Predictive Modeling for Customer Response in ABC Supermarket's Gold Membership Campaign

Machine Learning (CSELEC2C) - Laboratory Activity # 2

Dela Paz, Angelo Daniel A. ; Sta. Cruz, Levin Jacob

I. Introduction

1.1 Background of the Study

In the dynamic retail landscape, customer engagement strategies drive sales and foster brand loyalty. In its pursuit of enhancing year-end sales, ABC Supermarket is gearing up to launch a compelling Gold Membership offer tailored to existing customers. This exclusive promotion, priced at a discounted rate of $499 compared to its regular price of $999, promises a 20% discount on all purchases, presenting a lucrative opportunity for loyal patrons.

ABC Supermarket aims to minimize campaign costs for its Gold Membership promotion by targeting customers most likely to respond positively. Using the previous year's campaign data, the supermarket is developing a predictive model to classify customers based on their likelihood to embrace the Gold Membership. This model will allow ABC Supermarket to approach the forthcoming phone call campaign more strategically and cost-effectively.

This literature delves into the intricacies of building and fine-tuning a predictive model for customer response using several machine learning algorithms: Logistic Regression, Naive Bayes, Decision Trees, Support Vector Machine, and K-Nearest Neighbors. Their performance will be compared through each algorithm's accuracy, precision, recall, F1-score, and area under the curve (AUC). By comparing these metrics, we can identify the algorithm that provides the most optimal performance for predicting customer response to the Gold Membership offer.

1.2 Scope and Limitations

This project aims to explore and compare the performance of various machine learning classification models for predicting customer response to a Gold Membership offer. The models under consideration include Logistic Regression, Naive Bayes, Decision Trees, Support Vector Machine, and K-Nearest Neighbors. The project aims to identify the optimal algorithm for this specific task, considering factors such as accuracy, precision, recall, F1-score, and area under the curve (AUC). It is important to note that ensemble learning techniques were not employed in this study due to the time constraint, and each algorithm was used independently.

The project will involve data preprocessing, model training, evaluation, and comparison. The dataset provided by the instructor will be utilized for this purpose. The project will be carried out using Jupyter Notebook, Python, and various machine-learning libraries in Python.

It is worth noting that the project used two different devices, which led to varying processing times. Two devices were used in the experimentation and testing: a MacBook Pro M1 Pro with 16GB RAM and an 8-core processor and a 2020 Lenovo Legion 5 with an AMD Ryzen 7 5800H CPU and an NVIDIA RTX 3060 GPU.

1.3 Exploratory Data Analysis

Several obstacles were encountered during the initial data exploration that necessitated attention during the data preprocessing stage. Certain features (i.e., date of birth) presented challenges in their current state, posing difficulties for the analysis. Additionally, we observed a significant imbalance in the response feature (the dependent variable). This imbalance could skew the results of our predictive model. Classification algorithms are biased towards the majority class.

This can lead to poor performance in identifying the minority class, even though it might be the class of interest. As shown below, a significant imbalance exists between rows that accepted and did not accept calls. This should be balanced to maximize model precision and recall metrics.

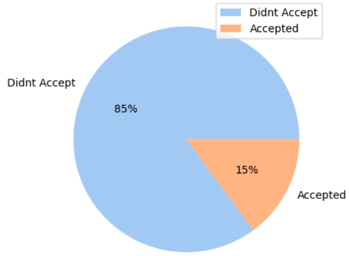


Figure 1.1 Response Percentage

Similarly, there is a significant imbalance in the response towards the complaint; only one percent of people had complaints, compared to 99% who did not.

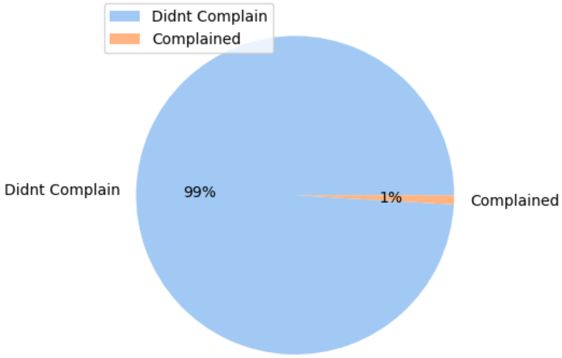


Figure 1.2 Complaint Percentage

The relationship between the number of teenagers and kids at home and the response to the offer last campaign is also explored as it may be relevant since teenagers and kids have individual and different needs from adults.

