Eric Yarger

D212 Task 1

Clustering Techniques

A1: Proposal of Question

Identifying clusters from our organization's dataset is a project that has the potential to provide insights that are both profitable for our organization and increase patient care quality. A Hamilton Project study, analyzing measures collected in the National Health and Nutrition Examination Survey conducted by the CDC, has shown that individual income affects obesity rates, stress load, and self-reported health levels (Schanzenbach, D., Mumford, M., Nunn, R., & Bauer, L., 2016).

Our team of analysts has been asked to conduct an analysis using an unsupervised clustering algorithm that has the potential to yield valuable business insights into our dataset. The question proposed for this analysis is:

"What is the optimal number of clusters, using K-means clustering algorithm, for variables 'Income' and 'Additional_charges'?"

A2: Defined Goal

The goal of this analysis is to provide our organization with insight into how Income, a multifaceted demographic feature, clusters with feature Additional_charges. Both features are found in our organization's medical_clean dataset. In this dataset, Additional_charges represents the average amount charged to patients for miscellaneous procedures, medicines, treatments, etc. Income represents annual income of the patient or primary insurance holder, as reported at the time of admission.

B1: Explanation of Clustering Technique

K-means is an unsupervised clustering algorithm. It's used to partition unlabeled data into a specific number ('k' number of) unique groups. Put another way, k-means finds features that share characteristics and classifies them together, forming clusters (Jeffares, 2019).

K-means works by choosing a value for k (the # of clusters) and choosing an initial center coordinate (centroid) for each cluster. Then each case is assigned to the nearest cluster, chosen by how close it resides to the centroid. The centroids are updated and calibrated as this process iterates. These steps iterate until all observations are assigned to a cluster and there is no change in the centroid for all clusters. The algorithm then can be said to have converged, with the end result being the final clusters.

To build on this explanation, the advantages and disadvantages of k-means, according to Google's Machine Learning Education Platform (K-means advantages and disadvantages, n.d.), are laid out below.

- Advantages
    - Simple to implement.
    - Scales to large data sets.
    - Guarantees convergence
    - Easily adapts to new examples in the data.
    - Generalizes to different sized and shaped clusters.
- Disadvantages
    - K-means the number of k is chosen manually
    - Results are dependent on initial values
    - Has trouble clustering data of varying size and density.
    - Outliers can have a large effect on the centroids of clusters
    - As the number of dimensions increases, k-means becomes less effective at distinguishing between examples.

Care must be taken to ensure that k-means converges to the global, not local minima. Good practice is to run k-means multiple times, each time with different randomized initial centroids selected. The results from this process with the best solution are selected for the end k-means analysis. This helps avoid the possibility that the results converged on a local minima.

The expected outcome of k-means clustering for this analysis is that K-means clustering will return a small 'k' of clusters, with the expected number 'k' being 3. The silhouette score will be calculated for the K-Means clustering results. This measures the accuracy of the K-Means clustering. It is expected that the silhouette score will be positive, and will be high enough to show that the clusters exhibit meaningful separation between the clusters, and meaningful distance between points in each cluster.

B2: Summary of Technique Assumption

Key assumptions of k-means is discussed in a Datacamp video hosted by Karolis Urbonas, Head of Data Science at Amazon (Data pre-processing | Python, n.d.). From this resource, the key assumptions of k-means are:

- Symmetric distribution of variables.
  - This means the distribution is not skewed, all variables are to the same mean.
- All variables have the same average values.
  - This ensures every measure gets the same weight in the k-means calculation.
- The variance of each variable is scaled to the same level.
  - This helps the algorithm to converge correctly and assigns equal variance to each variable.

B3: Packages or Libraries List

Tool	Description	How it was used in this analysis
matplotlib.pyplot	Free comprehensive library for creating visualizations in Python.	Useful for visualizing histograms and the shape of variable data during data cleaning.
pandas	Powerful open source data analysis tool.	Used for creating two-dimensional data frames during the data preparation stage.
numpy	Allows Python to process larger arrays than it could normally handle.	Used for visualizing data in the cleaning and preparation stages.
scipy	Open source Python library for statistical analysis and scientific computing.	Statistical analysis functions used for outlier removal during data preparation.

platform	Function used to identify platform and environment specs.	Used to print Python's version in jupyter notebook.
Sklearn.cluster's Kmeans	Tool used to perform K-means clustering algorithm.	Used to cluster the dataset into 'k' number of pre-specified clusters.
Sklearn.metrics silhouette_score	Metric used to evaluate the quality of clusters created by K-Means.	Used to evaluate K-Means clustering results for 'i' # of 'k' in our analysis.
Yellowbrick.cluster's SilhouetteVisualizer	Displays the silhouette coefficient for each sample from analysis results.	Used to visualize and evaluate the results of K-Means clustering.

