

Springboard DSC

Capstone Project 2:

*(Customer Segmentation &
Predicting Customer Response
to Marketing Offer)*

By: Rachid Rezzik

April 2019

The Client and Their Desired Solution:

An anonymous company (“the client”) has tracked the results of six marketing campaigns for the exact same offer. The offer is a discount on a bundle of products including wine, meat, fish, sweets, and a small piece of gold jewelry. With response data from over two thousand customers who received the offer six times, the company would like to predict customer acceptance rates to view which customers will accept the offer more often, less often, or not at all. A segmentation of its customers has been requested as well.

The client wants to utilize their historical campaign data to run future campaigns that are more targeted, reducing unnecessary customer contact costs (while also limiting any associated decrease in revenue). The requested segmentation will allow the client to gain a better understanding of the key features for each customer group to successfully market the bundle offer and any other current/future offers to their customers.

Utilizing each customer’s features and offer acceptance rate, I will train a model to predict the rate at which any given customer would accept the offer. If a customer is predicted to contain an acceptance rate above zero, I will then assign them to an acceptance rate group (Low/Medium/ High). A second model will be trained to predict which acceptance rate group the customer belongs to.

I will also segment customers based on their features utilizing unsupervised machine learning techniques. With the segmentation and historical acceptance rate for each customer, I can gain insight into what features increase a customer’s value to the client and how the offer performed within respective customer value groups.

The features that contained the most favorable/unfavorable influence on the offer acceptance rate will be extracted from the historical data. From there, the marketing team can formulate new approaches for the offer, based on certain customer features. The marketing team could also use the information extracted from the customer segmentation, the historical performance of the bundle offer, or a combination of both to formulate new offers that would more effectively target customer groups.

The code I developed for this project is available within my GitHub repository ¹.

Note: At the start of my “Data_Analysis” notebook in the “code” folder, the reader can find the assumptions/alterations I made to the original project proposed on Kaggle ².

¹ https://github.com/RachidRezzik/Market_Camp

² <https://www.kaggle.com/rodsaldanha/arketing-campaign>

Data Wrangling / Feature Engineering:

Acquiring the marketing dataset was very simple as I just downloaded it from Kaggle. The dataset contained records for 2,240 customers and their individual responses to the six marketing campaigns. For each customer, unique features like their birth year, education, relationship status, income, number of kids, number of teens, enrollment date, number of purchases by channel, and the amount of money they've spent on certain products were provided.

Upon my first inspection of the dataset, it was apparent that income data was not available for several customers. In order to impute the missing data, I utilized the average income for the customer's respective education and age group. Education was already provided as a categorical feature, but I needed to do some wrangling to engineer the age groups of the customers.

Using each customer's birth year, I calculated their age and provided their age group (Ex: "Thirties"). With each customer's education and age group, I imputed the missing values based on the average income for that customer's education and age group. For the number of kids and teens each customer has, I simply summed these features to form the "Children" feature.

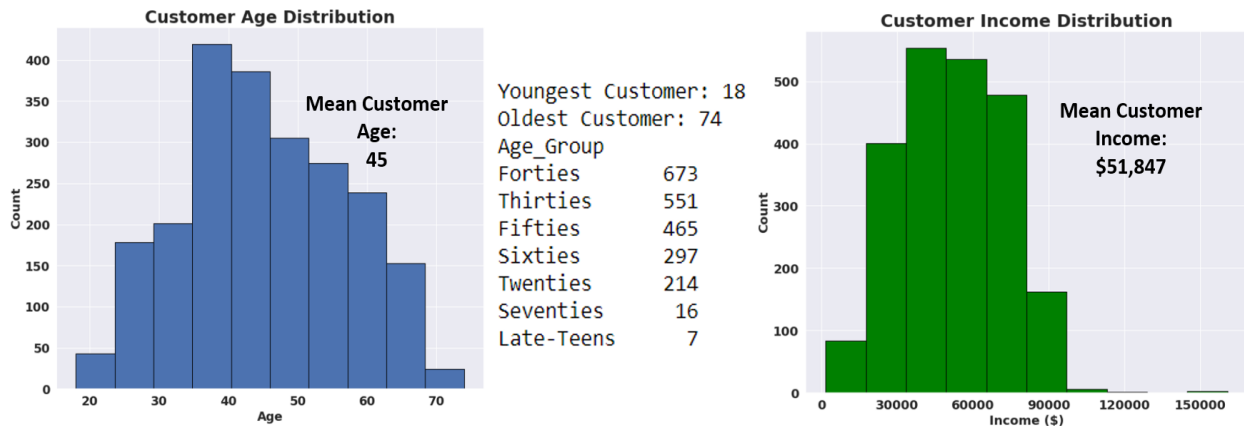
I then believed it would be beneficial to engineer some features that effectively describe a customer's spending habits. To do so, I calculated each customer's average purchase frequency ("Avg_Purch_Freq") and their average amount spent per purchase ("Avg_Spend (\$)").

To provide some target variables to later predict, I engineered the "Accept_One" feature (indicating if the customer accepted at least one of the six offer attempts) and the "Accept_Rate" feature (indicates the number of offers accepted out of six → Ex: "(3/6)") for building classifiers. The "Cust_Accept (%)" feature ($\text{Accept_Rate} = (3/6) \rightarrow 50\%$) was also engineered to aid in the analysis provided in the "Data Storytelling / Inferential Statistics" and "Machine Learning – Customer Segmentation" sections.

Data Storytelling / Inferential Statistics

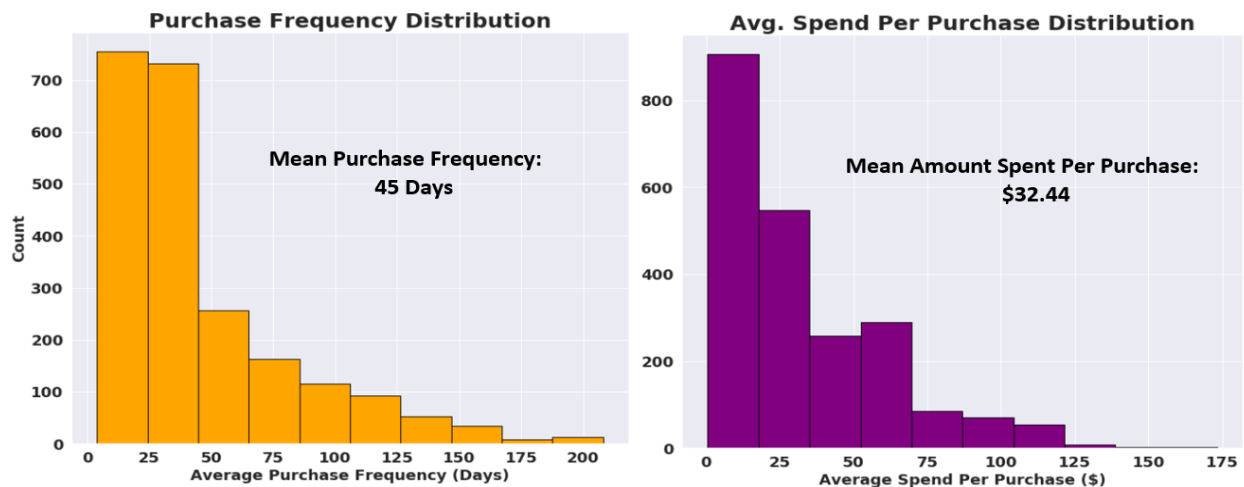
After wrangling the data into a suitable format, I began to look for any curious feature values / outliers that I could drop. In total, I found 17 customers out of 2240 (0.76%) that I deemed beneficial to drop for the inferential statistics provided in this section.

To get to know the customers a bit better, I first looked at the customer age and income distributions.



The client's customers averaged a yearly income of roughly \$52,000. The largest age group contained people in their forties and the mean customer age was 45. The largest "Relationship_Status" category belonged to those who were married, and the largest "Education" category belonged to those who held a bachelor's degree.

