Matplotlib_and_seabon

October 16, 2021

[ ]:
```
import numpy as np
import pandas as pd
```

[ ]:
```
import seaborn as sns
```

[ ]:
```
import matplotlib.pyplot as plt
```

[ ]:
```
stocks= pd.read_csv('stocks.csv')
```

[ ]:
```
stocks.head()
```

[ ]:

	Date	AAPL	IBM	CSCO	MSFT
0	2000-01-03	111.937502	116.0000	108.0625	116.5625
1	2000-01-04	102.500003	112.0625	102.0000	112.6250
2	2000-01-05	103.999997	116.0000	101.6875	113.8125
3	2000-01-06	94.999998	114.0000	100.0000	110.0000
4	2000-01-07	99.500001	113.5000	105.8750	111.4375

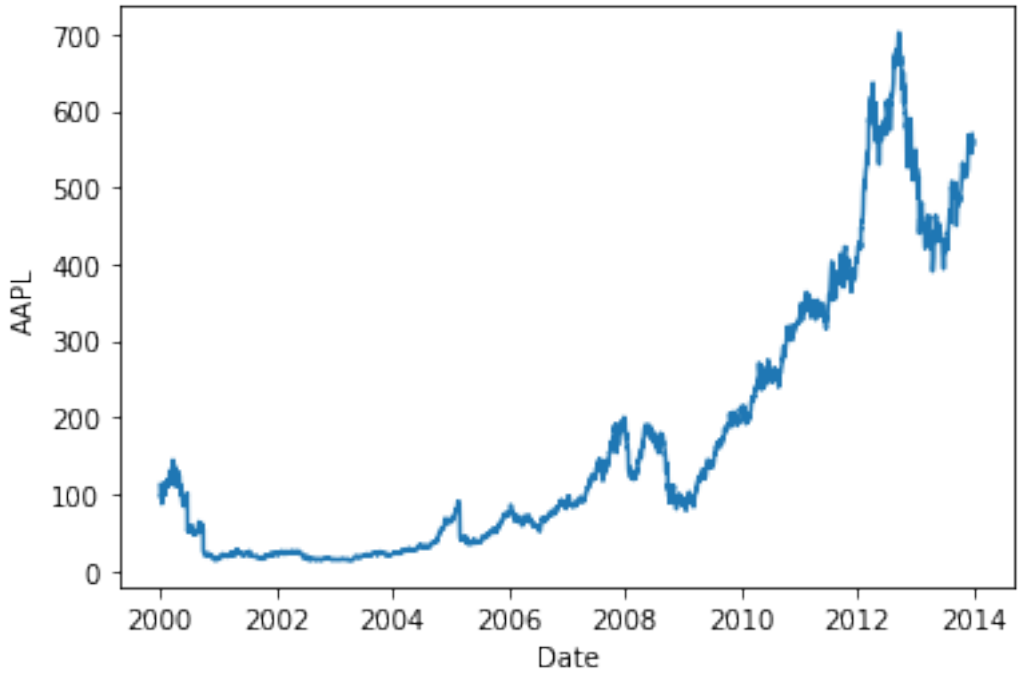
[ ]:
```
stocks['Date']=pd.to_datetime(stocks['Date'])
stocks.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3521 entries, 0 to 3520
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    3521 non-null   datetime64[ns]
 1   AAPL    3521 non-null   float64
 2   IBM     3521 non-null   float64
 3   CSCO    3521 non-null   float64
 4   MSFT    3521 non-null   float64
dtypes: datetime64[ns](1), float64(4)
memory usage: 137.7 KB
```

