VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

B. Tech., Winter Semester 2022-2023

CSE3020 - Data Visualization

Project Report

Business Marketing Efficiency Analysis and Prediction

Team Members

Manigandan R – 20BCE1223
Janapati Sarath Chandra – 20BCE1138
Rishi Nair – 20BCE1922

Course Faculty

Dr. Joshan Athanesious J

Abstract:

Businesses require marketing strategies to be better at providing services to account holders that they may require. Giving customers what they need is a great way to build trust between the customer and the organization. Thus, targeted marketing is an essential requirement when it comes to marketing strategies, and this is where machine learning can be used. Machine learning models can be used to accurately predict customers' buying habits and this information can be further used to create an effective marketing strategy for businesses that will help them avoid losses and get better business results. Using models like random forest, decision trees, logistic regression and multivariate regression we provide businesses with useful data to enable them to reach a broader audience.

Objective of project:

The objective of this project is to utilize machine learning algorithms such as random forest and decision tree to develop predictive models that can assist businesses in customizing their marketing strategies and streamlining their processes to improve customer service, increase the effectiveness of targeted campaigns, and reduce unnecessary costs. The aim is to create a Business Marketing Efficiency Predictor that can accurately predict the efficiency of a business's marketing strategies, identify the key factors that influence marketing efficiency, and provide actionable insights for optimizing those factors. The ultimate goal is to help businesses achieve greater profitability by leveraging data-driven insights to enhance customer engagement and optimize marketing efforts.

Methods used in the project:

- Data Collection: Collecting and cleaning data related to the business's marketing efforts, customer demographics, and customer behavior using tools such as web scraping, data mining, and data cleansing techniques.
- Feature Selection: Identifying the relevant features that have a significant impact on marketing efficiency using feature selection techniques such as correlation analysis, chi-squared test, and mutual information gain.
- Data Preprocessing: Preprocessing the data using techniques such as scaling, normalization, and dimensionality reduction to prepare it for machine learning algorithms.
- Algorithm Selection: Choosing suitable machine learning algorithms such as random forest and decision tree based on the nature of the problem and the characteristics of the data.

- Model Training and Evaluation: Training the selected models using the prepared data and evaluating their performance using appropriate metrics such as accuracy, precision, recall, and F1 score.
- Model Optimization: Fine-tuning the model parameters using hyperparameter optimization techniques such as grid search and random search to improve its performance.
- Model Interpretation: Interpreting the models to gain insights into the key factors that influence marketing efficiency and generating actionable recommendations for optimizing those factors.

Outcome of project:

The outcome of the project would be a machine learning-based solution that can provide businesses with a predictive model to enhance the effectiveness of their marketing strategies. The model would be able to analyze the business's customer data and marketing efforts to provide personalized recommendations for optimizing marketing campaigns, which would increase the business's profitability and customer satisfaction.

By leveraging advanced data analytics and machine learning techniques, the "Business Marketing Efficiency Predictor" would enable businesses to streamline their marketing processes, reduce unnecessary costs, and make informed decisions about their marketing strategies. The model would also provide insights into the customer's behavior and preferences, which would help the business to better understand its customer base and improve customer engagement.

Scope of project:

The scope of the project involves the development of a predictive model that can help businesses improve the efficiency and effectiveness of their marketing strategies. The project would involve collecting and analyzing data related to the business's marketing efforts, customer demographics, and customer behavior.

The scope of the project would also include designing and developing a user-friendly interface such as a web application or dashboard that would allow businesses to input their marketing strategies and receive real-time recommendations. The interface would be intuitive and easy to use, and it would enable businesses to streamline their marketing processes and make data-driven decisions.

1. Introduction:

Marketing has changed in recent times, and it has been heavily incorporating technology as a part of it. Technological advancements, especially in the field of data science has changed the way of approach towards targeted marketing or marketing strategies in general. Businesses deal with thousands of customers and they need to be able to stay profitable alongside providing the best possible services to their customers. Having trust between customers and businesses is essential for its success as a business. To be able to give customers exactly what they require, businesses need to adapt to the changes in the field of marketing. The use of machine learning and data science is one of those changes that need to be adapted. Using machine learning we can predict whether a customer will buy a product or not, how much money they are going to spend on the products and how many new products they will buy. This information can be vital to come up with an effective targeted marketing approach.

2. Literature Review:

Having a competitive advantage in sales and marketing gives a great advantage and increases the retailer revenue. So, if the company knows what customer wants to purchase and when the customer will purchase, they can market that product at the time to the customer and make the customer purchase it. This will provide a great competitive advantage. The next purchase date prediction is done using the Next Purchase Date predictor (NPD). Whenever a customer purchases a product from a retailer, it is recorded. This data and historical data are given to the Next Purchase Date (NPD) predictor which uses machine learning to predict when the customer will be the product again. Based on this info the product is marketed at the same time.

In [1], they have used models like Recurrent Neural Network, Linear Regression, Artificial Neural Network and Extreme Gradient Boosting to develop NPD. Instacart online grocery shopping dataset was used to develop the model. It has more than 3 million records from 2,00,000 unique customers. The features of the dataset consist of orders, departments, products, aisles, and order products table. The order table contains a feature day_since_prior_order which tells the gap in terms of days between two consecutive orders. The target feature was derived from the dataset which is the next purchase date per user product pair. The records which had outliers were removed and if null records were there for continuous values, mean was taken and it was filled. Finally, the extracted features are days between orders per product, days since prior order per product and other features such as user id, product id, max days, min days and average days. First, they have implemented Recurrent Neural

Networks in Python using PyTorch. For the activation layer, they used a rectified linear unit (ReLU) with a mean squared error loss criterion. The model has ten hidden layers and the learning rate was 0.01. Second, they implemented a linear regression model.They implemented linear regression using scikit's SGDRegressor which is Stochastic Gradient Descent Regressor. The learning rate of the model was 0.001. The third model which was implemented was XGBoost. It was implemented using the XGBoost package in Python. They considered non-sequence-based features in this model also. The learning rate of this model was 0.1 and the criterion which the model had was Root mean squared error. The fourth model which they developed was the Artificial Neural Network. A rectified linear unit (ReLU) activation function was used with a mean squared error loss criterion and an Adam optimiser. This model had twelve neurons in the input layer meaning it had twelve input features, one hidden layer with eight neurons and one output layer which had one neuron.

In [2], they have used two linear models, three machine learning models and two deep learning models to perform sales prediction and the results have been compared. The main reason for performing an efficient sales prediction is to get ahead of the ecommerce competition as the number of internet users are rising rapidly and the number of products sold online is also increasing so customer satisfaction plays an important role here. Generally, the indicators for sales forecast are categorized into three which are attribute characteristic index of the product such as price of the product, product customer praise rate, product promotion information etc. The second category is product reviews and derivative indicators for the particular product, review text for that product, review time, number of reviews that product has received, reviewer information. The third category is product search information through online mode. In this paper, they have considered the impact of the first category on the sales prediction. In [2], they have used the M5 dataset. The M5 dataset mainly contains data which is used for sales forecasting. They have obtained the data from Walmart. This dataset consisted of information of 3049 products which are categorized into household products, food products and hobbies products. It included their prices, selling time, and branch name. They collected data from 10 Walmart branches from three states including their historical daily unit sales data. For any uncertain values in price, they took the average of the price of that product category. For null values in selling time, they dropped the whole row. They have used multivariate regression models which have many independent variables. The next model which they have used is XGBoost which is a gradient boosting decision tree. Here, the model will generate k trees with different functions. To get the result we add all the final results of the trees. The third machine learning model was developed using a random forest algorithm which falls under ensemble learning.

This model generated many trees and the final result is chosen using majority rule.In deep learning, they have used MLP andLSTM model. In the multilayer perceptron model, one input layer, one hidden layer and one output layer was included. In the LSTM model, 1 input layer, two hidden layers and one output layer are included. In both, they have used Rectified Linear Unit (ReLU) as activation function. Linear models such as Triple exponential smoothing and ARIMA models were used. These models were implemented using Python 3.7 and packages like scikit and TensorFlow were used. Root mean squared error and training time was used to evaluate the model. The dataset was divided based on dates for testing and validation. It has been mentioned that as more input features are added, the performance increases. At the same time, they have mentioned that the regression model has performed better than complex machine learning and deep learning models.

In [3], the focus is on Dynamic Pricing and its effect on customers buying a product. Dynamic pricing is one of the price optimization mechanisms where the price of a particular product is decided based on customer demands. In other words, the price of the product is determined by the market conditions. A good example for this method will be flight ticket booking. So, this type of pricing also helps to maintain a low price during low demand for the product and high price during high demand for the product. In order to attract many customers, many vendors are adapting dynamic pricing. So, a better method is needed to price the product in a better way using machine learning so that the customer will buy the product. They implemented the model in such a way that they first identified and divided the customer segments into a group then they set a price range (dynamically) for each group and finally they predicted whether the customer will buy the product or not.

The dataset had two schemas – transaction dataset and offer dataset. The transaction dataset had all the info related to customer transactions and the offer dataset had information regarding different offers and price discounts for the product. In the transaction dataset all the features were categorical except purchase amount and purchase quantity which were continuous. In the offer dataset only product quantity and offer value were continuous whereas other features were categorical. For preprocessing tools like R, Excel, SAS were used. Many derived variables such purchase by category, purchase by quantity - PQT, purchase by offer - POR, purchase by brand - PBD, purchase by company PCY, purchase by category - PCT and purchase by channel - PCN were added. The outliers from the dataset were removed by removing the whole record. For grouping customer segments, K Means clustering algorithm was used which is one of the unsupervised algorithms. Clustering of customers was done based on the derived attributes mentioned above. The overall coefficient of variation was found to be 83%. A Total of four clusters

were formed. After the formation of a cluster, for each cluster a dynamic price range is determined. They have given different prices for different segments based on the segment characteristics.

