

# TC1002S Herramientas computacionales: el arte de la analítica

This is a notebook with all your work for the final evidence of this course

## Niveles de dominio a demostrar con la evidencia

SING0202A

Interpreta interacciones entre variables relevantes en un problema, como base para la construcción de modelos bivariados basados en datos de un fenómeno investigado que le permita reproducir la respuesta del mismo. Es capaz de construir modelos bivariados que expliquen el comportamiento de un fenómeno.

## Student information

- Name: Daniel Estrada Ocaña
- ID: A01369854
- My career: ITC

## ✓ Importing libraries

```
1 import numpy as np
2 import pandas as pd
3 import seaborn as sns
4 import matplotlib.pyplot as plt
5 from sklearn.cluster import KMeans
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
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
1000
```

1 # Define where you are running the code: colab or local  
2 RunInColab = True # (False: no | True: yes)  
3  
4 # If running in colab:  
5 if RunInColab:  
6 # Mount your google drive in google colab  
7 from google.colab import drive  
8 drive.mount('/content/drive')  
9  
10 # Find location  
11 #!pwd  
12 #!ls  
13 #!ls "/content/drive/My Drive/Colab Notebooks/MachineLearningWithPython/"  
14  
15 # Define path del proyecto  
16 Ruta = "/content/drive/MyDrive/Sistemas/4to\_semestre/semanaTec/TC1002S/"  
17  
18 else:  
19 # Define path del proyecto  
20 Ruta = ""  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
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  
1000

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force\_remount=True).

## ✓ PART 1

## Use your assigned dataset

## ✓ A1 Load data

```

1 # Dataset url
2 url = Ruta + "Evidencia/A01369854_X.csv"
3
4 # Load the dataset
5 df = pd.read_csv(url)

```

## ✓ A2 Data management

Print the first 7 rows

```
1 df.head(7)
```

	Unnamed: 0	x1	x2	x3	x4
0	0	-2.637008	6.469373	-8.065872	10.382298
1	1	-7.198506	-2.450691	-10.125392	-8.711022
2	2	-4.965669	-1.941930	-13.139206	-5.410215
3	3	8.318951	9.783888	9.011195	-3.365455
4	4	5.610515	9.479026	9.551860	-7.023893
5	5	9.083749	11.584535	10.599982	-2.966507
6	6	6.649827	10.571454	9.169478	-3.110495

Next steps:

 [View recommended plots](#)

Print the last 4 rows

```
1 df.tail(4)
```

	Unnamed: 0	x1	x2	x3	x4
388	388	0.742929	8.652554	-3.018461	6.537593
389	389	2.357353	6.295590	-5.362303	8.979738
390	390	2.650814	10.639732	-2.007908	11.072933
391	391	-7.072400	-0.704284	-10.758445	-6.019089

How many rows and columns are in your data?

Use the `shape` method

```
1 df.shape
```

```
(392, 5)
```

Print the name of all columns

Use the `columns` method

```
1 df.columns
```

```
Index(['Unnamed: 0', 'x1', 'x2', 'x3', 'x4'], dtype='object')
```

What is the data type in each column

Use the `dtypes` method

```
1 df.dtypes
```

```

Unnamed: 0    int64
x1            float64
x2            float64
x3            float64

```

```
x4          float64
dtype: object
```

What is the meaning of rows and columns?

```
1 # Your responses here
2
3 # 1) Rows refers to the number of observations in our dataset
4
5 # 2) Columns refers to our variables or features in our dataset
6
```

