## Task 1: Data mining using K-means

Using the k-means algorithm, we will examine our data to draw conclusions on how certain features impact readmission rates to the hospital. The goal of this analysis is to discover hidden relationships between our features in order to better understand which features lead to higher readmission rates. Utilizing the k-means algorithm, we can discover relationships between our feature variables and hospital readmissions using data mining best practices. K-means is a data mining technique that discovers how features are grouped together by clustering them around the closest mean centroid. In this paper, I will use K-means to try and understand how our data naturally clusters in hopes of answering our research question.

For this data set I will be using k-means as the algorithm for analyzing the data. K-means is an unsupervised machine learning model that seeks to find how data clusters itself and provides a strategy for examining the data and discovering unknown relationships. We can then analyze those relationships and draw conclusions on our customers' behavior and how that behavior is correlated with readmission rates.

The k-means clustering algorithm works by placing random centroids in the feature vector space and then measures the euclidean distance between each feature observation and the centroid. In order for this operation to succeed, observations must be numeric. Therefore, this method works by assuming that each input is a discrete or continuous number; categorical variables are not allowed. After all observations are aligned with a random centroid, they are averaged and this average becomes the new centroid for running the algorithm again. This process repeats until no observation changes its cluster membership.

An issue with using k-means is that we always have to specify the number of k we want before running the algorithm. There are some methods that can be used to find the ideal number of k and the one I will be using in this analysis uses the Sum of Squared Errors as the metric, displayed in an Elbow Plot. The Elbow Plot assigns k to the X axis and the SSE to the Y axis; the SSE value is called inertia when doing k-means and can be called on the Kmeans object in Scikit-learn.

The reason that we need to find the appropriate level of k is because the scale at which we look at the clusters can impact our understanding of clusters. For instance, if we have 2 groups of 4 versus if we have 4 groups of 2; "zoom out" of the vector space and you can notice more and different sets of clusters. Therefore, selecting the appropriate k is a question of granularity. Using the Elbow Method, we will search for our ideal k which is determined by the inertia metric (which is just a range of Sum of Squared Error values starting at 0.)

In this analysis I will be using the Anaconda package for Python which comes included with all the necessary libraries that we will use to implement k-means such as Scikit_learn, numpy, pandas, and matplotlib.

The most import pre-processioning goal in machine learning is making sure your data set is clean and not missing any values, which I have done shown in the code attached to this paper. After we make sure our values are all clean and not missing any data we can move onto modeling. We then perform K-means and show the results of our clusters. For this model I will be using the following discrete values taken from the medical dataset: Initial_days and TotalCharge. Then I will compare these values and examine their relationship to readmissions in the conclusion.

My steps include importing the proper libraries into the python console and loading the .CSV file provided with the course into a pandas dataframe. Then I will call the appropriate data science methods on the data to check for null or missing values, drop features that we aren't using in the model, and print the results to the console. Then we will fit our model to our data and analyze the results. Finally, I will export the cleaned dataset used for the analysis as a .CSV file. An important metric is the inertia metric which is used to evaluate the algorithms' progress; when the difference between the previous cluster’s mean and the new cluster’s mean is below the inertia threshold the algorithm stops.

