Fabian Salvador III
Nathanael Chris Abrigo
3CSC

Laboratory Exercise #2 - Classifying Year-End Offer Purchases Using SVM

I. Introduction

ABC Supermarket aims to boost sales for their year-end promotion by offering a discounted gold membership at $499, which is usually priced at $999, exclusively for existing customers.The primary objective of this exercise is to develop a predictive model that classifies customers who might purchase the offer, using the data gathered during the last year's campaign.

The dataset includes demographic details like education level, marital status, and household composition, alongside financial information such as income. It also incorporates past purchasing behavior, including spending on various product categories, purchase frequency, and preferred purchasing channels. Additionally, factors like customer complaints within the last two years and recency of purchases are considered. The target variable is the customer's response to the previous campaign, indicating whether they accepted the offer or not.

Logistic Regression is a fundamental supervised learning algorithm used for binary classification tasks. It models the probability of a binary outcome by fitting a logistic curve to the data, making it particularly useful for predicting the probability of a categorical outcome based on one or more predictor variables. Despite its simplicity, logistic regression is robust and interpretable, making it a popular choice for various applications [1].

Support Vector Machine (SVM) is a powerful supervised learning algorithm used for both classification and regression tasks. SVM aims to find the optimal hyperplane that best separates data points into different classes while maximizing the margin between classes. It is particularly effective in handling high-dimensional data and can efficiently deal with non-linear decision boundaries through the use of kernel functions [2] .

II. Methodology

1. Installing and Importing

In this initial step, essential libraries are installed and imported to facilitate the development of the predictive model. Commonly used libraries, such as NumPy, Pandas, Matplotlib, and scikit-learn.

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, roc_auc_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np

