Untitled1

October 14, 2019

0.0.1 Situation

Have you ever hopped on a scooter and realized the battery is dead? What an upsetting experience!

In order to prevent that horrible user experience from happening, the data science team is focusing its efforts on coming up with the best scooter charging strategy.

You can find a data set below with scooters' current geolocation and power level. Power level ranges from 0 - 5 (0 as completely out of battery, 5 as fully charged). It takes 5 hours to charge a scooter's power from 0 to 5. TechPointX talent team also has a mega charging bus that drives around to pick up scooters and charge them inside. Unfortunately, the bus can only park and start at location 20.19 (xcoordinate), 20.19 (ycoordinate) and only travel 50 miles per hour.

0.0.2 Your Task

Review the data set, and draw any conclusions you can find from the data set. Try to identify the most popular scooter location, demonstrate your findings using data visualization tools, calculate operation time cost (Operation Time Cost: How long it takes to fully charge all the scooters), and come up with the most efficient scooter charging strategy.

[15]:
```
# Importing Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.pyplot import subplots
from sklearn.cluster import KMeans
```

Let's start by reading data.

[2]:
```
scooter_data = pd.read_csv("2019-XTern- Work Sample Assessment Data Science-DS.
 ↪csv")
scooter_data
```

[2]:

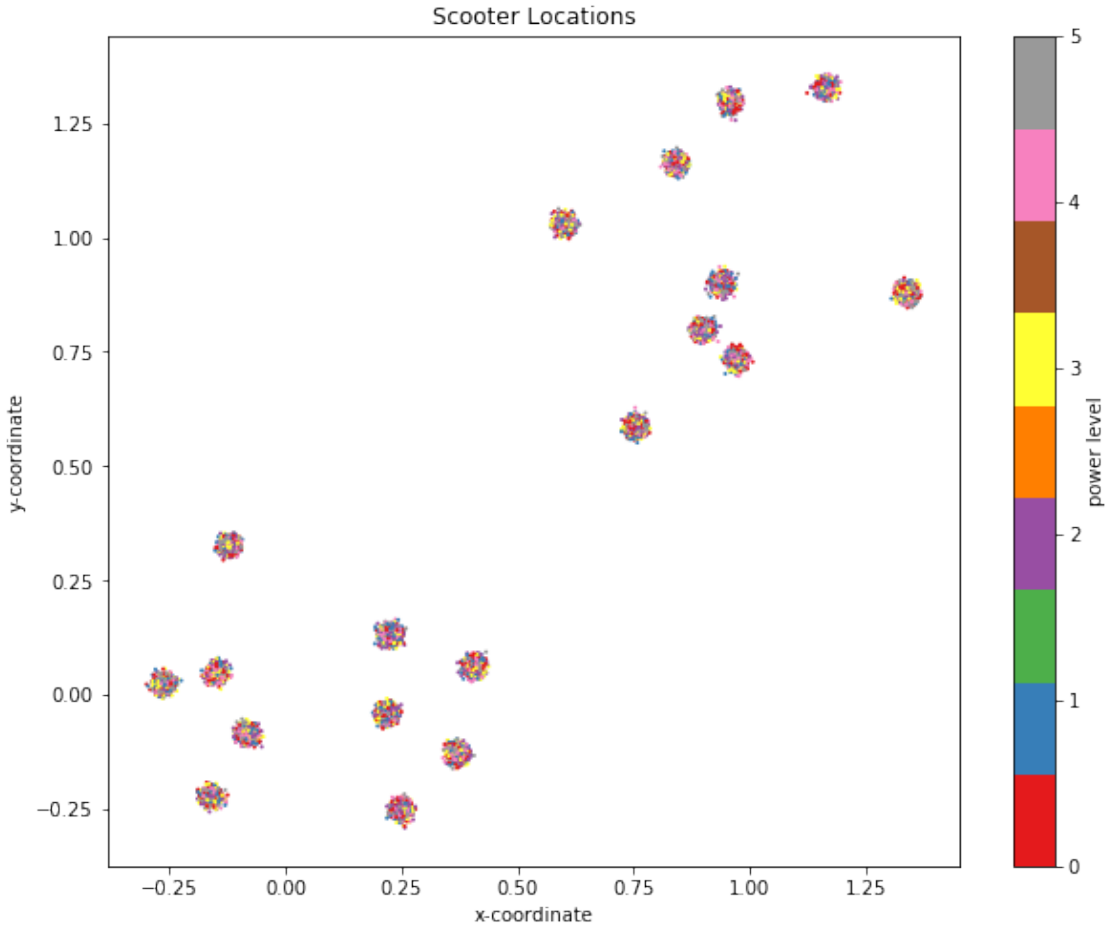
	scooter_id	xcoordinate	ycoordinate	power_level
0	0	0.906835	0.776484	0
1	1	0.928587	0.804964	2
2	2	0.904091	0.784043	1
3	3	0.906752	0.804461	0
4	4	0.900641	0.781683	4
5	5	0.899680	0.790893	2
6	6	0.873765	0.808707	4

