CSE145 - HW2
Yotam Golan
yogolan@ucsc.edu

1 Introduction

The purpose of this project was to mine the data from Customer_Churn.xlsx, and utilize its 20,000 entries into insightful and workable data. I aimed to predict whether the 'Leave' variable, which symbolizes whether a customer will stay with the company. To this end, I first visualized the data to better understand it. Then, using a k-mean algorithm to cluster the data, I further analyzed its properties. Finally, using a neural network and a decision tree, I tried to predict whether a customer would leave a company (churn).

2 How To Run The Code

This assignment uses Python 3.8 with pandas, matplotlib, keras, and sklearn. To run the code, place all the files (Customer_Churn.xlsx, preprocess.py, predict.py, visual.py, kfinder.py) in the same folder. Run preprocess first to process the Customer_Churn into a more useful .csv (Customer_Churn_processed.csv). The order *predict.py* (which has the predictive algorithms, a neural network and a decision tree) and *visual.py* (which has a bar graph, correlation map, and the k-means) are run is inconsequential. No file requires any arguments and can be run directly from the CMD, assuming the correct libraries are available.

Note that all work is being on the created Customer_Churn_processed.csv, so preprocess must be run first.

3 Data Understanding and Visualization

To better understand the given data, I decided to visual the information gain. This is a measure of which attribute improves entropy over its entire segment. To this end, I created a bar graph that ranks the information gain of all attributes in decreasing order. As shown in **Figure 1**, this helps understand the relative importance of each attribute.

The second visualization tool I used was to create a correlation map, which shows correlation between attributes. As shown in **Figure 2**, this correlation map helps visualize how certain attributes relate to each other, and help understand their relative importance and effect.

The code for both of these visualizations is available in visual.py

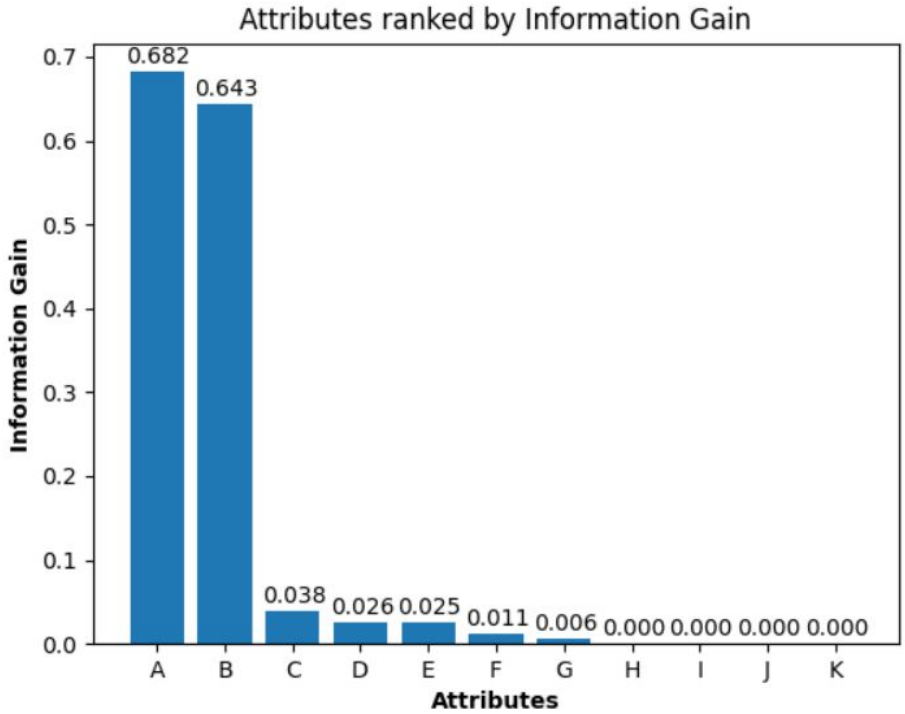

Figure 1: Attributes ranked by their information gain

The attributes left to right are: House, Income, Overage, Handset, Calls Over 15 Mins, Leftover, Average Call Length, Report Satisfaction, College, Plan Change, and Reported Usage.

