

Codes

1.1 visualization

```

1  #!/usr/bin/env python
2  # coding: utf-8
3
4  # ## 加载数据
5
6  # In[1]:
7
8
9  import geopandas as gpd
10 import pandas as pd
11 food_df_clean = pd.read_csv("../dataset/
    food_preprocessed.csv")
12 gdf = gpd.read_file("../map/food.shp")
13
14
15 # ## 相关性系数图
16
17 # In[ ]:
18
19
20 import altair as alt
21 corrs = (
22     food_df_clean[["点评数", "口味", "环境", "
        服务", "人均消费"]]
23     .loc[food_df_clean["点评数"] > 5]
24     .corr(method="pearson")
25     .stack()
26     .reset_index()
27     .rename(columns={"level_0": "变量1", "
        level_1": "变量2", 0: "相关系数"})
28     .assign(corr_label=lambda x: x['相关系数'
        ].map("{:.2f}".format))
29 )
30 corr_chart = (
31     alt.Chart(
32         corrs
33     )
34     .mark_rect()
35     .encode(
36         x=alt.X("变量1:O", title=None),
37         y=alt.Y("变量2:O", title=None),
38         color="相关系数:Q",
39         tooltip=["相关系数"]
40     )
41     .properties(
42         width=300,
43         height=300
44     )
45 )
46
47 text = (
48     corr_chart
49     .mark_text()
50     .encode(
51         text="corr_label",
52         color=alt.condition(
53             alt.datum.相关系数 > 0.5,
54             alt.value("white"),
55             alt.value("black")
56         )
57     )
58 )
59
60 corr_chart + text
61
62
63 # ## 菜品类别平均排行图
64
65 # In[18]:
66
67
68 import altair as alt
69 alt.data_transformers.disable_max_rows()
70
71 def generate(field, color_scheme):
72
73     bars = (
74         alt.Chart(
75             food_df_clean
76             .groupby("类别")[field]
77             .mean()
78             .reset_index()
79             .assign(label=lambda x: x[field].map("
                {:.2f}".format))
80         )
81         .encode(
82             x=alt.X(f"{{field}}:Q", title=f"平均{
                field}"),
83             y=alt.Y("类别:N", sort="-x"),
84             color=alt.Color(f"{{field}}:Q").scale(
85                 scheme=color_scheme).legend(orient
86                 ="Left", title=f"平均{{field}}"),
87             tooltip=[alt.Tooltip(field, title=f"平
                均{{field}}")]
88         )
89         .mark_bar()
90     )
91
92     text = (
93         bars
94         .mark_text(
95             align="Left",
96             dx=3
97         )
98         .encode(
99             text="Label:Q",
100         )
101     )
102
103     return bars + text
104
105
106 ((generate("人均消费", "reds") & generate("
    服务", "bluegreen")).resolve_scale(color
107     ="independent") | (generate("口味", "
108     plasma") & generate("环境", "spectral"))
109     .resolve_scale(color="independent")).
110     resolve_scale(color="independent")
111
112
113 # ## 人均消费热力分布图
114
115 # In[96]:
116
117 import altair as alt
118 click_district = alt.selection_point(fields
119     =["行政区"])
120
121 choropleth = (
122     alt.Chart(
123         gdf.rename(columns={"dt_id": "行政区", "
124             dt_name": "行政区名称"})
125     )
126     .mark_geoshape(
127         stroke="white",
128         strokeWidth=1
129     )
130     .encode(
131         color=alt.Color("人均消费:Q").scale(
132             scheme="reds").legend(orient="Left")
133     )
134     .properties(
135         width=500
136     )
137 )
138
139 text = (
140     bars.mark_text(
141         align="Left",
142         baseline="middle",
143         dx=3,
144         limit=50
145     )
146     .encode(
147         text="人均消费:Q"
148     )
149 )
150
151 opacity=alt.condition(click_district,
152     alt.value(1), alt.value(0.2))
153
154 .transform_lookup(
155     lookup="行政区",
156     from_=alt.LookupData(data=food_df_clean.
157         groupby("行政区")["人均消费"].mean()
158         .reset_index(), key="行政区", fields
159         =["人均消费"])
160 )
161
162 .properties(
163     height=600,
164     width=600,
165 )
166
167 .add_params(
168     click_district
169 )
170
171 bars = (
172     alt.Chart(
173         food_df_clean.groupby("行政区")["人均消
174             费"].mean().reset_index()
175     )
176     .mark_bar()
177     .encode(
178         x="人均消费",
179         opacity=alt.condition(click_district,
180             alt.value(1), alt.value(0.2)),
181         color=alt.Color("人均消费:Q").scale(
182             scheme="reds"),
183         y=alt.Y("行政区").sort("-x"),
184         tooltip=["人均消费:Q"]
185     )
186     .add_params(
187         click_district
188     )
189     .properties(
190         width=500
191     )
192 )
193
194 text = (
195     bars.mark_text(
196         align="Left",
197         baseline="middle",
198         dx=3,
199         limit=50
200     )
201     .encode(
202         text="人均消费:Q"
203     )
204 )