7	7	0.913476	0.789742	5
8	8	0.915256	0.790685	4
9	9	0.916273	0.785860	5
10	10	0.906847	0.805807	0
11	11	0.905919	0.811344	3
12	12	0.901493	0.791249	1
13	13	0.904264	0.797024	1
14	14	0.898657	0.799972	2
15	15	0.885372	0.780831	1
16	16	0.884462	0.799252	2
17	17	0.926481	0.802019	4
18	18	0.904266	0.803249	2
19	19	0.904425	0.792057	0
20	20	0.875620	0.801346	0
21	21	0.900621	0.810840	3
22	22	0.894725	0.806605	4
23	23	0.911556	0.784681	5
24	24	0.896705	0.801677	4
25	25	0.895725	0.790989	2
26	26	0.894769	0.783909	2
27	27	0.885053	0.794401	2
28	28	0.912030	0.794157	0
29	29	0.905292	0.797360	1
...	...	...	...	...
25638	25638	0.944468	1.297093	0
25639	25639	0.970065	1.288834	4
25640	25640	0.966601	1.286650	2
25641	25641	0.978843	1.301902	5
25642	25642	0.956143	1.287155	0
25643	25643	0.950733	1.302323	0
25644	25644	0.959514	1.303935	4
25645	25645	0.970499	1.307217	5
25646	25646	0.943649	1.313249	3
25647	25647	0.957838	1.276586	2
25648	25648	0.959622	1.296374	3
25649	25649	0.979135	1.299700	4
25650	25650	0.978101	1.279493	0
25651	25651	0.952058	1.290147	0
25652	25652	0.946096	1.303115	3
25653	25653	0.950750	1.295193	1
25654	25654	0.963100	1.302630	3
25655	25655	0.961126	1.298540	2
25656	25656	0.963377	1.309476	4
25657	25657	0.947093	1.298891	3
25658	25658	0.971392	1.291350	3
25659	25659	0.961979	1.284334	5
25660	25660	0.972953	1.297746	5

```
25661       25661     0.945709     1.266022            1
25662       25662     0.969872     1.291282            0
25663       25663     0.943601     1.292008            0
25664       25664     0.971847     1.282083            0
25665       25665     0.956886     1.286652            4
25666       25666     0.959825     1.297630            1
25667       25667     0.963641     1.300093            4

[25668 rows x 4 columns]
```

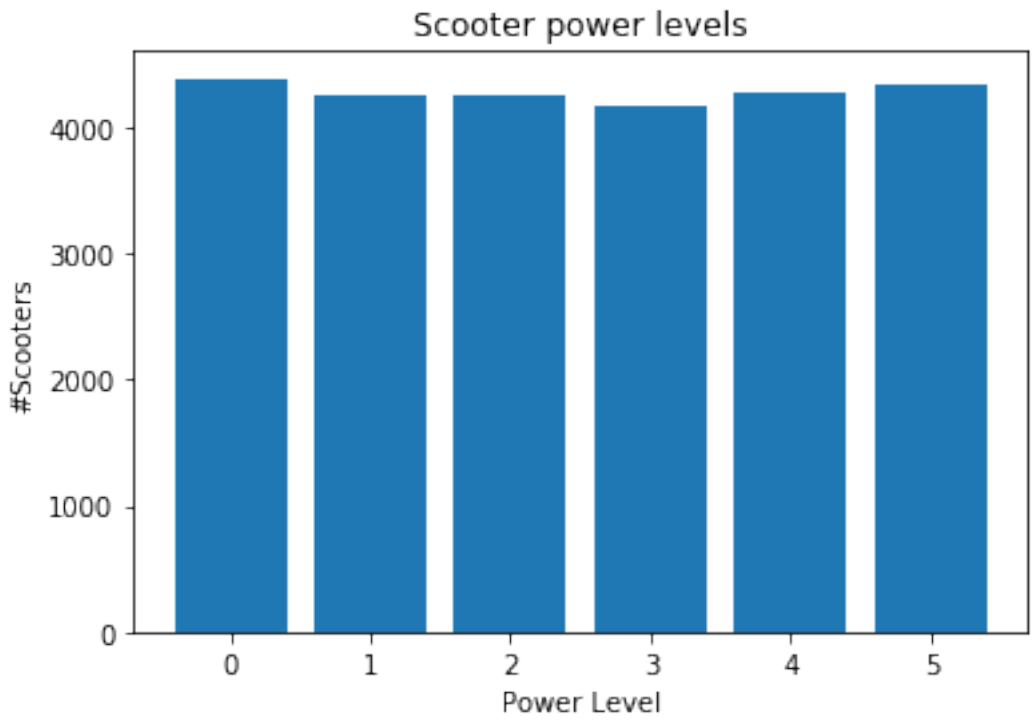
To understand the scooters location and how they are spread, scatter plot has been created. Different colors have been picked for different power levels.