Print a statistical summary of your columns

```
1 df.describe()
```

	Unnamed: 0	x1	x2	x3	x4
count	392.000000	392.000000	392.000000	392.000000	392.000000
mean	195.500000	3.173315	5.889524	1.342545	-2.138052
std	113.304898	6.302593	4.864176	8.243437	7.199664
min	0.000000	-9.861084	-8.021386	-13.703612	-11.843976
25%	97.750000	-2.333433	3.361996	-6.725301	-7.434815
50%	195.500000	4.962112	7.413908	2.431116	-5.101422
75%	293.250000	8.113206	9.376306	9.184893	2.440813
max	391.000000	15.449343	13.684451	15.239239	13.346037

```
1 # 1) What is the mininum and maximum values of each variable
2
3 # x1: min= -9.871084, max= 15.449343
4 # x2: min= -8.021386, max= 13.684451
5 # x3: min= -13.703612, max= 15.239239
6 # x4: min= -11.843976, max= 13.346037
7
8 # 2) What is the mean and standar deviation of each variable
9
10 # x1: mean= 3.173315, standar deviation= 6.302593
11 # x2: mean= 5.889524, standar deviation= 4.864176
12 # x3: mean= 1.342545, standar deviation= 8.243437
13 # x4: mean= -2.138052, standar deviatio= 7.199664
14
15
16 # 3) What the 25%, 50% and 75% represent?
17
18 # x1: 25%= -2.333433, 50%= 4.962112, 75%= 8.113206
19 # x2: 25%= 3.361996, 50%= 7.413908, 75%= 9.376306
20 # x3: 25%= -6.725301, 50%= 2.431116, 75%= 9.184893
21 # x4: 25%= -7.434815, 50%= -5.101422, 75%= 2.440813
22
```

Rename the columns using the same name with capital letters

```
1 newNames = {'x1':'X1', 'x2':'X2', 'x3':'X3', 'x4':'X4'}
2 df = df.rename(columns=newNames)
3 df
```

	Unnamed: 0	X1	X2	X3	X4
0	0	-2.637008	6.469373	-8.065872	10.382298
1	1	-7.198506	-2.450691	-10.125392	-8.711022
2	2	-4.965669	-1.941930	-13.139206	-5.410215
3	3	8.318951	9.783888	9.011195	-3.365455
4	4	5.610515	9.479026	9.551860	-7.023893
...	...	...	...	...	...
387	387	8.867089	8.961678	9.061760	-2.688653
388	388	0.742929	8.652554	-3.018461	6.537593
389	389	2.357353	6.295590	-5.362303	8.979738
390	390	2.650814	10.639732	-2.007908	11.072933
391	391	-7.072400	-0.704284	-10.758445	-6.019089

392 rows × 5 columns

Next steps: [View recommended plots](#)

Rename the columns to their original names

```
1 oriNames = {'X1':'x1', 'X2':'x2', 'X3':'x3', 'X4':'x4'}
2 df = df.rename(columns=oriNames)
3 df
```

	Unnamed: 0	x1	x2	x3	x4
0	0	-2.637008	6.469373	-8.065872	10.382298
1	1	-7.198506	-2.450691	-10.125392	-8.711022
2	2	-4.965669	-1.941930	-13.139206	-5.410215
3	3	8.318951	9.783888	9.011195	-3.365455
4	4	5.610515	9.479026	9.551860	-7.023893
...	...	...	...	...	...
387	387	8.867089	8.961678	9.061760	-2.688653
388	388	0.742929	8.652554	-3.018461	6.537593
389	389	2.357353	6.295590	-5.362303	8.979738
390	390	2.650814	10.639732	-2.007908	11.072933
391	391	-7.072400	-0.704284	-10.758445	-6.019089

392 rows × 5 columns

Next steps: [View recommended plots](#)

Use two different alternatives to get one of the columns

```
1 a = df.x1
2 b = df.loc[:, 'x1']
3
4 print(a,b)
5
```

0	-2.637008
1	-7.198506
2	-4.965669
3	8.318951
4	5.610515
...	...
387	8.867089
388	0.742929
389	2.357353
390	2.650814
391	-7.072400

```
Name: x1, Length: 392, dtype: float64 0      -2.637008
1      -7.198506
2      -4.965669
3       8.318951
4       5.610515
...
387     8.867089
388     0.742929
389     2.357353
390     2.650814
391    -7.072400
Name: x1, Length: 392, dtype: float64
```