from datetime import datetime
from sklearn.preprocessing import LabelEncoder
from scipy import stats
from sklearn.model_selection import GridSearchCV
```

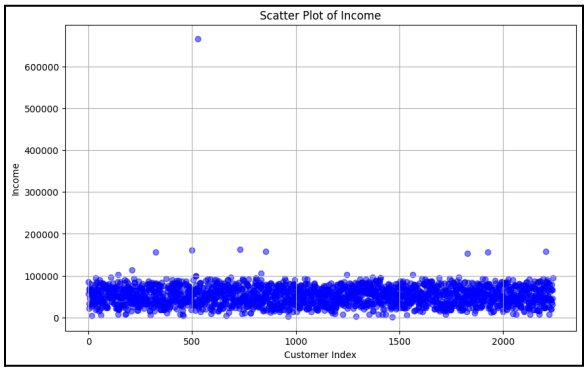
2. Data Loading

The dataset is loaded into a variable, marking the commencement of the data exploration and modeling process. It is loaded into a Pandas DataFrame for easy manipulation and analysis.

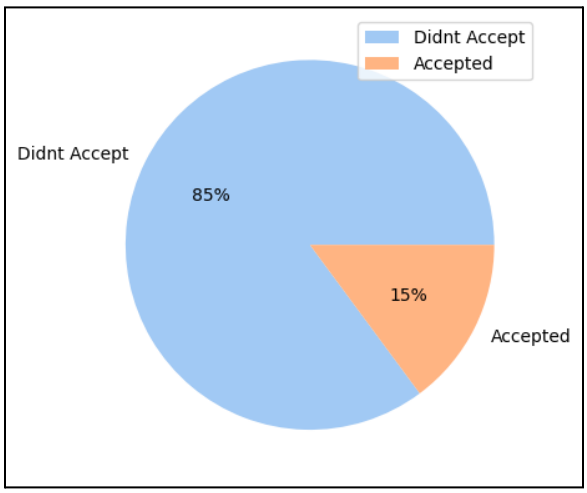
```
# Loading the data to the dataframe
df = pd.read_excel("marketing_data.xlsx")
df.shape
```

3. Exploratory Data Analysis

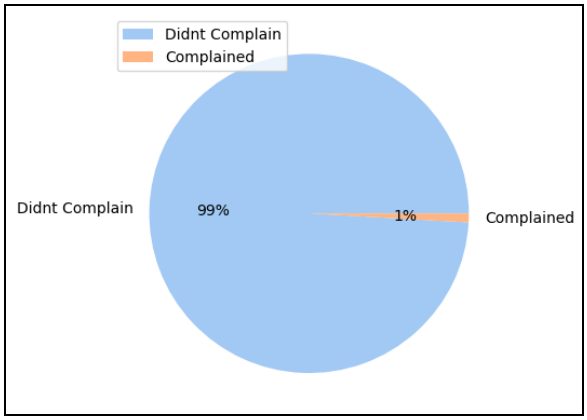
This crucial phase involves gaining insights into the dataset's structure and characteristics. Descriptive statistics, visualizations, and correlation analyses are employed to understand the distribution of variables, identify patterns, and detect potential outliers.



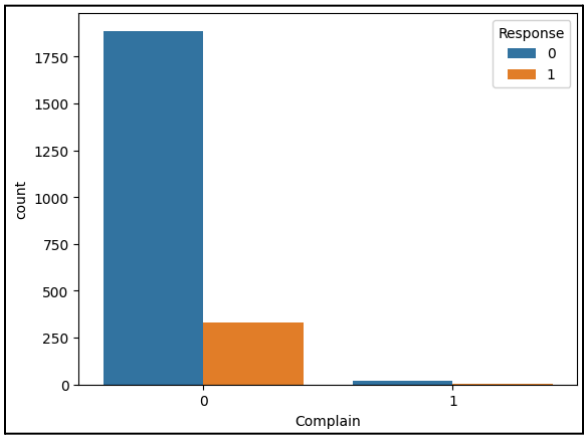
Outlier Detection in 'Income'



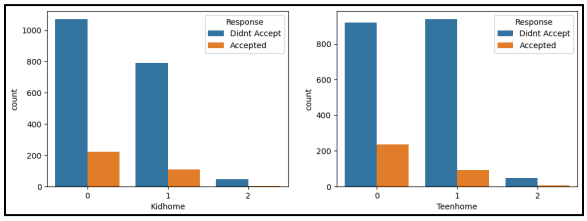
Assessment of 'Response'



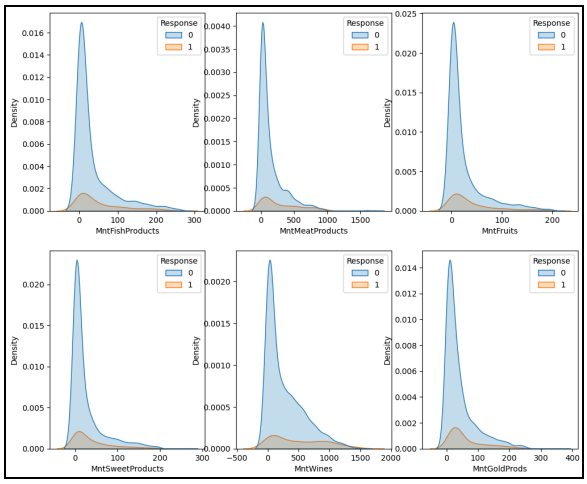
Proportion of 'Complain'



Relation Between 'Complain' and 'Response'



Relation of 'Kidhome' and 'Teenhome' on 'Response'



Amount spent on products in relation to 'Response'

4. Preprocessing

With insights from EDA, the dataset undergoes preprocessing to address issues such as missing values, outliers, and feature engineering (Hastie, Tibshirani, & Friedman, 2009). Data is cleaned, and transformations are applied to ensure it meets the requirements of the chosen modeling technique. Common preprocessing steps include handling missing data, encoding categorical variables, and scaling features.

Feature Engineering

For the feature ‘Year_Birth’, this data was transformed to ‘Age’ where it signifies the current age of the customer. ‘Age’ was then used to verify if the customer is of valid age or at least 18 years old when the account was created since most stores that involve creation of accounts require the customers to be of legal age.

For the feature ‘Income’, this data has null values which were transformed into zero but was later on decided to get the mean of its associated ‘Education’. For example if the education level of the customer is ‘PhD’, then instead of remaining the zero in ‘Income’, it was decided to take the average income of all the customers that are ‘PhD’ to have a more realistic ‘Income’ value.

For the feature ‘Dt_Customer’, this data was transformed into ‘Days_Since_Enrollment’ where it signifies how long a customer enrolled in the company.

