Evaluation of the Clustering Algorithm for Recommender Systems

Robert Akinie

***Abstract*— In end user perspective, travel and tourism is mostly explorative in nature and repetitive travels to same locations are minimal. As such, travelers have to take decisions regarding their destinations and associated facilities to be consumed without adequate prior or personal knowledge. Tourism recommenders are the best solutions in this scenario. Current recommender systems are computationally intensive, leading to other use-case algorithms that serve the same purpose. This is where clustering algorithms can come into play [5]. This type of recommender systems can reduce high computational power needs. This paper describes a project that works on this idea, and whose goal is to develop a machine learning model to evaluate the usage of clustering for recommender systems.**

I. Introduction

A. Background

In end user perspective, travel and tourism is mostly explorative in nature and repetitive travels to same locations are minimal. So, travelers have to take decisions regarding their destinations and associated facilities to be consumed without adequate prior or personal knowledge. The best option available is to leverage social media and internet, but the amount of time required to extract relevant information is too high. Tourism recommenders are the best solutions in this scenario. Recommender systems helps in terms of automated filtering, processing, personalization and contextualization of the huge volume of data that is available and growing on a daily basis on the internet and the social media [1].

Recommender systems are software tools that take information about a user, then predicting output to the user in terms of suggesting relevant items. The output is given in terms of relevant scores, which are responsible for ranking. Traditional recommender systems use collaborative filtering algorithms [2][3] in social contexts to predict probable options for a user based on past traits of similar users. However, with ever-increasing volume of data, collaborative filtering recommender systems are highly computational intensive and thereby not scalable in big data context [4]. A possible solution is to reduce dimensionality. But, even if we reduce dimensionality of features, we might still have many objects to compute the distance to. This is where clustering algorithms can come into play [5]. This type of recommender systems can reduce high computational power needs. This paper describes a project that works on this idea, and whose goal is to develop a machine learning model to evaluate the usage of clustering for recommender systems.

Section 2 provides a survey for a more in-depth hypothesis for clustering and the algorithm chosen for this experimental analysis. Section 3 lays down the materials and methods used for this investigation. Section 4 details the model development for this experimental analysis. Sections 5, 6 and 7 conclude this paper by providing results and discussions.

II. Literature review

A. A. Clustering

Clustering [6], is an unsupervised learning technique, which consists of assigning items to groups so that the items in the same groups are more similar than items in different groups: the goal is to discover natural (or meaningful) groups that exist in the data. Unsupervised learning is employed in datasets that have only input variables, with no equivalent output values. Its goal is to understand and model the underlying distribution of data to learn more about it [1]. Some use cases for clustering include anomaly detection, market segmentation, text summarization, etc.

Similarity between items, and groups, is determined using a distance measure, such as Euclidean, Minkowski, Manhattan distances, amongst others. The goal of a clustering algorithm is to minimize intra-cluster distances while maximizing inter-cluster distances.

Clustering algorithms are generally classified into two categories: partitional and hierarchical. Partitional clustering algorithms divide data items into non-overlapping clusters such that each data item is in exactly one cluster. Hierarchical clustering algorithms successively cluster items within found clusters, producing a set of nested clusters organized as a hierarchical tree [5]. For the use case and objective, a partitioning algorithm, k-means clustering, was chosen, which will be outlined below.

B. B. k-means Algorithm

k-means algorithm is one of the most popular unsupervised machine learning models. [5] describes the algorithm as follows: The function partitions the data set of N items into k disjoint subsets Sj that contain Nj items so that they are as close to each other as possible according a given distance measure. Each cluster in the partition is defined by its Nj members and by its centroid λj . The centroid for each cluster is the point to which the sum of distances from all items in that cluster is minimized. Thus, we can define the k-means algorithm as an iterative process to minimize (1).

$$E = \sum_1^k \sum_{n \in S_j} d(x_n, \lambda_j) \tag{1}$$

where x_n is a vector representing the n-th item, λj is the centroid of the item in Sj and d is the distance measure. The k-means

algorithm moves items between clusters until E cannot be decreased further.

1. Determine k, the total number of clusters to be formed
2. Identify k initial centroids to start with. Normally a random sampling is performed within the dataset to identify the initial points
3. For each entity in the dataset
   a. Calculate the distance between the entity and each of the k centroids
   b. Assign the entity to the cluster whose centroid is the closest
4. For each of the k clusters
   a. Update the cluster centroid by re-calculating the mean values of all the entities in the cluster.
5. Iteratively minimize the total within cluster variation. That is, iterate steps 3 and 4 until there is no changes to cluster assignments or the maximum number of iterations are reached

Figure 1. k-means algorithm

In [1], four clustering algorithms, k-means, k-mediods, Clustering for Large Applications, and Fuzzy C-means clustering were considered in the evaluation of partitioning clustering algorithms for social media data in tourism domain. As per literature, no single partitioning clustering algorithm is considered as superior for all clustering requirements. From the results gathered during this analysis, k-means algorithm outperformed other partitioning clustering algorithms for the datasets used. This reason, as well as its simplicity and efficiency [5] is what leads to the choice of algorithm in model development. Also, the dataset used is for this model is a social media data in the same domain.

III. Materials and Methods

A. Proposed Methodology

The proposed method for this project is the use of a machine learning model to determine the relationship between several travelers. This relationship is determined with clustering, specifically k-means clustering algorithm, to find similarity between traveler ratings of these destinations. The project would also focus on determining the optimal number of groups to put these travelers in. Results would then be used to evaluate the use case of clustering for recommender systems.

