Machine Learning

Final Project

iFood CRM Data Analyst Case

Presentation by

Group 8

Nguyen Duc Minh - 2212560036
Nguyen Linh Nhi - 2212560043
Nghiem Quynh Anh - 2212560006
Nguyen Le Chau - 2212560011

Content

01

Problem Definition

Business Context

Objective of ML Model

Business Impact

Business Context

About ifood

iFood is the **lead food delivery app** in Brazil, present in over a thousand cities.

Consider a well-established company operating in the retail food sector. Presently they have around **several hundred thousand registered customers** and serve almost **one million consumers** a year.

5 major product categories

- Rare meat products
- Specially prepared fish
- Wines
- Exotic fruits
- Sweet products

3 main sales channels

- Physical stores
- Catalogs
- Company's website

Business Context

What challenge is iFood facing?

Globally, the company has enjoyed **solid revenues** and a healthy bottom line over **the past three years**. However,...

*the profit growth outlook for the **next three years** is **less promising**.*

Thus,...

*The **marketing department** was pressured to spend its annual budget more wisely.*

J2BD

Enhance the effectiveness of marketing activities, particularly focusing on marketing campaigns...

By hiring a small team of data scientists to build a predictive model to support direct marketing initiatives

> *In the retail food sector, predicting marketing campaign results can help the company foster a quantitative approach in decision-making, target the right customer segments, and maximize profitability.*

Business Objective

Propensity Scorecard

- Absolute Yes/No predictions can be misleading, especially in imbalanced business cases like churn or default.
- Propensity scores rank customers by their likelihood of responding positively to campaigns.

10,000
Maximum score for most potential customer profiles

0
Minimum score indicating least potential customer profiles

Benefit
Balances campaign scale with successful customer acquisitio

Outcome
Identifies optimal target segments for better precision.

Optimize campaign reach

Maximize ROI

Objective Of ML Model

Primary Objective

- To create a predictive model that will **evaluate the propensity scorecard**, which allows iFood to **target** only those customers **most likely to purchase** the gadget.
- This sixth campaign focuses on promoting a new gadget to the Customer Database.

The pilot campaign

To build the model, a pilot campaign was conducted with a **random sample** of 2,240 customers who were **contacted by phone** regarding the gadget.

Key metrics of the pilot campaign include:

- **Total cost:** 6,720 MU
- **Revenue generated:** 3,674 MU
- **Profit:** -3,046 MU
- **Success rate:** 15%

Other applications

Developing this model to predicts customer behavior, applying it to the broader customer base, which also enable iFood to:

- **Yield the highest profit** for the upcoming direct marketing campaigns.
- Analyze **characteristics of customers** who are **willing** to buy the gadget

Business Impact

Predict which customers are most likely to buy allows iFood to **boost the success rate** of **future campaigns**, improving both short-term performance and long-term **customer loyalty**

Improved Marketing Effectiveness

IMPACT

Increased Revenue

Focus marketing efforts on high-potential leads, thus **increasing** the **conversion rate**, generating **more sales** with the same marketing budget

Operational Efficiency

Optimize campaign timing, **offer personalization**, and refine customer segmentation, improving overall **campaign management efficiency**

02

Dataset Introduction

About Dataset
(Feature and Description)

It contains **a flag** for those **customers** who **responded the campaign**, **by buying** the product.

The data set contains socio-demographic and firmographic features

2.240 CUSTOMERS
WERE CONTACTED

The dataset contains

25 ATTRIBUTES
WERE RECORED

Feature	Attribute
AcceptedCmp1	1 if customer accepted the offer in the 1st campaign, 0 otherwise
AcceptedCmp2	1 if customer accepted the offer in the 2nd campaign, 0 otherwise
AcceptedCmp3	1 if customer accepted the offer in the 3rd campaign, 0 otherwise
AcceptedCmp4	1 if customer accepted the offer in the 4th campaign, 0 otherwise
AcceptedCmp5	1 if customer accepted the offer in the 5th campaign, 0 otherwise
Response (target)	1 if customer accepted the offer in the last campaign, 0 otherwise
Complain	1 if customer complained in the last 2 years
DtCustomer	Customer's level of education
Education	Customer's marital status

Feature	Attribute
Marital	Customer's marital status
Kidhome	Number of small children in customer's household
Teenhome	Number of teenagers in customer's household
Income	Customer's yearly household income
MntFishProducts	Amount spent on fish products in the last 2 years
MntMeatProducts	Amount spent on meat products in the last 2 years
MntFruits	Amount spent on fruits products in the last 2 years
MntSweetProducts	Amount spent on sweet products in the last 2 years
MntWines	Amount spent on wines products in the last 2 years

Campaign response Personal information Purchase Behavior

Feature	Attribute
MntGoldProds	Amount spent on gold products in the last 2 years
NumDealsPurchases	Number of purchases made with discount
NumCatalogPurchases	Number of purchases made using catalog
NumStorePurchases	Number of purchases made directly in stores
NumWebPurchases	Number of purchases made through company's website
NumWebVisitsMonth	Number of visits to company’s website in the last month
Recency	Number of days since the last purchase

Dataset Detailed Overview

	dtype	valid_instances	total_null	null_pct	unique	unique_values
ID	int64	2240	0	0.0	2240	[5524, 2174, 4141, 6182, 5324, 7446, 965, 6177...
Year_Birth	int64	2240	0	0.0	59	[1957, 1954, 1965, 1984, 1981, 1967, 1971, 198...
Education	object	2240	0	0.0	5	[Graduation, PhD, Master, Basic, 2n Cycle]
Marital_Status	object	2240	0	0.0	8	[Single, Together, Married, Divorced, Widow, A...
Income	float64	2216	24	1.08	1974	[58138.0, 46344.0, 71613.0, 26646.0, 58293.0, ...
Kidhome	int64	2240	0	0.0	3	[0, 1, 2]
Teenhome	int64	2240	0	0.0	3	[0, 1, 2]
Dt_Customer	object	2240	0	0.0	663	[2012-09-04, 2014-03-08, 2013-08-21, 2014-02-1...
Recency	int64	2240	0	0.0	100	[58, 38, 26, 94, 16, 34, 32, 19, 68, 11, 59, 8...
MntWines	int64	2240	0	0.0	776	[635, 11, 426, 173, 520, 235, 76, 14, 28, 5, 6...
MntFruits	int64	2240	0	0.0	158	[88, 1, 49, 4, 43, 42, 65, 10, 0, 5, 16, 61, 2...
MntMeatProducts	int64	2240	0	0.0	558	[546, 6, 127, 20, 118, 98, 164, 56, 24, 11, 48...
MntFishProducts	int64	2240	0	0.0	182	[172, 2, 111, 10, 46, 0, 50, 3, 1, 11, 225, 6,...
MntSweetProducts	int64	2240	0	0.0	177	[88, 1, 21, 3, 27, 42, 49, 2, 112, 5, 68, 13, ...
MntGoldProds	int64	2240	0	0.0	213	[88, 6, 42, 5, 15, 14, 27, 23, 2, 13, 1, 16, 3...
NumDealsPurchases	int64	2240	0	0.0	15	[3, 2, 1, 5, 4, 15, 7, 0, 6, 9, 12, 8, 10, 13,...
NumWebPurchases	int64	2240	0	0.0	15	[8, 1, 2, 5, 6, 7, 4, 3, 11, 0, 27, 10, 9, 23,...
NumCatalogPurchases	int64	2240	0	0.0	14	[10, 1, 2, 0, 3, 4, 6, 28, 9, 5, 8, 7, 11, 22]
NumStorePurchases	int64	2240	0	0.0	14	[4, 2, 10, 6, 7, 0, 3, 8, 5, 12, 9, 13, 11, 1]
NumWebVisitsMonth	int64	2240	0	0.0	16	[7, 5, 4, 6, 8, 9, 20, 2, 3, 1, 10, 0, 14, 19,...
AcceptedCmp3	int64	2240	0	0.0	2	[0, 1]
AcceptedCmp4	int64	2240	0	0.0	2	[0, 1]
AcceptedCmp5	int64	2240	0	0.0	2	[0, 1]
AcceptedCmp1	int64	2240	0	0.0	2	[0, 1]
AcceptedCmp2	int64	2240	0	0.0	2	[0, 1]
Complain	int64	2240	0	0.0	2	[0, 1]
Z_CostContact	int64	2240	0	0.0	1	[3]
Z_Revenue	int64	2240	0	0.0	1	[11]
Response	int64	2240	0	0.0	2	[1, 0]

03

EDA And Data-Preprocessing

Data Cleaning

- *Outliers*
- *Missing Data*

Exploratory Data Analysis (EDA)

Feature Engineering

- *Feature Generation*
- *Feature Discretization*
- *One-hot Coding*

Feature Evaluation & Selection

- *Correlation Matrix - Tackling Multicollinearity*
- *Information Value Score (IV Model)*

DEALING WITH

MISSING DATA

	dtype	valid_instances	unique	total_null	null_pct	duplicates
ID	int64	2240	2240	0	0.0	0
Year_Birth	int64	2240	59	0	0.0	2181
Education	object	2240	5	0	0.0	2235
Marital_Status	object	2240	8	0	0.0	2232
Income	float64	2216	1974	24	1.08	265
Kidhome	int64	2240	3	0	0.0	2237
Teenhome	int64	2240	3	0	0.0	2237
Dt_Customer	object	2240	663	0	0.0	1577
Recency	int64	2240	100	0	0.0	2140
MntWines	int64	2240	776	0	0.0	1464
MntFruits	int64	2240	158	0	0.0	2082
MntMeatProducts	int64	2240	558	0	0.0	1682
MntFishProducts	int64	2240	182	0	0.0	2058
MntSweetProducts	int64	2240	177	0	0.0	2063
MntGoldProds	int64	2240	213	0	0.0	2027
NumDealsPurchases	int64	2240	15	0	0.0	2225
NumWebPurchases	int64	2240	15	0	0.0	2225
NumCatalogPurchases	int64	2240	14	0	0.0	2226
NumStorePurchases	int64	2240	14	0	0.0	2226
NumWebVisitsMonth	int64	2240	16	0	0.0	2224
AcceptedCmp3	int64	2240	2	0	0.0	2238
AcceptedCmp4	int64	2240	2	0	0.0	2238
AcceptedCmp5	int64	2240	2	0	0.0	2238
AcceptedCmp1	int64	2240	2	0	0.0	2238
AcceptedCmp2	int64	2240	2	0	0.0	2238
Complain	int64	2240	2	0	0.0	2238
Z_CostContact	int64	2240	1	0	0.0	2239
Z_Revenue	int64	2240	1	0	0.0	2239
Response	int64	2240	2	0	0.0	2238

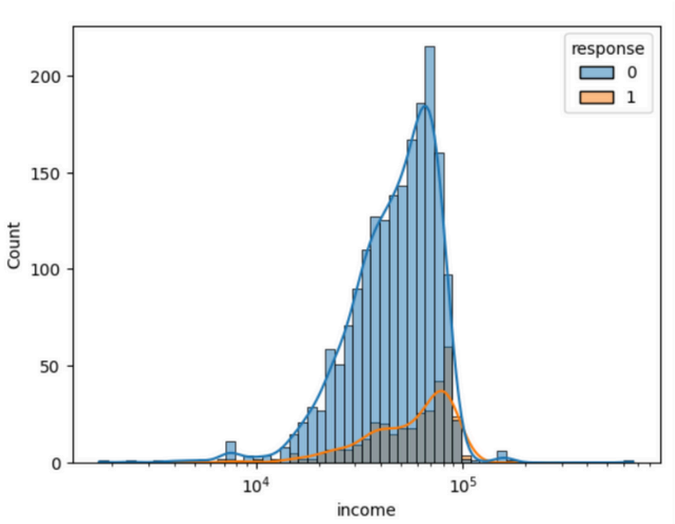
VALIDATION RESULTS USING LGBMRegressor:

```
Early stopping, best iteration is:
[52]    valid_0's rmse: 9490.66 valid_0's l2: 9.00726e+07
Root Mean Squared Error: 13643.049344097215
R2 Score: 0.7215512846402143
```

Normally with Minimal missing data like this, implementing imputation is not necessary. However, I will try to implement model-based method since there are already a number of associations with other features (especially amount spent)

With the std for income being 25173.076661
(outliers only account for 0.36%)

```
Outliers in income: 8  - Accounting for 0.36% of data
Upper 5 outliers:
+------+------+------------+------------+----------------+----------+
|      |  id  | year_birth | education  | marital_status |  income  |
+------+------+------------+------------+----------------+----------+
| 2233 | 9432 |    1977    | Graduation |    Together    | 666666.0 |
| 617  | 1503 |    1976    |    PhD     |    Together    | 162397.0 |
| 687  | 1501 |    1982    |    PhD     |    Married     | 160803.0 |
| 1300 | 5336 |    1971    |   Master   |    Together    | 157733.0 |
| 164  | 8475 |    1973    |    PhD     |    Married     | 157243.0 |
+------+------+------------+------------+----------------+----------+
Lower 5 outliers:
+------+------+------------+------------+----------------+----------+
|      |  id  | year_birth | education  | marital_status |  income  |
+------+------+------------+------------+----------------+----------+
| 1300 | 5336 |    1971    |   Master   |    Together    | 157733.0 |
| 164  | 8475 |    1973    |    PhD     |    Married     | 157243.0 |
| 1653 | 4931 |    1977    | Graduation |    Together    | 157146.0 |
| 2132 | 11181|    1949    |    PhD     |    Married     | 156924.0 |
| 655  | 5555 |    1975    | Graduation |    Divorced    | 153924.0 |
+------+------+------------+------------+----------------+----------+
```



⇒ **This is an acceptable result for imputation.**

FEATURE ENGINEERING

- Feature Generation
- Feature Discretization
- One-hot Coding

Convert Year of Birth to Age

Description: Calculate THE AGE OF INDIVIDUAL from their year of birth.

Transformation:

- **Create an Age feature** by subtracting the year of birth from the current year.
- **Optional Binning**: Group ages into bins (e.g., 18–25, 26–35, etc.) to reduce variability, basing on distribution (later EDA).

Ordinal Encoding for Education

Description: Transform education levels into ORDERED INTERGER VALUES

Transformation

- Map education categories (e.g., "High School," "Bachelor's," "Master's") to integers based on a defined order (e.g., 1, 2, 3).

Normalize Income

Description: STANDARDIZE OR NORMALIZE the income feature for scale consistency.

Transformation:

- Apply normalization or standardization to the income attribute to ensure it falls within a common range.

Create Binary Features for Children and Teenagers at Home

Description: FLAG THE PRESENCE of children or teenagers at home.

