Predicting Term Deposit Offer Subscription in Bank Marketing Campaigns Using Machine Learning Algorithms

Blessing Oriaguaman

*College of Engineering, Environment and Science*

Coventry University

Coventry, England

[oriaguamab@uni.coventry.ac.uk](mailto:oriaguamab@uni.coventry.ac.uk)

[[1]](#footnote-2)

*Abstract*— This paper investigates customer responses to term deposit offers in a Portuguese banking marketing campaign. Five machine learning algorithms (K Nearest Neighbor, Decision Tree, Random Forest, Logistic Regression, Gradient Boosted regression tree) were implemented and compared. The dataset underwent preprocessing to address issues like class imbalance, categorical feature encoding and feature reduction to enhance predictive accuracy. Performance evaluation of each algorithm's accuracy on training data and effectiveness in predicting term deposit subscription was conducted through analysis and visualization.

*Keywords*— *Term deposit; K-Nearest Neighbor (KNN); Decision Tree; Random Forest; Logistic Regression, Gradient Boosting regression; Data preprocessing; Class imbalance; Feature extraction; categorical feature encoding*

# introduction

In the banking industry, understanding customer behavior is paramount for devising effective marketing strategies. One crucial aspect of bank marketing campaigns is the promotion of term deposit offers to customers. Identifying customers who are likely to subscribe to term deposits enables banks to tailor their marketing efforts effectively, thereby optimizing resources and maximizing subscription rates.

Recent advancements in machine learning algorithms offer promising avenues for predicting customer responses in bank marketing campaigns. Algorithms such as K-Nearest Neighbor (KNN), Decision Tree, Random Forest, and Gradient Boosting have emerged as popular choices for predictive modeling in various domains [11]

The aim of this research is to investigate the predictability of customer responses to term deposit offers, employing a comparative analysis of machine learning algorithms. Its objective is implementing and evaluating the performance of KNN, Decision Tree, Random Forest, Logistic Regression and Gradient Boosted Regression tree algorithms in predicting term deposit subscription.

This paper addresses three main research questions: Firstly, which machine learning algorithms will better predict term deposit subscriptions? Secondly, which class balancing technique will yield the best result? And finally, to what extent does machine learning enhance marketing strategies in the banking industry?

In conducting this research, various considerations were addressed. Ethically, customer data was anonymized and handled with utmost confidentiality. Legally, compliance with GDPR, data protection laws, and banking regulations was ensured. Professionally, integrity, impartiality, and objectivity were maintained in analysis and reporting to avoid bias. Socially, the potential impact of predictive modeling on individuals and society, including unintended consequences, was carefully considered.

Previous research was conducted by Zaki et al. [18] using the same dataset used in this research to predict bank deposit subscription using stochastic Gradient Descent (SGD), KNN, Logistic Regression, Gaussian Naïve Bayes (GNB), Decision Tree and Random Forest Algorithms. The Results from their research showed that Random Forest Classifier had the highest accuracy score of 87.5% and KNN had the least of 82.5%. The researchers concluded that Random Forest is the best algorithm for predicting bank term deposit subscription.

# The data set

The dataset utilized in this study was sourced from the UCI Machine Learning Repository, comprising 45,211 instances, and 17 features. Each instance corresponds to phone calls made to clients to determine the client's inclination towards subscribing to the bank term deposit, categorized as either 'yes' or 'no'. Table 1 presents a summarized overview of the dataset, showing the attributes, their types [14].

# table 1. dataset attributes and types

A table of text with black text

Description automatically generated

# experimental setup

The data originally collected from the UCI Machine Learning Repository exhibits class imbalance and when training was done, KNN, Decision Tree, Random Forest, and Logistic Regression achieved 88% Accuracy, Random Forest had 85% on the imbalanced data. The dataset also includes categorical features, necessitating categorical data encoding. This section addresses data preprocessing tasks such as the management of class imbalance issues, categorical encoding, feature scaling, dimensionality and feature reduction.