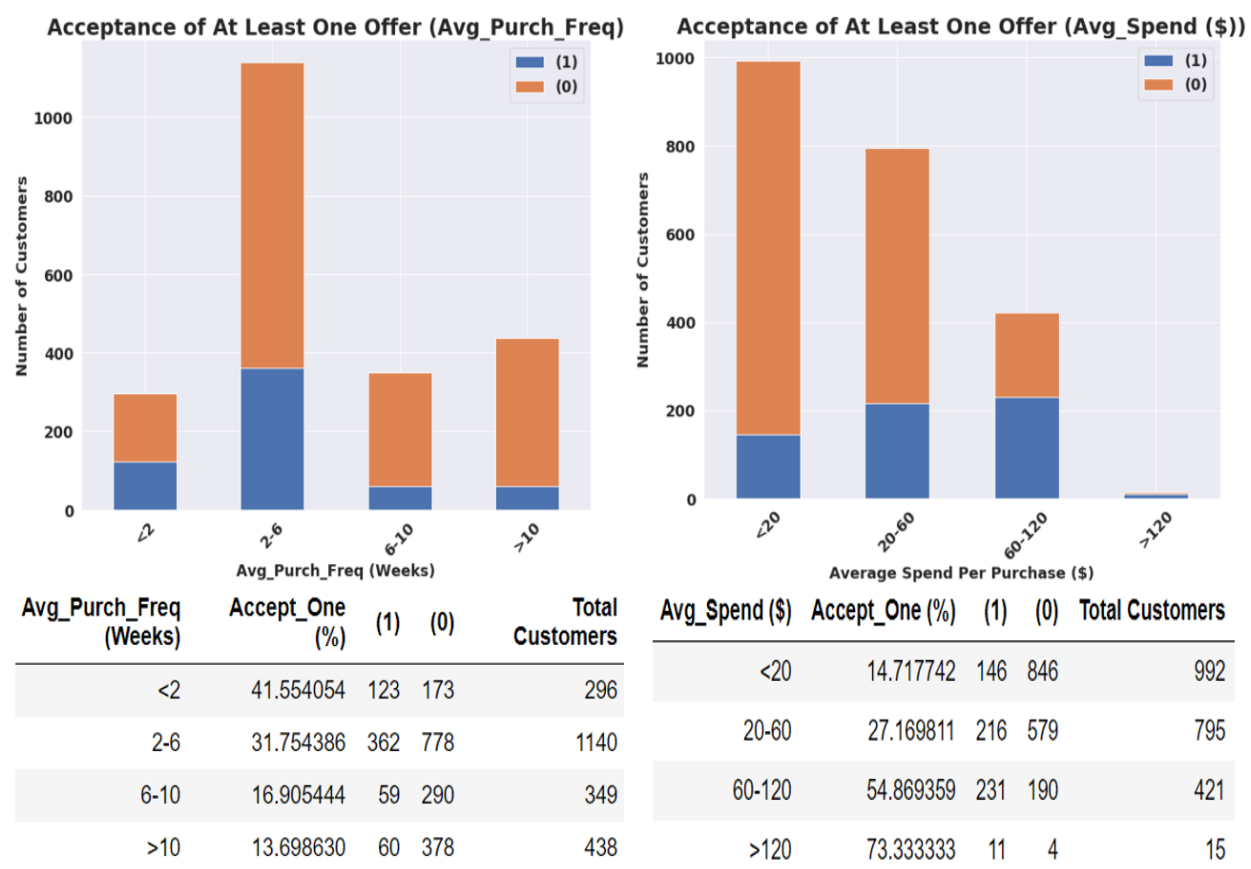
Let's take a look at the spending habits of the customers through their purchase frequency and average spend per purchase distributions.



To begin analyzing the success of the six marketing campaigns, let's view the customer acceptance rate counts provided below.

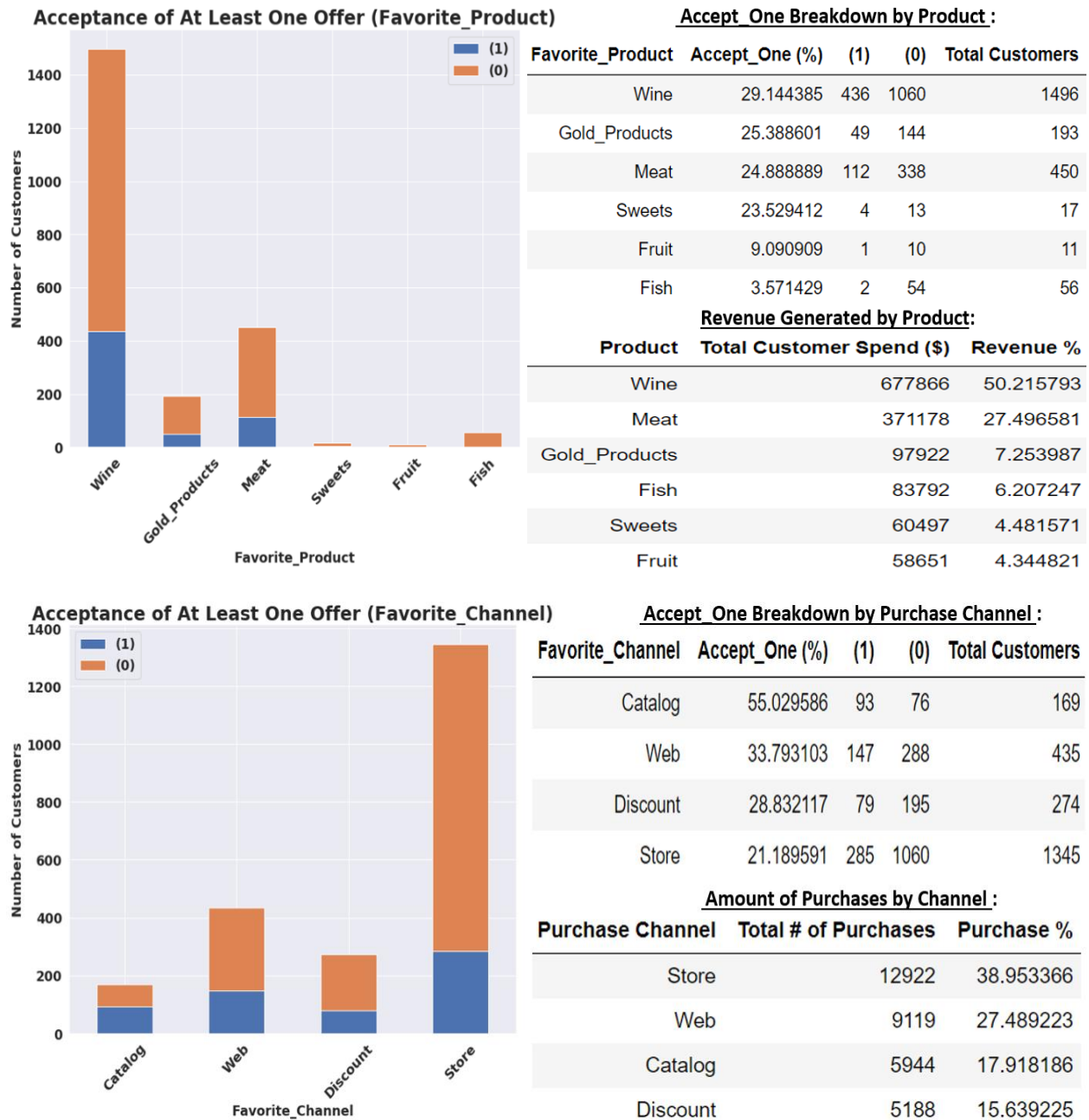
Accept_Rate	Value	Counts
(0/6)		1619
(1/6)		365
(2/6)		142
(3/6)		51
(4/6)		36
(5/6)		10

Roughly 73% of customers did not accept the offer at least once during the six marketing campaigns. In total, the offer was accepted 996 times out of 13,338 attempts for an overall acceptance percentage of 7.47%. After previously viewing the distributions for “Purchase_Frequency” and “Avg_Spend (\$)”, let’s see if we can spot any interesting trends between these features and the acceptance of at least one of the six offer attempts (“Accept_One” = 1).



