Part 2: Bonus 1

Based on the NY Parking Violation data, given a Black vehicle parking illegally at 34510, 10030, 34050 (streat codes). What is the probability that it will get an ticket? (very rough prediction).

Design

The K-Means algorithm is implemented with PySpark with the following steps:

1. Initialze spark session
2. Load in the dataset as DataFrame for preprocessing. Filter the dataframe color in black, and then selecting columns of Street Code. Transform the filter dataframe into rdd.

```
black_list = ['BK', 'BLK', 'BK/', 'BK.', 'BLK.', 'BLAC', 'Black', 'BCK', 'BC', 'B LAC']
black = df.filter(df['Vehicle Color'].isin(black_list))
dataPts = black.select(black['Street Code1'], black['Street Code2'], black['Street
Code3']).na.drop()
dataRDD = dataPts.rdd.map(lambda r: (r[0], r[1], r[2]))
```

3. K-Means

After preprocessing, I start to implement the Kmeans algorithm.

First, randomly select 4 rows as intial centroid:

```
ini_centroid = dataRDD.takeSample(False, 4)
```

Then, the K-Means algorithm was run iteratively. For each data point, we calculate their distances to the 4 initial centroids, and assign them to the cluster of their closest centroid. Next, for each cluster, we recalculate the new centroid by getting the mean of each column. By doing so, we have 4 new centriods, and we recalculate the distance between each data points to the new centroids. We iterate this process until the new centroids equal to the old centroids, or until the maximum times of iteration is reached.

- calculate each data's distance to the centroid
- asign each data to the closet cdntroid
- calculate the new centroid per cluster by finding their mean
- iteration: calculate each data's distance to the NEW centroid
- stop iteration when old_centroids = new_centroids or when the maximum number of iteration is reached

```
iters = 0
old_centroid = int_centroid

for m in range(40):
    map1 = dataRDD.map(lambda x: closestCenter(x, old_centroid))
    reduce1 = map1.groupByKey()
    map2 = reduce1.map(lambda x: cal_centroid(x)).collect() # collect a list
    new_centroid = map2
    converge = 0
    for i in range(k):
        if new_centroid[i] == old_centroid[i]:
            converge += 1
        else:
            diff = 0.0009
            closeDiff = [round((a - b)**2, 6) for a, b in zip(new_centroid[i], old_centroid[i])]
            if all(v <= diff for v in closeDiff):
                converge += 1
    if converge >= 4:
        print("Converge at the %s iteration\n" %(iters))
```

```
            print("\nFinal Centroids: %s" %(new_centroid))
            break
        else:
            iters += 1
            print("Iteration - %s round" %(iters))
            old_centroid = new_centroid
            print('Update:',old_centroid,'\n')
```

4. Probability of Getting Tickets

Given a car parked at street_code = [34510, 10030, 34050], the probability that a car will get tickets is defined by:

** % of gettiing tickets = (total tickets in the cluster)/((average illegal car in a street) * (the number of street codes in the cluster)) **

```
street_code = [34510, 10030, 34050]
closet = closestCenter(street_code, new_centroid)
map3 = dataRDD.filter(lambda x: closestCenter(x, new_centroid)[0] == closet[0]).collect()
count = len(map3)
token = dict(Counter(map3))
counter = len(token)
maxV = max(token.items(),key = itemgetter(1))[1]
probability = round(count/(maxV * counter), 6)
print ("Probability of getting ticets:\n")
print (probability)
```

The result is as following:

```
2019-04-23 17:44:37 INFO  DAGScheduler:54 - ResultStage 21 (collect at /spark-examples/labs/l
ab2/part2/./parking-kmeans.py:90) finished in 0.110 s
2019-04-23 17:44:37 INFO  DAGScheduler:54 - Job 12 finished: collect at /spark-examples/labs/
lab2/part2/./parking-kmeans.py:90, took 0.304983 s
Converge at the 8 iteration


Final Centroids: [[20571.907, 29129.7116, 30848.4605], [49073.6102, 13849.8898, 17240.4661],
[54765.4206, 59625.0561, 52164.3832], [5006.5079, 2299.828, 1649.1402]]
2019-04-23 17:44:37 INFO  SparkContext:54 - Starting job: collect at /spark-examples/labs/lab
2/part2/./parking-kmeans.py:115
```