

```
1 import pandas as pd
2 df = pd.read_csv('models.csv')
3
```

```
1 print(df.columns)
2 print(df.info())
3 col=df.columns
```

↗ Index(['mpg', 'cylinders', 'displacement', 'horsepower', 'weight', 'acceleration', 'model year', 'origin', 'car name'], dtype='object')  
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 398 entries, 0 to 397  
Data columns (total 9 columns):  
# Column Non-Null Count Dtype  
--- ---  
0 mpg 398 non-null float64  
1 cylinders 398 non-null int64  
2 displacement 398 non-null float64  
3 horsepower 398 non-null object  
4 weight 398 non-null int64  
5 acceleration 398 non-null float64  
6 model year 398 non-null int64  
7 origin 398 non-null int64  
8 car name 398 non-null object  
dtypes: float64(3), int64(4), object(2)  
memory usage: 28.1+ KB  
None

```
1 df.head(5)
2
3
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino

```
1 column_names = list(df.columns)
```

▼ To check number of unique in each column

```
1 for i,item in enumerate(column_names):
2     print(f'{item}-->\n')
3     print(df[column_names[i]].value_counts())
4     print("\n")
5     print(df[column_names[i]].nunique())
6
7
8
```

mpg-->

13.0	20
14.0	19
18.0	17
15.0	16
26.0	14
	..
31.9	1
16.9	1
18.2	1
22.3	1
44.0	1

Name: mpg, Length: 129, dtype: int64

129  
cylinders-->

4	204
8	103
6	84
3	4
5	3

Name: cylinders, dtype: int64

5  
displacement-->

97.0	21
98.0	18
350.0	18
318.0	17
250.0	17
	..
104.0	1
110.0	1
130.0	1
111.0	1
144.0	1


Name: displacement, Length: 82, dtype: int64

82  
horsepower-->

150	22
-----	----

90	20
88	19
110	18
100	17
	..
61	1
93	1
148	1
152	1
82	1

1 df.describe().T

	count	mean	std	min	25%	50%	75%	max	
mpg	398.0	23.514573	7.815984	9.0	17.500	23.0	29.000	46.6	
cylinders	398.0	5.454774	1.701004	3.0	4.000	4.0	8.000	8.0	
displacement	398.0	193.425879	104.269838	68.0	104.250	148.5	262.000	455.0	
weight	398.0	2970.424623	846.841774	1613.0	2223.750	2803.5	3608.000	5140.0	
acceleration	398.0	15.568090	2.757689	8.0	13.825	15.5	17.175	24.8	
model year	398.0	76.010050	3.697627	70.0	73.000	76.0	79.000	82.0	
origin	398.0	1.572864	0.802055	1.0	1.000	1.0	2.000	3.0	

```
1 from scipy.stats import pearsonr
2
3 import pandas as pd
4 import numpy as np
5 import matplotlib.pyplot as plt
6 import seaborn as sns
7 import random
```

1 df1=df

▼ Use regular expression to extract only 2-3 first characters from column car name 🚗

1 import re

```
1 df1['Type'] = df['car name'].str.extract('(.{2,3})', expand=False)
2 print(df1['Type'].T.value_counts())
```

for	51
che	47
ply	31
amc	28

```
dod      28
toy      26
dat      23
vol      21
bui      17
pon      16
mer      14
hon      13
maz      10
old      10
peu       8
fia       8
aud       7
chr       6
vw        6
ren       5
ope       4
saa       4
sub       4
max       2
bmw       2
cad       2
hi        1
cap       1
vok       1
tri       1
nis       1
Name: Type, dtype: int64
```

```
1 df1['Type'] = df['Type'].astype('category')
2
3 # save new version of category codes
4 df1['Type'] = df['Type'].cat.codes
```

▼ To clean horsepower column

```
1 df1['horsepower']=df1['horsepower'].replace('?', '150')
2 df1['horsepower']=df1['horsepower'].astype('int')
```

```
1 print(df1['horsepower'].unique)
```

```
<bound method Series.unique of 0      130
1      165
2      150
3      150
4      140
...
393     86
394     52
395     84
396     79
```

```
397         82
Name: horsepower, Length: 398, dtype: int64)
```


1 df1.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 10 columns):
#   Column          Non-Null Count  Dtype  
---  -
0    mpg             398 non-null   float64
1    cylinders       398 non-null   int64  
2    displacement    398 non-null   float64
3    horsepower      398 non-null   int64  
4    weight          398 non-null   int64  
5    acceleration    398 non-null   float64
6    model year     398 non-null   int64  
7    origin          398 non-null   int64  
8    car name       398 non-null   object  
9    Type           398 non-null   int8    
dtypes: float64(3), int64(5), int8(1), object(1)
memory usage: 28.5+ KB
```

1 df1['horsepower'].unique()

```
array([130, 165, 150, 140, 198, 220, 215, 225, 190, 170, 160,  95,  97,
        85,  88,  46,  87,  90, 113, 200, 210, 193, 100, 105, 175, 153,
       180, 110,  72,  86,  70,  76,  65,  69,  60,  80,  54, 208, 155,
       112,  92, 145, 137, 158, 167,  94, 107, 230,  49,  75,  91, 122,
        67,  83,  78,  52,  61,  93, 148, 129,  96,  71,  98, 115,  53,
        81,  79, 120, 152, 102, 108,  68,  58, 149,  89,  63,  48,  66,
       139, 103, 125, 133, 138, 135, 142,  77,  62, 132,  84,  64,  74,
       116,  82])
```

```
1 df1.drop(labels="car name", axis=1, inplace=True)
2
3 df1.head()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	Type	
0	18.0	8	307.0	130	3504	12.0	70	1	6	
1	15.0	8	350.0	165	3693	11.5	70	1	3	
2	18.0	8	318.0	150	3436	11.0	70	1	21	
3	16.0	8	304.0	150	3433	12.0	70	1	0	
4	17.0	8	302.0	140	3449	10.5	70	1	11	