[3]:
```
plt.figure(figsize=(10, 8))
plt.scatter(scooter_data['xcoordinate'], scooter_data['ycoordinate'],␣
 ↪marker='o', c = scooter_data['power_level'],
            cmap = plt.get_cmap('Set1'), s = 1)
cbar = plt.colorbar()
plt.title('Scooter Locations')
plt.xlabel("x-coordinate")
plt.ylabel("y-coordinate")
cbar.ax.set_ylabel('power level', rotation=90)
plt.show()
```



To understand the data better, a plot with number of scooters vs their power level has been drawn below.

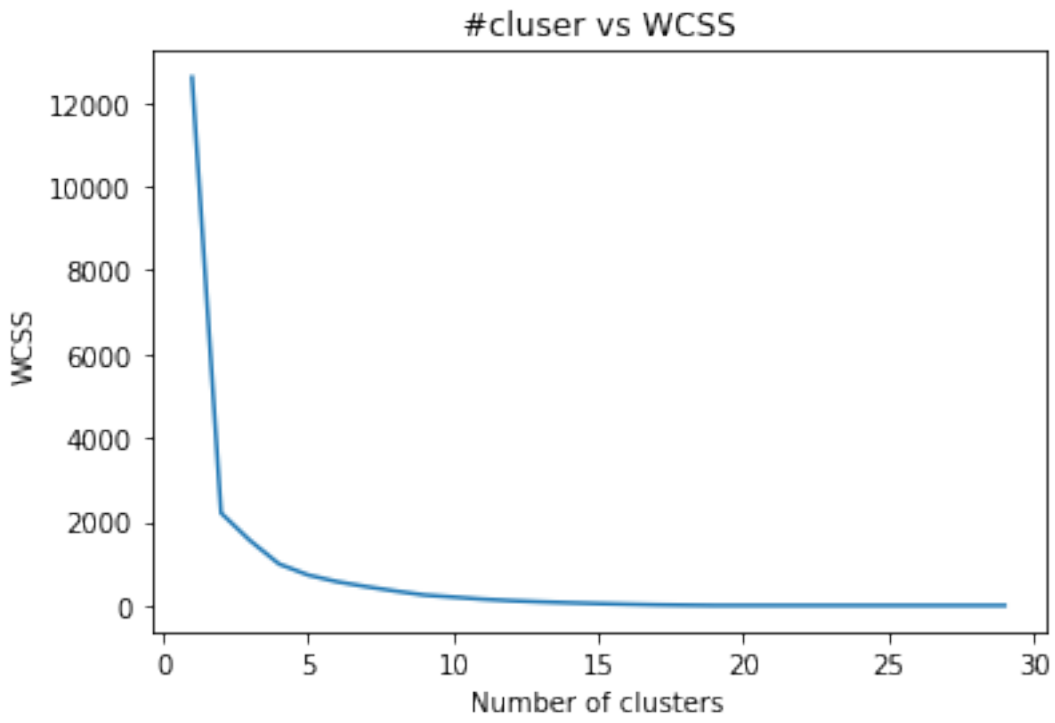
[4]:
```
battery_levels = []
number_of_scooters = []
for battery_level in sorted(scooter_data['power_level'].unique()):
    battery_levels.append(battery_level)
    number_of_scooters.append(scooter_data[scooter_data['power_level'] ==␣
 ↪battery_level]['scooter_id'].count())

plt.bar(battery_levels, number_of_scooters)
plt.xlabel('Power Level')
plt.ylabel('#Scooters')
plt.title('Scooter power levels')
plt.show()
```



It can be observed that number of scooters per power level is almost same.