In [4], there is a wealth of literature on the many approaches and analytics used to anticipate bank default. Demyanyk and Iftekhar (2010) provide an overview of several publications that analyze, predict, and offer solutions for possible financial crises or bank failures. For the sake of completeness, Appendix A provides a synopsis of the literature that addresses the issue of anticipating bank failures, describing the statistical methods and dataset used. CAMELS indicators are the focus of a significant portion of the work on predicting bank collapse. This is an acronym representing the measures that investors and regulators often use to determine if a financial institution is sound: capital, asset quality, management, earnings, liquidity, and sensitivity to market risk. A number of empirical investigations additionally integrate CAMELS with additional indicators in an effort to improve the model's capacity for explanation. On the other hand, there is conflicting information regarding the factors that are crucial for predicting the future bank insolvencies. According to Poghosyan (2009), measures for profit, capitalization etc can be used to spot weak banks. According to Berger and Bouwman (2013), capital has a favorable effect on the chances of survival and market shares of small banks. While Mayes and Stremmel (2012) suggested that risk-weighted capital ratios underperform the leverage ratio. This study uses a wide range of CAMEL-related variables, together with other transformations, in an effort to provide a conclusive answer as to what factors cause banks to default. This is done in order to discover the variables that have the most explanatory power and to rank them. Simultaneously, the modeling approaches used to forecast bank insolvencies range. Messai and Gallali (2015) demonstrated that the ANN method outperformed the other models. By Ekinci and Halil, a comparison of artificial intelligence techniques was introduced. Until now in this paper, none of those studies applied Random Forests; instead, they compare the effectiveness of standard modeling techniques for forecasting bank failures, including such Logit and LDA, with more advanced systems, such as SVM and NN. The method described in this paper has a significant advantage over the majority of the literature because we include a wide range of explanatory variables whereas the assessment of the model uses statistical methods in mainly predicting bank distress. Models such as Random Forest, Logistic Regression, Support Vector Machine, Linear Discriminant Investigation, Artificial Neural Networks, and Random Forest of Conditional Inference Trees which are included in this research paper analysis. Over 660 additional variables are assessed based on their potential predictive value. By accomplishing this, they address a wide range of issues raised in the literature,

whether they are related to the final model that is used or the method of variable selection. To measure the effectiveness of each model, they also used a complete assessment approach. In doing so, they take into consideration out of sample and out of time validation samples in addition to various accuracy and discrimination tests. Finally, they expand the dataset that has been used in the great majority of earlier studies.

In [5], the performance of various early warning models for systemic bank failures as predicted outside of a sample is compared in this paper using advanced economies during the previous 45 years. They have compared three machine learning methods that have just been published in the literature with a standard logit technique. In this paper they also showed that even while machine learning techniques frequently achieve a very good in-sample fit, the logit approach outperforms them in recursive out of sample evaluations. This article suggests that more improvements to machine learning models are required before they can effectively enhance systemic banking crisis prediction. These models identify external imbalances, asset price booms, and credit expansions as the main indicators of systemic banking crises, which is consistent with economic intuition. From a predefined grid, they have chosen the best hyperparameters using cross validation. Cross-validation is a technique used to determine how effectively a model can predict outcomes from data that has not yet been observed. They are divided into estimated and test samples. Cross-validation is comparable to out-of-sample prediction in this regard, although it doesn't focus as much on the time dimension of the dataset. They employ panel block leave-p-out cross validation. In this variant, the training sample consists of all other observations besides the test sample, which is a block of twelve consecutive quarters from all nations. Until to the point when all sample observations have been collected, this is done repeatedly for every possible block. There are two benefits to using entire blocks of observations in the test sample as compared to randomly selecting some. As with recursive out-of-sample estimation, it first captures the data's serial and cross-sectional correlation. Second, the dataset's limited number of possible divisions allows for a thorough cross-validation of all available combinations. In this paper, early warning models for systemic banking crises were analyzed using data from 15 advanced nations between 1970 and 2016. They assess logit approach along with machine learning methods which performs in a quasi-real time forecasting experiment. The logit strategy turns out to be very difficult to overcome, typically resulting in fewer out of sample prediction errors than the machine learning techniques. This finding supports the view that a great in-sample fit of machine learning algorithms should not always be interpreted as a sign of a good prediction performance, as it could also be a symptom of overfitting. Additionally, compared to the logit model, it appears that these techniques perform more consistently across

configuration differences. These findings imply that one cannot automatically assume that machine learning techniques would perform well in out-of-sample prediction scenarios. Instead, more research is required to determine the conditions in which these approaches give glaring advantages as well as any potential adjustments that could enhance their stability and effectiveness in early warning applications. According to the data taken by this research paper, neural networks are the most promising machine learning technique. Therefore, future studies could investigate whether current developments in NN can be successfully used to foretell financial crises.

In [6], the banking industry's efficiency and depositor’s security have been questioned. Efficiency and performance analysis in the banking sector have become a popular topic as part of steps to strengthen the industry and win back customers' trust. This is necessary as stakeholders must identify the root causes of the banking sector's inefficiencies. The research has recommended nonparametric techniques like Data Envelopment Analysis (DEA) as a reliable indicator of banks' effectiveness and performance. The prediction of various nonparametric and nonlinear problems has also been recognized as a good application for machine learning techniques. This study used 444 Ghanaian bank branch’s samples and developed three machine learning models to assess bank performance (DMUs). The conclusions were contradicted with the corresponding DEA efficiency ratings. Finally, the three machine learning algorithm models' prediction ratings were analyzed. According to the findings, the decision tree and C5.0 algorithm offered the most accurate predictive model. It predicted the 134-holdout sample dataset (30% banks) with 100% accuracy and a P value of 0.00. The random forest algorithm, which had a predictive accuracy of 98.5% and a P value of 0.00, came in second place to the DT, and the neural network, which had an accuracy of 86.6% and a P value of 0.66, came in third. The best performing classification models were found in this study by using machine learning methods. The decision tree, random forest, and artificial neural network machine learning methods were used. We withheld a portion of the dataset for testing in order to assess the models' performance with actual data. As a result, the data set was divided into 30% for testing and 70% for training and validation. The test dataset was evaluated for the performance analysis. The CCR DEA's efficiency class classification is used as the response variable. Three well-known machine learning algorithms were used for predictive models, and they were contrasted against one another using several performance indicators. 70% of the bank's branches dataset from Ghana's commercial banks were used in this study to train and validate each of the three models. The effectiveness of the remaining 30% of the bank branches was predicted using the proposed models. Using a holdout sample data set, the machine learning algorithm models that performed the best were

identified. This research showed that most Ghanaian banks (a total of 369, or 83.1%), used the "cutoff point" effectively to run their overall banking operations. Even though many of these banks had effective overall banking operations, they had ineffective deposit collection efficiency (47 banks represented 10.59%) and particularly inefficient deposit investment efficiency (only 2 banks represented 0.45%). Therefore, the report advises Ghanaian bank managers and other stakeholders to re-evaluate their effectiveness and performance in procuring deposits and investing the deposits. This implies that managers and other interested parties shouldn't solely or excessively rely on their own overall efficacy. The study concluded that banks in Ghana might utilize the findings to forecast their own efficiencies.

3. Materials and Methods

3.1 Information about models

There are three target variables namely b_tgt, int_tgt, cnt_tgt and we have trained a model for each target variable. The generalized algorithm to load, analyze, preprocess, train and evaluate the models followed in this project is as follows:

Step 1: Load the dataset into memory
Step 2: Perform exploratory data analysis to understand the structure and distribution of the data
Step 3: Preprocess the data, including cleaning and transformation of the data to prepare it for analysis
Step 4: Check for null values and determine the best course of action for handling them
Step 5: Check for correlation between variables and remove highly correlated variables
Step 6: Use data visualization techniques to gain insights and identify patterns in the data
Step 7: Identify outliers in the data and decide whether to remove keep them since they might have logical meaning
Step 8: Check for skewness in the data and normalize the data using log or square root transformation
Step 9: Split the data into train and test dataset
Step 10: Train the model using the selected data using default parameters (Logistic Regression, Decision Tree and Random Forest for b_tgt; Multivariate Regression for int_tgt; Multivariate Regression and Poisson Regression for cnt_tgt)
Step 11: Perform hyperparameter tuning using RandomizedSearchCV to optimize the model

Step 12: Validate and test the model using additional data to evaluate its performance and accuracy based on evaluation metrics

- Algorithm for Decision Tree is as follows:

Step 1: Begin the tree with the root node, says S, which contains the complete dataset.
Step 2: Find the best attribute in the dataset using Attribute Selection Measure.
Step 3: Divide the S into subsets that contain possible values for the best attributes.
Step 4: Generate the decision tree node, which contains the best attribute.
Step 5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and call the final node as a leaf node.

- Algorithm for Random Forest is as follows:

Step 1: Select random K data points from the training set.
Step 2: Build the decision trees associated with the selected data points (Subsets).
Step 3: Choose the number N for decision trees that you want to build.
Step 4: Repeat Step 1 & 2.

3.2 Dataset Description

The dataset for this project is taken from a financial company, basically records of account holders. The dataset has a total of 1060038 observations. It has 26 columns which describe different details of consumers of the firm. The target variables are B_TGT, INT_TGT and CNT_TGT. The description of each variable in the dataset is given below. The necessary data preprocessing is done on the dataset for better results.

- Categorical Inputs:

CAT_INPUT1	Account Activity Level
CAT_INPUT2	Customer Value Level

- Interval Inputs:

RFM1	Average Sales Past Three Years
RFM2	Average Sales Lifetime

RFM3	Average Sales Past Three Years Dir Promo Resp
RFM4	Last Product Purchase Amount
RFM5	Count Purchased Past Three Years
RFM6	Count Purchased Lifetime
RFM7	Count Purchased Past Three Years Dir Promo Resp
RFM8	Count Purchased Lifetime Dir Promo Resp
RFM9	Months Since Last Purchase
RFM10	Count Total Promos Past Year

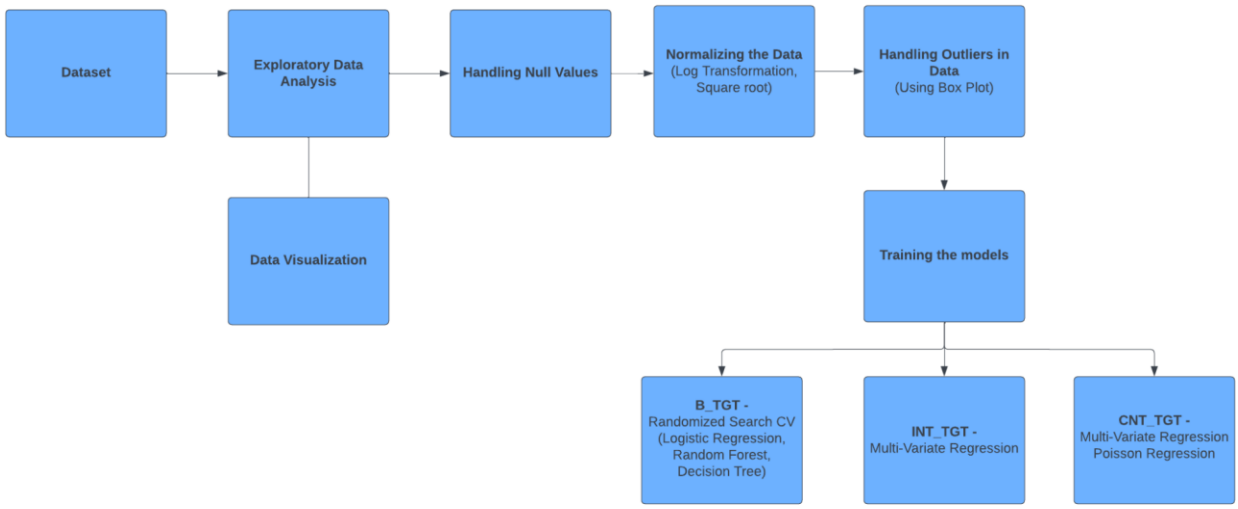
- Demographic Inputs:

DEMOG_AGE	Customer Age
DEMOG_GENF	Female Binary (Yes/No)
DEMOG_GENM	Male Binary (Yes/No)
DEMOG_HO	Homeowner Binary (Yes/No)
DEMOG_HOMEVAL	Home Value
DEMOG_INC	Income
DEMOG_PR	Percentage Retired in The Area

- Target variables:

B_TGT	New Product (Binary)(yes/no)
INT_TGT	New Sales (Interval)
CNT_TGT	Count Number New Products

3.3 Architecture and Explanation



For Business Marketing Efficiency Analysis and Prediction, we took a dataset which is related to the bank and business. We did the necessary data analysis and visualization for training the models. The training of models is divided into categories as shown in the architecture. The target values we have taken are used for analysis and prediction. As our target variables are categorical, we have applied fine tuning and hyper parameters using RandomizedSearchCV.