```
1 from sklearn.datasets import make_regression
2 from sklearn.model_selection import train_test_split
3 from sklearn.feature_selection import SelectKBest
```

```
1 from sklearn.feature_selection import mutual_info_regression
```

# 1-Mutual information as Assignment required(x=data,y=mpg)

▼  $I(X ; Y) = H(X) - H(X | Y)$

```
1 df_temp=df1
```

```
1 from sklearn.model_selection import train_test_split
2 X_train,X_test,y_train,y_test=train_test_split(df_temp.drop(labels=[ 'mpg' ], axis=1),
3         df_temp[ 'mpg' ],
4         test_size=0.3,
5         random_state=0)
6 from sklearn.feature_selection import mutual_info_regression
7 mutual_info = mutual_info_regression(X_train.fillna(0), y_train)
8 mutual_info
```

```
array([0.56049991, 0.7834542 , 0.71731231, 0.77523539, 0.13261398,
        0.29481409, 0.24267984, 0.21358316])
```

Double-click (or enter) to edit

▼ Highly Mutual information gain

Target = mpg

```
1 mutual_info = pd.Series(mutual_info)
2 mutual_info.index = X_train.columns
3 m1=mutual_info.sort_values(ascending=False).to_frame()
```

```
1 from google.colab import data_table
2
3 data_table.enable_dataframe_formatter()
4 data_table.DataTable(m1, include_index=True)
```

index	0
displacement	0.7834541975190548
weight	0.7752353889847736
horsepower	0.7173123149839054
...	0.5581688888888889

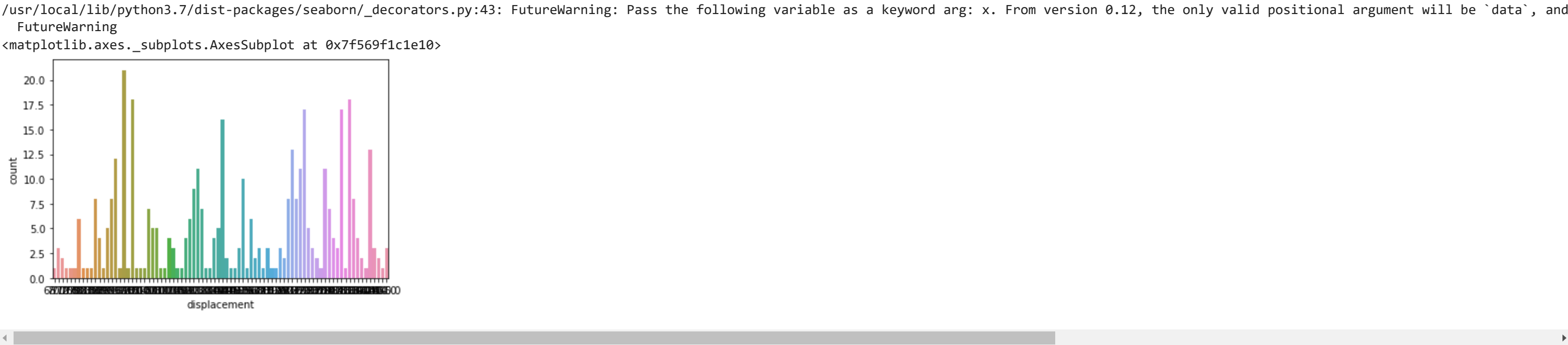
2-Mutual information as Assignment required(x=data,y=displacement)

Type	0.21358315745584155
------	---------------------

```
1 df_temp1=df1
```

Double-click (or enter) to edit

```
1 sns.countplot(df_temp1['displacement']) # We should here turn to discrete as it is the target
```



```
1 df_temp1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
#   Column      Non-Null Count  Dtype
---  -
0    mpg         398 non-null   float64
1    cylinders   398 non-null   int64
2    displacement 398 non-null   category
3    horsepower  398 non-null   int64
4    weight      398 non-null   int64
5    acceleration 398 non-null   float64
6    model year  398 non-null   int64
7    origin      398 non-null   int64
8    Type        398 non-null   int8
```

```
dtypes: category(1), float64(2), int64(5), int8(1)
```

```
1 df_temp1['displacement']=df_temp1['displacement'].cat.codes
```

```
1 df_temp1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  -
 0   mpg             398 non-null   float64
 1   cylinders       398 non-null   int64
 2   displacement    398 non-null   int8
 3   horsepower      398 non-null   int64
 4   weight          398 non-null   int64
 5   acceleration    398 non-null   float64
 6   model year      398 non-null   int64
 7   origin          398 non-null   int64
 8   Type            398 non-null   int8
dtypes: float64(2), int64(5), int8(2)
memory usage: 22.7 KB
```

```
1 from sklearn.model_selection import train_test_split
2 X_train,X_test,y_train,y_test=train_test_split(df_temp1, df_temp1['displacement'],
3         test_size=0.3,
4         random_state=0)
5 from sklearn.feature_selection import mutual_info_regression
6 mutual_info = mutual_info_regression(X_train, y_train)
7 mutual_info
```

