**Super Market Campaign analysis**

Using Pyspark

horizontal line

# 

# **Introduction**

All You Need Supermarket is preparing for a year-end sale and plans to introduce a new offer exclusively for existing customers. The offer, called gold membership, provides a 20% discount on all purchases for a price of $499, which is usually sold at $999. To promote the offer, the supermarket plans to conduct a phone call campaign targeting its existing customers. The management believes that building a predictive model to identify potential customers who are more likely to purchase the offer can help reduce campaign costs. They have provided data gathered from last year's campaign to build the predictive model. This involves analyzing the provided data and building a predictive model using machine learning techniques. Additionally, the model needs to be production-ready using pipelines. The end goal is to provide the supermarket with insights that can help them identify potential customers who are likely to purchase the gold membership offer, reducing campaign costs and increasing sales

## **Objective**

The objective of this project is to identify the factors that influence a customer's response to the gold membership offer and predict the probability of a customer giving a positive response.

### **Research Questions to answer**

1. Analyze the factors that would influence customers response
2. Do customers have any complaints for the company?
3. Can the company target customers who buy premium products?
4. Can the company provide offers to any specific category of customers?
5. Building model to predict the probability of a customer giving positive response

## 

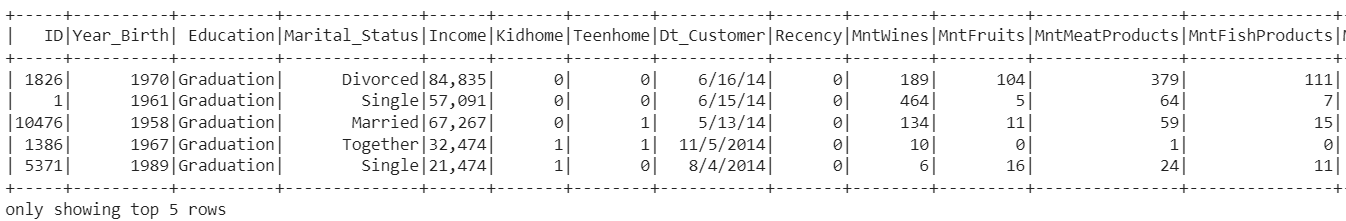
## **Milestones**

1. Data Loading and exploration using spark
2. EDA and Visualization
3. Data Preprocessing and Feature Engineering
4. Model Training & Evaluation
5. Conclusion
6. Data Analysis Tableau

### **Data Loading and Exploration using Spark**

We will need a spark session to load the data so the session can be created using the below code.

| # Importing and configuration of Spark Session from pyspark.sql import SparkSession spark = SparkSession.builder.appName("Super Market Campaign analysis with PySpark").getOrCreate() |
| --- |

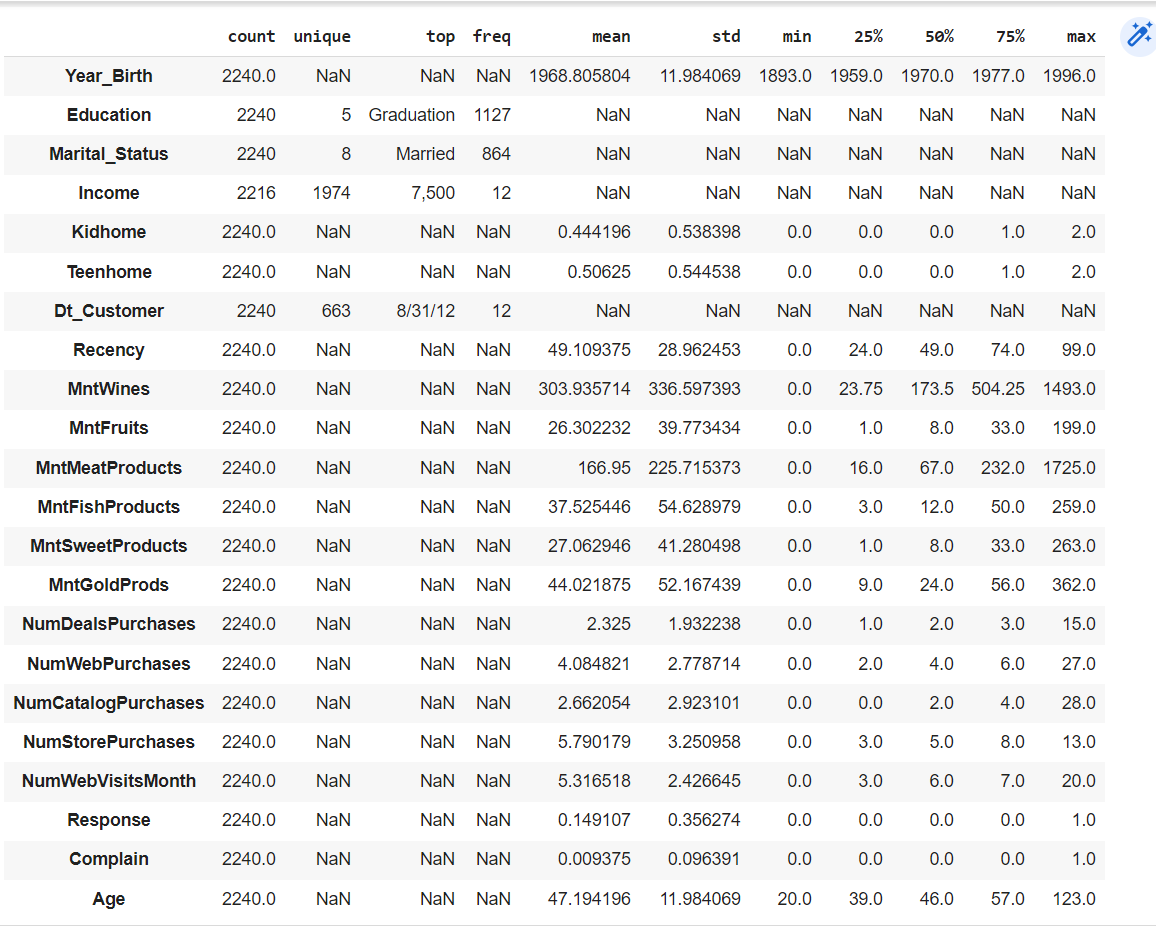
Data exploration can be done using the show() function and the printschema() can be used to show the schema of the data. The full table would be difficult to visualize so few columns can be seen in the below image  


