**Part I: Research Question**

A.  Describe the purpose of this data mining report by doing the following:

1. Can a K-means clustering model successfully group patients into two groups, patients that were readmitted and patients that weren’t, using the available numeric variables in the dataset?
2. One goal of this analysis is to determine whether available data can be used to predict a patient’s readmittance status. If the model performs well, we expect a confusion matrix comparing labels of the test data to predictions of the test predictors to show high values on the diagonal, which represent correct predictions, and values close to zero in the corners, which represent incorrect predictions.

**Part II: Technique Justification**

B.  Explain the reasons for your chosen clustering technique from part A1 by doing the following:

1. K means clustering identifies K clusters, K being a number that is definable by the user, by guessing K random points as the centers of the clusters and calculating the distance from each data point to the nearest cluster center and calculating the mean of each of the clusters. The algorithm repeats this process until each cluster is approximately the same size, with approximately the same mean.
2. K means clustering assumes that clusters are spatially grouped, meaning that the clusters are roughly circular, or spherical, and are not overlapping or nested within one another (Tait, 2017). This can lead to problems when clusters are not spherical in nature, for instance, if the center of a circle is one cluster and the other cluster is everything outside of the boundary of the first cluster, K means clustering will not be able to differentiate between these clusters.
3. We will be using Python in Jupyter-Notebook for this analysis with the addition of the Pandas library for data organization and sorting, Numpy for aggregate calculations and the multi-slicing method \_r, Seaborn for creating heatmaps, and Sklearn for preprocessing, the clustering algorithm, and model evaluation.

**Part III: Data Preparation**

C.  Perform data preparation for the chosen dataset by doing the following

1. To prevent the KMeans algorithm to see each predictor variable as equally important, the data needs to be normalized, meaning numeric variables are scaled from zero to one. Because the KMeans algorithm is designed to work with only numeric data, every predictor variable will be normalized through the Sklearn MinMaxScaler method.
2. KMeans is designed to work with numeric data, not categorical. Sklearn has KMeans-like algorithms that work with mixed or categorical data, but this analysis focuses on the KMeans algorithm. The numeric predictor variables we use for this analysis are Lat (Continuous), Lng (Continuous), Population (Discrete), Children (Discrete), Age (Discrete), VitD\_levels (Continuous), Doc\_visits (Discrete), Full\_meals\_eaten (Discrete), vitD\_supp (Discrete), Initial\_days (Discrete), TotalCharge (Continuous), and Additional\_charges (Continuous). ReAdmis will be used as the target variable and is the only categorical variable used in the analysis.
3. First, the data is loaded into a pandas data frame and the desired predictor variables selected and placed into a new pandas dataframe.

data = pd.read\_csv('medical\_clean.csv')

data\_sel = data.iloc[:, np.r\_[8:11, 14:16, 19:24, 39:42]]

Next, we identify any predictor variables that are multicollinear and remove them from the dataset with the help of a heatmap constructed from a correlation matrix of the predictor variables in the dataframe.

sns.heatmap(data\_sel.corr(), linewidths=1,

cmap=sns.color\_palette("viridis", as\_cmap=True))

Chart

Description automatically generated

data\_sel = data\_sel.drop(['Additional\_charges', 'TotalCharge'], axis=1).copy()

sns.heatmap(data\_sel.corr(), linewidths=1, cmap=sns.color\_palette("viridis", as\_cmap=True))

Chart

Description automatically generated with medium confidence

With the multicollinearity addressed, we now divide the dataframe into two data frames, one for the target variable and another for the remaining predictor variables. The predictor variables, being all numeric, are checked for the presence of outliers and observations containing one or more values deemed to be outliers dropped. The remaining observations then have their features normalized with the sklearn MinMaxScaler method. Finally, the target variable is encoded so that values of “Yes” become one and values of “No” become zero.

data\_tar = data\_sel['ReAdmis'].copy()

data\_sel = data\_sel.drop(['ReAdmis'], axis=1).copy()

for i in range(len(data\_sel.columns)):

#Compute the mean and standard deviation of each column

mean, std = np.mean(data\_sel.iloc[:,i]), np.std(data\_sel.iloc[:,i])

#Set the upper and lower bounds at three standard deviations from the mean

upper, lower = mean + 3 \* std, mean - 3 \* std

#Record the index for each row that contains a value outside the previously set boundaries

drop = [inx for inx, x in enumerate(data\_sel.iloc[:, i]) if x < lower or x > upper]