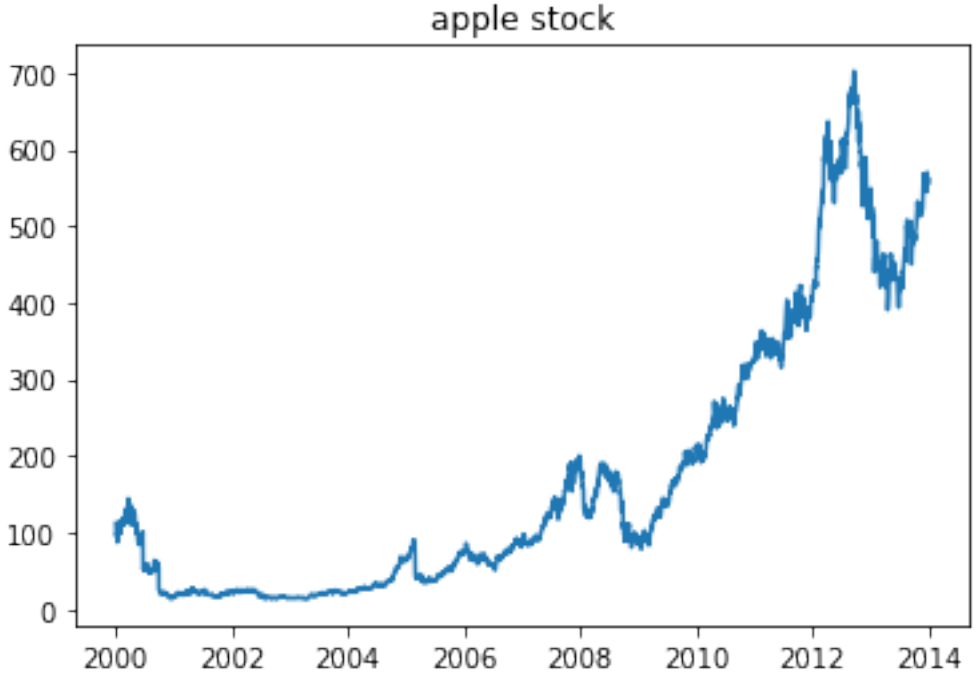
0.1 Line Plot

[ ]:
```
sns.lineplot(x='Date',y='AAPL',data=stocks)
```

[ ]:
```
<AxesSubplot:xlabel='Date', ylabel='AAPL'>
```

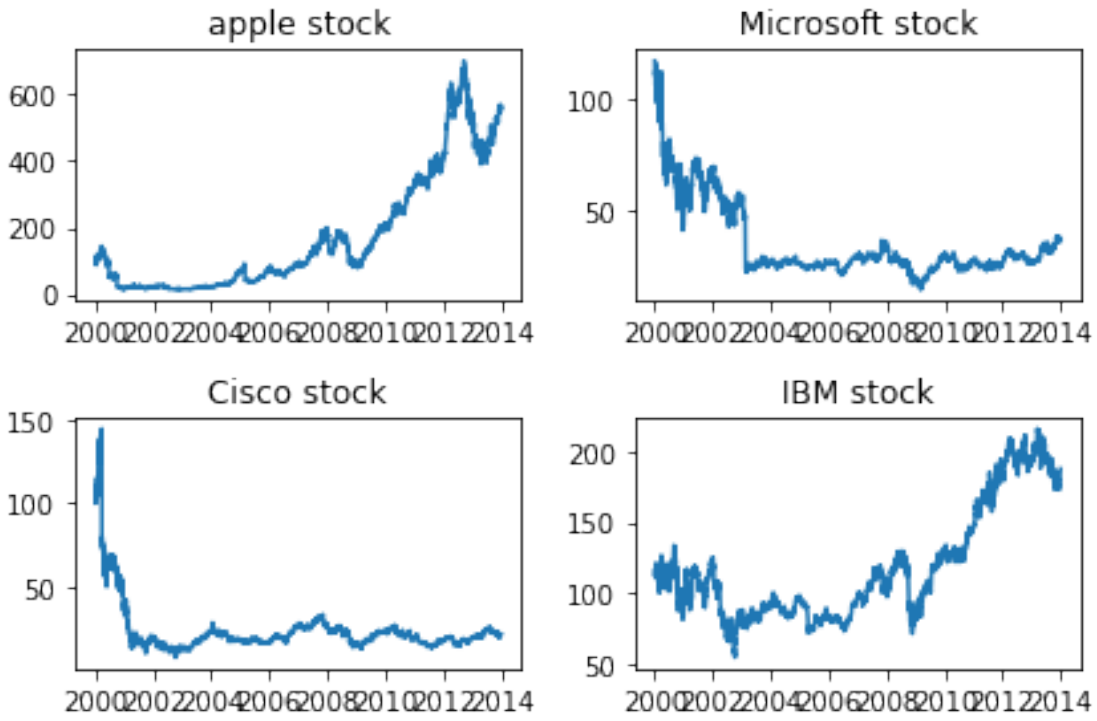


[ ]:
```
# Now using matplotlib
plt.plot(stocks['Date'],stocks['AAPL'])
plt.title('apple stock')
plt.show()
```