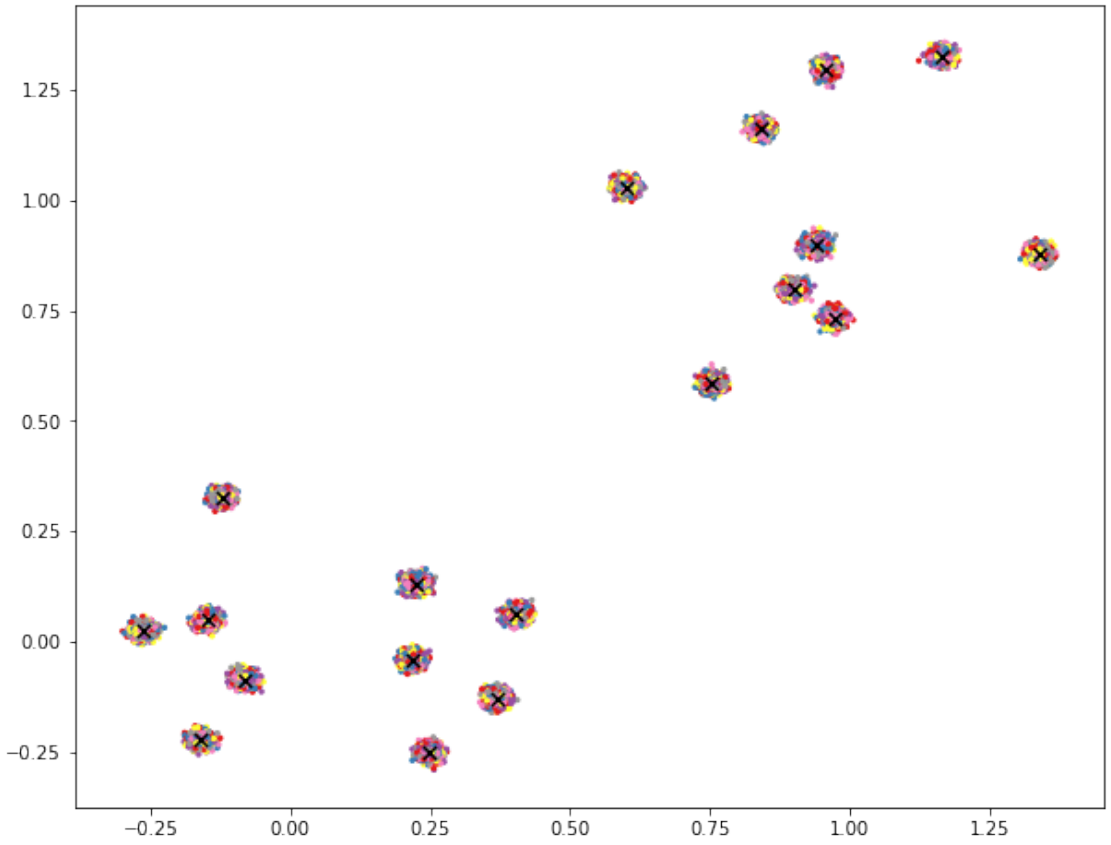
To cluster the scooters, we can directly divid into 19 clusters(by observing the data), however, we can verify it by plotting wcss with # of clusters.

