```python
import pandas as pd
df = pd.read_csv('models.csv')

```

```python
print(df.columns)
print(df.info())
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
```

```python
df.head(5)


```

	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

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

To check number of unique in each column

```python
for i,item in enumerate(column_names):
  print(f'{item}-->\n')
  print(df[column_names[i]].value_counts())
  print("\n")
  print(df[column_names[i]].nunique()) 
```

```
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
```

```
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**

```
from scipy.stats import pearsonr

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import random
```

```
df1=df
```

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

```
import re
```

```
df1['Type'] = df['car name'].str.extract('(.{2,3})', expand=False)
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
```

```
df1['Type'] = df['Type'].astype('category')

# save new version of category codes
df1['Type'] = df['Type'].cat.codes
```

To clean horsepower column

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

```
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>
```

```python
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
```

```python
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])
```

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

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**

```python
from sklearn.datasets import make_regression
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectKBest
```

```
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)

```
df_temp=df1
```

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

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

```
from google.colab import data_table

data_table.enable_dataframe_formatter()
data_table.DataTable(m1, include_index=True)
```

1 to 8 of 8 entries Filter

index	0
displacement	0.7834541975190548
weight	0.7752353889847736
horsepower	0.7173123149839054
Type	0.21358315745584155

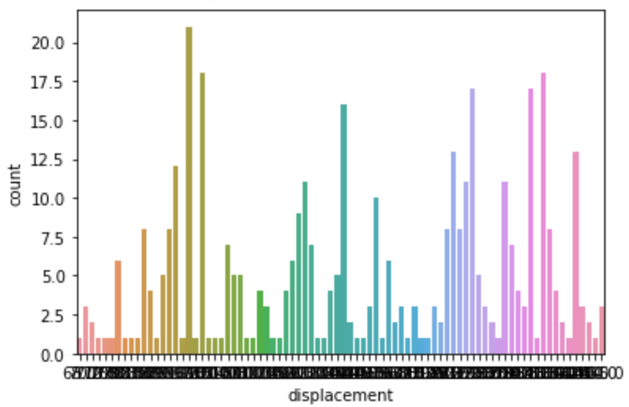
2-Mutual information as Assignment required(x=data,y=displacement)

```
df_temp1=df1
```

Double-click (or enter) to edit

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

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f569f1c1e10>
```




```
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)
```

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

```
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
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(df_temp1, df_temp1['displacement'],
    test_size=0.3,
    random_state=0)
from sklearn.feature_selection import mutual_info_regression
mutual_info = mutual_info_regression(X_train, y_train)
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

```
mutual_info = pd.Series(mutual_info)

mutual_info.index= X_train.columns

m2=mutual_info.sort_values(ascending=False).to_frame()
m2
```

1 to 9 of 9 entries Filter

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)

```
df_temp2=df1
```

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

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

```
mutual_info = pd.Series(mutual_info)

mutual_info.index= X_train.columns

m3=mutual_info.sort_values(ascending=False).to_frame()
m3
```

1 to 8 of 8 entries Filter

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

[ ] ↳ 3 cells hidden

mpg 0.5307099215450 65

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

Type 0.2027509161573695

```
df_temp4=df1
```

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

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

```
mutual_info = pd.Series(mutual_info)

mutual_info.index= X_train.columns

m5=mutual_info.sort_values(ascending=False).to_frame()
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)

```
df_temp5=df1
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(df_temp5.drop(labels=['acceleration'], axis=1),
    df_temp5['acceleration'],
    test_size=0.3,
    random_state=0)
from sklearn.feature_selection import mutual_info_regression
mutual_info = mutual_info_regression(X_train.fillna(0), y_train)
mutual_info
```

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

```
mutual_info = pd.Series(mutual_info)

mutual_info.index= X_train.columns

m5=mutual_info.sort_values(ascending=False).to_frame()
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)

```
df_temp6=df1
```