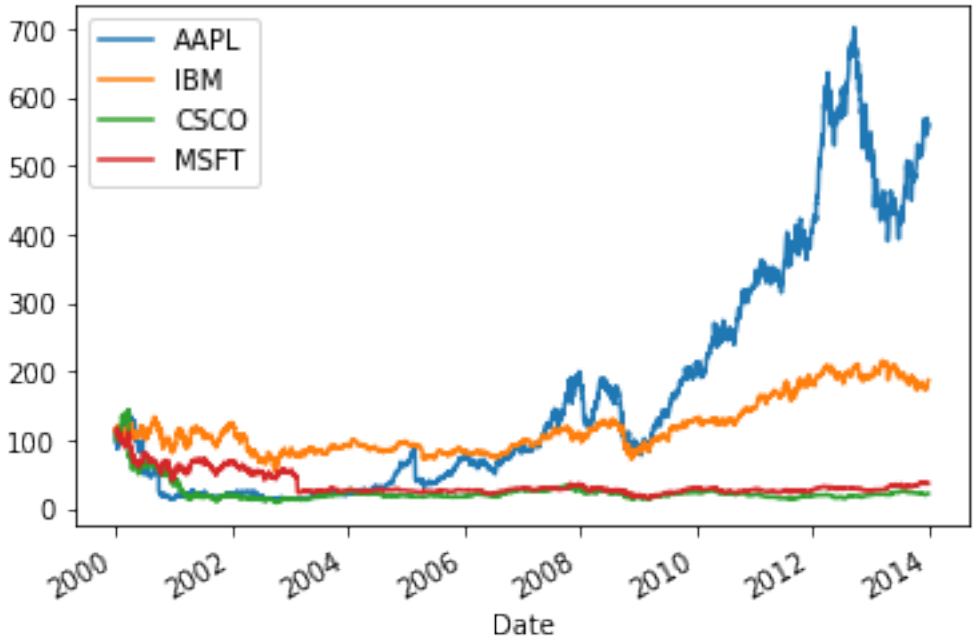
[ ]:
```
plt.subplot(2,2,1)
plt.plot(stocks['Date'],stocks['AAPL'])
plt.title('apple stock')

plt.subplot(2,2,2)
plt.plot(stocks['Date'],stocks['MSFT'])
plt.title('Microsoft stock')

plt.subplot(2,2,3)
plt.plot(stocks['Date'],stocks['CSCO'])
plt.title('Cisco stock')

plt.subplot(2,2,4)
plt.plot(stocks['Date'],stocks['IBM'])
plt.title('IBM stock')

plt.tight_layout()
plt.show
```

[ ]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



[ ]:
```
stocks= stocks.set_index('Date')
stocks.plot()
```

[ ]:
```
<AxesSubplot:xlabel='Date'>
```



0.2 Scatterplots

captures relationship between 2 continues variables

[ ]:
```
cars=pd.read_csv('cars.csv')
cars.head()
```

[ ]:

	name	sports_car	suv	wagon	minivan	pickup	\
0	Chevrolet Aveo 4dr	False	False	False	False	False	
1	Chevrolet Aveo LS 4dr hatch	False	False	False	False	False	
2	Chevrolet Cavalier 2dr	False	False	False	False	False	
3	Chevrolet Cavalier 4dr	False	False	False	False	False	
4	Chevrolet Cavalier LS 2dr	False	False	False	False	False	

	all_wheel	rear_wheel	Price	Dealer_Cost	Engine_size	cylenders	\
0	False	False	11690	10965	1.6	4	
1	False	False	12585	11802	1.6	4	
2	False	False	14610	13697	2.2	4	
3	False	False	14810	13884	2.2	4	
4	False	False	16385	15357	2.2	4	

```
   horsepower  city_miles_per_galloon  highway_miles_per_Gallon  weight  \
0         103                    28.0                      34.0  2370.0
1         103                    28.0                      34.0  2348.0
2         140                    26.0                      37.0  2617.0
3         140                    26.0                      37.0  2676.0
4         140                    26.0                      37.0  2617.0

   base_wheeel  length  width
0         98.0   167.0   66.0
1         98.0   153.0   66.0
2        104.0   183.0   69.0
3        104.0   183.0   68.0
4        104.0   183.0   69.0
```