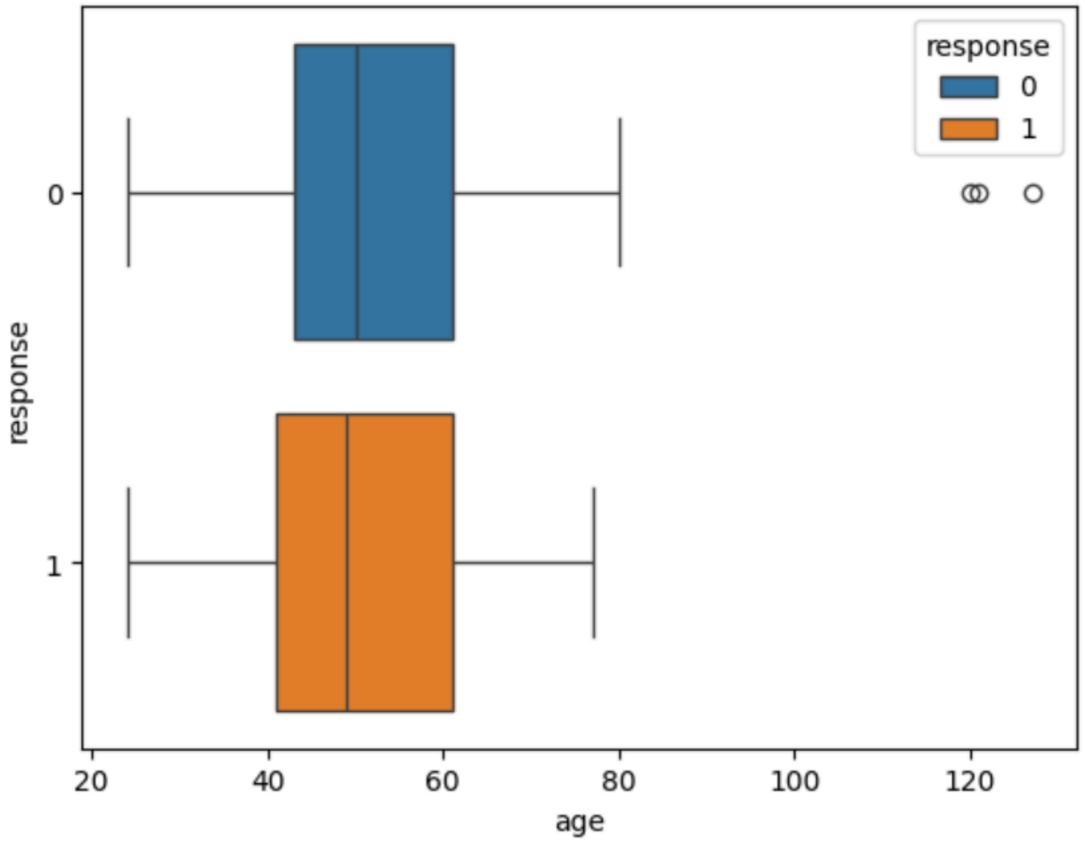
Transformation:

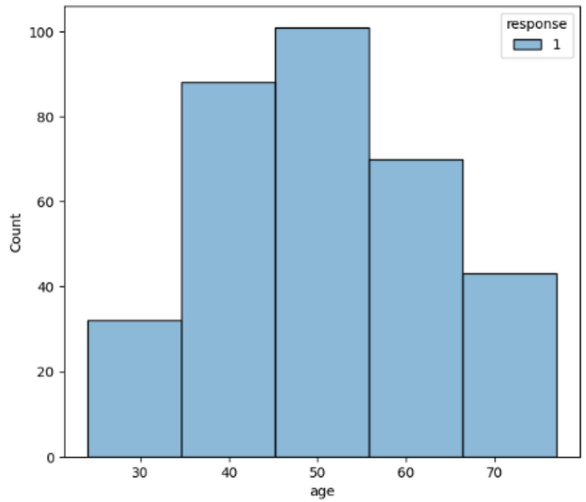
- Create two binary features, **Has_Kid** and **Has_Teen** or **have_kids_or_teens**, indicating the presence of children and teenagers at home.

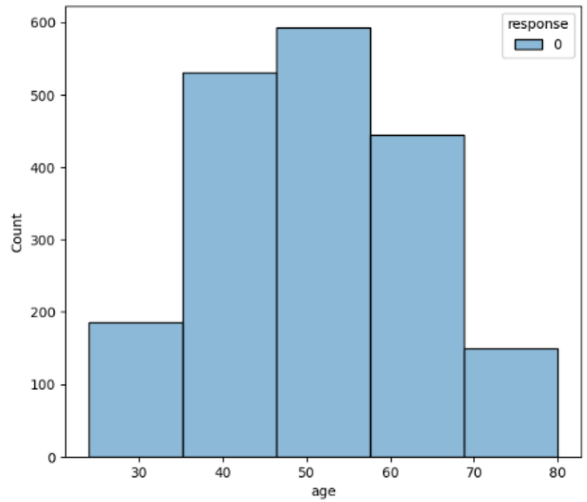
Note: The two attributes: `Z_CostContact` & `Z_Revenue` consist of only singular value, which bears little importance for analysis => **Needs dropping**

Group Ages

Into Bins To Reduce Variability







THE **VARIANCE** BETWEEN THE TWO GROUP IS **SURELY < 0.99**

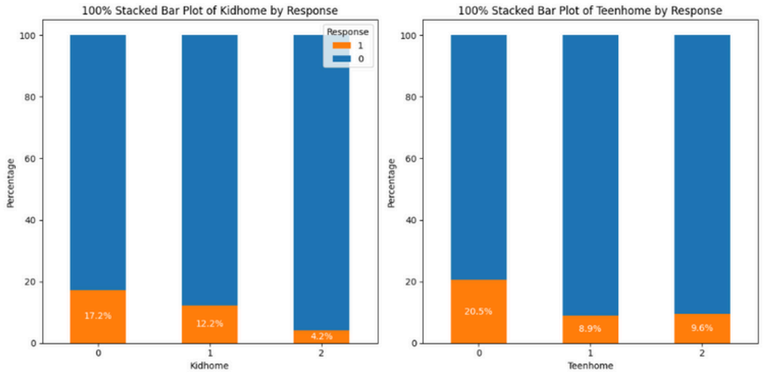
⇒ Further supported by a t-test:

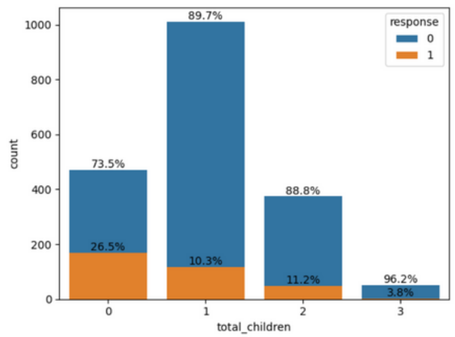
```
T-statistic: -0.9810244356163194, P-value: 0.3271109227335284
The difference in age between the two classes of response is not statistically significant.
```

⇒ Unnecessary to bin ages beforehand

Binary Feature

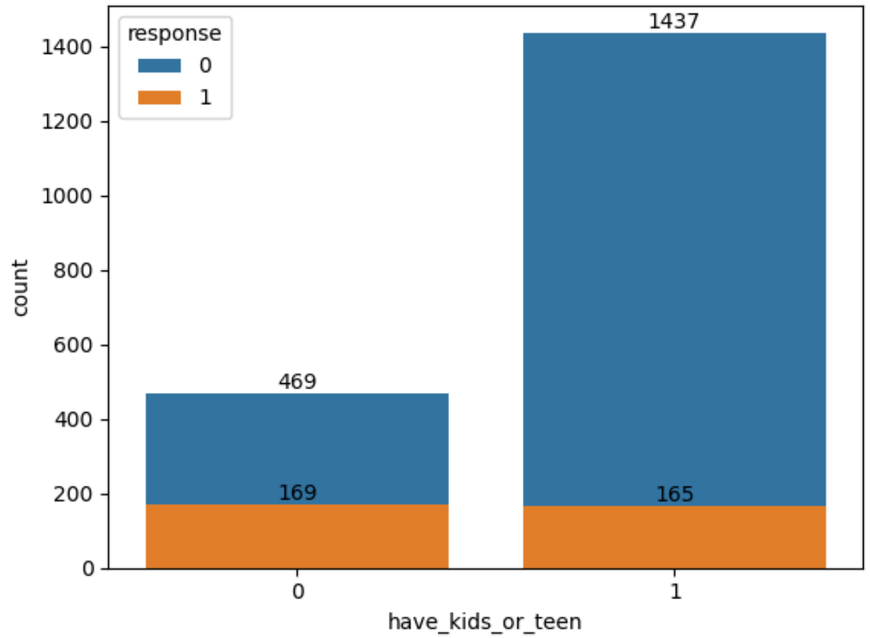
For Children and Kids Features





⇒ Noticeably, those without kids or teens witnessed a **higher probability of accepting the offer** in the last campaign. Therefore, this will be discretized into a separate group, opposing the other **"with kids" - have_kids_or_teen**

```
T-statistic: -8.9256473741134, P-value: 1.3521340970340653e-17
The difference in have_kids_or_teen between the two classes of response is statistically significant.
```



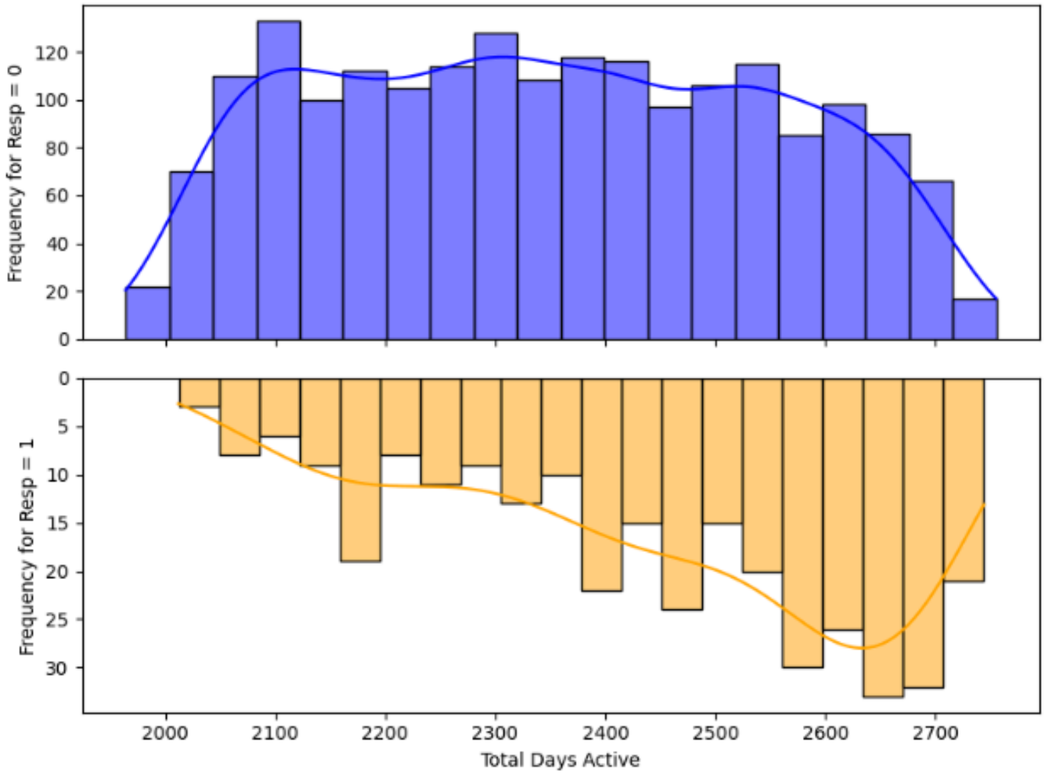
⇒ There is no need to have the **total number of children** since the **have_kids_or_teen generally displays a more significant pattern**

Total Days

Active Distribution Between Class

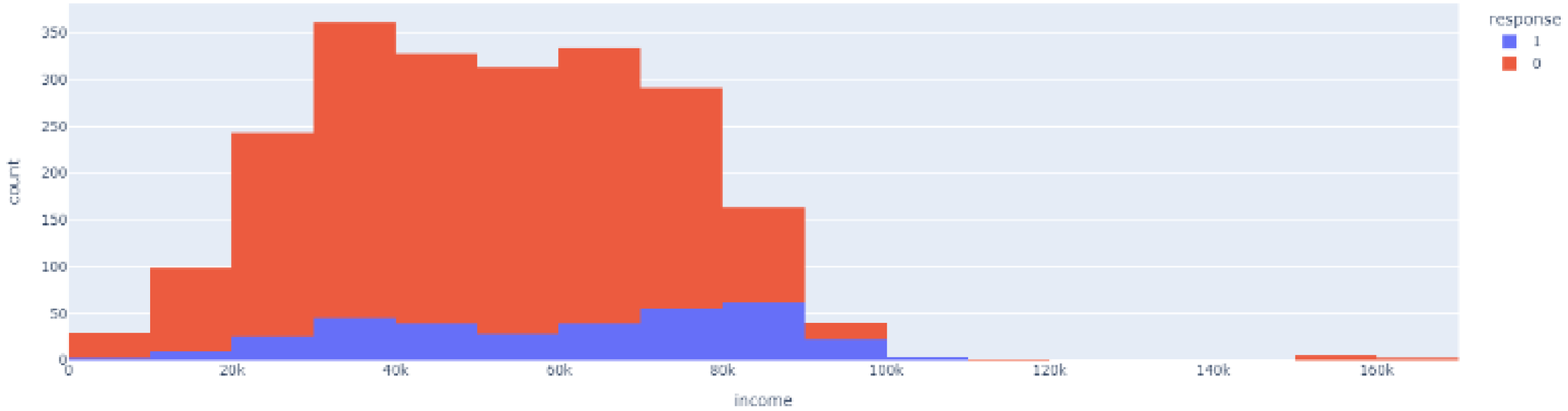
⇒ Discretize based on **quartiles** labelling (['Low', 'Below Average', 'Above Average', 'High']) before ordinally encoding these

Inference: the **higher the total active days**, the more likely one is to respond more positively to the campaign



Distribution of Income

In relation to Response

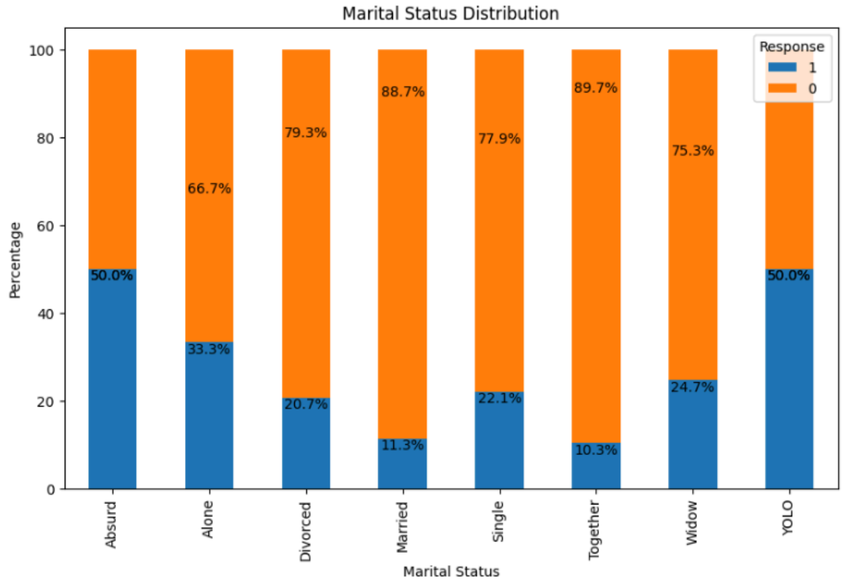


We may sense a pattern of distribution within the chart where those with higher income tend to stand a **higher chance/likelihood** of accepting the offer.