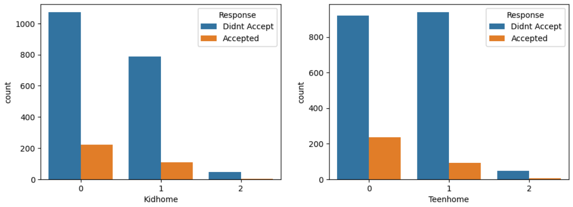


Figure 1.3 Teenhome and Kidhome Response

Lastly, the relationship between categorical features and continuous features and response is visualized to determine the distribution and the approach to handling these features.

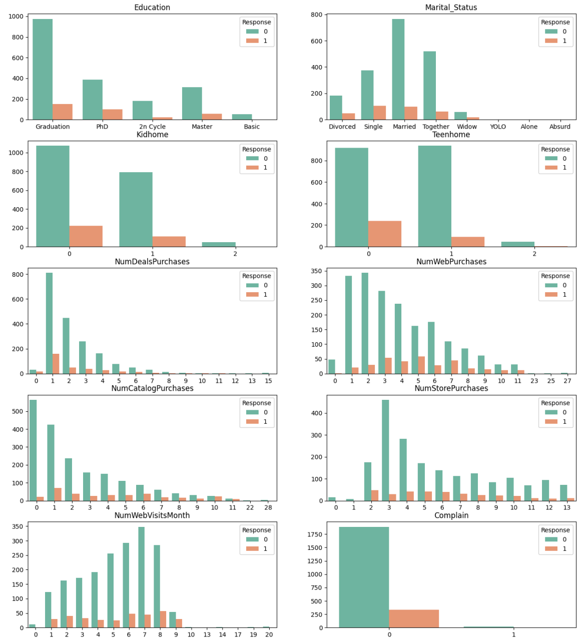


Figure 1.4 Categorical Feature Analysis

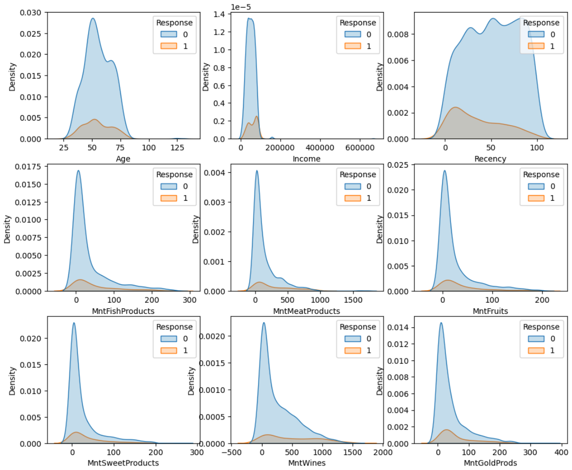


Figure 1.5 Continuous Feature Analysis

Recognizing these challenges, thorough data preprocessing is necessary to ensure our model's robustness and accuracy. Key steps, such as data cleaning, oversampling/undersampling, and standardization, were identified as essential for effectively addressing these issues. Due to this ratio, the developers decided to use sampling methods in order to handle the imbalance within the data; with the use of sampling methods, the data will be balanced, and the model will not be biased towards the majority of the responses.

II. Methodology

2.1 Libraries

The model's development will utilize multiple libraries for the prediction model:

- **Pandas**, a Library for data manipulation, will work with the .xlsx dataset, leveraging its DataFrame structure and compatibility with NumPy and Scikit-learn.This is an essential library when handling datasets in machine learning as it can read through the dataset and enables Python to manipulate the data.
- **NumPy** will provide scientific computation capabilities, particularly for array and matrix operations. It is short for Numerical Python
- **Matplotlib and Seaborn**, two libraries specializing in data visualization, will be used to illustrate the relationships between the independent and dependent variables. This helps provide graphs that allow the developers to easily view and visualize the data and determine the appropriate approach to handling the data.
- To address the imbalance classes, **RandomOverSampler, SMOTE, RandomUnderSampler, and NearMiss** were imported from the Imbalanced-learn Library. These classes provide a balanced dataset to improve the model's results. For example, in the given dataset, the distribution is 15-85 relative to the responses; these resampling techniques will balance the distribution to 50-50.
- **GridSearchCV,** a class from the scikit-learn library, specializes in handling hyperparameters, which are external configurations that influence the model's behavior. GridSearchCV explores all possible combinations of the hyperparameters automatically using cross-validation. This class aims to find the best combination of hyperparameters that yields the best performance. In this case, GridSearchCV is used on all the models to fine-tune the hyperparameters and improve the results of the models.
- Finally, five models were imported from the scikit-learn Library for testing and comparison: **Logistic Regression, Naive Bayes, Decision Tree, Support**