The printschema() function shows the schema of the data

| root  |-- ID: integer (nullable = true)  |-- Year\_Birth: integer (nullable = true)  |-- Education: string (nullable = true)  |-- Marital\_Status: string (nullable = true)  |-- Income: string (nullable = true)  |-- Kidhome: integer (nullable = true)  |-- Teenhome: integer (nullable = true)  |-- Dt\_Customer: string (nullable = true)  |-- Recency: integer (nullable = true)  |-- MntWines: integer (nullable = true)  |-- MntFruits: integer (nullable = true)  |-- MntMeatProducts: integer (nullable = true)  |-- MntFishProducts: integer (nullable = true)  |-- MntSweetProducts: integer (nullable = true)  |-- MntGoldProds: integer (nullable = true)  |-- NumDealsPurchases: integer (nullable = true)  |-- NumWebPurchases: integer (nullable = true)  |-- NumCatalogPurchases: integer (nullable = true)  |-- NumStorePurchases: integer (nullable = true)  |-- NumWebVisitsMonth: integer (nullable = true)  |-- Response: integer (nullable = true)  |-- Complain: integer (nullable = true) |
| --- |

We can also observe that there are about 2240 rows and 22 columns of data. Also while using the below code to check for null values, we have observed that there are only 24 null values in the Income column and there are no values in any other columns.

| data.filter(data[i].isNull()).count() |
| --- |

While using the describe() function on the data, we have obtained the following table  


Observations,

* The Year\_Birth column contains values ranging from 1893 to 1996, indicating a wide age range among customers.
* Some users have a birth year of 1900 or earlier, which is unlikely given that the campaign being analyzed took place in 2016. This could be a data reporting error that requires further investigation.
* The Income column has a maximum value of 666666, which is much higher than the mean and may represent an outlier in the data.
* Certain columns such as Income, MntFruits, MntWines, MntMeatProducts, MntFishProducts, and MntSweetProducts appear to have outliers on the high end, as indicated by the large difference between the 75th percentile and maximum values.
* The Recency column has a mean and median value of approximately 49 days, suggesting that most customers made a purchase relatively recently.
* In the last two years, customers spent the highest mean amount on wines (approximately 304 units), followed by meat products (approximately 167 units).

We have also observed that there are no null values in the data.

To calculate the age of customers in the dataset, we subtract the year 2016 as the variables cover the last two years, and the dataset only includes customers registered until 2014. However, before performing this calculation, we must first convert the string values to dates to allow for subtraction. So we have used the below code for the conversion

| from pyspark.sql.functions import year, expr data = data.withColumn("Age", expr("2016 - year(to\_date(CAST(Year\_Birth AS STRING), 'yyyy'))")) data.sort("Age").select("Age") |
| --- |

Upon analysis, it appears that the Year\_Birth values for three observations are likely incorrect, as they indicate extremely high ages. Replacing these values with other data points may not be a suitable solution. As such, we will remove these three observations from the dataset.

Observations from categorical columns,

* In the context of education, the terms "2n cycle" and "Master" are used interchangeably. Thus, we could merge these two categories for ease of analysis.
* The "Marital Status" variable contains several categories, including "Alone," "Absurd," and "YOLO." These could be combined with the "Single" and "Together" categories to simplify the variable, leaving only "Married" and "Unmarried" categories.
* Over the past two years, only 21 customers have filed complaints.
* The dataset contains 1906 observations for the 0 class and 334 observations for class 1.
* The customer registration data spans only three years.

We will also add a new column showing the total amount spent and this can also be observed below

| data = data.selectExpr("\*", "MntWines + MntFruits + MntMeatProducts + MntFishProducts + MntSweetProducts + MntGoldProds as Total\_Amount\_Spent") |
| --- |

### **EDA and Visualization**

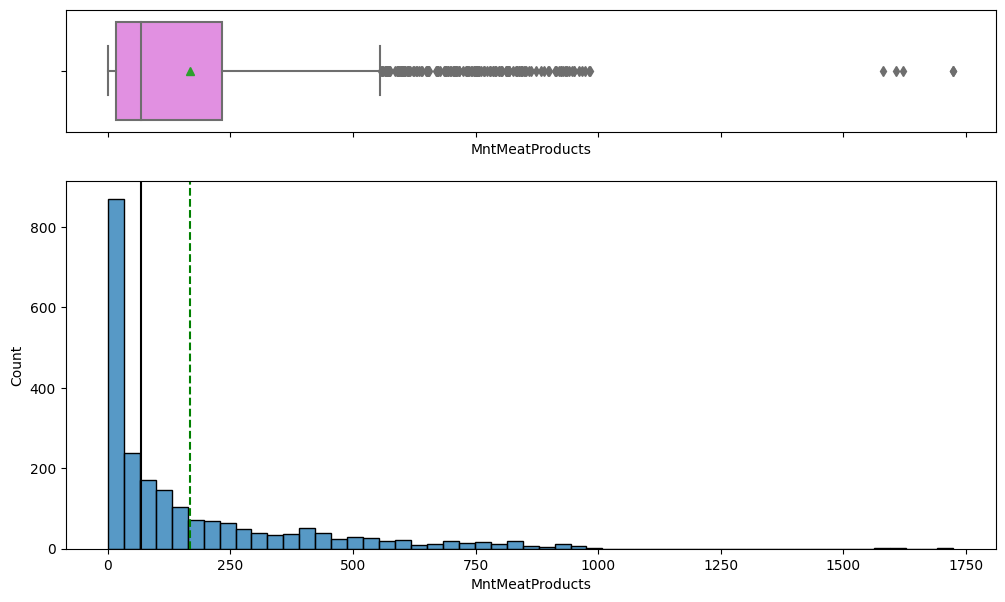
Exploratory Data Analysis (EDA) is a crucial step in the data analysis process, as it helps to gain a deeper understanding of the dataset. It involves examining and visualizing the data to uncover patterns, relationships, and anomalies that may be present.

