In [2]:	<pre>#Adding various required libraries from ast import increment_lineno import pandas as pd from sklearn.cluster import KMeans import folium import matplotlib.pyplot as plt import seaborn as sns %matplotlib inline #Code for reading the csv file containing the data df = pd.read_csv("Kerala Blackspots with nearest hospitals finalised dataset (csv) (2).csv")</pre>
Out[2]:	SI. Name of District Name of Police Station Name of Landmark Name of Location Name of Location Name of Location Name of Location Name of Road National
	1 2 Kollam Chavara PS Regent Lake Palace - 110m before Hindustan Pet Parimanam Temple 8.954698 76.534274 Highway 66 8.955805 76.534519 357 NH 17 99 2 3 Thirtugenenthanuram CITY TRAFFIC Tax Towers - Karamana Karamana Lunction 8.482015 76.967072 Highway 66 8.481656 76.963152 314 NH 8 94
	3 4 Malappuram KUTTIPURAM Vyapara Bhavan - 200m after Highway Junction Kuttipuram Highway Junction 10.842810 76.030021 National Highway 66 (NH 66) 10.854681 76.037121 306 NH 12 86 4 5 Thiruvananthapuram Balaramapuram Oil Petrol Pump Balaramapuram Junction 8.430501 77.046417 National Highway 66 (NH 66) 8.426988 77.043805 300 NH 6 92
In [3]:	#Assigning columns "Accident Severity Index", "Number of Fatalities", "Sum of Fatal & Grievous Injury Crashes" to a variable 'z' z = df[["Accident Severity Index", "Number of Fatalities", "Sum of Fatal & Grievous Injury Crashes"]] #Printing z
Out[3]:	Accident Severity Index Number of Fatalities Sum of Fatal & Grievous Injury Crashes 0 366 15 102 1 357 17 99
	2 314 8 94 3 306 12 86 4 300 6 92 Below 3 lines of code is for finding the optimum number of clusters using Elbow method.
In [5]:	<pre>wcss = [] for i in range(1,11): model = KMeans(n_clusters=i) y_kmeans = model.fit_predict(z) wcss.append(model.inertia_)</pre>
	C:\Users\abita\anaconda3\envs\Intel_code\lib\site-packages\sklearn\cluster_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning warnings.warn(C:\Users\abita\anaconda3\envs\Intel_code\lib\site-packages\sklearn\cluster_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1. warnings.warn(
	C:\Users\abita\anaconda3\envs\Intel_code\lib\site-packages\sklearn\cluster_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning warnings.warn(C:\Users\abita\anaconda3\envs\Intel_code\lib\site-packages\sklearn\cluster_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1. warnings.warn(
	C:\Users\abita\anaconda3\envs\Intel_code\lib\site-packages\sklearn\cluster_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set th e value of `n_init` explicitly to suppress the warning warnings.warn(C:\Users\abita\anaconda3\envs\Intel_code\lib\site-packages\sklearn\cluster_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are l ess chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1. warnings.warn(C:\Users\abita\anaconda3\envs\Intel_code\lib\site-packages\sklearn\cluster_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set th
	e value of `n_init` explicitly to suppress the warning warnings.warn(C:\Users\abita\anaconda3\envs\Intel_code\lib\site-packages\sklearn\cluster_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are l ess chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1. warnings.warn(C:\Users\abita\anaconda3\envs\Intel_code\lib\site-packages\sklearn\cluster_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set th e value of `n_init` explicitly to suppress the warning
	warnings.warn(C:\Users\abita\anaconda3\envs\Intel_code\lib\site-packages\sklearn\cluster_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are l ess chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1. warnings.warn(C:\Users\abita\anaconda3\envs\Intel_code\lib\site-packages\sklearn\cluster_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set th e value of `n_init` explicitly to suppress the warning warnings.warn(
	C:\Users\abita\anaconda3\envs\Intel_code\lib\site-packages\sklearn\cluster_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1. warnings.warn(C:\Users\abita\anaconda3\envs\Intel_code\lib\site-packages\sklearn\cluster_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning warnings.warn(C:\Users\abita\anaconda3\envs\Intel_code\lib\site-packages\sklearn\cluster_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are l
	ess chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1. warnings.warn(C:\Users\abita\anaconda3\envs\Intel_code\lib\site-packages\sklearn\cluster_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning warnings.warn(C:\Users\abita\anaconda3\envs\Intel_code\lib\site-packages\sklearn\cluster_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are 1 ess chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