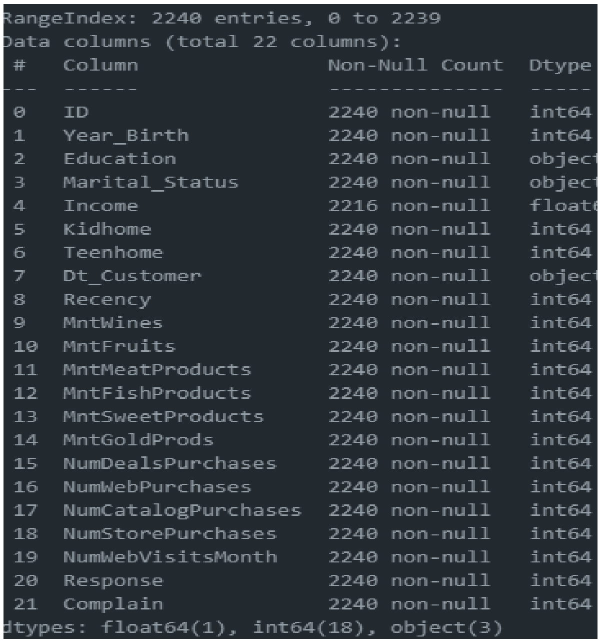
Vector Machine, and K-Nearest Neighbors.

2.2 Data Profiling and Handling

Mr. Suarez, the instructor for the course, provided the Dataset. The dataset used in the model. It contains 2,240 customers of ABC Supermarket, with 22 columns, each representing a specific feature that can be used for training the model. The specific features of the dataset are the following:

- **Response (target)**
- **Complain**
- **DtCustomer**
- **Education**
- **Marital**
- **Kidhome**
- **Teenhome**
- **Income**
- **MntFishProducts**
- **MntMeatProducts**
- **MntFruits**
- **MntSweetProducts**
- **MntWines**
- **MntGoldProds**
- **NumDealsPurchases**
- **NumCatalogPurchases**
- **NumStorePurchases**
- **NumWebPurchases**
- **NumWebVisitsMonth**
- **Recency**
- **ID**
- **Year_Birth**

During data profiling, the dataset is mostly complete except for the income feature where there are null values in 24 rows. To address this, the mean income of the dataset was inserted into the rows with null incomes. The model will utilize all dataset features except ID and DtCustomer.



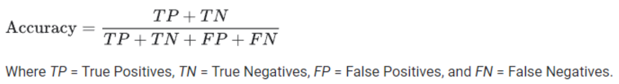
```
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 22 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID                   2240 non-null   int64
 1   Year_Birth           2240 non-null   int64
 2   Education            2240 non-null   objec
 3   Marital_Status       2240 non-null   objec
 4   Income               2216 non-null   float6
 5   Kidhome              2240 non-null   int64
 6   Teenhome             2240 non-null   int64
 7   Dt_Customer          2240 non-null   objec
 8   Recency              2240 non-null   int64
 9   MntWines             2240 non-null   int64
 10  MntFruits            2240 non-null   int64
 11  MntMeatProducts      2240 non-null   int64
 12  MntFishProducts      2240 non-null   int64
 13  MntSweetProducts     2240 non-null   int64
 14  MntGoldProds         2240 non-null   int64
 15  NumDealsPurchases    2240 non-null   int64
 16  NumWebPurchases      2240 non-null   int64
 17  NumCatalogPurchases  2240 non-null   int64
 18  NumStorePurchases    2240 non-null   int64
 19  NumWebVisitsMonth    2240 non-null   int64
 20  Response             2240 non-null   int64
 21  Complain             2240 non-null   int64
dtypes: float64(1), int64(18), object(3)
```

Figure 2.1 Dataset Information

The dataset will undergo oversampling (RandomOverSampler and SMOTE) and undersampling (RandomUnderSampler and NearMiss) techniques to address the class imbalance issue. Oversampling involves replicating examples from the minority class, while undersampling reduces the size of the majority class. This will help ensure a more balanced representation of both classes in the training data (Brownlee, 2021). Following the data preprocessing steps, the dataset will be split into training and testing sets using a 70/30 split ratio.

2.3 Metrics

Each machine learning algorithm will be evaluated by five metrics commonly used in machine learning evaluation. The metrics are accuracy, precision, recall, F1, and AUC. Informally, accuracy is the fraction of predictions the model got right. For binary classification problems such as the problem of this study, accuracy can be calculated as follows:



$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Where *TP* = True Positives, *TN* = True Negatives, *FP* = False Positives, and *FN* = False Negatives.