Get a slice of your data set: second and thrid columns and rows from 62 to 72

```
1 df.iloc[62:72, 2:4]
```

	x2	x3
62	-8.021386	-8.064350
63	6.346805	8.935645
64	7.172351	11.568654
65	0.237704	-6.323923
66	8.952647	-1.928209
67	6.119641	-0.663169
68	9.973920	-1.747450
69	12.474618	9.468129
70	-3.208477	-10.540776
71	6.457696	-5.976781

For the second and thrid columns, calculate the number of null and not null values and verify that their sum equals the total number of rows

```
1 columnas = ['x2','x3']
2
3 print('Null values: \n', df[columnas].isnull().sum())
4 print('Not Null values: \n', df[columnas].notnull().sum())
5 print('Total number of rows:', df.shape[0])
6
7
8 print(df.shape[0]==df[columnas].notnull().sum())
```

```
Null values:
x2      0
x3      0
dtype: int64
Not Null values:
x2     392
x3     392
dtype: int64
Total number of rows: 392
x2      True
x3      True
dtype: bool
```

Discard the last column

```
1 df.drop('x4', axis=1, inplace=True)
2
3 df
```

	Unnamed: 0	x1	x2	x3
0	0	-2.637008	6.469373	-8.065872
1	1	-7.198506	-2.450691	-10.125392
2	2	-4.965669	-1.941930	-13.139206
3	3	8.318951	9.783888	9.011195
4	4	5.610515	9.479026	9.551860
...	...	...	...	...
387	387	8.867089	8.961678	9.061760
388	388	0.742929	8.652554	-3.018461
389	389	2.357353	6.295590	-5.362303
390	390	2.650814	10.639732	-2.007908
391	391	-7.072400	-0.704284	-10.758445

392 rows × 4 columns

Next steps:

☒ View recommended plots

## Questions

Based on the previous results, provide a description of your dataset

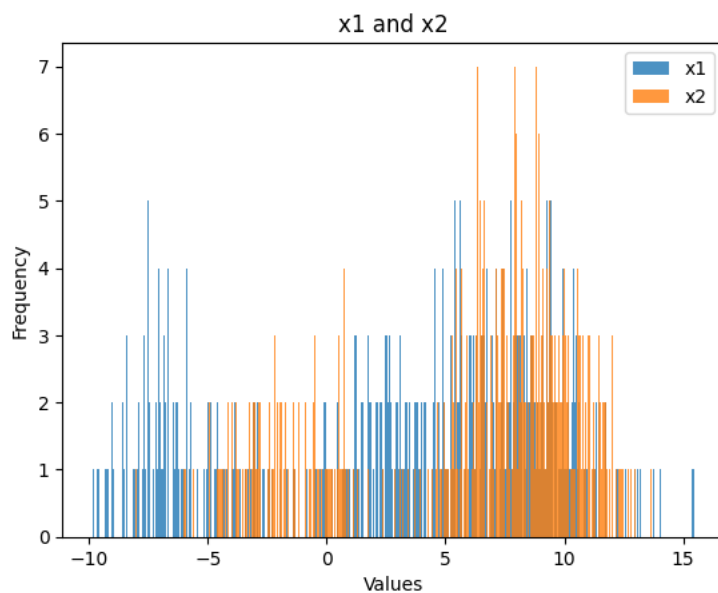
Your response:

My dataset has 392 observations (rows) and 4 variables (columns), it consists of numerical values across four columns labeled x1, x2, x3, and x4. Without any more information, it seems like a tabular dataset with numerical values across different variables.

## ✓ A3 Data visualization

Plot in the same figure the histogram of two variables

```
1 plt.hist(df['x1'], bins=len(df), alpha=0.8, label='x1')
2 plt.hist(df['x2'], bins=len(df), alpha=0.8, label='x2')
3 plt.title('x1 and x2')
4 plt.xlabel('Values')
5 plt.ylabel('Frequency')
6 plt.legend()
7 plt.show()
```