	warnings.warn(C:\Users\abita\anaconda3\envs\Intel_code\lib\site-packages\sklearn\cluster_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning warnings.warn(C:\Users\abita\anaconda3\envs\Intel_code\lib\site-packages\sklearn\cluster_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1. warnings.warn(
	C:\Users\abita\anaconda3\envs\Intel_code\lib\site-packages\sklearn\cluster_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning warnings.warn(C:\Users\abita\anaconda3\envs\Intel_code\lib\site-packages\sklearn\cluster_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1. warnings.warn(
<pre>In [6]: Out[6]: In [7]:</pre>	model.inertia_ 9796.47560565871 #Plotting it into the Elbow graph plt.plot(range(1, 11), wcss)
	plt.xlabel('Number of clusters') plt.ylabel('WCSS') plt.title('Elbow Method') plt.show() Elbow Method
	500000 -
	SS 300000 -
	100000 -
	0 - 2 4 6 8 10 Number of clusters Therefore the optimum number of clusters is 3 from the Elbow graph
In [9]:	<pre>#Clustering the data into 3 clusters model = KMeans(n_clusters=3) y_kmeans = model.fit_predict(z) C:\Users\abita\anaconda3\envs\Intel_code\lib\site-packages\sklearn\cluster_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set th</pre>
In [10]:	<pre>e value of `n_init` explicitly to suppress the warning warnings.warn(C:\Users\abita\anaconda3\envs\Intel_code\lib\site-packages\sklearn\cluster_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are l ess chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1. warnings.warn(df['y'] = y_kmeans</pre>
In [11]: In [12]:	<pre>#Downloading the data to be clustered df.to_csv("final_data_with_cluster.csv") #Printing data to be mapped with k means variable 'y' df.head()</pre>
Out[12]:	SI. Name of District No. Name of Police Station Name of Police Station Name of Landmark Name of Location Name of Location Name of Location Name of Road National Hospital Near Longitude Name of Road National Highway 66 (NH 66) National Highway 66 (NH 66) Name of Road National Highway 66 (NH 66)
	12KollamChavara PSRegent Lake Palace - 110m before Hindustan PetParimanam Temple Pet8.95469876.534274National Highway 66 (NH 66)8.95580576.534519357NH1799223ThiruvananthapuramCITY TRAFFICTax Towers - Karamana JunctionKaramana Junction8.48201576.967072Highway 66 (NH 66)8.48165676.963152314NH8942
	3 4 Malappuram KUTTIPURAM Vyapara Bhavan - 200m after Highway Junction Unation Junction 10.842810 76.030021 Highway 66 (NH 66) 10.854681 76.037121 306 NH 12 86 2 4 5 Thiruvananthapuram Balaramapuram Oil Petrol Pump Oil Petrol Pump Junction Unation Junction Section 10.842810 76.030021 Highway 66 (NH 66) 10.854681 76.037121 306 NH 12 86 2
In [13]: Out[13]:	<pre>#Creating clusters plt.scatter(df['Longitude'], df['Lattitude'],c=df['y'],) <matplotlib.collections.pathcollection 0x1c39bf4ee30="" at=""></matplotlib.collections.pathcollection></pre>
	11.0 -
	10.0 -
	9.5 - 9.0 - 8.5 -
In [15]:	75.8 76.0 76.2 76.4 76.6 76.8 77.0 below 3 lines are for categorising and grouping the locations based on (low,moderate and high) import seaborn as sns
In [16]: Out[16]:	<pre>import seaboff as sits import matplotlib.pyplot as plt df['y'] = pd.Categorical(df.y) df.y.value_counts() 0 153 1 69</pre>
In [17]: Out[17]:	<pre>2 16 Name: y, dtype: int64 sns.countplot(x ='y', data = df) <axessubplot:xlabel='y', ylabel="count"></axessubplot:xlabel='y',></pre>
	140 - 120 -
	100 - tig 80 -
	60 -
	40 -
In [18]: In [19]:	#Below codes are for deviding the clusters into 3 groups (cluster1, cluster2 and cluster3) cluster1 = df[['Lattitude', "Longitude"]][df['y'] == 0].values.tolist() cluster2 = df[['Lattitude', "Longitude"]][df['y'] == 1].values.tolist() cluster3 = df[['Lattitude', "Longitude"]][df['y'] == 2].values.tolist() #Printing the Kerala state openstreet map
	#Below codes are for deviding the clusters into 3 groups (cluster1, cluster2 and cluster3) cluster1 = df[['Lattitude', "Longitude"]][df['y'] == 0].values.tolist() cluster2 = df[['Lattitude', "Longitude"]][df['y'] == 1].values.tolist() cluster3 = df[['Lattitude', "Longitude"]][df['y'] == 2].values.tolist()
In [19]:	#Below codes are for deviding the clusters into 3 groups (cluster1, cluster2 and cluster3) cluster1 = dff[['Lattitude', "Longitude"]][df['y'] == 0].values.tolist() cluster2 = dff[['Lattitude', "Longitude"]][df['y'] == 1].values.tolist() cluster3 = dff[['Lattitude', "Longitude"]][df['y'] == 2].values.tolist() #Printing the Kerala state openstreet map kerala_map = folium.Map(location=[10.8505, 76.2711], zoom_start=8, tiles = "openstreetmap") kerala_map #Bassan Bengaluru Velore Kanchipuram Mangaluru Dassina Ranchipuram
