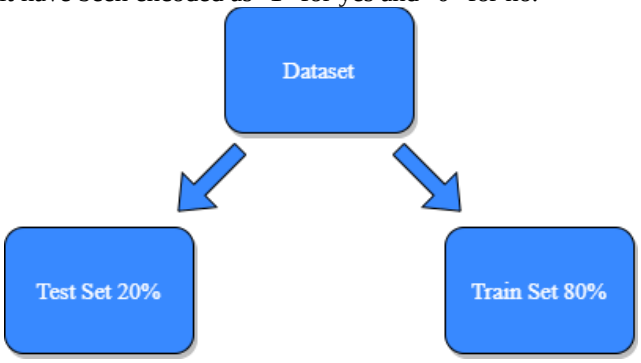

Figure 5. Dataset split

After label encoding is done, the next step is to split the dataset into training and testing sets. The creation of the training set and testing set will be divided into a ratio of 80% for the training set and 20% for the testing set. In this research, when splitting the dataset, a random state value of 11 is used without considering the variations that would result from using other random state values. This is solely used to obtain consistent results each time the code is executed.

3.3. Predictive Model Creation

In creating the predictive model with the LightGBM algorithm, the first step is hyperparameter tuning. This process is performed to obtain the best parameter settings that result in the highest accuracy.

```
from sklearn.model_selection import GridSearchCV
import lightgbm as lgb

# Determine the hyperparameters to be optimized and their value ranges.
param_grid = {
    'learning_rate': [0.1, 0.01, 0.001],
    'n_estimators': [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000]
}

# Initialize the LightGBM model.
model = lgb.LGBMClassifier()

# Create GridSearchCV Object
grid_search = GridSearchCV(model, param_grid, scoring='accuracy', cv=5)

# Finding Parameter
grid_search.fit(X_train, y_train)

# Print the best parameters and the best evaluation score.
print("Best Parameters: ", grid_search.best_params_)
print("Best Score: ", grid_search.best_score_)
```

```
Best Parameters:  {'learning_rate': 0.1, 'n_estimators': 100}
Best Score:  0.8625823632924614
```

Figure 6. Source code for hyper-parameter tuning

Hyperparameter tuning is carried out by determining the values and ranges of parameters to be used. Once the best parameters are identified, the next step is to train the predictive model using the obtained parameters.

```
model = lgb.LGBMClassifier(learning_rate=grid_search.best_params_['learning_rate'],
                           n_estimators=grid_search.best_params_['n_estimators'])
model.fit(X_train, y_train)
```

```
▾ LGBMClassifier
LGBMClassifier()
```

```
y_pred= model.predict(X_test)
```

Figure 7. Source code for Model Predictive Creation

3.4. Predictive Model Performance Analysis

The performance analysis is conducted by creating a confusion matrix and checking for overfitting.

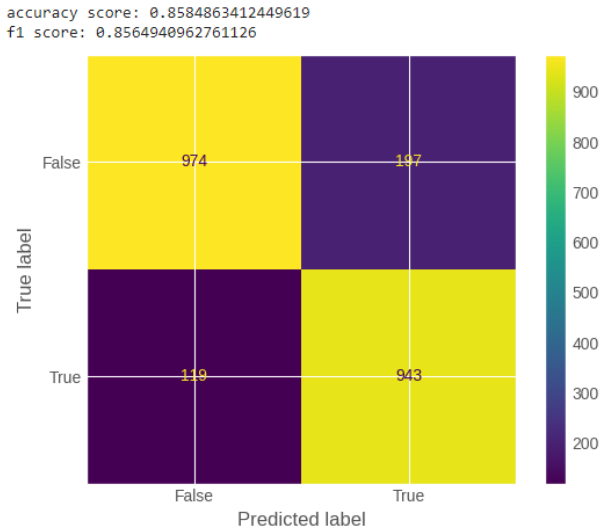

Figure 8. Confusion Matrix

Overall, the generated model performs quite well in terms of accuracy, with an accuracy score of 0.8585 and an F1 score of 0.8565. It is evident that LightGBM exhibits good performance in identifying customers not interested in the campaign (high True Negatives of 974) and customers interested in the campaign (high True Positives of 943). Next, an overfitting check will be conducted by comparing the accuracy score and F1 score between the training set and test set. This is essential to determine if there is overfitting in the predictive model created. To examine the accuracy score and F1 score of the training set and test set, you can use source code similar to the one shown in the following image.

```
# Evaluation of Accuracy Score and F1 Score on the Train Set.
y_train_pred = model.predict(X_train)
train_cm = metrics.confusion_matrix(y_train, y_train_pred)
train_accuracy = accuracy_score(y_train, y_train_pred)
train_f1 = f1_score(y_train, y_train_pred)

# Print the accuracy score and F1 score on the train set.
print("Train accuracy score:", train_accuracy)
print("Train F1 score:", train_f1)
```

```
Train accuracy score: 0.9114122522118938
Train F1 score: 0.9096928873159036
```