The offer was popular with those who purchased products more frequently and spent more per purchase, making it successful in appealing to frequent customers who generate the largest amount of revenue for the client.

We have seen how spend and purchase frequency affects acceptance, but how about each customer’s favorite product and purchasing channel?

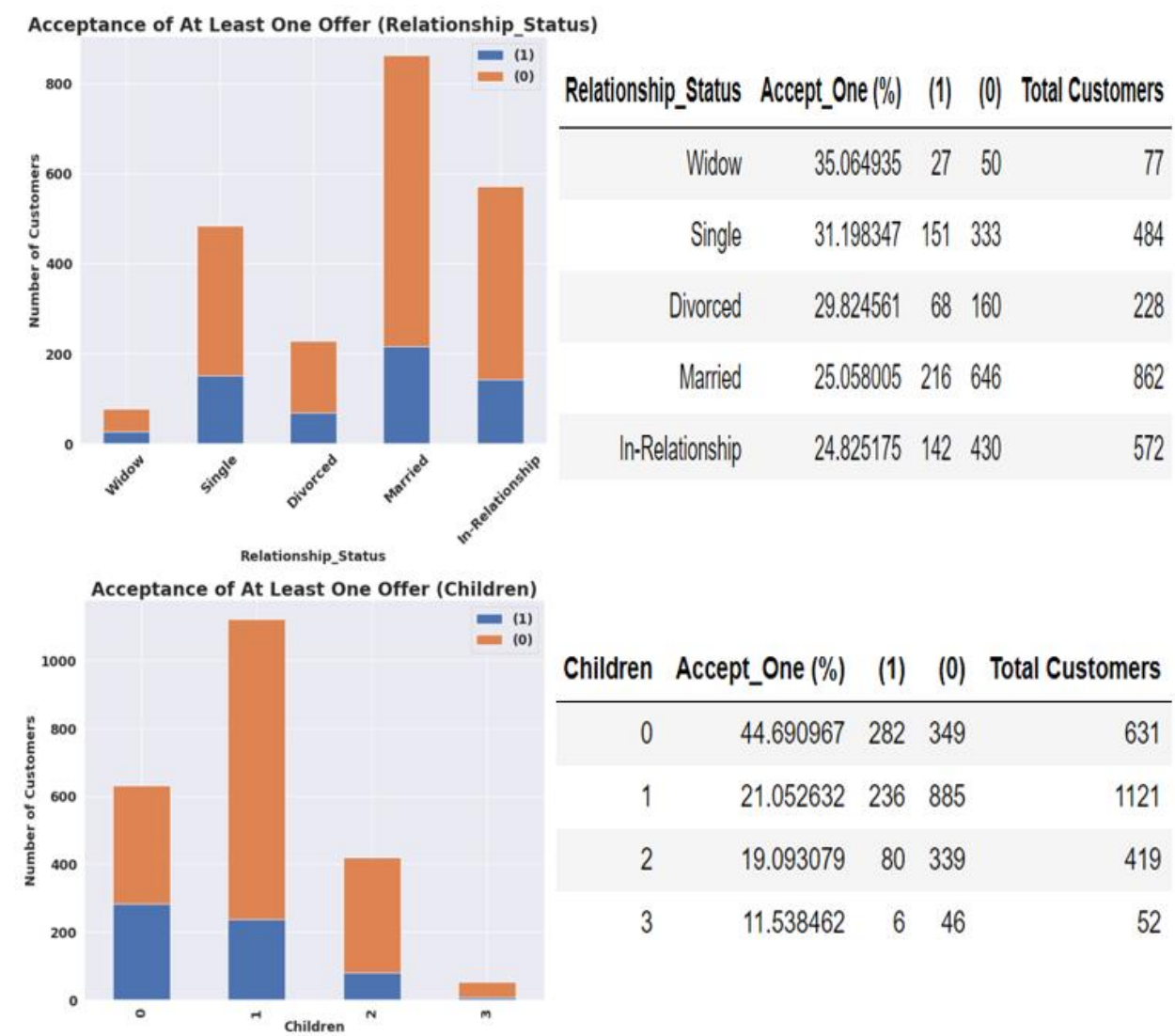


The offer was the most popular with those who enjoy wine products. This is encouraging as roughly 50% of the revenue generated from the client's customers was from wine products. Meat generated the second largest revenue for the client, so the acceptance among those who favor meat products was also positive. As for fish products (6% of revenue generated), the offer appeared to be very unpopular.

Roughly 39% of customer purchases were made in the store, representing the majority of customer's favorite channels. Web purchases (27%) and catalog purchases (18%) came in second and third, respectively. It appears that the offer was quite successful among those who

prefer purchasing products through the client’s catalog as 55% of these customers accepted the offer.

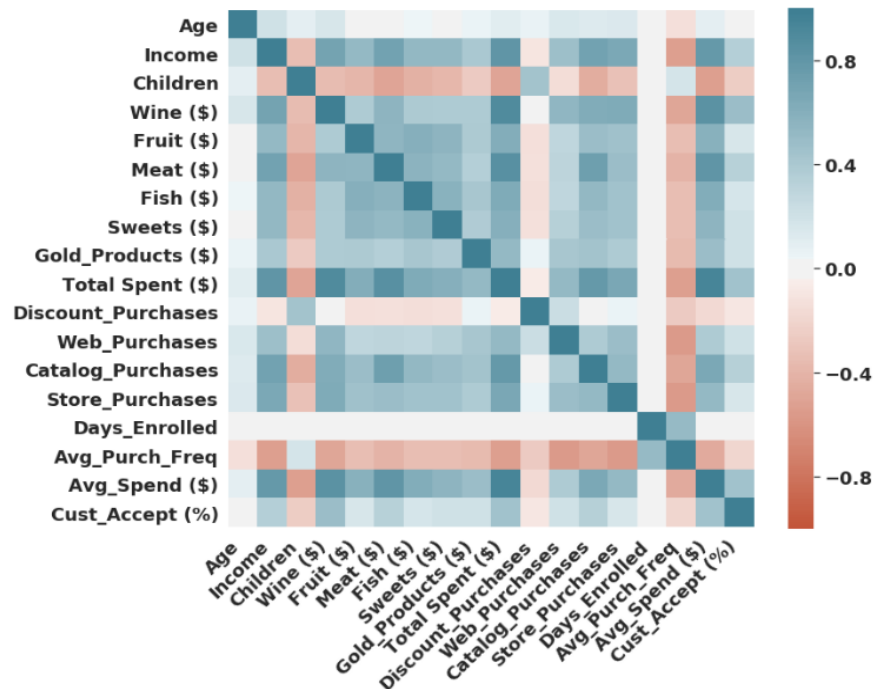
Another useful exercise would be to analyze customer response to the offer based on their relationship status and number of children.



Those who were not in a relationship with another individual (Single, Widow, Divorced) contained better response to the offer compared to those who were (Married, In-Relationship). Out of those who were not in a relationship, the offer found its best success among those who were labeled as “Single”.

Roughly 45% of customers who had no children accepted the offer at least once. The offer decreased in popularity among customers with more children.

We can take a look at the correlation matrix for the numerical features included in this study. From the figure, we can see what features positively or negatively influenced acceptance percentage and the correlation between the features.



A customer's income, total money spent on all of the client's products, money spent on wine and/or meat products, number of catalog purchases, and average amount of money spent per purchase contained the best positive correlation with offer acceptance percentage. These features all contained positive correlation with each other, prompting me to later drop "Total Spent (\$)" to improve the performance/evaluation of my classifiers.

The number of children per customer mostly decreased correlation across the board. Only the number of discount purchases, age, and average purchase frequency contained positive correlation with the number of children. Average purchase frequency also contained negative correlation with many of the features and offer acceptance percentage.