In [19]:	##Blow codes are for deviding the clusters into 3 groups (cluster1, cluster2 and cluster3) cluster1 = off['\tattitude', '\text{Longitude'}][\text{off}']' = 8].values.tolist() cluster2 = off['\text{Littude', '\text{Longitude'}}[\text{off}']' = 1].values.tolist() cluster3 = off['\text{Littude', '\text{Longitude'}}[\text{off}']' = 2].values.tolist() cluster3 = off['\text{Littude', '\text{Longitude'}}[\text{off}'][\text{off}']' = 2].values.tolist() ### Printing the Kerala state openstreet map kerala map = folium.Map(location=[10.8585, 76.2711], zoom start=8, tiles = "openstreetmap") kerala map = folium.Map(location=[10.8585, 76.2711], zoom start=8, tiles = "openstreetmap") kerala map = folium.Map(location=[10.8585, 76.2711], zoom start=8, tiles = "openstreetmap") kerala map = folium.Map(location=[10.8585, 76.2711], zoom start=8, tiles = "openstreetmap") kerala map = folium.Map(location=[10.8585, 76.2711], zoom start=8, tiles = "openstreetmap") kerala map = folium.Map(location=[10.8585, 76.2711], zoom start=8, tiles = "openstreetmap") kerala map = folium.Map(location=[10.8585, 76.2711], zoom start=8, tiles = "openstreetmap") kerala map = folium.Map(location=[10.8585, 76.2711], zoom start=8, tiles = "openstreetmap") kerala map = folium.Map(location=[10.8585, 76.2711], zoom start=8, tiles = "openstreetmap") kerala map = folium.Map(location=[10.8585, 76.2711], zoom start=8, tiles = "openstreetmap") kerala map = folium.Map(location=[10.8585, 76.2711], zoom start=8, tiles = "openstreetmap") kerala map = folium.Map(location=[10.8585, 76.2711], zoom start=8, tiles = "openstreetmap") kerala map = folium.Map(location=[10.8585, 76.2711], zoom start=8, tiles = "openstreetmap") kerala map = folium.Map(location=[10.8585, 76.2711], zoom start=8, tiles = "openstreetmap") kerala map = folium.Map(location=[10.8585, 76.2711], zoom start=8, tiles = "openstreetmap") kerala map = folium.Map(location=[10.8585, 76.2711], zoom start=8, tiles = "openstreetmap") kerala map = folium.Map(location=[10.8585, 76.2711], zoom start=8, tiles = "ope
In [19]:	#Below codes are for deviding the clusters into 3 groups (cluster) cluster2 and cluster3) cluster1 = df[['lattitude', "longitude"][df['y'] == 0].values.tolist() cluster2 = df[['lattitude', "longitude"][df['y'] == 1].values.tolist() cluster3 = df[['lattitude', "longitude"][df['y'] == 2].values.tolist() #Printing the Kerala state openstreet map Kerala_map ### Mangahuru Dauman Mangahuru Dauman Mangahuru D
In [19]:	### Secure codes are for deviding the clusters into 3 groups (clusters, clusters and clusters) clusters = off [!lattitude], "Congitude=][df['y'] == 0] values.tolist() clusters = off [!lattitude], "Congitude=][df['y'] == 1] values.tolist() clusters = off [!lattitude], "Congitude=][df['y'] == 2] values.tolist() ###################################
In [19]:	**************************************
In [19]:	### Additional Codes are for Sevicing the closeers into 3 proups (cluster) and clusters) #### Clusters of [['Latitude", "conjitude"]][f[']"] == 0] yasles tolist() clusters = of [['Latitude", "conjitude"]][f[']"] == 1) yasles tolist() clusters = of [['Latitude", "conjitude"]][f[']"] == 2] yasles tolist() clusters = of [['Latitude", "conjitude"]][f[']"] == 2] yasles tolist() ###################################
In [19]:	## August a control and the secondary the clusters and a ground (Canterd, Clusters) and clusters) Canterd, Clusters, Secondary Canterd, Clusters, Secon
In [19]:	Service can be considered to the contract of t
In [19]: Out[19]:	Action codes on for processing the classics of the process (classics codes on for process) (classics) (cla
In [19]: Out[19]: In [21]:	Section of the analysis of the control in the control of the contr
In [19]: Out[19]: In [21]:	Setting and the control of the contr
In [19]: Out[19]: In [21]:	State in the cost to emboding a more state of the cost and analysis of the cost to emboding a more state of the cost to em
In [19]: Out[19]: In [21]:	Series and rick to decorate an explanation and a gradient measurement of a gradient series and a gradient seri
In [19]: Out[19]: In [21]:	Size A Part of the standard of the grant of the standard of th
In [19]: Out[19]: In [21]:	Private lands in entertaining maging to transport to the meritained and science of the control o
In [19]: Out[19]: In [21]:	Experience of the control of the con