B. Datasets

We consider a travel reviews dataset, populated from the Web. This dataset contains review on destinations in 10 categories from several travelers. Each traveler rating is Each traveler rating is mapped as Excellent (4), Very Good (3), Average (2), Poor (1), and Terrible (0) and average rating is used against each category per user. Each category represents traveler feedback on a certain destination. The following Table 1 shows some information about the dataset.

The k-means algorithm [1] is explained in Figure 1.

TABLE I. DETAILS OF DATASET

Dataset	User's average feedback/rating information on 10 categories of attractions in East Asia captured from tripadvisor.com
Data Size	(980, 11)
Data Type	Object, float64
Attributes	11

C. Preprocessing

In order to develop a machine learning model, the data used must be prepared and cleaned. This is to identify inaccurate, corrupt or irrelevant data points, and then modifying, replacing or removing such instances. This section shows the steps taken in preprocessing the data, as well as showing the results of some of the steps.

The dataset used is first looked at, in order to give the Machine Learning Engineer a general state of the data. A peek of the data can be seen in Figure 2. Descriptive Statistics of the data provides an initial look at the data distribution. These statistics proves such info such as the mean, median and standard deviation of each attribute in the data. The statistics also gives an initial look into potential inaccurate, corrupt, or missing data points, as well as outliers.

Outliers are data points that differ significantly from other observations the data, and these may be generated either from measurement variability, or experimental error. For this project, outliers would be defined as data points greater/less than three standard deviation away from the mean. Based on the number of outliers relative to the dataset, they would either be removed, or replaced with the mean of the data. Outliers can be visualized using Box Plots. Box plots generally display five-number summary of attributes of selected data, as well as outliers. Figures 2 and 3 display the data peek, and Descriptive Statistics. Table 2 shows the number of outliers detected per attribute. Figures 4 and 5 display the statistics after data is modified, and a box plot of the data before modification.

TABLE II. OUTLIER OBSERVATION

CATEGORY	FREQUENCY
1	17
2	21
3	1
4	20
5	7
6	2
7	1
8	7
9	10
10	0

	User ID	Category 1	Category 2	Category 3	Category 4	Category 5	Category 6	Category 7	Category 8	Category 9	Category 10
0	User 1	0.93	1.80	2.29	0.62	0.80	2.42	3.19	2.79	1.82	2.42
1	User 2	1.02	2.20	2.66	0.64	1.42	3.18	3.21	2.63	1.86	2.32
2	User 3	1.22	0.80	0.54	0.53	0.24	1.54	3.18	2.80	1.31	2.50
3	User 4	0.45	1.80	0.29	0.57	0.46	1.52	3.18	2.96	1.57	2.86
4	User 5	0.51	1.20	1.18	0.57	1.54	2.02	3.18	2.78	1.18	2.54
5	User 6	0.99	1.28	0.72	0.27	0.74	1.26	3.17	2.89	1.66	3.66
6	User 7	0.90	1.36	0.26	0.32	0.86	1.58	3.17	2.66	1.22	3.22
7	User 8	0.74	1.40	0.22	0.41	0.82	1.50	3.17	2.81	1.54	2.88
8	User 9	1.12	1.76	1.04	0.64	0.82	2.14	3.18	2.79	1.41	2.54
9	User 10	0.70	1.36	0.22	0.26	1.50	1.54	3.17	2.82	2.24	3.12
10	User 11	1.47	1.00	0.70	0.75	1.66	2.76	3.18	2.89	1.66	2.62
11	User 12	0.96	2.96	0.29	0.38	0.88	2.08	3.17	2.93	1.66	3.42
12	User 13	0.74	1.44	2.75	0.45	0.98	1.74	3.20	2.87	1.38	2.34
13	User 14	0.58	1.64	2.27	0.45	1.26	1.72	3.19	2.91	2.30	2.74
14	User 15	0.96	1.68	2.29	0.51	1.20	2.84	3.20	2.82	2.02	2.46
15	User 16	1.25	2.52	1.76	0.50	1.46	2.08	3.19	2.74	1.41	2.32
16	User 17	0.86	1.04	1.76	0.34	0.06	1.10	3.18	2.73	1.15	2.98
17	User 18	0.61	1.96	2.49	0.66	1.34	1.78	3.20	3.04	1.15	2.42
18	User 19	0.67	1.36	1.36	0.38	0.82	3.38	3.18	2.86	1.79	2.80
19	User 20	0.80	1.04	2.10	0.58	1.18	1.98	3.19	2.93	1.22	2.48

Figure 2. Data peek

	Category 1	Category 2	Category 3	Category 4	Category 5	Category 6	Category 7	Category 8	Category 9	Category 10
count	980.000000	980.000000	980.000000	980.000000	980.000000	980.000000	980.000000	980.000000	980.000000	980.000000
mean	0.893194	1.352612	1.013306	0.532500	0.939735	1.842898	3.180939	2.835061	1.569439	2.799224
std	0.326912	0.478280	0.788607	0.279731	0.437430	0.539538	0.007824	0.137505	0.364629	0.321380
min	0.340000	0.000000	0.130000	0.150000	0.060000	0.140000	3.160000	2.420000	0.740000	2.140000
25%	0.670000	1.080000	0.270000	0.410000	0.640000	1.460000	3.180000	2.740000	1.310000	2.540000
50%	0.830000	1.280000	0.820000	0.500000	0.900000	1.800000	3.180000	2.820000	1.540000	2.780000
75%	1.020000	1.560000	1.572500	0.580000	1.200000	2.200000	3.180000	2.910000	1.760000	3.040000
max	3.220000	3.640000	3.620000	3.440000	3.300000	3.760000	3.210000	3.390000	3.170000	3.660000

