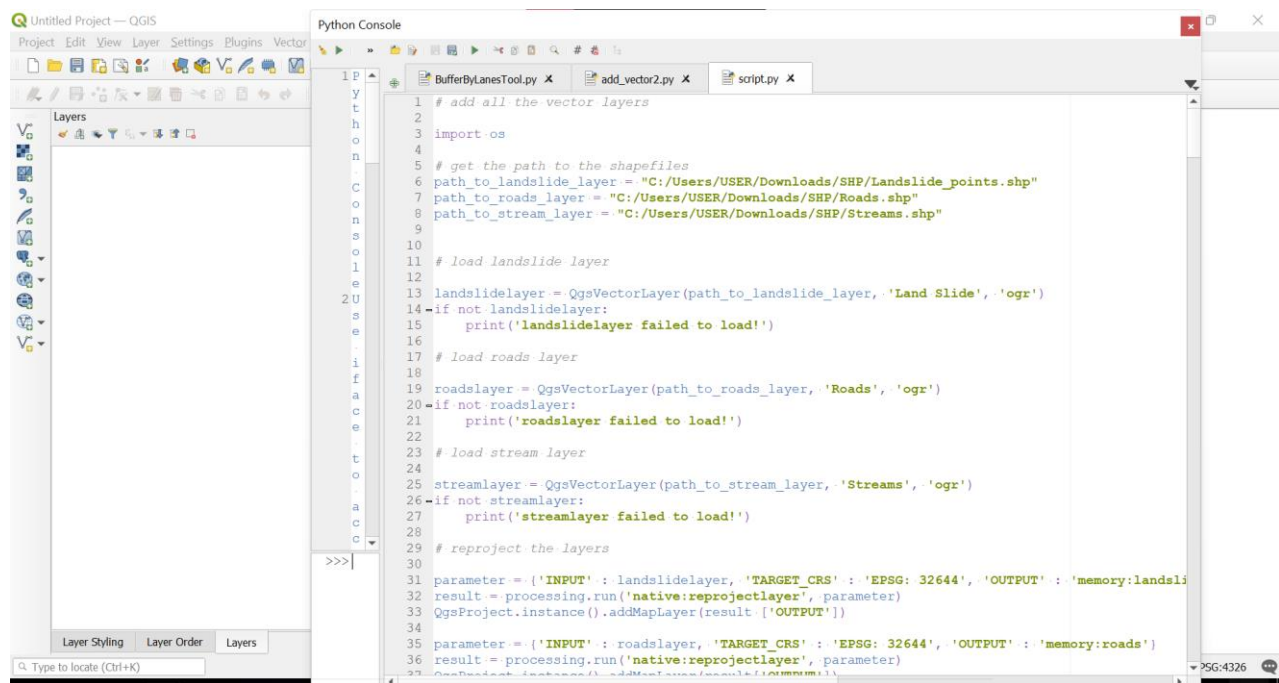


# APPLICATION OF SPATIAL DATA TO IDENTIFY ROAD RISKS IN NUWARA-ELIYA, AMBAGAMUWA AREA, SRI LANKA

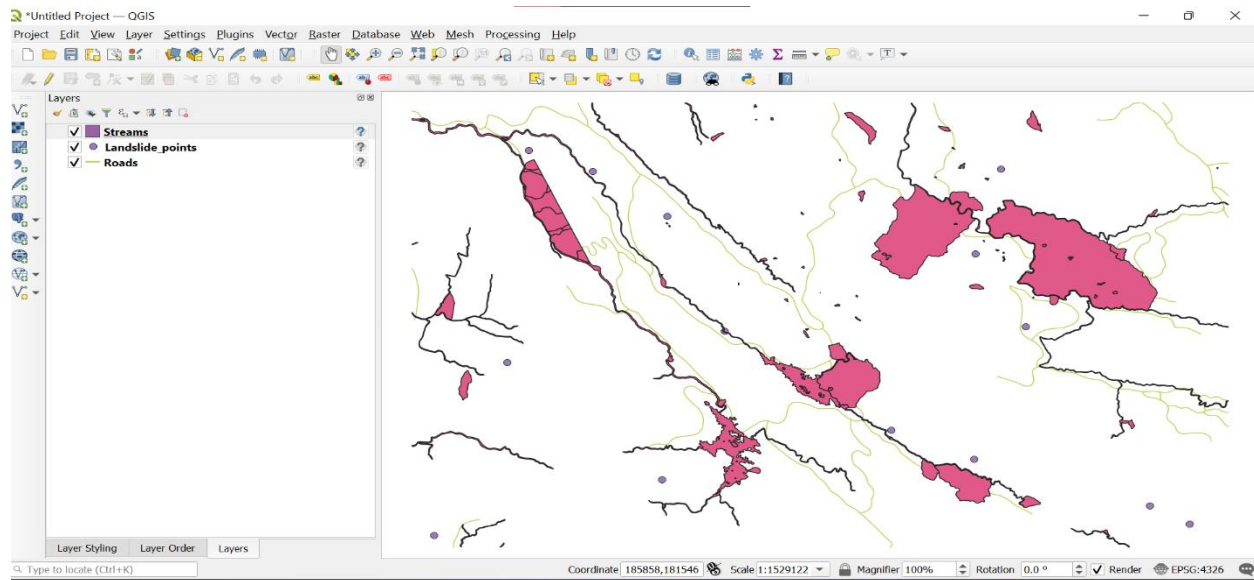
Screenshots of the processes are as follows:

- i. Import the data into QGIS environment using python snippet as shown below



The screenshot shows the QGIS interface with the Python Console open. The console displays a Python script that loads three vector layers (Landslide points, Roads, and Streams) and reprojects them to a common CRS (EPSG: 32644). The script uses the QGIS API to load the layers and the processing module to reproject them. The Layers panel on the left is empty, and the Project panel at the top shows the current project settings.

```
1 # add all the vector layers
2
3 import os
4
5 # get the path to the shapefiles
6 path_to_landslide_layer = "C:/Users/USER/Downloads/SHP/Landslide_points.shp"
7 path_to_roads_layer = "C:/Users/USER/Downloads/SHP/Roads.shp"
8 path_to_stream_layer = "C:/Users/USER/Downloads/SHP/Streams.shp"
9
10
11 # load landslide layer
12
13 landslidelayer = QgsVectorLayer(path_to_landslide_layer, 'Land Slide', 'ogr')
14 if not landslidelayer:
15     print('landslidelayer failed to load!')
16
17 # load roads layer
18
19 roadslayer = QgsVectorLayer(path_to_roads_layer, 'Roads', 'ogr')
20 if not roadslayer:
21     print('roadslayer failed to load!')
22
23 # load stream layer
24
25 streamlayer = QgsVectorLayer(path_to_stream_layer, 'Streams', 'ogr')
26 if not streamlayer:
27     print('streamlayer failed to load!')