[5]:
```
# Verifying number of cluster, should be approximtely 19
wcss = [] # Within-Cluster-Sum-of-Squares
X = scooter_data[['xcoordinate', 'ycoordinate']]
for i in range(1, 30):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10,␣
 ↪random_state=0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 30), wcss)
plt.title('#cluser vs WCSS')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



WCSS is pretty much same after 19, therefore, n_cluster=19 for K means clustering algorithm.

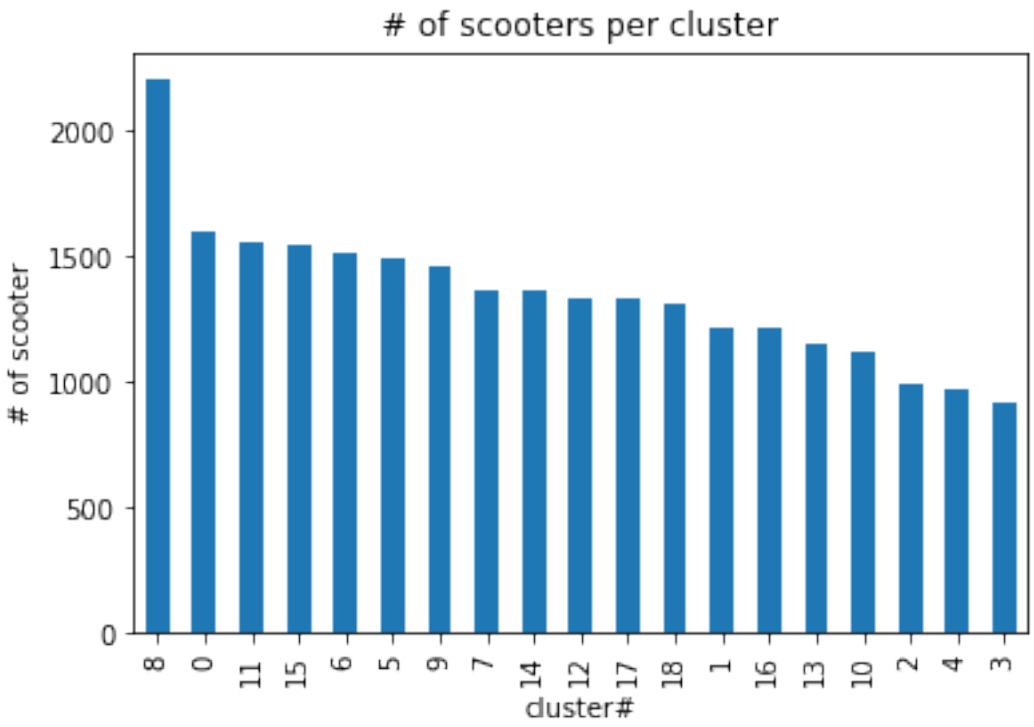
[6]:
```
plt.figure(figsize=(10, 8))
Kmeans = KMeans(n_clusters=19)
X = scooter_data[['xcoordinate', 'ycoordinate']]
Kmeans.fit(X)
labels = Kmeans.predict(X)
centroids = Kmeans.cluster_centers_

plt.scatter(scooter_data['xcoordinate'], scooter_data['ycoordinate'],␣
 ↪marker='o', c = scooter_data['power_level'],
            cmap = plt.get_cmap('Set1'), s = 5)
for idx, centroid in enumerate(centroids):
    plt.scatter(*centroid, s=50, c='black', marker = 'x')
plt.show()
```



To understand the most popular scooter location, lets plot the number of scooters in each cluster.

[29]:
```
scooter_data.insert(0, 'cluster#', labels)
X = scooter_data[['cluster#', 'scooter_id','power_level']]
ax = X['cluster#'].value_counts().plot.bar(title = '# of scooters per cluster')
ax.set_xlabel("cluster#")
ax.set_ylabel("# of scooter")
```

[29]:
```
Text(0, 0.5, '# of scooter')
```



Cluster number 8 has most number of scooters. Therefore, most popular location is x = 0.22556029, y = 0.13144752

0.0.3 Assumptions for Optimal Scooter Charging Strategy

1. A completely charged scooter works entire day, i.e, a scooter at power level 5 can work entire day.
2. Therefore, each battery level works roughly around 24/5 = 4.8 hours
3. Let's assume the charging bus size is 1500.
4. Charging should take place at night, to avoid scooter shortage atleast in the most popular areas.

0.0.4 Observations based on above assumptions

1. Anything with more than 2 power level is enough to cover half of the day.

0.0.5 Strategy

1. First target scooters with power level 0, 1 and 2 from the cluster which is mid way in popularity, i.e, start from cluster 6 and work its was to the most popular cluster.
2. The benefit of this strategy will be that: by the time you reach the most popular cluster, you will already have some charged scooters which can be directly replaced. This would prevent the shortage of scooters at more popular locations at any point of time.

3. First round of charging will charge all scooter with power level 0,1 and 2. The second round will charge all scooters with power level 3, 4 and 5, which in a realistic world will now have less power than before making them ideal for pick up.

0.0.6 For example:

First round: power level [0, 1, 2] 1. The charging bus will first pick 1500 or less scooters from cluster 12 then to cluster 14, cluster 7, all the way to most popular cluster 8. 2. At each cluster, charging bus will drop the scooters which are charged and will pick up new scooters as per the capacity. 3. By the time it reaches cluster 8 (most popular), it will already have charged scooters to replace to avoid the shortage. 4. Once it reaches to cluster 8, it will continue its way from the least popular region (cluster 3), then to cluster 4, cluster 2, all the way to cluster 8.

Second Round: power level [3, 4] 1. At the end of round 1, charging bus reached back to cluster 8 and it will follow the same stragety as round 1.