1. Class Imbalance

The analysis revealed a significant class imbalance: 36,548 responses indicated "no" to subscribing to term deposits, while only 4,640 responded "yes" as shown in figure 1. Imbalanced data challenges machine learning, as standard algorithms favor the majority class, risking misclassification of minority instances and impacting performance[5][8].

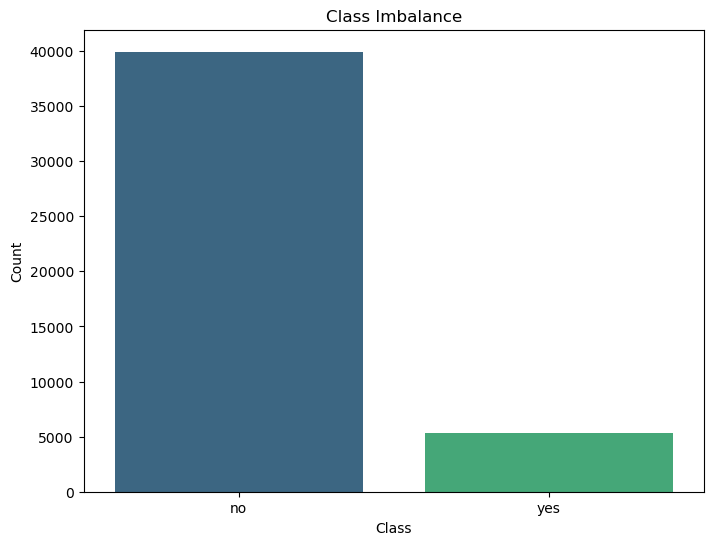


Fig. 1. Class imbalance

To address the curse of data imbalance three techniques was evaluated.

1. Synthetic Minority Over-Sampling Technique (SMOTE)

This technique is utilized to generate synthetic instances for the minority class. It functions by randomly selecting a sample from the minority class and finding its K-nearest neighbors as illustrated in Figure 2. Then, an artificial sample is inserted between the selected sample and its neighbors [17]. It resulted in creating a new dataset of 79,844 samples.

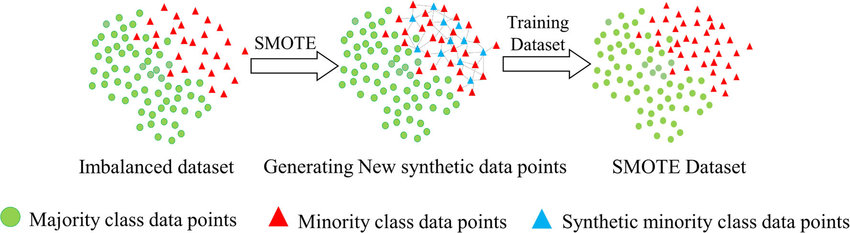


Fig. 2. Illustration of the SMOTE technique

1. Random Over Sampling

This method involves randomly duplicating samples from the minority class until the class distribution is balanced. It may lead to overfitting due to the replication of minority class samples [7]. Balancing the dataset using Random Over-sampling duplicated the 'Yes' class thus creating a new dataset of 79,844 samples.

1. Random Under Sampling

This technique entails randomly eliminating samples from the majority class to achieve a balanced class distribution. It runs the risk of discarding valuable information and resulting in data loss [6]. This approach will decrease the 'No' class resulting in a sample size of 10,578.

In this paper, the three techniques were applied within scikit-learn. Refer to the *Appendix* for the code utilized to implement these techniques.

1. Categorical Feature Encoding

Most machine learning models accept numerical variables, it becomes imperative to preprocess categorical variables. This involves transforming these categorical variables into numerical representations to ensure the model can comprehend and extract meaningful information [1]. In this paper, ten categorical features were encoded using the *OneHotEncoder* method available in the *Scikit-learn* library.