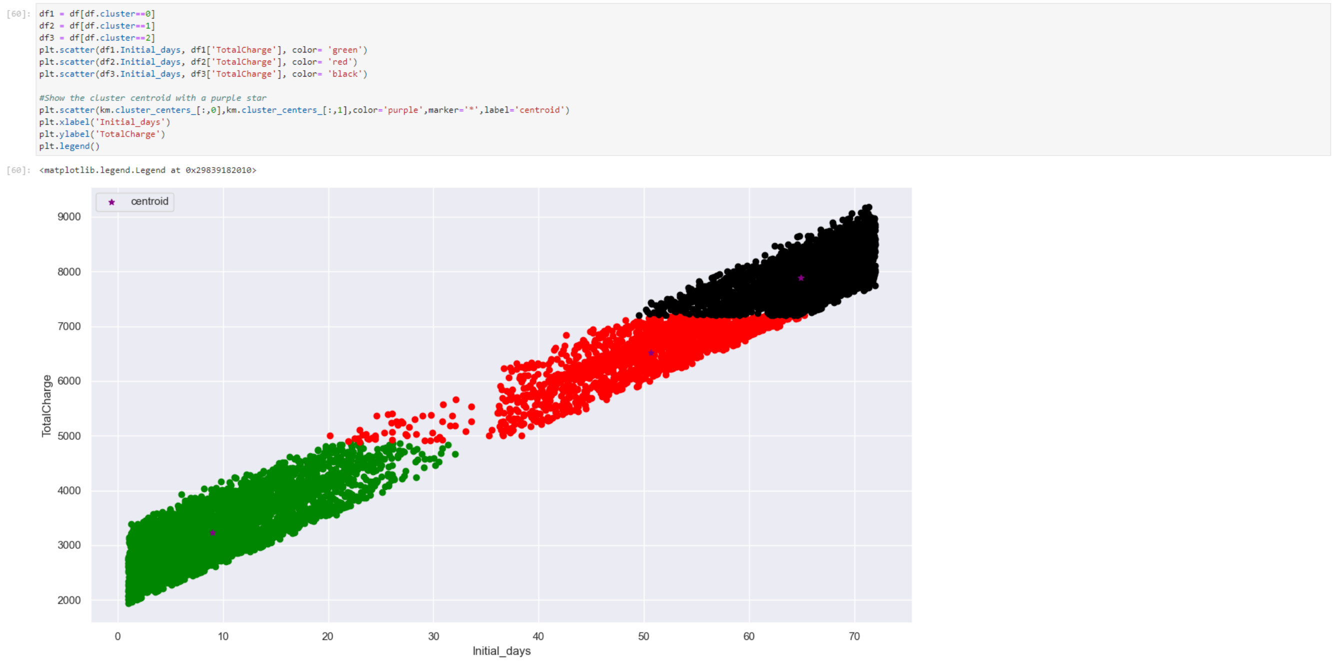
The analysis technique used is k-means as it provides us with a clear way to cluster data and look at the relationships between data points. The basic steps of the algorithm are summarized nicely by the official documents for Scikit-learn:

**1. Initialization:** directly after starting it, the initial centroids (cluster centers) are chosen. Scikit-learn supports two ways for doing this: firstly, random, which selects k samples from the dataset at random. Secondly, k-means++, which optimizes this process.
**2. Centroid assignment:** each sample in the dataset is assigned to the nearest centroid.
**3. Centroid correction:** new centroids are created by computing new means for the assignments created in step 2.
**4. Difference comparison:** for each centroid, the difference between old and new is compared, and the algorithm stops when the difference is lower than a threshold called inertia, or tolerance. Otherwise, it moves back to step 2.

The k-means algorithm is shown by the following formula:

$$\sum_{i=0}^{n} \min_{\mu_j \in C}(||x_i - \mu_j||^2)$$

Accuracy of k-means is difficult to measure because k-means is unsupervised: the true population labels are unknown to the algorithm. Therefore you must know the actual labels or ‘ground truth’ in order to measure accuracy internally. An ‘alternative to internal criteria is direct evaluation in the application of interest’[1]. An example of this would be like the hospital comparing readmission rates using multiple different clustering techniques over a given time period and see if certain clustering techniques lead to lower readmission rates.