Figure 3. Data Statistics

	Category 1	Category 2	Category 3	Category 4	Category 5	Category 6	Category 7	Category 8	Category 9	Category 10
count	900.00000	900.000000	900.000000	900.000000	900.000000	900.000000	900.000000	900.000000	900.000000	900.000000
mean	0.87300	1.316044	1.027433	0.498689	0.922000	1.831767	3.180911	2.833800	1.552122	2.803422
std	0.28125	0.404957	0.786580	0.140036	0.399966	0.528134	0.007811	0.130946	0.336284	0.319471
min	0.34000	0.000000	0.130000	0.170000	0.060000	0.380000	3.160000	2.480000	0.740000	2.260000
25%	0.67000	1.080000	0.290000	0.410000	0.640000	1.460000	3.180000	2.740000	1.310000	2.540000
50%	0.83000	1.280000	0.870000	0.490000	0.880000	1.780000	3.180000	2.820000	1.500000	2.780000
75%	1.02000	1.520000	1.580000	0.570000	1.200000	2.180000	3.180000	2.910000	1.760000	3.040000
max	1.87000	2.760000	3.120000	1.300000	2.180000	3.380000	3.200000	3.230000	2.620000	3.660000

Figure 4. Data Statistics of modified data

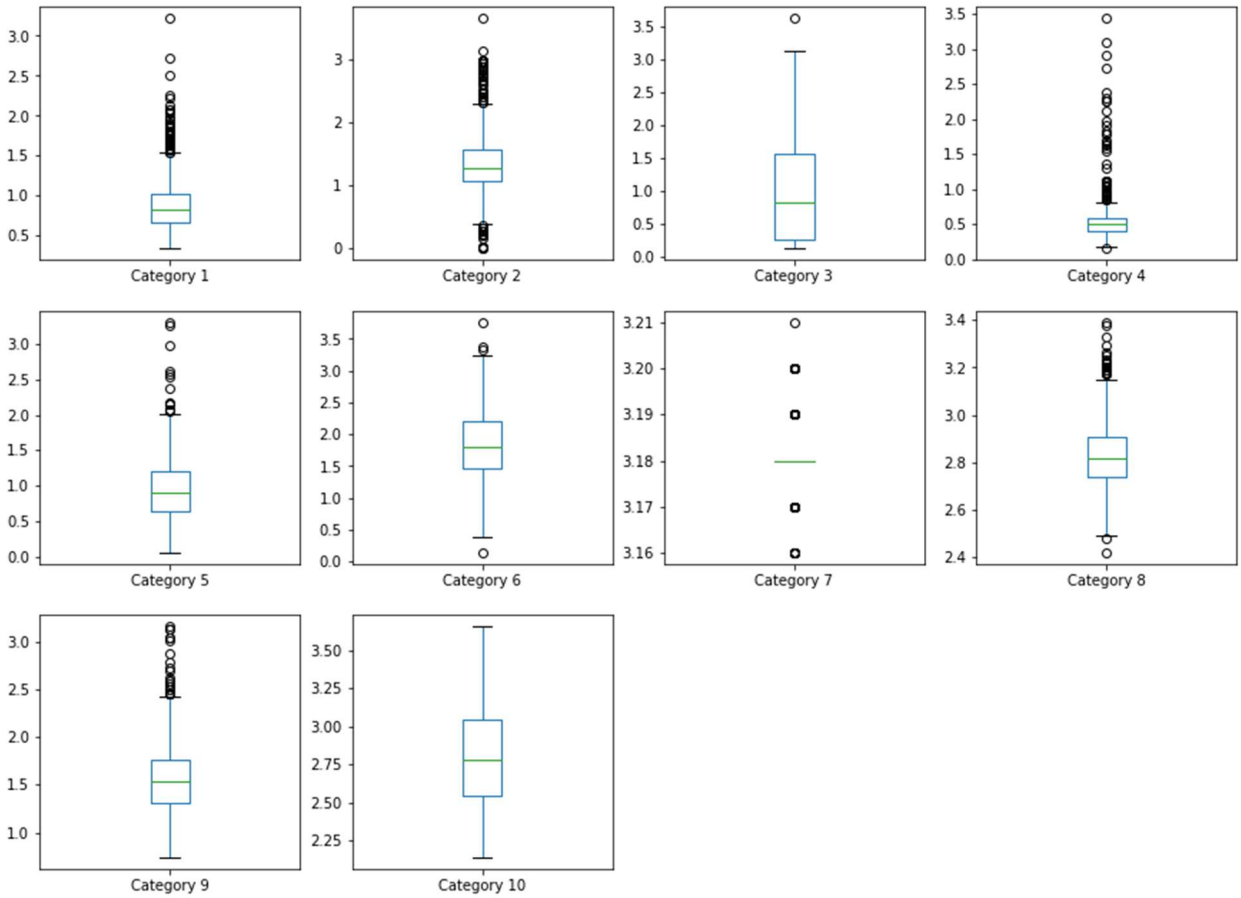

Figure 5. Box plot of data before outlier removal

Looking at the number of outliers in Figure 3, the number is small relative to the number of instances on the dataset. These outliers will be removed as a result. Figure 4 shows the data statistics after these outliers are removed. Looking at Figures 4 and 5, Category 7 seems to have data observations worth looking at, as shown by its standard deviation value. Most of the values are centered very tightly around the mean. The other categories seem to have appreciable amount of variance relative to Category 7.

The data after outlier removal has 900 instances. There are other visualization methods that show the distribution of data observations intra attribute, and provide relationship inter attributes. Figures 6 and 7 display a histogram and density plot of the data. The density plot is similar to the histogram, except it does not show number of observations per bin. Figures 8, 9 and 10 display the relationship of the attributes.