1. Feature Scaling (Standardization)

Datasets for training machine learning models may have unpredictable values with varying scales, leading to disparities in comparisons. Techniques like StandardScaler() from Scikit-learn were used in this study to address these issues. StandardScaler() adjusts each feature by subtracting its mean and dividing by its standard deviation, resulting in data with mean 0 and standard deviation 1 [3][9][13]. The formula is presented below as formula (1).

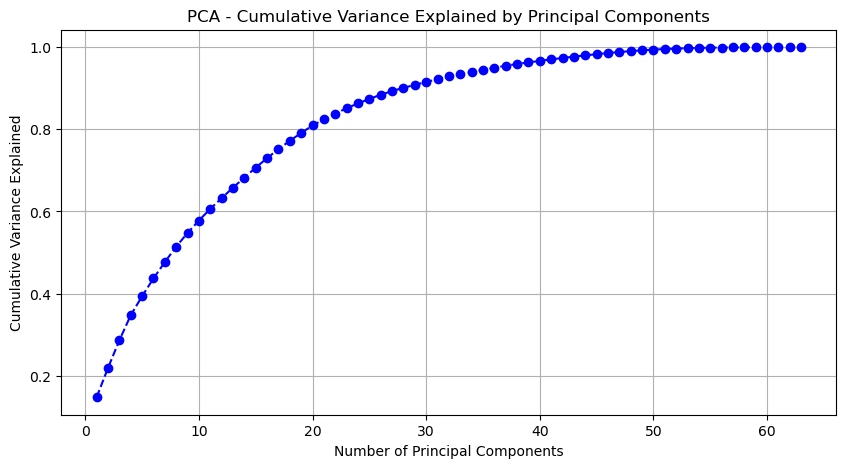
A mathematical equation with black text

Description automatically generated

(1)

1. Feature Reduction and Extraction

Feature reduction techniques aim to address the curse of dimensionality by reducing the number of features in the dataset while minimizing information loss [2]. Awan (2023) listed four methods for dimensionality reduction: Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), t-Distributed Stochastic Neighbor Embedding (t-SNE), and Autoencoders. In this study, PCA was employed for feature reduction and extraction.

A graph of a number of blue and white lines

Description automatically generatedFig. 3. PCA Individual Component Variance

Fig. 4. PCA Cumulative Variance

Analyzing PCA results in Figure 3 and 4 reveals that 36 components explain over 90% of cumulative variance. Hence, only these first 36 features were retained for further analysis, capturing the majority (90%) of dataset variance.

# machine learning classification techniques

1. K-Nearest Neighbor (KNN)

The k-NN algorithm is considered one of the most straightforward machine learning algorithms. Constructing the model involves solely storing the training dataset. When predicting a new data point, the algorithm identifies the nearest neighbors within the training dataset to make its prediction [10].

1. Logistic Regression

It estimates the probability of an outcome for a categorical variable and produces values between 0 and 1. Primarily employed in classification tasks, it determines the likelihood of different categories [15].

1. Decision Tree

A decision tree constructs a hierarchical structure of if/else questions based on the features of the dataset, ultimately leading to a prediction or decision at the leaf nodes [10].

1. Random Forest

This algorithm aims to mitigate overfitting and enhance generalization performance by aggregating predictions from different decision trees, which individually may be prone to overfitting on specific subsets of the data [10].

1. Gradient Boosted Regression Trees

This method combines several decision trees to form a more robust model. Despite its name suggesting a focus on regression, it's versatile and can effectively handle classification tasks as well [10].

# application of techniques and results

The models were trained on datasets split into 70% for training and 30% for testing. Model generalization was assessed, and evaluation metrics obtained for all algorithms. Cross-validation with 5 folds was employed to prevent overfitting. Implementation was conducted on a Dell G5 15 Laptop with Intel(R)\_Core(TM) i7-9750H CPU @ 2.60GHz processor and 16GB RAM.

1. K-Nearest Neighbor

KNN was applied utilizing the KNeighborsClassifier class from the scikit-learn library on the datasets. The outcome is presented with Confusion Matrices and ROC curve in Figure 5 and 6 and detailed in TABLE 2.

