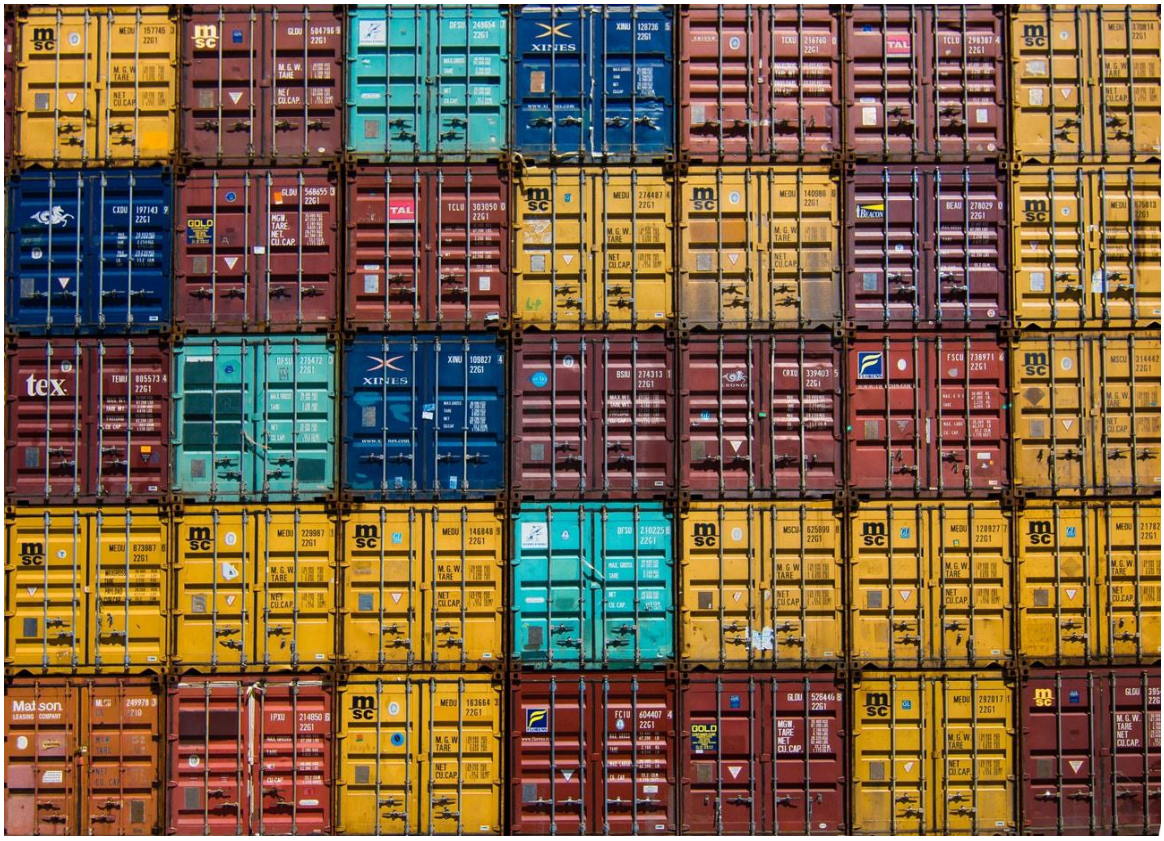
MScBA

Mining Big Datasets – Assignment I

Vogiatzis Georgios-p2821827

Prasinos Michail -p2821823



Part 1

We have used Python programming language for this assignment and we have imported the csv file as a pandas dataframe and dropped the column “Class” as instructed, in order to proceed with the following tasks.

The first 5 rows of our dataset are as follows:

	Age	Job	Marital	Education	Default	Balance	Housing	Loan
0	44	technician	single	secondary	no	29	yes	no
1	33	entrepreneur	married	secondary	no	2	yes	yes
2	35	management	married	tertiary	no	231	yes	no
3	28	management	single	tertiary	no	447	yes	yes
4	42	entrepreneur	divorced	tertiary	yes	2	yes	no

Part 2

Having a first look at our variables, we need to apply necessary transformations to our ordinal variable which is “Education”. Our variable has 3 levels which are transformed for Primary, Secondary & Tertiary to [1,2,3] respectively.

```
secondary    23131
tertiary     13261
primary       6800
Name: Education, dtype: int64
```

```
2    23131
3    13261
1     6800
Name: Education, dtype: int64
```

In order to proceed, we need to define variables which are necessary for the creation of the function that calculates dissimilarity.