[ ]:
```
cars.shape
```

[ ]:
```
(428, 19)
```

[ ]:
```
cylenders= cars.cylenders.value_counts()
cylenders
```

[ ]:
```
 6     190
 4     136
 8      87
 5       7
 12      3
-1       2
 10      2
 3       1
Name: cylenders, dtype: int64
```

notice that cylinders 4, 6 and 8 is the mejority

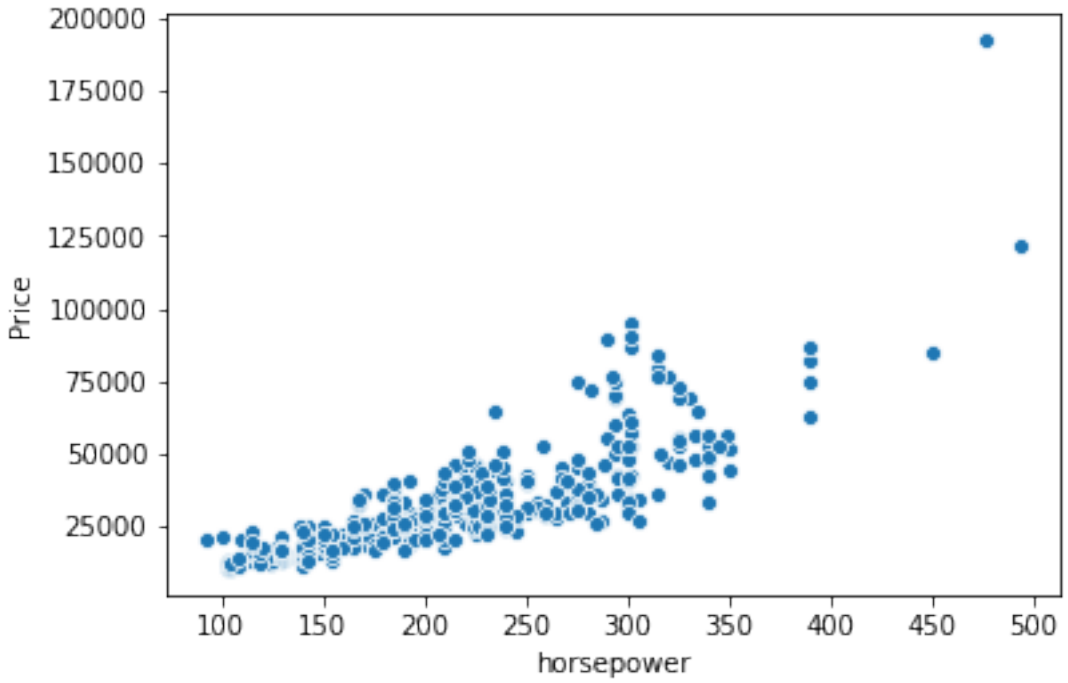
[ ]:
```
cars_common= cars[cars.cylenders.isin([4,6,8])]
cars_common
```

[ ]:
```
                              name  sports_car    suv  wagon  minivan  \
0                 Chevrolet Aveo 4dr       False  False  False    False
1        Chevrolet Aveo LS 4dr hatch       False  False  False    False
2             Chevrolet Cavalier 2dr       False  False  False    False
3             Chevrolet Cavalier 4dr       False  False  False    False
4           Chevrolet Cavalier LS 2dr       False  False  False    False
..                                …         …    …     …      …
423         Nissan Titan King Cab XE       False  False  False    False
424                      Subaru Baja       False  False  False    False
425                   Toyota Tacoma       False  False  False    False
426     Toyota Tundra Regular Cab V6       False  False  False    False
427  Toyota Tundra Access Cab V6 SR5       False  False  False    False
```

```
     pickup  all_wheel  rear_wheel  Price  Dealer_Cost  Engine_size  \
0     False      False       False  11690        10965          1.6
1     False      False       False  12585        11802          1.6
2     False      False       False  14610        13697          2.2
3     False      False       False  14810        13884          2.2
4     False      False       False  16385        15357          2.2
..      …          …          …     …            …              …
423    True       True       False  26650        24926          5.6
424    True       True       False  24520        22304          2.5
425    True      False        True  12800        11879          2.4
426    True      False        True  16495        14978          3.4
427    True       True       False  25935        23520          3.4

     cylenders  horsepower  city_miles_per_galloon  highway_miles_per_Gallon  \
0            4         103                    28.0                      34.0
1            4         103                    28.0                      34.0
2            4         140                    26.0                      37.0
3            4         140                    26.0                      37.0
4            4         140                    26.0                      37.0
..         …          …                      …                         …
423          8         305                    14.0                      18.0
424          4         165                    21.0                      28.0
425          4         142                    22.0                      27.0
426          6         190                    16.0                      20.0
427          6         190                    14.0                      17.0

     weight  base_wheeel  length  width
0    2370.0         98.0   167.0   66.0
1    2348.0         98.0   153.0   66.0
2    2617.0        104.0   183.0   69.0
3    2676.0        104.0   183.0   68.0
4    2617.0        104.0   183.0   69.0
..      …          …      …       …
423  5287.0        140.0     NaN     NaN
424  3485.0        104.0     NaN     NaN
425  2750.0        103.0     NaN     NaN
426  3925.0        128.0     NaN     NaN
427  4435.0        128.0     NaN     NaN

[413 rows x 19 columns]
```