28
29 # reproject the layers
30
31 parameter = {'INPUT': landslidelayer, 'TARGET_CRS': 'EPSG: 32644', 'OUTPUT': 'memory:landslide'}
32 result = processing.run('native:reprojectlayer', parameter)
33 QgsProject.instance().addMapLayer(result['OUTPUT'])
34
35 parameter = {'INPUT': roadslayer, 'TARGET_CRS': 'EPSG: 32644', 'OUTPUT': 'memory:roads'}
36 result = processing.run('native:reprojectlayer', parameter)
37 QgsProject.instance().addMapLayer(result['OUTPUT'])
```



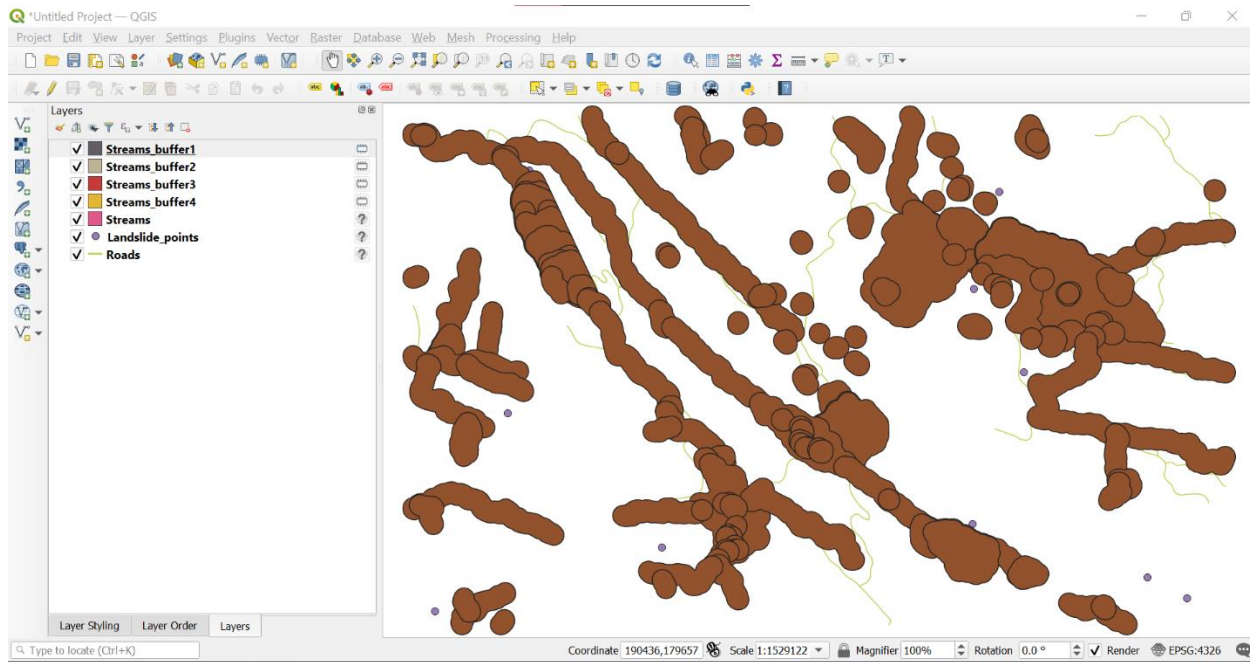
The result the algorithm when run.

- ii. The next procedure is to buffer the Landslide points and stream. The algorithm used is shown below with the resulting buffer

The screenshot shows the QGIS interface with the Python Console open. The console displays a script that performs two buffering operations. The first operation buffers the 'Streams' layer with a distance of 100 and saves the result as 'Streams\_buffer4.shp'. The second operation buffers the 'Streams' layer with a distance of 200 and saves the result as 'Streams\_buffer3.shp'. The script uses the QGIS API to access the project instance, map layers, and write the buffered geometries to shapefiles.