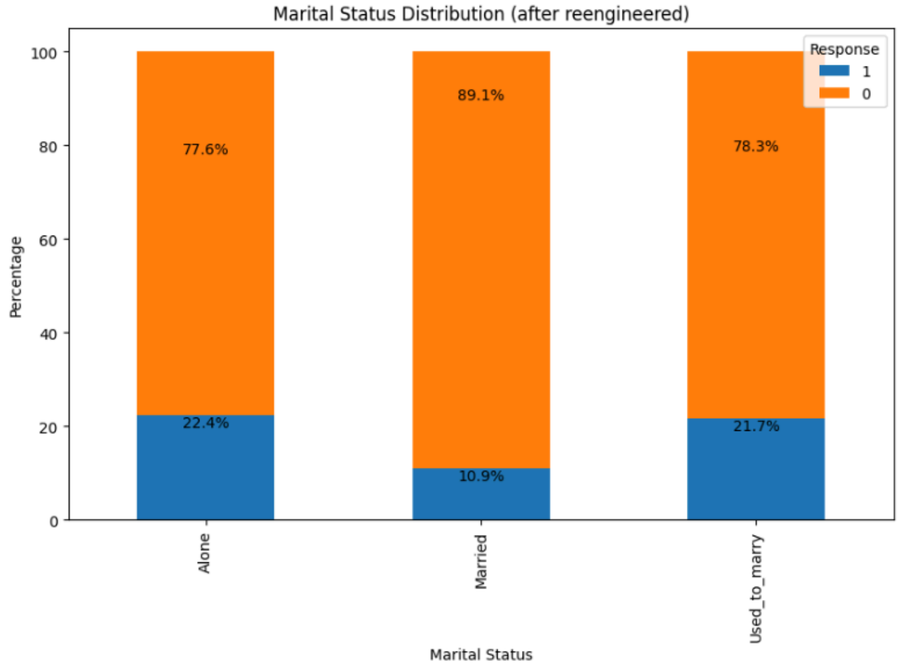
However, since the underlying pattern is not so obvious, we gonna **defer it to IV model with binning methods** to see this attribute's predictive power, or probably leave it to the tree model to discretize

Marital Status

Clustering




```
alone_col = ['Alone', 'YOLO', 'Absurd', 'Single']
married_col = ['Married', 'Together']
used_to_marry = ['Divorced', 'Widow']
```

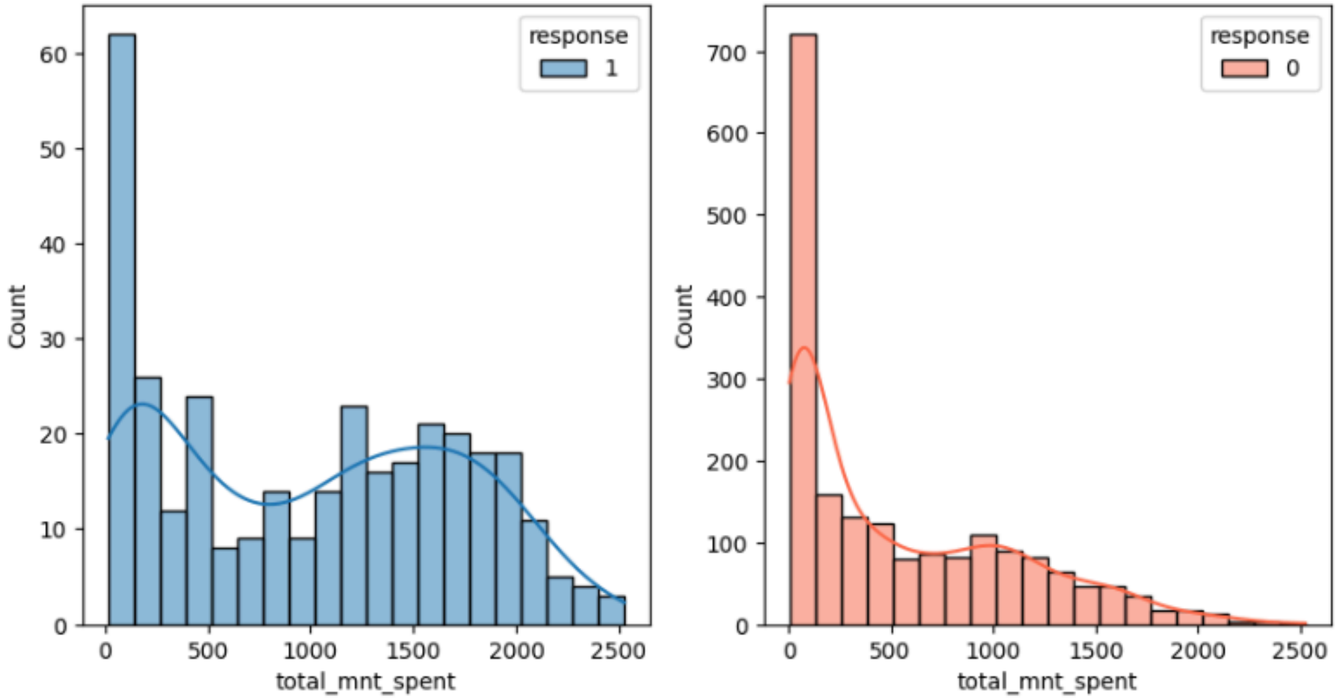


PATTERNS ARE MORE CONSPICUOUS FOR EACH OF THE DEFINED GROUPS!!!

Insight: With Along having the highest propensity, and Married having significantly lower rate

Total_mnt_spent

Aggregation Of Amount For Products

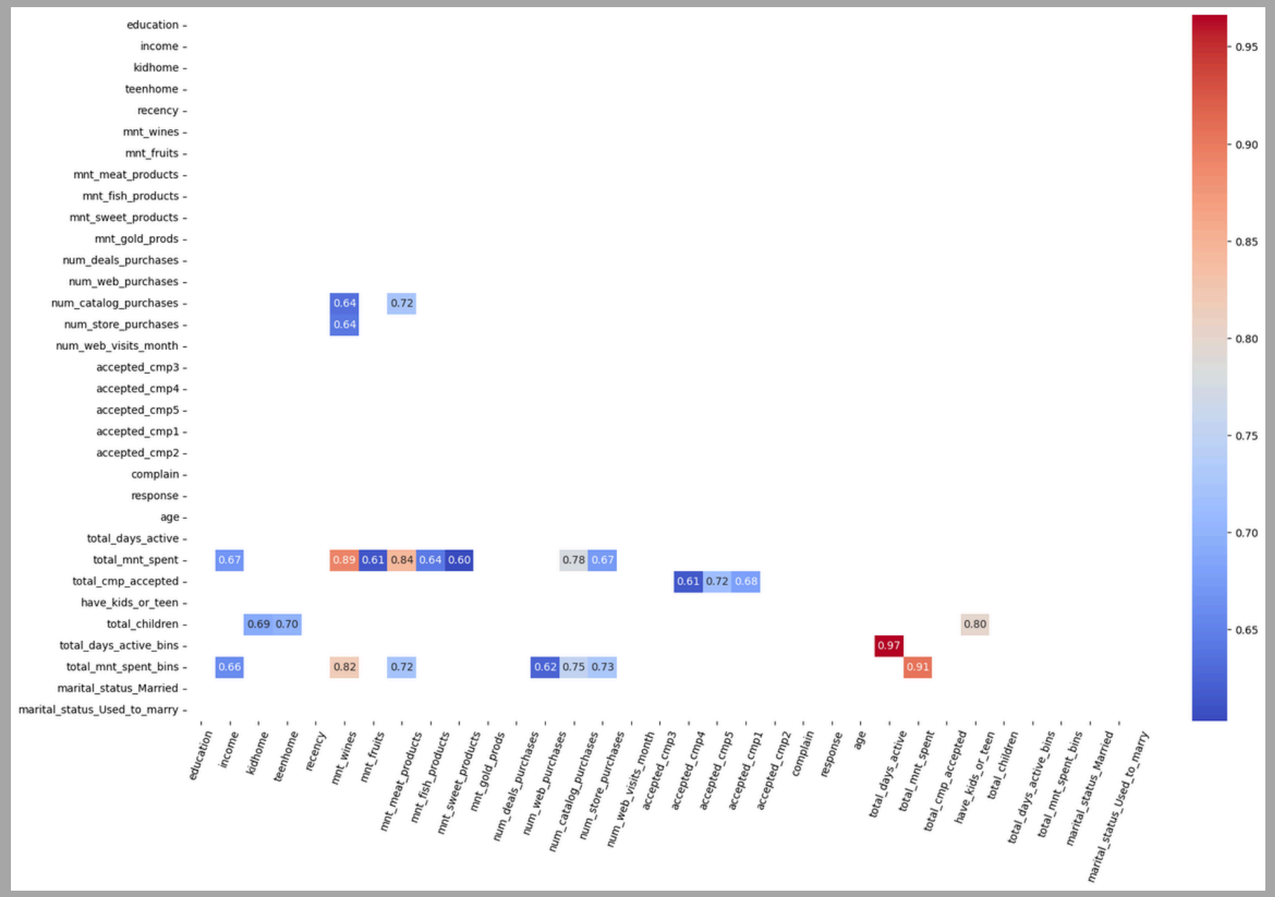


With **Response = 0**, distribution is obviously right skewed, however, there is a second peak around 1600 (seemingly **a bimodal distribution for Positive Respondents**)

Higher amount spent, higher chance of responding to the campaign, preliminarily we will **discretize based on 4 quartiles**

FEATURE EVAUATION AND SELECTION

Correlation Matrix Heatmap - Tackling Multicollinearity



FEATURE EVAUATION AND SELECTION

Information Value Scoring Model

Listed side are the predictive power of each attribute in relation to the target var “Response”

Variable	IV
total_cmp_accepted	1.069102
total_mnt_spent	0.645576
mnt_meat_products	0.587571
accepted_cmp5	0.544276
total_mnt_spent_bins	0.496176
mnt_wines	0.483095
total_days_active	0.462772
mnt_gold_prods	0.448182
accepted_cmp1	0.439637
total_days_active_bins	0.365807
num_catalog_purchases	0.357837
income	0.357560
recency	0.350256
num_web_purchases	0.347206
accepted_cmp3	0.342463
total_children	0.314885
have_kids_or_teen	0.297264
num_store_purchases	0.232975
mnt_sweet_products	0.232895
mnt_fruits	0.217650
teenhome	0.217560
accepted_cmp4	0.177987
marital_status_Married	0.168880
mnt_fish_products	0.154146
accepted_cmp2	0.136106
num_web_visits_month	0.129615
education	0.090297
num_deals_purchases	0.075562
kidhome	0.061901
age	0.051926
marital_status_Used_to_marry	0.040165
complain	0.000023

Information Value	Predictive Power
< 0.02	Not Useful
0.02 to 0.1	Weak Predictor
0.1 to 0.3	Medium Predictor
0.3 to 0.5	Strong Predictor
> 0.5	Suspicious

Inference from IV score Model & Correlational Matrix

- `**complain**` has a significantly **low IV score**, therefore, we can neglect this attribute.
- Besides, `**kidhome**` and `**teenhome**` can be excluded from the model to avoid multicollinearity. The same idea also holds true with `total_children`, availing ourselves of only `**have_kids_or_teen**` is already comparable.
- With the case of `**total_mnt_spent**`, current method of discretization is not optimal since its relation with the target var is not completely linear, therefore we will drop its discretized version **(`total_mnt_spent_bins`)** and instead, leave the original attribute to the model to decide (in case of trees model), otherwise, increase the number of bins.
- The same is **true** for `**total_days_active**` and its binned attribute.

04

Model Implementation

Quỳnh Anh's Model

- *Logistic Regression - Linearity Testing*
- *SMOTE Upsampling Integration*

Linh Nhi's Model

- *Random Forest Model - Bootstrap Aggregating*
- *SMOTE Upsampling Integration*

Lê Châu's Model

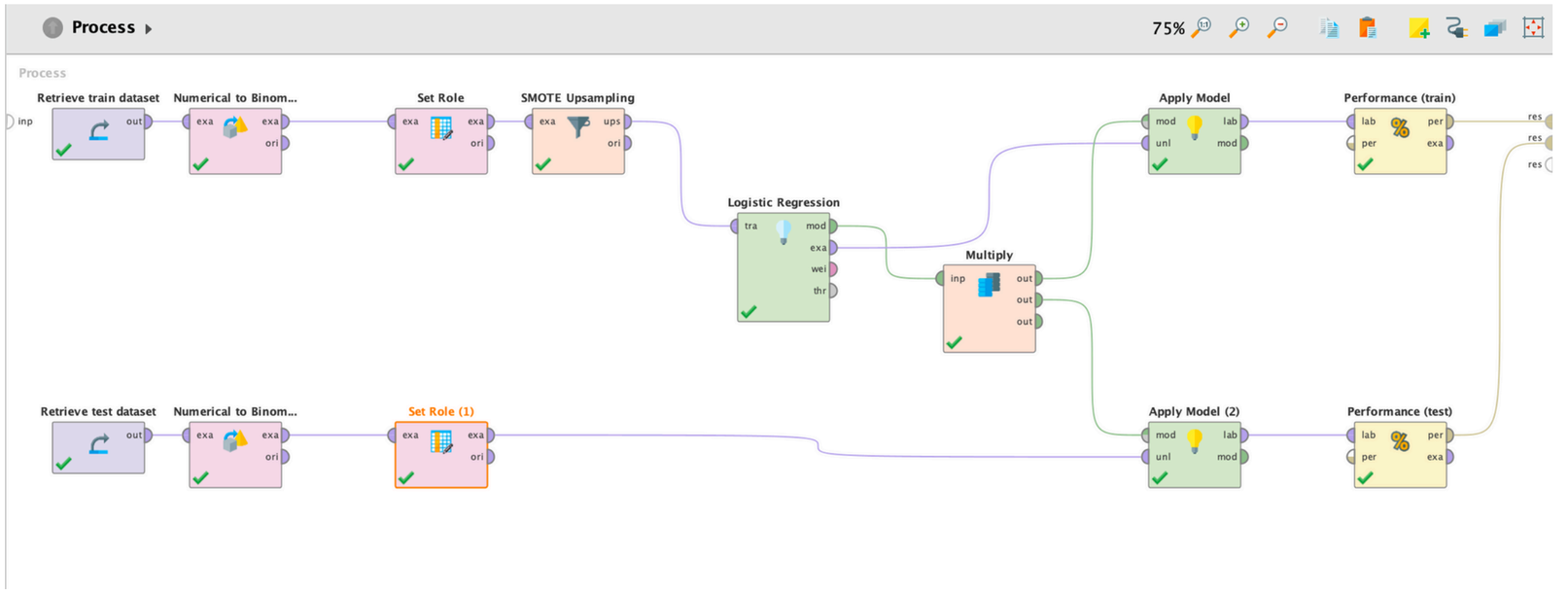
- *Gradient Boosting - Boosted Model*
- *SMOTE Upsampling Integration*

Đức Minh's Model

- *XGBoost vs. LGBM Classifier - Boosted Model*
- *Hypertuning using BayesSearchCV with skopt library*

QUYNH ANH'S MODEL

LOGISTIC REGRESSION

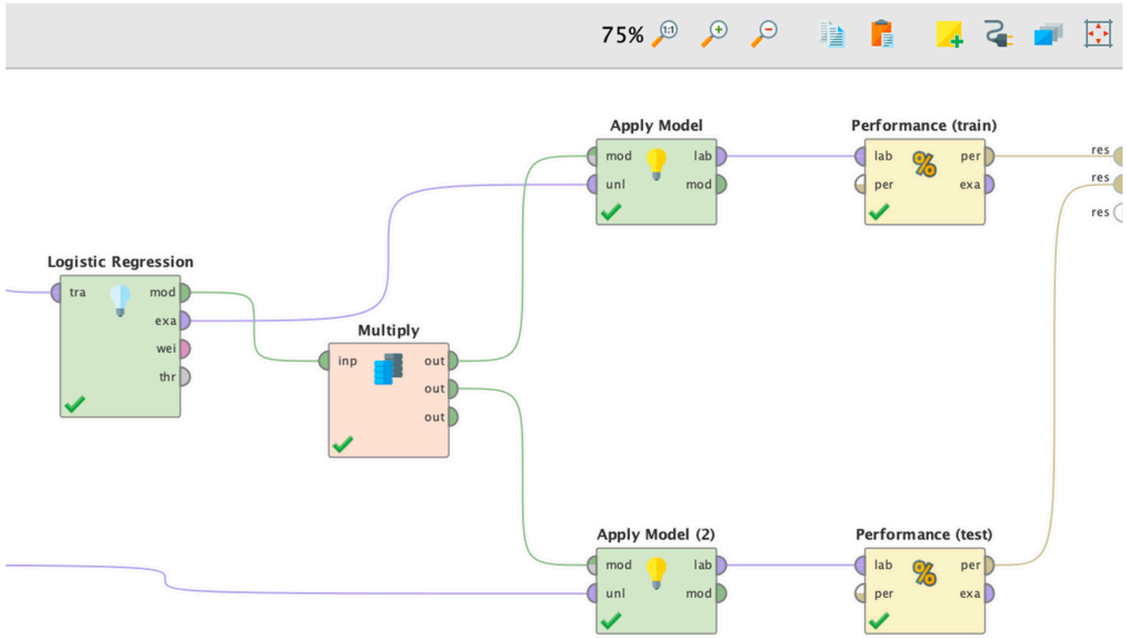


Reason for Choosing Logistic Regression

- The dataset contains information on **2,240 customers**, which is **relatively small**. Logistic Regression performs well on small to medium-sized datasets, making it a suitable choice for this pilot campaign data.
- One of the tasks is to predict whether a customer will respond to a marketing campaign (target variable: **Response = 1** if the customer accepted the offer, 0 otherwise). This is a **binary classification problem**. Logistic Regression models the probability of a binary outcome using the logistic function, making it a natural fit for predicting customer responses.
- One of the objectives mentioned by the CMO is to understand the **characteristic features** of customers who are likely to respond to the campaign. Logistic Regression provides **clear and interpretable coefficients**, allowing us to **determine the impact** of each feature (e.g., Income, MntWines, NumWebPurchases) on the likelihood of a customer responding.
- The **pilot campaign** data shows a **15% response rate**, indicating an **imbalance** between the positive and negative classes. Logistic Regression can handle this imbalance effectively by **using class weighting**. This adjustment helps the model pay more attention to the **minority class (respondents)**, increasing its sensitivity and improving the accuracy of predicting positive responses.

