Introduction

- Required Modules
  - Pandas
  - Levenshtein
  - Geopy

- Steps
  - Importing modules
  - Reading Input Dataset
  - Adding New Column
  - Comparing levenshtein distance whether it is less than 5 or not
  - Measuring Distance for entries which are within 200 meters
  - Classifying them 0 or 1
  - Creating output csv file after result

1. Import Modules

```
from Levenshtein import distance as lev #using levenshtein module we can find maximum number of single-character edits required
```

```
import pandas as p  #using pandas to read csv files
import geopy #using geopy to measure distance between entries
```

2. Read Input Data csv file

```
data = p.read_csv("assignment_data.csv") #reading input csv file

data #printing input csv file
```

	name	latitude	longitude
0	kbyJYJzqEVkEmpc	13.022860	77.688911
1	qbgufTPzZEkxNHV	13.093675	77.702779
2	XWIXkuyDacCmsaE	12.902879	77.633207
3	PWMPzEhgQSIJFtLI	12.864532	77.731435
4	PWMPzEhgQSIJFtL	12.865321	77.731473
...	...	...	...
11809	OtytDvqWIjeFdAt	12.864681	77.561012
11810	PGHxYVTKBPvJzRcr	12.878795	77.549211
11811	PGHxYVTKBPvJzRc	12.879561	77.549147
11812	KIOFLLHNAvTQOkU	13.061351	77.661640
11813	BKEsZzrJiKhKIae	13.019981	77.679988

11814 rows × 3 columns

3. Creating seperate column called " is_similar " to classify entries to 0 or 1. By default we are going to use 0 for now later it will be updated to required value.

```
is_similar = [] #creating empty list to create is_similar column to classify

for e in range(len(data)): #iterating total entries for is_similar column (11814 entries)
  is_similar.append(0) #adding values to list
data["is_similar"] =  is_similar #assigning column values to column

data
```

	name	latitude	longitude	is_similar
0	kbyJYJzqEVkEmpc	13.022860	77.688911	0
1	qbgufTPzZEkxNHV	13.093675	77.702779	0
2	XWIXkuyDacCmsaE	12.902879	77.633207	0
3	PWMPzEhgQSIJFtLl	12.864532	77.731435	0
4	PWMPzEhgQSIJFtL	12.865321	77.731473	0
...	...	...	...	...
11809	OtytDvqWIjeFdAt	12.864681	77.561012	0
11810	PGHxYVTKBPvJzRcr	12.878795	77.549211	0
11811	PGHxYVTKBPvJzRc	12.879561	77.549147	0
11812	KlOFLLHNAvTQOkU	13.061351	77.661640	0
11813	BKEsZzrJiKhKlae	13.019981	77.679988	0

11814 rows × 4 columns

4. Creating New list for only dataset values for perfoming string operations or string comparision.

```
dataf = [] #creating seperate list to append only values from dataframe

for i in data.values:
  dataf.append(i)
```

5. Now create new list again to seperate common named entries to use for comparision using levenshtein distance.

```
similar = [] #creating empty list to add and store similar named entries.
for c in range(len(dataf)):
  if c<len(dataf)-1:
    if lev(dataf[c][0],dataf[c+1][0])<5: #Measuring levenshtein distance if less than 5.
      similar.extend([dataf[c][0],dataf[c][1],dataf[c][2],dataf[c][3]]) #adding them into similar list

      similar.extend([dataf[c+1][0],dataf[c+1][1],dataf[c+1][2],dataf[c+1][3]]) #adding them into similar list

similar #displaying list after measuring levenshtein distance
```

```
['PWMPzEhgQSIJFtLl',
 12.864532497107326,
 77.73143549579069,
 0,
 'PWMPzEhgQSIJFtL',
 12.865321165438637,
 77.7314729992191,
 0,
 'lqKiDFBZBTWUez',
 12.983261295304194,
 77.67860107706478,
 0,
 'lqKiDFBZBTWXUez',
 12.98372612827221,
 77.67840867602989,
 0,
 'sjislLxiXZmXLXrA',
 12.873688787580445,
 77.50616775642054,
 0,
 'sjislLxiXZmXLXr',
 12.874079493237168,
 77.50552922468972,
 0,
 'tnbuHQPEFCTHbM',
 12.990843895648302,
 77.5440597112632,
 0,
 'KtnbuHQPEFCTHbM',
 12.99117668224228,
 77.54470053564766,
 0,
 'zQDGraifUKPvJXrh',
 12.912486140806315,
 77.7199094108049,
 0,
 'zQDGraifUKPvJXr',
 12.912090285613802,
 77.72061060421989,
 0,
```

```
      'IiUaIKLepwZKsK',
      12.910422441231065,
      77.59244420713948,
      0,
      'IiUaIKLepwZlKsK',
      12.91128729120406,
      77.59196143907485,
      0,
      'AHzXGncOYekvbc',
      12.906721488679834,
      77.703402737544,
      0,
      'AHzXGncOYekMvbc',
      12.905894120524152,
      77.70314403967068,
      0,
      'DkRreiaFouYeugQi',
      13.028659386729384,
```