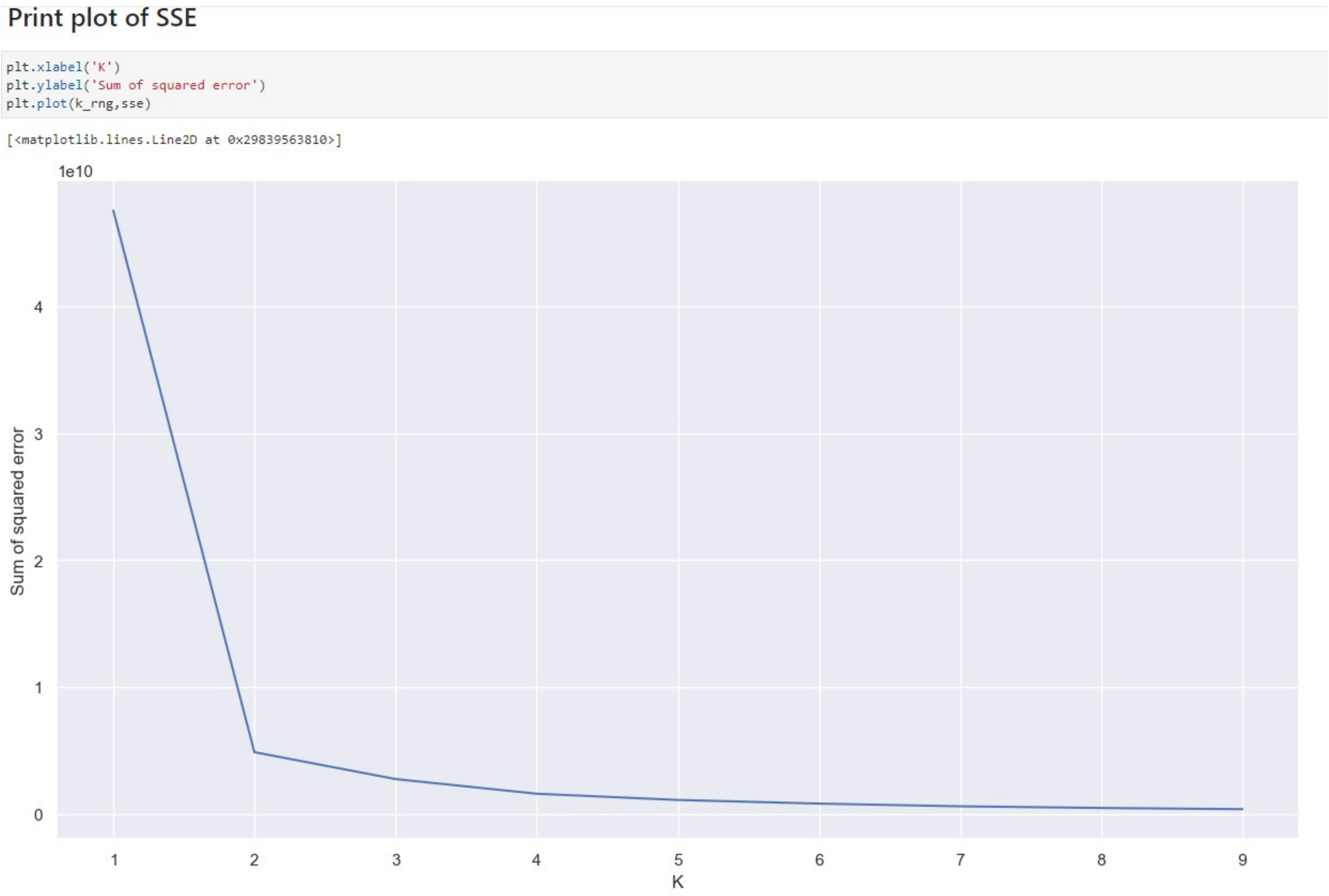


```
[60]: df1 = df[df.cluster==0]
      df2 = df[df.cluster==1]
      df3 = df[df.cluster==2]
      plt.scatter(df1.Initial_days, df1['TotalCharge'], color= 'green')
      plt.scatter(df2.Initial_days, df2['TotalCharge'], color= 'red')
      plt.scatter(df3.Initial_days, df3['TotalCharge'], color= 'black')

      #Show the cluster centroid with a purple star
      plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color='purple',marker='*',label='centroid')
      plt.xlabel('Initial_days')
      plt.ylabel('TotalCharge')
      plt.legend()

[60]: <matplotlib.legend.Legend at 0x29839182010>
```

Looking at the scatter plot cluster for TotalCharge and Initial_days, we can see that using a k of 2 showed a roughly 50-50 split of our customers and that Initial_days has a centroid of 60 days. The ideal number of k was 2, according to the elbow plot, which lead to two clusters of Initial_days vs TotalCharge, one green and one red. The original k of 3 did not increase the strength of our clusters since the Sum or Squared Errors (inertia) metric was not minimized that much between k=2 and k=3. Therefore, a k of 2 was used to cluster. This implies that our customers group themselves into 2 clusters normally and we can focus our attention on the single cluster that has a higher positive correlation with readmission.