QUYNH ANH'S MODEL

How LOGISTIC REGRESSION *works conceptually?*



- **Modeling the Probability:** Logistic Regression estimates the probability that a customer will accept the offer **(Response = 1)**. It does this by modeling the **log-odds** of the **positive class** as a **linear function** of the input features *(e.g., Income, NumWebPurchases, MntWines)*.

$$\text{log-odds} = \beta_0 + \beta_1 \times X_1 + \beta_2 \times X_2 + \ldots + \beta_n \times X_n$$

intercept — *coefficient of X* — *feature*

The log-odds are then transformed using the **logistic (sigmoid) function** to obtain a **probability score** between **0** and **1**.

$$P(\text{Response} = 1) = \frac{1}{1 + e^{-(\beta_0 + \sum \beta_i X_i)}}$$

- **Decision Threshold:** The model outputs a probability score for each customer. By applying a decision threshold (typically 0.5), the model classifies customers:
  - If ***P(Response=1) ≥ 0.5***, the customer is predicted to respond.
  - If ***P(Response=1) < 0.5***, the customer is predicted not to respond.

- **Interpretability:** Logistic Regression provides interpretable coefficients, which tell us the **direction** and **magnitude of the impact** of each **feature** on the **response probability**. For instance, a *positive coefficient* for *Income* would indicate that *higher income* is associated with a *higher likelihood of responding*.

QUYNH ANH'S MODEL

Feature Engineering before Applying LOGISTIC REGRESSION?

To make the **fairest comparison** among models, the original dataset has been SPLITTED into a **TRAIN SET** and a **TEST SET** for model building.

Numerical to Binominal

Convert the **target variable *(Response)*** from a numerical format (0 or 1) to a binominal **(categorical) format**, specifically representing it as two categories: Yes (1) and No (0).

Indicate that this is a binary classification problem, helping the model understand that it is predicting a categorical outcome (Yes or No) rather than a continuous numerical value.

Set Role

Set **target variable *(Response)*** as **Label Role**, which the model aims to predict.

Ensure that the model knows which column is the target variable (Response) and which columns are predictors (e.g., Income, NumWebPurchases).

SMOTE Upsampling

To **handle imbalance** dataset, because in our dataset, the **response rate** is only **15%**, making it imbalanced *(more non-respondents than respondents)*.

May lead to the model being biased towards the majority class *(non-respondents)*, resulting in poor performance for predicting.

SMOTE generates **synthetic samples** for the minority class *(respondents)* by interpolating between existing samples. This increases the number of positive examples in the training data, helping the model learn better.

MODEL EVALUATION

LOGISTIC REGRESSION

Performance (Test)

- **accuracy:** 86.05%
- **precision:** 93.22% (positive class: false)
- **recall:** 90.16% (positive class: false)
- **AUC:** 0.878
- **f1 score:** 57.34%

	true false	true true	class precision
pred. false	**687**	**50**	93.22%
pred. true	**75**	**84**	52.83%
class recall	90.16%	62.69%	

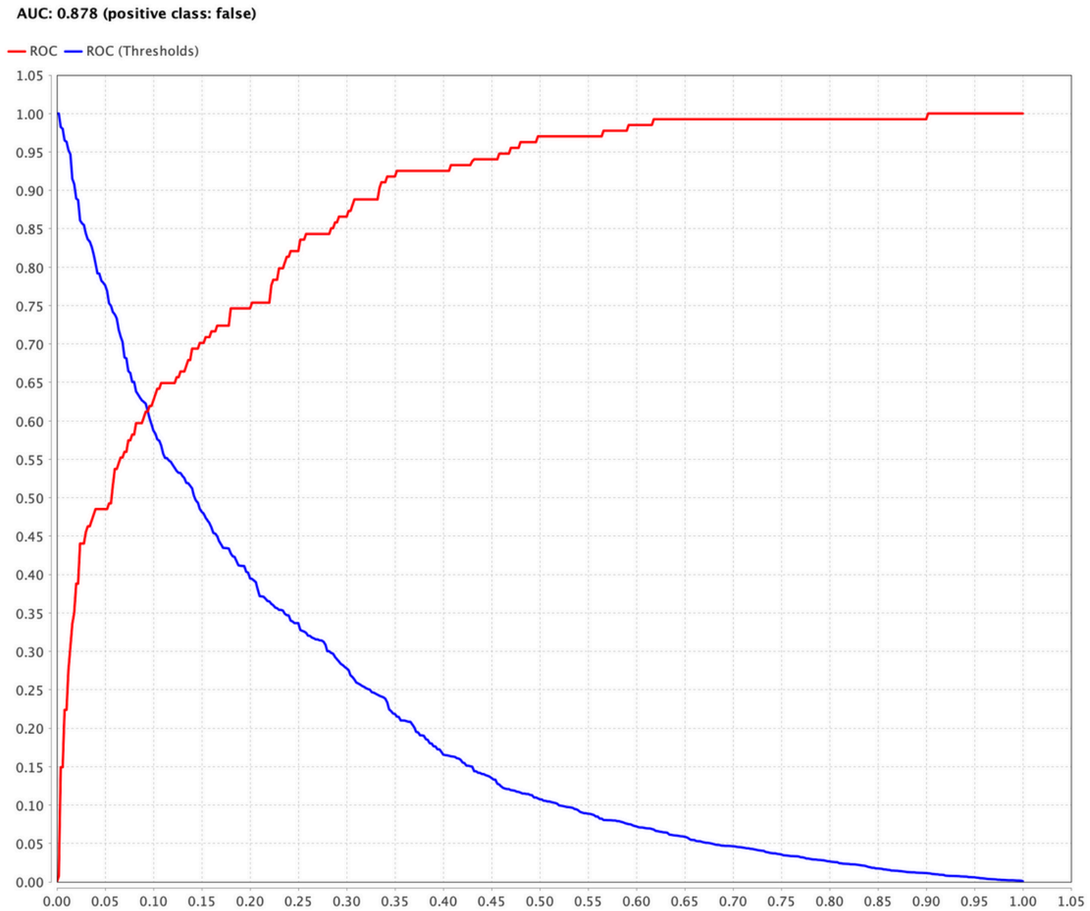
Model's Performance Evaluation

- The model is performing well in identifying non-respondents (high true negatives), but it struggles with identifying actual respondents (moderate true positives and higher false negatives).
- An **accuracy** of **86.05%** suggests that the model is correctly classifying a large portion of the customers. However, in imbalanced datasets like this one (with only 15% respondents), accuracy can be misleading because it is influenced heavily by the majority class (non-respondents).
- The **precision** for predicting respondents is only **52.83%**, which indicates that many of the customers predicted as respondents did not actually respond. This can lead to inefficient marketing efforts.
- A **moderate recall** score of **62.69%** indicates that the model is able to capture a good portion of the respondents but is still missing about 37% of them. In a marketing context, missing potential respondents could mean lost revenue opportunities.

MODEL EVALUATION

LOGISTIC REGRESSION

Performance (Test)



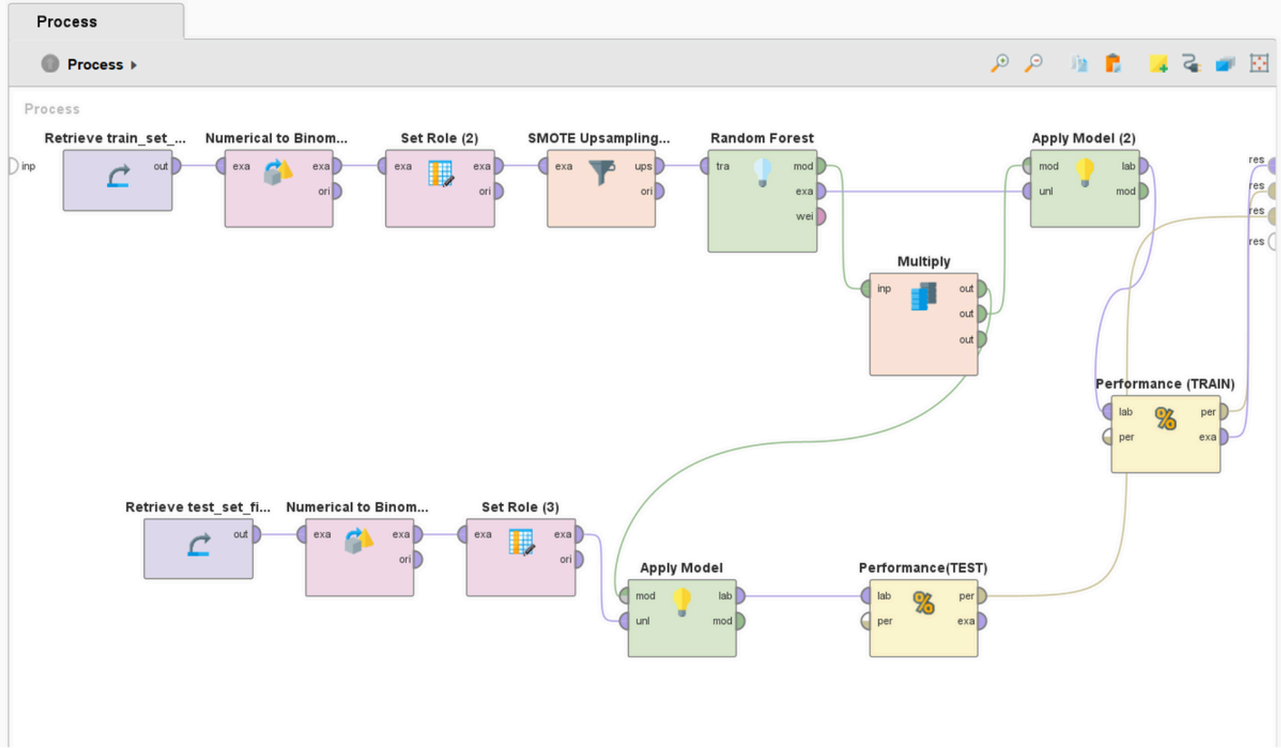
Model's Performance Evaluation

- An **AUC** of **0.878** is a strong indicator of good model performance. It suggests that the model has a high ability to differentiate between respondents and non-respondents.

- The high AUC score indicates that, overall, the model can **effectively rank customers** based on their likelihood of responding, even if the precision and recall need improvement.

LINH NHI'S MODEL

RANDOM FOREST



Reason for Choosing Random Forest

- The original dataset selected **contains 25 attributes** and information **was collected from 2,240 customers.**

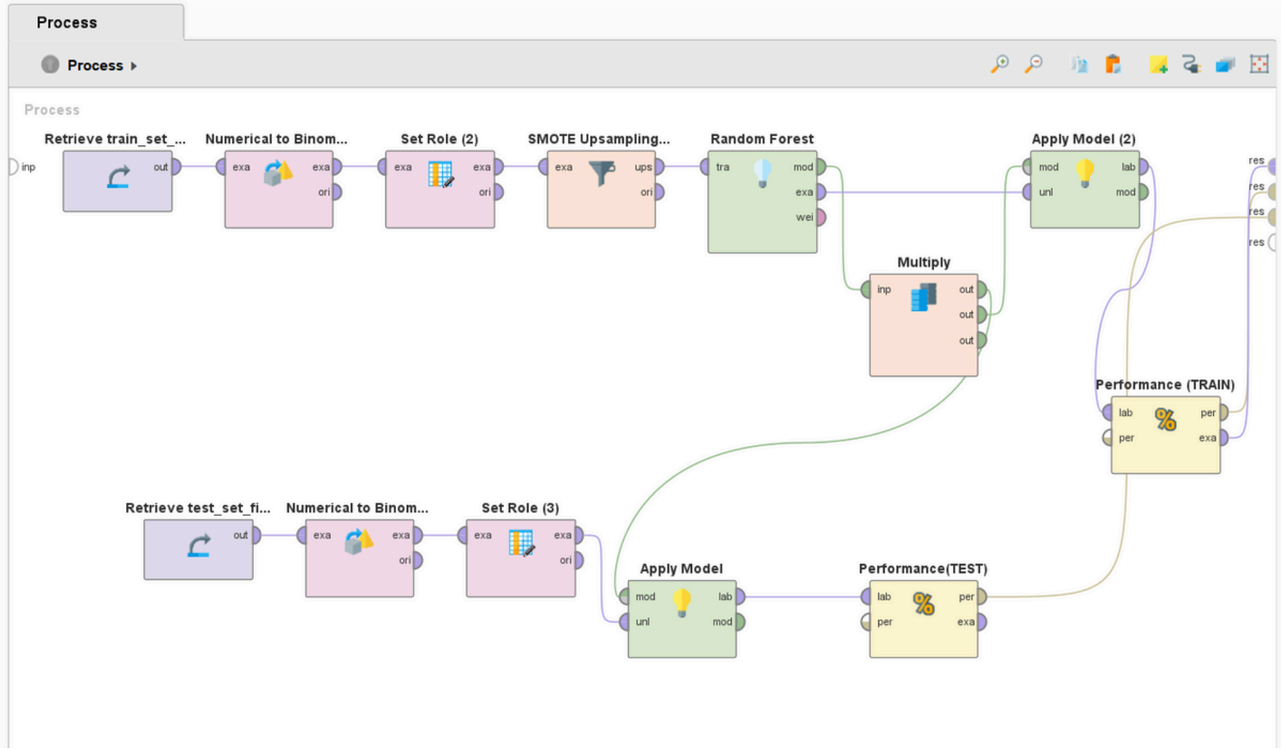
- Decision Trees use all possible feature splits and Random Forest uses a subset of those features. This means Random Forest has **higher generalization capability** than Decision Trees and will **reduce overfitting.**

- In addition, this is a **complex classification** problem suitable for using random forest.

- Random Forest is a **nonlinear model**, which allows it to **learn complex relationships** between variables.

LINH NHI'S MODEL

RANDOM FOREST

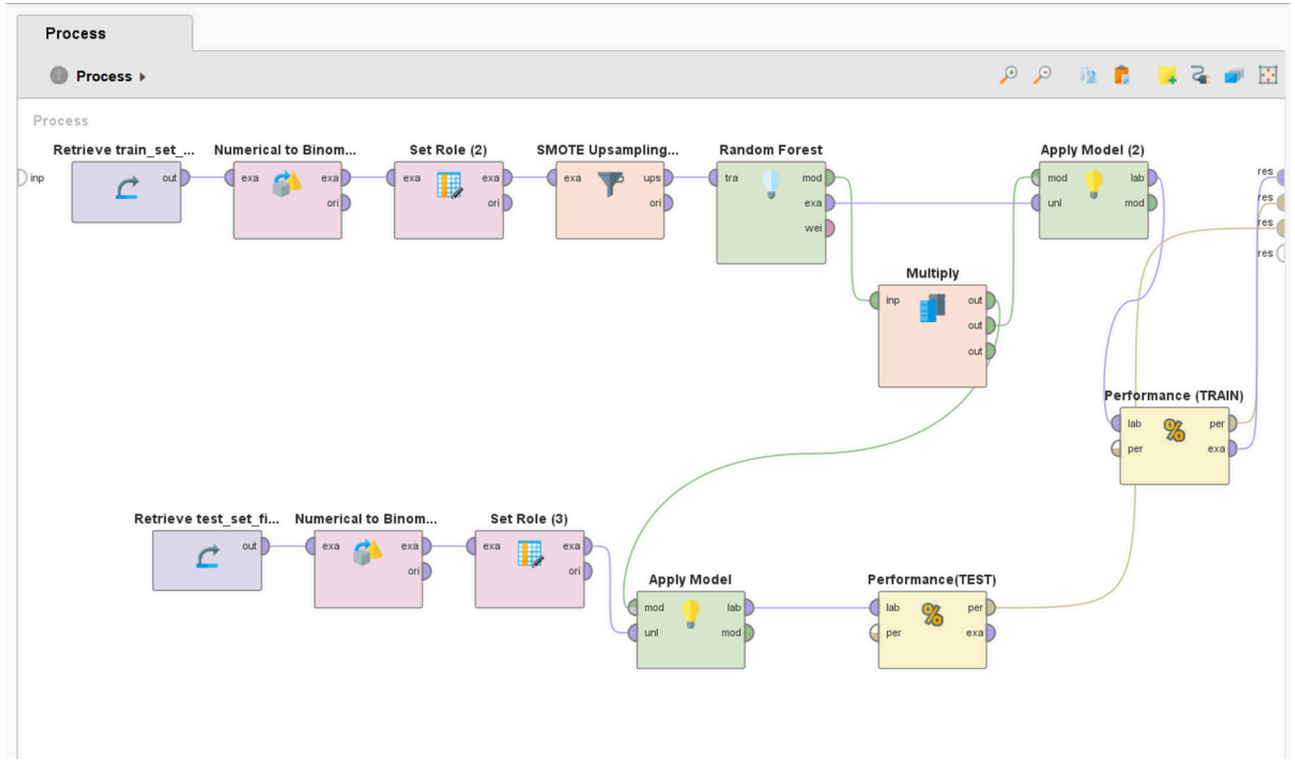


How model works conceptually

- The random forest classifier can be used to solve for regression or classification problems.
- The trees are built independently, with each tree using only a **random subset of attributes.**
- The random forest algorithm is made up of a collection of decision trees, and each tree in the ensemble is comprised of a data sample drawn from a training set with replacement, called the bootstrap sample.
- For a classification task, **a majority vote** - the most frequent categorical variable will yield the predicted class.
- Finally, the **out-of-bag sample** is then used for cross-validation, finalizing that prediction.