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

```
X_train,X_test,y_train,y_test=train_test_split(df_temp6.drop(labels=['origin'], axis=1),
    df_temp6['origin'],
    test_size=0.3,
    random_state=0)
from sklearn.feature_selection import mutual_info_regression
mutual_info = mutual_info_regression(X_train.fillna(0), y_train)
mutual_info
```

```
array([0.18172519, 0.32229184, 0.29801911, 0.32620247, 0.26334895,
       0.11576109, 0.02994163, 0.82833741])
```

```
mutual_info = pd.Series(mutual_info)

mutual_info.index= X_train.columns

m6=mutual_info.sort_values(ascending=False).to_frame()
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)

```
df_temp7=df1
```

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

```
mutual_info = mutual_info_regression(X_train.fillna(0), y_train)
```

```
array([0.3091082 , 0.08153791, 0.07698712, 0.23386751, 0.15912621,
       0.05708936, 0.04178806, 0.        ])
```

```
mutual_info = pd.Series(mutual_info)

mutual_info.index= X_train.columns

m7=mutual_info.sort_values(ascending=False).to_frame()
m7
```

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

```

```

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

```
df_temp8=df1
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(df_temp8.drop(labels=['Type'], axis=1),
    df_temp8['Type'],
    test_size=0.3,
    random_state=0)
from sklearn.feature_selection import mutual_info_regression
mutual_info = mutual_info_regression(X_train.fillna(0), y_train)
mutual_info
```

```
array([0.21585391, 0.2229849 , 0.33500552, 0.41323777, 0.17744422,
       0.02772262, 0.02386089, 0.85473016])
```

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

```

mutual_info.index= X_train.columns

m8=mutual_info.sort_values(ascending=False).to_frame()
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)

```
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

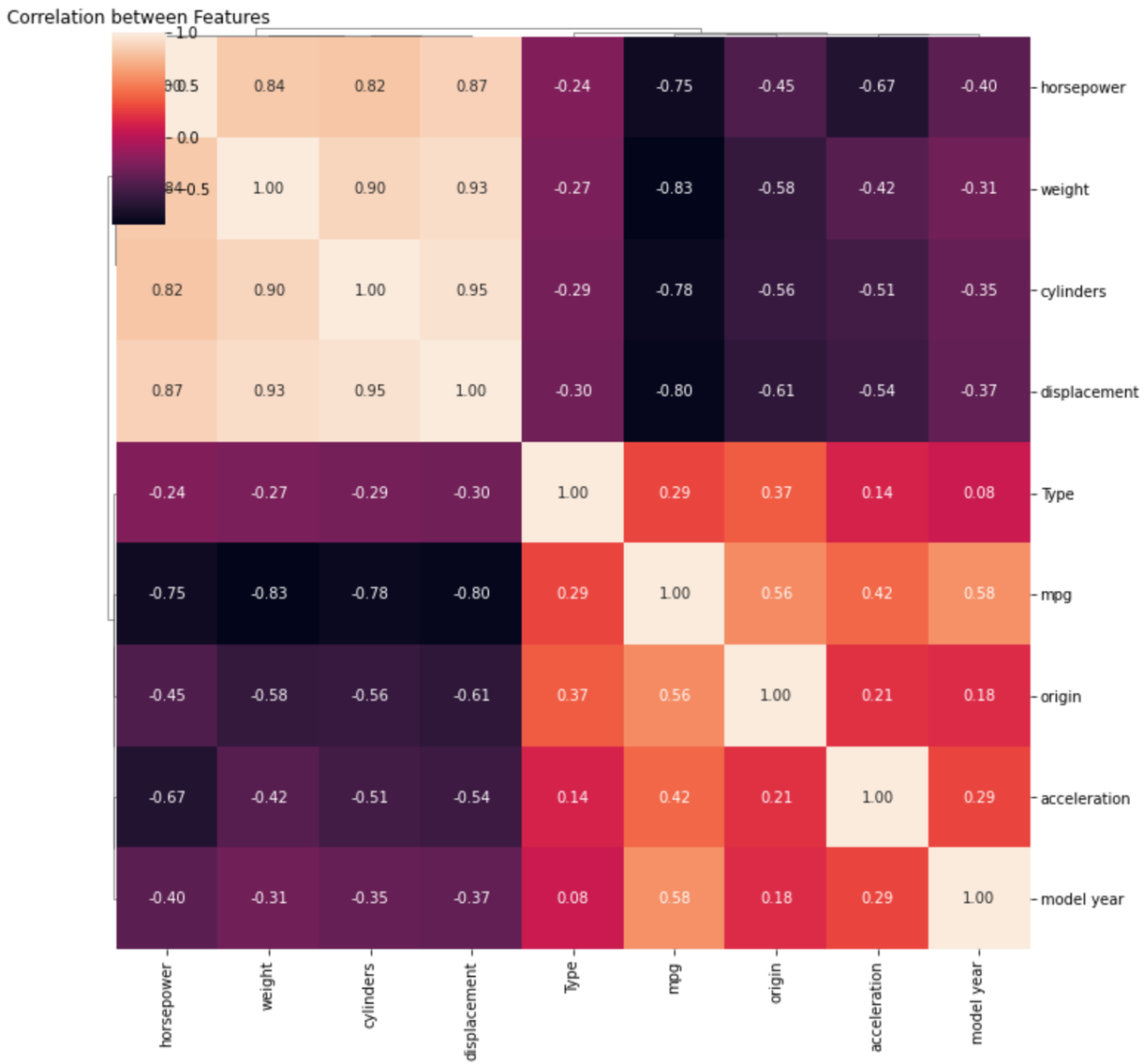
```
corr_matrix = df1.corr(method='pearson')


plt.figure(figsize=(25,25))
sns.clustermap(corr_matrix, annot=True, fmt = ".2f", dendrogram_ratio=0.01)

```

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

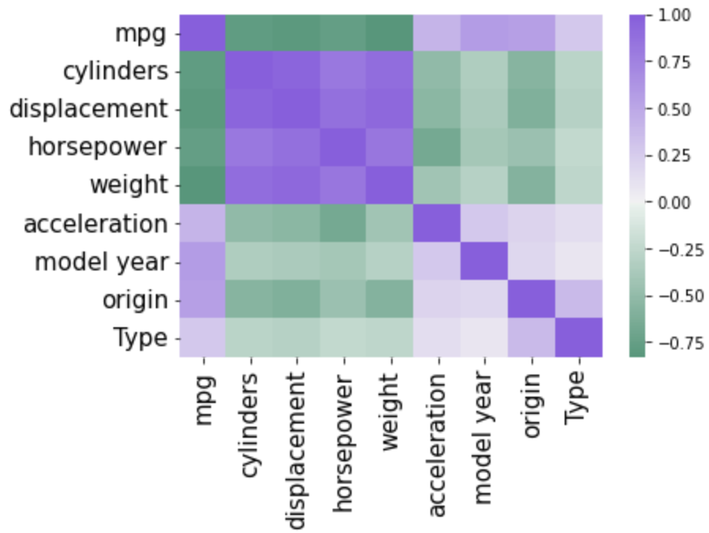
```
<Figure size 1800x1800 with 0 Axes>
```




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

# Create heatmap using the .corr method on df, set colormap to cmap
sns.heatmap(df1.corr(), center=0, cmap=colors)
plt.xticks(fontsize= 15)
plt.yticks(fontsize= 15)
```

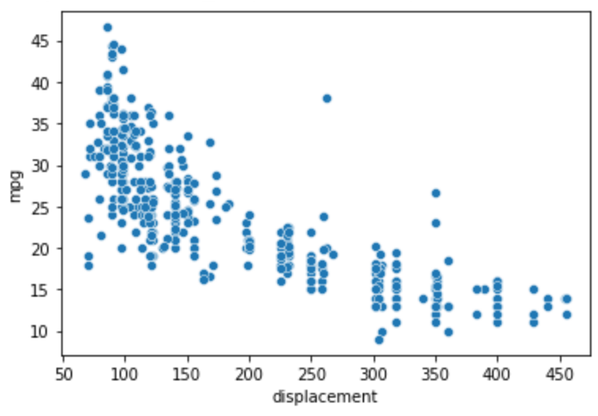
```
(array([0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5, 7.5, 8.5]),
 <a list of 9 Text major ticklabel objects>)
```