During EDA, statistical and graphical techniques are used to explore the data, such as histograms, scatter plots, and correlation matrices. This allows for the identification of potential outliers, missing values, and inconsistencies in the data.

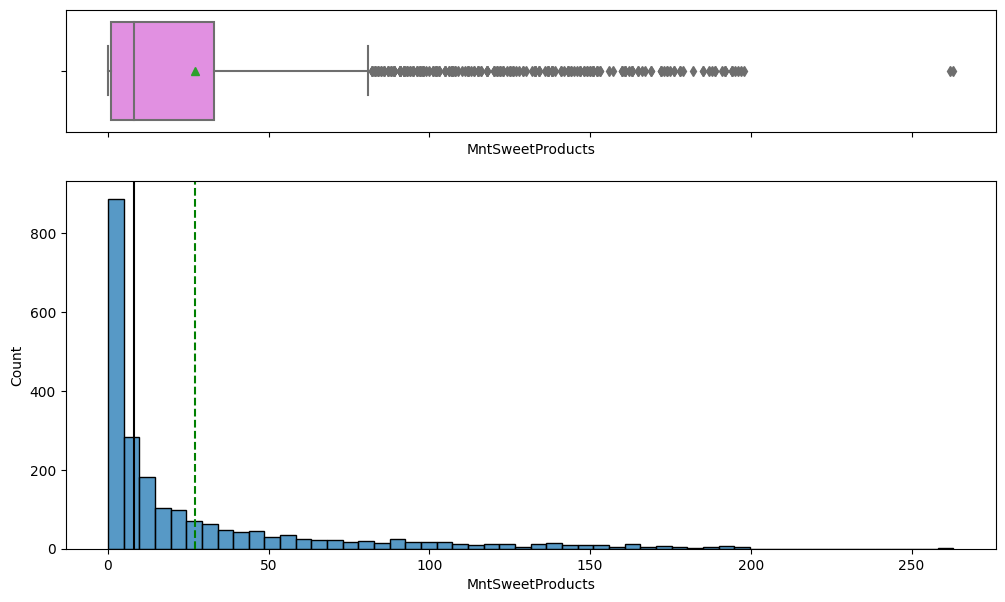
EDA is an iterative process, as the insights gained from initial analysis may lead to the discovery of new questions, requiring further investigation. This process helps to refine hypotheses and inform the choice of statistical models to use for subsequent analysis.

So now we will perform some EDA on this data. Pyspark dataframe is not suited for plotting visualizations so we would convert this to pandas dataframe using toPandas() function and then perform Exploratory Data Analysis.

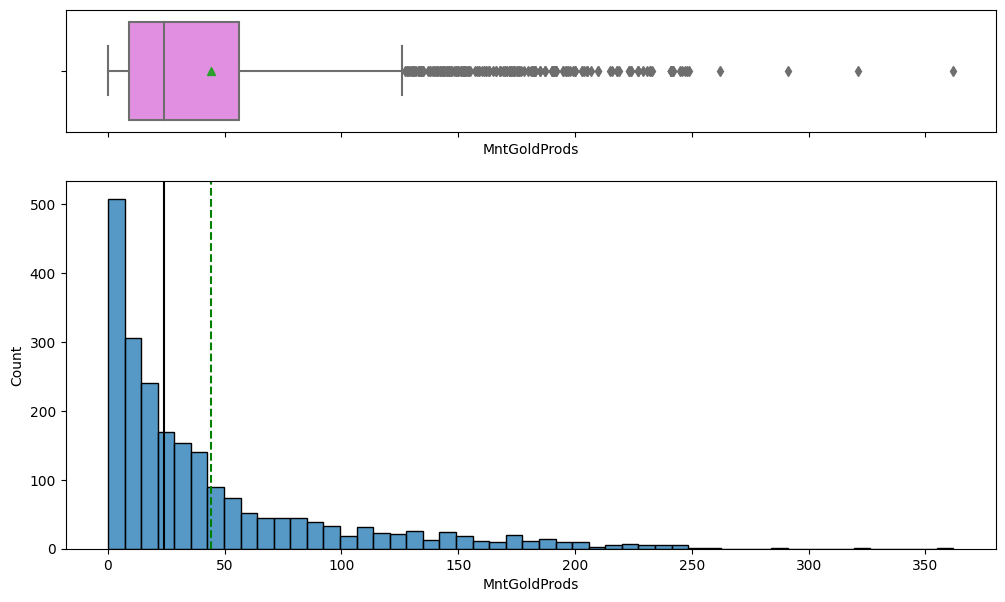
Univariate analysis,



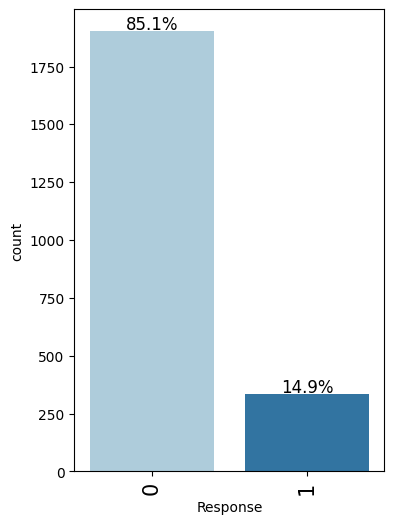
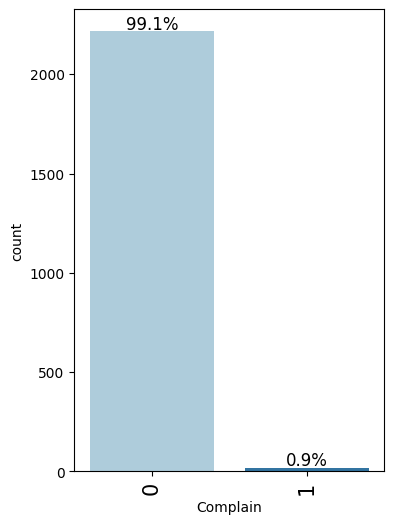
* The data on the amount spent on meat products appears to be highly skewed towards the right, indicating a few high spenders in the dataset.
* Some observations in this variable may be deemed as outliers due to their significant deviation from the rest of the data.
* To address potential outliers in the dataset, one potential solution would be to cap the variable at the next highest value, effectively limiting the impact of extreme values on subsequent analysis.