C1: Data Preprocessing

The data needs to be preprocessed before our selected method, k-means clustering, will return viable results. The data preprocessing goal for this analysis is to begin our clustering with a dataset that is optimized for k-means (Ryzhkov, 2020). These steps are:

- Step P1: Data Cleaning.
    - This involves: duplicate removal, removing unnecessary columns, handling inconsistent data, and removal of irrelevant observations and errors.
- Step P2: Handling missing data.
    - This involves: imputation of missing data by feature analysis. Replacing missing values with the best option, through mean, median, or mode.
- Step P3: Data Transformation
    - Scaling and standardizing features so that the dataset is optimized for k-means clustering.
    - Square-Root Transformation on dataset to mitigate feature skewness on K-Means analysis results.

This meets our preprocessing goal of beginning our clustering with a dataset optimized for k-means.

C2: Dataset Variables

This Section identifies the variables used in this analysis, and if it is continuous or categorical.

Variable	Continuous or Categorical
Income	Continuous
Additional_charges	Continuous

C3: Steps for Analysis

The table below explains each step used to prepare the data for analysis. The 'Code Segment' column identifies the section in the accompanying jupyter notebook that contains the code for that step.

Step	Code Segment	Explanation
1	P1	Data Cleaning. Check for duplicates and missing values. Remove unnecessary columns. Handle outliers. Save cleaned data to variable 'dfs'.
2	P2	Handling Missing Data. Check for and handle any missing data in our set. Remove/Impute if necessary.
3	P3	Data Transformation. Data Scaling/Normalization using Min/Max Scaling technique in preparation for k-means clustering. This scales our features so that k-means returns relevant results. Square-Root Transformation to mitigate skewness in features. This helps ensure K-Means algorithm has an optimal data set to cluster.

C4: Cleaned Dataset

An Excel file of the cleaned dataset is included with this submission.

D1: Output and Intermediate Calculations

This section will describe the analysis technique used to analyze the data, K-Means, and will include all screenshots of intermediate calculations.

The cleaned dataset provided in section C4 is used for the analysis. KMeans is imported from scikit's sklearn.cluster. KMeans is looped for k = i, with i being 1-11. KMeans clustering result is fitted to our dataset. The Elbow Method graph is plotted. Provided below is a screenshot of the analysis to this point.

K-means clustering

[275]:
```python
# Prepared dataset cleaned_data
# Code reference (Sklearn.cluster.KMeans, n.d.)
import sklearn
```

[276]:
```python
from sklearn.cluster import KMeans
```

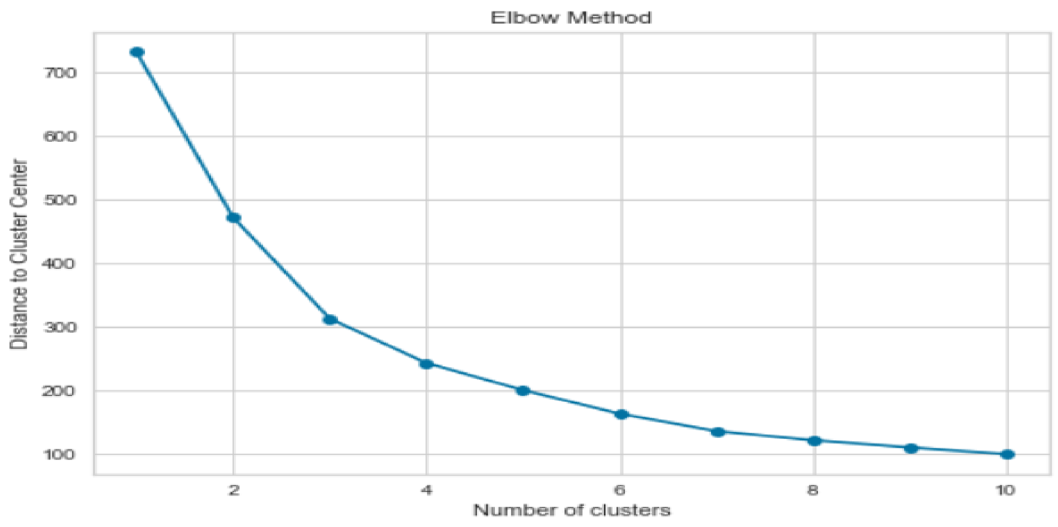
[277]:
```python
X
```

[277]:

	Income	Additional_charges
0	0.828765	0.734745
1	0.608912	0.726606
2	0.336131	0.723897
3	0.560916	0.599672
4	0.091589	0.146735
...	...	...
9995	0.603415	0.459824
9996	0.343301	0.961752
9997	0.722957	0.665567
9998	0.484602	0.411918
9999	0.704951	0.557135

9857 rows × 2 columns

[278]:
```python
# Initial KMeans.  run for 1-11 clusters.
# Elbow Plot
#Code Reference (Machine Learning - K-means, n.d.)
# Sum of square distance from cluster center
ssdistance = []


# Kmeans from 1 to 11 loop
for i in range(1,11):
    kmeans = KMeans(n_clusters=i)
    kmeans.fit(X)
    ssdistance.append(kmeans.inertia_)

plt.plot(range(1,11), ssdistance, marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('Distance to Cluster Center')
plt.show()
```



From the graph above, the potential optimal # of clusters from the elbow graph appears to be 2 or 3. Next, silhouette scores are calculated for k= 2,3,4,5, and 10. Silhouette_score is imported from sklearn.metrics. The silhouette score is a method for finding the optimal number of clusters, and returns a number between -1 and 1. Further information regarding how silhouette scores are calculated can be found in section E1 of this document. The highest silhouette score was for k=3, at .352. This means that the optimal number of clusters for our dataset is 3. Provided below is a screenshot of the silhouette score being calculated for the stated values for ‘k’.

79]:
```
from sklearn.metrics import silhouette_score
# Code Reference (Arvai, 2020)
```

80]:
```
# KMeans clusters = 2
kmeans2 = KMeans(n_clusters = 2, init='k-means++', random_state=10)

# Fit the model
kmeans2.fit(X)

# Silhouette Score
kmeans2_silhouette = silhouette_score(X, kmeans2.labels_)

# Print Silhouette Score
print('Silhouette Score: %.3f' %kmeans2_silhouette)
```
```
Silhouette Score: 0.320
```

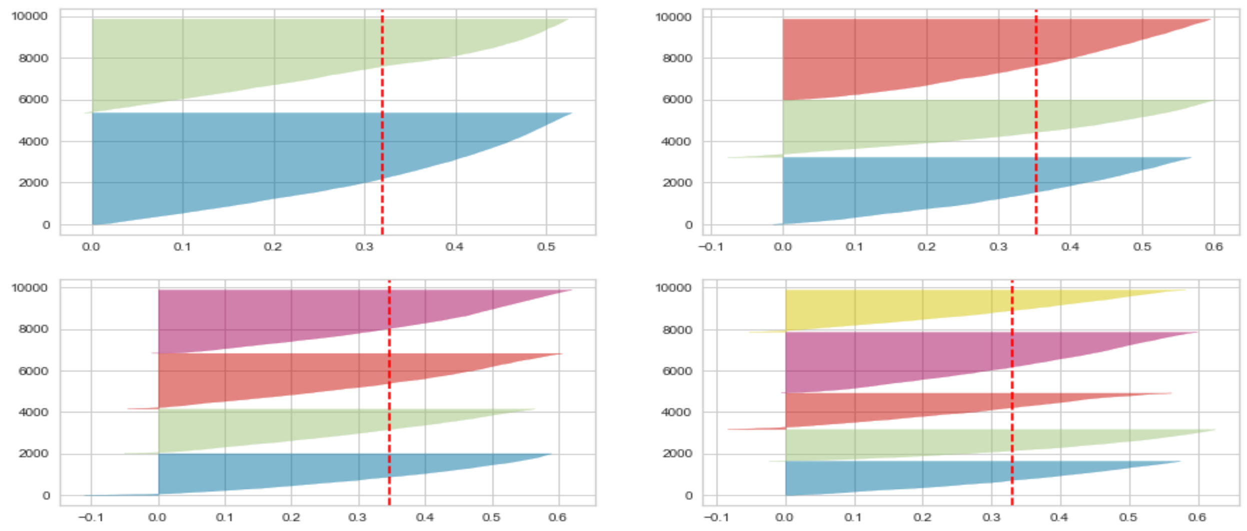
81]:
```
# KMeans clusters = 3
kmeans3 = KMeans(n_clusters = 3, init='k-means++', random_state=10)

# Fit the model
kmeans3.fit(X)

# Silhouette Score
kmeans3_silhouette = silhouette_score(X, kmeans3.labels_)

# Print Silhouette Score
print('Silhouette Score: %.3f' %kmeans3_silhouette)
```
```
Silhouette Score: 0.352
```