Most of the attributes seem to possess a skewed Gaussian distribution, with tails on one end. The skew measures the asymmetry of the attribute values around the mean. Most of the ML models assume a Gaussian distribution, with no skew. Methods such as Box-Cox transformation or log transformations might improve the skew of each attribute. Category 4 possesses the most skew.

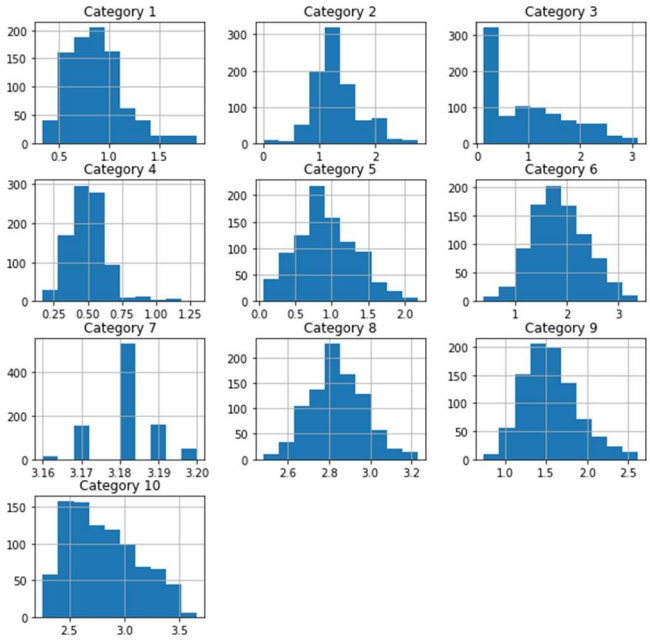

Figure 6. Histogram

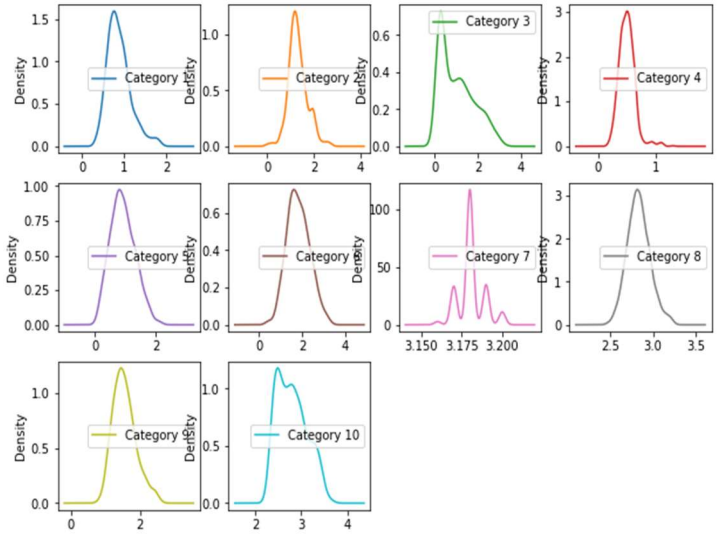

Figure 7. Density plot

Category 3 has a more exponential curve, and Category 7 has its observations very close to the mean. Normalizing these attributes would make them more Gaussian. Table 3 contains information regarding the skew of these attributes.

TABLE III. ATTRIBUTE SKEW

CATEGORY	SKEW
1	0.953425
2	0.458210
3	0.667650
4	1.154586
5	0.337809
6	0.204859
7	0.372823
8	0.297548
9	0.603750
10	0.427020

Categories 5 and 6, 10 and 4 seem to be slightly correlated, whilst Category 7 has good correlation with Categories 3 and 10. The remaining attributes have low correlation with each other.

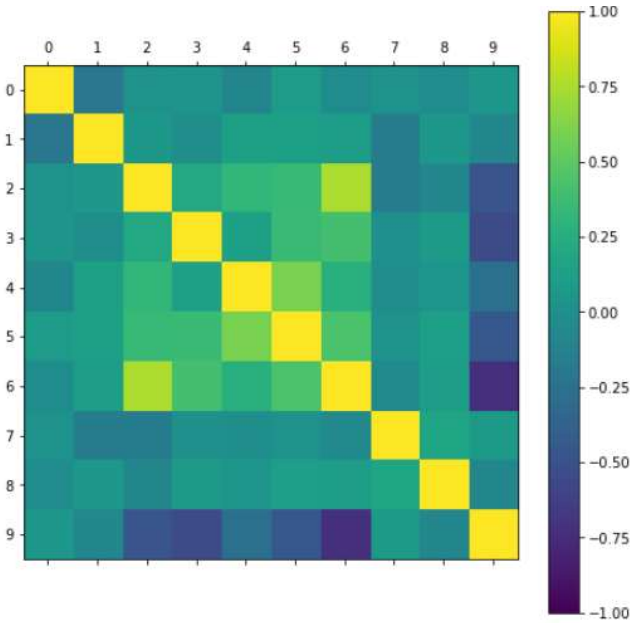

Figure 8. Visual Correlation Matrix

Figure 9 is the scatter matrix of the data. It visually conveys the correlation between attributes as points, as well as a histogram of the attributes. This section reveals some interesting trends of some attributes, like Category 7. Figure 10 reveals what is going in Category 7. This attribute has an interesting data pattern. A lot of users gave ratings between 3.16 and 3.20. This section also shows some of the steps to consider in the feature engineering phase, such as feature transformations and data normalization. In terms initial findings, this shows that users from the remaining attributes, who have very low correlation values, would be grouped together. The remaining sections would seek to show if these initial findings are valid.