```

```

173 )
174 .add_params(
175     click_district
176 )
177 )
178 (choropleth | (bars + text)).show()
179
180
181 # In[146]:
182
183
184 import altair as alt
185
186 click_district = alt.selection_point(fields
187     =["行政区"], value="A")
188
189 choropleth = (
190     alt.Chart(
191         gdf.rename(columns={"dt_id": "行政区", "
192             dt_name": "行政区名称"}))
193     )
194     .mark_geoshape(
195         stroke="white",
196         strokeWidth=1,
197         fill="red"
198     )
199     .encode(
200         tooltip=["行政区名称"],
201         opacity=alt.condition(click_district,
202             alt.value(1), alt.value(0.2))
203     )
204     .add_params(
205         click_district
206     )
207 )
208 count_bar = (
209     alt.Chart(
210         pd.concat(
211             map(
212                 lambda x: food_df_clean[
213                     food_df_clean["行政区"] == x][
214                         "类别"]
215                     .value_counts()
216                     .reset_index()
217                     .assign(行政区=x)
218             , food_df_clean["行政区"].unique()
219         )

```

```

219 )
220 )
221 .mark_bar()
222 .transform_filter(
223     click_district
224 )
225 .encode(
226     x=alt.X("count:Q", title="总数"),
227     # opacity=alt.condition(click_district,
228         alt.value(1), alt.value(0.2)),
229     color=alt.Color("count:Q").scale(scheme=
230         "reds"),
231     y=alt.Y("类别:N").sort("-x"),
232     tooltip=[alt.Tooltip("count:Q", title="
233         总数")]
234 )
235 )
236 .add_params(
237     click_district
238 )
239 )
240 .properties(
241     width=500
242 )
243 count_text = (
244     count_bar.mark_text(
245         align="left",
246         baseline="middle",
247         dx=3,
248         limit=50
249     )
250 )
251 .encode(
252     text="count:Q"
253 )
254 .add_params(
255     click_district
256 )
257 )
258 choropleth | (count_bar + count_text)
259
260 # In[20]:
261
262 food_df_clean[(food_df_clean["类别"] == "烧
263     烤")][ "人均消费"].describe()
264
265 # In[50]:

```

```

266 import math
267 bbq = food_df_clean[(food_df_clean["类别"]
268     == "烧烤").copy()
269 bbq = bbq[(bbq["点评数"] > 20) & (bbq["人均
270     消费"] < 500)]
271 bbq["avg_score"] = bbq[["口味", "环境", "服
272     务"]].mean(axis=1).apply(math.log2)
273 bbq = bbq.groupby("avg_score").mean(
274     numeric_only=True).reset_index()
275 print(bbq)
276 alt.Chart(bbq).mark_line().encode(
277     x="avg_score:Q",
278     y="人均消费:Q"
279 ).properties(
280     width=900
281 ).interactive()
282
283 # In[117]:
284
285 df = food_df_clean.copy()
286 df = df[df["行政区"] != "L"]
287 df["点评数_filtered"] = df["点评数"] < 20
288 df["点评数_filtered"].value_counts()[True]
289 unpopular_df = df.groupby("行政区").agg(
290     unpopular_rate=("点评数_filtered",
291         lambda x: x.value_counts()[True] / x.
292         count()).reset_index()
293 )
294 print(unpopular_df)
295
296 (
297     alt.Chart(
298         gdf.rename(columns={"dt_id": "行政区", "
299             dt_name": "行政区名称"}))
300     )
301     .mark_geoshape(
302         stroke="white",
303         strokeWidth=1
304     )
305     .encode(
306         color=alt.Color("unpopular_rate:Q").
307             scale(scheme="reds").legend(orient="
308             left"),
309         tooltip=["行政区名称", "unpopular_rate:Q
310             "]
311     )
312     .transform_lookup(
313         lookup="行政区",
314         from_=alt.LookupData(data=unpopular_df,
315             key="行政区", fields=["
316             unpopular_rate"])