```
array([0.63570915, 0.82320876, 1.64020157, 0.7304797 , 0.93577007,
        0.20017386, 0.0855606 , 0.32673863, 0.35972083])
```

## ▼ Highly Mutual information gain

**target = displacement**

```
1 mutual_info = pd.Series(mutual_info)
2
3 mutual_info.index= X_train.columns
4
5 m2=mutual_info.sort_values(ascending=False).to_frame()
6 m2
```



index	0 ▼
displacement	1.6402015660596843
weight	0.9357700710469548
cylinders	0.8232087633850549
horsepower	0.7304796993104103
mpg	0.6357091481272019
Type	0.35972082899951596
origin	0.32673863209376197

### 3-Mutual information as Assignment required(x=data,y=cylinders)

```
1 df_temp2=df1
```

```
1 from sklearn.model_selection import train_test_split
2 X_train,X_test,y_train,y_test=train_test_split(df_temp2.drop(labels=['cylinders'], axis=1),
3         df_temp2['cylinders'],
4         test_size=0.3,
5         random_state=0)
6 from sklearn.feature_selection import mutual_info_regression
7 mutual_info = mutual_info_regression(X_train.fillna(0), y_train)
8 mutual_info
```

```
array([0.53070992, 0.83156221, 0.62176275, 0.69320031, 0.22605598,
        0.12703738, 0.26831269, 0.20275092])
```

```
1 mutual_info = pd.Series(mutual_info)
2
3 mutual_info.index= X_train.columns
4
5 m3=mutual_info.sort_values(ascending=False).to_frame()
6 m3
```

4-Mutual information as Assignment required(x=data,y=weight)

[ ] 3 cells hidden

max	0.520700021515065
-----	-------------------

5-Mutual information as Assignment required(x=data,y=horsepower)

type	0.2027509161513695
------	--------------------

1 df\_temp4=df1

```
1 from sklearn.model_selection import train_test_split
2 X_train,X_test,y_train,y_test=train_test_split(df_temp4.drop(labels=[ 'horsepower' ], axis=1),
3         df_temp4[ 'horsepower' ],
4         test_size=0.3,
5         random_state=0)
6 from sklearn.feature_selection import mutual_info_regression
7 mutual_info = mutual_info_regression(X_train.fillna(0), y_train)
8 mutual_info
```

array([0.73939659, 0.68574292, 0.75896749, 0.8003739 , 0.3250427 ,
 0.22309322, 0.31916568, 0.42429563])

```
1 mutual_info = pd.Series(mutual_info)
2
3 mutual_info.index= X_train.columns
4
5 m5=mutual_info.sort_values(ascending=False).to_frame()
6 m5
```

1 to 8 of 8 entries

Filter

?

index	0
weight	0.800373902571315
displacement	0.7589674911159447
mpg	0.7393965904713986
cylinders	0.6857429203371184
Type	0.4242956334289114
acceleration	0.3250427024517828
origin	0.3191656768978477
model year	0.22309322484323513

Show 25 per page

▼ 6-Mutual information as Assignment required(x=data,y=acceleration)

```
1 df_temp5=df1

1 from sklearn.model_selection import train_test_split
2 X_train,X_test,y_train,y_test=train_test_split(df_temp5.drop(labels=['acceleration'], axis=1),
3         df_temp5['acceleration'],
4         test_size=0.3,
5         random_state=0)
6 from sklearn.feature_selection import mutual_info_regression
7 mutual_info = mutual_info_regression(X_train.fillna(0), y_train)
8 mutual_info

array([0.14624436, 0.22289052, 0.19554837, 0.30907361, 0.16818006,
        0.04683496, 0.08098417, 0.0406512  ])

1 mutual_info = pd.Series(mutual_info)
2
3 mutual_info.index= X_train.columns
4
5 m5=mutual_info.sort_values(ascending=False).to_frame()
6 m5
```

1 to 8 of 8 entries Filter  ?

index	0
horsepower	0.3090736117985924
cylinders	0.22289051994816522
displacement	0.1955483698595577
weight	0.16818006489815485
mpg	0.14624436066256052
origin	0.08098417237763811
model year	0.04683495636948276
Type	0.04065119711338738

Show 

25 ▼

 per page

▼ 7-Mutual information as Assignment required(x=data,y=origin)

```
1 df_temp6=df1
```

```
1 from sklearn.model_selection import train_test_split
```

```
2 X_train,X_test,y_train,y_test=train_test_split(df_temp6.drop(labels=['origin'], axis=1),
3         df_temp6['origin'],
4         test_size=0.3,
5         random_state=0)
6 from sklearn.feature_selection import mutual_info_regression
7 mutual_info = mutual_info_regression(X_train.fillna(0), y_train)
8 mutual_info
array([0.18172519, 0.32229184, 0.29801911, 0.32620247, 0.26334895,
       0.11576109, 0.02994163, 0.82833741])

1 mutual_info = pd.Series(mutual_info)
2
3 mutual_info.index= X_train.columns
4
5 m6=mutual_info.sort_values(ascending=False).to_frame()
6 m6
```

1 to 8 of 8 entries Filter  ?

index	0
Type	0.8283374108821122
horsepower	0.32620246822876364
cylinders	0.32229183887743096
displacement	0.298019106108089
weight	0.2633489489397567
mpg	0.1817251913133502
acceleration	0.11576108545158359
model year	0.029941625573480124

Show 

25

 per page

▼ 8-Mutual information as Assignment required(x=data,y=model year)