4. Proposed Work:

Whether a new product will be bought or not is represented by the b_tgt variable and it can be predicted with the use of logistic regression, random forest, and decision tree algorithms. By using RandomizedSearchCV we can do hyper parameter tuning to refine the model. After predicting the purchase of a new product, we propose 2 more models to predict how many products a customer will purchase, which is the cnt_tgt variable. This can be done with the use of

multivariate regression and Poisson regression. The final model is again multivariate regression which is used to predict how much amount a customer will spend on a new product.

4.1 Novelty

Predicting and analyzing the efficiency of business marketing efforts using machine learning is a relatively novel approach, and it has significant potential to improve marketing strategies by targeting necessary customers. From the literature survey most prediction models only predict a categorical variable of whether a customer will buy a product or not. They do not provide more information on the purchase which is essential for coming up with a marketing strategy that will increase business profits. While predicting, we are not just predicting if a customer will buy a product or not (b_tgt) but also how many products that customer will buy (cnt_tgt) and how much he will spend on new products (int_tgt).

4.2 Project Contributions

Registration Number	Name	Contribution
20BCE1223	Manigandan R	Preprocessing & Exploratory Data Analysis
20BCE1922	Rishi Nair	Exploratory Data Analysis & Data preprocessing
20BCE1138	Janapati Sarath Chandra	Preprocessing & Training

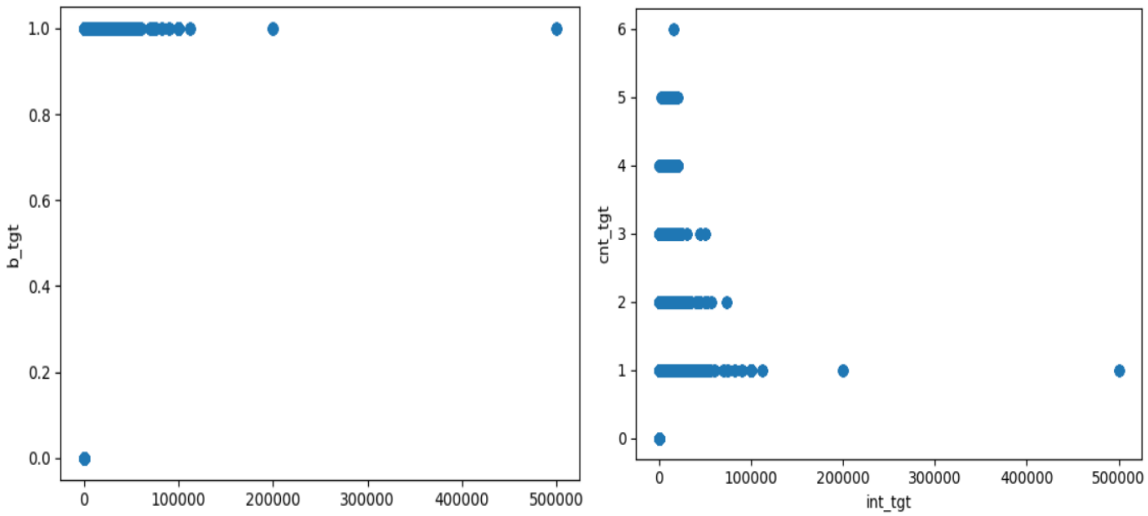
5. Results and Discussion:

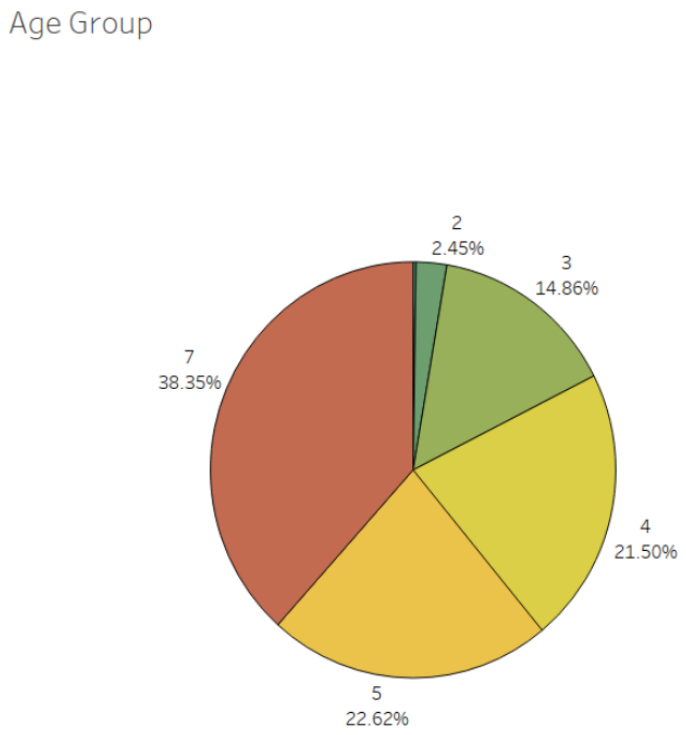
5.1 Results

For b_tgt variable we concluded that Random Forest performs better with 0.99 recall score. For the int_tgt variable, multivariate regression performs better because of less root mean square error of 38.4. Finally, for the cnt_tgt variable we also got multivariate regression as the best model with less root mean square error of 0.58 but we can use the Poisson model also since the error rate is not significant here.

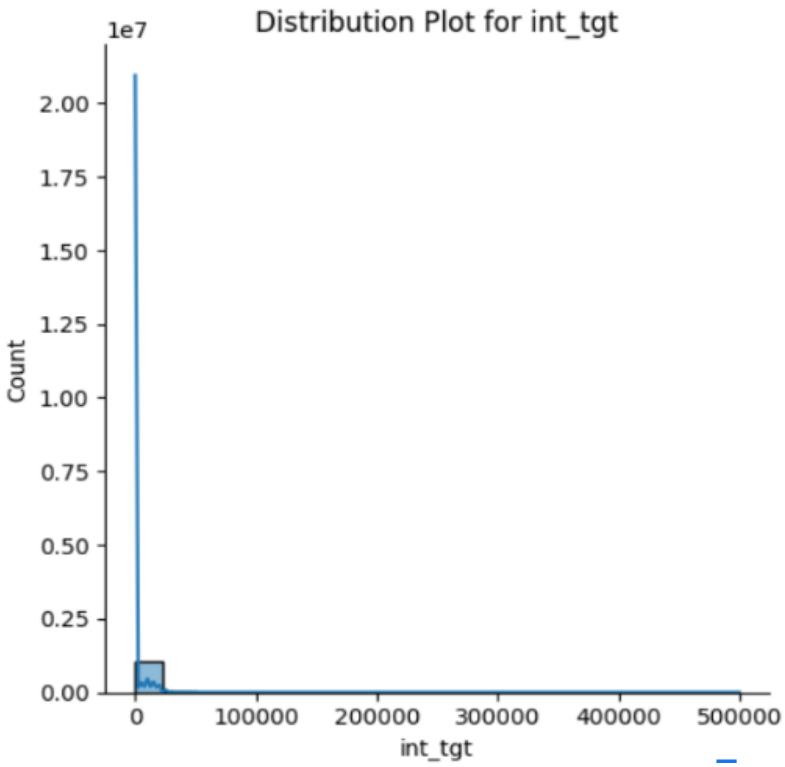
5.2 Figures

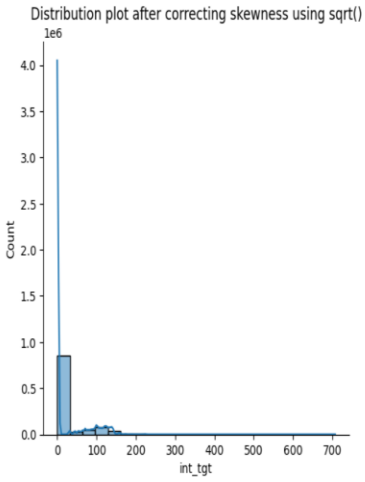
- Exploratory Data Analysis:

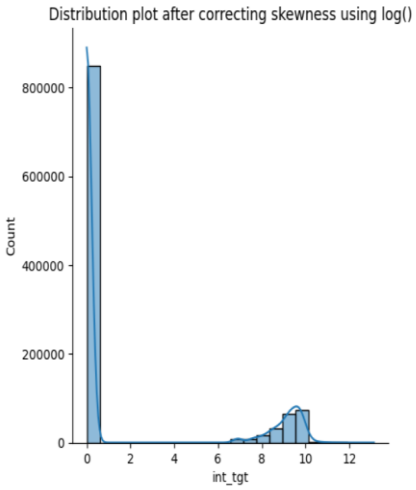


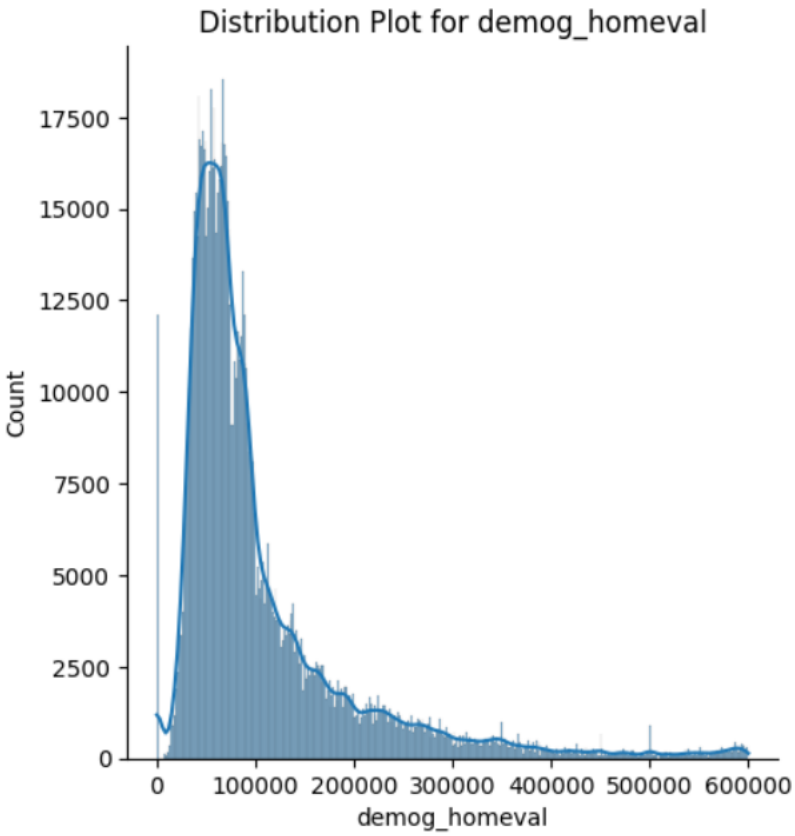


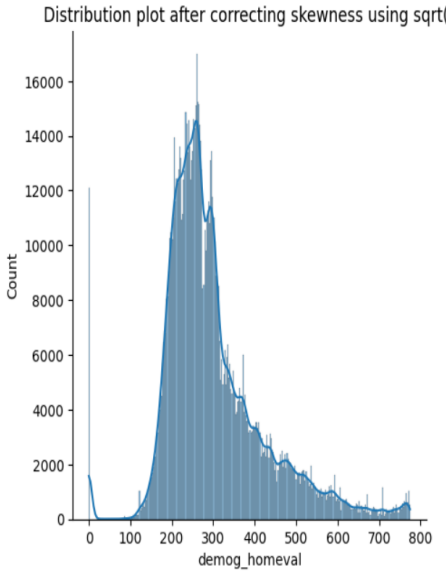
- Normalizing the Data:

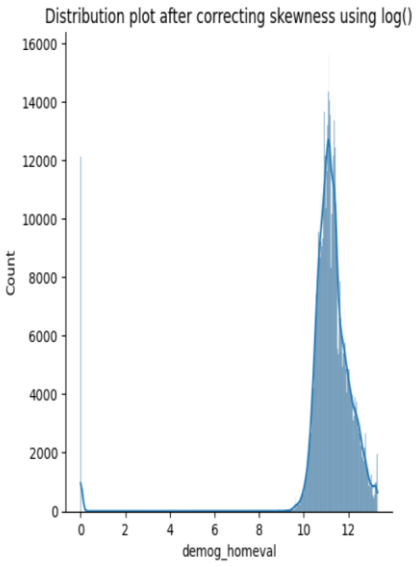

Distribution Plot for int_tgt
1e7
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Count
0
100000
200000
300000
400000
500000
int_tgt


Distribution plot after correcting skewness using sqrt()
1e6
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Count
0
100
200
300
400
500
600
700
int_tgt

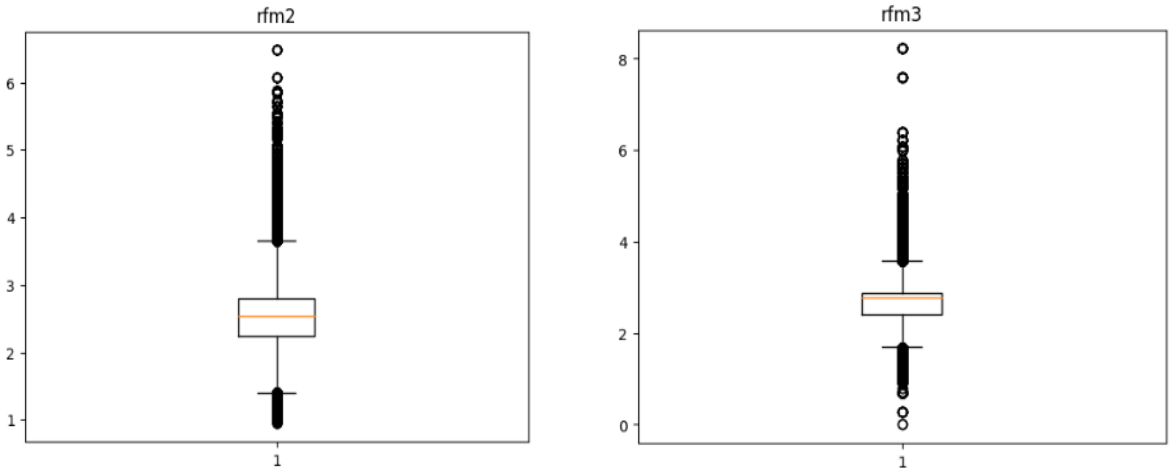

Distribution plot after correcting skewness using log()
800000
600000
400000
200000
0
Count
0
2
4
6
8
10
12
int_tgt


Distribution Plot for demog_homeval
Count
demog_homeval
0
2500
5000
7500
10000
12500
15000
17500
0
100000
200000
300000
400000
500000
600000

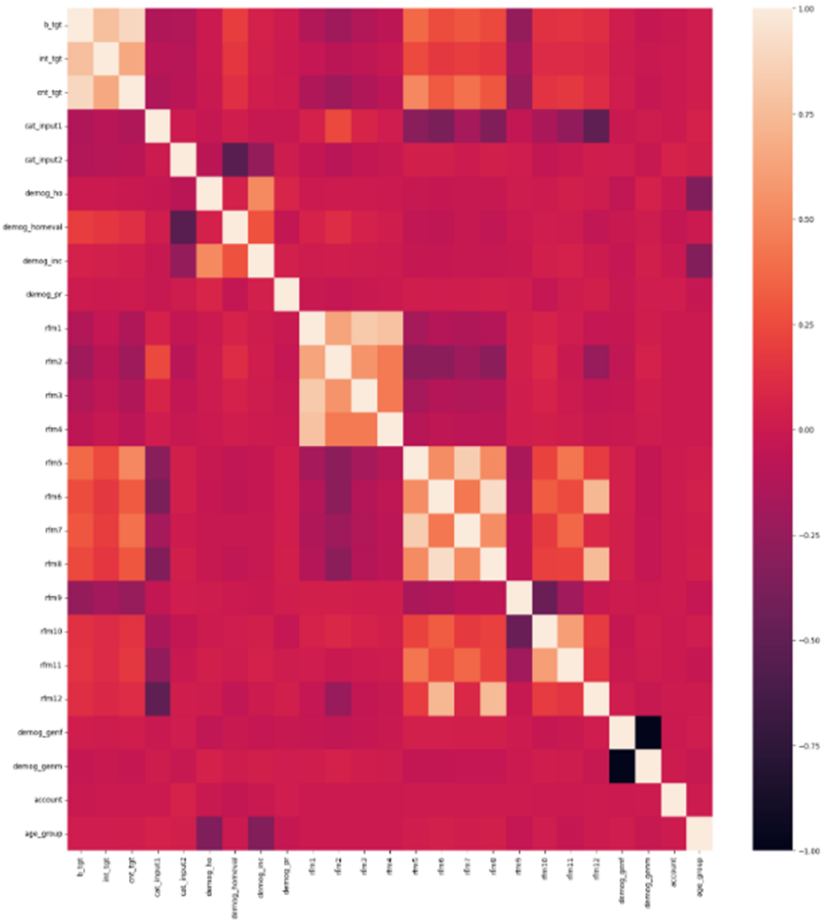

Distribution plot after correcting skewness using sqrt()
Count
demog_homeval
0
2000
4000
6000
8000
10000
12000
14000
16000
0
100
200
300
400
500
600
700
800


Distribution plot after correcting skewness using log()
Count
demog_homeval
0
2000
4000
6000
8000
10000
12000
14000
16000
0
2
4
6
8
10
12

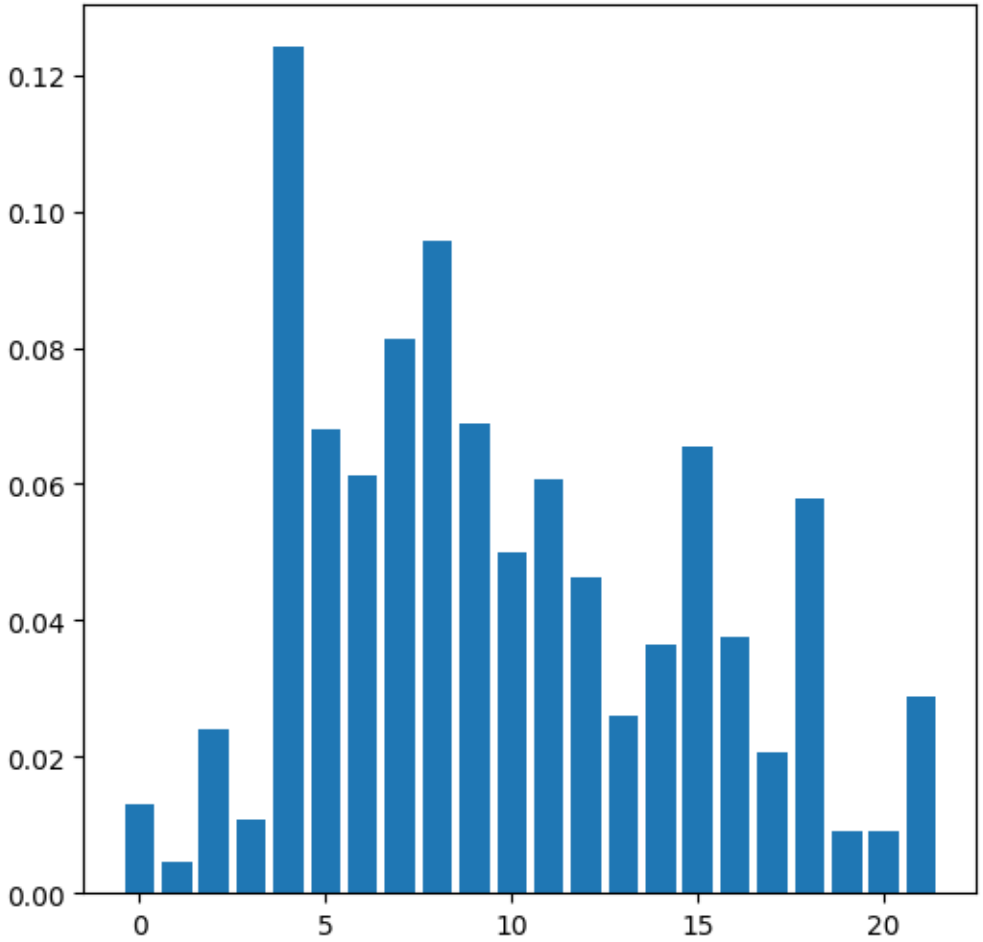
- Outlier Removal:



- Heatmap:



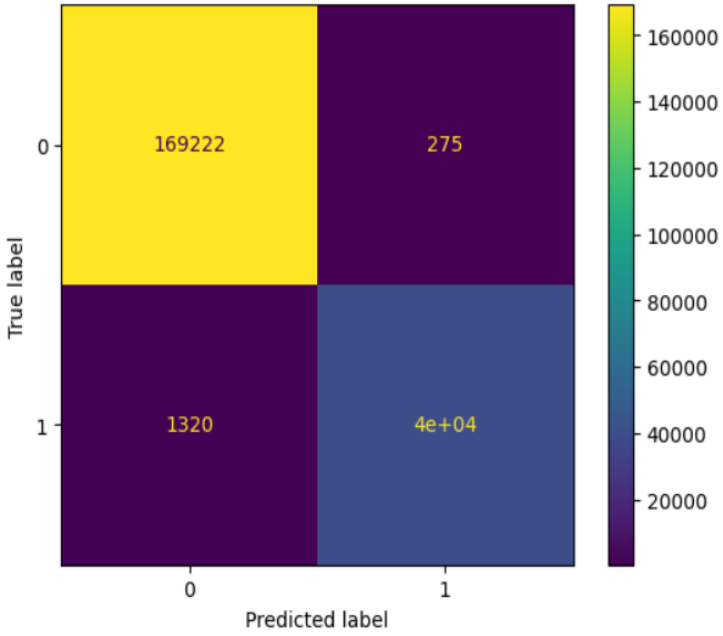
- Feature Selection:



- Comparison Table:

S. No.	Model	Training Score	Best Parameters	Accuracy
1	Logistic Regression	85%	Solver: newton-cg, multiclass: ovr	85%
2	Random Forest	99%	N estimators: 300, max features: log2	99%
3	Decision Tree	98%	splitter: best, max features: auto	98%

➢ Confusion matrix For Random Forest:



5.3 Explanation

Exploratory Data Analysis: int_tgt we have 848529 null values in target variable. So, we assume that customers have not spent any money in purchasing products so we substitute by 0 but because of this we get logical error. We found that the samples that has total sales (int_tgt) as 0 but they have bought a new product (b_tgt = 1) and count of new product (cnt_tgt) is greater than 1. This is ambiguous. So we are finding those variables that have total sales (int_tgt) as 0 but they have bought a new product (b_tgt = 1) and count of new product (cnt_tgt) is greater than 1 and dropping those samples. Along with ths we plot pie chart for age group to visualize the composition.

Normalizing Data: If a customer's record has that they have not purchased any product then we can change null values in the int_tgt column to 0. Any customer with null value in cnt_tgt is removed. There are also other records that need to be removed like account holders that have b_tgt as "1" but have a null value for int_tgt. Average of the column "Average Sales Past Three Years Dir Promo Resp" can be used to instead of null values. After these removals, the dataset has reduced the number of records. After data cleaning, the data selection is crucial for training the models. So, we are removing two variables account and dataset as they are less required other than remaining variables. After that we did data normalization because of the range of observations. We did either square root or log transformation as the variables are right skewed.

Outlier Removal: We plot box plots, and we use Inter Quartile Range of 10% and 90% to remove outliers.

Feature selection: In order to improve the accuracy of the models, we performed feature selection. The graph visualizes the importance of all the features in the dataset.

Comparison Table: Comparison of all 3 models (Random Forest, Decision tree, Logistic Regression) for B_TGT variable with best parameters and accuracy scores. It can be inferred that Random Forest gives the best accuracy score of 0.99.

6. Conclusion

We did analysis on 3 different classification models to predict the variable “if a customer will buy a product”. More models like multivariate regression are used to predict how much amount will be spent to buy a new product. Similarly, for the last target variable i.e., “how many products will the customer buy” we are using multivariate regression and Poisson regression. These target variables provide key information to business organizations which can be used to fine tune marketing strategies which will generate more profits for the organizations and also avoid the chances of those organizations going bankrupt. This is also useful to reach customers with the products they would want and hence it improves customer experience.

GitHub Link:
https://github.com/NotManigandan/Business-Marketing-Efficiency-Analysis-and-Prediction.git

Dataset Link:
https://drive.google.com/file/d/1w4QwZNLrT2FcpiLKQN1fe3T7cLuCPUgm/view?usp=sharing

7. References

1. Droomer, M., & Bekker, J. (2020, November). USING MACHINE LEARNING TO PREDICT THE NEXT PURCHASE DATE FOR AN INDIVIDUAL RETAIL CUSTOMER. South African Journal of Industrial Engineering,31(3). https://doi.org/10.7166/31-3-2419

2. Huo, Z. (2021, March). Sales Prediction based on Machine Learning. 2021 2nd International Conference on E-Commerce and Internet Technology (ECIT). https://doi.org/10.1109/ecit52743.2021.00093

3. Gupta, R., & Pathak, C. (2014). A Machine Learning Framework for Predicting Purchase by Online Customers based on Dynamic Pricing. Procedia Computer Science, 36, 599–605.https://doi.org/10.1016/j.procs.2014.09.060

4. Petropoulos, A., Siakoulis, V., Stavroulakis,E.and Vlachogiannakis, N.E.,2020.Predicting bank insolvencies using machine learning techniques. International Journal of Forecasting, 36(3), pp.1092-1113.

5. Beutel, J., List, S. and von Schweinitz, G., 2019. Does machine learning help us predict banking crises?. Journal of Financial Stability, 45,p.100693.

6. Appiahene, P., Missah, Y.M. and Najim, U., 2020. Predicting bank operational efficiency using machine learning algorithm: comparative study of decision tree, random forest, and neural networks. Advances in fuzzy systems, 2020.

8. Appendix

In [8]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from scipy.stats import skew
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.linear_model import LogisticRegression
from sklearn import svm
from sklearn.model_selection import RandomizedSearchCV
import lazypredict
from lazypredict.Supervised import LazyClassifier
import tensorflow as tf
import keras
from tensorflow.keras import Model, Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.losses import MeanSquaredLogarithmicError
from keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.optimizers import Adam
from sklearn.linear_model import LinearRegression
from sklearn import linear_model
import sklearn.metrics as metrics
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.tree import export_graphviz
import six
import sys
from six import StringIO
import pickle
```

Importing the dataset

In [3]:
```
filePath = "C:\\Users\hiooy\\Documents\\TARP\\bankdataset.csv"
originalDF = pd.read_csv(filePath)
originalDF.head()
```

Out[3]:

	Unnamed: 0	b_tgt	int_tgt	cnt_tgt	cat_input1	cat_input2	demog_age	demog_ho	demog_homeval	demog_inc	...	rfm7	rfm8	rfm9	rfm10	rfm11	rfm12
0	1	1	7000.00	NaN	X	A	NaN	0	57600.00	52106.00	...	4	6	5	20	9	92
1	2	1	7000.00	2.00	X	A	NaN	0	57587.00	52106.00	...	4	6	5	20	9	92
2	3	1	15000.00	2.00	X	A	NaN	0	44167.00	42422.00	...	3	8	16	27	11	91
3	4	0	NaN	0.00	X	A	68.00	0	90587.00	59785.00	...	2	7	15	19	9	123
4	5	0	NaN	0.00	X	A	NaN	0	100313.00	0.00	...	5	19	24	13	6	128

5 rows × 27 columns

In [4]:
```
originalDF.shape
```

Out[4]:
```
(1060038, 27)
```

The original bank dataset contains 1060038 samples and 27 attributes.

In [4]:
```
originalDF.drop(['Unnamed: 0'], axis=1, inplace=True)
originalDF.drop(['dataset'], axis=1, inplace=True)
```

Removed the first attribute and another attribute named dataset because it just displays the serial number which is not useful for our model.

Handling Null Values

In [5]:
```
originalDF.isnull().sum()
```

Out[5]:
```
b_tgt               0
int_tgt        848529
cnt_tgt             1
cat_input1          0
cat_input2          0
demog_age      266861
demog_ho            0
demog_homeval       0
demog_inc           0
demog_pr            0
rfm1                0
rfm2                0
rfm3           225786
rfm4                0
rfm5                0
rfm6                0
rfm7                0
rfm8                0
rfm9                0
rfm10               0
rfm11               0
rfm12               0
demog_genf          0
demog_genm          0
account             0
dtype: int64
```

We can see that int_tgt, cnt_tgt, demog_age and rfm3 attributes have null values.

In [6]:
```
originalDF.dtypes
```

Out[6]:
```
b_tgt             int64
int_tgt         float64
cnt_tgt         float64
cat_input1       object
cat_input2       object
demog_age       float64
demog_ho          int64
demog_homeval   float64
demog_inc       float64
demog_pr          int64
rfm1            float64
rfm2            float64
rfm3            float64
rfm4            float64
rfm5              int64
rfm6              int64
rfm7              int64
rfm8              int64
rfm9              int64
rfm10             int64
rfm11             int64
rfm12             int64
demog_genf        int64
demog_genm        int64
account         float64
dtype: object
```

All the columns that have null values are of float64 datatype.

In [7]:
```
originalDF.dropna(axis=0, subset=['cnt_tgt'], inplace=True)
```

Removing the entire row which had a null value in cnt_tgt.

In [8]:
```
originalDF['int_tgt'] = originalDF['int_tgt'].replace(np.nan, 0)
```

Replacing null values in int_tgt by 0.

In [9]:
```
meanrfm3 = round(originalDF['rfm3'].mean(),2)
originalDF['rfm3'] = originalDF['rfm3'].replace(np.nan, meanrfm3)
```

Replacing null values in rfm3 by mean value of the column.

In [10]:
```
originalDF['demog_age'].unique()
```

Out[10]:
```
array([nan, 68., 26., 74., 83., 70., 77., 81., 61., 55., 84., 76., 63.,
       41., 58., 73., 79., 35., 44., 67., 45., 16., 43., 52., 80., 71.,
       46., 85., 51.,  8., 40., 87., 49., 34., 27., 88., 82., 59., 69.,
       60., 65., 18., 54., 53., 78., 66., 86., 72., 75., 38., 42., 57.,
       17., 48., 47., 62., 56., 50.,  7., 37., 64., 32., 36., 33., 31.,
       29., 39., 30., 28., 25., 21., 23.,  6.,  5., 24., 22., 15.,  9.,
       19., 20.,  1., 11.,  0., 14., 10.,  2., 12.,  4.,  3., -1., 89.,
       13.])
```

We can see that there are too many unique age values in demog_age feature ranging from -1 to 89 which also includes null values. So, we need to bin or group it and make it ordinal values.

Exploratory Data Analysis

In [13]:
```
salesZeroDF = originalDF[(originalDF['b_tgt']==1) & (originalDF['cnt_tgt']>=1) & (originalDF['int_tgt']==0)]
salesZeroDF.shape
```

Out[13]:
```
(3916, 25)
```

Finding the samples that has total sales (int_tgt) as 0 but they have bought a new product (b_tgt = 1) and count of new product (cnt_tgt) is greater than 1.

In [14]:
```
originalDF = pd.concat([originalDF, salesZeroDF]).drop_duplicates(keep=False)
originalDF.shape
```

Out[14]:
```
(1056121, 25)
```

Removing the samples found in the previous code cell.

In [11]:
```
age_group = []
for age in originalDF['demog_age']:
    if age <= 17.0:
        age_group.append(1)
    elif age > 17.0 and age <= 35.0:
        age_group.append(2)
    elif age > 35.0 and age <= 53.0:
        age_group.append(3)
    elif age > 53.0 and age <= 71.0:
        age_group.append(4)
    elif age > 71.0 and age <= 89.0:
        age_group.append(5)
    else:
        age_group.append(7)
originalDF['age_group'] = age_group
originalDF.drop(['demog_age'], axis=1, inplace=True)
```

In [12]:
```
le = LabelEncoder()
originalDF.cat_input1 = le.fit_transform(originalDF.cat_input1)
originalDF.cat_input2 = le.fit_transform(originalDF.cat_input2)
```

cat_input1 and cat_input2 has text values. So we use LabelEncoder to encode it to numerical values.

In [15]:
```
def expand_categories(values):
    result = []
    s = values.value_counts()
    t = float(len(values))
    for v in s.index:
        result.append("{}: {}%".format(v,round(100*(s[v]/t),2)))
    return "[{}]".format(", ".join(result))

def analyze(df):
    cols = df.columns.values
    total = float(len(df))
    for col in cols:
        uniques = df[col].unique()
        unique_count = len(uniques)
        if unique_count>100:
            print("--> {}:{} ({}%)".format(col,unique_count,int(((unique_count)/total)*100)))
        else:
            print("--> {}:{}".format(col,expand_categories(df[col])))
            expand_categories(df[col])
```

We are analyzing the data by looking at the unique values present in the dataset by calling analyze() function. We have displayed the percentage of the unique value in the column if the number of unique values is less than 100 to save display space. This is done by the expand_categories() function.

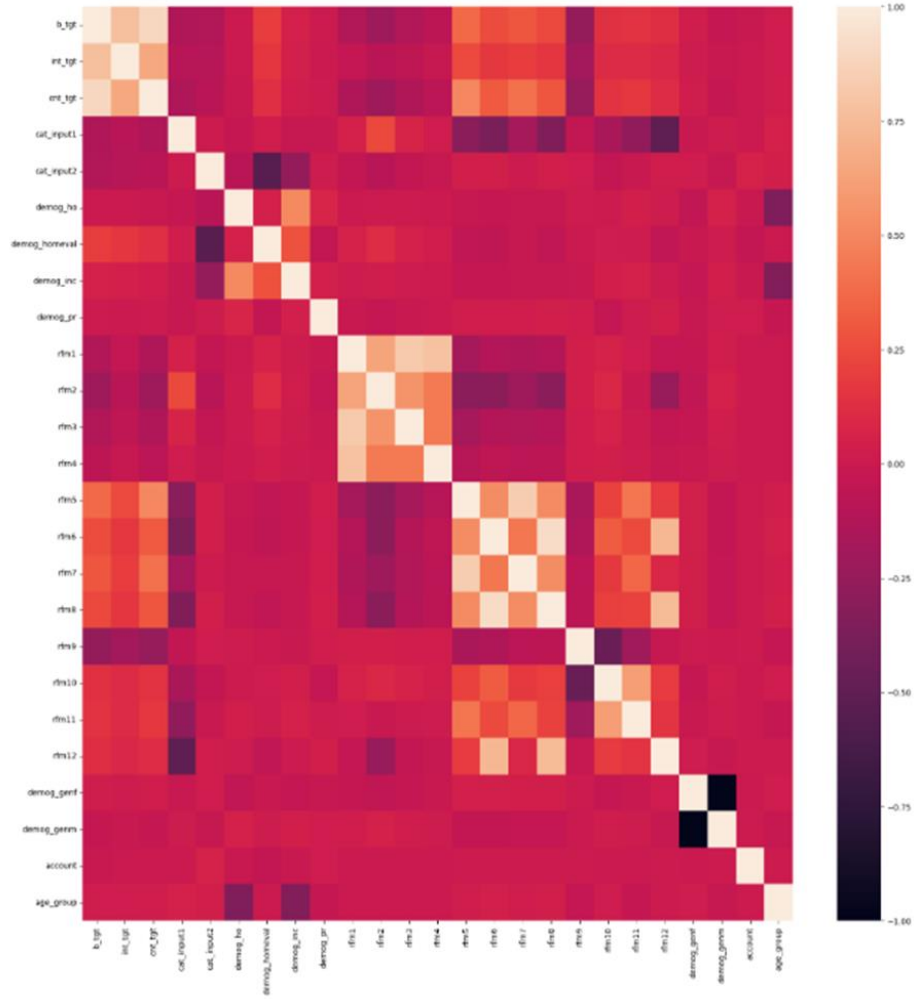
In [36]:
```
analyze(originalDF)
```

```
--> b_tgt:[0: 80.34%, 1: 19.66%]
--> int_tgt:[0.0: 80.34%, 10000.0: 1.71%, 15000.0: 1.32%, 5000.0: 1.2%, 20000.0: 0.96%, 12000.0: 0.94%, 11000.0: 0.92%, 7000.0:
0.91%, 8000.0: 0.89%, 6000.0: 0.88%, 9000.0: 0.87%, 14000.0: 0.87%, 16000.0: 0.87%, 19000.0: 0.86%, 13000.0: 0.83%, 3000.0: 0.8
3%, 17000.0: 0.83%, 4000.0: 0.82%, 18000.0: 0.81%, 1000.0: 0.79%, 2000.0: 0.76%, 25000.0: 0.35%, 21000.0: 0.09%, 30000.0: 0.0
7%, 50000.0: 0.05%, 40000.0: 0.03%, 22000.0: 0.02%, 35000.0: 0.02%, 100000.0: 0.02%, 26000.0: 0.02%, 12500.0: 0.01%, 24000.0:
0.01%, 23000.0: 0.01%, 27000.0: 0.01%, 36000.0: 0.01%, 31000.0: 0.01%, 45000.0: 0.01%, 7500.0: 0.01%, 33000.0: 0.0%, 28000.0:
0.0%, 37000.0: 0.0%, 2500.0: 0.0%, 52000.0: 0.0%, 38000.0: 0.0%, 200000.0: 0.0%, 70000.0: 0.0%, 51000.0: 0.0%, 39000.0: 0.0%, 3
4000.0: 0.0%, 32000.0: 0.0%, 60000.0: 0.0%, 29000.0: 0.0%, 75000.0: 0.0%, 42000.0: 0.0%, 320.0: 0.0%, 54000.0: 0.0%, 22500.0:
0.0%, 9370.0: 0.0%, 5450.0: 0.0%, 26500.0: 0.0%, 82000.0: 0.0%, 112000.0: 0.0%, 44000.0: 0.0%, 47000.0: 0.0%, 56000.0: 0.0%, 90
000.0: 0.0%, 57000.0: 0.0%, 55000.0: 0.0%, 500000.0: 0.0%, 73000.0: 0.0%]
--> cnt_tgt:[0.0: 80.34%, 1.0: 10.78%, 2.0: 7.01%, 3.0: 1.6%, 4.0: 0.24%, 5.0: 0.03%, 6.0: 0.0%]
--> cat_input1:[0: 78.39%, 2: 14.27%, 1: 7.35%]
--> cat_input2:[4: 36.59%, 1: 18.14%, 0: 17.74%, 2: 15.99%, 3: 11.55%]
--> demog_ho:[1: 55.04%, 0: 44.96%]
--> demog_homeval:226112 (21%)
--> demog_inc:48867 (4%)
--> demog_pr:[31: 4.51%, 30: 4.49%, 32: 4.34%, 29: 4.3%, 33: 4.19%, 28: 4.13%, 34: 4.02%, 27: 3.85%, 35: 3.78%, 26: 3.65%, 36:
3.42%, 25: 3.31%, 37: 3.23%, 24: 2.97%, 0: 2.95%, 38: 2.91%, 23: 2.67%, 39: 2.54%, 22: 2.34%, 40: 2.29%, 41: 2.11%, 21: 1.98%,
42: 1.86%, 20: 1.78%, 43: 1.6%, 19: 1.54%, 44: 1.37%, 18: 1.3%, 45: 1.24%, 17: 1.12%, 46: 1.12%, 16: 0.96%, 47: 0.94%, 48: 0.8
2%, 15: 0.8%, 49: 0.73%, 14: 0.71%, 50: 0.65%, 1: 0.57%, 13: 0.56%, 51: 0.54%, 12: 0.45%, 52: 0.45%, 11: 0.39%, 53: 0.37%, 10:
0.33%, 54: 0.32%, 9: 0.29%, 55: 0.28%, 56: 0.27%, 8: 0.25%, 7: 0.24%, 57: 0.24%, 6: 0.2%, 58: 0.18%, 59: 0.15%, 5: 0.13%, 61:
0.13%, 60: 0.13%, 4: 0.12%, 62: 0.1%, 63: 0.09%, 3: 0.08%, 65: 0.08%, 64: 0.07%, 66: 0.07%, 2: 0.06%, 67: 0.04%, 70: 0.04%, 69:
0.04%, 68: 0.03%, 77: 0.03%, 71: 0.03%, 73: 0.02%, 76: 0.02%, 78: 0.01%, 72: 0.01%, 79: 0.01%, 74: 0.01%, 99: 0.01%, 75: 0.01%,
85: 0.01%, 80: 0.0%, 82: 0.0%, 84: 0.0%, 83: 0.0%, 86: 0.0%, 98: 0.0%, 100: 0.0%, 97: 0.0%, 91: 0.0%, 89: 0.0%, 90: 0.0%, 81:
0.0%, 87: 0.0%, 96: 0.0%, 101: 0.0%, 88: 0.0%, 92: 0.0%]
--> rfm1:1461 (0%)
--> rfm2:3152 (0%)
--> rfm3:869 (0%)
--> rfm4:272 (0%)
--> rfm5:[2: 24.93%, 1: 21.19%, 3: 18.23%, 4: 12.2%, 5: 7.69%, 0: 5.04%, 6: 4.7%, 7: 2.71%, 8: 1.52%, 9: 0.87%, 10: 0.48%, 11:
```