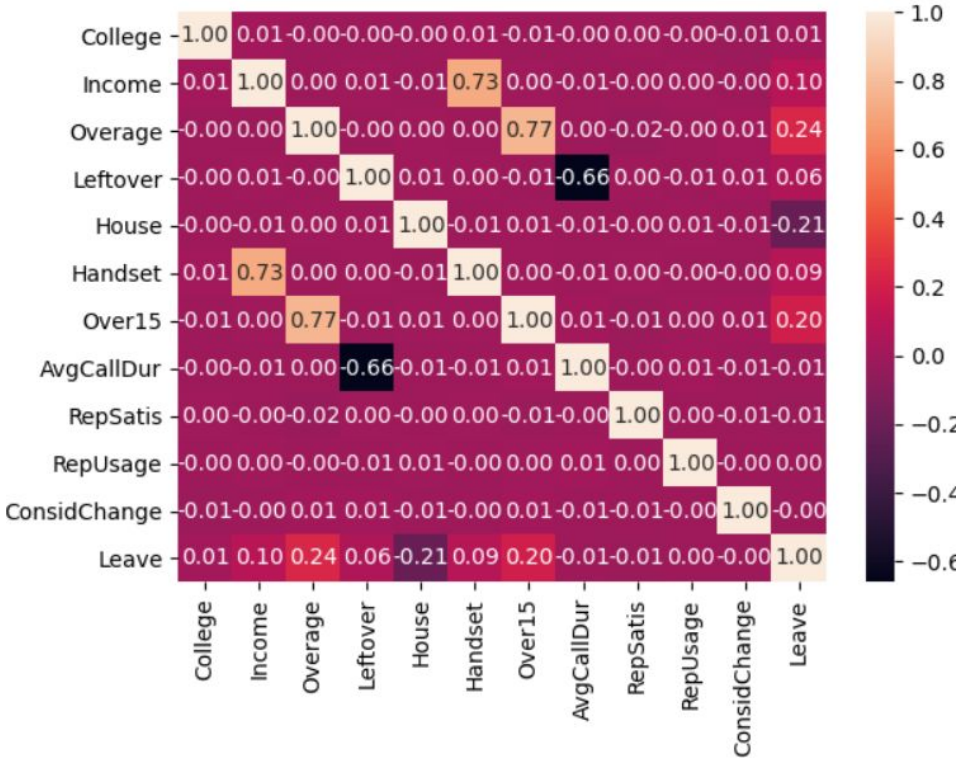

Figure 2: Correlation map

4 Choosing The Proper K

Prior to finding a k-mean model, the data must first be scaled so magnitude differences in the data would be significantly smaller. To this end, I used the sklearn's StandardScaler. Following this, I used sklearn's KMeans method to find the model that would best fit the data.

Finding an appropriate number of clusters involved creating k-mean models recursively, each with a varying number of clusters (k). I checked k from 1 through 20, and decided upon 5 as a good compromise. This was done by creating a sum squared error ploy, and a silhouette plot. Both using sklearn. As seen in **Figure 3** and **Figure 4**, 5 clusters fell within the 'elbow' of the graphs, which was the main reason I selected it.