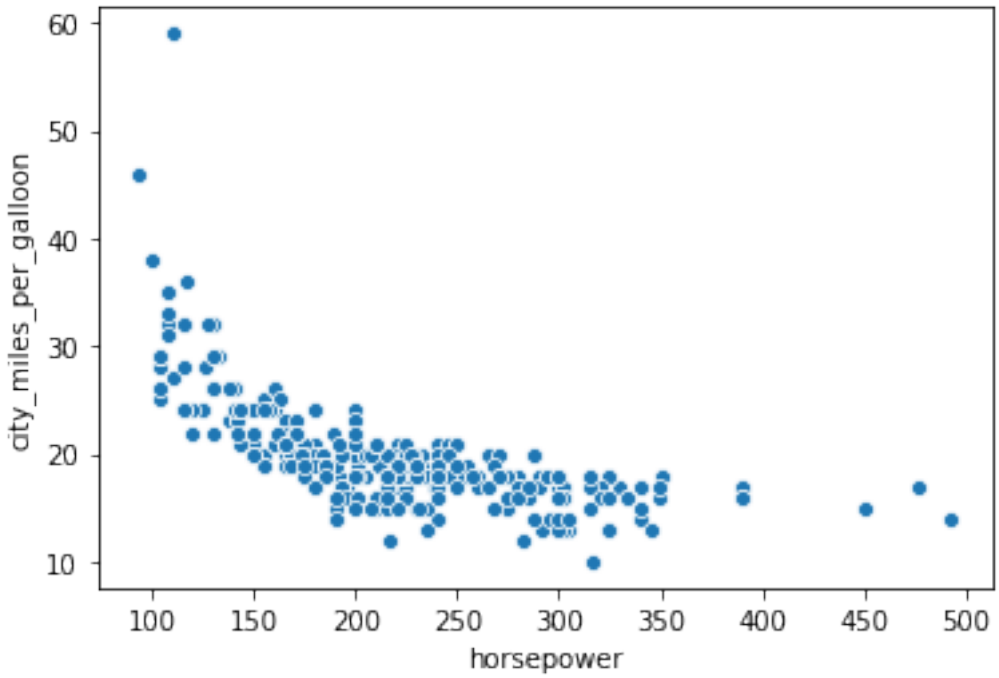
[ ]:
```
sns.scatterplot(x='horsepower',y='Price',data=cars_common)
```