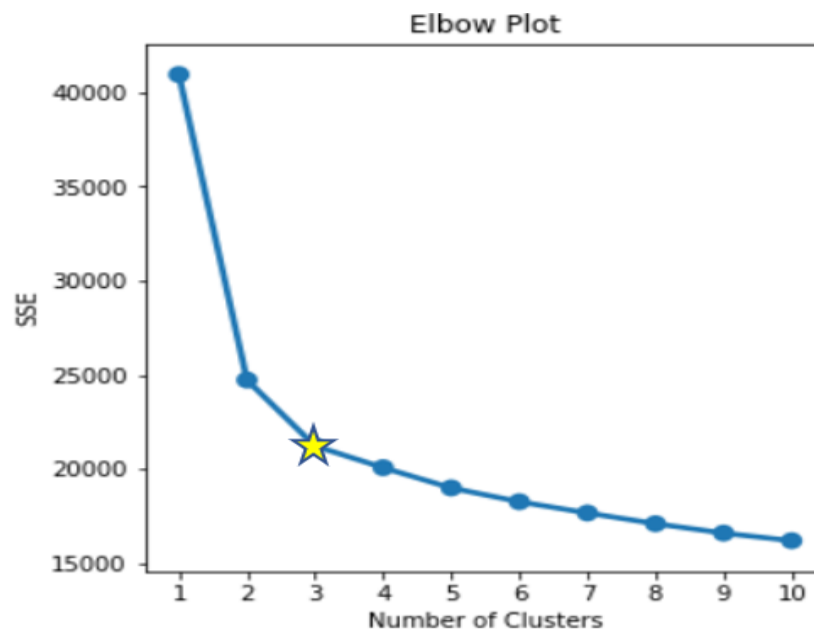
Machine Learning (*Unsupervised – Customer Segmentation*):

To begin clustering the client's customers, I first needed to unskew the numerical features with a $\log(1+x)$ transform. This particular transform was utilized due to several of the customer features being expressed as count data containing zeros. I then proceeded to use sklearn's StandardScaler() ³ to standardize features by removing the mean and scaling to unit variance. For the categorical data, I utilized panda's get_dummies() ⁴ method to numerically encode the values.

³ <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>

⁴ https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.get_dummies.html

I decided I would utilize K-means ⁵ as my clustering algorithm. To find the ideal number of clusters for my customer segmentation, I ran the algorithm for ten different k-values (number of clusters) and plotted each resulting Sum of Squared Errors ("SSE") values.



The goal here is to choose the smallest value of k in which the SSE is still relatively low. In this case, three was found to be the optimal number of clusters.

To begin the segmentation, I utilized K-means to cluster customers based on their purchase frequency ("FrequencyCluster"). The same process was used to cluster customers based on their average spend per purchase ("SpendCluster") and the total amount of revenue they generated for the client ("RevenueCluster"). For each of the three cluster types, its individual clusters were ranked by value to the client.


FrequencyCluster	Avg_Purch_Freq	SpendCluster	Avg_Spend (\$)	RevenueCluster	Total Spent (\$)
0	13.384433	0	72.666833	0	1207.000000
1	32.967652	1	26.659581	1	300.275605
2	96.111263	2	7.726328	2	50.792350

With each customer containing a value ranking (0,1,2) for each cluster type, I calculated each customer's "Overall_Score". "Accept_One" and "Cust_Accept (%)" figures were also provided to indicate how well the offer did among those who are deemed to be more valuable to the client.

⁵ <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>

	Avg_Purch_Freq	Avg_Spend (\$)	Total Spent (\$)	Accept_One	Cust_Accept (%)
Overall_Score					
0	13.342309	71.207550	1491.935897	0.547009	19.800570
1	24.640631	59.876261	1227.216981	0.428302	12.138365
2	23.839952	31.456801	641.983444	0.245033	6.181015
3	34.712721	21.665775	326.111486	0.206081	4.391892
4	52.901852	15.210619	157.304054	0.148649	2.927928
5	44.370824	8.660294	63.248963	0.145228	3.181189
6	101.568698	7.325148	45.415254	0.120763	2.471751

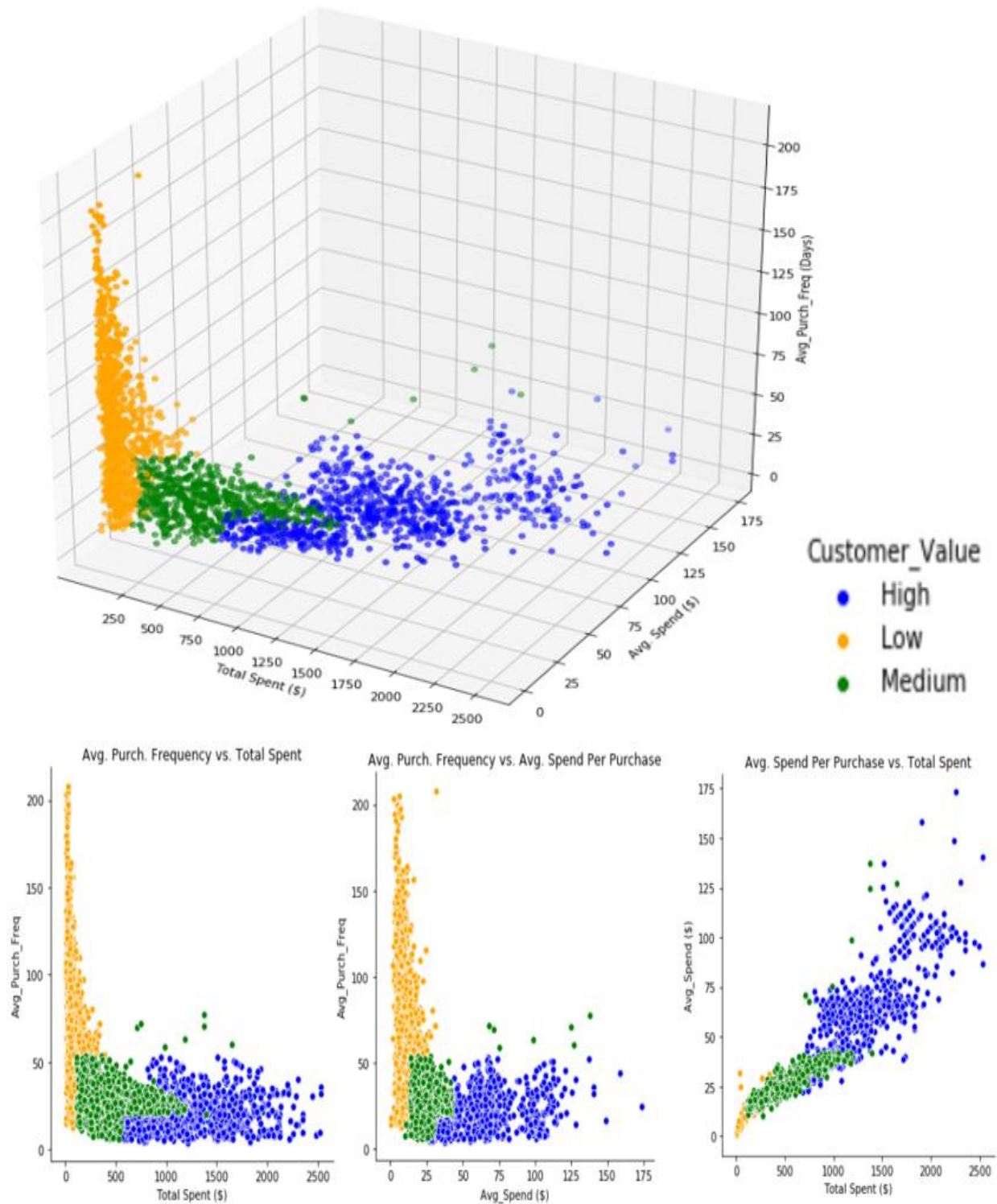
The offer performed better among customers containing higher value to the client. Next, I needed to provide the client with “Customer_Value” labels for each of their customers. Below is a summary of how I assigned respective labels to each customer, in addition to the resulting average metrics for each value group.