Figure 9 Accuration and f1-score for training set

```
# Evaluation of Accuracy Score and F1 Score on the test set
test_cm = metrics.confusion_matrix(y_test, y_pred)
test_accuracy = accuracy_score(y_test, y_pred)
test_f1 = f1_score(y_test, y_pred)

# Print the accuracy score and F1 score on the test set
print("Test accuracy score:", test_accuracy)
print("Test F1 score:", test_f1)
```

```
Test accuracy score: 0.8584863412449619
Test F1 score: 0.8564940962761126
```

Figure 10 Accuration and f1-score test set

This check is essential to ensure that the model has the ability to generalize well to new data and not be overly focused on the training data. Additionally, this check helps prevent errors in interpretation. If the model experiences overfitting, the evaluation results on the training data may appear very good, but its performance will significantly drop when applied to the testing data or real-world situations. It can also be observed that there are differences in the accuracy score and F1 score between the training set and test set. The accuracy score and F1 score on the training set are higher than the test set, but the difference is not too significant. Overall, there is no strong indication that the model is experiencing overfitting.

4. CONCLUSION AND RECOMMENDATIONS

The conclusion of this research is that the predictive model created using the LightGBM algorithm has achieved quite good performance in identifying customers interested and not interested in the campaign. The model has an accuracy of 0.8585 and an F1 score of 0.8565, indicating its capability in making predictions. LightGBM demonstrates good performance in identifying customers not interested in the campaign (high True Negatives of 974) and customers interested in the campaign (high True Positives of 943). The overfitting check results show that the accuracy score and F1 score on the training set are higher than the test set, but the difference is not too significant. Overall, there is no strong indication that the model is experiencing overfitting. It can be concluded that this model successfully generalizes well to new data and performs well in making predictions without showing signs of overfitting. There are several recommendations for further research and development, including, creating a predictive model with even better performance to reduce the number of false positives and false negatives, thus improving the effectiveness of the generated model. Implementing interpretation techniques to gain insights into the predictive model. Using more recent data for the research to keep the model up-to-date. Considering different random state values to observe variations in the results. Expanding the range of parameter values in hyperparameter tuning to obtain the best parameters and evaluation scores.

REFERENCES

[1] I. Didy, "Marketing Campaign: Kunci Keberhasilan Bisnis yang Wajib Kamu Tahu," *Glints*, 2021. [Online]. Available: https://glints.com/id/lowahttps://glints.com/id/lowongan/marketing-campaign-adalah/#.ZBf0hHZByUkongan/marketing-campaign-adalah/#.ZBf0hHZByUk.

[2] M. Bugaj, K. Wrobel, and J. Iwaniec, "Model Explainability using SHAP Values for LightGBM Predictions," in *2021 IEEE XVIIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH)*, 2021, pp. 102–106.

[3] E. Al Daoud, "Comparison between XGBoost, LightGBM and CatBoost Using a Home Credit Dataset," vol. 13, no. 1, pp. 6–10, 2019.

[4] H. Alshari, A. Y. Saleh, and A. Odabaş, "Comparison of Gradient Boosting Decision Tree Algorithms for CPU Performance CPU Performansı için Gradyan Artırıcı Karar Ağacı Algoritmalarının Karşılaştırılması," *Erciyes Univ. J. Instutue Sci. Technol.*, vol. 37, no. 1, pp. 157–168, 2021.

[5] G. Ke *et al.*, "LightGBM: A highly efficient gradient boosting decision tree," *Adv. Neural Inf. Process. Syst.*, vol. 2017-December, no. Nips, pp. 3147–3155, 2017.

[6] S. Moro, P. Cortez, and P. Rita, "A data-driven approach to predict the success of bank telemarketing," *Decis. Support Syst.*, vol. 62, pp. 22–31, Jun. 2014.

[7] D. Pilendia, "Pemanfaatan Adobe Flash sebagai Dasar Pengembangan Bahan Ajar Fisika : Studi Literatur," *J. Tunas Pendidik.*, vol. 2, no. 2, 2020.