```
1 df_temp7=df1
```

```
1 from sklearn.model_selection import train_test_split
2 X_train,X_test,y_train,y_test=train_test_split(df_temp7.drop(labels=['model year'], axis=1),
3         df_temp7['model year'],
4         test_size=0.3,
5         random_state=0)
6 from sklearn.feature_selection import mutual_info_regression
```

```
7 mutual_info = mutual_info_regression(X_train.fillna(0), y_train)
8
9
10 array([0.3091082 , 0.08153791, 0.07698712, 0.23386751, 0.15912621,
11        0.05708936, 0.04178806, 0.
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
```

1 to 8 of 8 entries Filter  ?

index	0
mpg	0.3091081989568094
horsepower	0.23386750865763384
weight	0.1591262093060939
cylinders	0.08153790768753266
displacement	0.07698712299547283
acceleration	0.05708935843315732
origin	0.04178805873598912
Type	0.0

Show 

25

 per page

```
1
```

▼ 9-Mutual information as Assignment required(x=data,y=Type)

```
1 df_temp8=df1
2
3
4
5
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
```

```
1 mutual_info = pd.Series(mutual_info)
```

```
2
3 mutual_info.index= X_train.columns
4
5 m8=mutual_info.sort_values(ascending=False).to_frame()
6 m8
```

1 to 8 of 8 entries

Filter

?

index	0
origin	0.8547301573971928
horsepower	0.4132377717545732
displacement	0.33500552442209086
cylinders	0.2229849025808952
mpg	0.2158539090072149
weight	0.17744422336415733
acceleration	0.02772262281742588
model year	0.023860885026222967

Show 25 per page

▼ Use Pearson correlation which represent Maximal information criterion (MIC) as in our book  page(60)

1 df1.corr(method = 'pearson ')

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	Type	
mpg	1.000000	-0.775396	-0.804203	-0.753177	-0.831741	0.420289	0.579267	0.563450	0.288368	
cylinders	-0.775396	1.000000	0.950721	0.818454	0.896017	-0.505419	-0.348746	-0.562543	-0.286512	
displacement	-0.804203	0.950721	1.000000	0.873330	0.932824	-0.543684	-0.370164	-0.609409	-0.302291	
horsepower	-0.753177	0.818454	0.873330	1.000000	0.841770	-0.665833	-0.397772	-0.454271	-0.236643	
weight	-0.831741	0.896017	0.932824	0.841770	1.000000	-0.417457	-0.306564	-0.581024	-0.265872	
acceleration	0.420289	-0.505419	-0.543684	-0.665833	-0.417457	1.000000	0.288137	0.205873	0.138012	
model year	0.579267	-0.348746	-0.370164	-0.397772	-0.306564	0.288137	1.000000	0.180662	0.077134	
origin	0.563450	-0.562543	-0.609409	-0.454271	-0.581024	0.205873	0.180662	1.000000	0.374745	
Type	0.288368	-0.286512	-0.302291	-0.236643	-0.265872	0.138012	0.077134	0.374745	1.000000	

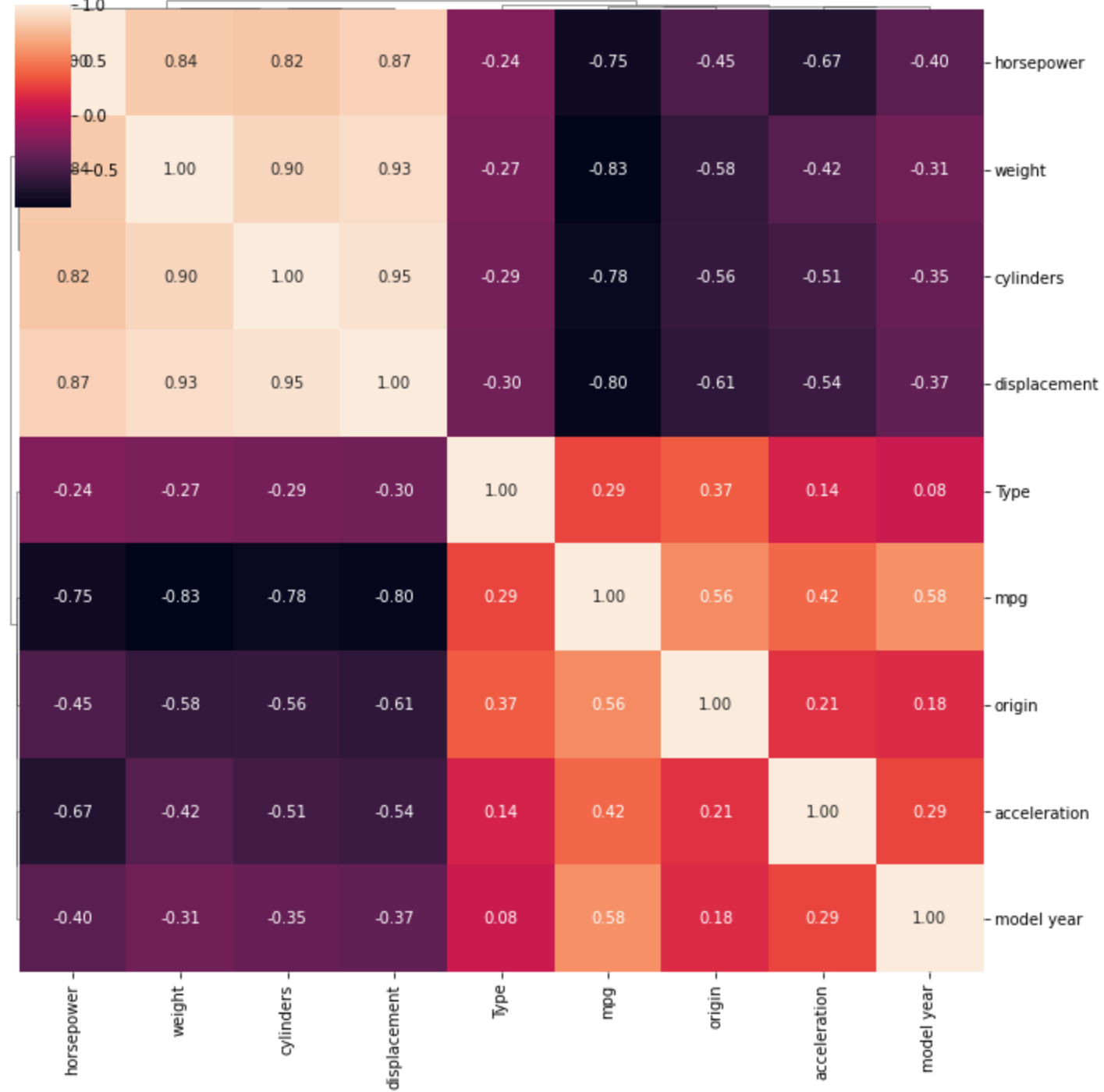
```
1 corr_matrix = df1.corr(method='pearson')
2
3
4 plt.figure(figsize=(25,25))
5 sns.clustermap(corr_matrix, annot=True, fmt = ".2f", dendrogram_ratio=0.01)
6
```