The code for both of these models is available in *kfinder.py.*

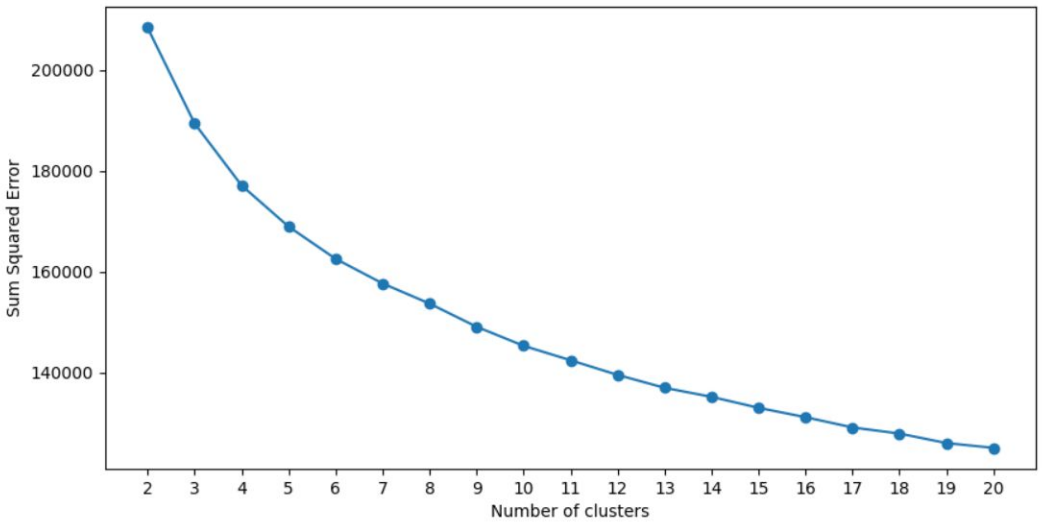

Figure 3: Sum Square Error

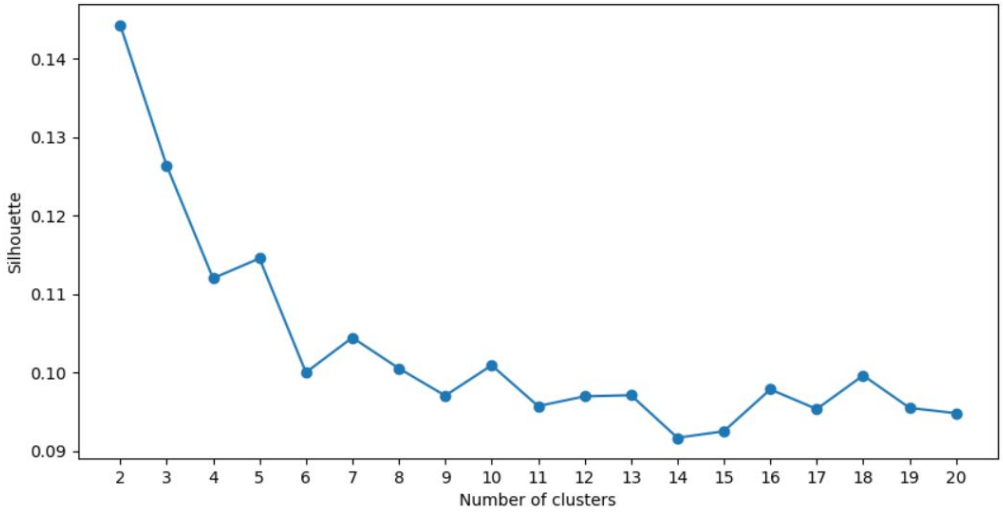

Figure 4: Silhouette

5 Cluster Analysis

The following 5 clusters each have the mean and standard deviations for each attribute within their reach. The values, left to right are: College, Income, Overage, Leftover, House, Handset Price, Over 15 Min Calls per Month, Avg Call Duration, Reported Satis., Reported Usage Lvl., Consid. Change Plan, Leave.

Several attributes are binary values (College and Leave) and are label encoded. Thus, their mean represents the frequency of that attribute within the cluster.

As can be seen, the attributes College/House/Reported Satisfaction/Reported Usage Lvl/Considered Change Plan, had relatively little variations between the clusters. This signifies, to me, that they had little impact on the cluster formation and are likely not important for our needs.

We can see that Cluster 2 had higher average Income, Overage, Handset Price, Over 15 minute Calls per Month, and proportion of 'Leave'. Cluster 1, conversely, had lower Income, Overage, Leftover, Handset Price, and Over 15 Min Calls. However, it also shows a much higher rate of stays. Additionally, cluster 1 also has the most points within. This would indicate to me that the largest portion of the customers tends towards below average values for their attributes.

Cluster 3 tended towards average values in most attributes, and was largely denoted by its very low Handset price, high OVer 15 Min Calls per Month, and above average Leave values. The below average income and large number of points would indicate that this cluster supports the previous observation of the plurality of customers having below average values for their attributes, and tending towards a lower income level.

Clusters 4 and 5 both have marked similarity to clusters 1 and 2 respectively. The biggest difference appears to lie in 4's below average Overage, Handset Price, Call Length, and Leave. 5 appears to consist of the wealthier customers who are not prone to leaving like cluster 2, and use their service significantly less.