Figure 2.1 Accuracy in Binary Classification Models

According to Shung (2020), precision describes how precise/accurate a model is out of those predicted positively. Recall calculates how many Actual Positives a model captures by labeling it as Positive (True Positive). Lastly, F1 is a function of Precision and Recall, which seeks a balance between the two metrics. For this study, the F1 score is a better metric for evaluating the performance of machine learning models on imbalanced datasets than accuracy because accuracy is misleading when the dataset is imbalanced, as a model can achieve a high accuracy score simply by predicting the majority class for all samples. The F1 score, on the other hand, considers both precision and recall, which are more informative metrics for imbalanced datasets.

III. Experiments

This section describes the experiments used in developing the machine learning models. The results of these experiments will be discussed in the following section. The primary objective of these experiments was to evaluate the performance of various machine learning algorithms and identify the most suitable model for the given problem. The experiments assessed the models' accuracy, robustness, and computational efficiency. To ensure the reliability of the results, the experiments were conducted using a rigorous methodology, as stated above. The results from these experiments will be discussed in the following chapter.

3.1 Data Preprocessing

3.1.1 Feature Selection

As stated in the methodology section, 19 out of the 21 possible features were developed to develop the model. The ID feature, being a unique customer identifier, does not provide meaningful information to predict customer response to the Gold Membership offer. Hence, it will be excluded from the model's input variables. The DtCustomer Additionally, the date of birth feature will be converted to age by subtracting from the current year. The other features in the dataset were retained as they can provide valuable information, especially after applying data preprocessing methods.

3.1.2 Outlier Detection and Removal

Upon further examination of the continuous features, particularly Age and Income, it was discovered that there were distant outliers. Outliers are data points that significantly deviate from the rest of the dataset. To ensure the robustness and accuracy of the model, these distant outliers in these features will be removed during data preprocessing. For all models, Age, Income, MntMeatProducts, MntWines, MntSweetProducts, and MntGoldProducts were all subject to outlier removal. Notably, outlier boundaries were selected at the discretion of the proponents, and no mathematical methods were used to identify outliers. After outlier removal, the resulting dataset contained 2211 entries.

3.2 Encoding

3.2.1 Ordinal Encoding

In ordinal encoding, each unique category value is assigned an integer value. The integer values have a naturally ordered relationship, and machine learning algorithms can understand and harness this relationship (Brownlee, 2020). This encoding technique was applied to the

"Education" feature as a person's educational status has a natural order. Additionally, some feature values were converted to match similar educational statuses to reduce the number of integers used in the encoding. An example is converting "Graduation" to "Bachelors" in the Education feature.

3.2.2 One Hot Encoding

One-hot encoding is a technique used to convert categorical values into numeric values for a machine learning model (One Hot Encoding in Machine Learning, 2023). The technique was applied to the marital status feature since there is no established order in a person's marital status, thus eliminating the need for ordinal encoding.

3.3 Resampling

As discussed in the introduction, there is a severe imbalance between responding and non-responding entries. This issue can be addressed through resampling (oversampling and undersampling) and cost-sensitive learning. This study used four resampling techniques to balance the positive and negative response classes.

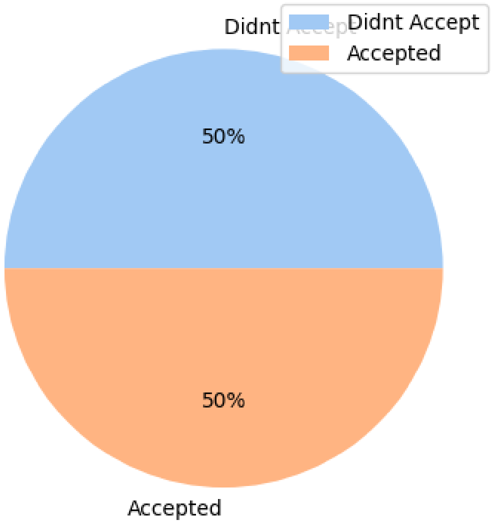


Figure 3.2 Response Percentage after Resampling

3.3.1 Oversampling

In oversampling, examples from the minority class can be chosen and added to the new "more balanced" training dataset multiple times; they are selected from the original training dataset, added to the new training dataset, and then returned or "replaced" in the original dataset, allowing them to be selected again. In addition, this technique is particularly effective in models that need a good split of the data, which includes support vector machines and decision trees (Brown, 2021). A disadvantage of using oversampling is the risk of overfitting. Since oversampling can involve replicating existing minority class examples, models are more likely to overfit the data, performing well on the training set but poorly on unseen data.

This study used RandomOverSampler and SMOTE from the Imbalanced-learn Library to oversample the positive response values. A RandomOverSampler randomly replicates existing samples from the minority class to increase its representation, In contrast, SMOTE generates new synthetic samples for the minority class by interpolating between existing minority class examples near each other in feature space. The choice of the two methods can be decided by preference of several factors, including simplicity and the severity of the imbalance. For the sake of the study, both methods were used, and their results will be compared in the results section of the literature.