For the feature 'Is_Married', this data was derived from 'Marital_Status' to further signify if the customer is married or not. ‘Married’ and ‘Together’ inputs for the ‘Marital_Status’ were considered 1 or true in 'Is_Married' while the other inputs were considered 0 or false. ‘Together’ input was decided to be considered married since being together with a significant other also more or less act like married couples.

For the feature ‘Total_Amount_Spent’, this data was derived from the sum of all products which are 'MntFishProducts', 'MntMeatProducts', 'MntFruits', 'MntSweetProducts', 'MntWines', and 'MntGoldProds’. This is to signify the total spending of the customer buying all types of products.

For the feature ‘Has_Children’, this data was derived from the sum of ‘Kidhome’ and ‘Teenhome’ which signifies if the customer has children since children can also contribute to the amount of products the customer needs to purchase.

For the feature ‘Total_Spent_to_Income_Ratio', this data was derived from dividing 'Total_Amount_Spent' over 'Income' which signifies the relationship between the both

features. Since the customer can only spend based on his/her income unless there is external aid such as giveaways.

For the feature ‘Enrollment_Month', this data was derived from ‘'Dt_Customer' which extracted the month from the date of enrollment in the company. This feature can look into which among the months have the most enrollments since seasons like Christmas can have an effect on the number of enrollments.

For the feature 'Spent_Per_Household_Member', this data was derived from 'Total_Amount_Spent' and 'Household_Size'. 'Household_Size' was then derived from the sum of 'Kidhome', 'Teenhome’ and 'Is_Married' plus one. The feature 'Spent_Per_Household_Member' highlights the estimated total amount spent by each member of the household.

For the feature 'PreferredChannel', this data was derived from comparing which have the most number of purchases among the types of purchases which are 'NumDealsPurchases', 'NumCatalogPurchases', 'NumStorePurchases', and 'NumWebPurchases'. This feature will write down which among the four was the customer’s most preferred type of purchase.

Since some of the features which are ‘Education’ and ‘PreferredChannel’ are not booleans which consist of 1’s and 0’s, or numerical values, this feature requires label encoding to have a number value so that the model can process the data.

Handling Outliers