Cluster 1: 6026 points

mean	0.508578	56181.198119	32.201584	7.622402	494845.290003	263.230947	2.610195	8.184922	1.573243	1.832234	2.429561	0.326790
std	0.499968	25820.571273	39.809702	9.732596	251499.649155	94.179490	3.133763	3.947429	1.626530	1.516688	1.339895	0.469078

Cluster 2: 2214 points

mean	0.508582	128172.241192	193.700542	24.172990	488598.157182	641.151310	18.893406	6.031165	1.542457	1.780488	2.509485	0.822042
std	0.500039	22030.490681	42.991768	26.786741	250871.545458	159.926735	6.844831	4.416256	1.624417	1.492763	1.317493	0.382564

Cluster 3: 4014 points

mean	0.491715	56175.209064	195.502680	19.973441	489011.130361	263.153509	19.301657	6.249756	1.519250	1.812135	2.551901	0.597953
std	0.499992	25154.666335	40.710297	23.566222	252317.297291	93.219962	6.463617	4.315800	1.627398	1.513459	1.314252	0.490371

Cluster 4: 3702 points

mean	0.487304	66079.329552	45.548352	60.586980	500468.583739	303.721772	3.516748	1.789573	1.583468	1.800648	2.527283	0.510805
std	0.499906	33474.002811	56.604605	18.847426	254378.907628	136.451372	4.479816	1.432539	1.625395	1.515557	1.301667	0.499951

Cluster 5: 3918 points

mean	0.514803	129177.247065	31.794028	18.372384	490546.309597	656.642930	2.584992	6.330015	1.601072	1.824655	2.492853	0.435426
std	0.499845	21365.204561	37.558909	21.759282	252848.558881	151.743099	2.856950	4.250165	1.642202	1.510117	1.328129	0.495876

The raw data is available in *clusters.txt.*

6 Predictive Models

I chose to use a neural network and a decision tree for my predictive models. This was due to my concurrent taking of an applied ML course, and the prominence of decision trees in previous lectures.

Neural Network

The neural network I created was a simplistic fully-feedforward neural network. Using Keras Sequential as my base model, I trained and validated the neural network on a 75/25 training/testing split of the dataset. I tried 3 different layouts for my network, all using accuracy as the primary metric, the ‘adam’ optimizer, and used binary cross-entropy for their loss function. Each model had the starting layer have 100 nodes, and concluded on a single node with a sigmoid function for the Leave/Stay decision. Each was trained for 40 epochs.

The created models achieved the testing accuracies of 72.82%, 74.38%, and 70.19% for figures 5-7 respectively.