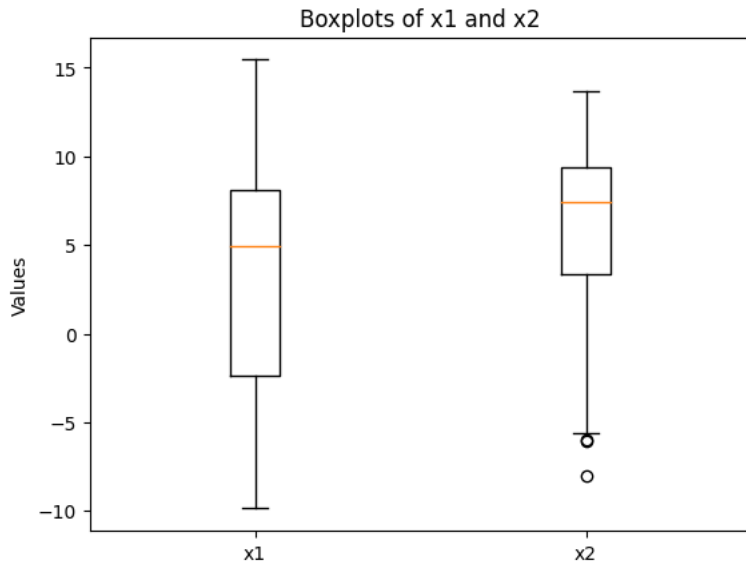
Based on these plots, provide a description of your data:

Your response here:

We can see that the histogram of x1 and x2 appears to have a roughly symmetric distribution, centered around the mean, with values ranging from negative to positive mostly in range between -5-10. While analyzed separately, there may be some overlap between the distributions of x1 and x2, indicating potential correlation or relationship between these variables

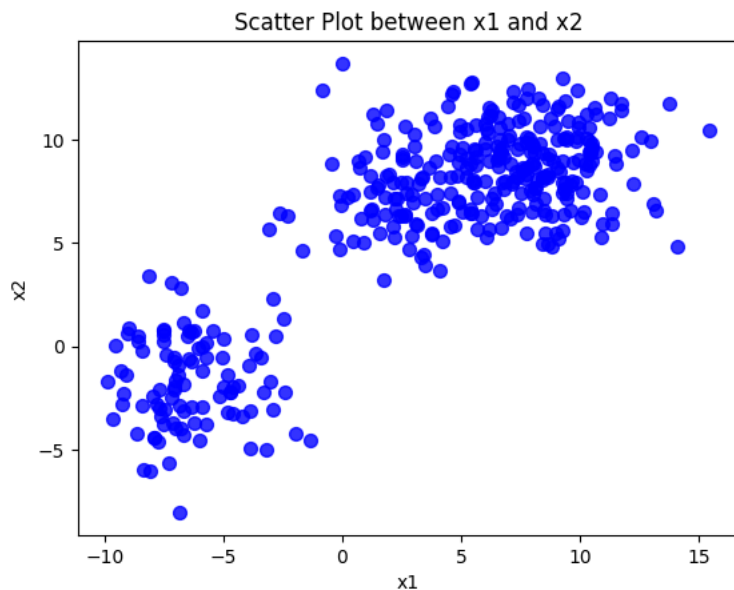
Plot in the same figure the boxplot of two variables

```
1 plt.boxplot([df['x1'], df['x2']])
2 plt.title('Boxplots of x1 and x2')
3 plt.ylabel('Values')
4 plt.xticks([1, 2], ['x1', 'x2'])
5 plt.show()
```



Plot the scatter plot of two variables

```
1 plt.scatter(df.x1, df.x2, color='b', s=50, marker='o', alpha=0.8)
2 plt.title('Scatter Plot between x1 and x2')
3 plt.xlabel('x1')
4 plt.ylabel('x2')
5 plt.show()
```



## Questions

Based on the previos plots, provide a description of yout dataset

Your response:

Based on the histograms and scatter plot between x1 and x2, both variables exhibit symmetric distributions with values spanning from negative to positive. The scatter plot suggests a positive correlation between x1 and x2, as data points tend to form a linear pattern. We can observe that there are 3 groups (cluster) or even more, but two are more visible.