[ ]:
```
<AxesSubplot:xlabel='horsepower', ylabel='Price'>
```



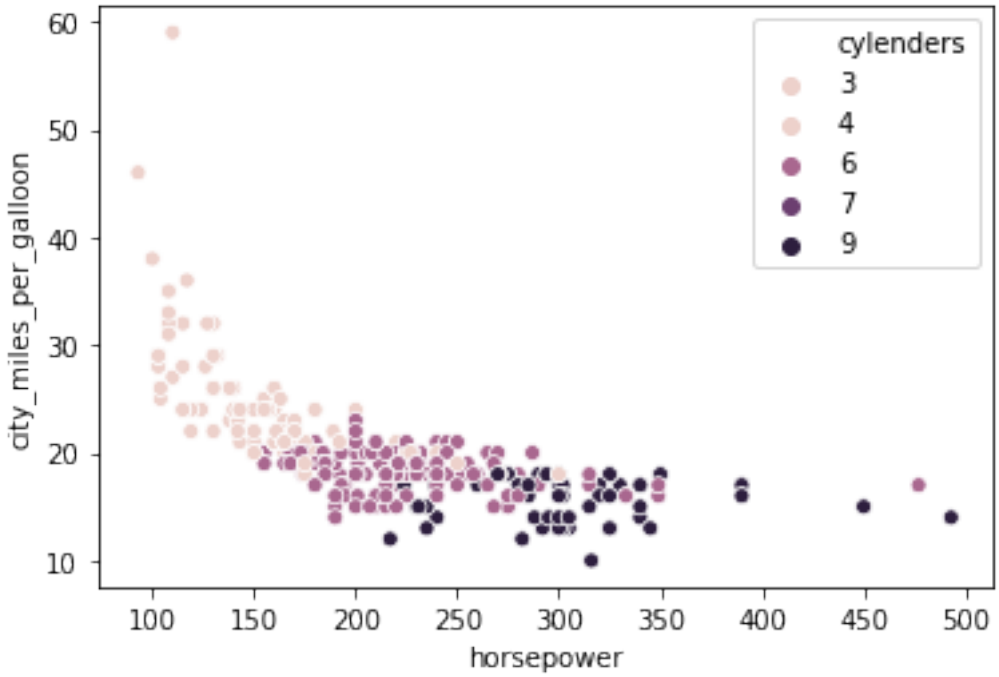
[ ]:
```
sns.scatterplot(x='horsepower',y='city_miles_per_galloon',data=cars_common)
```

[ ]:
```
<AxesSubplot:xlabel='horsepower', ylabel='city_miles_per_galloon'>
```



[ ]:
```
sns.
↪scatterplot(x='horsepower',y='city_miles_per_galloon',data=cars_common,hue='cylenders')
```

[ ]: <AxesSubplot:xlabel='horsepower', ylabel='city_miles_per_galloon'>



[ ]:
```
cars_common[['cylenders']].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 413 entries, 0 to 427
Data columns (total 1 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   cylenders  413 non-null    int64
dtypes: int64(1)
memory usage: 22.6 KB
```

Now change to categories, because cylenders is integer

[ ]:
```
cars_common['cylenders']=cars_common['cylenders'].astype('category')
cars_common[['cylenders']].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 413 entries, 0 to 427
Data columns (total 1 columns):
 #   Column     Non-Null Count  Dtype
```

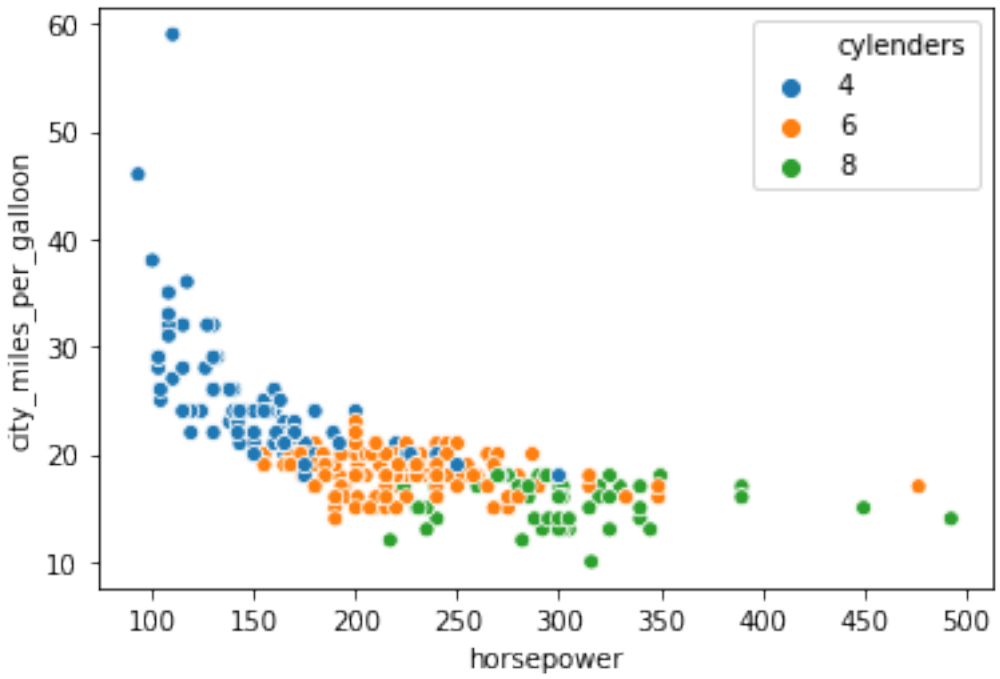
```
---  ------     --------------  -----
 0   cylenders  413 non-null    category
dtypes: category(1)
memory usage: 19.9 KB
```

```
C:\Users\aduzo\Anaconda3\lib\site-packages\ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

Cylenders is now categorical type

[ ]:
```
sns.
↪scatterplot(x='horsepower',y='city_miles_per_galloon',data=cars_common,hue='cylenders')
```

[ ]:
```
<AxesSubplot:xlabel='horsepower', ylabel='city_miles_per_galloon'>
```



0.3 Count plot

[ ]:
```
#### barplots
retail= pd.read_csv('online_retail2.csv')
retail= retail.drop_duplicates()
retail= retail.dropna(axis=0,how='any')
retail.Country.value_counts()
```

[ ]:
```
United Kingdom          716115
Germany                  17339
EIRE                     16014
France                   13897
Netherlands               5137
Spain                     3754
Belgium                   3110
Switzerland               3058
Portugal                  2414
Australia                 1890
Channel Islands           1646
Italy                     1507
Sweden                    1343
Norway                    1308
Cyprus                    1157
Finland                   1049
Austria                    938
Denmark                    797
Greece                     663
Japan                      565
USA                        535
Poland                     527
Unspecified                521
United Arab Emirates       386
Singapore                  346
Israel                     321
Malta                      299
Canada                     228
Iceland                    222
Lithuania                  154
RSA                        123
Brazil                      94
Thailand                    76
Korea                       63
European Community          61
Bahrain                     59
West Indies                 54
Lebanon                     45
Nigeria                     30
```