These variables are the maximum and minimum rank for the ordinal variable as well as the max & min values for variables Age & Balance.

Our function calculates dissimilarity for any 2 given inputs, based on the dissimilarity of every variable of our dataframe and sets the id = id – 2, in order for our variables to correspond to the id (line) of the csv file.

Dissimilarity is calculated as shown below for every variable category:

Ordinal Variables (Education)

d(a,b) = |rank(a)- rank(b)|/(max_rank – min_rank)

Categorical Variables (Job, Marital, Default, Housing ,Loan)

d(a,b) = 0 if a and b are identical

d(a,b) = 1 if a and b are unalike

Numerical Variables (Age,Balance)

d(a,b) = |a-b|/(max_value – min_value)

We have also created a version of the function for calculating dissimilarity with weights for each variable based on importance criteria of the bank:

$w_{age}=2$, $w_{balance}=3$, $w_{job}=2$, $w_{marital}=1$, $w_{loan}=3$, $w_{housing}=3$, $w_{education}=1$, $w_{default}=3$

After the calculation of the dissimilarity for each variable the dissimilarity between 2 different customers, the total dissimilarity between them is the given as the output of our function and it is calculated as:

$d(a,b)= (d_{age}+ d_{balance}+ d_{job}+ d_{marital}+ d_{loan}+ d_{housing}+ d_{education}+ d_{default})/8$ *without weights*

and as:

$d(a,b)= (w_{age} * d_{age}+ w_{balance} * d_{balance}+ w_{job} * d_{job}+ w_{marital} * d_{marital}+ w_{loan} * d_{loan}+ w_{housing} * d_{housing}+ w_{education} * d_{education}+ w_{default} * d_{default})/18$ *with weights.*

```
#example dissimilarity between id=2 and id=5204
calcDissim(2,5204,df)
```

```
Out[8]: 0.210762925406444
```

```
#example weighted dissimilarity between id=2 and id=5204
calcDissimWeighted(2,5204,df)
```

```
Out[9]: 0.1598051126631375
```

Part 3

For the last part of our assignment we created a function that takes as input an id (# of the line on the csv) and calculates the 10 nearest neighbors of this customer, excluding himself. To achieve that, we had to use our previous defined function. We loop through the dataframe and we apply the dissimilarity function between the given id and each line, producing a list. Then we sort the list by value assending and we slice the first 10 results. Results shown below:

```
#examples for ids=1230,5032,10001,24035,28948,35099,37693,39543,40002,42192
outList1230 = calc10nn(1230,df)
outList5032 = calc10nn(5032,df)
outList10001 = calc10nn(10001,df)
outList24035 = calc10nn(24035,df)
outList28948 = calc10nn(28948,df)
outList35099 = calc10nn(35099,df)
outList37693 = calc10nn(37693,df)
outList39543 = calc10nn(39543,df)
outList40002 = calc10nn(40002,df)
outList42192 = calc10nn(42192,df)

# formatting for our list
f = '{}|{}'

#print list using defined format
for i in outList1230:
    print(f.format(*i))
for i in outList5032:
    print(f.format(*i))
for i in outList10001:
    print(f.format(*i))
for i in outList24035:
    print(f.format(*i))
for i in outList28948:
    print(f.format(*i))
for i in outList35099:
    print(f.format(*i))
for i in outList37693:
    print(f.format(*i))
for i in outList39543:
    print(f.format(*i))
for i in outList40002:
    print(f.format(*i))
for i in outList42192:
    print(f.format(*i))
```

```
#examples for ids=1230,5032,10001,24035,28948,35099,37693,39543,40002,42192 weighted
woutList1230 = calc10nnWeighted(1230,df)
woutList5032 = calc10nnWeighted(5032,df)
woutList10001 = calc10nnWeighted(10001,df)
woutList24035 = calc10nnWeighted(24035,df)
woutList28948 = calc10nnWeighted(28948,df)
woutList35099 = calc10nnWeighted(35099,df)
woutList37693 = calc10nnWeighted(37693,df)
woutList39543 = calc10nnWeighted(39543,df)
woutList40002 = calc10nnWeighted(40002,df)
woutList42192= calc10nnWeighted(42192,df)



#print list(wighted results) using defined format
for i in woutList1230:
    print(f.format(*i))
for i in woutList5032:
    print(f.format(*i))
for i in woutList10001:
    print(f.format(*i))
for i in woutList24035:
    print(f.format(*i))
for i in woutList28948:
    print(f.format(*i))
for i in woutList35099:
    print(f.format(*i))
for i in woutList37693:
    print(f.format(*i))
for i in woutList39543:
    print(f.format(*i))
for i in woutList40002:
    print(f.format(*i))
for i in woutList42192:
    print(f.format(*i))
```