* The data distribution for the expenditure on sweet products is skewed towards the right side.
* It is worth noting that there is an observation on the extreme right side of the distribution, which appears to be an outlier.
* Although these extreme values may represent genuine market trends, we may opt to limit or cap them to avoid potential distortion of the analysis results.
* It's important to exercise caution and consider the impact of such a decision on the overall data analysis, as it may affect the accuracy and validity of the findings.
* Therefore, we will carefully evaluate the potential benefits and drawbacks of capping the extreme values before making a final decision.

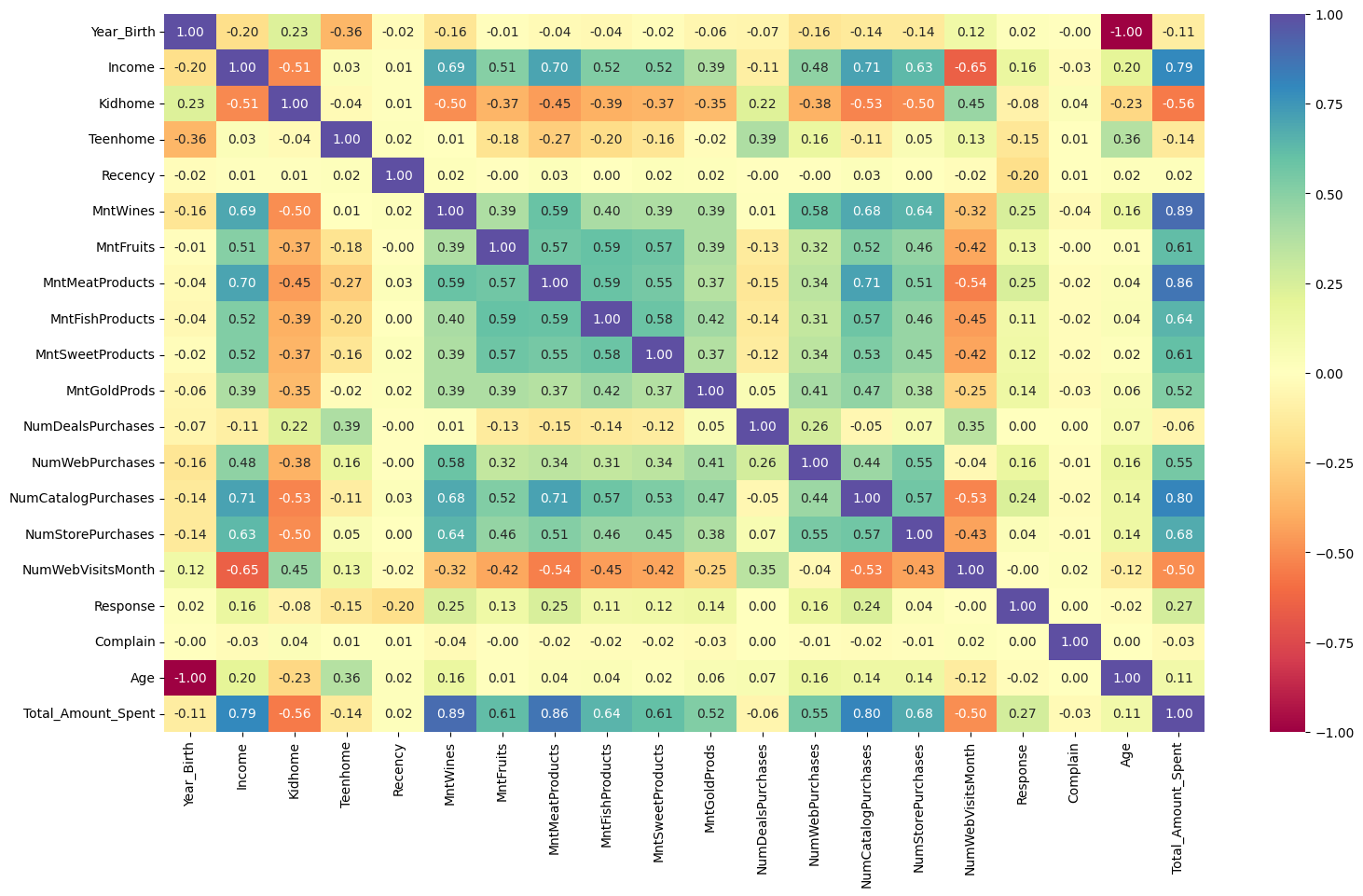


* The distribution of the amount spent on gold products exhibits a right-skewed pattern, indicating that most of the observations lie towards the lower end of the range.
* Some data points in the amount spent on gold products stand out as outliers, which could be attributed to certain market trends. While it is not advisable to remove all such data points, we may consider capping some of the extreme values to prevent them from unduly influencing the analysis.
* It is worth noting that these outliers can provide valuable insights into the market behavior and should not be dismissed outright.
* By capping the extreme values, we can still obtain a representative picture of the central tendency and spread of the data, without compromising the integrity of the analysis.
* It is important to exercise caution and apply appropriate statistical techniques when dealing with skewed distributions and outliers, as they can significantly affect the conclusions drawn from the data.