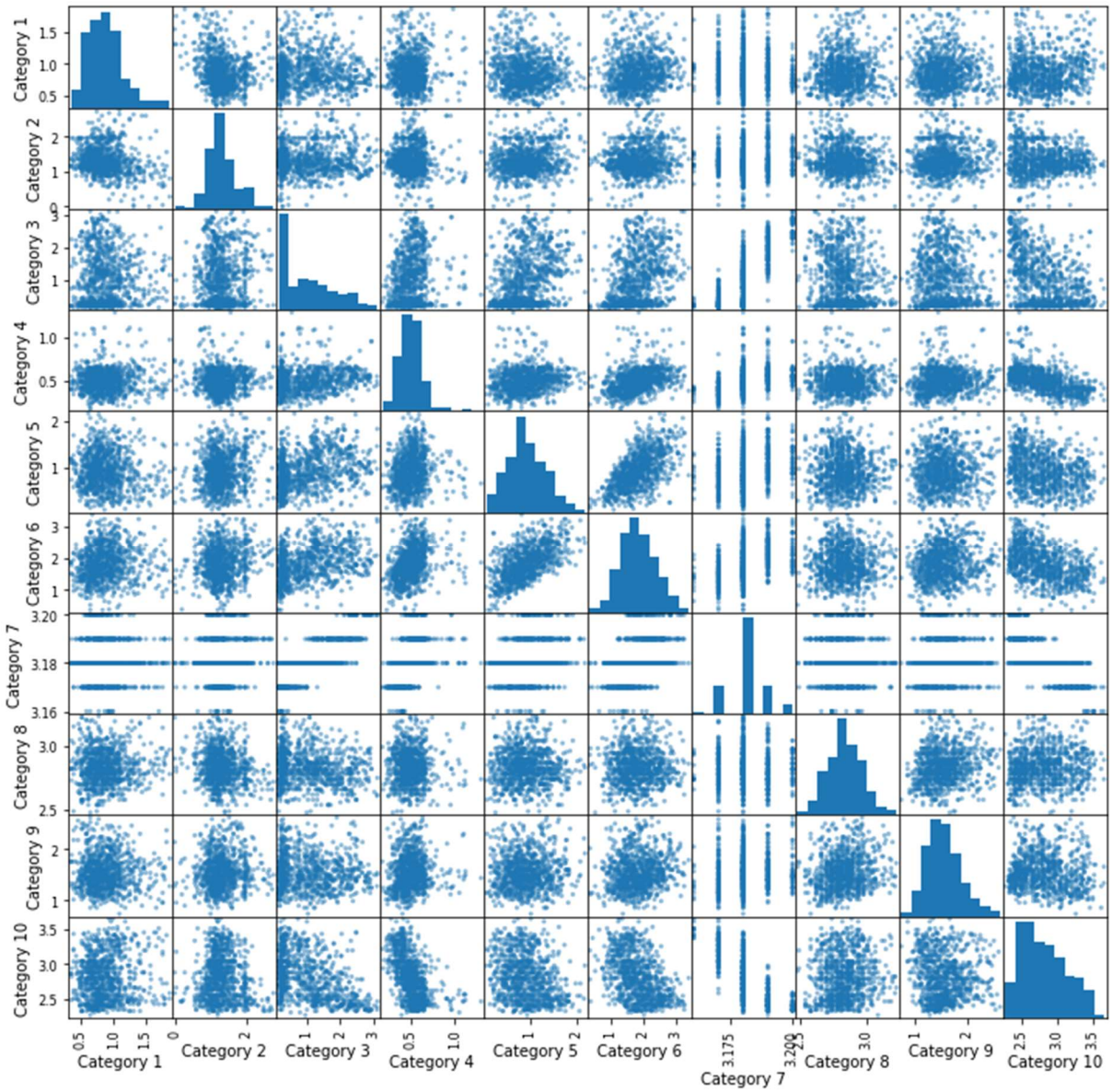

Figure 9. Scatter Matrix

	Category 1	Category 2	Category 3	Category 4	Category 5	Category 6	Category 7	Category 8	Category 9	Category 10
Category 1	1.000000	-0.203714	0.019264	0.031506	-0.079332	0.101675	-0.023726	0.029434	-0.030241	0.065009
Category 2	-0.203714	1.000000	0.062608	-0.000296	0.139896	0.138225	0.119576	-0.160382	0.063659	-0.070633
Category 3	0.019264	0.062608	1.000000	0.205320	0.326714	0.353734	0.755052	-0.162256	-0.075848	-0.470151
Category 4	0.031506	-0.000296	0.205320	1.000000	0.145688	0.361058	0.407246	0.000569	0.092705	-0.533703
Category 5	-0.079332	0.139896	0.326714	0.145688	1.000000	0.596323	0.270148	-0.002176	0.053344	-0.260657
Category 6	0.101675	0.138225	0.353734	0.361058	0.596323	1.000000	0.429973	0.026133	0.131014	-0.447165
Category 7	-0.023726	0.119576	0.755052	0.407246	0.270148	0.429973	1.000000	-0.042212	0.113217	-0.726572
Category 8	0.029434	-0.160382	-0.162256	0.000569	-0.002176	0.026133	-0.042212	1.000000	0.173568	0.093405
Category 9	-0.030241	0.063659	-0.075848	0.092705	0.053344	0.131014	0.113217	0.173568	1.000000	-0.077562
Category 10	0.065009	-0.070633	-0.470151	-0.533703	-0.260657	-0.447165	-0.726572	0.093405	-0.077562	1.000000

Figure 10. Correlation Matrix

D. Feature Engineering/Train Data

Feature Engineering is an important step in completing any Machine Learning (ML) project. This stage ensures that models built for deployment perform very well. Features are essential for making ML models work, hence this step is necessary for making ML models accurate. Building ML models is the main goal for problems that require Machine Learning as solutions. This section displays the results from the various feature engineering steps that were taken in preparing the data for the modeling phase. Outliers were removed from the data before the feature engineering steps took place.