```

1 Python Console
2
3 BufferByLanesTool.py x add_vector2.py x script.py x
4
5 1 # stream buffer for weight 4
6 2 layerName = 'Streams'
7 3 outputFile = "C:/Users/USER/Downloads/SHP/Streams_buffer4.shp"
8 4 bufferDist = 100
9 5
10 6
11 7 layers = QgsProject.instance().mapLayersByName(layerName)
12 8 layer = layers[0]
13 9 fields = layer.fields()
14 10 feats = layer.getFeatures()
15 11
16 12 writer = QgsVectorFileWriter(outputFile, 'UTF-8', fields, \
17 13 QgsWkbTypes.Polygon, \
18 14 layer.sourceCrs(), 'ESRI Shapefile')
19 15
20 16 for i in feats:
21 17     geom = i.geometry()
22 18     buff = geom.buffer(bufferDist, 4)
23 19     i.setGeometry(buff)
24 20     writer.addFeature(i)
25 21
26 22 iface.addVectorLayer(outputFile, '', 'ogr')
27 23
28 24
29 25 # stream buffer for weight 3
30 26 layerName = 'Streams'
31 27 outputFile = "C:/Users/USER/Downloads/SHP/Streams_buffer3.shp"
32 28 bufferDist = 200
33 29
34 30
35 31 layers = QgsProject.instance().mapLayersByName(layerName)
36 32 layer = layers[0]
37 33 fields = layer.fields()
38 34 feats = layer.getFeatures()
39 35
40 36 writer = QgsVectorFileWriter(outputFile, 'UTF-8', fields, \
41 37 QgsWkbTypes.Polygon, \
42 38 layer.sourceCrs(), 'ESRI Shapefile')
43 39
44 40
45 41
46 42
47 43
48 44
49 45
50 46
51 47
52 48
53 49
54 50
55 51
56 52
57 53
58 54
59 55
60 56
61 57
62 58
63 59
64 60
65 61
66 62
67 63
68 64
69 65
70 66
71 67
72 68
73 69
74 70
75 71
76 72
77 73
78 74
79 75
80 76
81 77
82 78
83 79
84 80
85 81
86 82
87 83
88 84
89 85
90 86
91 87
92 88
93 89
94 90
95 91
96 92
97 93
98 94
99 95
100 96
101 97
102 98
103 99
104 100
105 101
106 102
107 103
108 104
109 105
110 106
111 107
112 108
113 109
114 110
115 111
116 112
117 113
118 114
119 115
120 116
121 117
122 118
123 119
124 120
125 121
126 122
127 123
128 124
129 125
130 126
131 127
132 128
133 129
134 130
135 131
136 132
137 133
138 134
139 135
140 136
141 137
142 138
143 139
144 140
145 141
146 142
147 143
148 144
149 145
150 146
151 147
152 148
153 149
154 150
155 151
156 152
157 153
158 154
159 155
160 156
161 157
162 158
163 159
164 160
165 161
166 162
167 163
168 164
169 165
170 166
171 167
172 168
173 169
174 170
175 171
176 172
177 173
178 174
179 175
180 176
181 177
182 178
183 179
184 180
185 181
186 182
187 183
188 184
189 185
190 186
191 187
192 188
193 189
194 190
195 191
196 192
197 193
198 194
199 195
200 196
201 197
202 198
203 199
204 200
205 201
206 202
207 203
208 204
209 205
210 206
211 207
212 208
213 209
214 210
215 211
216 212
217 213
218 214
219 215
220 216
221 217
222 218
223 219
224 220
225 221
226 222
227 223
228 224
229 225
230 226
231 227
232 228
233 229
234 230
235 231
236 232
237 233
238 234
239 235
240 236
241 237
242 238
243 239
244 240
245 241
246 242
247 243
248 244
249 245
250 246
251 247
252 248
253 249
254 250
255 251
256 252
257 253
258 254
259 255
260 256
261 257
262 258
263 259
264 260
265 261
266 262
267 263
268 264
269 265
270 266
271 267
272 268
273 269
274 270
275 271
276 272
277 273
278 274
279 275
280 276
281 277
282 278
283 279
284 280
285 281
286 282
287 283
288 284
289 285
290 286
291 287
292 288
293 289
294 290
295 291
296 292
297 293
298 294
299 295
300 296
301 297
302 298