LINH NHI'S MODEL

RANDOM FOREST

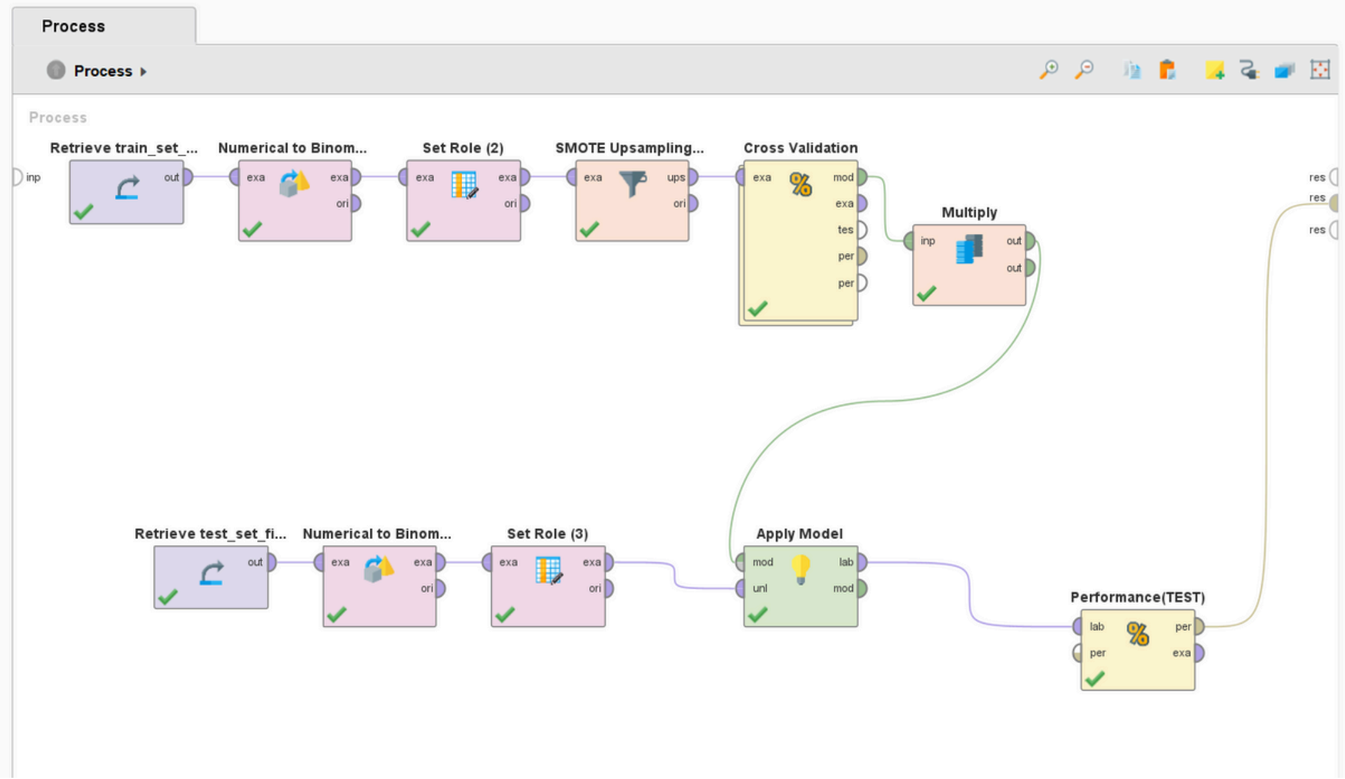


Feature Engineering

- To make the **fairest comparison** among models, the original dataset has been SPLIT into a **TRAIN SET** and a **TEST SET** for model building.
- **Explanation for Numerical to Binomial**: Since the target attribute of the dataset is RESPONSE - whether the customer responds or does not respond, converting to binary data 0 and 1 will optimize classification.
- **Explanation for Set Role**: This is used to set the role of the target variable (RESPONSE) - using other variables to predict the value of this variable. Setting the RESPONSE variable as LABEL designates it as the variable predicting whether there is a RESPONSE or NOT RESPONSE.
- **Explanation for SMOTE Upsampling**: In the original data, for the RESPONSE attribute, class 0 is tending to be more prevalent than class 1. Therefore, if kept as is, there is a risk of overfitting towards more 0s, leading to high accuracy. Therefore, SMOTE must be used to balance class 1 with 0.

LINH NHI'S MODEL

RANDOM FOREST



Try to Train with Cross Validation...

- **The results did not change significantly**, and may even be lower. (lower than ~0.3% compared to the result of Random Forest only)

- This could be due to the **small size of the training set** (as it was previously split 60:40) and the original dataset is not particularly large. This **affects the ability** of Random Forest **to learn complex relationships** in the data.

- With a small dataset, splitting into subsets can lead to a **lack of representation** of certain features or data samples in the training set. This **prevents** the model from **learning important relationships** in the data.

MODEL EVALUATION

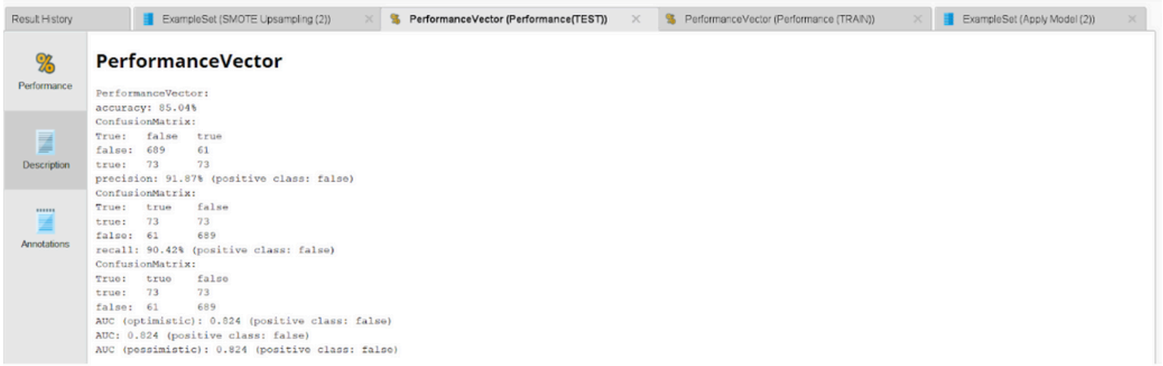
RANDOM FOREST

Performance (Test) - Accuracy: 85.04%

	true false	true true	class precision
pred. false	**689**	**61**	91.87%
pred. true	**73**	**73**	50.00%
class recall	90.42%	54.48%	

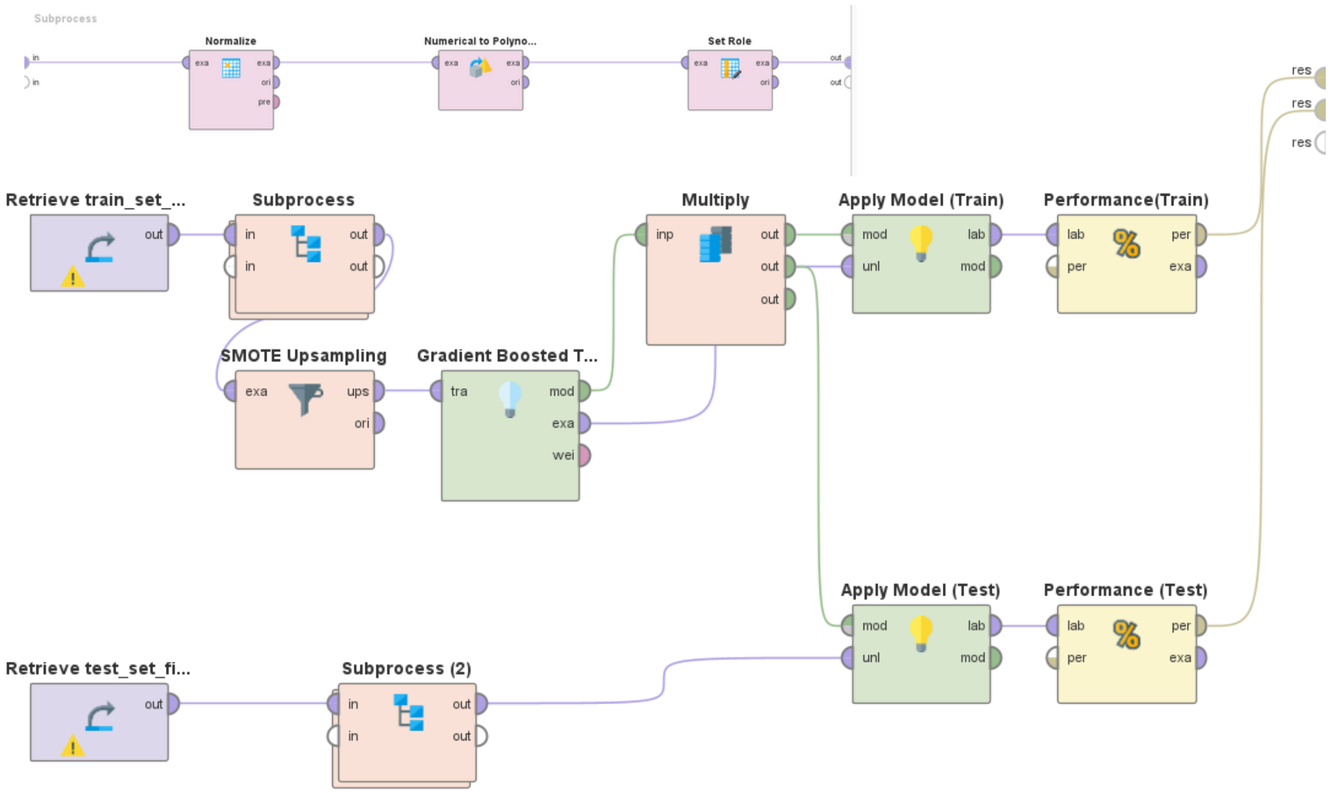
Model's performance evaluation

- The model is **stable** and **does not overfit**.
- The model is **quite good** at detecting **negative cases** (false).
- However, it is **weak** in detecting **positive cases** (true).
- The model has good performance. With an AUC of 0.824, the model is performing well and has the capability to accurately classify the data classes.



LE CHAU'S MODEL

GRADIENT BOOSTED TREES

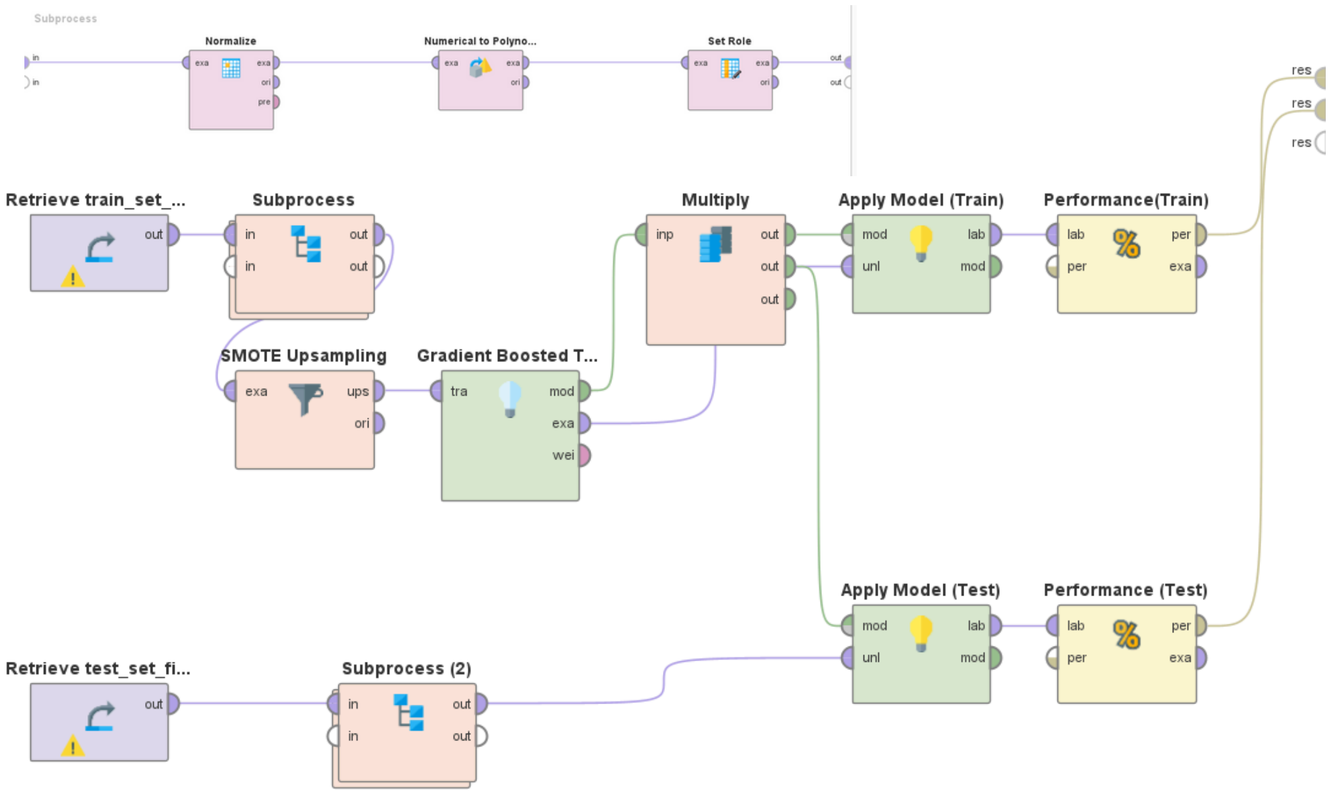


Reason for Choosing Gradient Boosted Trees

- Capable of handling complex nonlinear relationships
- Gradient Boosted Trees can effectively capture nonlinear relationships among features, something that linear models like Logistic Regression struggle to manage efficiently.
- With a dataset of 2,240 rows, Gradient Boosted Trees can be trained effectively without processing time issues.
- This method allows for adjusting weights for minority classes, increasing recall for less common classes. This helps mitigate overfitting when using SMOTE for data balancing.
- Parameters like learning rate and number of trees can be tuned to achieve a balance between precision and recall. It is relatively effective in preventing overfitting.