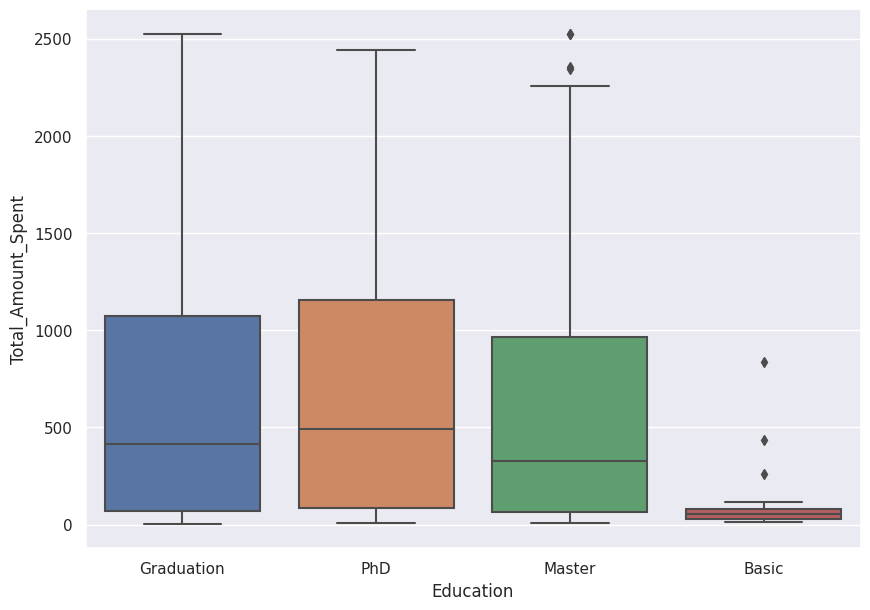


* Nearly all of the customers, around 99%, did not register any complaints within the previous 24 months. It is possible that this is due to the company offering satisfactory services or it could also indicate that there are limited channels available for customers to provide feedback.
* Around 85% of the customers responded with a negative answer in the previous campaign, indicating that the target variable's class distribution is imbalanced. This means that we have a relatively small percentage of observations, approximately 15%, where the response was affirmative.

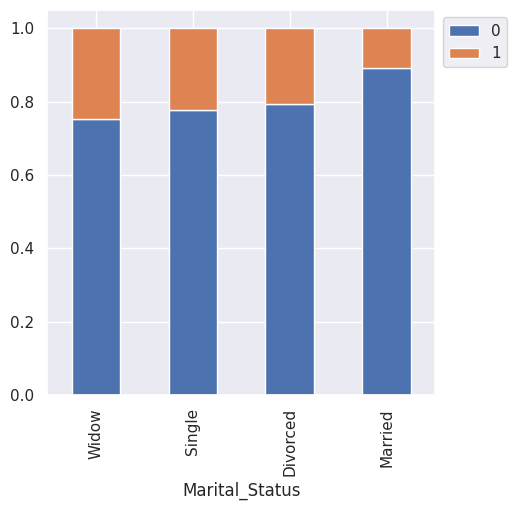
Bivariate Analysis



* A negative correlation between age and year of birth is observed. To avoid collinearity, we may consider dropping one of the variables.
* Since registration month and quarter are derived from the same column, it is expected to see a high correlation between them. Hence, we may consider dropping one of the columns to avoid multicollinearity.
* Due to almost perfect correlation between month and quarter, we may consider removing one of these variables to avoid redundancy.
* Total amount spent shows correlation with other relevant variables, thus dropping this column might be a viable option.
* As expected, customers with higher income tend to have a higher number of purchases. Therefore, we may consider exploring the relationship between these two variables to gain more insights.



* Upon conducting our analysis, we observed that there is a positive correlation between the level of education and the amount spent by the customers. Specifically, we found that as the level of education increases, so does the amount spent.
* Additionally, our observations revealed that customers with graduate-level education tend to spend slightly more than customers with master-level education. This difference, while not significant, could indicate potential trends or patterns that may warrant further investigation.



* Through our observation, we noticed that the number of married customers in the dataset is significantly higher than those who are single or divorced. However, we also discovered that divorced or widowed customers are more inclined to take the offer presented in the dataset.
* Moreover, our analysis revealed that single customers are more likely to take the offer compared to married customers.

### **Data Preprocessing and Feature Engineering**

We observed that feature engineering plays a crucial role in machine learning projects as it enhances the model's ability to comprehend the data. To improve the performance of our model, we employed various feature engineering techniques, including methods for handling categorical and numerical attributes.

We have imputed the null values in the income column using the mean value of income using the below code

| from pyspark.sql.functions import mean, when, col # Calculate mean of Income column mean\_income = spark\_df.select(mean("Income")).collect()[0][0] # Replace missing values in Income column with mean value spark\_df = spark\_df.withColumn("Income", when(col("Income").isNull(), mean\_income).otherwise(col("Income"))) |
| --- |

We also need to convert the categorical columns into string type so that we can encode them using a string indexer and one hot encoder.

| from pyspark.sql.types import StringType spark\_df = spark\_df.withColumn("Kidhome",spark\_df["Kidhome"].cast(StringType())) spark\_df = spark\_df.withColumn("Teenhome",spark\_df["Teenhome"].cast(StringType())) spark\_df = spark\_df.withColumn("Response",spark\_df["Response"].cast(StringType())) spark\_df = spark\_df.withColumn("Complain",spark\_df["Complain"].cast(StringType())) |
| --- |

For Categorical attributes - We would use stringindexer to handle the categorical features and transform them.We observed that in Pyspark, a common technique for data preprocessing is StringIndexer. It enables the conversion of categorical string features into numerical features by assigning a unique numerical index to each category present in the feature column, based on the frequency of occurrence. This transformation is beneficial for machine learning models that require numerical inputs, as it allows the use of categorical data in these models.