3.3.2 Undersampling

In undersampling, examples from the majority class are deleted from the training dataset. This reduces the number of examples in the majority class in the transformed version of the training dataset. This process can be repeated until the desired class distribution is achieved, such as an equal number of examples for each class (Brown, 2021). However, the limitation of undersampling a dataset is the potential of deleting entries from the majority class, which may contain important information, thus reducing the model's overall performance.

This study used RandomUnderSampler and NearMiss from Imbalanced-learn as undersampling methods on the dataset. Similar to its random oversampling counterpart, RandomUnderSampler randomly selects examples from the majority class to delete. NearMiss selects majority class samples to remove based on their distances to minority class samples.

3.5 Machine Learning Algorithms

3.5.1 Logistic Regression

Logistic Regression, used to predict the probability of a categorical dependent variable, is the first machine learning algorithm used in this study Logistic regression relies on the logistic function (or sigmoid function) to convert linear combinations of features into probabilities. This function creates an "S"-shaped curve that naturally restricts the output from 0 to 1. The logistic regression model used was fitted with the following hyperparameters, and combinations of these hyperparameters were tested using GridSearchCV:

- **C:** 100, 1000
- **Penalty:** l1, l2
- **Solver:** liblenear, saga, sag
- **Class weight:** balanced, none

3.5.2 Naive Bayes

Naive Bayes is a family of supervised machine learning algorithms used for classification tasks based on Bayes' theorem, a fundamental principle in probability theory that deals with conditional probabilities. Naive Bayes assumes that features in a dataset are independent of each other, given the class label. While this assumption is not always accurate in real-world scenarios, it simplifies the model and makes it easier to apply. Similarly to the logistic regression model, GridSearchCV was used to combine and test the following hyperparameters:

- Variable Smoothing: 1e-9, 1e-8, 1e-7, 1e-6, 1e-5
- Priors: None, [0.5, 0.5], [0.6, 0.4], [0.4, 0.6]

3.5.3 Decision Trees

A non-parametric supervised learning algorithm, the decision tree can be used both in classification and regression problems. Its hierarchical tree structure comprises a root node, branches, internal nodes, and leaf nodes, providing a versatile framework for data analysis (What Is a Decision Tree | IBM, n.d.). During the development of the model, some hyperparameters like max depth, minimum samples split, and max-leaf nodes were chosen to decrease the risk of overfitting due to the use of a resampled dataset. The developed decision tree model was fitted with the following hyperparameters:

- Criterion: gini, entropy
- Max depth: 5, 10, 15
- Splitter: best, random
- Minimum samples split: 10, 15, 20
- Minimum samples leaf: 5, 10, 15
- Max leaf nodes: 10, 20, 50, 100
- Max features: sqrt, log2

3.5.4 Support Vector Machine

According to the Scikit-learn documentation website, a support vector machine (SVM) is a a set of supervised learning methods used for classification, regression and outliers detection. (*1.4. Support Vector Machines*, n.d.). An SVM operates by finding the optimal hyperplane in an N-dimensional space where it can separate the different classes being tested. It then identifies the two classes' closest points to establish the hyperplane's margin. During testing, the model used hyperparameters to establish the kernel of the SVM, which will affect the overall shape of the decision boundary. The SVM model was the slowest-performing model due to the number of hyperparameters tested. The hyperparameters used during testing are the following:

- Class Weight: balanced, none
- Gamma: 1, 0.1, 0.01, 0.001, 0.0001, 'auto'
- C: 0.1, 1, 10, 100, 1000

3.5.5 K-Nearest Neighbors

As stated in GeekforGeeks (n.d), the K-Nearest Neighbors (KNN) algorithm is a supervised machine learning algorithm that classifies new data points based on the similarity to their k most similar neighbors in the training data. KNN is a non-parametric algorithm, meaning it makes no assumptions about the data distribution. The value of k is a hyperparameter that must be selected. For the study, among the hyperparameters tested for the KNN model is the metric, which can affect the overall distance of the k nearest points, thus affecting performance. Other hyperparameters, such as N neighbors, were selected to avoid overfitting due to resampled data usage. The hyperparameters used in the study are the following:

- N Neighbors: 3, 5, 7
- Weights: uniform, distance
- Algorithm: auto, ball_tree, kd_tree, brute
- P: 1, 2
- Leaf Size: 10, 20, 30, 40
- Metric: minkowski, euclidean, manhattan

IV. Results and Analysis

In this section, the top three highest F1 metrics among tests of each machine learning algorithm were selected. F1, the balance between precision and recall, is assumed to provide the highest class-based performance in classification. The results are the mean testing and training scores, as each combination of parameters was cross-validated during modeling.