Overall_Score: 0, 1 → Customer_Value: High		<u>Value</u>	<u># Customers</u>
Overall_Score: 2, 3 → Customer_Value: Medium		High	764
Overall_Score: 4, 5, 6 → Customer_Value: Low		Medium	598
		Low	861

	Avg_Purch_Freq	Avg_Spend (\$)	Total Spent (\$)	Accept_One	Cust_Accept (%)
Customer_Value					
High	21.180150	63.346839	1308.295812	0.464660	14.485166
Medium	29.221791	26.610407	485.632107	0.225753	5.295429
Low	77.193110	9.054323	69.639954	0.132404	2.748742

On average, we see that the high value group contained less time between purchases, spent more money per visit, and generated 70% of total revenue for the client. The offer was accepted at least once out of the six attempts for 46% of high-value customers with an overall average acceptance percentage of 14%. On the other hand, we see that the low value customers contained more time between purchases, spent less money per visit, and generated 4% of total revenue for the client. The offer was accepted at least once out of the six attempts for 13% of low-value customers with an overall average acceptance percentage of 3%.

Below is a visual representation of the customer value segmentation, utilizing the metrics previously discussed. We can see that the lower value group is higher and to the bottom-left, while the high value group is lower and to the upper-right.



Moving forward, I looked to identify any other trends in customer value caused by certain features.

Basic Customer Characteristics

	Age	Income	Children	Days_Enrolled	Accept_One	Cust_Accept (%)
Customer_Value						
High	46.725131	72333.340767	0.494764	436.540576	0.464660	14.485166
Medium	47.237458	53162.978290	1.088629	497.301003	0.225753	5.295429
Low	42.140534	32754.475097	1.261324	497.527294	0.132404	2.748742

As identified previously in the “Data Storytelling / Inferential Statistics” section, the customers offering more value to the client are those who generate a higher income and have fewer children.

Percentage of Revenue Generated – By Product

	Wine (%)	Fruit (%)	Meat (%)	Fish (%)	Sweets (%)	Gold_Products (%)	Accept_One	Cust_Accept (%)
Customer_Value								
High	50.085290	4.310314	28.610462	6.194738	4.547173	6.252023	0.464660	14.485166
Medium	53.516568	4.293118	21.213320	5.929793	4.228313	10.818889	0.225753	5.295429
Low	36.811064	5.931377	24.327950	8.845264	6.181877	17.902467	0.132404	2.748742

The low value customer group spent the highest percentage of their money on gold products, fish, sweets, and fruit. All of these products contained a weak positive correlation with offer acceptance percentage.

Percent of Purchases – By Channel

	Discount (%)	Web (%)	Catalog (%)	Store (%)	Accept_One	Cust_Accept (%)
Customer_Value						
High	9.324511	26.230318	25.472909	38.972261	0.464660	14.485166
Medium	18.352325	30.598122	13.216972	37.832580	0.225753	5.295429
Low	25.736462	24.526019	4.952051	44.785468	0.132404	2.748742

Higher value customers preferred purchasing through the catalog more often than those with decreasing value, backing the trend seen in the “Data Storytelling / Inferential Statistics” section.

Machine Learning (*Supervised – Predicting Acceptance of At Least One Offer*):

The client first and foremost wants to predict if a customer will accept at least one offer in six attempts (Accept_One = 1, “class 1”) or not (Accept_One = 0, “class 0”). When training a classifier to solve this problem, class imbalance presented an issue as the vast majority of customers (73%) did not accept at least one offer. To offer the client more realistic performance results, I applied stratification to maintain this imbalance in the training and test sets.

Acknowledging the imbalance in the training set, I proceeded to train and evaluate the performance of multiple models utilizing the imbalanced data, undersampling the majority class, and oversampling the minority class. For each approach, I trained a RandomForestClassifier⁶ and then looked to improve on its test scores by utilizing XGBoost’s (“XGB”) ⁷ parallel tree boosting.

In the end, I was able to improve the performance for each case by utilizing the XGB algorithm. For that reason, I will only show the performance metrics associated with the XGB algorithm. RFC metrics are available in the “Machine_Learning” notebook for anyone interested.

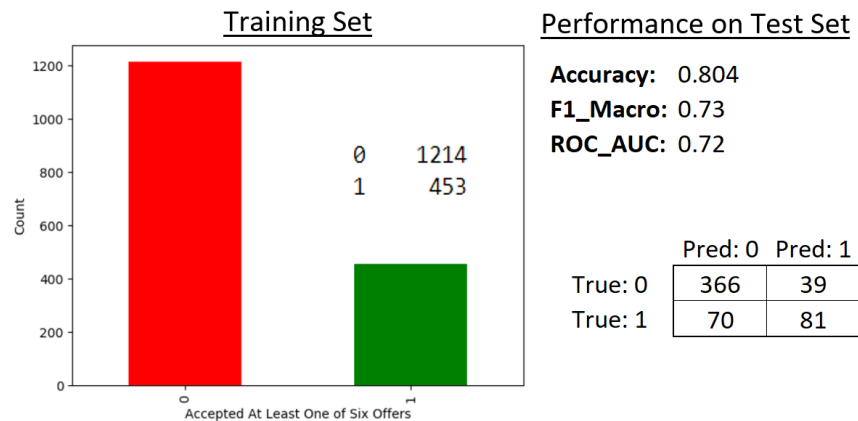
For each XGB model I trained, the hyperparameters I tuned include the shrinkage of the weights associated to features after each round (‘learning_rate’), the number of trees included (‘n_estimators’), the subsample ratio of the training instances (‘subsample’), the subsample ratio of columns when constructing each tree (‘colsample_bytree’), the minimum loss reduction required to make a further partition on a leaf node of the tree (‘gamma’), the maximum number of nodes allowed from the root to the farthest leaf on the tree (‘max_depth’), and the minimum sum of instance weight needed in a child (‘min_child_weight’). A summary of the steps I took to tune each XGBoost model is provided in the “Machine_Learning” notebook.

Model Performance Summaries:

(Imbalanced – Base Model)

⁶ <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

⁷ <https://xgboost.readthedocs.io/en/latest/>

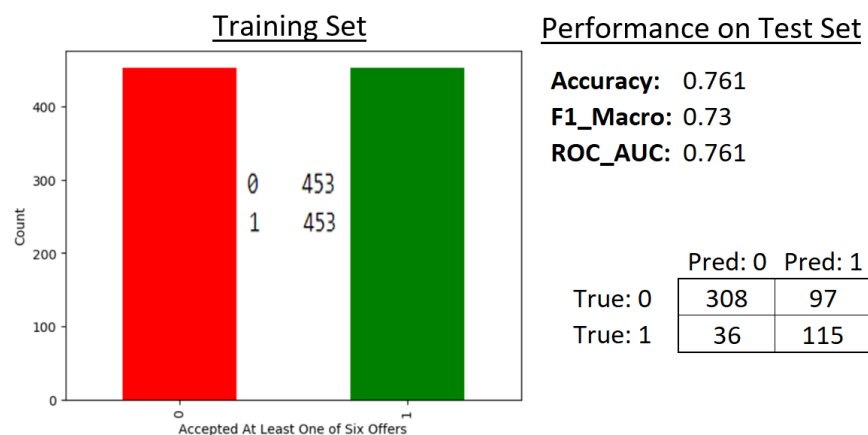


In comparing the performance of the “imbalanced” model to the other two, I mainly focused on the “F1_Macro” and “ROC_AUC” metrics. The F1_Macro score indicates each model’s harmonic mean of recall and precision (“f1 score”) in both classes, while the ROC_AUC score essentially indicates the model’s capability of predicting 0s as 0s and 1s as 1s.

Accuracy can be deceiving and needs to be monitored in conjunction with the above-mentioned metrics. The imbalanced training data leads the model to see many more class 0 customers than class 1 customers, resulting in better performance in accurately predicting class 0 customers than class 1. The client needs to, as often as possible, correctly identify those that will accept at least one of six offer attempts. Poorly identifying customers that will accept at least one offer loses the client revenue and customer exposure to the products included in their bundle offer.