The code for these models is available in *predict.py.*

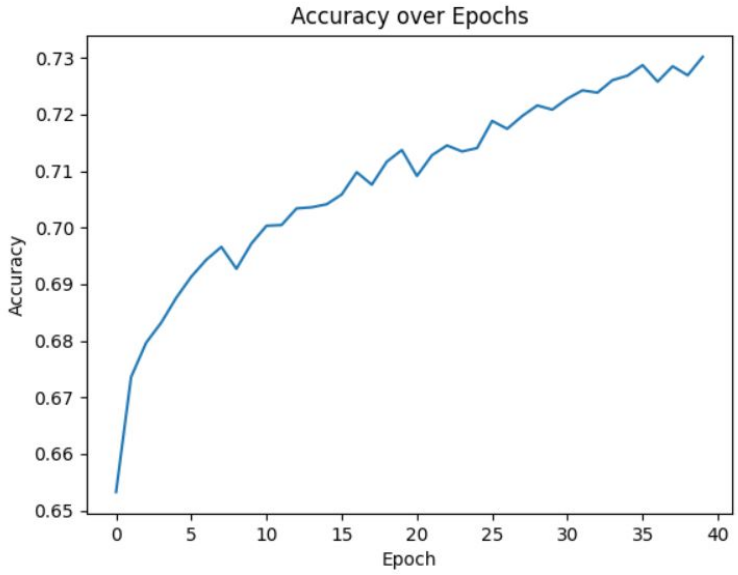

Figure 5: Dimensions:100x30x1, Accuracy: 72.82%

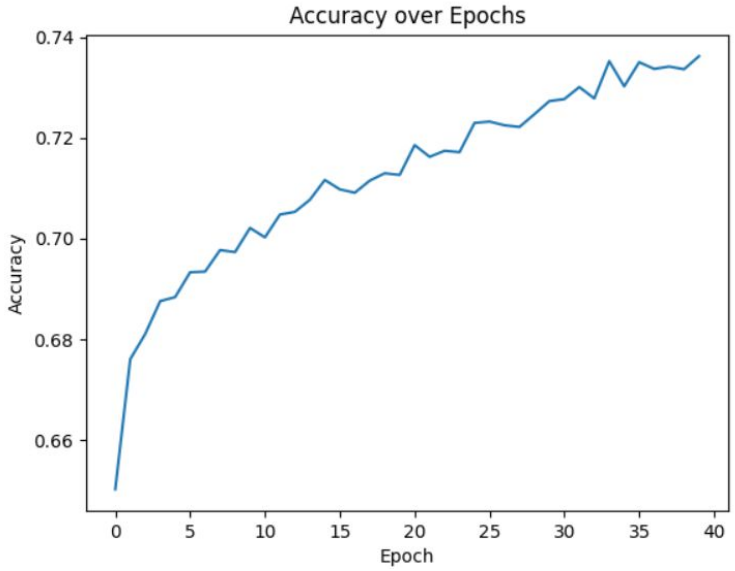

Figure 6: Dimensions:100x30x15x1, Accuracy: 74.38%

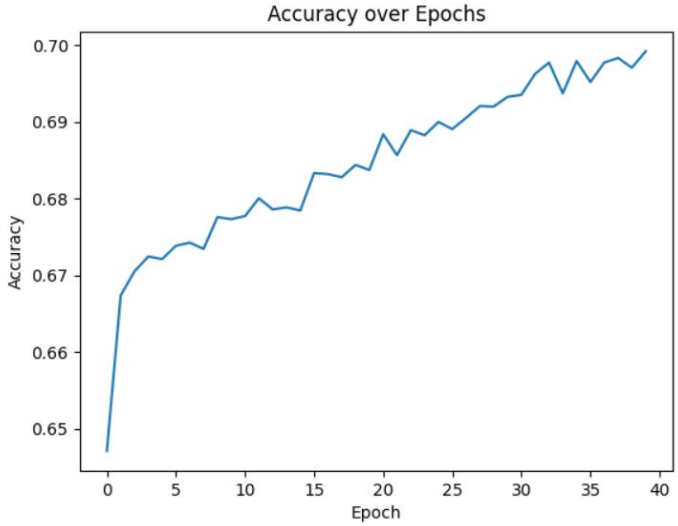

Figure 7: Dimensions:100x1, Accuracy: 70.19%

Decision Tree

I also used a decision tree to try and predict leave/stay. To this end, I used sklearn's DecisionTreeClassifier method. It uses GINI impurity to measure the split quality, and allows for a max depth of 8 to avoid over learning. This model was trained on the same 75/25 dataset split as the neural network. Running it several times, the decision tree had a consistent average accuracy of 70% on the testing data.

The code for this is available in *predict.py.*

7 Conclusion and Analysis

Of the two predictive models, the neural network performed measurably better then the decision tree. While the decision tree managed a final accuracy of 68% on the testing data, the 100x30x15x1 neural network managed a 74.38% accuracy on the testing data. All the models were trained and tested on the same 75/25 training/testing dataset split, so the probability of hidden patterns affecting the gathered data is minimal.

Overall, the data is quite balanced. Looking at the Leave attribute, there are 9852 entries for ‘stay’ and 10149 for ‘leave’. This represents 49.26% as leave, which is close enough to 50% for it to be called balanced. Thus, I did not worry about the impact that unbalanced data could have on model performance.

There are some minor correlations visible in the data. This is most easily seen in the correlation map above, but a minor correlation exists between Overage and churn as well as Over 15 Min Calls and churn. A more minor positive correlation also exists between income and churn. Conversely, a small negative correlation exists between House and churn. The rest of the attributes had relatively insignificant correlations. With these correlations in mind, I would recommend that the company prioritize customers with high income and house value, and low overcharges and long calls.