As a basis for comparison, the results of models were subject to their own preprocessing methods, which include outlier removal, encoding, resampling, etc. In addition, this section also discusses the effects of removing the various preprocessing techniques used on the dataset for each algorithm.

4.1 Logistic Regression

Table 4.1 Mean F1 Scores for Logistic Regression

Mean Testing Score	Mean Training Score
0.860521	0.877081
0.859108	0.875340
0.858137	0.876392

According to table 4.1, logistic regression was 3rd out of the five machine learning algorithms that were used, with its highest mean F1 score being 0.86. The hyperparameters tested for the highest scoring mean F1 is {'C': 100, 'class_weight': None, 'penalty': 'l1', 'solver': 'liblinear'}. It is worth noting that some combinations mentioned in the experiments section yielded null scores due to their incompatibility.

In addition, the logistic regression is one of only two models tested wherein overfitting was only slightly encountered during testing with a difference of test score of 0.003. The model where the results originated was subject to outlier removal, encoding, and NearMiss undersampling. Notably, encoding is needed for this model since string values from the categorical features can not be used for a logistic regression model. Removing outlier removal reduced the mean F1 by 0.002. This indicates that the outlier removal process did not significantly impact the model's overall performance.

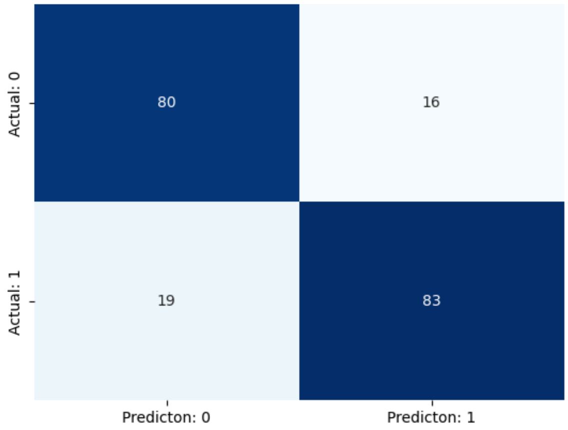


Figure 4.1 Logistic Regression Confusion Matrix

4.2 Naive Bayes

Table 4.2 Mean F1 Scores for Naive Bayes

Mean Testing Score	Mean Training Score
0.754406	0.762128
0.754406	0.765719
0.754406	0.764304

The naive Bayes model ranked last among its peers in terms of mean F1 score with 0.75. Despite the ranking, the model is still relatively reliable in classifying customers who will respond to a phone call. The NB model is also one of the two with a very small amount of overfitting. The hyperparameters tested in the top scoring F1 are {'priors': None, 'var_smoothing': 1e-06}. For the preprocessing methods, outlier removal, encoding, and NearMiss were used for the model. Keeping the outliers resulted in a decrease of 0.007, which is not significant. It is evident from the confusion matrix that there is a significantly greater number of false positives (33) than false negatives (10). This indicates a lower precision value for the model.

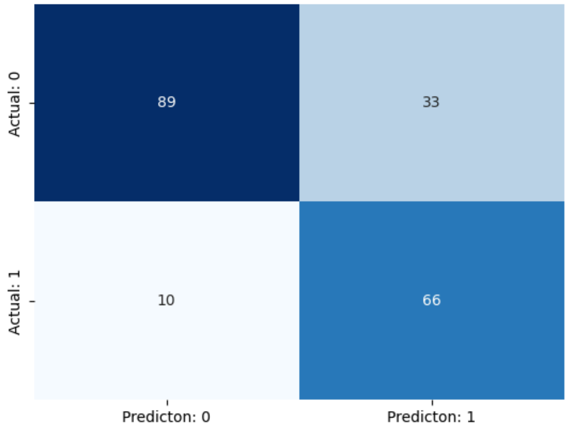


Figure 4.2 Confusion Matrix for Naive Bayes Model

4.3 Decision Trees

Table 4.3 Mean F1 Scores for Decision Tree

Mean Testing Score	Mean Training Score
0.846971	0.900410
0.845444	0.903349
0.843841	0.889134

The decision tree model ranked 4th out of the five models with a mean F1 score of 0.85. The hyperparameters used in the best scoring run are {'criterion': 'entropy', 'max_depth': 15, 'max_features': 'log2', 'max_leaf_nodes': 100, 'min_samples_leaf': 5, 'min_samples_split': 10, 'splitter': 'best'}.