**Loading the dataframe again to recover the fourth column that was eliminated before**

```
1 df = pd.read_csv(url)
2
3 df
```

	Unnamed: 0	x1	x2	x3	x4
0	0	-2.637008	6.469373	-8.065872	10.382298
1	1	-7.198506	-2.450691	-10.125392	-8.711022
2	2	-4.965669	-1.941930	-13.139206	-5.410215
3	3	8.318951	9.783888	9.011195	-3.365455
4	4	5.610515	9.479026	9.551860	-7.023893
...	...	...	...	...	...
387	387	8.867089	8.961678	9.061760	-2.688653
388	388	0.742929	8.652554	-3.018461	6.537593
389	389	2.357353	6.295590	-5.362303	8.979738
390	390	2.650814	10.639732	-2.007908	11.072933
391	391	-7.072400	-0.704284	-10.758445	-6.019089

392 rows × 5 columns

Next steps: [View recommended plots](#)

## A4 Kmeans

Do Kmeans clustering assuming a number of clusters accorging to your scatter plot

```
1 K = 3
2
3 km = KMeans(n_clusters=K, n_init="auto")
4
5 yestimated = km.fit_predict(df[['x1','x2','x3','x4']])
6
7
8 yestimated
```

array([0, 2, 2, 1, 1, 1, 1, 2, 2, 1, 1, 2, 0, 0, 2, 1, 1, 1, 1, 2, 0, 2,  
 2, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 2, 0, 1, 1, 0, 1, 1, 1, 2,  
 1, 1, 2, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 2, 2, 1, 1, 2,  
 0, 0, 0, 1, 2, 0, 1, 1, 1, 0, 0, 2, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1,  
 1, 2, 2, 0, 1, 2, 1, 0, 1, 1, 2, 0, 2, 1, 1, 0, 1, 0, 2, 1, 1, 0,  
 1, 2, 1, 2, 1, 1, 1, 2, 1, 0, 2, 1, 0, 2, 2, 2, 0, 1, 0, 2, 1, 0,  
 1, 2, 2, 1, 1, 2, 1, 1, 1, 1, 2, 1, 2, 1, 0, 1, 1, 2, 1, 1, 1, 1,  
 0, 2, 2, 2, 1, 2, 2, 1, 1, 1, 1, 2, 1, 0, 0, 1, 2, 2, 0, 1, 1, 1,  
 1, 2, 1, 0, 1, 1, 2, 2, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 2, 0, 2,  
 2, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 2, 1, 0, 1, 1, 0,  
 2, 1, 0, 1, 1, 1, 1, 2, 0, 1, 1, 1, 2, 1, 1, 0, 1, 1, 2, 2, 1, 0,  
 1, 1, 0, 0, 1, 1, 1, 2, 0, 1, 1, 0, 1, 1, 0, 2, 1, 2, 0, 1, 2, 1,  
 1, 1, 1, 2, 1, 1, 2, 0, 1, 0, 1, 2, 1, 0, 1, 1, 2, 2, 0, 0, 1,  
 1, 1, 0, 0, 1, 0, 0, 1, 2, 1, 2, 0, 1, 0, 1, 2, 1, 1, 2, 1, 1, 2,  
 2, 1, 1, 2, 1, 0, 0, 1, 1, 0, 1, 0, 1, 2, 1, 2, 1, 1, 2, 1, 1, 1,  
 0, 1, 2, 2, 1, 2, 2, 0, 1, 1, 0, 1, 0, 2, 2, 0, 1, 2, 2, 2, 2, 2, 2])



```
2, 1, 0, 1, 2, 1, 1, 1, 2, 1, 2, 2, 0, 0, 1, 0, 1, 1, 2, 0, 0, 2,
0, 0, 1, 2, 2, 0, 0, 0, 1, 1, 1, 2, 0, 1, 0, 0, 0, 2], dtype=int32)
```

Add to your dataset a column with the estimated cluster to each data point

```
1 df['yestimated'] = yestimated
2
3 df
```