```
--> rfm5:[2: 24.93%, 1: 21.19%, 3: 18.23%, 4: 12.2%, 5: 7.69%, 0: 5.04%, 6: 4.7%, 7: 2.71%, 8: 1.52%, 9: 0.87%, 10: 0.48%, 11:
0.23%, 12: 0.12%, 13: 0.06%, 14: 0.03%, 15: 0.01%, 16: 0.0%, 17: 0.0%, 18: 0.0%]
--> rfm6:108 (0%)
--> rfm7:[1: 30.97%, 0: 23.6%, 2: 21.98%, 3: 12.02%, 4: 5.96%, 5: 2.93%, 6: 1.47%, 7: 0.7%, 8: 0.28%, 9: 0.08%, 10: 0.01%, 11:
0.0%]
--> rfm8:[1: 15.71%, 2: 14.32%, 3: 11.13%, 4: 8.89%, 5: 7.37%, 0: 7.28%, 6: 6.34%, 7: 5.32%, 8: 4.5%, 9: 3.79%, 10: 3.12%, 11:
2.64%, 12: 2.14%, 13: 1.71%, 14: 1.29%, 15: 1.06%, 16: 0.78%, 17: 0.61%, 18: 0.47%, 19: 0.38%, 20: 0.3%, 21: 0.24%, 22: 0.18%,
23: 0.14%, 24: 0.11%, 25: 0.07%, 26: 0.05%, 27: 0.03%, 28: 0.02%, 29: 0.01%, 30: 0.01%, 34: 0.0%, 31: 0.0%, 35: 0.0%, 33: 0.0%,
45: 0.0%, 32: 0.0%, 46: 0.0%, 36: 0.0%]
--> rfm9:[18: 17.63%, 17: 13.82%, 16: 11.64%, 19: 9.97%, 15: 9.21%, 20: 6.35%, 21: 6.07%, 24: 3.99%, 26: 3.7%, 22: 3.6%, 23: 3.
28%, 25: 3.22%, 14: 1.88%, 27: 1.08%, 5: 0.84%, 4: 0.77%, 6: 0.73%, 7: 0.51%, 8: 0.39%, 9: 0.28%, 13: 0.24%, 10: 0.24%, 11: 0.1
7%, 12: 0.16%, 3: 0.15%, 28: 0.08%, 2: 0.0%, 29: 0.0%]
--> rfm10:[13: 22.25%, 12: 18.93%, 11: 12.4%, 14: 11.34%, 10: 9.66%, 9: 4.88%, 15: 4.26%, 8: 3.0%, 7: 2.02%, 16: 1.47%, 6: 1.0
3%, 17: 0.63%, 5: 0.52%, 26: 0.47%, 27: 0.46%, 29: 0.45%, 28: 0.45%, 18: 0.44%, 19: 0.43%, 20: 0.41%, 23: 0.41%, 25: 0.41%, 21:
0.41%, 24: 0.41%, 30: 0.39%, 22: 0.39%, 4: 0.36%, 31: 0.32%, 32: 0.27%, 33: 0.22%, 34: 0.19%, 35: 0.14%, 36: 0.11%, 3: 0.09%, 3
7: 0.08%, 38: 0.07%, 39: 0.06%, 40: 0.04%, 41: 0.03%, 42: 0.02%, 43: 0.02%, 44: 0.02%, 2: 0.01%, 45: 0.01%, 47: 0.01%, 46: 0.0
1%, 48: 0.0%, 52: 0.0%, 1: 0.0%, 49: 0.0%, 56: 0.0%, 51: 0.0%, 58: 0.0%, 50: 0.0%, 57: 0.0%, 60: 0.0%, 59: 0.0%, 53: 0.0%, 55:
0.0%, 76: 0.0%, 61: 0.0%, 68: 0.0%, 67: 0.0%, 0: 0.0%, 77: 0.0%, 69: 0.0%, 54: 0.0%, 75: 0.0%, 62: 0.0%]
--> rfm11:[6: 39.69%, 5: 24.35%, 4: 15.71%, 7: 9.61%, 3: 5.77%, 2: 1.74%, 8: 1.04%, 9: 0.53%, 1: 0.45%, 11: 0.44%, 10: 0.4%, 1
2: 0.13%, 0: 0.07%, 13: 0.03%, 14: 0.01%, 16: 0.01%, 15: 0.01%, 17: 0.0%, 18: 0.0%, 20: 0.0%, 21: 0.0%, 19: 0.0%, 22: 0.0%]
--> rfm12:215 (0%)
--> demog_genf:[1: 56.21%, 0: 43.79%]
--> demog_genm:[0: 56.21%, 1: 43.79%]
--> account:1056121 (100%)
--> age_group:[7: 25.16%, 4: 24.69%, 3: 22.74%, 5: 20.78%, 2: 5.63%, 1: 1.0%]
```

In [20]:
```
plt.subplots(figsize=(20,20))
sns.heatmap(originalDF.corr())
```

Out[20]: <AxesSubplot: >



We can see that target variables are correlated but we cant remove them. Some of the variables have correlation but they are meaningful features so we cant remove these features also. Other variables have less correlation so we can ignore those features

In [21]:
```
originalDF.skew(axis=0)
```

Out[21]:
```
b_tgt            1.53
int_tgt         10.57
cnt_tgt          2.43
cat_input1       1.66
cat_input2      -0.22
demog_ho        -0.20
demog_homeval    2.46
demog_inc        0.23
demog_pr        -0.15
rfm1           103.26
rfm2            11.13
rfm3           129.53
rfm4           206.88
rfm5             1.23
rfm6             1.91
rfm7             1.23
rfm8             1.42
rfm9            -0.60
rfm10            2.86
rfm11            0.32
rfm12            0.30
demog_genf      -0.25
demog_genm       0.25
account         -0.00
age_group        0.26
dtype: float64
```

We can see that the data is not normally distributed and it is skewed. So we take either square root or log tranformation based on which normalizes the data better.

In [31]:
```
Q1 = originalDF.rfm12.quantile(0.1)
Q3 = originalDF.rfm12.quantile(0.9)
IQR = Q3 - Q1
lower_limit = Q1 - 1.5*IQR
upper_limit = Q3 + 1.5*IQR
originalDF[(originalDF.rfm12<lower_limit)|(originalDF.rfm12>upper_limit)]
```

Out[31]:

	b_tgt	int_tgt	cnt_tgt	cat_input1	cat_input2	demog_ho	demog_homeval	demog_inc	demog_pr	rfm1	...	rfm7	rfm8	rfm9	rfm10	rfm11	rfm12	d
322	0	0.00	0.00	0	0	0	258.84	0.00	0	3.65	...	0.69	2.20	10	3.50	2.20	289	
51377	1	94.87	1.00	0	4	0	151.99	0.00	26	2.56	...	1.61	3.09	5	3.37	2.30	571	
61721	0	0.00	0.00	0	4	0	228.63	0.00	39	2.34	...	0.69	2.40	15	3.18	2.20	282	
96689	0	0.00	0.00	0	0	0	258.79	0.00	0	3.62	...	0.69	2.20	10	3.50	2.20	289	
147744	1	94.87	1.00	0	4	0	151.96	0.00	25	2.56	...	1.39	3.04	4	3.33	2.20	570	
158088	0	0.00	0.00	1	4	0	228.70	0.00	39	2.34	...	0.69	2.40	15	3.18	2.20	282	
193056	0	0.00	0.00	0	0	0	258.80	0.00	0	3.65	...	0.00	2.08	9	3.47	2.08	288	
244111	1	94.87	1.00	0	4	0	151.90	0.00	25	2.56	...	1.39	3.04	4	3.33	2.20	570	
254455	0	0.00	0.00	0	4	0	228.70	0.00	39	2.34	...	0.69	2.40	15	3.18	2.20	282	
289423	0	0.00	0.00	0	0	0	258.82	0.00	0	3.65	...	0.69	2.20	10	3.50	2.20	289	
340478	1	94.87	1.00	0	4	0	152.06	0.00	25	2.56	...	1.39	3.04	4	3.33	2.20	570	
350822	0	0.00	0.00	0	4	0	228.64	0.00	39	2.34	...	0.69	2.40	15	3.18	2.20	282	
385790	0	0.00	0.00	0	0	0	258.78	0.00	0	3.65	...	0.69	2.20	10	3.50	2.20	289	
436845	1	94.87	1.00	0	4	0	152.07	0.00	25	2.56	...	1.39	3.04	4	3.33	2.20	570	
447189	0	0.00	0.00	0	4	0	228.72	0.00	39	2.34	...	0.69	2.40	15	3.18	2.20	282	

In [32]:
```
originalDF = originalDF[(originalDF.rfm12>lower_limit)&(originalDF.rfm12<upper_limit)]
```

We are removing only the 32 outlier present in rfm12 column and not other coulm because they all are useful samples which willincrease the scope of the model unlike the outliers present in rfm12 which can be visualized through box plot.

In [33]:
```
df = originalDF
```

In [34]:
```
df.drop(['account'], axis=1, inplace=True)
df
```

Out[34]:

	b_tgt	int_tgt	cnt_tgt	cat_input1	cat_input2	demog_ho	demog_homeval	demog_inc	demog_pr	rfm1	...	rfm6	rfm7	rfm8	rfm9	rfm10	rfm11	rfn
1	1	83.67	2.00	0	0	0	239.97	52106.00	24	1.90	...	1.42	1.61	1.95	5	3.04	2.30	
2	1	122.47	2.00	0	0	0	210.16	42422.00	0	2.62	...	1.34	1.39	2.20	16	3.33	2.48	
3	0	0.00	0.00	0	0	0	300.98	59785.00	32	3.12	...	1.41	1.10	2.08	15	3.00	2.30	
4	0	0.00	0.00	0	0	0	316.72	0.00	0	2.12	...	1.54	1.79	3.00	24	2.64	1.95	
5	0	0.00	0.00	0	0	0	163.16	34444.00	0	3.26	...	1.29	0.00	1.61	26	2.64	1.79	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1060033	0	0.00	0.00	2	4	1	209.83	28584.00	28	2.77	...	0.53	0.69	0.69	23	2.20	1.39	
1060034	0	0.00	0.00	2	4	1	221.38	48129.00	24	2.77	...	0.53	0.69	0.69	23	2.20	1.39	
1060035	0	0.00	0.00	2	4	0	221.53	0.00	33	2.77	...	0.00	0.00	0.00	15	2.30	1.39	
1060036	1	122.47	1.00	1	4	0	215.15	31205.00	34	1.95	...	0.87	1.10	1.10	16	2.64	1.95	

Data Selection

In [35]:
```
X = df.drop(['b_tgt', 'int_tgt', 'cnt_tgt'], axis=1, inplace=False)
y = df[['b_tgt', 'int_tgt', 'cnt_tgt']]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=42)
y1_train = y_train['b_tgt']
y1_test = y_test['b_tgt']
y2_train = y_train['int_tgt']
y2_test = y_test['int_tgt']
y3_train = y_train['cnt_tgt']
y3_test = y_test['cnt_tgt']
```

Splitting into 80% train data and 20% test data.

Predicting b_tgt variable using classification algorithms

In [54]:
```
model_params = {
    'Logistic Regression' : {
        'model': LogisticRegression(),
        'params': {
            'C': [0.1, 2, 4, 6, 8, 10],
            'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'],
            'multi_class': ['auto', 'ovr', 'multinomial']
        }
    },
    'Random Forest': {
        'model': RandomForestClassifier(),
        'params' : {
            'n_estimators': [10, 50, 100, 200, 300, 400, 500],
            'criterion': ['gini', 'entropy', 'log_loss'],
            'max_features': ['None', 'sqrt', 'log2']
        }
    },
    'Decision Tree' : {
        'model': tree.DecisionTreeClassifier(),
        'params': {
            'criterion': ['gini', 'entropy', 'log_loss'],
            'splitter': ['best', 'random'],
            'max_features': ['auto', 'sqrt', 'log2']
        }
    }
}
```

In [59]:
```
scores1 = []