It is important to note that there is considerable overfitting at around a 0.05 difference between test and training scores. Using other hyperparameters like max depth and max-leaf nodes greatly reduced overfitting and increased performance. Overfitting became more significant without these hyperparameters, with the training scores reaching 0.99.

A possible source of the overfitting is randomly oversampled data for this model. Using SMOTE and undersampling resulted in similar amounts of overfitting while having slightly lower scores. In contrast, a score of 0.40 was identified when no resampling methods were used, which is a significant decrease from other resampling methods. It is also noteworthy that the removal of encoding and outlier removal resulted in an insignificant decrease in resulting scores.

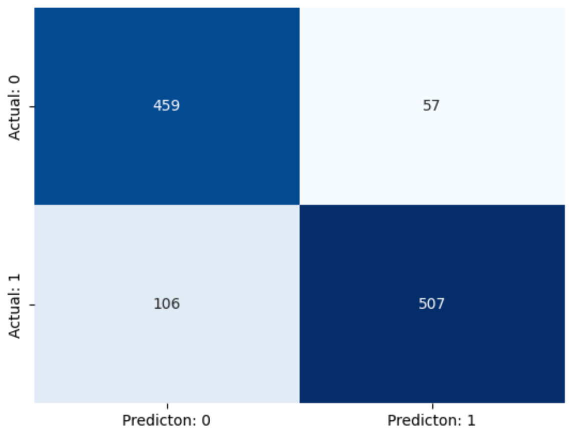


Figure 4.3 Confusion Matrix for Decision Tree

4.4 Support Vector Machine

Table 4.4 Mean F1 Scores for Support Vector Machine

Mean Testing Score	Mean Training Score
0.977347	0.994335
0.975848	0.994335
0.975066	0.993856

Out of all tested machine learning algorithms, the support vector machine model ranked the highest with a mean F1 score of 0.98. This indicates a near-perfect performance in classifying call-accepting customers from non-accepting customers. The hyperparameters tested for the highest run are {'C': 1, 'class_weight': 'balanced', 'gamma': 1}. In addition, the default kernel for the SVM model (rbf) was used. Despite a small amount of overfitting (0.02), this can be assumed to be insignificant due to the model's high test performance.

It is worth noting that the high performance of the decision tree model is only applicable to the randomly oversampled dataset. Using SMOTE and undersampling resulted in significantly lower test scores, indicating that the model's performance is sensitive to the sampling method. Additionally, other preprocessing methods did not significantly change the resulting test score.

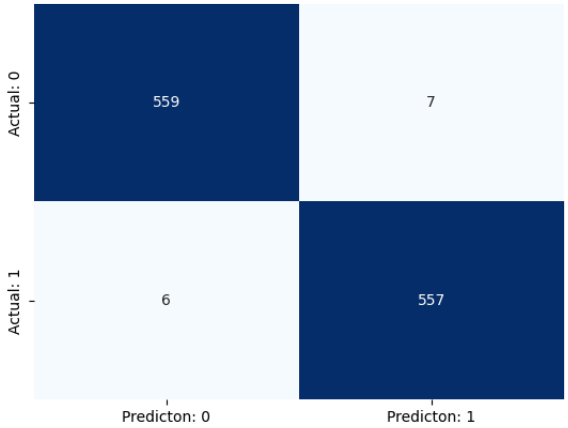


Figure 4.4 Confusion Matrix for SVM

4.5 K-Nearest Neighbors

Table 4.5 Mean F1 Scores for K-Nearest Neighbors

Mean Testing Score	Mean Training Score
0.887076	0.994517
0.886766	0.994517
0.879594	0.994517

The KNN model ranked 2nd out of the five tested models with a mean F1 score of 0.89. The best combination of parameters used was {'algorithm': 'ball_tree', 'leaf_size': 40, 'metric': 'minkowski', 'n_neighbors': 3, 'p': 1, 'weights': 'distance'} with the use of all preprocessing methods. Based on the table above, the KNN model was subject to the highest overfitting out of all models. Despite several efforts to minimize overfitting through hyperparameter tuning and resampling, no solution has been identified and must be looked into further in future research. Experiments with not removing outliers resulted in no significant change in test scores.

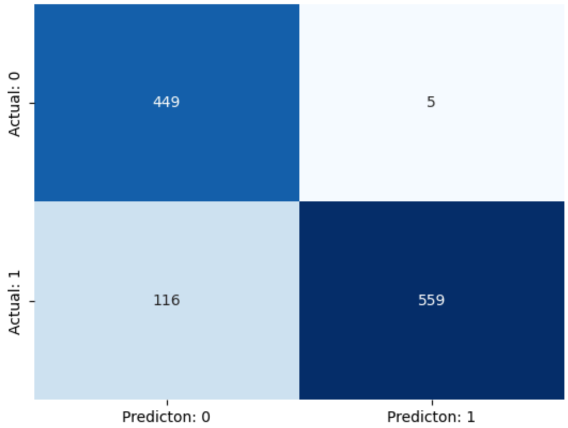


Figure 4.5 Confusion Matrix for KNN

As depicted above, most misclassifications were false negatives. There are two potential explanations for the model's behavior. First, the model might not be adequately responsive to the positive class. Second, the model might favor the negative class. Sampling bias or using biased features may contribute to this.