	Unnamed: 0	x1	x2	x3	x4	yestimated
0	0	-2.637008	6.469373	-8.065872	10.382298	0
1	1	-7.198506	-2.450691	-10.125392	-8.711022	2
2	2	-4.965669	-1.941930	-13.139206	-5.410215	2
3	3	8.318951	9.783888	9.011195	-3.365455	1
4	4	5.610515	9.479026	9.551860	-7.023893	1
...	...	...	...	...	...	...
387	387	8.867089	8.961678	9.061760	-2.688653	1
388	388	0.742929	8.652554	-3.018461	6.537593	0
389	389	2.357353	6.295590	-5.362303	8.979738	0
390	390	2.650814	10.639732	-2.007908	11.072933	0
391	391	-7.072400	-0.704284	-10.758445	-6.019089	2

392 rows × 6 columns

Next steps: [View recommended plots](#)

Print the number associated to each cluster

```
1 df.yestimated.unique()
array([0, 2, 1], dtype=int32)
```

Print the centroids

```
1 km.cluster_centers_
array([[ 2.74604124,  7.39604345, -5.00441767,  9.52674172],
       [ 8.13425613,  8.93631206,  9.29529553, -5.3125144 ],
       [-6.32129211, -1.710573  , -8.21599227, -7.4539196 ]])
```

Print the inertia metric

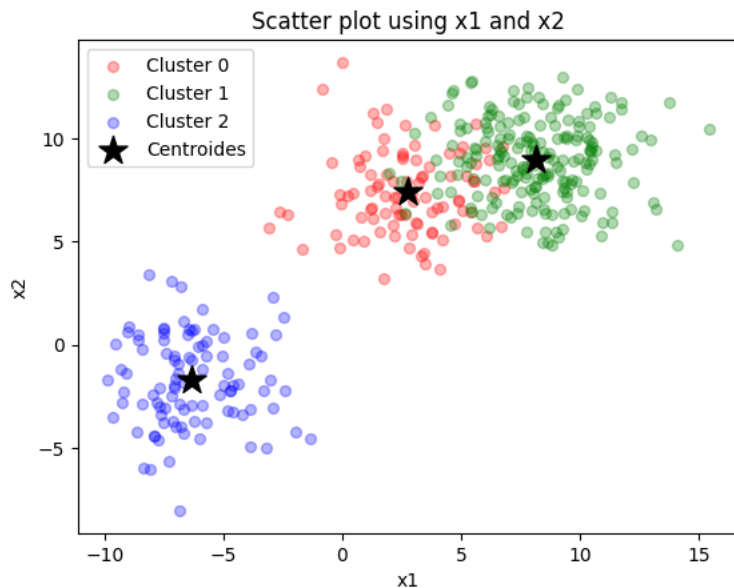
```
1 km.inertia_
6864.785555633871
```

Plot a scatter plot of your data using different color for each cluster. Also plot the centroids

```

1 df0 = df[df.yestimated==0]
2 df1 = df[df.yestimated==1]
3 df2 = df[df.yestimated==2]
4
5
6 # Scatter plot of each cluster
7 plt.scatter(df0.x1, df0.x2, label='Cluster 0', c='r', marker='o', s=32, alpha=0.3)
8 plt.scatter(df1.x1, df1.x2, label='Cluster 1', c='g', marker='o', s=32, alpha=0.3)
9 plt.scatter(df2.x1, df2.x2, label='Cluster 2', c='b', marker='o', s=32, alpha=0.3)
10
11 # Plot centroids
12 plt.scatter(km.cluster_centers_[0], km.cluster_centers_[1], color='black', marker='*', label='Centroides', s=256)
13
14 plt.title('Scatter plot using x1 and x2')
15 plt.xlabel('x1')
16 plt.ylabel('x2')
17 plt.legend()
18 plt.show()

```



## Questions

Provides a detailed description of your results

Your response:

We can see that creating 3 cluster points is pretty accurate, on this scatter plot we only use x1 and x2, but to know for sure if using any other variable will be as precise we can create and allow point. Just to verify that the number of cluster points is correct.

## ✓ A5 Elbow plot

Compute the Elbow plot

```

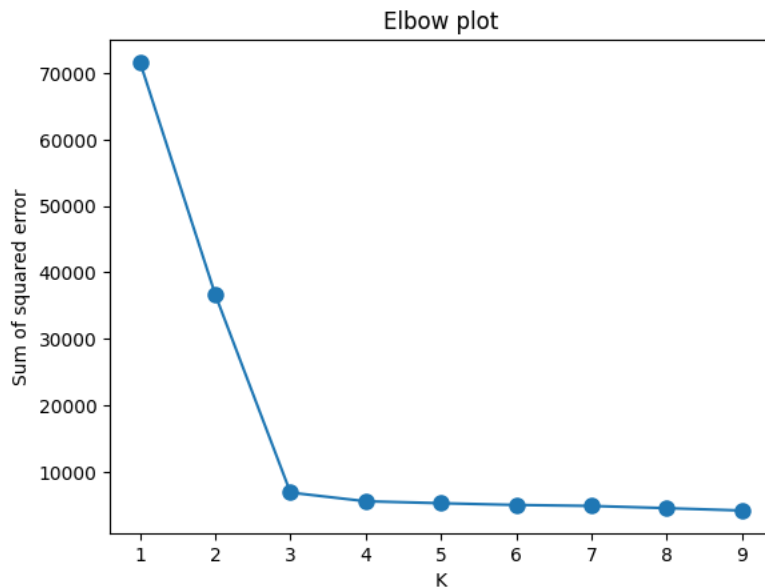
1 sse = []
2
3 # Define values of k
4 k_rng = range(1,10)
5
6 # For each k
7 for k in k_rng:
8     # Create model
9     km = KMeans(n_clusters=k, n_init="auto")
10    # Do K-means clustering
11    km.fit_predict(df[['x1', 'x2', 'x3', 'x4']])
12    # Save sse for each k
13    sse.append(km.inertia_)

```

```

1 plt.plot(k_rng,sse, 'o-', markersize=8)
2
3 plt.title('Elbow plot')
4 plt.xlabel('K')
5 plt.ylabel('Sum of squared error')
6 plt.show()

```



## Questions

What is the best number of clusters K? (argue your response)

Your response: K = 3, we can be sure of this because on the elbow plot, that's where the decrease rate gets lower.

Does this number of clusters agree with your initial guess? (argue your response, no problem at all if they do not agree)

Your response:

Yes, the agreement between the number of clusters obtained from the elbow point method and the initial guess suggests that the structure of the data aligns well with the prior assumption regarding the number of distinct groups or clusters present. This agreement validates the initial hypothesis and indicates that the clustering algorithm successfully captured the inherent patterns in the data. It also enhances the interpretability and consistency of the analysis, providing confidence in the understanding of the data's organization.

## ✓ PART 2

## Descripción de tu percepción del nivel de desarrollo de la subcompetencia

### SING0202A Interpretación de variables

Escribe tu descripción del nivel de logro del siguiente criterio de la subcompetencia

**Interpreta interacciones.** Interpreta interacciones entre variables relevantes en un problema, como base para la construcción de modelos bivariados basados en datos de un fenómeno investigado que le permita reproducir la respuesta del mismo.

Tu respuesta:

En base a esta subcompetencia, creo que la interpretación de variables es fácil de desarrollar. Siempre tenemos que comprender el tipo de información con la que estamos trabajando, ya sean números enteros, flotantes, o incluso strings, y de esta forma será más fácil de comprender el uso o el origen de estas variables. Cuando no viene ninguna descripción o interpretación de las variables es necesario analizar toda la información para poder darle una interpretación y sea así más fácil su manejo para desarrollar modelos y utilizar la información de manera sencilla y correcta.

Escribe tu descripción del nivel de logro del siguiente criterio de la subcompetencia

**Construcción de modelos.** Es capaz de construir modelos bivariados que expliquen el comportamiento de un fenómeno.

Tu respuesta:

Gracias a la evidencia puedo comprobar que soy capaz de contruir modelos bivariados, manejando de manera correcta las variables, comprendiendo su tipo e incluso interpretando su significado para poder visualizar la información, y así usarla con fines de modelos de entrenamiento de IA.

```
1 #FINAL DEL DOCUMENTO
2 #
```