LE CHAU'S MODEL

GRADIENT BOOSTED TREES

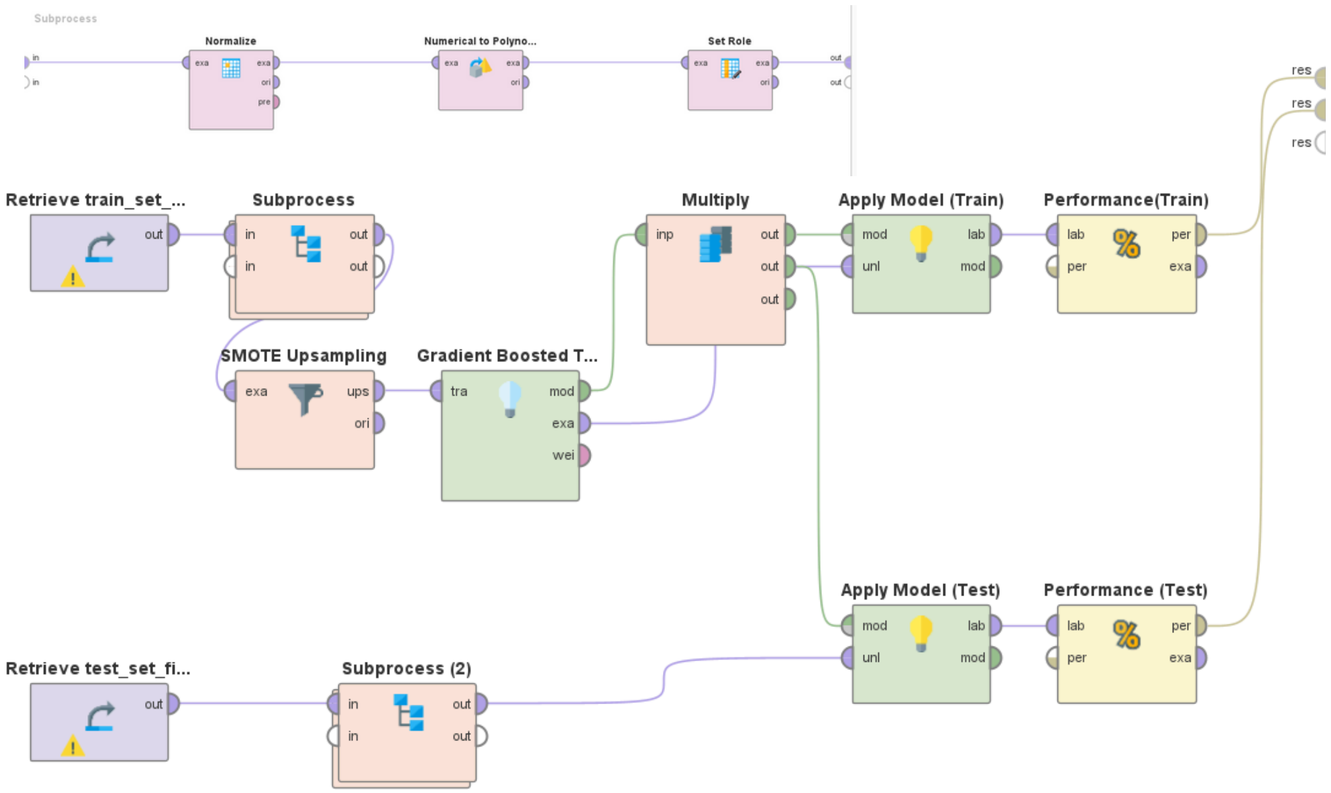


How model works conceptually

- Gradient Boosted Trees is a learning method based on multiple sequential trees, each constructed to gradually improve the model. Each new tree is created to correct the errors of the previous ones, thus making the model more accurate.
- After training the first tree, RapidMiner calculates the error for each data point compared to its actual value.
- Data points with larger errors are assigned higher weights so that the next tree can focus on correcting these errors.
- This process repeats until the maximum number of trees is reached, or until the error reaches an acceptable threshold.

LE CHAU'S MODEL

GRADIENT BOOSTED TREES



Feature Engineering

- **Subprocess and Subprocess (2) include:**
  - Normalize: All attributes except for 'income' due to the presence of relatively large outliers.
  - Numerical to Polynomial: Convert the type of 'response' to apply it in the model.
  - Set Role: Assign 'response' as the label to run and train the model.

- SMOTE Upsampling: In the original dataset, the attribute 'response' has a majority class of 0, which could lead to overfitting by biasing towards 0 and yielding higher accuracy. Therefore, SMOTE is used to generate more instances with a response value of 1.

However, after testing this model, rather than upsampling around 1,000 rows with a response value of 0 to balance the two classes, I found that setting the upsampling size to 450 rows was optimal. This minimizes the noise introduced during upsampling, improving the model's effectiveness.

MODEL EVALUATION

Gradient Boosted Trees

Model's performance evaluation

- The model is **stable** and has a certain degree of **overfit**.
- The model is noticeably better at detecting value 0 compare to detecting value 1
- This is understandable for an **imbalance data set**, the model is adjusted so that the precision and recall of class 1 could be balanced and acceptable

Performance (Train) : 93.20% Accuracy

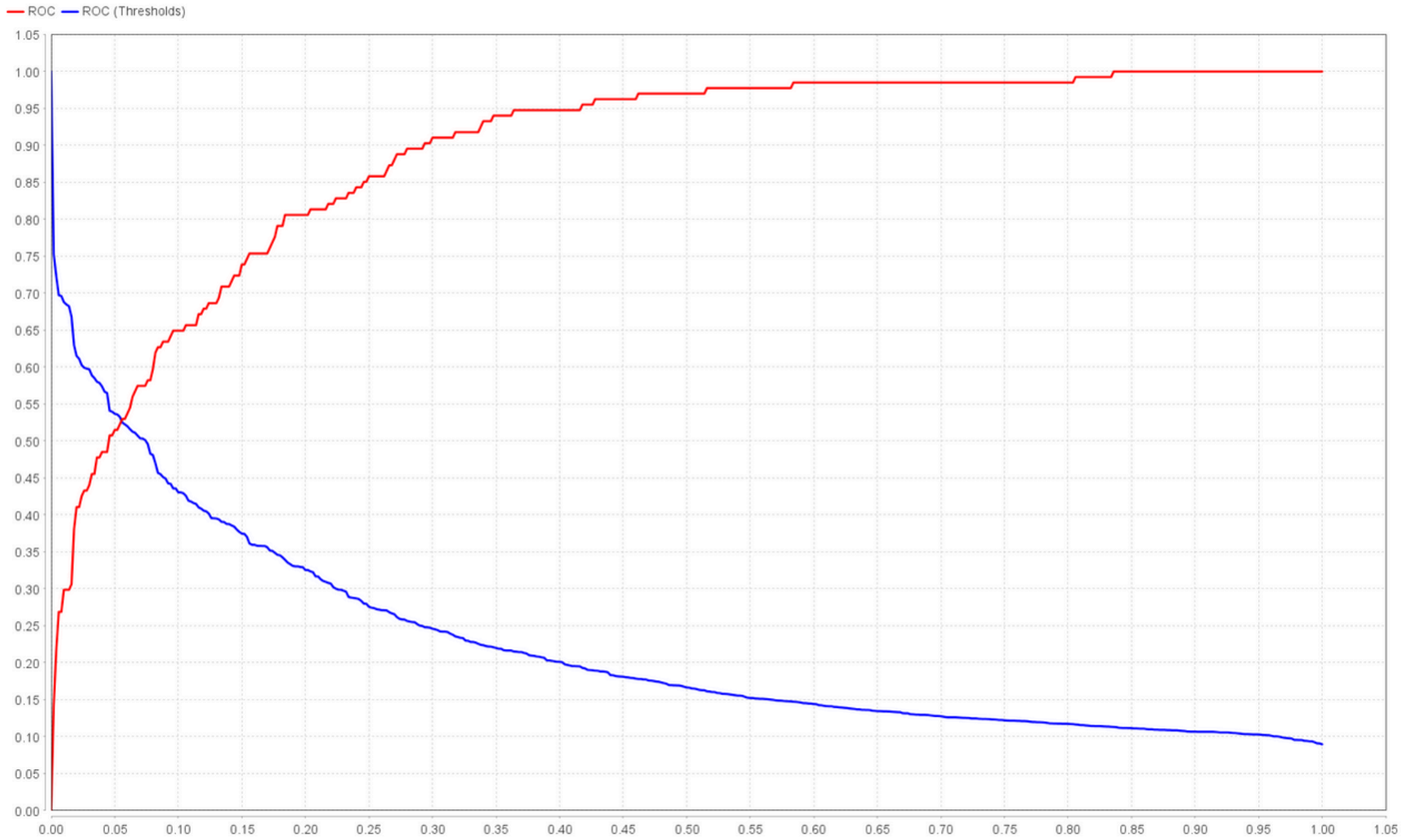
	true 0	true 1	class precision
pred. 0	55	583	91.38%
pred. 1	1089	67	94.20%
class recall	89.69%	95.19%	

Performance (Test) : 87.05% Accuracy

	true 0	true 1	class precision
pred. 0	696	50	93.30%
pred. 1	66	84	56.00%
class recall	91.34%	62.69%	

MODEL EVALUATION

Gradient Boosted Trees



Model's performance evaluation

- An AUC of 0.891 indicates that the model performs well in distinguishing between classes.
- The ROC curve has a steep ascent near the origin, which is favorable. This shows that the model achieves a high sensitivity
- However, there's a gradual rise in the curve as it moves towards the upper right corner, suggest that it struggles with finer distinctions at some thresholds.

DUC MINH'S MODEL CONSTRUCTION

WHY XGBOOST & LGBM SHORTLISTED?

1. Scalability and Robustness for Large Datasets: As the customer base in the grocery industry grows (also after deployment samples would be larger), datasets may scale from medium to large. These two models are robust algorithms designed to handle medium to large-scale data efficiently, making them suitable for this context. Also, they are efficient in handling high-dimensional datasets

2. Effective Handling of Class Imbalance: The target variable `response` exhibits significant class imbalance. This model offers parameters like **`scale_pos_weight`** (XGBoost) and **`is_unbalance`** or **`scale_pos_weight`** (LightGBM) to address this issue, minimizing bias toward the majority class.

3. Prevention of Overfitting through Regularization: Built-in regularization techniques in the model help prevent overfitting, enhancing the model's ability to generalize to new data.

4. Capturing Complex Nonlinear Relationships: As powerful gradient-boosting algorithms, they excel at modeling complex nonlinear patterns in data, often outperforming other classification methods.

5. Efficient Training with Parallel and Distributed Computing: Support for parallel computing and distributed training accelerates the training process, making them efficient for large datasets.

6. Insightful Feature Importance Scores: This model provides feature importance metrics, helping identify the most influential factors contributing to response acceptance rate/propensity, which is valuable for strategic decision-making.

DUC MINH'S MODEL CONSTRUCTION

HOW MODEL WORKS CONCEPTUALLY?

MODEL	XGBoost Classifier	LGBM Classifier
Overview	• Optimized implementation of the **gradient boosting framework** - high performance and efficiency • Ensemble of decision trees sequentially, each new tree aims to correct errors made by the previous.	• A gradient boosting framework that emphasizes **speed** and **memory efficiency** • Designed to handle **large-scale data** and provide **faster training**
How It Works	1. **Level-Wise Tree Growth:** Balance tree expansion across all leaves. 2. **Second-Order Optimization:** Uses both gradients and Hessians for precise optimization. 3. **Regularization:** Incorporates L1 and L2 regularization to prevent overfitting. 4. **Parallel Processing:** Supports parallel computation to speed up training. 5. **Sparse Data Handling:** Efficiently manages missing values and sparse datasets.	1. **Leaf-Wise Tree Growth:** Splits the leaf with the highest loss reduction, leading to deeper trees and faster convergence. 2. **Histogram-Based Algorithms:** Converts continuous features into discrete bins to reduce computational cost 3. **Optimizations:** a. **Gradient-Based One-Side Sampling (GOSS):** Focuses on instances with larger gradients. b. **Exclusive Feature Bundling (EFB):** Bundles mutually exclusive features to reduce dimensionality. 4. **Categorical Feature Support:** directly w/o one-hot encoding 5. **Parallel and GPU Computing**
Use Cases	Suited for various problems where **accuracy is critical**, performing well on **smaller datasets without overfitting**.	Ideal for **large-scale data** where training speed and resource efficiency are paramount.

DUC MINH'S MODEL CONSTRUCTION

BAYESSEARCHCV INTEGRATION FOR HYPERTUNING

Define Parameter Space → **Construct CV & Model** → **Set up BayesSearchCV** → **Fit Model**

- Define **parameter space** for **BayesSearchCV**
- Adjusted for each of the two models

```
param_space_xgb_final = {
    'n_estimators': (100, 200),
    'max_depth': (3, 7),
    'learning_rate': (0.01, 0.1, 'log-uniform'),
    'subsample': (0.8, 1),
    'colsample_bytree': (0.8, 1),
    'scale_pos_weight': (scale_pos_weight_final,
    'reg_alpha': (0, 1),
    'reg_lambda': (0, 1),
    'min_child_weight': (1, 5)
}
```

- Use **StratifiedKFold** Method with **5 folds, shuffle** set = **True**
- Construct **XGBClassifer** framework before fitting

```
cv = StratifiedKFold(n_splits=5,
                     shuffle=True,
                     random_state=42)
xgb_model = XGBClassifier(random_state=42,
                          use_label_encoder=False
                          eval_metric='logloss')
```

- Assigning predefined frameworks to parameters (**model** & **params space**)
- Setting scoring and refit to **"f1"** to balance trade-off

```
# Set up BayesSearchCV
bayes_search_xgb_final = BayesSearchCV(
    estimator=xgb_model,
    search_spaces=param_space_xgb_final,
    scoring={'accuracy': make_scorer(accuracy_
             'f1': make_scorer(f1_score)},  #
    refit='f1',
    cv=cv,
    n_jobs=-1,
    verbose=1,
    random_state=1992
)
```

- Fit **BayesSearchCV** to find best set of hyperparams
- Refit **best estimator** with holdout set for evaluation

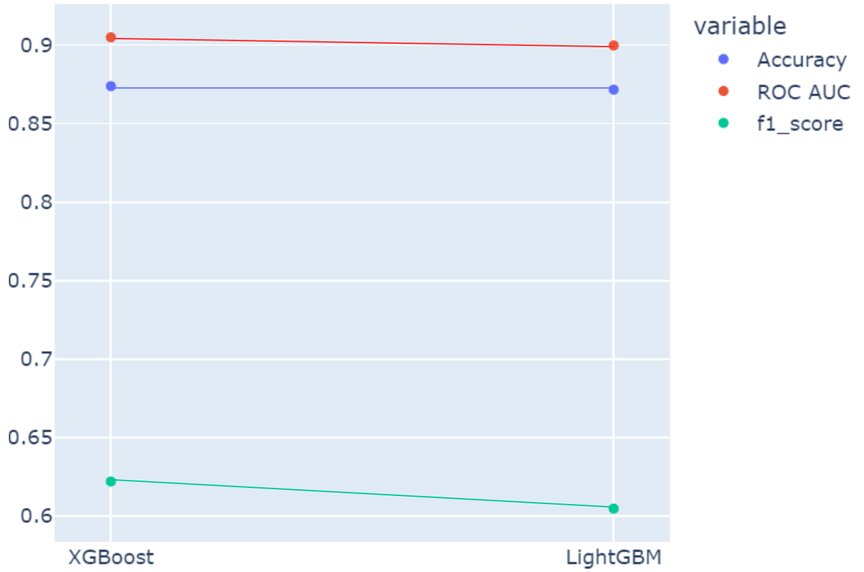
```
# Fit the model
bayes_search_xgb_final.fit(X_train_final, y_train_final)

# Get the best estimator
best_xgb_final = bayes_search_xgb_final.best_estimator_

# Evaluate XGBoost on test set
y_pred_xgb_final = best_xgb_final.predict(X_test_final)
y_pred_proba_xgb_final = best_xgb_final.predict_proba(X_test_final)[:,
```

DUC MINH'S MODEL CONSTRUCTION

XGBOOST VS. LGBM: COMPARISON & SELECTION

Model's Performance Evaluation

Model	Accuracy	ROC AUC	f1_score
XGBoost	0.873884	0.904993	0.622074
LightGBM	0.871652	0.899802	0.604811



- Between the two boosted models, while differences are subtle, **XGBoostRegressor** shows slightly **better performance** compared to LGBM across all metrics
- This may be due to **XGBoost's "depth-wise" tree growth strategy**, which tends to handle smaller datasets like ours better and mitigates overfitting, whereas LGBM's "leaf-wise" approach can be more prone to overfitting in such cases.

XGBoostRegressor Chosen!!