82]:
```
# KMeans clusters = 4
kmeans4 = KMeans(n_clusters = 4, init='k-means++', random_state=10)

# Fit the model
kmeans4.fit(X)

# Silhouette Score
kmeans4_silhouette = silhouette_score(X, kmeans4.labels_)

# Print Silhouette Score
print('Silhouette Score: %.3f' %kmeans4_silhouette)
```
```
Silhouette Score: 0.347
```

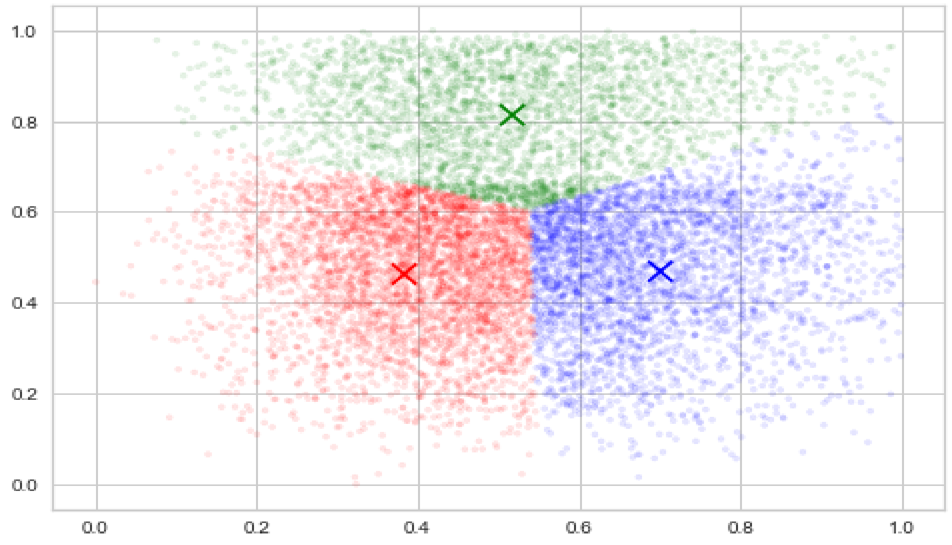
83]:
```
# KMeans clusters = 5
kmeans5 = KMeans(n_clusters = 5, init='k-means++', random_state=10)

# Fit the model
kmeans5.fit(X)

# Silhouette Score
kmeans5_silhouette = silhouette_score(X, kmeans5.labels_)

# Print Silhouette Score
print('Silhouette Score: %.3f' %kmeans5_silhouette)
```
```
Silhouette Score: 0.332
```

84]:
```
# KMeans clusters = 10
kmeans10 = KMeans(n_clusters = 10, init='k-means++', random_state=10)

# Fit the model
kmeans10.fit(X)

# Silhouette Score
kmeans10_silhouette = silhouette_score(X, kmeans10.labels_)

# Print Silhouette Score
print('Silhouette Score: %.3f' %kmeans10_silhouette)
```
```
Silhouette Score: 0.332
```

Next, SilhouetteVisualizer is imported from yellowbrick.cluster. SilhouetteVisualizer visualizes KMeans instances, providing an accessible and informative tool for assessing cluster size and quality in each instance. From the screenshot below, starting clockwise in the top left corner, these are visuals for Kmeans instances where k = 2,3,4,5. The results of this step are discussed in detail in section E2. Provided below is a screenshot of the resulting output.



Next the clusters are visualized on a 2-D graph as a color-separated scatterplot. Each The centroid of each cluster is marked with an 'X'. This helps visualize the cluster density and separation. Provided is a screenshot of the resulting output.



D2: Code Execution

Included with the submission is a PDF of the JupyterNotebook containing all code used in the analysis, with outputs. The kernel is restarted and all code rerun in the Panopto accompanying this submission to show the code runs free of errors.

E1: Accuracy of Clustering Technique

Accuracy is the measure of comparing the true label of a datapoint to the predicted label. Where K-Means is an unsupervised clustering algorithm, and no predicted labels exist, the ‘accuracy’ comparison of the results in this sense technically doesn’t exist. There are metrics that allow for the evaluation of clusters, though. Variance minimization, such as Within Cluster Sum of Squares (WCSS) and Between Clusters Sum of Squares (BCSS) are metrics that do this (Accuracy for K-Means clustering, n.d.).