A screenshot of a graph

Description automatically generated

Fig. 5. Confusion Matrices for K-Nearest Classifier

A collage of graphs

Description automatically generatedA collage of graphs

Description automatically generated

Fig. 6. ROC Curves for K-Nearest Classifier

# table 2 – knn result summary

A table with numbers and text

Description automatically generated

The SMOTE-augmented dataset achieved the highest accuracy of 92% in the K-Nearest Neighbors model, with consistent accuracy observed regardless of cross-validation.

1. Logistic Regression

Logistic regression was performed on datasets balanced using the three distinct techniques, employing the LogisticRegression module from the scikit-learn library. The outcomes are visualized with Confusion Matrice and ROC Curve in Fig 7 and 8 and tabulated in Table 3.

A screenshot of a graph

Description automatically generated

Fig. 7. Confusion Matrices for Logistic Regression

Fig. 8. ROC Curves for Logistic Regression

# table 3 – logistic regression result summary

A table with numbers and text

Description automatically generated

The results suggest that SMOTE-generated datasets outperform both over-sampling and under-sampling datasets, achieving over 93.7% accuracy without cross-validation, comparable to KNN algorithm performance.

1. Decision Tree

A collage of different colored squares

Description automatically generatedDecision tree analysis was performed on the three distinct balanced datasets, employing the DecisionTreeClassifier module from the scikit-learn library using a max\_depth of 7. The results are depicted in Fig 9 and 10 and summarized in Table 4.

Fig. 9. Confusion Matrices for Decision Tree

A collage of graphs

Description automatically generated

Fig. 10. ROC Curves for Decision Tree

# table 4 – decision tree result summary

A table with numbers and text

Description automatically generated

Compared to KNN and Logistic Regression, the Decision Tree model showed slightly lower performance on the SMOTE dataset, with 89% accuracy with cross-validation and 91% without.

1. Random Forest

The Random Forest algorithm was implemented using the RandomForestClassifier module from the scikit-learn library. The outcomes are visualized in Fig 11 and 12 detailed in Table 5.

A screenshot of a graph

Description automatically generated

Fig. 11. Confusion Matrices for Random Forest

A collage of graphs

Description automatically generated

Fig. 12. ROC Curves for Random Forest

# table 5 – random forest result summary

A table with numbers and text

Description automatically generated

The SMOTE dataset in the Random Forest model outperformed other balanced datasets, achieving 90% accuracy with cross-validation and 92% without. The Over-sampling dataset achieved 96% accuracy with 5-fold cross-validation, but its performance dropped significantly without cross-validation, reaching only 69% accuracy.

1. Gradient Boosted Regression Tree

A screenshot of a graph

Description automatically generatedThe Gradient Boosted Regression Tree was instantiated using the GradientBoostingClassifier from the scikit-learn library with a maximum depth of 4. The results are detailed in Figure 13 and 14 and summarized in Table 6.

Fig. 13. Confusion Matrices for Gradient Boosted Tree

A screenshot of a graph

Description automatically generated

Fig. 14. ROC Curves for Gradient Boosted Tree

# table 6 – gradient boosted tree result summary

A table with numbers and symbols

Description automatically generated with medium confidence

The SMOTE dataset yielded the most favorable performance in the Gradient Boosted Regression algorithm, achieving an accuracy of 91% with 5-fold cross-validation and 93% without cross-validation.

# conclusion

The SMOTE technique demonstrated superior effectiveness in addressing class imbalance across all models. The aggregated results for the SMOTE class balancing technique are presented in Table 7.

# table 7 – smote result summary A table of numbers and a number of trees Description automatically generated with medium confidence

Logistic Regression emerged as the most effective model, achieving AUC score of 0.98 and 93.7% accuracy without cross-validation, GBRT closely followed with same AUC and accuracy score. Although both have the same metrics, in terms of computational speed, Logistic Regression outperformed GBRT with an execution time of 0.1027seconds, while GBRT exhibited the slowest with 214.5077seconds.