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

```

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

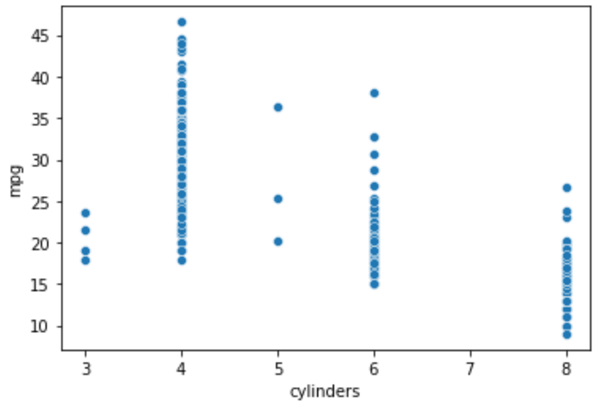



```
sns.scatterplot(x='horsepower', y='mpg', data=df)
```

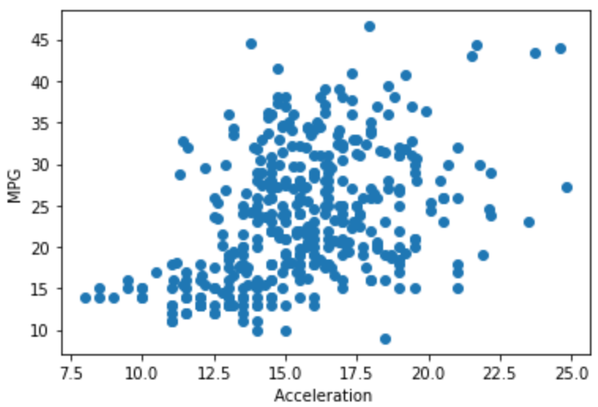
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f56a3fc9590>
```


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

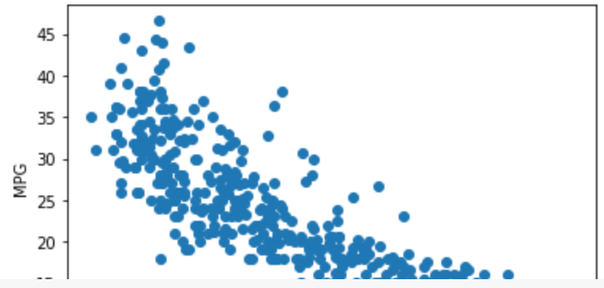
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f56a3ab7910>
```




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

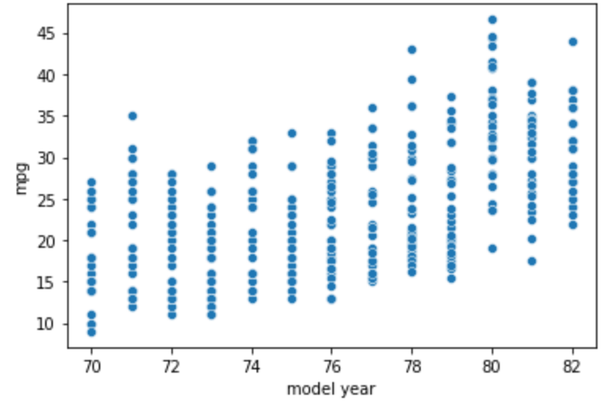



```
plt.scatter(x = df.weight, y = df.mpg)
plt.xlabel('Weight ')
plt.ylabel('MPG')
plt.show()
```

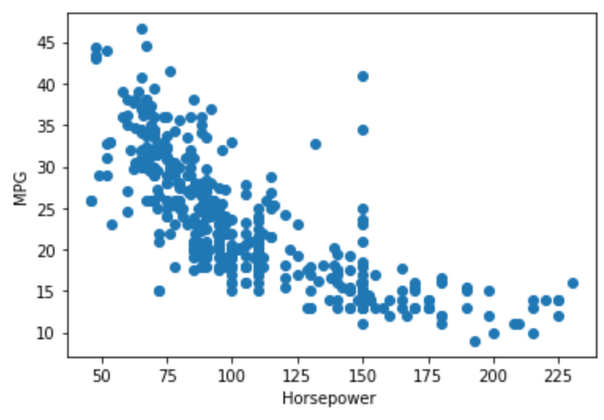



```
sns.scatterplot(x='model year', y='mpg', data=df)
```

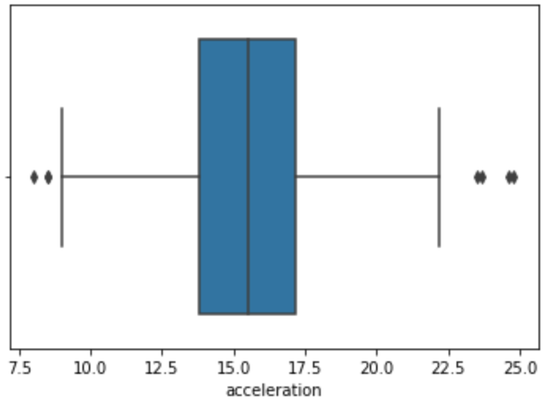
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f56a39e9790>
```



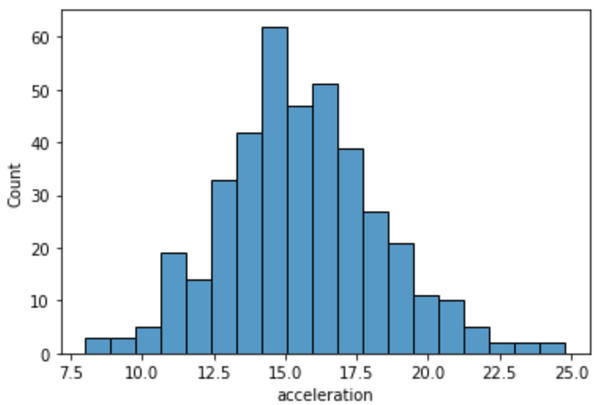

```
plt.scatter(x = df.horsepower, y = df.mpg)
plt.xlabel('Horsepower ')
plt.ylabel('MPG')
plt.show()
```




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



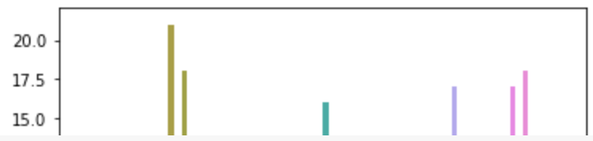

```
sns.histplot(x='acceleration', data=df)
plt.show()
plt.close()
```




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

```
2.1425696320799963
```