303 299
304 300
305 301
306 302
307 303
308 304
309 305
310 306
311 307
312 308
313 309
314 310
315 311
316 312
317 313
318 314
319 315
320 316
321 317
322 318
323 319
324 320
325 321
326 322
327 323
328 324
329 325
330 326
331 327
332 328
333 329
334 330
335 331
336 332
337 333
338 334
339 335
340 336
341 337
342 338
343 339
344 340
345 341
346 342
347 343
348 344
349 345
350 346
351 347
352 348
353 349
354 350
355 351
356 352
357 353
358 354
359 355
360 356
361 357
362 358
363 359
364 360
365 361
366 362
367 363
368 364
369 365
370 366
371 367
372 368
373 369
374 370
375 371
376 372
377 373
378 374
379 375
380 376
381 377
382 378
383 379
384 380
385 381
386 382
387 383
388 384
389 385
390 386
391 387
392 388
393 389
394 390
395 391
396 392
397 393
398 394
399 395
400 396
401 397
402 398
403 399
404 400
405 401
406 402
407 403
408 404
409 405
410 406
411 407
412 408
413 409
414 410
415 411
416 412
417 413
418 414
419 415
420 416
421 417
422 418
423 419
424 420
425 421
426 422
427 423
428 424
429 425
430 426
431 427
432 428
433 429
434 430
435 431
436 432
437 433
438 434
439 435
440 436
441 437
442 438
443 439
444 440
445 441
446 442
447 443
448 444
449 445
450 446
451 447
452 448
453 449
454 450
455 451
456 452
457 453
458 454
459 455
460 456
461 457
462 458
463 459
464 460
465 461
466 462
467 463
468 464
469 465
470 466
471 467
472 468
473 469
474 470
475 471
476 472
477 473
478 474
479 475
480 476
481 477
482 478
483 479
484 480
485 481
486 482
487 483
488 484
489 485
490 486
491 487
492 488
493 489
494 490
495 491
496 492
497 493
498 494
499 495
500 496
501 497
502 498
503 499
504 500
505 501
506 502
507 503
508 504
509 505
510 506
511 507
512 508
513 509
514 510
515 511
516 512
517 513
518 514
519 515
520 516
521 517
522 518
523 519
524 520
525 521
526 522
527 523
528 524
529 525
530 526
531 527
532 528
533 529
534 530
535 531
536 532
537 533
538 534
539 535
540 536
541 537
542 538
543 539
544 540
545 541
546 542
547 543
548 544
549 545
550 546
551 547
552 548
553 549
554 550
555 551
556 552
557 553
558 554
559 555
560 556
561 557
562 558
563 559
564 560
565 561
566 562
567 563
568 564
569 565
570 566
571 567
572 568
573 569
574 570
575 571
576 572
577 573
578 574
579 575
580 576
581 577
582 578
583 579
584 580
585 581
586 582
587 583
588 584
589 585
590 586
591 587
592 588
593 589
594 590
595 591
596 592
597 593
598 594
599 595
600 596
601 597
602 598
603 599
604 600
605 601
606 602
607 603
608 604
609 605
610 606
611 607
612 608
613 609
614 610
615 611
616 612
617 613
618 614
619 615
620 616
621 617
622 618
623 619
624 620
625 621
626 622
627 623
628 624
629 625
630 626
631 627
632 628
633 629
634 630
635 631
636 632
637 633
638 634
639 635
640 636
641 637
642 638
643 639
644 640
645 641
646 642
647 643
648 644
649 645
650 646
651 647
652 648
653 649
654 650
655 651
656 652
657 653
658 654
659 655
660 656
661 657
662 658
663 659
664 660
665 661
666 662
667 663
668 664
669 665
670 666
671 667
672 668
673 669
674 670
675 671
676 672
677 673
678 674
679 675
680 676
681 677
682 678
683 679
684 680
685 681
686 682
687 683
688 684
689 685
690 686
691 687
692 688
693 689
694 690
695 691
696 692
697 693
698 694
699 695
700 696