```
Czech Republic              30
Saudi Arabia                10
Name: Country, dtype: int64
```

[ ]:
```
retail.shape
```

[ ]:
```
(797885, 8)
```

[ ]:
```
ire_Fran= retail[retail.Country.isin(['EIRE','France'])]
ire_Fran
```

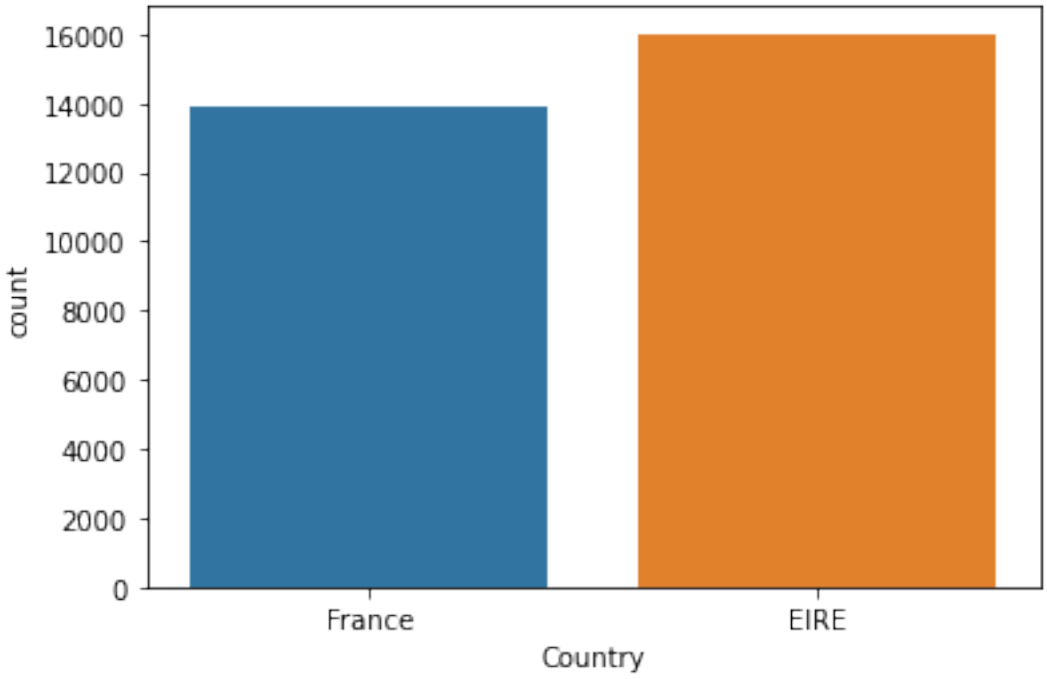
[ ]:
```
        Invoice StockCode                          Description  Quantity  \
71       489439     22065     CHRISTMAS PUDDING TRINKET POT          12
72       489439     22138      BAKING SET 9 PIECE RETROSPOT           9
73       489439     22139  RETRO SPOT TEA SET CERAMIC 11 PC           9
74       489439     22352   LUNCHBOX WITH CUTLERY RETROSPOT          12
75       489439    85014A   BLACK/BLUE DOTS RUFFLED UMBRELLA          3
…          …       …                                    …       …
1067366  581587     22899       CHILDREN'S APRON DOLLY GIRL           6
1067367  581587     23254      CHILDRENS CUTLERY DOLLY GIRL           4
1067368  581587     23255    CHILDRENS CUTLERY CIRCUS PARADE          4
1067369  581587     22138      BAKING SET 9 PIECE RETROSPOT           3
1067370  581587      POST                              POSTAGE         1