6. Finding or measuring distance between entries and classifying them to 0 or 1. If the distance between them is within 200 meters then it is updated as 1 if not it will remain same.

```
from geopy.distance import geodesic #importing required module to find distance between entries

for a in range(0,len(data)):
  if a<len(data)-1:
    newport_ri = (data.values[a][1], data.values[a][2])
    cleveland_oh = (data.values[a+1][1],data.values[a+1][2])
    re = str(geodesic(newport_ri, cleveland_oh).meters) #calcuting distance in meters and converting into string
    fre = float(re) #coverting again into float to check.
    if fre<200: #checking whether the distance is within 200 meters or not.
      if data.values[a][0] in similar and data.values[a+1][0] in similar: #checking if it is within 200 meters and names are als
        data.iloc[a,[3]] = 1 #updating value of similar named entry and which has distance within 200 meters.
        data.iloc[a+1,[3]]=1 #updating value of similar named entry and which has distance within 200 meters.


  else:
    break # if not breaking the loop because not needed
```

7. Finally, converting the result dataframe into csv which is our output csv file.

```
output = data.to_csv('output_data.csv', index = False) #creating output csv file after results
```

8. To verify result printing first 50 entries from output csv file whether is_similar is updated or not.

```
udf = p.read_csv("output_data.csv") #verifying output_data.csv whether updation success or not

udf.head(50) #printing first 50 entries from output_data.csv
```

	name	latitude	longitude	is_similar
0	kbyJYJzqEVkEmpc	13.022860	77.688911	0
1	qbgufTPzZEkxNHV	13.093675	77.702779	0
2	XWIXkuyDacCmsaE	12.902879	77.633207	0
3	PWMPzEhgQSIJFtLl	12.864532	77.731435	1
4	PWMPzEhgQSIJFtL	12.865321	77.731473	1
5	MHveToenbpcMPeW	13.070456	77.610955	0
6	WItnvWIntfoiilf	13.002356	77.683548	0
7	DPjNIjFDGUdRaNP	12.982655	77.593377	0
8	SMLuzwDVgJxZDJm	13.065282	77.472500	0
9	vAbbuCYbFLCXICj	12.990463	77.672949	0
10	rLfczOfLhEoRnGZ	12.909938	77.685575	0
11	edieHBeImKgBQAJ	12.938959	77.465870	0
12	GOwLhWmXEfYOmVX	12.901291	77.673744	0
13	orOQKFbRAoHeqcJ	12.889252	77.596945	0
14	CdWAxHeYVjKBRoq	13.093738	77.713078	0
15	lqKiDFBZBTWUez	12.983261	77.678601	1
16	lqKiDFBZBTWXUez	12.983726	77.678409	1
17	eNmJDTcGphYUOiL	12.901054	77.728352	0
18	sjislLxiXZmXLXrA	12.873689	77.506168	1
19	sjislLxiXZmXLXr	12.874079	77.505529	1
20	PMiOQVrAkdXaIku	12.999143	77.624271	0
21	tnbuHQPEFCTHbM	12.990844	77.544060	1
22	KtnbuHQPEFCTHbM	12.991177	77.544701	1
23	XLAftyVglwWYsGW	12.912192	77.739024	0
24	rZxvAzqBqmxjOvp	13.082419	77.497992	0
25	zQDGraifUKPvJXrh	12.912486	77.719909	1
26	zQDGraifUKPvJXr	12.912090	77.720611	1
27	nDUUMkFmhvWSeKA	12.999386	77.524471	0
28	pTSAacswfCkLTrH	12.878809	77.590372	0
29	LnDIkeWZGiDzBbG	12.951540	77.472948	0
30	ChJfRzoluYXLLJA	12.888275	77.604594	0
31	SwTyWHAtgBNNwyO	12.983947	77.652708	0
32	kpkmIKHoUVIufgD	12.988933	77.613589	0
33	IcpEXtdzmmRLCWr	12.878666	77.583886	0
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
37	QFXbeahTXokzxXE	12.865956	77.738232	0
38	IcDAjGOxTMfSHPF	13.009244	77.472785	0
39	TVAaGCcKLfPKBSc	13.038365	77.651888	0
40	ZqYiEiXtnNdnMQd	12.890397	77.511787	0
41	skziWqCsJpYkKUr	13.050540	77.564465	0
42	EzXfmMXrcmOPODa	13.094561	77.701236	0
43	osLsWAOQUdGDZYQ	12.996804	77.676888	0
44	xaxGBTpqceCpeQo	13.082418	77.489562	0
45	vowfjrCAhFjsuFU	13.094635	77.675350	0
46	sTPYKDcRDklHquz	13.060097	77.529280	0
47	hERpHEmvMnatBqq	12.957687	77.449052	0
48	lwzJQhDNjFeZaGu	13.083994	77.643859	0

49	ZtGYasvPbFraLKt	12.941267	77.456080	0

✓ 0s completed at 11:54 AM