[8] R. Khan, S. Akbar, and T. A. Zahed, "Flight Delay Prediction Based on Gradient Boosting Ensemble Techniques," in *2022 16th International Conference on Open Source Systems and Technologies (ICOSST)*, 2022, pp. 1–5.

[9] P. Septiana Rizky, R. Haiban Hirzi, and U. Hidayaturrohman, "Perbandingan Metode LightGBM dan XGBoost dalam Menangani Data dengan Kelas Tidak Seimbang," *J Stat. J. Ilm. Teor. dan Apl. Stat.*, vol. 15, no. 2, pp. 228–236, Dec. 2022.

[10] J. Zhang, D. Mucs, U. Norinder, and F. Svensson, "LightGBM: An Effective and Scalable Algorithm for Prediction of Chemical Toxicity–Application to the Tox21 and Mutagenicity Data Sets," *J. Chem. Inf. Model.*, vol. 59, no. 10, pp. 4150–4158, Oct. 2019.

[11] K. Sahoo*, A. K. Samal, J. Pramanik, and S. K. Pani, "Exploratory Data Analysis using Python," *Int. J. Innov. Technol. Explor. Eng.*, vol. 8, no. 12, pp. 4727–4735, Oct. 2019.

[12] J. DSouza and S. Velan S., "Using Exploratory Data Analysis for Generating Inferences on the Correlation of COVID-19 cases," in *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 2020, pp. 1–6.

[13] S. Raschka, J. Patterson, and C. Nolet, "Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence," *Information*, vol. 11, no. 4, p. 193, Apr. 2020.

[14] P. Vadapalli, "Label Encoder vs One Hot Encoder in Machine Learning [2023]," *upgrad*, 2022. [Online]. Available: https://www.upgrad.com/blog/label-encoder-vs-one-hot-encoder/.

[15] M. Wang, L. Yue, X. Yang, X. Wang, Y. Han, and B. Yu, "Fertility-LightGBM: A fertility-related protein prediction model by multi-information fusion and light gradient boosting machine," *Biomed. Signal Process. Control*, vol. 68, p. 102630, Jul. 2021.

[16] Y. Xu, W. Cai, L. Wang, and T. Xie, "Intelligent Diagnosis of Rolling Bearing Fault Based on Improved Convolutional Neural Network and LightGBM," *Shock Vib.*, vol. 2021, pp. 1–8, Dec. 2021.

[17] E.-A. MINASTIREANU and G. MESNITA, "Light GBM Machine Learning Algorithm to Online

Click Fraud Detection," *J. Inf. Assur. Cybersecurity*, pp. 1–12, Apr. 2019.
[18] Q. Li, Z. Wen, and B. He, "Practical Federated Gradient Boosting Decision Trees," *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 04, pp. 4642–4649, Apr. 2020.
[19] G. Jiao and H. Xu, "Analysis and Comparison of Forecasting Algorithms for Telecom Customer Churn," *J. Phys. Conf. Ser.*, vol. 1881, no. 3, p. 032061, Apr. 2021.
[20] Q.-T. Bui *et al.*, "Gradient Boosting Machine and Object-Based CNN for Land Cover Classification," *Remote Sens.*, vol. 13, no. 14, p. 2709, Jul. 2021.
[21] H. Yun, "Prediction model of algal blooms using logistic regression and confusion matrix," *Int. J. Electr. Comput. Eng.*, vol. 11, no. 3, p. 2407, Jun. 2021.
[22] N. D. Marom, L. Rokach, and A. Shmilovici, "Using the confusion matrix for improving ensemble classifiers," in *2010 IEEE 26-th Convention of Electrical and Electronics Engineers in Israel*, 2010, pp. 000555–000559.
[23] D. Normawati and S. A. Prayogi, "Implementasi Naïve Bayes Classifier Dan Confusion Matrix Pada Analisis Sentimen Berbasis Teks Pada Twitter," *J. Sains Komput. Inform. (J-SAKTI*, vol. 5, no. 2, pp. 697–711, 2021.
[24] N. Hadianto, H. B. Novitasari, and A. Rahmawati, "Klasifikasi Peminjaman Nasabah Bank Menggunakan Metode Neural Network," *J. Pilar Nusa Mandiri*, vol. 15, no. 2, pp. 163–170, Sep. 2019.
[25] D. S. Y. Kartika, D. Herumurti, and A. Yuniarti, "Butterfly Image Classification Using Color Quantization Method on HSV Color Space and Local Binary Pattern," *IPTEK J. Proc. Ser.*, vol. 4, no. 1, p. 78, Jan. 2018.