701 697
702 698
703 699
704 700
705 701
706 702
707 703
708 704
709 705
710 706
711 707
712 708
713 709
714 710
715 711
716 712
717 713
718 714
719 715
720 716
721 717
722 718
723 719
724 720
725 721
726 722
727 723
728 724
729 725
730 726
731 727
732 728
733 729
734 730
735 731
736 732
737 733
738 734
739 735
740 736
741 737
742 738
743 739
744 740
745 741
746 742
747 743
748 744
749 745
750 746
751 747
752 748
753 749
754 750
755 751
756 752
757 753
758 754
759 755
760 756
761 757
762 758
763 759
764 760
765 761
766 762
767 763
768 764
769 765
770 766
771 767
772 768
773 769
774 770
775 771
776 772
777 773
778 774
779 775
780 776
781 777
782 778
783 779
784 780
785 781
786 782
787 783
788 784
789 785
790 786
791 787
792 788
793 789
794 790
795 791
796 792
797 793
798 794
799 795
800 796
801 797
802 798
803 799
804 800
805 801
806 802
807 803
808 804
809 805
810 806
811 807
812 808
813 809
814 810
815 811
816 812
817 813
818 814
819 815
820 816
821 817
822 818
823 819
824 820
825 821
826 822
827 823
828 824
829 825
830 826
831 827
832 828
833 829
834 830
835 831
836 832
837 833
838 834
839 835
840 836
841 837
842 838
843 839
844 840
845 841
846 842
847 843
848 844
849 845
850 846
851 847
852 848
853 849
854 850
855 851
856 852
857 853
858 854
859 855
860 856
861 857
862 858
863 859
864 860
865 861
866 862
867 863
868 864
869 865
870 866
871 867
872 868
873 869
874 870
875 871
876 872
877 873
878 874
879 875
880 876
881 877
882 878
883 879
884 880
885 881
886 882
887 883
888 884
889 885
890 886
891 887
892 888
893 889
894 890
895 891
896 892
897 893
898 894
899 895
900 896
901 897
902 898
903 899
904 900
905 901
906 902
907 903
908 904
909 905
910 906
911 907
912 908
913 909
914 910
915 911
916 912
917 913
918 914
919 915
920 916
921 917
922 918
923 919
924 920
925 921
926 922
927 923
928 924
929 925
930 926
931 927
932 928
933 929
934 930
935 931
936 932
937 933
938 934
939 935
940 936
941 937
942 938
943 939
944 940
945 941
946 942
947 943
948 944
949 945
950 946
951 947
952 948
953 949
954 950
955 951
956 952
957 953
958 954
959 955
960 956
961 957
962 958
963 959
964 960
965 961
966 962
967 963
968 964
969 965
970 966
971 967
972 968
973 969
974 970
975 971
976 972
977 973
978 974
979 975
980 976
981 977
982 978
983 979
984 980
985 981
986 982
987 983
988 984
989 985
990 986
991 987
992 988
993 989
994 990
995 991
996 992
997 993
998 994
999 995
1000 996
1001 997
1002 998
1003 999
1004 1000
1005 1001
1006 1002
1007 1003
1008 1004
1009 1005
1010 1006
1011 1007
1012 1008
1013 1009
1014 1010
1015 1011
1016 1012
1017 1013
1018 1014
1019 1015
1020 1016
1021 1017
1022 1018
1023 1019
1024 1020
1025 1021
1026 1022
1027 1023
1028 1024
1029 1025
1030 1026
1031 1027
1032 1028
1033 1029
1034 1030
1035 1031
1036 1032
1037 1033
1038 1034
1039 1035
1040 1036
1041 1037
1042 1038
1043 1039
1044 1040
1045 1041
1046 1042
1047 1043
1048 1044
1049 1045
1050 1046
1051 1047
1052 1048
1053 1049
1054 1050
1055 1051
1056 1052
1057 1053
1058 1054
1059 1055
1060 1056
1061 1057
1062 1058
1063 1059
1064 1060
1065 1061
1066 1062
1067 1063
1068 1064
1069 1065
1070 1066
1071 1067
1072 1068
1073 1069
1074 1070
1075 1071
1076 1072
1077 1073
1078 1074
