Propensity Testing

Genesis Taylor

In order to create a model that would be able to predict whether a customer would join a loyalty program, I started by importing and examining the data. There was a total of 7 columns, Unnamed: 0, purch_amt, gender, card_on_file, age, days_since_last_purch, and loyalty, with loyalty assumed as the column I was to predict. There were no null values so the next proponent that I checked were outliers. There were 3 columns with a large enough amount of unique values that I wanted to check. Those were purch_amt, days_since_last_purch, and age. When looking at the values for each of these columns, they all had negative values that were odd for the set, so I considered the size and deleted them. However, for age I decided to delete ages under 13 because usually people under 13 aren't allowed to sign up for loyalty cards, so their data was irrelevant to the project. For purch_amt I decided to handler more outliers besides negative values as well. The outliers on the higher end of the data were not at a level that I was comfortable with, so I used the formula from interquartile range to handle them. Once I executed the formula, I deleted any rows that were greater than the maximum from the formula created.

After outliers were handled, I feature engineered other columns from the data. For age, I created age groups as a separate column. I used the days_since_last_purchase column to created two more columns that included weeks and months since last purchase. For purch_amt, I created a column that groups the purchase amounts in $25 increments. From there, I decided to make "Unnamed: 0" my index because it had the same amount of unique values as there were rows. I then checked the variable that I was predicting to see its balance. There was a total of 86,666 False rows and 18,818 True rows. With that in mind, I continued setting up my data for prediction. I then separated the object and non-object columns for encoding. I encoded the non-objects using label encoder the recombined the two data sets. I check the correlation of the data and saw that there were non that were too closely correlated so I continued onto setting up the data for machine learning.

From there I broke the data into X and y and then performed and train_test_split with a test_size of 25%. After that was completed, resampling was done on the data with undersampling as the method. I chose undersampling because there was enough data to where I did not have to risk the errors that would come from oversampling, and I still would have enough data to accurately predict loyalty. Once every step for resampling was done, I scaled the data using Standard Scaler. I then began to run a few classifiers that I was familiar with from previous projects. Those were support vector, random forest, adaboost, gradient boosting, xgboost, extra trees, and lightgbm. Once those were done, I took the highest performing classifier (support vector) , altered the parameters which resulted in improved scoring.