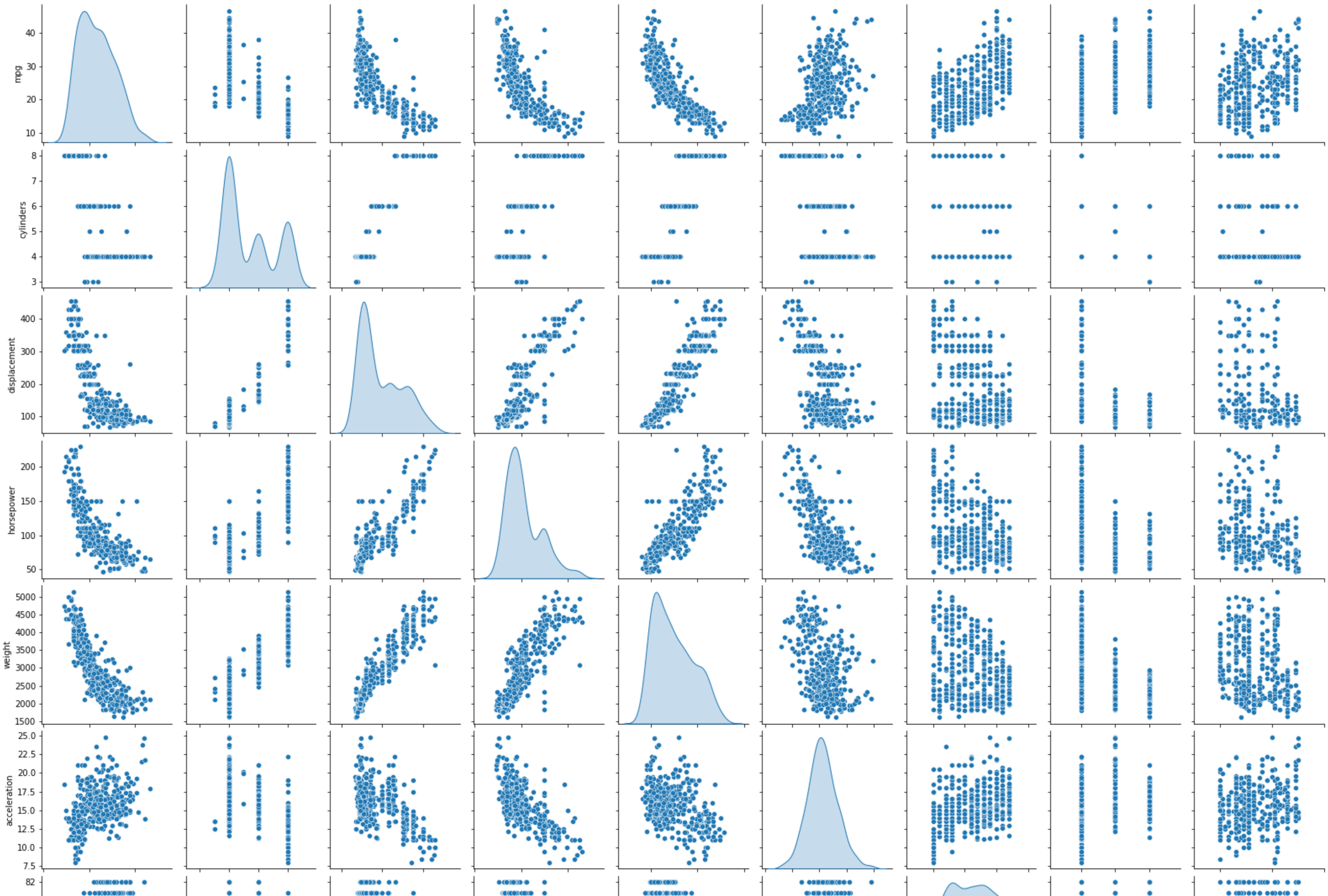
```
sns.countplot(x='displacement', data=df)
plt.show()
plt.close()
```

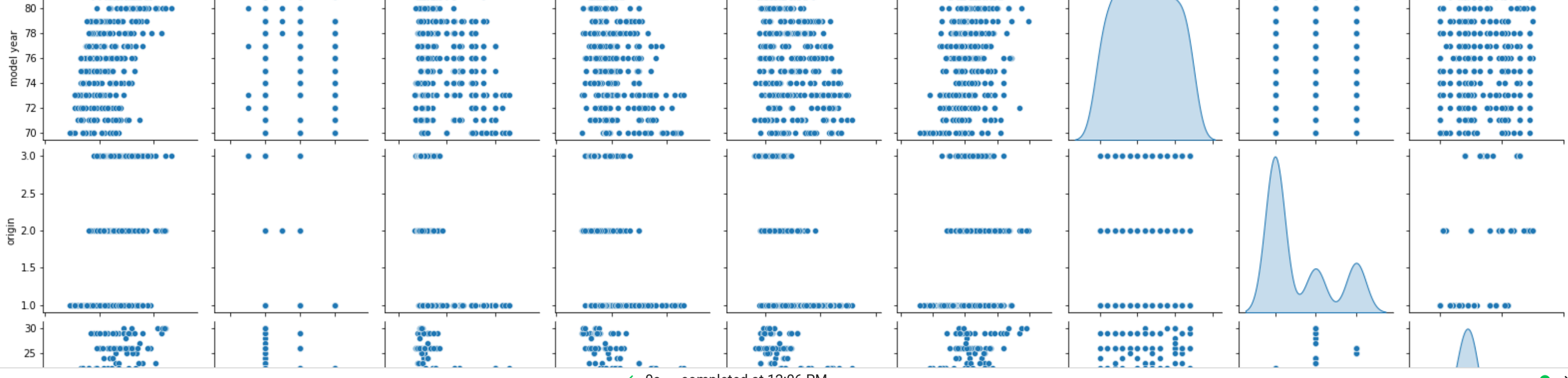



```
df.groupby('mpg').acceleration.max()
```

```
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
```

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


mpg
cylinders
displacement
horsepower
weight
acceleration



✓ 0s completed at 12:06 PM