1079 1075
1080 1076
1081 1077
1082 1078
1083 1079
1084 1080
1085 1081
1086 1082
1087 1083
1088 1084
1089 1085
1090 1086
1091 1087
1092 1088
1093 1089
1094 1090
1095 1091
1096 1092
1097 1093
1098 1094
1099 1095
1100 1096
1101 1097
1102 1098
1103 1099
1104 1100
1105 1101
1106 1102
1107 1103
1108 1104
1109 1105
1110 1106
1111 1107
1112 1108
1113 1109
1114 1110
1115 1111
1116 1112
1117 1113
1118 1114
1119 1115
1120 1116
1121 1117
1122 1118
1123 1119
1124 1120
1125 1121
1126 1122
1127 1123
1128 1124
1129 1125
1130 1126
1131 1127
1132 1128
1133 1129
1134 1130
1135 1131
1136 1132
1137 1133
1138 1134
1139 1135
1140 1136
1141 1137
1142 1138
1143 1139
1144 1140
1145 1141
1146 1142
1147 1143
1148 1144
1149 1145
1150 1146
1151 1147
1152 1148
1153 1149
1154 1150
1155 1151
1156 1152
1157 1153
1158 1154
1159 1155
1160 1156
1161 1157
1162 1158
1163 1159
1164 1160
1165 1161
1166 1162
1167 1163
1168 1164
1169 1165
1170 1166
1171 1167
1172 1168
1173 1169
1174 1170
1175 1171
1176 1172
1177 1173
1178 1174
1179 1175
1180 1176
1181 1177
1182 1178
1183 1179
1184 1180
1185 1181
1186 1182
1187 1183
1188 1184
1189 1185
1190 1186
1191 1187
1192 1188
1193 1189
1194 1190
1195 1191
1196 1192
1197 1193
1198 1194
1199 1195
1200 1196
1201 1197
1202 1198
1203 1199
1204 1200
1205 1201
1206 1202
1207 1203
1208 1204
1209 1205
1210 1206
1211 1207
1212 1208
1213 1209
1214 1210
1215 1211
1216 1212
1217 1213
1218 1214
1219 1215
1220 1216
1221 1217
1222 1218
1223 1219
1224 1220
1225 1221
1226 1222
1227 1223
1228 1224
1229 1225
1230 1226
1231 1227
1232 1228
1233 1229
1234 1230
1235 1231
1236 1232
1237 1233
1238 1234
1239 1235
1240 1236
1241 1237
1242 1238
1243 1239
1244 1240
1245 1241
1246 1242
1247 1243
1248 1244
1249 1245
1250 1246
1251 1247
1252 1248
1253 1249
1254 1250
1255 1251
1256 1252
1257 1253
1258 1254
1259 1255
1260 1256
1261 1257
1262 1258
1263 1259
1264 1260
1265 1261
1266 1262
1267 1263
1268 1264
1269 1265
1270 1266
1271 1267
1272 1268
1273 1269
1274 1270
1275 1271
1276 1272
1277 1273
1278 1274
1279 1275
1280 1276
1281 1277
1282 1278
1283 1279
1284 1280
1285 1281
1286 1282
1287 1283
1288 1284
1289 1285
1290 1286
1291 1287
1292 1288
1293 1289
1294 1290
1295 1291
1296 1292
1297 1293
1298 1294
1299 1295
1300 1296
1301 1297
1302 1298
1303 1299
1304 1300
1305 1301
1306 1302
1307 1303
1308 1304
1309 1305
1310 1306
1311 1307
1312 1308
1313 1309
1314 1310
1315 1311
1316 1312
1317 1313
1318 1314
1319 1315
1320 1316
1321 1317
1322 1318
1323 1319
1324 1320
1325 1321
1326 1322
1327 1323
1328 1324
1329 1325
1330 1326
1331 1327
1332 1328
1333 1329
1334 1330
1335 1331
1336 1332
1337 1333
1338 1334
1339 1335
1340 1336
1341 1337
1342 1338
1343 1339
1344 1340
1345 1341
1346 1342
1347 1343
1348 1344
1349 1345
1350 1346
1351 1347
1352 1348
1353 1349
1354 1350
1355 1351
1356 1352
1357 1353
1358 1354
1359 1355
1360 1356
1361 1357
1362 1358
1363 1359
1364 1360
1365 1361
1366 1362
1367 1363
1368 1364
1369 1365
1370 1366
1371 1367
1372 1368
1373 1369
1374 1370
1375 1371
1376 1372
1377 1373
1378 1374
1379 1375
1380 1376
1381 1377
1382 1378
1383 1379
1384 1380
1385 1381
```



iii. The next step is to clip the corresponding weight assigned to each layer. The algorithm is as shown below.