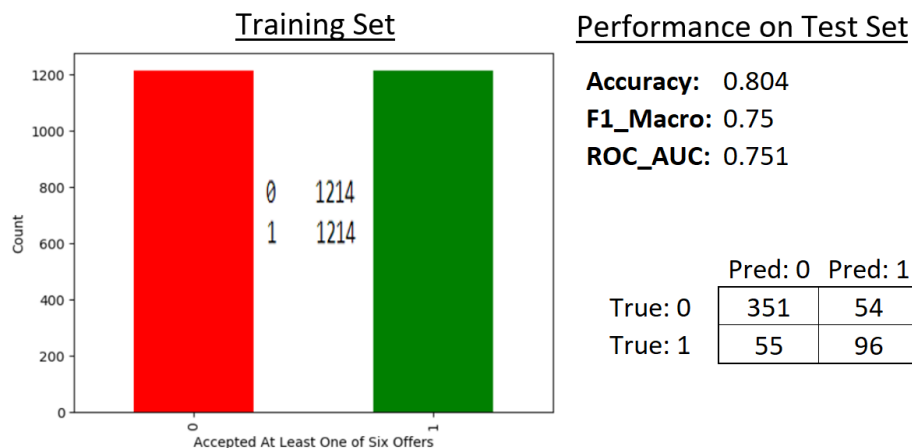
To best meet the client’s needs, I worked to increase class 1’s f1 score, limit any decrease in class 0’s f1 score, improve the ROC-AUC score, and maintain a similar accuracy on the test set. I first experimented with undersampling the majority class to equal the number of customers in the minority class.

(Undersampling)



The F1_Macro score was maintained and the ROC_AUC score increased with the model being trained on an even amount of both classes. While class 1 scores improved, class 0 scores worsened due to the reduction in training data. This can be seen in the testing accuracy and the confusion matrix provided. Let's now see what happened to model performance when I instead utilized training data that oversampled the minority class.

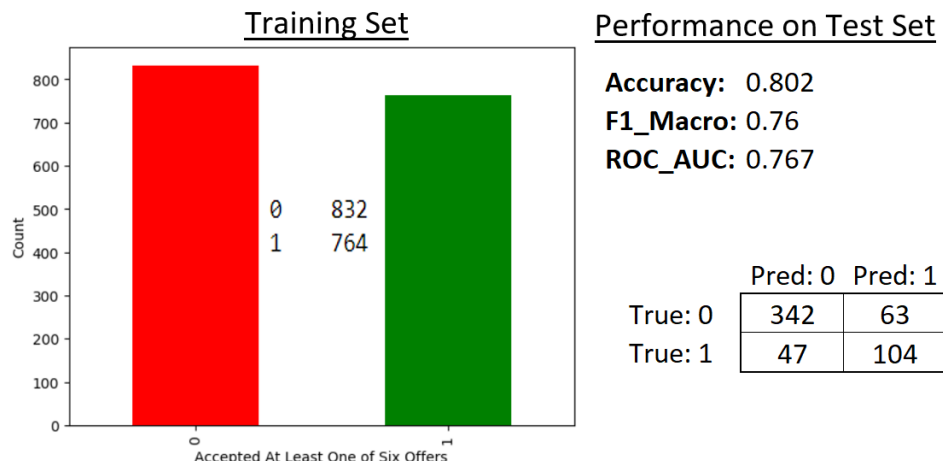
(Oversampling)



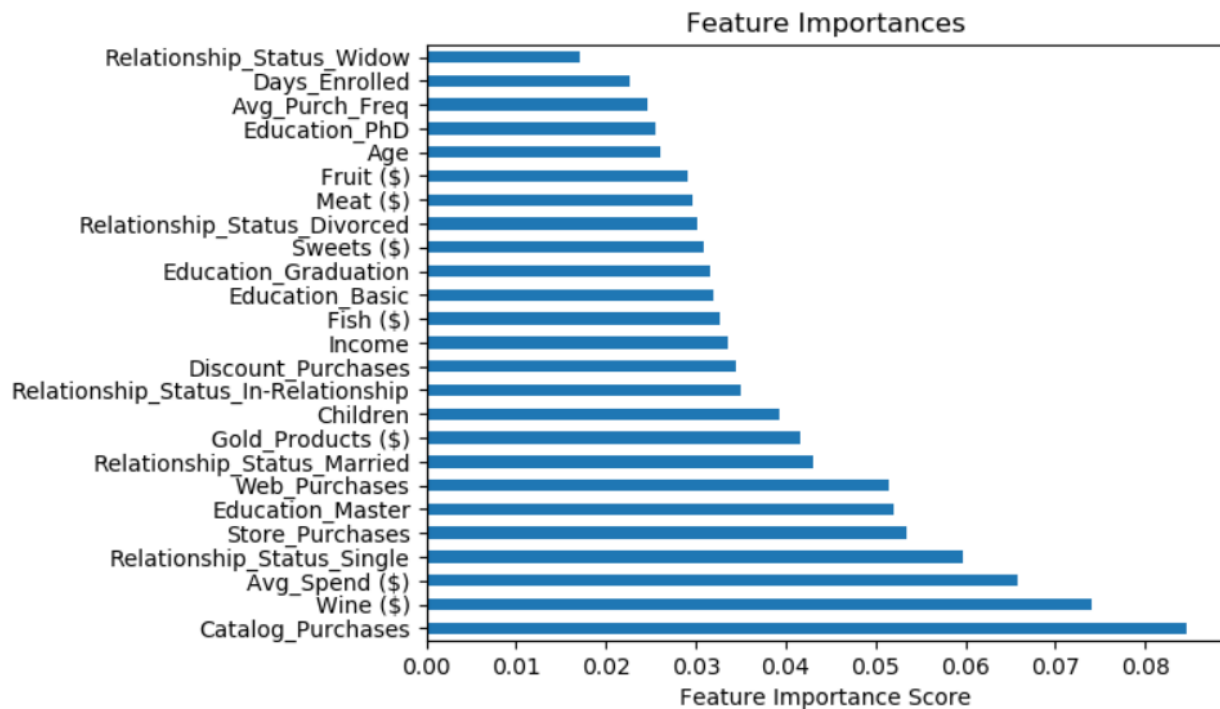
More of the balance I was looking for was provided in the resulting performance on the test set. In comparison to the base model, the F1_Macro score improved, the ROC_AUC improved, and the accuracy remained constant. In comparison to the model utilizing training data that undersampled the majority class, this model's class 1 performance worsened. This can be seen in the lower ROC_AUC score and confusion matrix.

With oversampling and undersampling both containing their own respective benefits, I believed a hybrid approach of the two methods could improve model performance. To do so, I reduced the oversampling of the minority class and the undersampling of the majority class by half. The resulting training set and performance metrics are provided below.

(Hybrid – Undersampling & Oversampling)

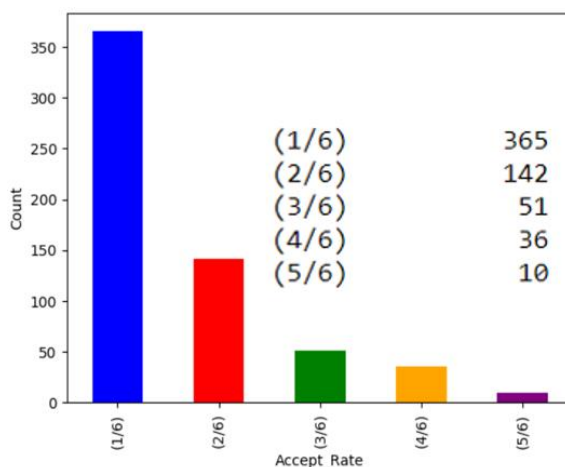


Here we see that this model achieved the best F1_Macro and ROC_AUC scores on the test set and met the original criteria of increasing class 1's f1 score, limiting any decrease in class 0's f1 score, improving the ROC-AUC score, and maintaining a similar accuracy on the test set to the base model. Below I'll provide the feature importance scores from the model.



Machine Learning (Supervised – Predicting Offer Acceptance Rate (Low/Medium/High))

In total, 604 customers (27%) accepted the offer at least once during the six attempts. Roughly 60% of these customers accepted the offer just once, representing the most common acceptance rate. Below I have provided a figure representing the acceptance rate value counts and how I went about creating three separate acceptance rate classes to later predict.