WCSS finds the Euclidean distance between a data point and the cluster's centroid. This is done for every point in the cluster. The values are summed and divided by the total number of points. Next, the average is calculated across all clusters. This measurement indicates the variability of points **within** a cluster. The larger the sum of squares, the more variability there is.

BCSS measures Euclidean distance and squared average distance between all cluster centroids. It is performed in the same manner as WCSS but **between** clusters, not within a cluster. BCSS indicates the variation between all clusters. The larger the number, the more spread out the clusters are. A smaller BCSS indicates the clusters are closer together. It measures the separation between clusters.

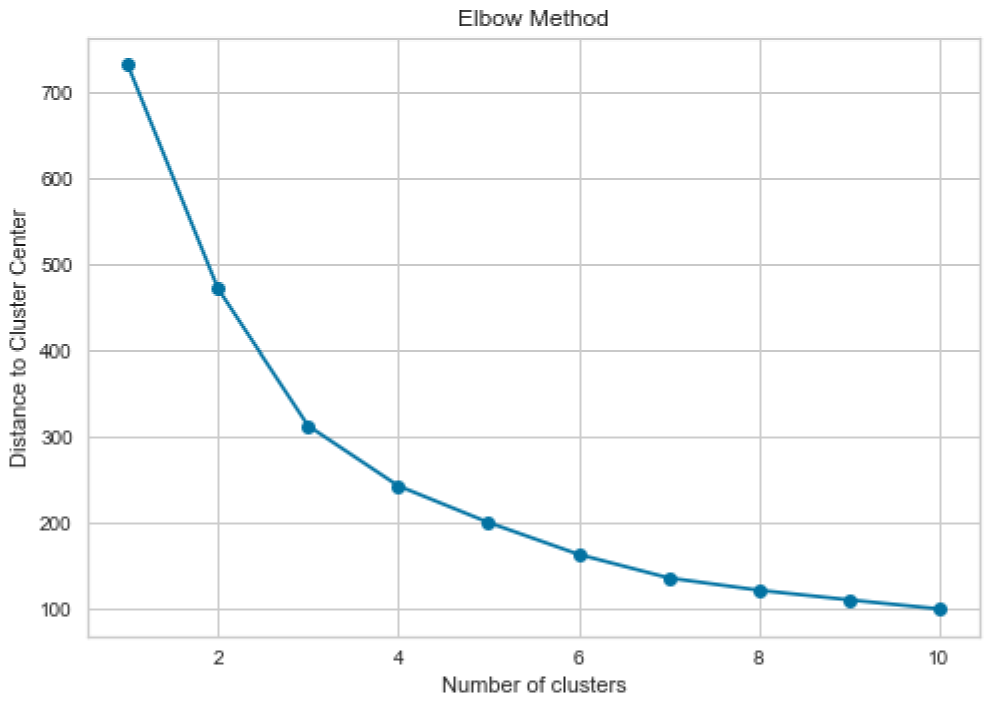
In this analysis, accuracy of the K-Means clustering algorithm is assessed by first plotting an elbow graph to initially visualize data cluster distribution by # ‘k’ clusters. Next sklearn’s Silhouette Score is calculated for multiple values ‘k’. The silhouette score combines the WCSS and BCSS. For this analysis the silhouette score was calculated for ‘k’ values 2, 3, 4, 5, and 10 ‘k’ clusters.

The silhouette score is a method for finding the optimal number of clusters, as well as interpreting the data consistency within the clusters. Silhouette score values range from values of -1 to 1. A higher value indicates that the point is well matched to the cluster it is assigned to, and ill matched to neighboring clusters. A negative silhouette score would indicate that the point is incorrectly clustered. The results of these metrics are discussed in the next section.

E2: Results and Implications

The results are visualized and discussed below. The results are given for the Elbow Graph, Silhouette Scores at different cluster numbers ‘k’, the clusters and their marked centroid visualized on a scatter plot, and visualized using Yellowbrick’s Silhouette Visualizer. Tables and screenshots of the results are provided below.

Below is a screenshot of the Elbow Method Graph.



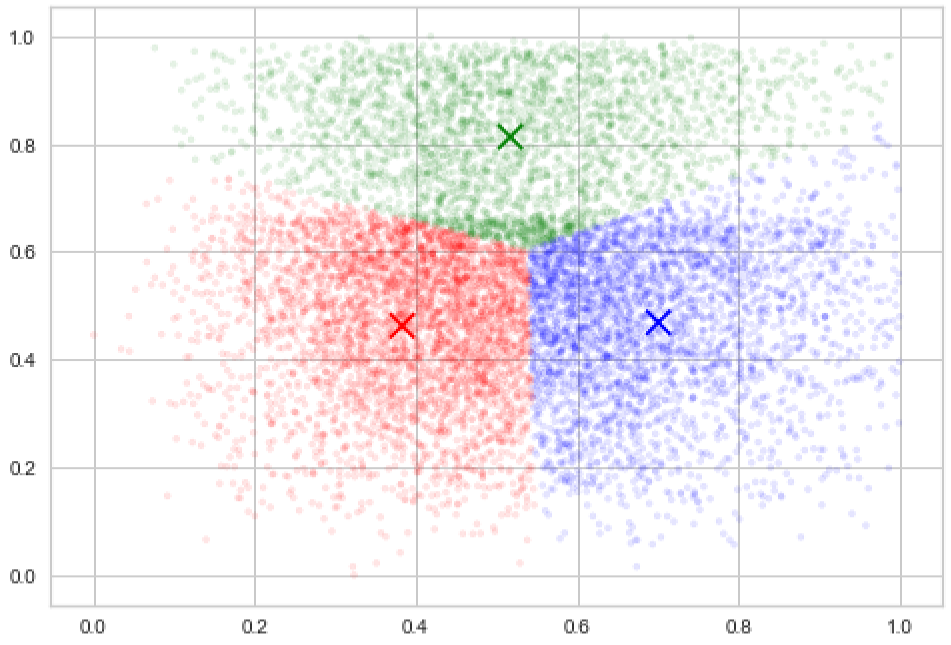
This graph is useful for determining the number of clusters in a data set. The ‘elbow’ of the curve, in our analysis k = 3, is chosen as the number of clusters to use.

Below is a table with the assigned ‘k’ number of clusters, and the resulting silhouette score.

Number of Clusters ‘k’	Silhouette Score
2	.320
3	.352
4	.347
5	.332
10	.332

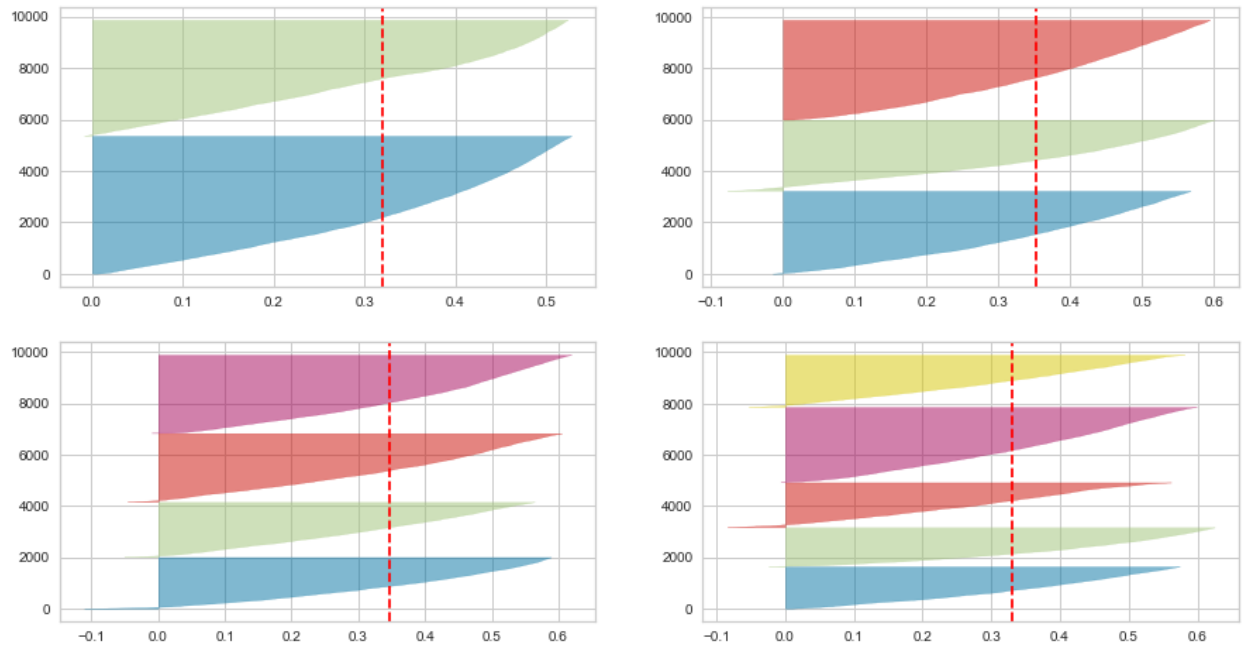
There is a bit of ambiguity baked into the interpretation of the Elbow Method graph. To remove this ambiguity K-Means clustering was ran with the ‘k’ number of clusters shown in the table above. The resulting silhouette score for each ‘k’ value is provided. The silhouette score is used to definitively choose k = 3. This is because the ‘k’ value clusters with the highest Silhouette Score was for 3 clusters, at .352.