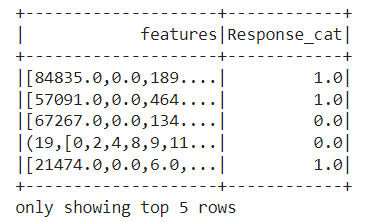
| from pyspark.ml.feature import StringIndexer str\_indexer=StringIndexer(inputCol='Education',outputCol='Education\_cat') str\_indexer.setHandleInvalid("error") cat\_indexed=str\_indexer.fit(spark\_df).transform(spark\_df) cat\_indexed.head(1) |
| --- |

And this is similarly done for all the categorical features like kidhome, teenhome, response, complain etc.

For Numerical attributes - We noticed the usage of VectorAssembler to handle the numerical features. VectorAssembler is a built-in functionality in Pyspark that enables the combination of several input columns into a single output column of vectors. By specifying a list of input column names, it generates a vector column where each element of the vector represents the value of a particular input column. This approach is valuable in data preparation for machine learning models that require input data in vector format.

| assembler=VectorAssembler(inputCols=[  'Income',  'Recency',  'MntWines',  'MntFruits',  'MntMeatProducts',  'MntFishProducts',  'MntSweetProducts',  'MntGoldProds',  'NumDealsPurchases',  'NumWebPurchases',  'NumCatalogPurchases',  'NumStorePurchases',  'NumWebVisitsMonth',  'Age',  'Education\_cat',  'Marital\_Status\_cat',  'Kidhome\_cat',  'Teenhome\_cat',  'Complain\_cat'],outputCol="features") output=assembler.transform(cat\_indexed) |
| --- |

And the output of the vectorassembler is a vector as shown below:



Further we have split data separately for training and testing. We have used a 70:30 percent split between the training and testing data.

### **Model Training and Evaluation**

We have trained the data using several models as shown below

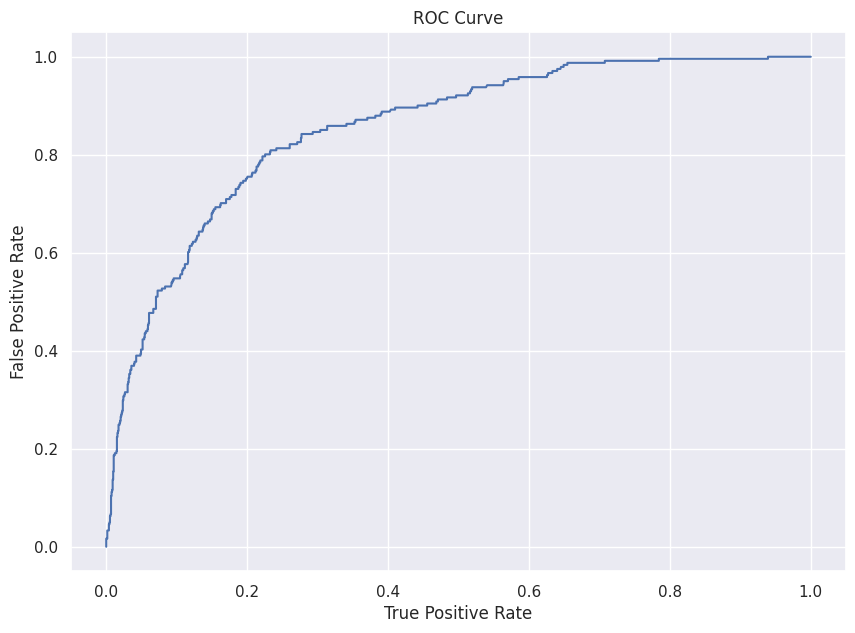
#### Logistic Regression:

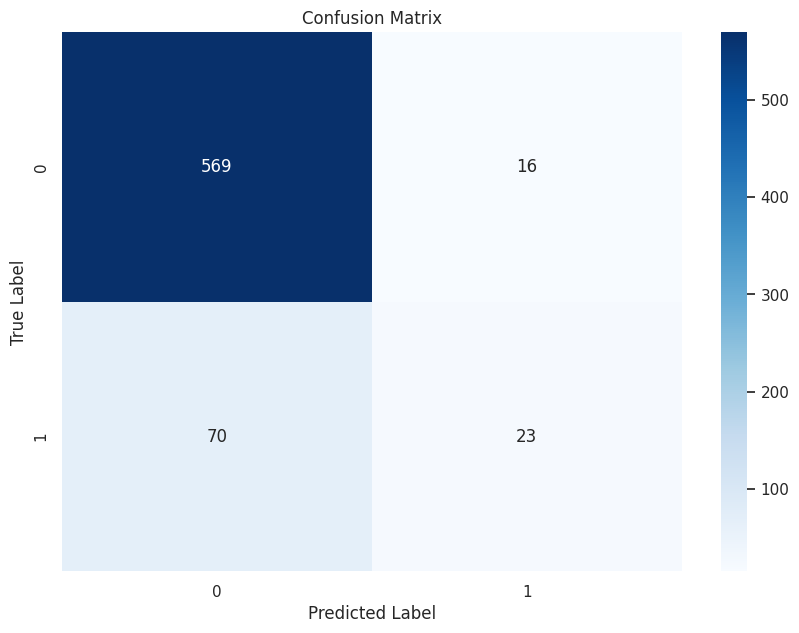
Logistic regression is a statistical approach utilized for examining and modeling the correlation between one or more independent variables and a dependent variable. It is a type of regression analysis that is commonly applied to solve binary classification issues, such as forecasting if a customer will churn or not. The logistic regression approach is based on the assumption that the connection between the dependent variable and independent variables is linear. The output from this method is converted into probabilities using a logistic function. These probabilities can be utilized to make decisions based on a specific threshold or to predict future outcomes. Logistic regression is a popular and widely used method in the field of machine learning and data science due to its simplicity, effectiveness, and interpretability in a wide range of applications.

| from pyspark.ml.classification import LogisticRegression *# Create Logistic Regression Classifier model* lr = LogisticRegression(featuresCol = 'features', labelCol = "Response\_cat")  *# Fit model on train data* lr\_model = lr.fit(train\_data) |
| --- |

Logistic regression is the first model used and evaluated it on the test data and found,

| AUC: 0.8039 Precision: 0.8492 Recall: 0.8732 F1-score: 0.8500 Accuracy: 0.8732 |
| --- |

We can also observe the ROC curve as shown below  


The confusion matrix can also be observed below  


We can observe that about 569 are true positives and most of them are correctly classified as the data is somewhat imbalanced so there are fewer True negatives.