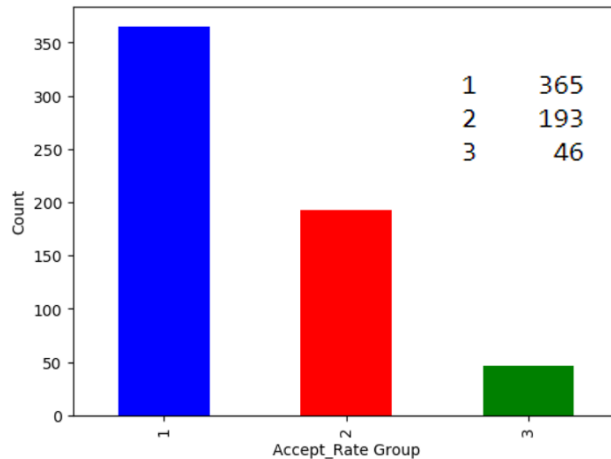


Accept_Rate: (1/6) → Low Acceptance Rate

Accept_Rate: (2/6), (3/6) → Medium Acceptance Rate

Accept_Rate: (4/6), (5/6) → High Acceptance Rate





With the above data containing the correct labels, one could find themselves splitting the data into training and test sets, fitting the model to the training data, and assessing the performance of the model on the test data. This, of course, was the method utilized in the last section. An important difference is that there was much more data (2,227 customer records) at my disposal compared to this classification problem (604 customer records).

If I were to implement the above approach to build an acceptance rate classifier, the model's performance would be significantly biased by the training-test split proportion and which customer records are assigned to each split. This bias also introduces larger performance variance on the test set, hurting the client's ability to receive reliable estimations of the model's performance and effectively implement strategies based on its predictions.

Cross-Validation ("CV") gives a more robust assessment of a model's generalization performance as it utilizes all the data for training. A quick summary of CV is provided below for the reader.

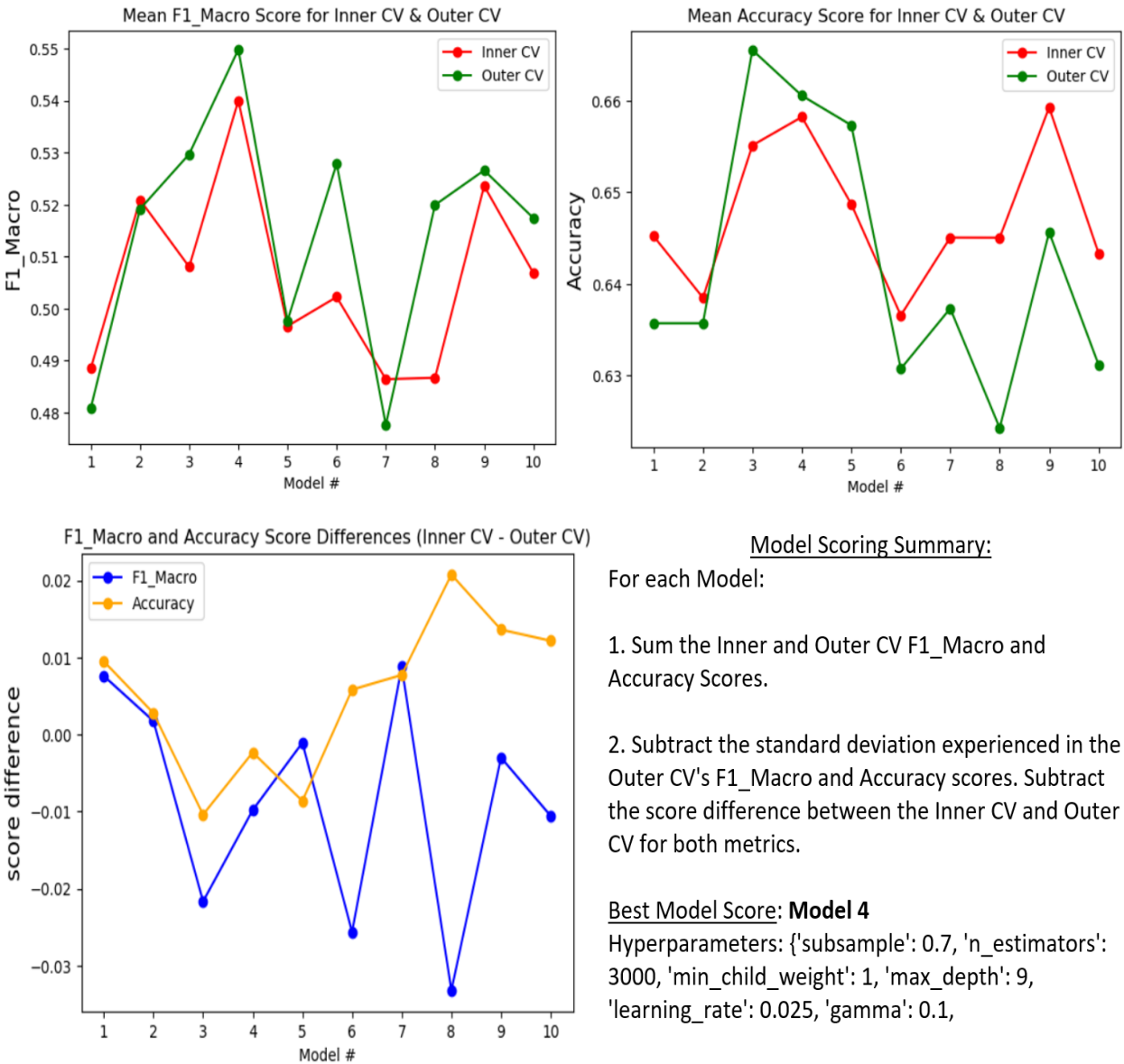
1. Split data into k folds
2. For i in k (the number of folds)
 - a. Fit a model to all data but the k-th fold
 - b. Evaluate the model's performance on the k-th fold

Because CV can be used for assessing the performance of a model, it can also be used to compare the performance of different models. In this case, I compared the performance of different XBG models, each containing their own optimized hyperparameter values. To do so, I utilized a Nested Cross-Validation ("NCV"), which estimates the generalization error of the underlying model and its hyperparameter search. Simply choosing the parameters that maximize a non-nested CV would bias the model to the dataset (as mentioned previously), resulting in an overly-optimistic score.

NCV allowed me to effectively use ten different train/validation/test set splits. In the inner loop (RandomizedSearchCV), the mean F1_Macro score was maximized by fitting a model to each

training set, and then selecting the best combination of hyperparameters over the validation set. In the outer loop (cross_val_score), the generalization error was estimated by averaging the test set scores over several dataset splits.

Though each of the model's best XGB hyperparameters were chosen based on the best average F1_Macro score on the validation set, the validation and test set's mean accuracy score were also recorded to help in evaluating each model's performance. The standard deviation experienced in the outer CV's F1_Macro and accuracy scores on the test set and the difference in the inner and outer CV's mean scores were recorded as well. Below I have provided a summary of the results and how I went about selecting the best model for the client.