iv. The next step is map cosmetics, in order to design the cartographically.

The code snippet is shown below

```
Python Console
| p |
| y |
| t |
| h |
| o |
| n |
| c |
| o |
| d |
| e |
| l |
| a |
| c |
| e |
| t |
| o |
| c |
| >>>

294
295 # map cosmetics
296
297 from random import randrange
298
299 # get the active layer (must be a vector layer)
300 layer = qgis.utils.iface.activeLayer()
301
302 # get unique values
303 fni = layer.fieldNameIndex('layer')
304 unique_values = layer.dataProvider().uniqueValues(fni)
305
306 # define categories
307 categories = []
308 for unique_value in unique_values:
309     # initialise the default symbol for this geometry type
310     symbol = QgsSymbolV2.defaultSymbol(layer.geometryType())
311
312     # configure a symbol layer
313     layer_style = {}
314     layer_style['color'] = '%d,%d,%d' % (randrange(0,256), randrange(0,256), randrange(0,256))
315     layer_style['outline'] = '#000000'
316     symbol_layer = QgsSimpleFillSymbolLayerV2.create(layer_style)
317
318     # replace default symbol layer with the configured one
319     if symbol_layer is not None:
320         symbol.changeSymbolLayer(0, symbol_layer)
321
322     # create renderer object
323     category = QgsRendererCategoryV2(unique_value, symbol, str(unique_value))
324     # entry for the list of category items
325     categories.append(category)
326
327 # create renderer object
328 renderer = QgsCategorizedSymbolRendererV2('test', categories)
329
330 # assign the created renderer to the layer
```

```
Python Console
1 Python Console Use iface tool acc
2
BufferByLanesTool.py x add_vector2.py x script.py x
272 geom = i.geometry()
273 buff = geom.buffer(bufferDist, 1)
274 i.setGeometry(buff)
275 writer.addFeature(i)
276
277 iface.addVectorLayer(outputFile, '', 'ogr')
278
279 # clipping of layers
280 import processing
281
282 #set input and output file names
283 polyPath = "C:/temp/result_merge.shp"
284 linePath = "C:/temp/road_merge.shp"
285 clipPath = "C:/temp/clipped.shp"
286
287 #run the clip tool
288 processing.run("native:clip", {'INPUT': linePath, \
289 'OVERLAY': polyPath, \
290 'OUTPUT': clipPath})
291
292 #add output to the qgis interface
293 iface.addVectorLayer(clipPath, '', 'ogr')
294
295 # map cosmetics
296 from random import randrange
297
298
299 # Get the active layer (must be a vector layer)
300 layer = qgis.utils.iface.activeLayer()
301
302 # get unique values
303 fni = layer.fieldNameIndex('layer')
304 unique_values = layer.dataProvider().uniqueValues(fni)
305
306 # define categories
307 categories = []
308 for unique_value in unique_values:
```

After running the code, the map generated is shown below:

