CSE145: HW2 Report

Daniel Chen
Student ID: 1602167
Email: dchen61@ucsc.edu

May 17th, 2020

1. Data Understanding

The file 'Customer_Churn.xlsx' contains 20,000 data instances, each with 12 attributes. The five categorical attributes are: COLLEGE, REPORTED_SATISFACTION, REPORTED_USAGE_LEVEL, CONSIDERING_CHANGE_OF_PLAN, and LEAVE. The seven numerical attributes are: INCOME, OVERAGE, LEFTOVER, HOUSE, HANDSET_PRICE, OVER_15MINS_CALLS_PER_MONTH, and AVERAGE_CALL_DURATION.

Here are some general statistics about each attribute:

Categorical Attributes:

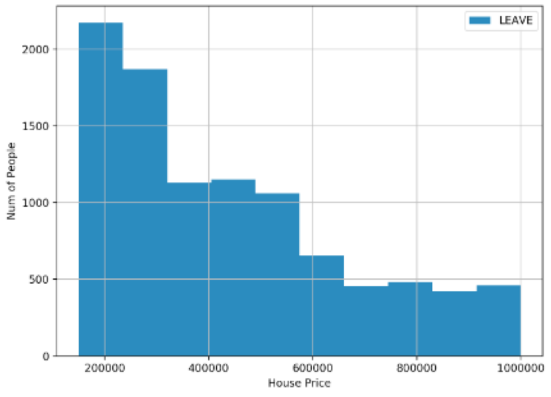
	COLLEGE	SATISFACTION	USAGE LEVEL	CHANGE PLAN	LEAVE
# unique classes	2	5	5	5	2
top class	'one'	'very_unsat'	'little'	'considering'	'stay'
freq	10048	7909	7875	7920	10148

Numerical Attributes:

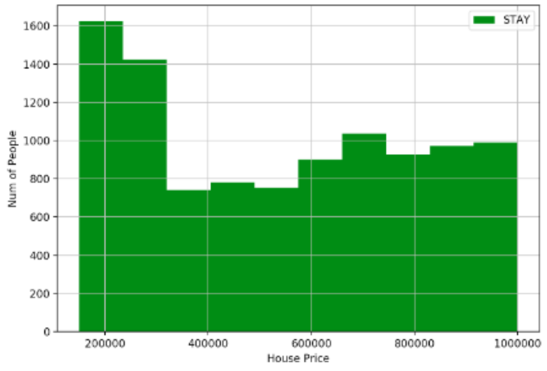
	INCOME	OVERAGE	LEFTOVER	HOUSE	HANDSET PRICE	OVER 15MINS	AVG CALL DUR
mean	80281.45	85.98	23.90	493155.26	389.62	8.00	6.00
standard dev	41680.59	85.99	26.82	252407.88	213.82	8.93	4.40
min	20007	-2	0	150002	130	0	1
25%	42217	0	0	263714.25	219	1	2
50%	75366.5	59	14	452259.5	326	4	5
75%	115881.75	179	41	702378	533.25	15	10
max	159983	335	89	999996	899	29	15

In this dataset, there are no duplicates or any missing values, so no action is needed to handle these issues. For the categorical attributes, the table shows how many unique classes are in the attribute, what is the most frequent class, and what the frequency of that class is. For the numerical attributes, the table shows the mean, standard deviation, the min value, the 25% quantile, 50% quantile, 75% quantile, and the max value. For the binary categorical attributes (COLLEGE and LEAVE), the data seems to be quite balanced. However, for SATISFACTION, USAGE_LEVEL, and CHANGE_PLAN, there are 5 unique classes and the most frequent (top) class makes up about 40% of the entire attribute.

To build my prediction models, I had to transform the categorical attributes into numerical ones. For COLLEGE, 'zero' became -1 and 'one' became 1. For LEAVE, 'LEAVE' became -1 and 'STAY' became 1. For the multi-class categorical attributes, I one-hot encoded them.



(a) People who left



(b) People who stayed

Figure 1: Comparison between house price and the number of people who left/stayed

Figure 1 shows two histograms comparing the house price and number of people who left/stayed. From these plots, people who have a higher house price tend to have a higher probability staying with the mobile phone provider. This figure is meaningful because it shows that house price is a useful attribute to look at when predicting whether a customer churns or not. This attribute will have a significant impact in classifying an instance.

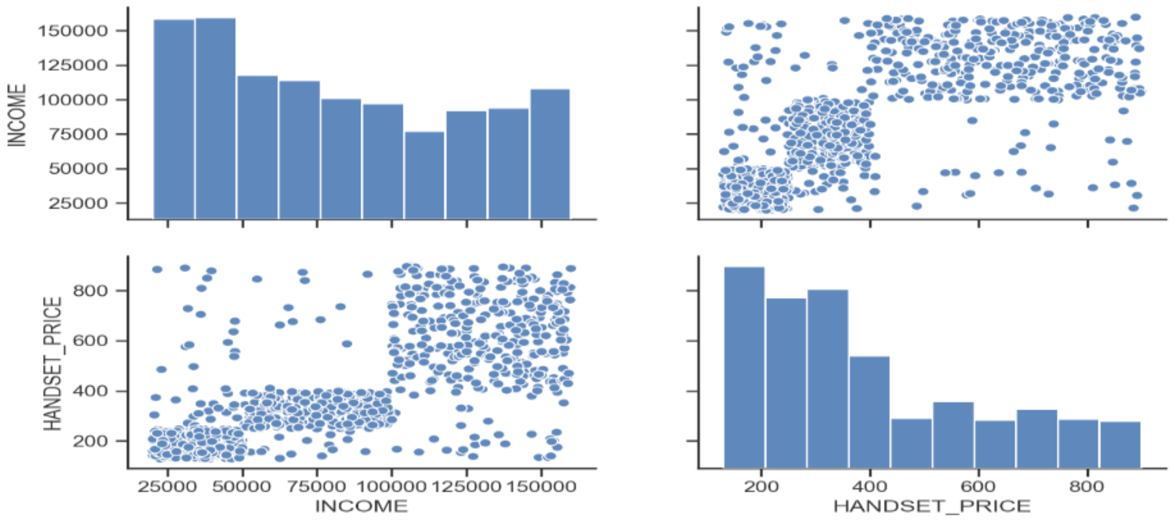


Figure 2: Pairplot for INCOME and HANDSET_PRICE

Figure 2 is a pairplot for the attributes INCOME and HANDSET_PRICE. I took a sample of 1,000 instances out of the 20,000 total to make it easier to visualize the data. From this figure, it is clear that there is a positive correlation between these two attributes. That is, as income increases, the price of the handset generally increases as well. Knowing that there is a correlation between these two attributes is meaningful, because it tells you that one of the two attributes won't have as much of an impact as others in a predictive model. If two variables had a perfect correlation, they would be considered as redundant, and only one is needed. After finding that there is a correlation, I calculated the Pearson's correlation coefficient, which was 0.727. Since the coefficient lies between 0.50 and 1.00, it is considered a strong positive correlation. I did not consider INCOME and HANDSET_PRICE to be redundant variables, because it is not a perfect correlation; however, it could still help in making a prediction, even though its impact would be less than the other attributes.

2. Customer Segmentation with k-Means

To find a high-quality clustering using K-means, I used both Python and RapidMiner Studio for this part. I used Python to find the optimal value for K, and RapidMiner Studio for everything else.

For K-means, the label column (LEAVE) will be ignored because this is an unsupervised algorithm. Including the label column would skew the clustering. To apply K-means, all the attributes need to be numerical. So for the categorical attributes in my dataset, I transformed them into numerical attributes as stated in Part 1. The classes for the categorical attributes were all encoded into numbers. Then, I applied z-transformation to normalize each data instance. The attribute LEFTOVER ranges from 0 to 89 and INCOME ranges from 20,007 to 159,983. I did not want INCOME to dominate other attributes when using distance measures. After normalizing, I applied the K-means algorithm on my dataset 10 times to test different numbers of K. So, I tested K=1, K=2, ..., K=10. After each run, I calculated the squared error for the clustered points using the sum of squared Euclidean distances of samples to their closest cluster center.

This is the graph after plotting the squared error for each number of clusters:

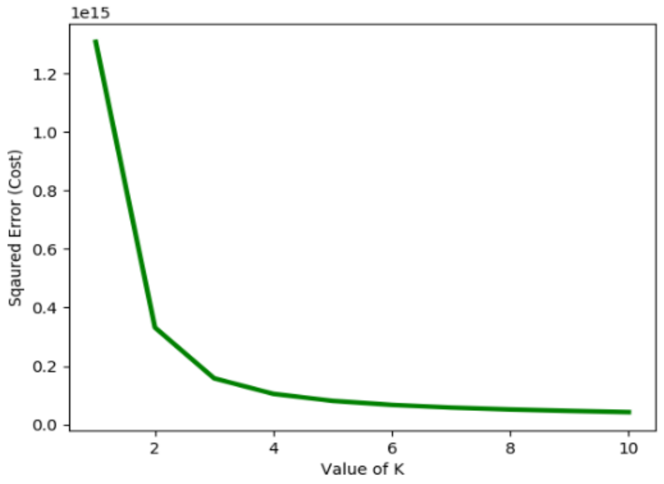


Figure 3: Plot for K value and squared error

To determine what the optimal K value is, I used the elbow method. From the graph, K=3 is where the bend (knee) is located, so K=3 is the optimal K value for my dataset.

Here are some statistics for my clustering:

	Num of items	Avg. within centroid dist
Cluster_0	9525	-21.155
Cluster_1	9450	-21.103
Cluster_2	1025	-19.134
Overall	20000	-21.027

To look at the performance of my clustering, I used the average within the centroid distance. This is how far all of the data instances within each cluster are from their centroid, using Euclidean distance.

For each of the three clusters, the centroid for each attribute seems to be very spread out. One cluster's centroid for a certain attribute is on one extreme, another cluster's centroid for that attribute is on the other extreme, and the third cluster's centroid is somewhere in the middle. For example, with the INCOME attribute, the centroid for cluster 0 is 0.008, the centroid for cluster 1 is -0.011, and the centroid for cluster 3 is 0.034.

Here is the plot for the centroids of each cluster:

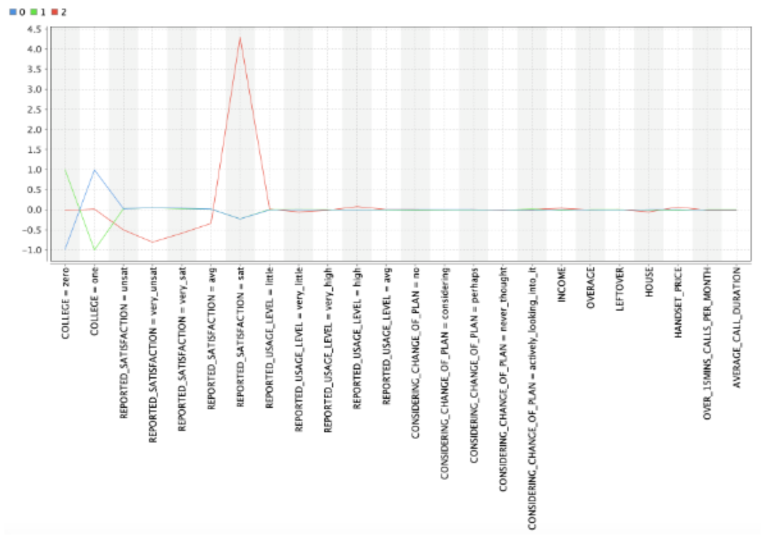


Figure 4: Plot of all the centroids for each cluster

From this graph, there are four things that obviously distinguish the three clusters. Cluster 0 has a huge dip for COLLEGE='zero' and a huge peak for COLLEGE='one'. Cluster 1 has the opposite, where it has a huge peak for COLLEGE='zero' and a huge dip for COLLEGE='one'. However, Cluster 3 stays in between for the COLLEGE attribute, and the centroid is much lower for REPORTED_SATISFACTION=unsat and REPORTED_SATISFACTION=very_unsat. For REPORTED_SATISFACTION=sat, it has a huge peak. For every other attribute, the centroids for the three clusters all fall around each other in a straight and constant line.

3. Explain Predictive Models

For predicting customer churn, the two models I chose to do the predicting were simple decision tree and gradient boosted decision tree. For the simple decision tree, I used the implementation from scikit-learn, and for the gradient boosted decision tree, I used the implementation from XGBoost.

First, I had to do data preprocessing because the Python libraries I used needed all attributes to be numerical. So, I changed the classes for the binary categorical attributes to -1 and 1, and one-hot encoded the multi-class categorical attributes. Then, I split the 20,000 data instances randomly into a training set, validation set, and a test set with a 80/10/10 split. The models I built will not see the test set at all during training, and this set will only be used to evaluate the performance of my models after training and hyperparameter tuning.

After data preprocessing, I built my models. For the simple decision tree and gradient boosted decision tree, there were several hyperparameters that I tried tuning. For my simple decision tree, I only tuned the max depth of the tree. For my gradient boosted decision tree, I tuned the max depth of the tree, the learning rate, the minimum sum needed in a child node (min_child_weight), and the different number of trees (n_estimators). Having a tree's max depth be too high will lead to overfitting. While tuning, I would test a range of values for one hyperparameter and keep the rest of them the same. I would keep the value for a hyperparameter that resulted in the highest accuracy on my validation set. At the end, the max depth for my simple decision tree is 4. For my gradient boosted decision tree, the max depth is 5, n_estimators is 500,

min_child_weight is 5, and the learning rate is 0.02.
Once my models were done, I ran them against my test set. The simple decision tree resulted in an accuracy of 70.275% and the gradient boosted decision tree had an accuracy of 71.20%.

Here is the comparison of the models' ROC curves:

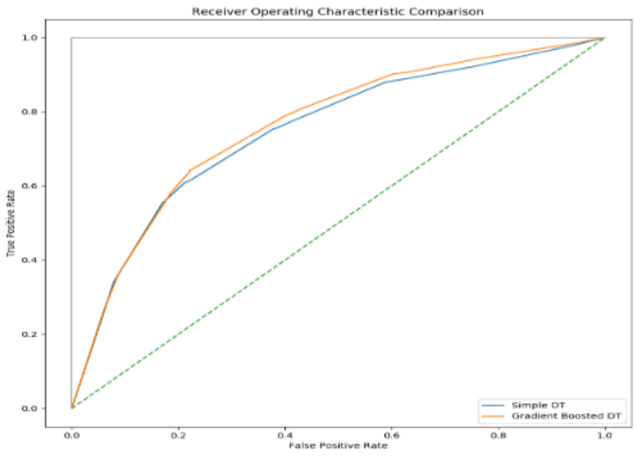
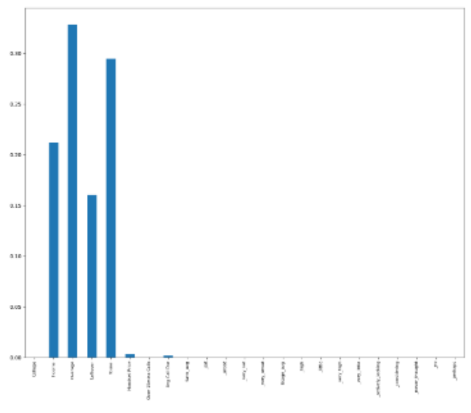


Figure 5: ROC curve comparison for simple DT and gradient boosted DT

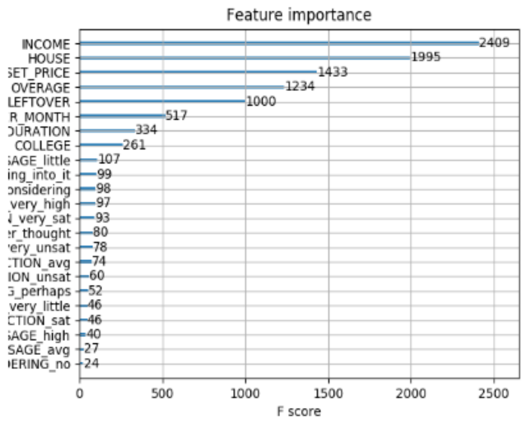
Based on the models' accuracy on the held-out test dataset and by looking at the ROC curve comparison, the gradient boosted decision tree is slightly better than the simple decision tree. This model will be better at predicting customer churn.

The data that has been provided is quite balanced. As reported in Part 1, there are 10,148 instances of 'STAY' and 9,852 instances of 'LEAVE'. So, there are many examples of both cases. Therefore, a model that always reports one label would result in an accuracy that is much worse than the gradient boosted decision tree's accuracy. Although some attributes are not so balanced, it is fine as long as the labels are balanced.

Here are the feature importances for the two models:



(a) Feature importance for simple DT



(b) Feature importance for gradient boosted DT

Figure 6: Feature importances for both models

For the simple decision tree, the most important features for predicting customer churn are: overage, house price, income, and leftover (overage being the most important and importance decreasing from there). For the gradient boosted decision tree, the five most important features are: income, house price, handset price, overage, and leftover (decreasing in importance). Looking at the feature importance is important, because it tells you which attributes are more useful in predicting customer churn.

To reduce churn, I would recommend focusing on looking at people's income and house prices. These were the top two features in predicting customer churn for the best performing model. Specifically, I would work on making the people who have lower incomes and house prices happier, because they are more likely to churn than the opposite.