#### Random Forest Classifer:

The second model built is a random forest classifier and it is built using the same features and target class and the model is evaluated on the test data using the below code

| *# Evaluate model using MulticlassClassificationEvaluator* evaluator = MulticlassClassificationEvaluator(labelCol="Response\_cat") precision = evaluator.evaluate(predictions, {evaluator.metricName: "weightedPrecision"}) recall = evaluator.evaluate(predictions, {evaluator.metricName: "weightedRecall"}) f1\_score = evaluator.evaluate(predictions, {evaluator.metricName: "f1"}) accuracy = evaluator.evaluate(predictions, {evaluator.metricName: "accuracy"}) print("Precision: {:.4f}".format(precision)) print("Recall: {:.4f}".format(recall)) print("F1-score: {:.4f}".format(f1\_score)) print("Accuracy: {:.4f}".format(accuracy)) |
| --- |

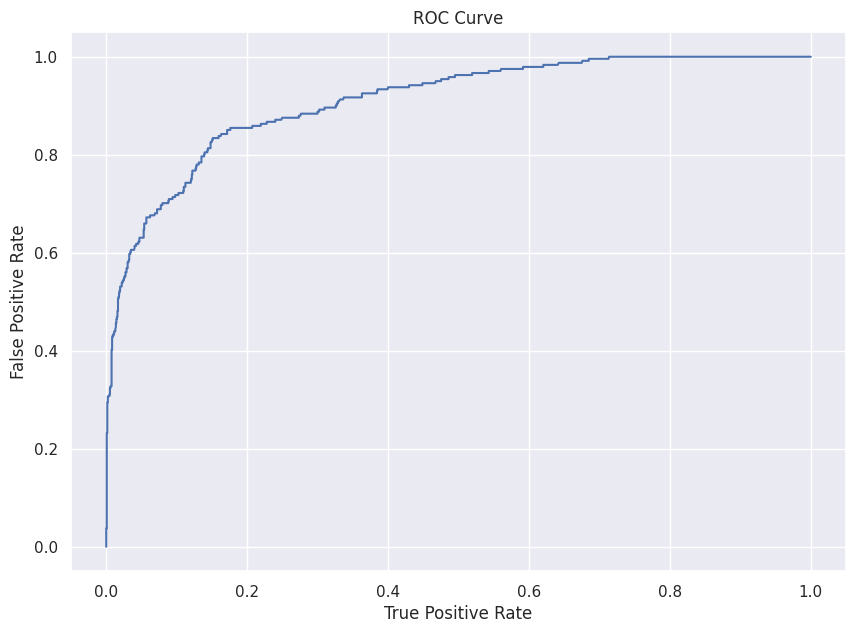
We observed the scores as shown below.

Precision: 0.8445

Recall: 0.8702

F1-score: 0.8293

Accuracy: 0.8702

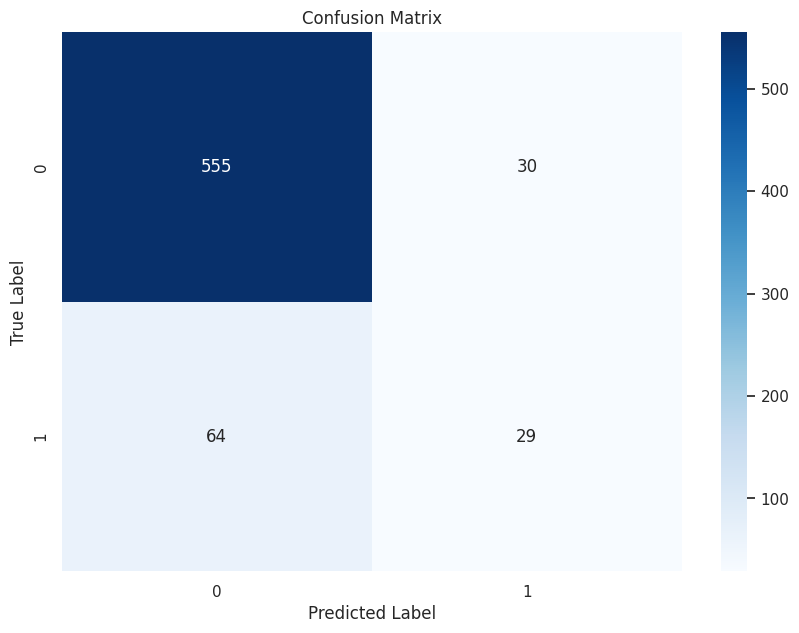
Also the ROC curve can be observed below  


Similarly the other tree based models are built like decision trees but the performance is not that great. FInally we have built a gradient based modeling using the belo code

| *# Create Gradient Boosting Classifier model* gbt = GBTClassifier(featuresCol = 'features', labelCol = "Response\_cat") *# Fit cross-validator on train data* gbt\_model = gbt.fit(train\_data) |
| --- |

This model seems to perform better than other models with scores as shown below

| AUC: 0.7740 Precision: 0.8410 Recall: 0.8614 F1-score: 0.8478 Accuracy: 0.8614 |
| --- |