UNWEIGHTED RESULTS:

id=1230

id	dissimilarity
14324	0.0013606939879795908
986	0.0018021535053474478
19027	0.001974651825758539
34966	0.001974651825758539
5461	0.0020472826975105767
25334	0.002271984456993445
29236	0.002683333408007931
41017	0.003264380382024237
13359	0.003302965532642507
868	0.0035378810084655057

id= 5032

id	dissimilarity
14406	5.674286855627985e-06
23663	5.674286855627985e-06
23836	7.944001597879178e-06
32811	7.944001597879178e-06
36171	7.944001597879178e-06
17137	1.2483431082381567e-05
1292	5.901258329853104e-05
40236	8.397944546329417e-05
28583	0.00015207088773082998
13666	0.00016568917618433714

id=10001

id	dissimilarity
28869	0.0
32585	0.0
33807	7.944001597879178e-06
22104	1.9292575309135147e-05
21333	2.950629164926552e-05
30922	3.5180578504893504e-05
38656	3.5180578504893504e-05
21184	4.0854865360521486e-05
20932	4.6529152216149476e-05
39806	4.766400958727507e-05

id=24035

id	dissimilarity
30976	0.0011972745265375047
7206	0.0017448063620870624
37871	0.0026765242637811775
21431	0.0028178508525048573
4695	0.0028206511499141284
4731	0.0028984257258547748
33571	0.0031912189276051784
25427	0.0033426855406319027
14417	0.0034306001408755935
24474	0.0034306001408755935

id= 28948

id|dissimilarity
9906|6.468687015415902e-05
10458|0.0001498011729885788
25341|0.0009112904690138544
25366|0.0016937377803864105
11059|0.0016971423524997872
26179|0.0016971423524997872
42389|0.002331527622958996
29504|0.003013576903005 48
21834|0.003263776107320131
28769|0.003280798967887015

id= 35099

id|dissimilarity
124|1.134857371125597e-06
20771|1.134857371125597e-06
35299|1.134857371125597e-06
1163|1.2483431082381567e-05
20857|2.3832004793637534e-05
23972|3.291086376264231e-05
31505|3.291086376264231e-05
37400|6.355201278303342e-05
20920|0.00013277831242169483
33724|0.0001475314582463276

id=37693

id|dissimilarity
17820|0.0003972000798939589
353|0.0005356526791712818
5565|0.0005356526791712818
31021|0.0005515406823670401
34866|0.0008431990267463185
21760|0.0008863236068490912
30805|0.0008863236068490912
37219|0.0010837887894249451
5841|0.0011428013727234761
4827|0.0012665008261761662

id= 39543

id|dissimilarity
34896|0.0
7631|9.078858969004775e-05
30305|0.0007671635828809036
5623|0.0014957420151435368
30337|0.0014957420151435368
36782|0.0019331926856939114
8878|0.001970112396274036
34578|0.002055226699108456
27395|0.0022821981733335754
10680|0.0023151090370962177

id= 40002

id|dissimilarity

42350|0.0
34231|0.0029267971601329144
285|0.004550173783509538
36292|0.004639827515828461
1682|0.006263204139205084
9063|0.00969320000537657
39727|0.010255484986750761
156|0.012174528801324146
11507|0.014290433523769278
9325|0.015340707174727476

id= 42192

id|dissimilarity
18791|8.057487334991739e-05
38755|8.624916020554537e-05
36919|0.00019633032520472826
14546|0.0002666914822145153
24260|0.0002666914822145153
30687|0.0005935304050986872
9078|0.0006933978537577397
24345|0.0008148275924681786
5924|0.0008885933215913424
20965|0.0009328527590652406