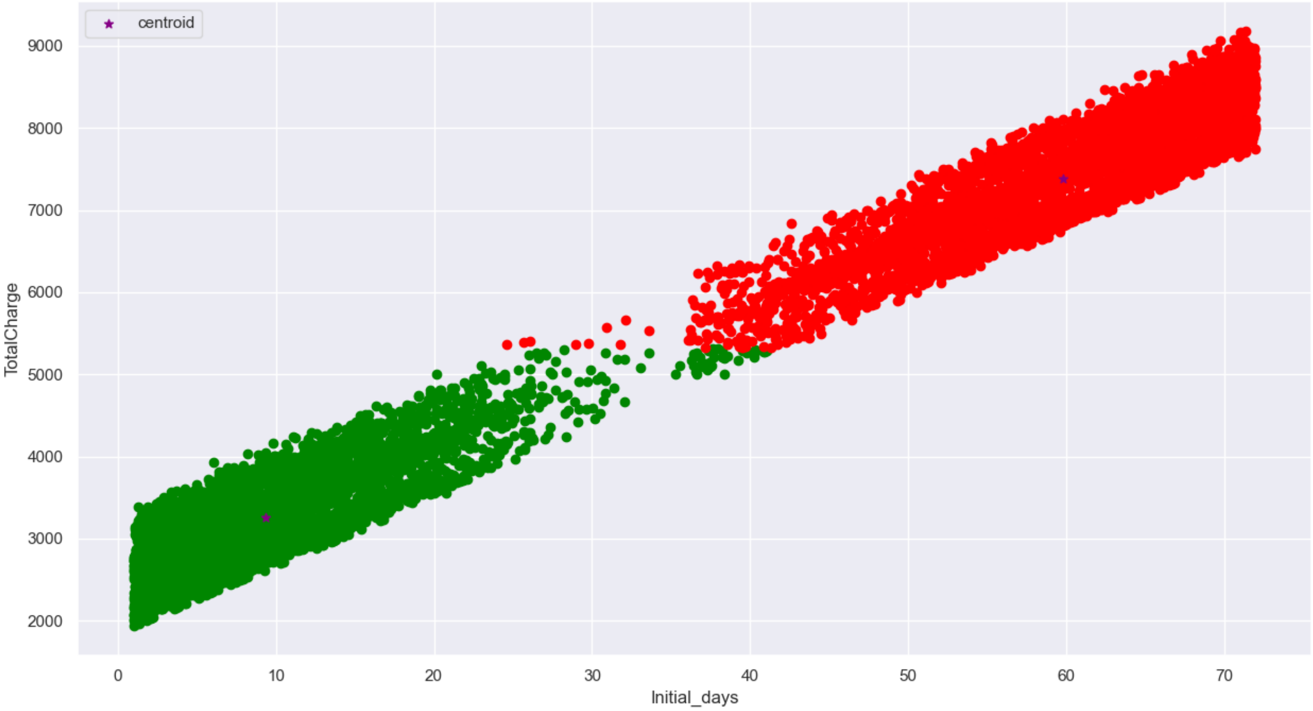
Print plot of SSE

```
plt.xlabel('K')
plt.ylabel('Sum of squared error')
plt.plot(k_rng,sse)
```

```
[<matplotlib.lines.Line2D at 0x29839563810>]
```

One limitation of this analysis is that we can not implement changes and see the results. If we could focus on one cluster over another and perform A/B testing of different data sets, then we could see if our clustering lead to positive business results for the hospital. Since we do not have that ability, our clusters' overall accuracy as a true model of the population is unknown.

Using the elbow plot and bi-variate summary statistics, we know that a higher amount of Initial_days is associated with higher readmission rates. Since higher initial_days are also positively correlated with higher total charges, the hospital could focus its efforts on the red cluster over the green. This is because the likelihood that you are in the red cluster increases your likelihood of being readmitted due to the positive correlation between Initial_days and ReAdmis. It is therefore my recommendation to the organization that they focus their attention on the customers who are in the red cluster (when k=2) shown below.



centroid
9000
8000
7000
6000
5000
4000
3000
2000
TotalCharge
0
10
20
30
40
50
60
70
Initial_days