There are various methods to consider when performing Feature Engineering. One of the methods is Feature Selection. There are ten attributes in this dataset. For clustering purposes, all the attributes would be considered. Looking at the data, and the problem being addressed, all the features would be kept. This is because the model will create a cluster of groups, based on similarity measure. Attributes with near zero correlation with other attributes may likely be in its own cluster, dependent on the number of predefined clusters. Such attributes would be described as having no relationship with other attributes, and hence users who give high reviews to such attributes are less likely to give similar ratings to the other non-correlated attributes. If all attributes also have very low correlation, it may mean that the data has no cluster structure.

Another method to consider is Data Transformation (DT). Applying DT has various reasons, such as better organization, when the data observations are not distributed normally, dimensional reduction, etc. Reducing the dimensionality of the dataset makes it easier to visualize model output. PCA would be applied to the data in achieving this. The data would be reduced to two dimensions, Component 1 and 2, for easy visualization. The curse of dimensionality reduction is the fact that there is data loss when a dataset's dimensions are scaled lower. In PCA, the explained variance ratio shows this data loss, in terms of showing the amount of variance that is kept in the modified data. The explained variance ratio is displayed in Table 4.

TABLE IV. PCA VARIANCE RATIO

	Component 1	Component 2
Variance Ratio (%)	48.07	18.13

The two components contain close to 70% of the explained variance. For visualization purposes, this transformed data would be used in the clustering model.

Clustering performance improves when all the data has been transformed to the same scale. [1] considered only Standardization in its data transformation step. For this step, Normalization, Standardization, and a Power Transformation of the data will be performed independently and compare their performances for clustering.

For a Power Transformation, Box-cox will not work here, because the data must be strictly positive. Hence another power transform will be used here, which is the Yeo-Johnson.

Standardization is considered here because the initial data has attributes with dissimilar variances. Such a condition will affect the weights given to some attributes in the clustering process. In this situation leaving variances unequal is equivalent to putting more weight on variables with smaller variance, so clusters will tend to be separated along variables with greater variance.

Normalization (MinMaxScaler) is also considered here to rescale all the attribute values to unit length. The Figures below show the various descriptive statistics for the independent data transformations.

	Component 1	Component 2
count	900.000000	900.000000
mean	0.383838	0.455086
std	0.204119	0.162904
min	0.000000	0.000000
25%	0.213604	0.343122
50%	0.331562	0.454680
75%	0.545662	0.564265
max	1.000000	1.000000

Figure 11 Normalized data Descriptive Statistics.

	Component 1	Component 2
count	9.000000e+02	9.000000e+02
mean	-2.405483e-17	-9.868649e-18
std	1.000556e+00	1.000556e+00
min	-1.881514e+00	-2.795146e+00
25%	-8.344607e-01	-6.876856e-01
50%	-2.562512e-01	-2.493083e-03
75%	7.932346e-01	6.705783e-01
max	3.020324e+00	3.346868e+00

Figure 12. Standardized data Descriptive Statistics

	Component 1	Component 2
count	9.000000e+02	9.000000e+02
mean	1.244683e-16	1.270589e-17
std	1.000556e+00	1.000556e+00
min	-2.428674e+00	-2.836112e+00
25%	-8.352838e-01	-6.845115e-01
50%	-1.127111e-01	4.392812e-03
75%	8.743474e-01	6.741012e-01
max	2.298960e+00	3.290689e+00

Figure 13. Yeo-Johnson Transformation data Descriptive Statistics

In building the model, the various datasets were split into train/test splits, and the train split was used for the optimal k analysis. The test split was used to predict the cluster that the test data would fall under.

IV. Machine Learning Model Development

The goal of this project is to cluster these user groups into similar classes. As stated in Section II, the algorithm that would be, and was employed in this process is the k-means clustering. Partitioning clustering requires to specify the number of clusters in the dataset. There are various methods to determine the optimal number k, such as the Elbow method, Dunn, and Silhouette analysis. After determining the optimal k for clustering, a model was built using that k, and the algorithm.

The k-means algorithm was imported from the sklearn library, which has a number of input arguments, such as n_clusters, random state, init, n_init, etc. N_clusters represent k optimal clusters. N_init represents the number of times the algorithm will run with different centroid seeds. Init is the method for initialization, with kmeans++ selecting initial cluster centers for k-mean clustering in a smart way to speed up convergence. Random state determines random number generation for centroid initialization.

The algorithm was fit on the data, then had the algorithm predict the cluster groups for the test data. The score method is an indication of how far the points are from the centroids. A better score is one that is closer to zero, whereas bad scores are very large negative numbers. [7] The results of this are shown and discussed in the results and discussion sections respectively.

V. Results

The following figures display the results of the steps in Section IV.

A. Optimal k selection

The Elbow method the elbow method is a heuristic used in determining the number of clusters in a data set. The method consists of plotting the explained variation as a function of the number of clusters and picking the elbow of the curve as the number of clusters to use. The following graph shows the elbow graph analysis on each of the data transformation techniques.