WEIGHTED RESULTS:

id=1230

id|w.dissimilarity
14324|0.0018142586506394542
986|0.002402871340463264
19027|0.002632869101011385
34966|0.002632869101011385
5461|0.0027297102633474356
29236|0.0028562771558431864
25334|0.0030293126093245935
42094|0.003442839569435179
41017|0.003631006454531594
13359|0.003682453322022621

id= 5032

id|w.dissimilarity
14406|7.56571580750398e-06
23663|7.56571580750398e-06
23836|1.0592002130505571e-05
32811|1.0592002130505571e-05
36171|1.0592002130505571e-05
17137|1.6644574776508756e-05
1292|7.868344439804139e-05
40236|0.0001119725939510589
28583|0.00020276118364110664
13666|0.00022091890157911617

id=10001

id	w.dissimilarity
28869	0.0
32585	0.0
33807	1.0592002130505571e-05
22104	2.572343374551353e-05
21333	3.9341722199020694e-05
30922	4.690743800652467e-05
38656	4.690743800652467e-05
21184	5.4473153814028644e-05
20932	6.203886962153264e-05
39806	6.355201278303342e-05

id=24035

id	w.dissimilarity
30976	0.0015963660353833396
7206	0.0016049077612820283
37871	0.002847198296874182
4695	0.0030393674783847826
14417	0.0031311320781660153
24474	0.0031311320781660153
25427	0.003735413332675149
21431	0.0037571344700064765
4731	0.003864567634473033
8045	0.00390791165308624

id= 28948

id	w.dissimilarity
9906	8.624916020554537e-05
10458	0.00019973489731810507
25341	0.0012150539586851393
25366	0.0015368163190144924
11059	0.0015413557484989949
26179	0.0015413557484989949
42389	0.00238720277577794
21834	0.002908700033425398
28769	0.00293139718084791
4365	0.0029904097641464413

id= 35099

id	w.dissimilarity
124	1.513143161500796e-06
20771	1.513143161500796e-06
35299	1.513143161500796e-06
1163	1.6644574776508756e-05
20857	3.177600639151671e-05
23972	4.3881151683523085e-05
31505	4.3881151683523085e-05
37400	8.473601704404457e-05
20920	0.0001770377498955931
33724	0.00019670861099510347

id=37693

id|w.dissimilarity
17820|0.0005296001065252785
353|0.0007142035722283756
5565|0.0007142035722283756
31021|0.0007353875764893868
34866|0.0011242653689950914
21760|0.0011817648091321216
30805|0.0011817648091321216
37219|0.0014450517192332601
5841|0.0015237351636313017
4827|0.0016886677682348883

id= 39543

id|w.dissimilarity
34896|0.0
7631|0.00012105145292006368
30305|0.001022884777174538
36782|0.0018560895260911604
5623|0.001994322686858049
30337|0.001994322686858049
8878|0.0026268165283653815
34578|0.0027403022654779413
34627|0.0029803548950862514
2088|0.0030388303453144664

id= 40002

id|w.dissimilarity
42350|0.0
34231|0.003902396213510553
285|0.005345397656511996
36292|0.005464935966270558
1682|0.006907937409272002
9063|0.009316763066331819
39727|0.009344975653330019
156|0.011903700739427864
11507|0.014003406314520652
9325|0.014682270460964196

id= 42192

id|w.dissimilarity
18791|0.00010743316446655652
38755|0.00011499888027406048
36919|0.0002617737669396377
14546|0.000355588642952687
24260|0.000355588642952687
30687|0.0007913738734649162
9078|0.0009245304716769862

24345|0.0010864367899575717
5924|0.0011847910954551231
20965|0.0012438036787536541