```

```

304 ).properties(
305     height=600,
306     width=600,
307 )
308 )
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

1.2 train

```

36     model_maps[dt_id + "," + classification]
37         = model.cluster_centers_
38     print(model.score(x))
39 print(model_maps)
40
41 # In[13]:
42
43
44
45 for k in model_maps:
46     model_maps[k] = list(map(list, model_maps[
47         k]))
48 with open("data.json", mode="w", encoding="
49     utf-8") as f:
50     f.write(json.dumps(model_maps,
51         ensure_ascii=False));
52
53 # In[14]:
54
55 from sklearn.cluster import KMeans
56 from sklearn.preprocessing import
57     StandardScaler
58 from sklearn.decomposition import PCA
59 import seaborn as sns
60 import matplotlib.pyplot as plt
61 # food_df_clean = food_df_clean[
62     food_df_clean["人均消费"] < 300]
63 # food_df_clean = food_df_clean[
64     food_df_clean["点评数"] > 50]
65 food_df_clean = food_df_clean[food_df_clean[
66     "行政区"] == "N"]
67 food_df_clean = food_df_clean[food_df_clean[
68     "类别"] == "浙菜"]
69
70 features = food_df_clean[["环境", "口味", "
71     服务", "人均消费"]]
72 x, y = features.iloc[:, 0:-1], features.iloc
73    [:, -1]
74 scaler = StandardScaler()
75 scaled_features = scaler.fit_transform(
76     features)
77
78 #pca 降维
79 a = x.apply(lambda row: row.sum(), axis=1)
80 pca = PCA(n_components=1)
81 reduced_features = pca.fit_transform(x)
82 df = pd.DataFrame({
83     "PCA": a,
84     "人均消费": y})

```

```

75 # kmeans = KMeans(n_clusters=3, random_state
76     =42)
77 # food_df_clean["Cluster"] = kmeans.
78     fit_predict(scaled_features)
79
80 plt.rcParams["font.sans-serif"] = ["SimHei"]
81 sns.scatterplot(x="PCA", y='人均消费', data=
82     df, palette="Set2")
83 plt.show()

```

2 K-means

2.1 k-means

Input: Number of nodes N_A , Number of blocks N_B , Node array A

Output: Separating result array R

Centers: array[Node], len= N_B ;

Randomly initializing Centers;

Cs: Current separation, array[int];

Ls: Last separation, array[int];

Function UpdateCs():

```

    Ls ← Cs;
    foreach  $i \in [1, N_A]$  do
        foreach  $j \in [1, N_B]$  do
            if  $dis(A_i, C_{s_i}) >$ 
                 $dis(A_i, Centers_j)$  then
                |  $C_{s_i} \leftarrow Centers_j$ 
            end
        end
    end
end

```

return

Function UpdateCenters():

```

    Csizes: array[int];
    foreach  $i \in [1, N_B]$  do
        |  $Csizes_i \leftarrow Cs.count(i);$ 
        |  $Centers_i \leftarrow 0;$ 
    end
    foreach  $i \in [1, N_A]$  do
        |  $Centers_{C_{s_i}} \leftarrow Centers_{C_{s_i}} +$ 
             $\frac{A_i}{Csizes_{C_{s_i}}};$ 
    end
end

```

return

UpdateCs();

while $Cs \neq Ls$ do

UpdateCenters();

UpdateCs();

end

return Cs ;

Algorithm 1: K-Means