Below is a screenshot of the scatter plot of the data set clusters.



The green, blue, and red ‘X’ marks each cluster’s centroid. This scatter plot is useful for visualizing the distance between clusters, and the distance of points within each cluster.

Provided below is a screenshot of the resulting visualization of the clusters using Yellowbrick’s Silhouette Visualizer.



To detail, the top left graph is for k=2, the top right is k=3, bottom left is k=4, bottom right is k=5. This visualization displays the silhouette coefficient for each K-Means clustering on a per-cluster basis. This helps analysts to visualize the density of clusters. It’s also useful for visualizing cluster imbalance.

From the above results of the clustering analysis, the overall result that answers the question of this analysis is that the optimal number of clusters is 3. This is evidenced by the following results. The silhouette score of k = 3 is .352. This metric takes into consideration the BCSS and WCSS discussed in section E1. The scatter plot visualizes that the clusters are not well separated. The Silhouette Visualizer visualization for k = 3 indicates that the clusters are slightly imbalanced and have similar but differing average densities.

The overall implication of the clustering analysis is that features ‘Income’ and ‘Additional_charges’ are optimally grouped into 3 clusters, given the data points in our organization’s data set. The cluster shapes, cluster separation distance, and cluster imbalance information can be used by our analyst team in further analysis.

E3: Limitation

One limitation of this analysis is that the data is not well separated. In a perfect world the data would exhibit separation like the image below:

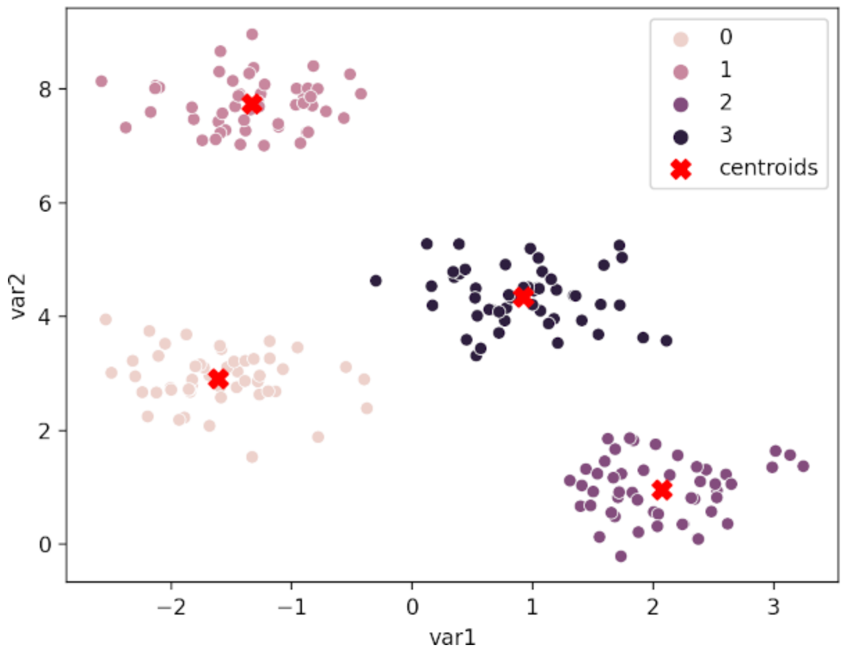
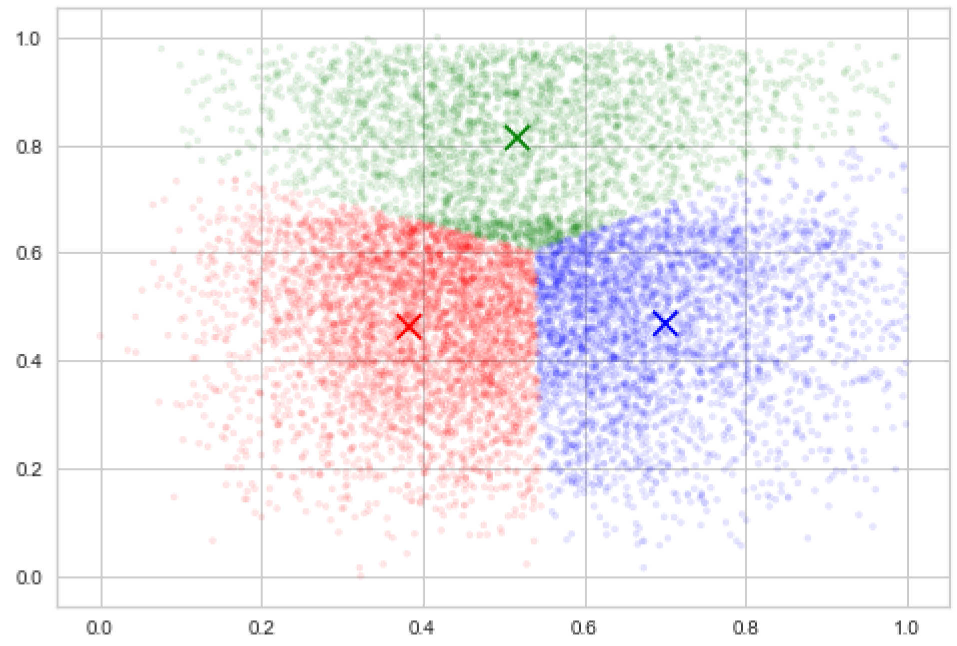


Image Source: (Bedre, n.d.)

Take this in comparison to our data set:



Our data does not exhibit this level of separation. This can make finding the optimal number of clusters difficult. It can also make the resulting clusters less 'accurate', as discussed using accuracy-like clustering metrics in section E1. This can affect the utility of our results for use in our organization.

E4: Course of Action

The recommended course of action is two-fold. First, our organization can use the results of this analysis as a starting point for further cluster analysis if the need is warranted. With a Silhouette Score of .352, the results of the analysis exhibit positive, but not very strong, clustering traits that can potentially be used in business cases in our organization.

Second, if our executive team decides to continue research into this question, I recommend investigating other clustering algorithms. K-Means clustering is a centroid-based clustering algorithm: it's possible that a density-based, distribution based, or hierarchical clustering algorithm would be more appropriate for clustering this specific data set. These two separate courses of action, to keep what we have, or to try a different method, can be more fully planned out once our executive team decides which direction they'd like to go with this.

G: Sources for Third-Party Code

Data normalization with pandas. (2020, December 11). Retrieved from https://www.geeksforgeeks.org/data-normalization-with-pandas/

Python | Pandas dataframe.skew(). (2022, July 15). Retrieved from https://www.geeksforgeeks.org/python-pandas-dataframe-skew/

Sklearn.cluster.KMeans. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

Machine learning - K-means. (n.d.). Retrieved from https://www.w3schools.com/python/python_ml_k-means.asp

Arvai, Kevin. (2020, July 20). K-means clustering in Python: A practical guide. Retrieved from https://realpython.com/k-means-clustering-python/

Kumar, Ajitesh. (2020, Sep. 17). KMeans silhouette score with Python examples - DZone AI. Retrieved from https://dzone.com/articles/kmeans-silhouette-score-explained-with-python-exam

Yellowbrick :: Anaconda.org. (n.d.). Retrieved from https://anaconda.org/DistrictDataLabs/yellowbrick

How to plot KMeans? (n.d.). Retrieved from https://stackoverflow.com/questions/61177418/how-to-plot-kmeans

H: Sources

Schanzenbach, D., Mumford, M., Nunn, R., & Bauer, L. (2016, Dec 12). Money lightens the load. Retrieved from https://www.hamiltonproject.org/papers/money_lightens_the_load

Ryzhkov, E. (2020, July 23). 5 stages of data Preprocessing for K-means clustering. Retrieved from https://medium.com/@evgen.ryzhkov/5-stages-of-data-preprocessing-for-k-means-clustering-b755426f9932

Jeffares, A. (2019, November 19). K-means: A complete introduction. Retrieved from https://towardsdatascience.com/k-means-a-complete-introduction-1702af9cd8c

K-means advantages and disadvantages. (n.d.). Retrieved from https://developers.google.com/machine-learning/clustering/algorithm/advantages-disadvantages

Data pre-processing | Python. (n.d.). Retrieved from https://campus.datacamp.com/courses/customer-segmentation-in-python/data-pre-processing-for-clustering?ex=1

Accuracy for K-Means clustering. (n.d.). Retrieved from https://datascience.stackexchange.com/questions/41317/accuracy-for-kmeans-clustering

Bedre, Renesh. (n.d.). *K-means clustering plot with centroid.* Reneshbedre.com. Retrieved from https://www.reneshbedre.com/blog/kmeans-clustering-python.html