***Note:** f1_score & AUC were chosen as the primary evaluation metric due to its reliability for imbalanced datasets and business objective of ranking customers, respectively*

DUC MINH'S MODEL CONSTRUCTION

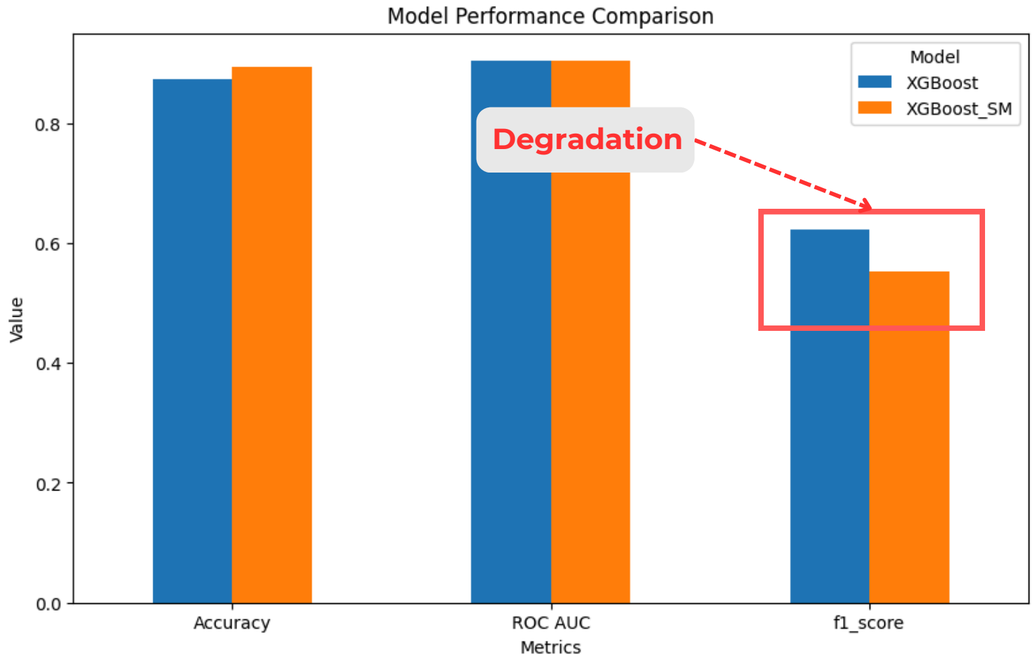
SMOTE UPSAMPLING LOWER XGBOOST PERFORMANCE?

SMOTE UPSAMPLING

```
# use SMOTE Upsampling to tackle class imbalance
from imblearn.over_sampling import SMOTE
smote = SMOTE(random_state=42)

X_train_sm, y_train_sm = smote.fit_resample(X_train_final, y_train_final)
```

CALCULATING SCALE_POS_WEIGHT HYPERPARAMS

```
# Calculate scale_pos_weight to handle class imbalance
negative_count_final = np.sum(y_train_final == 0)
positive_count_final = np.sum(y_train_final == 1)
scale_pos_weight_final = negative_count_final / positive_count_final
```



Inference from Comparison:

- Despite applying SMOTE Upsampling, there seems to be no improvement in model performance, even a small degradation in the eval metrics can be seen

- Therefore, sticking to the `scale_pos_weight` tends to be a more optimal solution for XGBoost Classifier