```
7 plt.title("Correlation between Features")
```

```
8 plt.show()
```

<Figure size 1800x1800 with 0 Axes>

Correlation between Features



1

```
1 colors = sns.diverging_palette(150, 275, as_cmap=True)
```

```
2
```

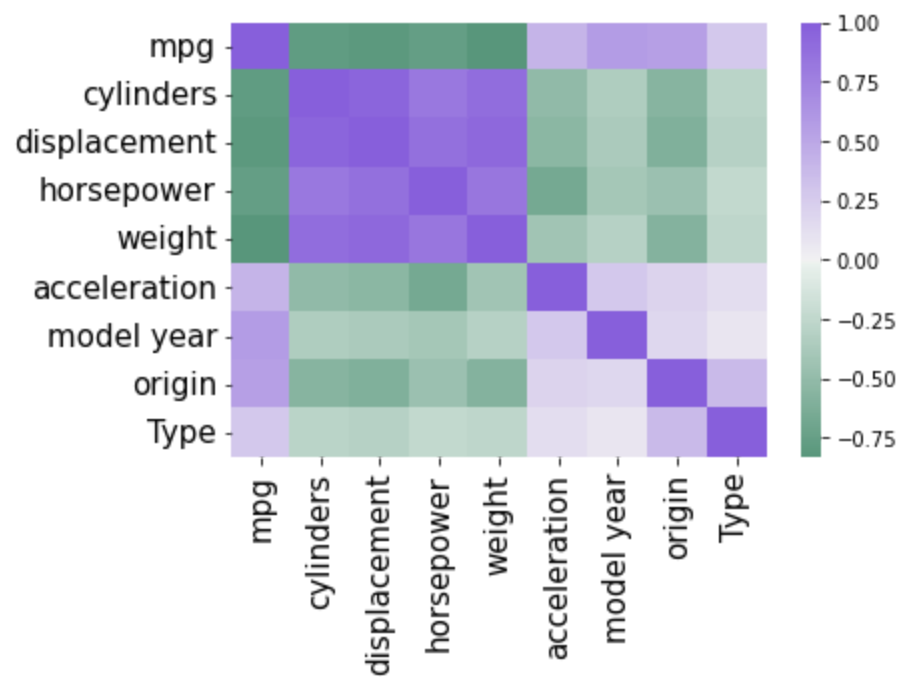
```
3 # Create heatmap using the .corr method on df, set colormap to cmap
```

```
4 sns.heatmap(df1.corr(), center=0, cmap=colors)
```

```
5 plt.xticks(fontsize= 15)
```

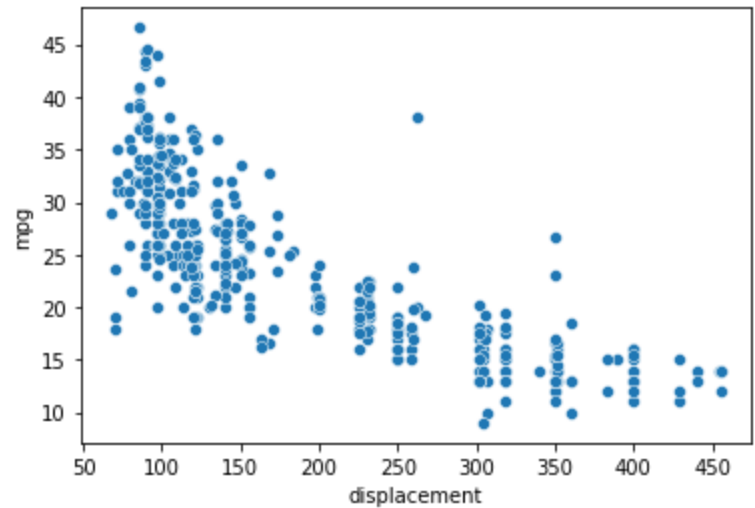
```
6 plt.yticks(fontsize= 15)
```

(array([0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5, 7.5, 8.5]),  
<a list of 9 Text major ticklabel objects>)