The confusion matrix can be seen below  


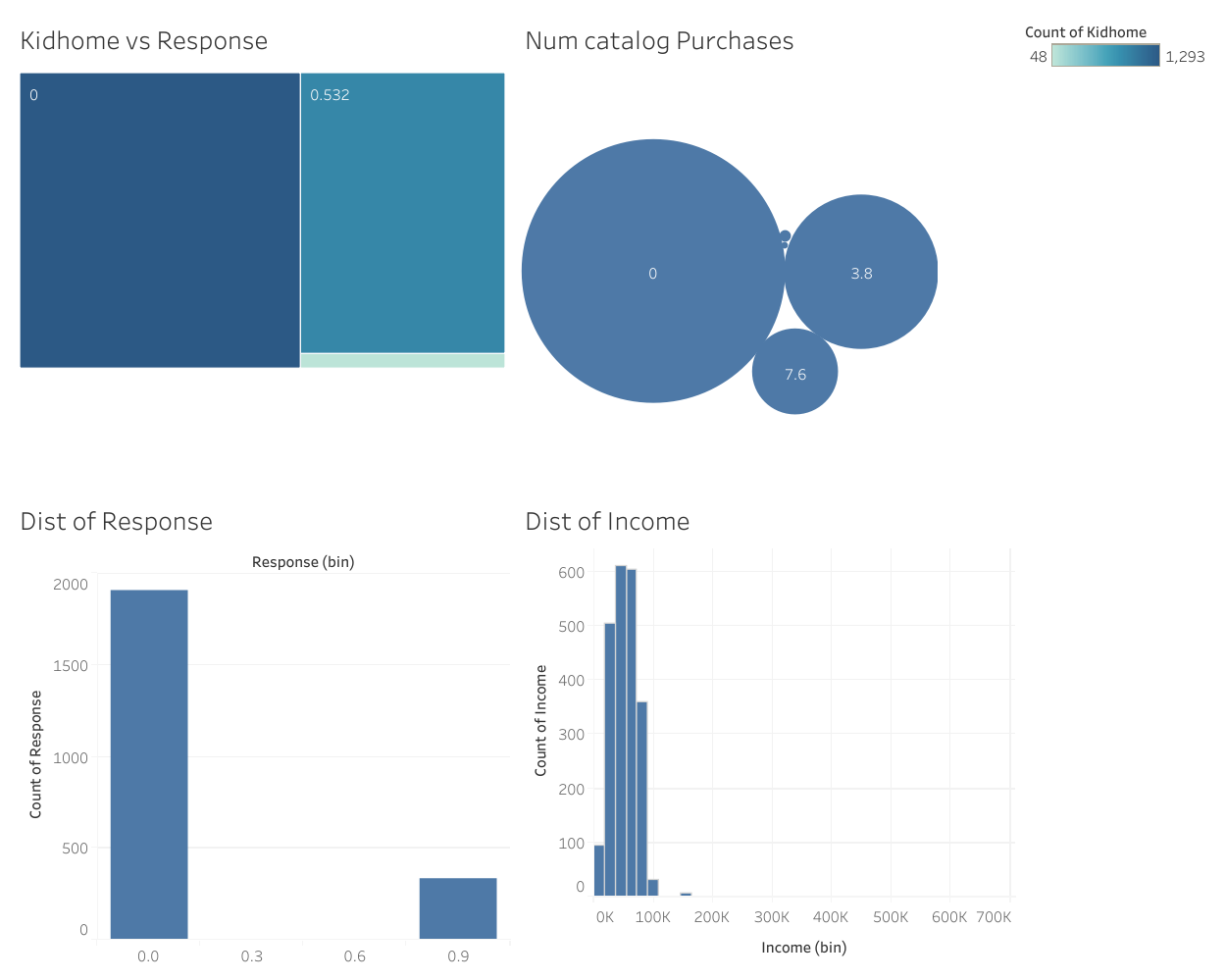
Here there is a good balance between True positives and True negatives. We have also obtained the best features using the GBT model with their scores as:

| Feature: 0, Score: 0.07800 Feature: 1, Score: 0.12451 Feature: 2, Score: 0.08239 Feature: 3, Score: 0.04770 Feature: 4, Score: 0.07316 Feature: 5, Score: 0.04616 Feature: 6, Score: 0.04718 Feature: 7, Score: 0.05563 Feature: 8, Score: 0.03905 Feature: 9, Score: 0.02851 Feature: 10, Score: 0.06371 Feature: 11, Score: 0.06170 Feature: 12, Score: 0.05392 Feature: 13, Score: 0.07160 Feature: 14, Score: 0.01734 Feature: 15, Score: 0.06697 Feature: 16, Score: 0.00241 Feature: 17, Score: 0.04005 Feature: 18, Score: 0.00000 |
| --- |

### **Conclusion**

* Based on our observations, there has been a decrease in the total amount spent over the years. This could be an indication of a decline in product quality or a lack of effective marketing strategies by the company.
  + To address this issue, the company should focus on continuously improving its marketing strategies.
* Our analysis revealed that the majority of customers (~99%) did not have any complaints in the past two years. However, this could be because there are limited options for customers to provide feedback.
* It would be beneficial for the company to create easy mechanisms for customers to share their feedback and identify any major concerns.
* The company should target customers who purchase premium products such as gold products or high-quality wines. These customers have a higher likelihood of making purchases and spending more.
* To further capitalize on this segment, the company should consider launching premium offers tailored to these customers.
* Additionally, these offers can also be extended to customers with higher income levels.
* The number of website visits is a critical factor that the company should consider. Customizing the website can help direct more traffic to it.
* To improve the website's effectiveness, the company can enhance the interface and provide easy check-in, check-out, and delivery options.
* Based on our findings, frequent buyers should be given more attention by the company and offered added benefits.
* By targeting these customers, the company can increase customer retention and loyalty.

### **Some Comparison Plots using Tableau**



From the above plot, we can observe the distribution of response and it seems to be imbalanced. Even the distribution of income seems to be normal and the relation between count of kids present in home and response can be analyzed.

## **Tech Stack used:**

Tableau, Jupyter notebook, Python, Pyspark, Pandas, Machine Learning frameworks.