```
# Removing Outliers

# Calculate Z-scores for each feature
z_scores = stats.zscore(X_train)

# Define a threshold for Z-score
threshold = 3

# Find indices of outliers
outlier_indices = (abs(z_scores) > threshold).any(axis=1)

# Remove outliers from the training data
X_train = X_train[~outlier_indices]
y_train = y_train[~outlier_indices]

# Print the number of outliers removed
print("Number of outliers removed:", sum(outlier_indices))
```

Outliers in the training data are handled through the calculation of Z-scores for each feature. A threshold is defined to identify outliers based on their Z-scores exceeding a certain limit. By applying this threshold, indices of outliers are determined, and these instances are subsequently removed from both the feature and target arrays. Finally, the code prints the number of outliers that have been successfully eliminated from the training data. This method of outlier handling helps ensure that the data used for model training is free from potentially influential or erroneous observations, thus enhancing the robustness and reliability of the subsequent analysis.

Splitting the data into training and testing sets.

```
# Generate synthetic data for demonstration
X, y = make_classification(n_samples=1000, n_features=20, n_classes=2, random_state=42)

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

This step ensures that the model is trained on a portion of the data and evaluated on unseen data to assess its generalization performance accurately.

Standardization of features using StandardScaler

```
# Initialize the StandardScaler
scaler = StandardScaler()

# Fit the scaler on the training data and transform both training and test data
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

A StandardScaler object is initialized to standardize the features of the dataset. The scaler is then fitted to the training data using the fit_transform method, which calculates the mean and standard deviation of each feature in the training set and scales the data accordingly. The resulting scaling parameters are then applied to both the training and test sets using the transform method to ensure that both sets are standardized in the same way, maintaining consistency between the training and testing data.

5. Modeling

Handling Class Imbalance using SMOTE

```
# Handling class imbalance with SMOTE
smote = SMOTE(random_state=42)
X_train_balanced, y_train_balanced = smote.fit_resample(X_train_scaled, y_train)
```

The Synthetic Minority Over-sampling Technique (SMOTE) is employed to address class imbalance in the dataset. SMOTE generates synthetic samples for the minority class by interpolating between existing minority class samples. By doing so, SMOTE helps to balance the class distribution in the training dataset, thereby mitigating the potential bias towards the majority class during model training. This technique aims to improve the model's performance by ensuring that both classes are adequately represented in the training data, leading to more robust and accurate predictions, particularly in scenarios where class imbalances are present.

Feature Selection using Random Forest Classifier

```
# Feature selection using Random Forest
clf_rf = RandomForestClassifier(random_state=42)
clf_rf.fit(X_train_balanced, y_train_balanced)
feature_importance = clf_rf.feature_importances_
selector = SelectFromModel(clf_rf, threshold=0.05)
selector.fit(X_train_balanced, y_train_balanced)
X_train_selected = selector.transform(X_train_balanced)
X_test_selected = selector.transform(X_test_scaled)
```

A Random Forest Classifier is utilized to perform feature selection. This process involves training a Random Forest model on the balanced training data to determine the importance of each feature in predicting the target variable. Subsequently, a threshold is applied to select the most relevant features based on their importance scores. Features that meet the specified threshold are retained, while those below the threshold are discarded. This technique helps to reduce the dimensionality of the dataset by retaining only the most informative features, thereby improving model performance, reducing overfitting, and enhancing interpretability.

Hyperparameter tuning using RandomizedSearchCV

```
# Define the hyperparameter distribution for Randomized Search
param_distribution = {'C': uniform(loc=0, scale=10),
                      'kernel': ['linear', 'rbf', 'poly', 'sigmoid'],
                      'gamma': ['scale', 'auto']}

# Perform Randomized Search with 5-fold cross-validation
random_search = RandomizedSearchCV(clf, param_distributions=param_distribution, n_iter=100, cv=5, scoring='accuracy', random_state=0)
random_search.fit(X_train_selected, y_train_balanced)

# Get the best hyperparameters
best_params = random_search.best_params_
print("Best Hyperparameters:", best_params)

# Initialize SVM classifier with the best hyperparameters
best_clf = SVC(C=best_params['C'], kernel=best_params['kernel'], gamma=best_params['gamma'], random_state=0)
```

A Randomized Search Cross-Validation technique is employed to optimize the hyperparameters of the model. The hyperparameters for the model, such as the regularization parameter C, kernel type, and gamma value, are defined along with their respective search spaces. The RandomSearchCV algorithm then conducts an efficient search over these hyperparameters, evaluating each combination using cross-validation to

identify the set of hyperparameters that yields the best model performance, as measured by the specified scoring metric (accuracy in this case). This process aids in improving the predictive accuracy and generalization capability of the model by finding the optimal hyperparameters tailored to the specific dataset and problem at hand.

6. Evaluation

The evaluation phase is used for assessing the performance and accuracy of the model. It involves both quantitative and qualitative evaluations.

```
# Evaluate performance metrics
acc_tuned = accuracy_score(y_test, preds_tuned)
prec_tuned = precision_score(y_test, preds_tuned)
rec_tuned = recall_score(y_test, preds_tuned)
f1_tuned = f1_score(y_test, preds_tuned)
auc_tuned = roc_auc_score(y_test, preds_tuned)

print("\nSVM:")
print("Accuracy: {:.4f}".format(acc_tuned))
print("Precision: {:.4f}".format(prec_tuned))
print("Recall: {:.4f}".format(rec_tuned))
print("F1 Score: {:.4f}".format(f1_tuned))
print("AUC: {:.4f}".format(auc_tuned))