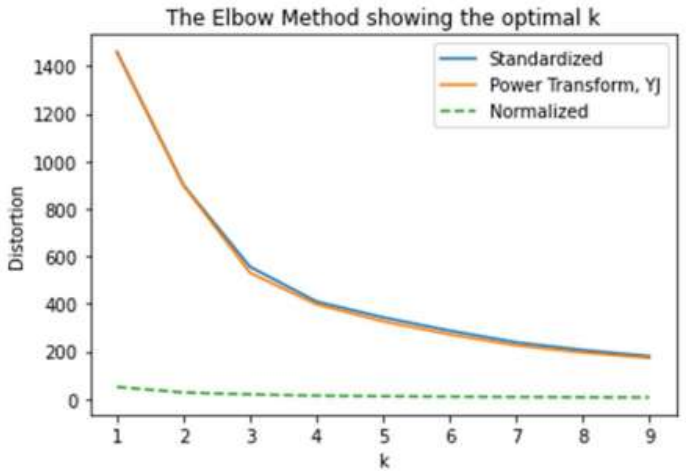

Figure 14. Optimal k selection with Elbow Method

From the Elbow analysis, it is not clear as to what optimal k to choose. There are no defining elbows from each of the data transformed, hence the Silhouette analysis. The Silhouette score measures how similar a point is to its own cluster, as compares to other clusters. The range of the score is between -1 and +1. Scores closer to 1 indicate that the sample is placed in its correct cluster. The best value is 1, and worst is -1. Scores close to 0 indicate overlapping clusters. The following graph displays the Silhouette analysis.

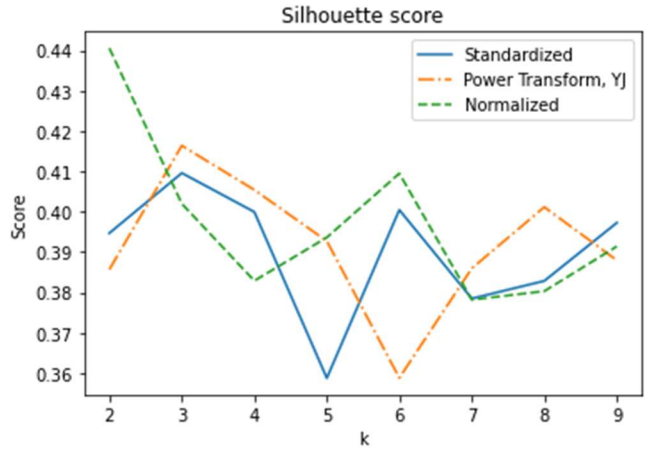

Figure 15. Optimal k selection with Silhouette score

From Figure 15, it can be inferred that the highest score occurs for when k = 2. That occurs with the normalized dataset. Hence the most optimal k for this data is k = 2. The scores from the analysis also show which data transformation contributes to better clustering. Normalized data was chosen because it obtained the highest silhouette score. The Silhouette scores for optimal k selection are displayed in Table 5. Results were also generated for the next best Silhouette score, which happens at k = 3 for the Yeo-Johnson transformed dataset, for comparison. The starred values in Table 5 represents these scores.

TABLE V. OPTIMAL k SELECTION

Score	Standardized	Normalized	Yeo-Johnson
2	0.3947	0.4405 *	0.3857
3	0.4096	0.4020	0.4164 *
4	0.4003	0.3829	0.4054
5	0.3584	0.3915	0.3927
6	0.4014	0.4099	0.3588
7	0.3701	0.3744	0.3859
8	0.3838	0.3833	0.4011
9	0.3960	0.3903	0.3896

B. Model Results

The method "labels" generates the number of transformed data observations under each cluster index. Table 6 shows the number of observations per cluster index. Table 7 contains the cluster centers of each k model. Table 8 displays the k-means score for each k model. Figures 16 and 17 displays the cluster groups generated from the model, with its locally optimal centroids for k = 2 and k = 3 respectively. The axes in the figures represent the cluster index.

TABLE VI. CLUSTER INDEX FREQUENCY FOR k ASSIGNED CLUSTERS

Index	k = 2^1	k = 3^2
0	288	252
1	432	284
2	N/A	184

TABLE VII. CLUSTER CENTERS

k = 2	k = 3
[0.59994058, 0.47710454] [0.23532361, 0.44562154]	[0.3779811,1.00815098] [-0.97965688, 0.30348866] [0.9311491, 0.98727642]

TABLE VIII. k-means SCORE METRIC

	k = 2	k = 3
Score	-26.5685	-529.9822

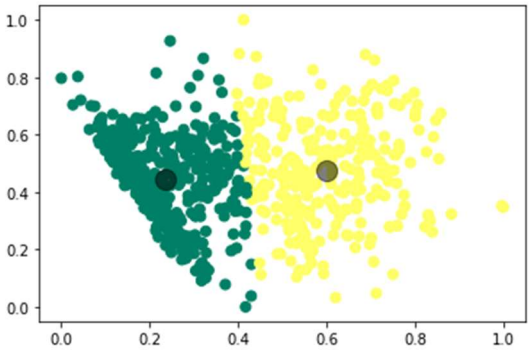

Figure 16. k = 2 cluster visualization

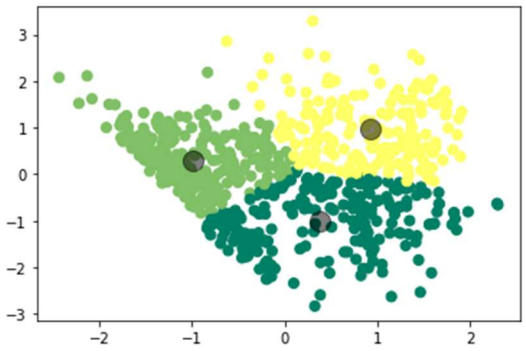

Figure 17. k = 3 cluster visualization

VI. DISCUSSIONS

Unlike classification and regression models, clustering models have no definite way of checking for performance and accuracy, such as precision and recall, accuracy etc. These types of performance metrics are for supervised learning algorithms, where there is an output value that can be measured against. From revised literature, the quality of a cluster is judged by the Silhouette score, which shows how well the data is clustered for the optimal k.