4.6 Summary of Results

Table 5.1 ML Algorithm Rankings

Machine Learning Algorithm	Highest Mean Test Score
1. Support Vector Machine	0.977347
2. K-Nearest Neighbors	0.887076

3. Logistic Regression	0.860521
4. Decision Tree	0.846971
5. Naive Bayes	0.754406

The support vector machine (SVM) model demonstrated exceptional performance in classifying customers who availed themselves of the year-end offer. It achieved a notable increase in the mean F1 test score compared to other models, with a significant difference of 0.09 from the next best-performing algorithm. This superior performance of the SVM model can be attributed to its ability to handle complex, non-linearly separable data effectively. In contrast, the Naive Bayes model performed the worst out of all machine learning algorithms. This may be because Naive Bayes often performs better with categorical features. The process of discretization, which is frequently employed when working with continuous data, can result in the loss of important information since it involves dividing the data into discrete bins. For a full view of the various runs using different types of resampling techniques, refer [here].

In all five machine learning models, overfitting was encountered in varying degrees. This can be attributed to resampled data (oversampled and undersampled). However, in most models except for the KNN model, overfitting was not significant enough to warrant concern. Despite this, future research must reduce overfitting for all models through a more comprehensive feature engineering process, other resampling methods, cost-sensitive learning, ensemble methods, and hyperparameter tuning.

V. Conclusion and Reflection

The study aimed to develop a machine learning model that reliably identifies users most likely to purchase the year-end offer of ABC Supermarket to reduce resource expenditure for the company. The study's proponents developed several models using different supervised machine-learning algorithms designed for classification problems to achieve this.

One difficulty the proponents have encountered during development is the classification algorithms' weakness in imbalanced datasets. The provided dataset for the models is highly imbalanced, requiring the developers to explore different methods to achieve efficient results. During development, the method of dealing with the imbalanced dataset is resampling, which manipulates the imbalance classes to be on a 50-50 distribution so that the models are not biased towards the majority class of the predicted responses, increasing classification metrics such as precision and recall. Another limitation of these models is overfitting. Due to the exploration of resampling methods, given that the developers tried oversampling and undersampling, the program's runtime significantly increased, making testing of the models significantly longer.

The results gathered during the testing of the various algorithms proved the effectiveness of the developed models, as the SVM model scored the highest in the F1 metric with a mean test score of 0.98. As observed in the confusion matrix in the previous section, the SVM model misclassified 13 out of the 1,129 entries in the testing set. In addition, the other models developed using other machine learning algorithms also scored a reliable mean F1 score, with no model having a top mean F1 score of 0.75.

In summary, given the problem presented, the proponents were able to develop an effective machine learning model that can classify availing customers for the ABC Supermarket year-end promo. As part of the reflection, we, the developers, were challenged throughout the lab exercise. Still, this activity proved to be a valuable learning experience for us as we were able to take our learnings from the previous lab activity, and we were able to construct several machine learning models with very high performance in terms of the metrics used in classification problems.

VI. References

1.4. Support Vector Machines. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/svm.html

Brownlee, J. (2021, January 4). Random Oversampling and Undersampling for Imbalanced Classification. MachineLearningMastery.com. https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/

Brownlee, J. (2020, August 17). Ordinal and One-Hot Encodings for Categorical Data. MachineLearningMastery.com. https://machinelearningmastery.com/one-hot-encoding-for-categorical-data/

Classification: Accuracy. (n.d.). Google for Developers. https://developers.google.com/machine-learning/crash-course/classification/accuracy

Shung, K. P. (2020, April 10). Accuracy, Precision, Recall or F1? - Towards Data Science. Medium. https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

Saini, A. (2024, January 23). Guide on Support Vector Machine (SVM) Algorithm. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/

One Hot Encoding in Machine Learning. (2023, April 18). GeeksforGeeks. https://www.geeksforgeeks.org/ml-one-hot-encoding-of-datasets-in-python/

What is a Decision Tree | IBM. (n.d.). https://www.ibm.com/topics/decision-trees#:~:text=A%20decision%20tree%20is%20a,internal%20nodes%20and%20leaf%20nodes.