```
1 sns.scatterplot(x='displacement', y='mpg', data=df)
2
```

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f56a3a82750>



```
1 sns.scatterplot(x='horsepower', y='mpg', data=df)
```

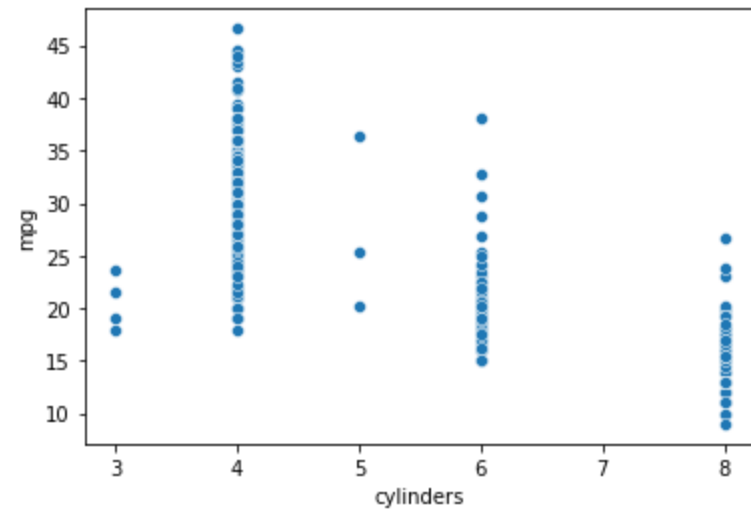


```
<matplotlib.axes._subplots.AxesSubplot at 0x7f56a3fc9590>
```

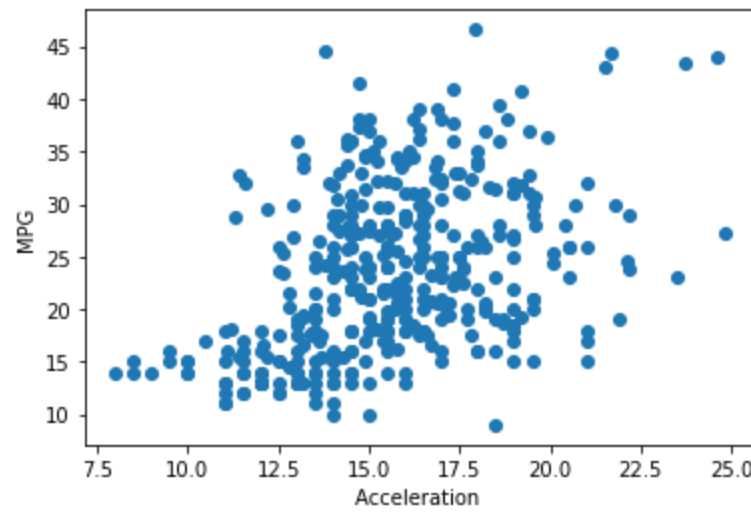


```
1 sns.scatterplot(x='cylinders', y='mpg', data=df)
```

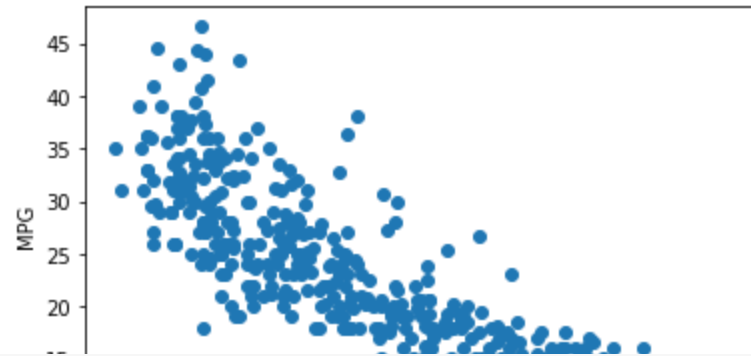
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f56a3ab7910>
```



```
1 plt.scatter(x = df.acceleration, y = df.mpg)
2 plt.xlabel('Acceleration ')
3 plt.ylabel('MPG')
4 plt.show()
```

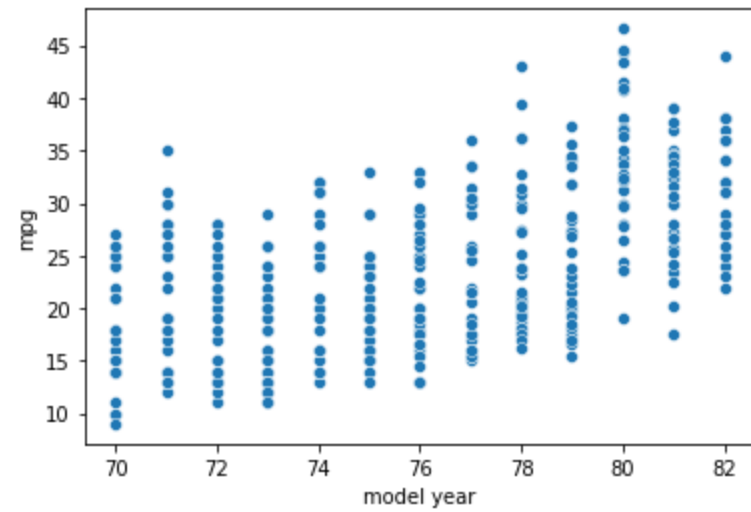


```
1 plt.scatter(x = df.weight, y = df.mpg)
2 plt.xlabel('Weight ')
3 plt.ylabel('MPG')
4 plt.show()
```