# Check for overfitting with tuned model
print('\nTraining set score: {:.4f}'.format(best_clf.score(X_train_selected, y_train_balanced)))
print('Test set score: {:.4f}'.format(best_clf.score(X_test_selected, y_test)))
```

These metrics include accuracy, precision, recall, F1 score, and area under the receiver operating characteristic curve (AUC). By computing these metrics, the effectiveness and robustness of the tuned model in classifying the test data are evaluated. Additionally, the training and test set scores are compared to assess potential overfitting or underfitting issues, providing insights into the generalization capability of the model.

III. Experiments

This section includes various experiments conducted throughout the activity.

Initial:

```
X = df.drop('Response', axis=1).iloc[:,1:][[
    'MntFishProducts',
    'MntMeatProducts',
    'MntFruits',
    'MntSweetProducts',
    'MntWines',
    'MntGoldProds'
    ]]
y = df['Response']
```

The initial model only consist of six features in the baseline modeling which are all products specifically 'MntFishProducts', 'MntMeatProducts', 'MntFruits', 'MntSweetProducts', 'MntWines', and 'MntGoldProds'.

1st Attempt:

```
X = df.drop('Response', axis=1).iloc[:,1:][[
    'Age',
    'MntFishProducts',
    'MntMeatProducts',
    'MntFruits',
    'MntSweetProducts',
    'MntWines',
    'MntGoldProds']]
```

After studying the initial program given, it is observed that some features are not included in the baseline modeling procedure. That is why it was decided to add the feature 'Age' to the encoding procedure since this value can affect the classification and was also suggested by the researcher's instructor.

2nd + 3rd Attempt

```
X = df.drop('Response', axis=1).iloc[:,1:][[
    'Education_encoded',
    'Marital_Status_encoded',
    'Days_Since_Enrollment',
    'Age',
    'MntFishProducts',
    'MntMeatProducts',
    'MntFruits',
    'MntSweetProducts',
    'MntWines',
    'MntGoldProds']]
```

After making the 1st Attempt, there has been a deliberation regarding the data transformation of some features. First, is how to utilize the data that are neither boolean or numeric values which are ‘Education’ and ‘Marital_Status’. It was discovered that using label encoding can help substitute the data into a numeric value. ‘Education_encoded’ and ‘Marital_Status_encoded’ are added on the baseline modeling.

Second, is the ‘Dt_Customer’ where the data consist of dates that are in different formats. This was fixed and since the model cannot process dates, it was decided to transform the feature into 'Days_Since_Enrollment' which is the number of days since the customer enrolled. 'Days_Since_Enrollment' is also added on the baseline modeling.

After making the 2nd Attempt, it was observed there are outliers that exist in the dataset. As seen in the scatter plot of ‘Income’ where there is data far away compared to the norm which could be eliminated. Therefore, it was decided to add an outlier detection and removal method in the program.

4th Attempt:

```
X = df.drop('Response', axis=1).iloc[:,1:][[
    'Complain',
    'Recency' ,
    'Total_Amount_Spent',
    'Has_Children',
    'Total Children',
    'Education_encoded',
    'Marital_Status_encoded',
    'Days_Since_Enrollment',
    'Age',
    'MntFishProducts',
    'MntMeatProducts',
    'MntFruits',
    'MntSweetProducts',
    'MntWines',
    'MntGoldProds',
    'NumDealsPurchases',
    'NumWebPurchases',
    'NumCatalogPurchases',
    'NumStorePurchases',
    'NumWebVisitsMonth'
    ]]
```

After making the 3rd Attempt, there have been reflections as to what more feature engineering can be done to improve the metrics. 'Total_Amount_Spent', 'Has_Children', 'Total Children' are derived and added on the baseline modeling. Also added 'Complain', 'Recency', and all purchases which are 'NumDealsPurchases', 'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth' in baseline modeling.

5th Attempt:

```
X = df.drop('Response', axis=1).iloc[:,1:][[
    'Income_to_Total_Spent_Ratio',
    'Complain',
    'Recency' ,
    'Has_Children',
    'Total Children',
    'Education_encoded',
    'Marital_Status_encoded',
    'Days_Since_Enrollment',
    'Age',
    'MntFishProducts',
    'MntMeatProducts',
    'MntFruits',
    'MntSweetProducts',
    'MntWines',
    'MntGoldProds',
    'NumDealsPurchases',
    'NumWebPurchases',
    'NumCatalogPurchases',
    'NumStorePurchases',
    'NumWebVisitsMonth'
    ]]
```

After making the 4th Attempt, 'Income_to_Total_Spent_Ratio' replaced the 'Total_Amount_Spent' on the baseline modeling since 'Income_to_Total_Spent_Ratio' better describes the relationship between 'Income' and 'Total_Amount_Spent'

6th Attempt:

```
X = df.drop('Response', axis=1).iloc[:,1:][[
    'Is_Married',
    'Income_to_Total_Spent_Ratio',
    'Complain',
    'Recency' ,
    'Has_Children',
    'Total Children',
    'Education_encoded',
    'Days_Since_Enrollment',
    'Age',
    'MntFishProducts',
    'MntMeatProducts',
    'MntFruits',
    'MntSweetProducts',
    'MntWines',
    'MntGoldProds',
    'NumDealsPurchases',
    'NumWebPurchases',
    'NumCatalogPurchases',
    'NumStorePurchases',
    'NumWebVisitsMonth'
    ]]
```

After making the 5th Attempt, 'Is_Married' replaced 'Marital_Status_encoded' on baseline modeling since 'Is_Married' better describes if the customer is single or married.

7th Attempt:

```
X = df.drop('Response', axis=1).iloc[:,1:][[
    'Kidhome',
    'Teenhome'
    'Is_Married',
    'Income_to_Total_Spent_Ratio',
    'Complain',
    'Recency' ,
    'Has_Children',
    'Education_encoded',
    'Days_Since_Enrollment',
    'Age',
    'MntFishProducts',
    'MntMeatProducts',
    'MntFruits',
    'MntSweetProducts',
    'MntWines',
    'MntGoldProds',
    'NumDealsPurchases',
    'NumWebPurchases',
    'NumCatalogPurchases',
    'NumStorePurchases',
    'NumWebVisitsMonth'
    ]]
```

After making the 6th Attempt, it was observed that 'Total_Children' negatively impacts the metrics of the classification which lead to 'Total_Children' being removed from the baseline modeling. It is also decided to add 'Kidhome' and 'Teenhome' on the baseline modeling.

8th + 9th Attempt:

```
X = df.drop('Response', axis=1).iloc[:,1:][[
    'Spent_Per_Household_Member',
    'Enrollment_Month' ,
    'Kidhome',
    'Teenhome'
    'Is_Married',
    'Income_to_Total_Spent_Ratio',
    'Complain',
    'Recency' ,
    'Has_Children',
    'Education_encoded',
    'Days_Since_Enrollment',
    'Age',
    'MntFishProducts',
    'MntMeatProducts',
    'MntFruits',
    'MntSweetProducts',
    'MntWines',
    'MntGoldProds',
    'NumDealsPurchases',
    'NumWebPurchases',
    'NumCatalogPurchases',
    'NumStorePurchases',
    'NumWebVisitsMonth'
    ]]
```

After making the 7th Attempt, there was a discovery of possible connections between total spending and total number of people. Also, the month in which the customer enrolled in the company. 'Spent_Per_Household_Member' and 'Enrollment_Month' was added on the baseline modeling.

After making the 8th Attempt, there are also actions to check the validity of the customer enrollment in the company. It is decided that customers should be at least 18 years old to be considered valid. If not valid, drop row.

10th Attempt Baseline Encoding:

```
X = df.drop('Response', axis=1).iloc[:,1:][[
    'PreferredChannel',
    'Spent_Per_Household_Member',
    'Enrollment_Month' ,
    'Kidhome',
    'Teenhome'
    'Is_Married',
    'Total_Spent_to_Income_Ratio',
    'Complain',
    'Recency' ,
    'Has_Children',
    'Education_encoded',
    'Days_Since_Enrollment',
    'Age',
    'MntFishProducts',
    'MntMeatProducts',
    'MntFruits',
    'MntSweetProducts',
    'MntWines',
    'MntGoldProds',
    'NumDealsPurchases',
    'NumWebPurchases',
    'NumCatalogPurchases',
    'NumStorePurchases',
    'NumWebVisitsMonth'
    ]]
```

After making the 9th Attempt, it was decided to change ‘Income_to_Total_Spent_Ratio’ into 'Total_Spent_to_Income_Ratio' since this better describes the relationship of total spending and income. It was also decided to add ‘PreferredChannel’ to the baseline modeling to show which among the types of purchases does the customer prefer.

Modeling Attempts

Logistic Regression + GridSearch

Recognizing the importance of hyperparameter tuning, we initially employed GridSearch to enhance our Logistic Regression model. This approach not only improved model performance but also effectively addressed overfitting post-preprocessing.

Logistic Regression + RandomizedSearch

Through trial and error, we discovered that RandomizedSearch outperformed GridSearch. As a result, we transitioned to RandomizedSearch for hyperparameter optimization, further refining our model.

Logistic Regression + RandomizedSearch + SMOTE + RandomForestClassifier

Seeking further improvement, we addressed class imbalance and feature selection by incorporating SMOTE and RandomForestClassifier. This significantly enhanced our model's performance.

SVM + RandomizedSearch + SMOTE + RandomForestClassifier

Through code modifications and experimentation, we found that SVM is a better alternative to Logistic Regression. This adjustment resulted in a small improvement in model performance, leading us to adopt SVM as our primary model.

Other Models

We also attempted to utilize other models including KNN, Naive Bayes, and Decision Trees.However, the results did not meet our expectations and it performed worse compared to SVM and Logistic Regression. Therefore, we opted to proceed with SVM.

IV. Results

This section will present the summary of changes and the results of the experiments made for the study.

Initial Results:	Accuracy: 0.8512 Precision: 0.5000 Recall: 0.0200 F1: 0.0385 AUC: 0.5083
1st Attempt Results: - converted 'Year_Birth' to its 'Age' equivalent - ‘Age’ added in baseline modeling	Accuracy: 0.8512 Precision: 0.5000 Recall: 0.0400 F1: 0.0741 AUC: 0.5165
2nd Attempt Results: - 'Education' and 'Marital Status' was converted using label encoding so that the model can process the data - 'Dt_Customer' converted to 'Days_Since_Enrollment' - 'Days_Since_Enrollment' added in baseline modeling	Accuracy: 0.8571 Precision: 0.6667 Recall: 0.0800 F1: 0.1429 AUC: 0.5365
3rd Attempt Results: - incorporated outlier detection and removal	Accuracy: 0.8497 Precision: 0.4783 Recall: 0.1100 F1: 0.1789 AUC: 0.5445

4th Attempt Results:

- added Feature Engineering which includes 'Total_Amount_Spent', 'Has_Children', 'Total Children' also added in baseline modeling
- added 'Complain', 'Recency' in baseline modeling
- added 'NumDealsPurchases', 'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth' in baseline modeling

Accuracy: 0.8557
Precision: 0.5200
Recall: 0.3900
F1: 0.4457
AUC: 0.6635

5th Attempt Results:

- Removed 'Total_Amount_Spent' on baseline modeling
- Added feature engineering ‘Income_to_Total_Spent_Ratio’ which consist of the relationship of ‘Income’ and 'Total_Amount_Spent'
- Added ‘Income_to_Total_Spent_Ratio’ on baseline modeling

Accuracy: 0.8616
Precision: 0.5467
Recall: 0.4100
F1: 0.4686
AUC: 0.6753

6th Attempt Results:

- Removed ‘Marital_Status_encoded’ on baseline modeling
- Added feature engineering ‘Is_Married’ on baseline modeling

Accuracy: 0.8646
Precision: 0.5542
Recall: 0.4600
F1: 0.5027
AUC: 0.6977

7th Attempt Results:

- removed 'Total_Children' on baseline modeling
- added 'Kidhome' and 'Teenhome'

Accuracy: 0.8661
Precision: 0.5595
Recall: 0.4700
F1: 0.5109
AUC: 0.7027

8th Attempt Results:

- added feature enginnering 'Spent_Per_Household_Member' and 'Enrollment_Month' on baseline modeling

Accuracy: 0.8705
Precision: 0.5765
Recall: 0.4900
F1: 0.5297
AUC: 0.7135

9th Attempt Results:

- added validity of 'Year_Birth' and 'Dt_Customer' and also check whether or not they are at least 18 years old.

Logistic Regression Accuracy: 0.8765
Logistic Regression Precision: 0.6049
Logistic Regression Recall: 0.4900
Logistic Regression F1: 0.5414
Logistic Regression AUC: 0.7170

10th Attempt Results:

- changed ‘Income_to_Total_Spent_Ratio’ into 'Total_Spent_to_Income_Ratio'
- added feature engineering 'PreferredChannel' on the baseline modeling

Logistic Regression Accuracy: 0.8839
Logistic Regression Precision: 0.6279
Logistic Regression Recall: 0.5400
Logistic Regression F1: 0.5806
Logistic Regression AUC: 0.7420

Modeling Results

Logistic Regression Base After Preprocessing

Logistic Regression Accuracy: 0.8839
Logistic Regression Precision: 0.6279
Logistic Regression Recall: 0.5400
Logistic Regression F1: 0.5806
Logistic Regression AUC: 0.7420

Logistic Regression + GridSearch

Accuracy: 0.8824
Precision: 0.6207
Recall: 0.5400
F1 Score: 0.5775
AUC: 0.7412

Training set score: 0.8918
Test set score: 0.8824

Logistic Regression + RandomizedSearch

Accuracy: 0.8550
Precision: 0.9149
Recall: 0.8037
F1 Score: 0.8557
AUC: 0.8589

Training set score: 0.8825
Test set score: 0.8550

Logistic Regression + RandomizedSearch + SMOTE + RandomForestClassifier

Accuracy: 0.8700
Precision: 0.9175
Recall: 0.8318
F1 Score: 0.8725
AUC: 0.8729

Training set score: 0.8686
Test set score: 0.8700

SVM + RandomizedSearch + SMOTE + RandomForestClassifier

Accuracy: 0.8750
Precision: 0.9184
Recall: 0.8411
F1 Score: 0.8780
AUC: 0.8775

Training set score: 0.8845
Test set score: 0.8750

V. Conclusion & Recommendations

With the final metrics of 0.8750 for accuracy, 0.9184 for precision, 0.8411 for recall, 0.8780 for F1 score, and 0.8775 for AUC, the model shows its decent performance across multiple evaluation criteria.

During the experimentation phase, multiple trials and errors and realizations about which features in the data set are crucial to classifying year-end offer purchases. Features like 'Year_Birth' are essential, but data transformation is necessary to utilize the data in that column. String features like 'Education' require the label encoding method to help the program process the

value. This study also requires feature engineering to create new features that can further describe the relationship between variables like 'Total_Spent_to_Income_Ratio' which describes the relationship between total spending and income.

This experimentation phase also required the researchers to search for methods and models that can provide better metrics for the study. Initially, the researchers stuck with the initial model provided which is Logistic Regression while doing the feature engineering. Later in the development of the project, the researchers tried applying different models such as Decision Trees, Naive Bayes, KNN, and SVM. The researchers also attempted to apply different suggestions such as applying hyperparameters, cross validation, and other more methods to further improve the efficiency of the model.

After multiple experiments, the model was finalized with the usage of SVM. The model involved handling class imbalance with SMOTE, feature selection using Random Forest, defining and performing the hyperparameter distribution for Randomized Search, and using an SVM classifier with the best hyperparameters to make the evaluation.

Based on the findings that are discovered during the course of the study, it is recommended that searching for a suitable model with the right methods to have better metrics is crucial for the efficiency and reliability of the model.

Another recommendation is to provide further features that can factor into the classifying year-end offer purchases. For instance, accessibility to the store, delivery time with web-purchases, and other more features that can affect the classification. Also, more feature engineering in the given dataset can help in the classification. These possible features may provide better metrics to the program, given that the data is processed thoroughly to ensure its integrity.

This study helped the researchers further understand how to apply the lessons about classification in a program to apply it in real life scenarios.

VI. References

[1] Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: data mining, inference, and prediction. Springer Science & Business Media.

[2] Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning, 20(3), 273-297.