Random Forest, KNN and Decision Tree performed relatively lower compared to Logistic Regression and GBRT. The consistently high AUC score for all models implies that they effectively distinguished between both classes indicating strong predictive power.

The result produced in this research did better than previous research done by Zaki et al, (2024) which stated that Random Forest Classifier had the highest accuracy score of 87.5% and KNN had the least of 82.5%.

# future research

Future research may focus on investigating the application of deep learning architectures, such as neural networks. This may unveil hidden patterns in complex banking datasets, offering deeper insights into customer behavior.

# references

1. Arashnic. (2021, December 30). An overview of categorical encoding methods. Kaggle. <https://www.kaggle.com/code/arashnic/an-overview-of-categorical-encoding-methods>
2. Awan, A. A. (2023, September 13). The Curse of dimensionality in Machine Learning: challenges, impacts, and solutions. <https://www.datacamp.com/blog/curse-of-dimensionality-machine-learning>
3. Bhandari, A. (2024a, January 4). Feature Scaling: Engineering, normalization, and Standardization (Updated 2024). Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/>
4. Bhandari, A. (2024b, January 8). Guide to AUC ROC Curve in Machine Learning : What is specificity? Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/>
5. GeeksforGeeks. (2021, December 19). How to handle imbalanced classes in Machine Learning. <https://www.geeksforgeeks.org/how-to-handle-imbalanced-classes-in-machine-learning/>
6. GfG. (2021, December 19). How to handle imbalanced classes in Machine Learning. GeeksforGeeks. <https://www.geeksforgeeks.org/how-to-handle-imbalanced-classes-in-machine-learning/>
7. He, H., & Ma, Y. (2013). Imbalanced learning: Foundations, Algorithms, and Applications. John Wiley & Sons.
8. Hong Nguyen, G., Bouzerdoum, A., & Lam Phung, S. (2009). Learning Pattern Classification Tasks with imbalanced data sets. pp-193 [Google book]. IntechOpen. <https://doi.org/10.5772/149>
9. Jaadi, Z. (2023, August 4). When and why to standardize your data. Built In. <https://builtin.com/data-science/when-and-why-standardize-your-data>
10. Müller, A. C., & Guido, S. (2016). Introduction to Machine Learning with Python: A Guide for Data Scientists. “O’Reilly Media, Inc.”
11. Sarker, I. H. (2021). Machine learning: algorithms, Real-World applications and research directions. SN Computer Science, 2(3). <https://doi.org/10.1007/s42979-021-00592-x>
12. sklearn.metrics.confusion\_matrix. (n.d.). Scikit-learn. <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html>
13. Team, T. A., & Team, T. A. (2022, July 5). Which Feature Scaling Technique To Use- Standardization vs. Towards AI. <https://towardsai.net/p/l/which-feature-scaling-technique-to-use-standardization-vs-normalization>
14. UCI Machine Learning Repository. (n.d.). <https://archive.ics.uci.edu/dataset/222/bank+marketing>
15. Yadav, A. L., Soni, K. K., & Khare, S. (2023). Heart Diseases Prediction using Machine Learning. 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT). <https://doi.org/10.1109/icccnt56998.2023.10306469>
16. Zaki, A., Khodadadi, N., Lim, W. H., & Towfek, S. K. (2024). Predictive analytics and machine learning in direct marketing for anticipating bank term deposit subscriptions. American Journal and Operations Research (AJBOR), 11(1), 79–88. <https://doi.org/10.54216/ajbor.110110>
17. Zhu, T., Lin, Y., & Liu, Y. (2017). Synthetic minority oversampling technique for multiclass imbalance problems. Pattern Recognition, 72, 327–340. <https://doi.org/10.1016/j.patcog.2017.07>.

# appendix

The Python programming files are accessible via this GitHub link: <https://github.com/Oriaguaman/bank_marketting>

The datasets was downloaded from UCI on this link: <https://archive.ics.uci.edu/dataset/222/bank+marketing>

1. [↑](#footnote-ref-2)