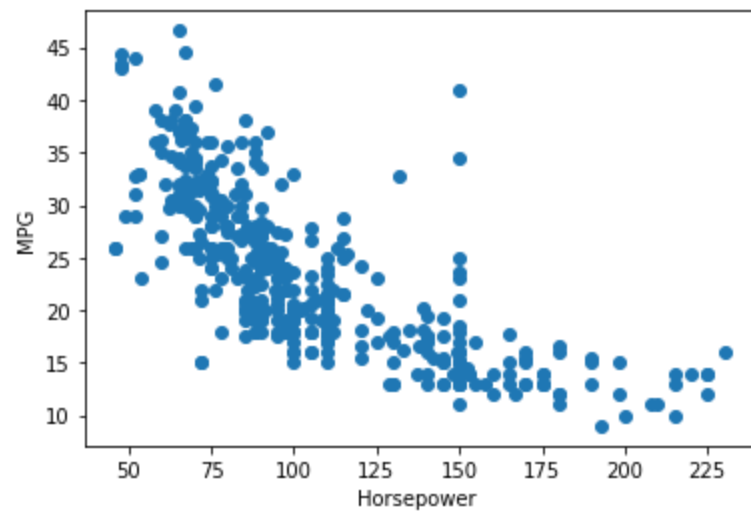


```
1 sns.scatterplot(x='model year', y='mpg', data=df)
```

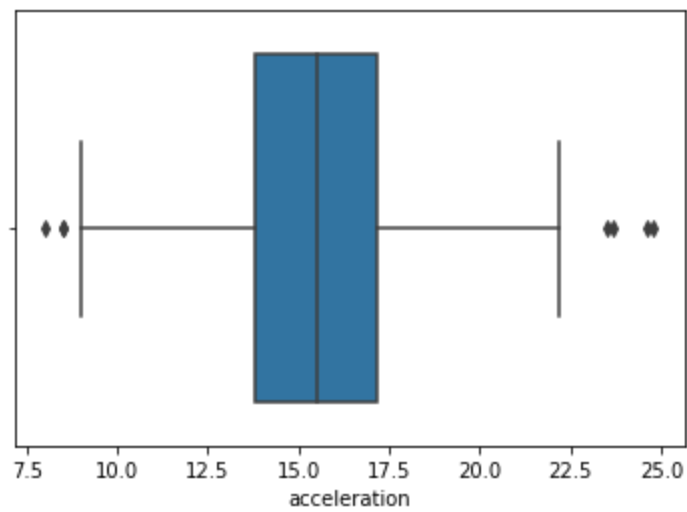
<matplotlib.axes.\_subplots.AxesSubplot at 0x7f56a39e9790>



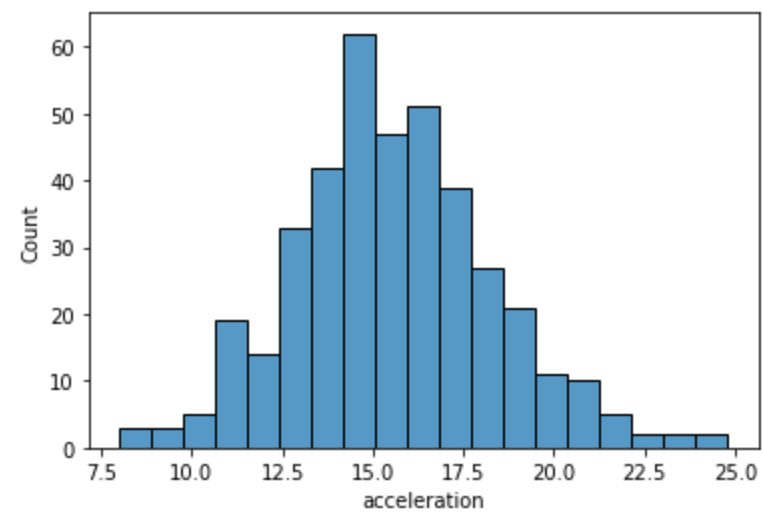
```
1 plt.scatter(x = df.horsepower, y = df.mpg)
2 plt.xlabel('Horsepower ')
3 plt.ylabel('MPG')
4 plt.show()
```



```
1 sns.boxplot(x='acceleration', data=df)
2 plt.show()
3 plt.close()
```



```
1 sns.histplot(x='acceleration', data=df)
2 plt.show()
3 plt.close()
```



```
1 df.acceleration.mad()
```

```
2.1425696320799963
```