#Compare the list of indices to be dropped with those within the dataframe and drop those that still remain

#within the data frame while ignoring those that were already dropped

for d in drop:

if d in data\_sel.index:

data\_sel = data\_sel.drop(d)

data\_tar = data\_tar.drop(d)

mm = MinMaxScaler() #Instantiate the MinMaxScaler method

data\_sel[data\_sel.columns] = mm.fit\_transform(data\_sel) #Normalize the data

data\_tar\_list = []

for inx, val in enumerate(data\_tar):

if val == 'Yes':

data\_tar\_list.append(1)

else:

data\_tar\_list.append(0)

data\_tar = pd.Series(data\_tar\_list)

Finally, the data is divided into training and test data with sklearn’s train\_test\_split method.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(data\_sel, data\_tar,

test\_size=0.2, random\_state=42)

1. A copy of the cleaned data is provided as the attachment OFM3\_cleaned.csv

**Part IV: Analysis**

D.  Perform the data analysis and report on the results by doing the following:

1. The implementation of the KMeans clustering algorithm is done by fitting the algorithm to the training data through the KMeans’ fit method. Once this model has been fitted to the training data, we can see what labels the model provides to see if it matches the labelling structure we use for our target variable.



Next, we use the model’s predict method to predict labels for the training data. We can check the model’s accuracy with a confusion matrix.



This confusion matrix shows that the model predicts the wrong label much more frequently than it predicts the correct label. This is likely due to the model labeling each cluster opposite to the labeling structure used for our target variable. When we invert these labels, such that a label of zero becomes one and a label of one becomes zero, we see that the confusion matrix represents a fairly accurate model.

Aligning the labels of the clustering model with those of our target variable does not actually change how accurate our model is, but simply aligns the data with the target variable’s encoding. This is necessary because the model was not provided any guidance on how to name the clusters, so the order was chosen internally, without consideration for the target variable. In the initial confusion matrix, we see that the values are low on the diagonal and high in the corners, implying that the labels for each cluster are opposite to the labels for our target variable. By inverting the cluster labels, the confusion matrix is made easier to read as the diagonal now represents a correct prediction.  
The score method provides us some insight to the density and size of the cluster. The score we receive for this model implies that our data points are not densely clustered and the clusters are large in size.



1. Provide the code used to perform the clustering analysis technique from part 2.

kmeans = KMeans(n\_clusters=2, random\_state=42).fit(X\_train)

np.unique(kmeans.labels\_)

label\_pred = kmeans.predict(X\_test)

confusion\_matrix(y\_test, label\_pred)

label\_pred\_corrected = [(x-1)\*(-1) for x in label\_pred]

confusion\_matrix(y\_test, label\_pred\_corrected)

kmeans.score(X\_test)

**Part V: Data Summary and Implications**

E.  Summarize your data analysis by doing the following:

1. When compared with the labels for the test set, the clustering model’s predictions on the test predictor values are astonishingly accurate. Only 239 of the nearly 2000 observations are incorrectly identified and all the patients that were readmitted were properly identified. The mislabeling observed in this confusion matrix is due to the assumption of the KMeans algorithm that clusters are of the same size, while our data has a smaller percentage of patients in the “No” group than the “Yes” group.
2. The results of this model tell us that the numeric features of patients that were readmitted, or not respectively, are similar and a strong basis of prediction for readmittance. This is not a surprise as the numeric variables include information on the length of a patient’s stay which has been identified as a strong predictive characteristic of patient readmittance in previous tasks.
3. KMeans clustering assumes that clusters are of the same size, leading the predictor method to predict the same number of observations for each label, while our data is comprised of a much higher amount of patients who were not readmitted than patients who were.
4. Although the model does a good job of clustering the data and has a strong predictive ability, the analysis does not provide any information about the causes of readmittance on its own. When combined with the analyses performed in previous courses’ tasks, we can strengthen the basis for our recommendation to focus on reducing the amount of time a patient stays in the hospital while still providing adequate care as it has been identified as being the greatest predictor of patient readmittance.

**Part VI: Demonstration**

F.  [https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=9fb99553-4d3e-4e97-a7f5-add001333909#](https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=9fb99553-4d3e-4e97-a7f5-add001333909)

G.  No web sources were used to supply code for this analysis and all code was written by the student.

H.  Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.

Tait, A. (2017, January 31). *Assumptions can ruin your K-means clusters*. Learning Tree Blog. Retrieved October 29, 2021, from https://blog.learningtree.com/assumptions-ruin-k-means-clusters/.