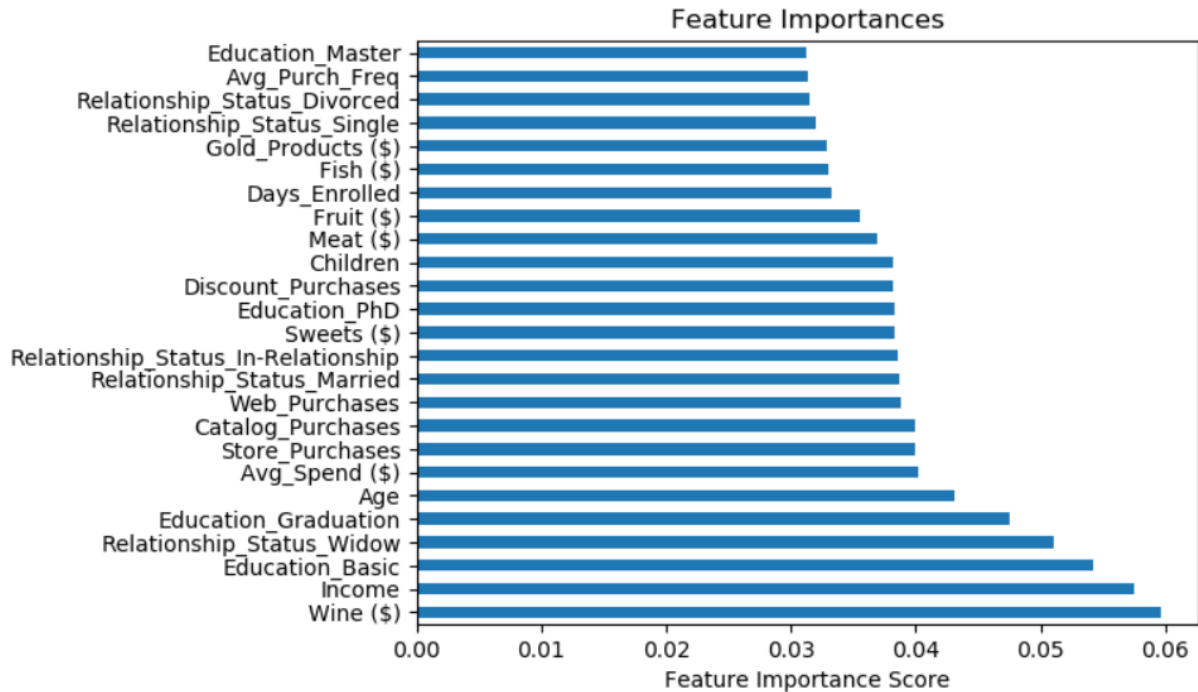


Model Scoring Summary:

- For each Model:
1. Sum the Inner and Outer CV F1_Macro and Accuracy Scores.
 2. Subtract the standard deviation experienced in the Outer CV's F1_Macro and Accuracy scores. Subtract the score difference between the Inner CV and Outer CV for both metrics.

Best Model Score: **Model 4**
Hyperparameters: {'subsample': 0.7, 'n_estimators': 3000, 'min_child_weight': 1, 'max_depth': 9, 'learning_rate': 0.025, 'gamma': 0.1,

Model	In(f1)	In(acc)	Out(f1_mean)	Out(f1_std)	Out(acc_mean)	Out(acc_std)	Diff(f1)	Diff(acc)
4	0.540029	0.65823	0.549839	0.044841	0.660555	0.023937	-0.00981	-0.002325



Model 4 was selected as the best model for the client based on its performance in accuracy, F1_macro, standard deviation experienced (less variance in its performance), and difference in inner CV and the outer CV performance (the better performance experienced in the outer CV indicates the model did not overfit the data from the training set).

(Conclusions)

To summarize the work presented in the report, I have performed an analysis of customer features and their impact on the acceptance of the bundle offer, provided a customer segmentation based on value to the client, trained a model to predict the acceptance of at least one of six offer attempts ("Model 1"), and trained a model to classify the rate at which customers will accept the offer ("Model 2").

The customer features that added value to the client in the segmentation were closely aligned with the features that improved the acceptance/acceptance rate of the bundle offer. These features, along with features deemed to be unfavorable, are provided below.

Most Favorable Customer Features:

- Higher purchase frequency and spend per purchase
- Higher Income → Larger amount spent on wine and meat products
- Increasing percentage of catalog purchases

Most Unfavorable Customer Features:

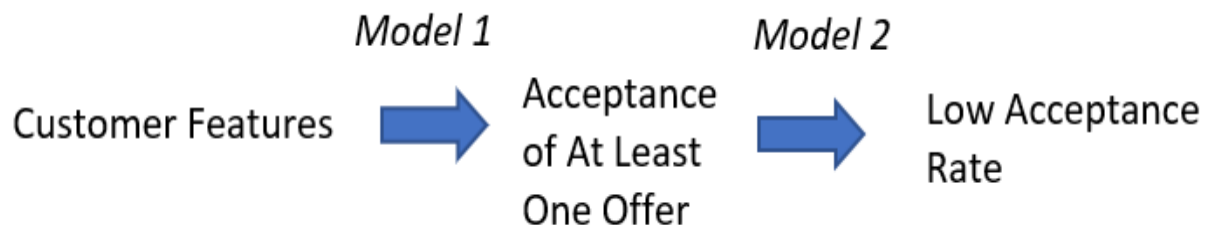
- Lower purchase frequency and spend per purchase
- Lower Income → Spend more on gold products, fish, sweets and fruit
- Increasing percentage of discount purchases
- Younger and have more children

(Future Work)

- The client can now proceed to run several more campaigns for the offer in which they only send it to customers predicted by Model 1 to accept at least one of six offer attempts. I can then analyze the improvement in the acceptance percentage of the offer from the resulting data.
- From those who accepted the offer at least once during the new campaigns, I would have more training data at my disposal to improve Model 2's performance.
- If the client commences the tracking of each customer's purchase frequency and average spend for a certain time (Ex: 1 month, couple of weeks) before running each new campaign, I believe we can get more predictive power from those features.
- The client and I could then begin optimizing the timing of future campaigns to certain customers. The optimization could be based on when their last purchase was, their average purchase frequency, what they most frequently purchase, and/or what they last purchased.

(Recommendations to the Client)

With the two models at the client's disposal, they can use Model 1 to predict if a customer will have an acceptance rate of at least "(1/6)" or not. Should they meet this minimal acceptance rate, the client can utilize Model 2 to predict if the customer will have a low, medium, or high acceptance rate.



Note: For any newly acquired customers, we can set a minimum amount of purchases before their data is utilized to predict their response to the offer and their value to the client. Focusing marketing efforts towards those who are predicted to be of higher value could prove worthwhile in retaining these customers.

The client can choose to formulate different approaches in sending the offer to customers who are predicted to belong to the low acceptance class versus the high acceptance class. Low acceptance customers could receive the offer less often than those who are high acceptance customers. For medium acceptance rate customers, a hybrid of the two approaches can be utilized. The client could also choose to send the offer at the rate of the low acceptance group or the high acceptance group (depending on the marketing budget for the offer).

The number of purchases made through the client's catalog stood out as one of the most important features in predicting a customer's acceptance rate and their value to the client. Thus, the client can work towards efficiently advertising the offer within the catalog. The placement of the advertisement and its presentation/composition within the catalog should be thoughtfully considered.

For customers predicted to not accept the bundle offer or contain low value to the client, I suggest utilizing smaller bundle offers to encourage higher spending/better retention. The low value group contained poor response to the size and price of the bundle offer, so presenting smaller bundle offers of the group's favorite products could be beneficial. When previously analyzing the percentage of revenue generated by product for each group, the low value group contained the highest percentage for gold products, fish, sweets, and fruit.

Certain combinations of these products could be offered, with the client recording the associated customer response. With the collected data, the client could optimize their marketing budget for lower-value customers. For this potential approach towards low value customers, it's important to note I'm trying to preserve some of the benefits of the original bundle offer (higher spending/exposure to the client's products). The client could find it most useful/efficient to simply offer discounts on single products.

(Consulted Resources)

Packages:

- Pandas - <https://pandas.pydata.org/docs/>
- Numpy - <https://numpy.org/doc/>
- Sklearn - <https://scikit-learn.org/stable/>
- Matplotlib - <https://matplotlib.org/3.2.1/contents.html>
- Seaborn - <https://seaborn.pydata.org/>
- Xgboost - <https://xgboost.readthedocs.io/en/latest/>
- Imblearn - <https://imbalanced-learn.readthedocs.io/en/stable/api.html>

Internet Resources:

- Analytics Vidhya - <https://www.analyticsvidhya.com/>
- Machine Learning Mastery - <https://machinelearningmastery.com/>

- TowardsDataScience - <https://towardsdatascience.com/>
- StackOverflow - <https://stackoverflow.com/>