There is also the k-means Score metric, which is the value that represents the sum of squares of the distances of points from their cluster centroids. As stated earlier in Section IV, the higher the absolute score method, the worse the clustering performance on the data. From Table 5, the best score was 0.44, for k = 2.

Looking critically at the Silhouette scores and the score method values, it cannot be said that clustering is a bad option for recommender systems. This is because this dataset had very low correlation between attributes to begin with. In applying dimensionality reduction, however, much of the data is lost, and that may contribute to the values of these metrics. Nevertheless, the scores give an indication of the optimal k to choose for model development for this dataset. Table 5 and 8 gives results for the optimal k value. Part of both clusters in Figure 16 were split off to form a new cluster index in Figure 17.

Subscript 1 represents Normalized dataset, and 2 represents Yeo-Johnson dataset.

Another observation made was the fact that after outlier removal, a box plot on the new data still showed outliers. A valid reason could be that after such removal, one gets a new distribution of your attributes with a new five number summary. With these new values, other values can be outliers that were not outliers with the old values of the original attributes. Figure 18 displays the box plot of the new data.

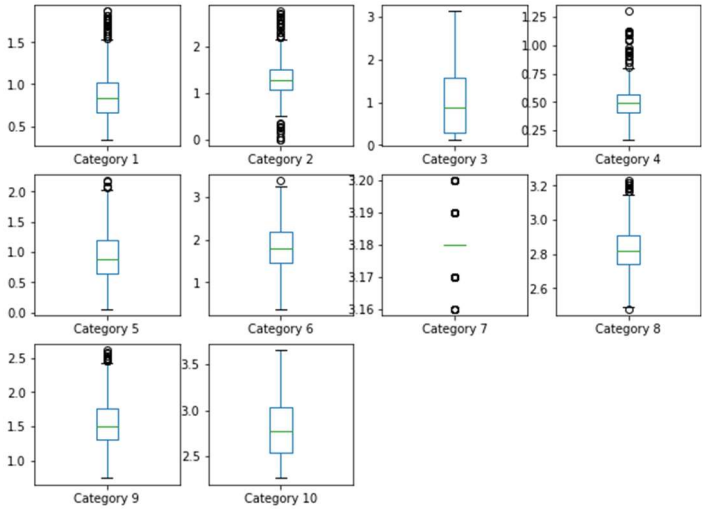

Figure 18. Box plot of new data.

Although clustering groups can be predicted using the test dataset, there are no ground truth labels for performance metrics such as accuracy. If the transformed valued from k-means can be classified using a classification algorithm, and using the cluster index as an output measure, then such performance metrics can be used here. Clustering is a good case for recommender systems, provided dataset attributes are more diverse, which increases likelihood of more defined relationships between attributes.

VII. CONCLUSIONS

Clustering is the process of populating similar items into clusters with high intra-cluster and low inter cluster similarity. In this paper, a clustering algorithm for a recommender system was evaluated using a social media dataset in the tourism domain. From the results gathered during this analysis, k-means algorithm performed averagely, based on how far away its silhouette score was from the desired value of +1, as elaborated in Section V.

Clustering can help to propose most relevant solutions to customers based on their profiles. Any information that reflects customer traits can become an input to clustering process. In this work, we considered user reviews, feedbacks, and rating information captured from forums and social media. However, travel managers have diverse opportunities to capture user traits and interests by tracking the types of queries coming to them, taking direct feedback via questionnaires or surveys, keeping track of the user transactions and monitoring the reviews on travel forums and portals [1]. Clustering with dimensionality reduction contributes to visualization of recommender systems, with the tradeoff being data loss.

VIII. REFERENCES

[1] Renjith, Shini, A. Sreekumar, and M. Jathavedan. 2018. Evaluation of Partitioning Clustering Algorithms for Processing Social Media Data in Tourism Domain. In 2018 IEEE Recent Advances in Intelligent Computational Systems (RAICS), 12731. IEEE.

[2] Renjith, Shini, and C. Anjali. "A personalized mobile travel recommender system using hybrid algorithm." In Computational Systems and Communications (ICCSC), 2014 First International Conference on, pp. 12-17. IEEE, 2014.

[3] Jiang, Shuhui, Xueming Qian, Tao Mei, and Yun Fu. "Personalized travel sequence recommendation on multi-source big social media." IEEE Transactions on Big Data 2, no. 1 (2016): 43-56.

[4] Renjith, Shini, and C. Anjali. "A personalized travel recommender model based on content-based prediction and collaborative recommendation." International Journal of Computer Science and Mobile Computing, ICMIC13 (2013): 66-73.

[5] Ricci F., Rokach L., Shapira B. (2011) Introduction to Recommender Systems Handbook. In: Ricci F., Rokach L., Shapira B., Kantor P. (eds) Recommender Systems Handbook. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-85820-3_1.

[6] Heckmann, D., Schwartz, T., Brandherm, B., Schmitz, M., von Wilamowitz-Moellendorff, M.: Gumo - the general user model ontology. In: User Modeling 2005, 10th International Conference, UM 2005, Edinburgh, Scotland, UK, July 24-29, 2005, Proceedings, pp. 428–432 (2005).

[7] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.