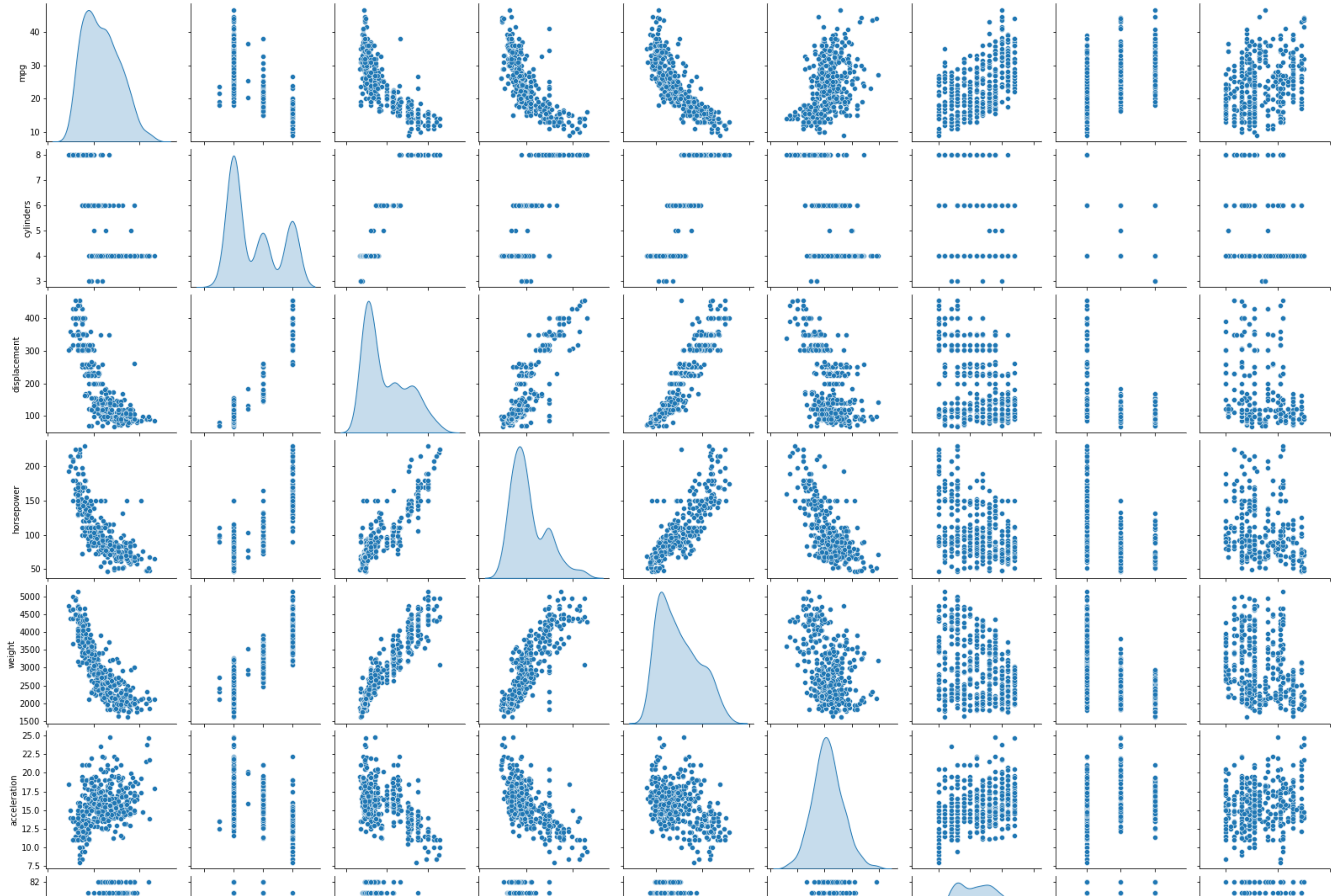
```
1 sns.countplot(x='displacement', data=df)
2 plt.show()
3 plt.close()
```

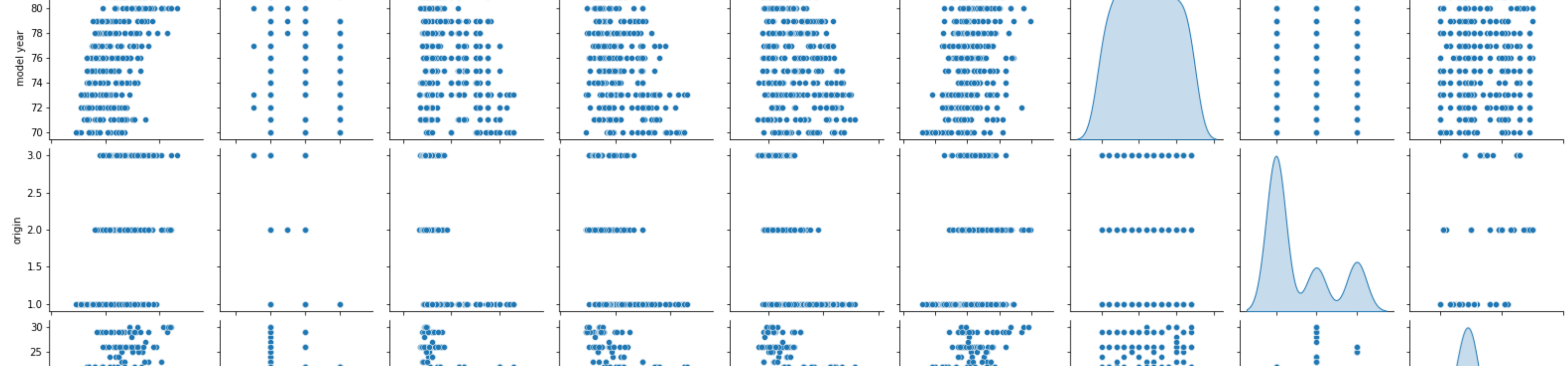


```
1 df.groupby('mpg').acceleration.max()
```

```
mpg
9.0    18.5
10.0    15.0
11.0    14.0
12.0    13.5
13.0    16.0
...
43.4    23.7
44.0    24.6
44.3    21.7
44.6    13.8
46.6    17.9
Name: acceleration, Length: 129, dtype: float64
```

```
1 sns.pairplot(df1, diag_kind="kde")
2 plt.show()
```





✓ 0s completed at 12:06 PM