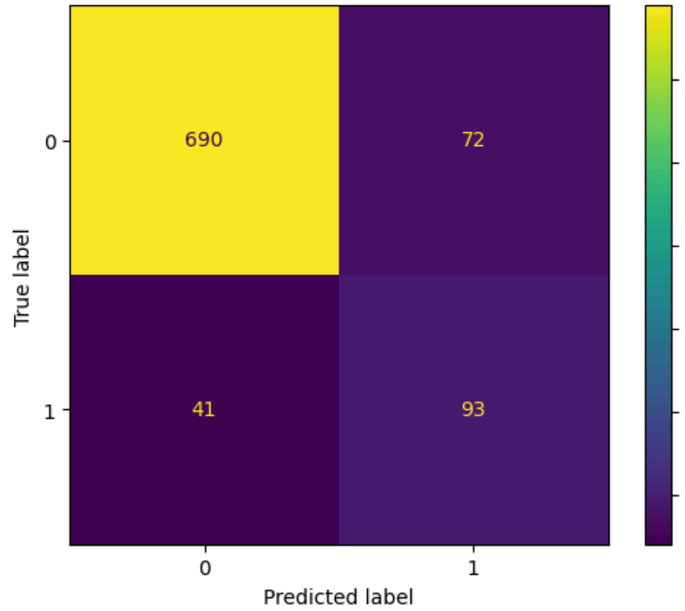
DUC MINH'S MODEL IMPLEMENTATION

PRELIMINARY XGBOOST MODEL EVALUATION

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=0.9999700920619599, device=None,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric='logloss', feature_types=None, gamma=None,
              grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.06349788745284023,
```

- Strong performance in predicting customer response, achieving an overall **accuracy of 87%**. It identifies non-responders (class 0) excellently with **94% precision** and **91% recall**, ensuring accurate classification of most non-responders.
- For responders (class 1), the model achieves **56% precision,** meaning a moderate portion of customers flagged as likely responders are correctly identified. With a **69% recall** and **f1 score of 62%**, there's room for improvement in capturing potential responders.

Overall, the model handles imbalanced data well, and slight optimization in reducing false negatives could further enhance its effectiveness in targeting customers.



TESTING SET PERFORMANCE:

	precision	recall	f1-score	support
0	0.94	0.91	0.92	762
1	0.56	0.69	0.62	134
accuracy			0.87	896
macro avg	0.75	0.80	0.77	896
weighted avg	0.89	0.87	0.88	896

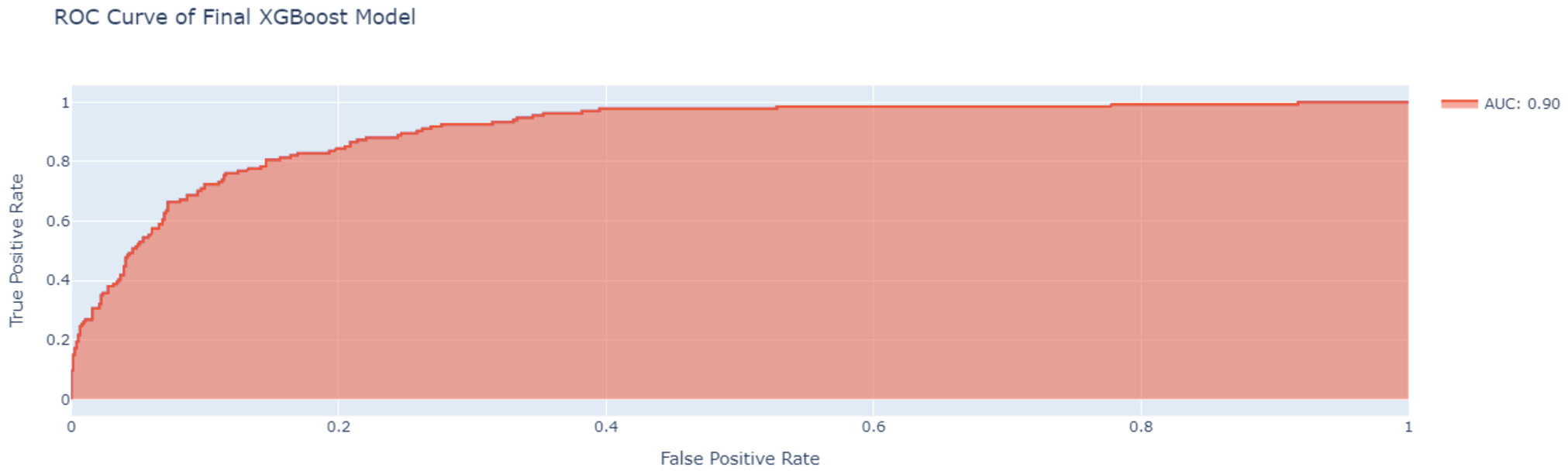
DUC MINH'S MODEL IMPLEMENTATION

PRELIMINARY XGBOOST MODEL EVALUATION

Evaluation Metrics (ROC Curve)

Note: AUC and Precision-Recall Score are important metrics to evaluate models with Class Imbalance

The ROC curve for the final XGBoost model shows impressive classification performance, with an AUC score of 0.90. The curve being close to the top-left corner means the model does a great job distinguishing between respondents and non-respondents even in an imbalanced dataset



05

Model Evaluation & Comparison

Models comparison

Strengths and Weaknesses of each model

Model Comparison

	Accuracy	Precision (Class 1)	Recall (Class 1)	F1 score (Class 1)	ROC-AUC (Class 1)
XGBoosting	87.38%	56%	69%	62%	0.9
Gradient Booted Trees	87.05%	56.00%	62.69%	59.16%	0.891
Random Forest	85.04%	50.00%	54.48%	52.14%	0.824
Logistic Regression	86.05%	52.83%	62.69%	57.34%	0.878

MODELS' STRENGTHS AND WEAKNESSES

MODEL	STRENGTHS	WEAKNESSES
XGBoosting (Python)	• Advanced Hyperparameter Tuning • Customization (customize logloss, focus on minority class, ...)	• Resource-Intensive, slower than other models while working with small and medium dataset • Complexity in Tuning
Gradient Boosted Trees *(Rapidminer)*	• Good speed for small & medium dataset • Basic, easy to use tuning options	• Limited Fine-Tuning • Limited Advanced Techniques • Become noticeably slower as dataset size increase

MODELS' STRENGTHS AND WEAKNESSES

MODEL	STRENGTHS	WEAKNESSES
Random Forest (Rapid Miner)	• Ability to handle multiple input variables (features) • Resilience to noise and outliers • Capability to assess feature importance to predict target attribute	• Interpretability - Random Forest models can be complex and difficult to explain in detail. • Computational resource requirements - Training a Random Forest can be computationally intensive, especially on large datasets.
Logistic Regression (Rapid Miner)	• Highly interpretable; clear feature impact • Fast, computationally efficient, even with larger datasets • Tolerant of minor missing data, simple preprocessing	• Limited in handling complex, nonlinear relationships • Struggles with the imbalanced nature of the dataset • Struggles with very high-dimensional or sparse datasets

06

Business Impact

Key Insight: Feature Importance

Propensity Score

Key Insights

Feature Importance Grouped by Key Aspects

01. Campaign Engagement

- The most influential predictor is **total campaigns accepted** (total_cmp_accepted), showing that customer response to campaigns strongly impacts the target outcome.
- Other campaign-related features, like **accepted_cmp2** and **accepted_cmp4**, also rank highly, reinforcing that engagement with specific campaigns is essential in predicting customer behavior.

02. Recency and Activity

Activity recency (recency) and **total days active** (total_days_active) are crucial indicators, highlighting the importance of recent customer engagement and overall interaction duration on the model's predictions.

03. Demographic Attributes

- **Demographic** factors provide significant predictive power, with **marital status (Married), income, education, and having children or teens (have_kids_or_teen)** ranking as important predictors.
- Together, these reflect the role of personal, economic, and family characteristics in shaping customer behavior, with age also contributing moderately.

04. Purchasing Behavior

- **Purchase history** metrics, including **store purchases** (num_store_purchases), **catalog purchases** (num_catalog_purchases), and **web visit frequency** (num_web_visits_month), along with **specific product spending** (e.g., meat and golds), have substantial predictive value.
- This set of features **underscores** that both the **frequency and diversity** of purchasing habits are vital in predicting outcomes.

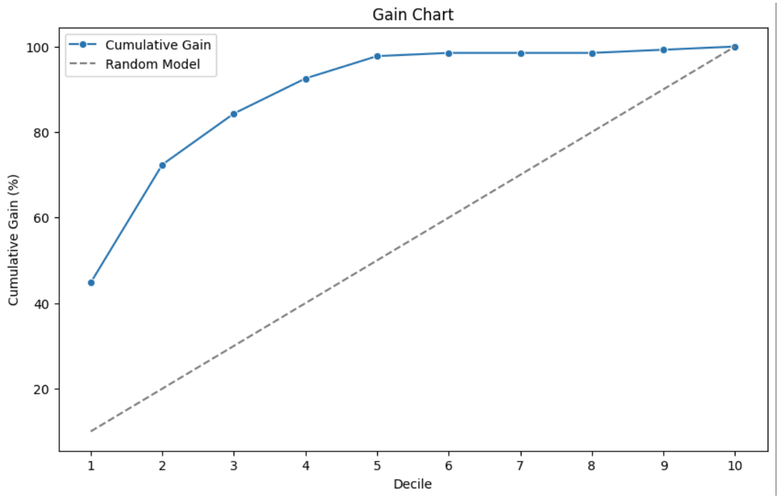
Propensity Score

Qualified Users Segmentation by Propensity Score

decile	Propensity_Scorecard	No_of_Observations	Event_Rate (%)	Cumulative_Response_Rate (%)	Cumulative_Gain (%)
1	7472 - 9953	90	66.67	6.7	44.78
2	4546 - 7466	90	41.11	10.83	72.39
3	2560 - 4533	89	17.98	12.61	84.33
4	1522 - 2527	90	12.22	13.84	92.54
5	855 - 1493	89	7.87	14.62	97.76
6	481 - 850	90	1.11	14.73	98.51
7	257 - 475	89	0.0	14.73	98.51
8	137 - 255	90	0.0	14.73	98.51
9	52 - 137	89	1.12	14.84	99.25
10	3 - 52	90	1.11	14.96	100.0

Most potential Lead

98% Client



When selecting the **top 10% of customers:**

- **Cum_resp_rate = 6.7%,** meaning the Acceptance rate = 6.7%
- **Gain = 44.78,** 44.78% of actual converted customers were obtained.

When selecting the **top 20% of customers:**

- **Cum_resp_rate = 10.83%,** meaning the Acceptance rate = 10.83%
- **Gain = 72.39,** 72.39% of actual converted customers were obtained.

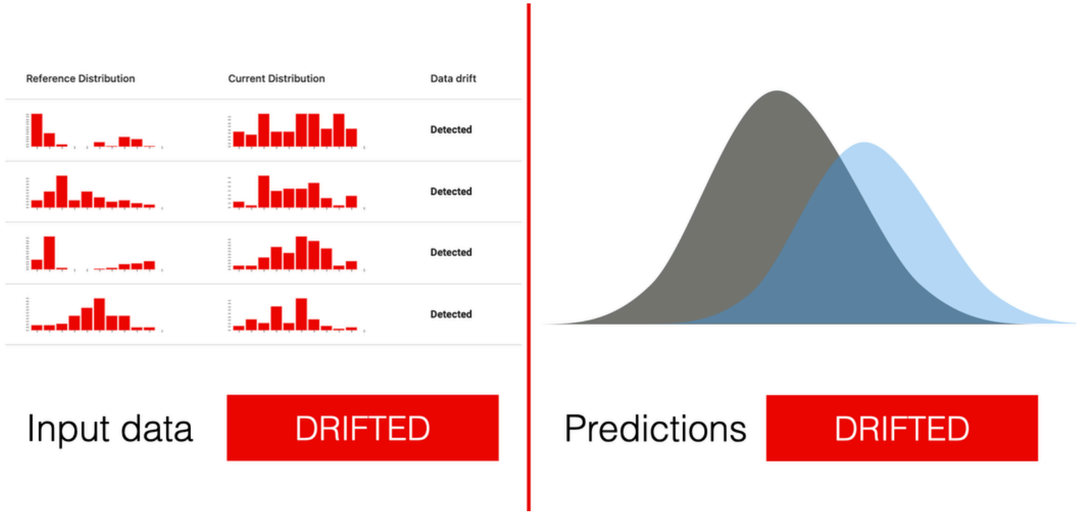
07

Further Discussion

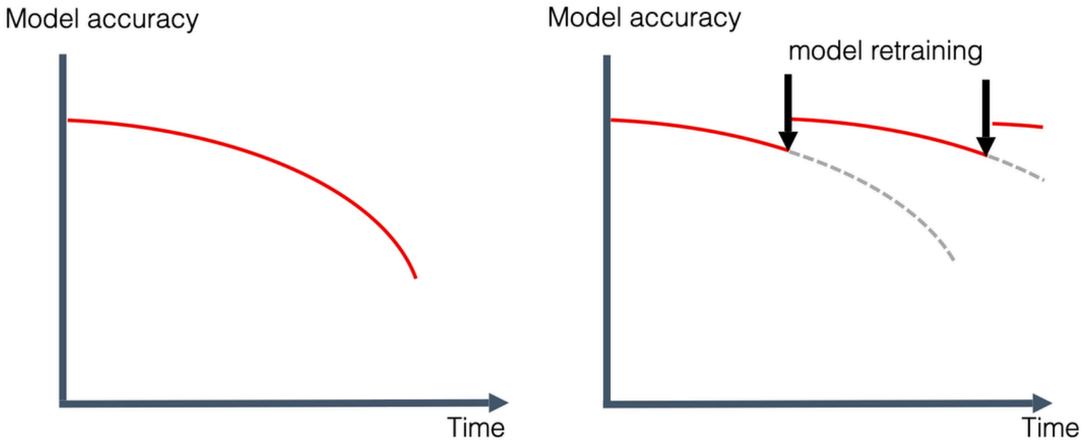
Models Monitoring (CI/CD)

CI/CD

Problems



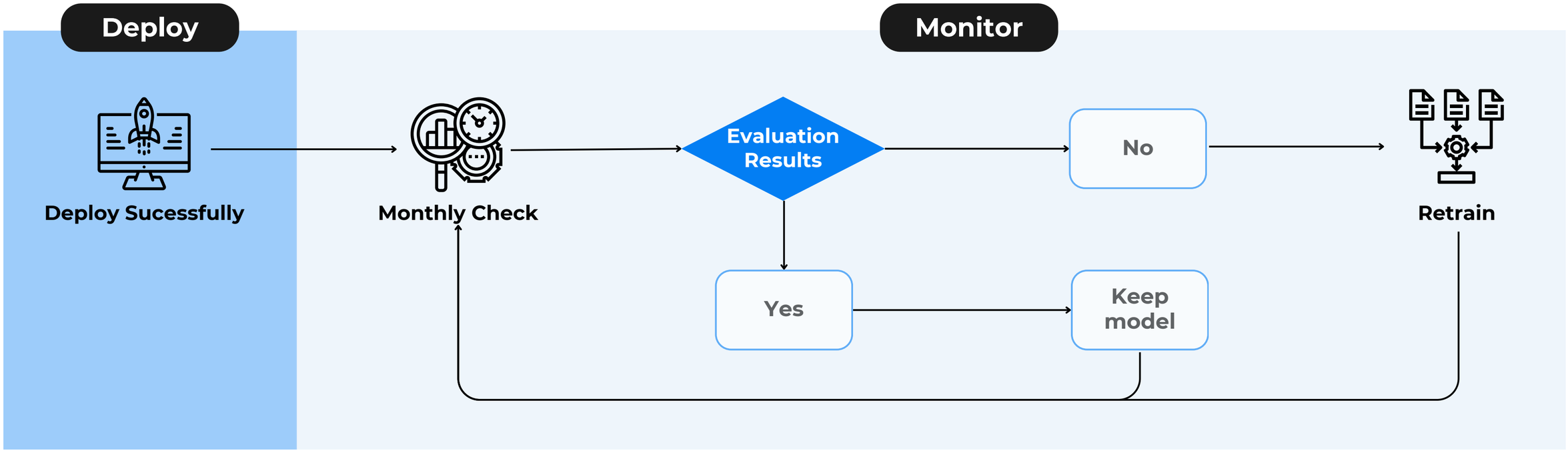
During the actual implementation process, customer behavior will **certainly change** related to purchasing behavior, product usage habits, and preferences, etc.



This results in the **model's effectiveness decreasing over time**, as the model is not able to predict based on this change in behavior.

CI/CD

Problems



Timeline

Timeline: 1 month after deploying model → Check the model **weekly** on **Saturday**

CI/CD

According to the traditional method

PSI

Monitor the PSI index of model input variables compared to reality

≥ 0.25

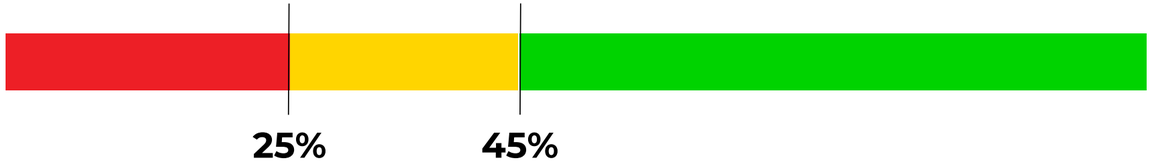
Customer behavior has changed markedly
on 50% of variables or on model results
=> Need to consider rebuilding the model

Gini

Monitor model quality during actual deployment via Gini

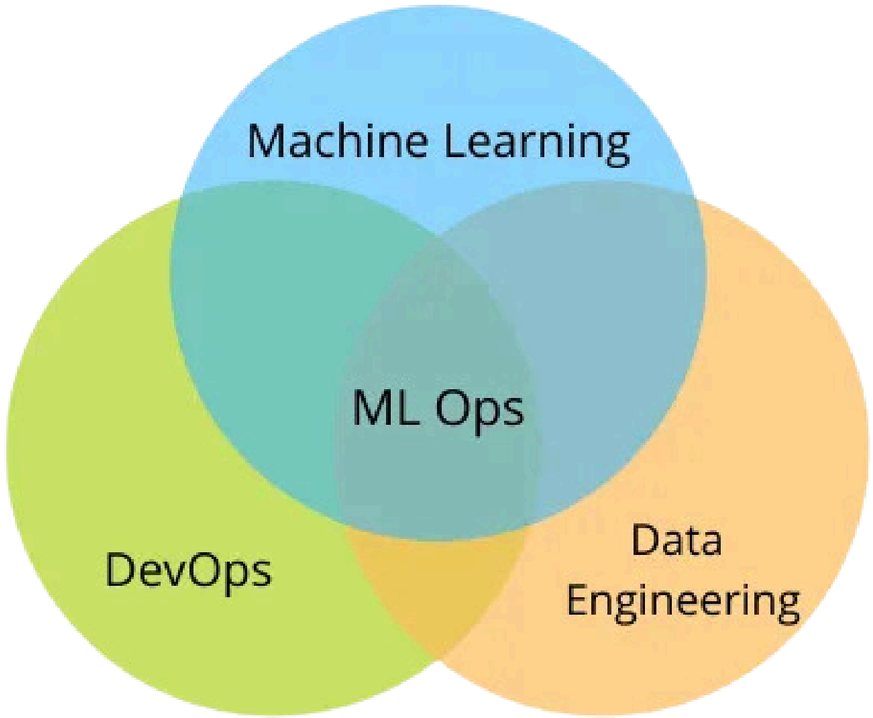
< 25%

Need to rebuild the model



CI/CD

According to modern methods - MLOps

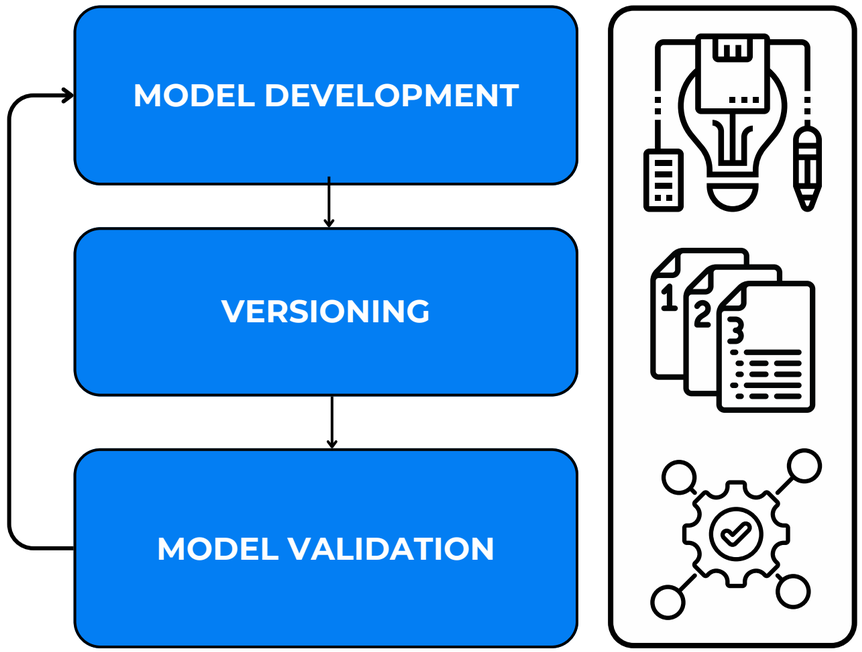


MLOps (Machine Learning Operations) is a combination of **Machine Learning** and **Operations** aimed at deploying and maintaining Machine Learning systems in production reliably and efficiently.

MLOps brings together approaches that automate the lifecycle of machine learning algorithms in production at all steps of building a Machine Learning system, from initial **model training to deployment and retraining based on new data.**

CI/CD

According to modern methods - MLOps

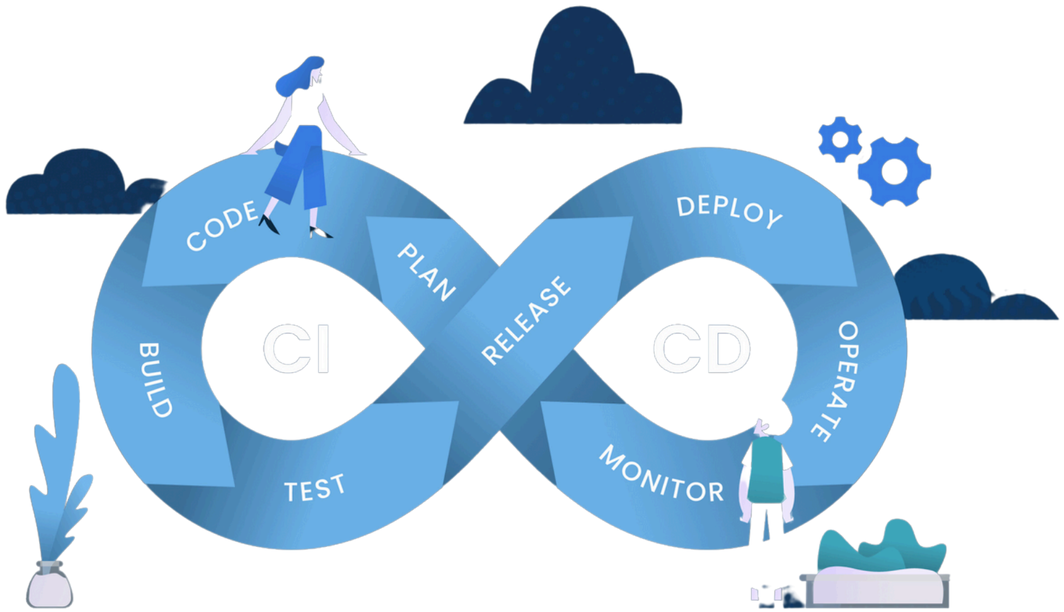


- **Model development:** selecting, training, and validating machine learning models.
- **Versioning:** Both source code and data are versioned to ensure reproducibility.
- **Model validation:** Models must be validated using different metrics and tested with unseen data to ensure they generalize well.

CI/CD

According to modern methods - MLOps

- **Continuous Integration (CI):** automatically builds and tests machine learning pipelines every time there is a change in the source code.
- **Continuous Deployment (CD):** automatically deploys models to production environments every time they are ready, ensuring that the latest model is always available to provide predictions.

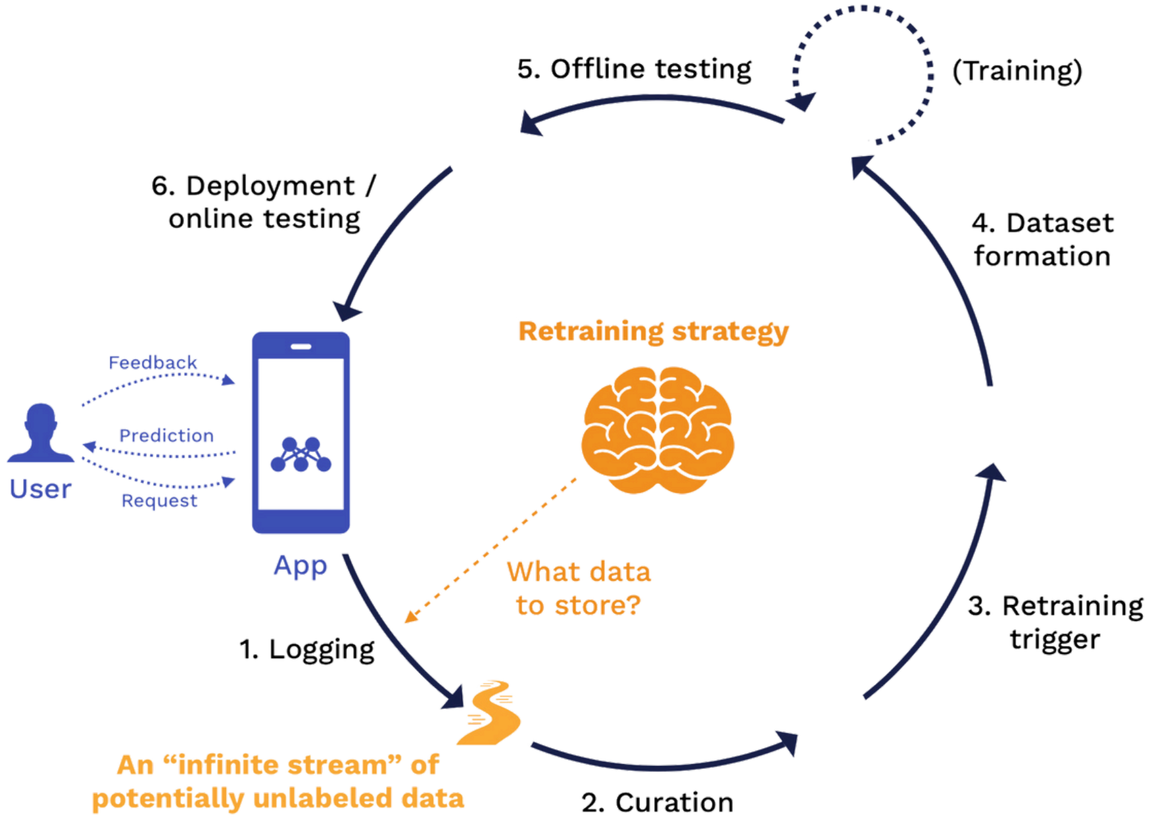


CI/CD

According to modern methods - MLOps

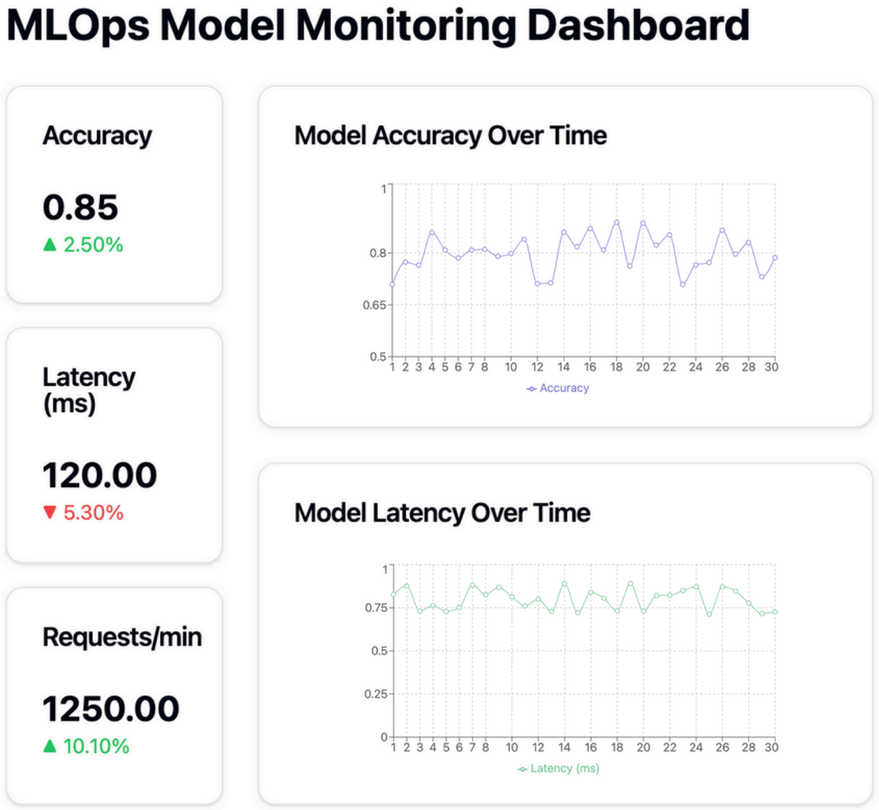
As new data becomes available, models may **need to be retrained** to **maintain their accuracy.**

Automated building ensures that models are retrained automatically when certain conditions are met.



CI/CD

According to modern methods - MLOps



Based on the MLOps system, build a dashboard to track model indicators

Accuracy

System latency & load

Model performance over time

Monitor and proactively evaluate model quality and corresponding issues

Thank you!

CRM Data Analyst Case