for model_name, mp in model_params.items():
    clf =  RandomizedSearchCV(mp['model'], mp['params'], cv=5, return_train_score=False)
    clf.fit(X_train, y1_train)
    print(model_name)
    scores1.append({
        'model': model_name,
        'best_score': clf.best_score_,
        'best_params': clf.best_params_,
        'accuracy' : clf.score(X_test, y1_test)
    })

acc1 = pd.DataFrame(scores1,columns=['model','best_score','best_params', 'accuracy'])
acc1
```

```
Logistic Regression
Random Forest
Decision Tree
```

Out[59]:

	model	best_score	best_params	accuracy
0	Logistic Regression	0.85	{'solver': 'newton-cg', 'multi_class': 'ovr', ...	0.85
1	Random Forest	0.99	{'n_estimators': 300, 'max_features': 'log2', ...	0.99
2	Decision Tree	0.98	{'splitter': 'best', 'max_features': 'auto', '...	0.98

In [60]:
```
maxAccuracy = acc1['accuracy'].max()
bestModel = acc1.loc[acc1['accuracy'] == maxAccuracy]
modelSelect = model_params[bestModel['model'].iloc[0]]
bestModelParameters = bestModel['best_params'].iloc[0]
bestModel
```

Out[60]:

	model	best_score	best_params	accuracy
1	Random Forest	0.99	{'n_estimators': 300, 'max_features': 'log2', ...	0.99

In [61]:
```
model = modelSelect['model']
model.set_params(**bestModelParameters)
```

Out[61]:
```
RandomForestClassifier(max_features='log2', n_estimators=300)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [68]:
```
mod = model.fit(X_train, y1_train)
```

In [69]:
```
with open('name.pkl', 'wb') as f3:
    pickle.dump(mod, f3)
```

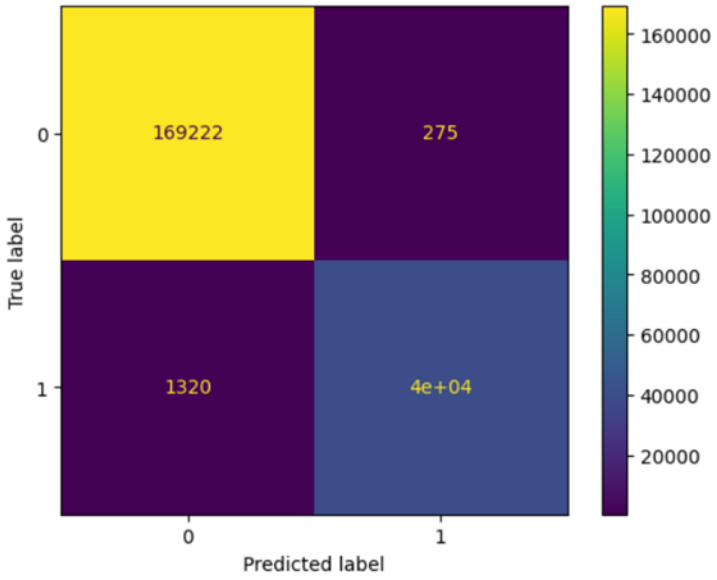
In [41]:
```
with open('name.pkl', 'rb') as f2:
        model1_1 = pickle.load(f2)
```

In [42]:
```
y1_predict = model1_1.predict(X_test)
print(classification_report(y1_test, y1_predict))
confusionMatrix = metrics.confusion_matrix(y1_test, y1_predict)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusionMatrix)
cm_display.plot()
plt.show()
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      1.00    169497
           1       0.99      0.97      0.98     41721

    accuracy                           0.99    211218
   macro avg       0.99      0.98      0.99    211218
weighted avg       0.99      0.99      0.99    211218
```



In [78]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y1_test,y1_predict)
```

Out[78]: 0.9924485602552813

Predicting int_tgt variable using prediction algorithm

In [36]:
```
# Predicting using Multi-Variate Regression
model2_1 = LinearRegression()
model2_1.fit(X_train, y2_train)
y2_predict = model2_1.predict(X_test)
mse = metrics.mean_squared_error(y2_test, y2_predict)
rmse = np.sqrt(mse)
print("Root Mean Square Error: "+str(rmse))
```

```
Root Mean Square Error: 38.44078838291896
```

Predicting cnt_tgt variable using prediction algorithms

In [37]:
```
# Predicting using Poisson Regression
model3_1 = linear_model.PoissonRegressor()
model3_1.fit(X_train, y3_train)
y3_predict = model3_1.predict(X_test)
mse = metrics.mean_squared_error(y3_test, y3_predict)
rmse = np.sqrt(mse)
print("Root Mean Square Error: "+str(rmse))
```

```
Root Mean Square Error: 0.6959821215646822
```

In [38]:
```
# Predicting using Multi-Variate Regression
model3_2 = LinearRegression()
model3_2.fit(X_train, y3_train)
y3_predict = model3_2.predict(X_test)
mse = metrics.mean_squared_error(y3_test, y3_predict)
rmse = np.sqrt(mse)
print("Root Mean Square Error: "+str(rmse))
```

```
Root Mean Square Error: 0.5810499742307964
```

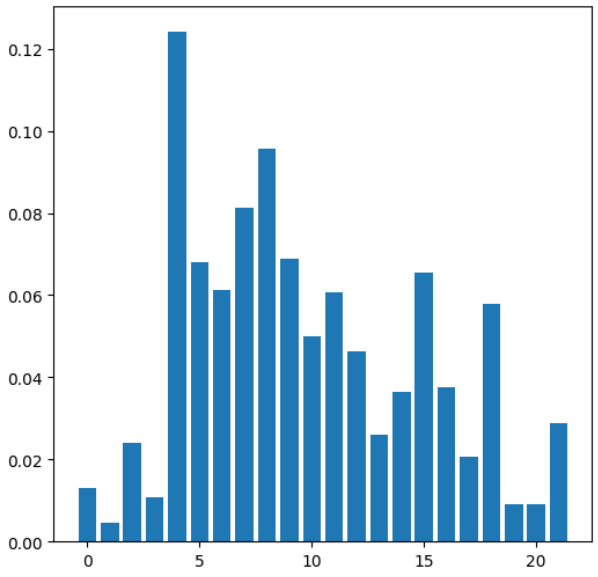
In [38]:
```
# Predicting using Multi-Variate Regression
model3_2 = LinearRegression()
model3_2.fit(X_train, y3_train)
y3_predict = model3_2.predict(X_test)
mse = metrics.mean_squared_error(y3_test, y3_predict)
rmse = np.sqrt(mse)
print("Root Mean Square Error: "+str(rmse))
```

```
Root Mean Square Error: 0.5810499742307964
```

Feature Selection

In [9]:
```
model = RandomForestRegressor(n_estimators=300,max_features='log2')
model.fit(X_train, y2_train)
importance = model.feature_importances_
```

In [10]:
```
plt.figure(figsize=(6,6))
plt.bar([x for x in range(len(importance))], importance)
plt.show()
```



In [11]:
```
q1=np.quantile(importance, .25)
q1
```

Out[11]:
```
0.021637328140448088
```

In [12]:
```
loc=[]
for i,v in enumerate(importance):
    if(v<q1):
        print('Feature: %0d, Score: %.5f' % (i,v))
        loc.append(i)
```

```
Feature: 0, Score: 0.01313
Feature: 1, Score: 0.00455
Feature: 3, Score: 0.01086
Feature: 17, Score: 0.02079
Feature: 19, Score: 0.00908
Feature: 20, Score: 0.00909
```

In [15]:
```
X_train = X_train.drop(X_train.columns[[0, 1, 3, 17, 19, 20]],axis = 1)
```

In [18]:
```
X_test = X_test.drop(X_test.columns[[0, 1, 3, 17, 19, 20]],axis = 1)
```

Predicting int_tgt variable using prediction algorithm after feature selection

In [19]:
```
# Predicting using Multi-Variate Regression
model2_1 = LinearRegression()
model2_1.fit(X_train, y2_train)
y2_predict = model2_1.predict(X_test)
mse = metrics.mean_squared_error(y2_test, y2_predict)
rmse = np.sqrt(mse)
print("Root Mean Square Error: "+str(rmse))
```

```
Root Mean Square Error: 38.44847443537332
```

Predicting cnt_tgt variable using prediction algorithms after feature selection

In [20]:
```
# Predicting using Poisson Regression
model3_1 = linear_model.PoissonRegressor()
model3_1.fit(X_train, y3_train)
y3_predict = model3_1.predict(X_test)
mse = metrics.mean_squared_error(y3_test, y3_predict)
rmse = np.sqrt(mse)
print("Root Mean Square Error: "+str(rmse))
```

```
Root Mean Square Error: 0.6959821215646822
```

In [21]:
```
# Predicting using Multi-Variate Regression
model3_2 = LinearRegression()
model3_2.fit(X_train, y3_train)
y3_predict = model3_2.predict(X_test)
mse = metrics.mean_squared_error(y3_test, y3_predict)
rmse = np.sqrt(mse)
print("Root Mean Square Error: "+str(rmse))
```

```
Root Mean Square Error: 0.5809123697857257
```