                 InvoiceDate  Price  Customer ID Country
71       2009-12-01 09:28:00   1.45      12682.0  France
72       2009-12-01 09:28:00   4.95      12682.0  France
73       2009-12-01 09:28:00   4.95      12682.0  France
74       2009-12-01 09:28:00   2.55      12682.0  France
75       2009-12-01 09:28:00   5.95      12682.0  France
…                      …    …           …      …
1067366  2011-12-09 12:50:00   2.10      12680.0  France
1067367  2011-12-09 12:50:00   4.15      12680.0  France
1067368  2011-12-09 12:50:00   4.15      12680.0  France
1067369  2011-12-09 12:50:00   4.95      12680.0  France
1067370  2011-12-09 12:50:00  18.00      12680.0  France

[29911 rows x 8 columns]
```

[ ]:
```
ire_Fran.shape
```

[ ]:
```
(29911, 8)
```

[ ]:
```
sns.countplot(x='Country',data= ire_Fran)
```

[ ]:
```
<AxesSubplot:xlabel='Country', ylabel='count'>
```



[ ]:
```
ire_Fran['InvoiceDate'].head()
```

[ ]:
```
71    2009-12-01 09:28:00
72    2009-12-01 09:28:00
73    2009-12-01 09:28:00
74    2009-12-01 09:28:00
75    2009-12-01 09:28:00
Name: InvoiceDate, dtype: object
```

0.4 Barplot

[ ]:
```
from datetime import datetime
```

[ ]:
```
ire_Fran[['InvoiceDate']].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 29911 entries, 71 to 1067370
Data columns (total 1 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   InvoiceDate  29911 non-null  object
dtypes: object(1)
memory usage: 1.5+ MB
```

This is an object, convert it to datetime

[ ]:
```
ire_Fran['InvoiceDate']=pd.to_datetime(ire_Fran['InvoiceDate'])
```

```
C:\Users\aduzo\Anaconda3\lib\site-packages\ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

[ ]:
```
ire_Fran[['InvoiceDate']].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 29911 entries, 71 to 1067370
Data columns (total 1 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   InvoiceDate  29911 non-null  datetime64[ns]
dtypes: datetime64[ns](1)
memory usage: 1.5 MB
```

Now it is of type datetime

[ ]:
```
ire_Fran['dayofweek']= ire_Fran['InvoiceDate'].dt.dayofweek
ire_Fran['dayofweek']
```

```
C:\Users\aduzo\Anaconda3\lib\site-packages\ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

[ ]:
```
71         1
72         1
73         1
74         1
75         1
          ..
1067366    4
1067367    4
1067368    4
1067369    4
1067370    4
Name: dayofweek, Length: 29911, dtype: int64
```

Create a dictionary and label days of week

[ ]:
```
week_dict= { 0 :'Monday',1: 'Tuesday',2: 'Wednesday',3: 'thursday',4:␣
↪'Friday',5: 'Saturday',6: 'Sunday'}
week_dict
```

[ ]:
```
{0: 'Monday',
 1: 'Tuesday',
 2: 'Wednesday',
 3: 'thursday',
 4: 'Friday',
 5: 'Saturday',
 6: 'Sunday'}
```

[ ]:
```
ire_Fran['day_of_wk']=ire_Fran['dayofweek'].map(week_dict)
ire_Fran['day_of_wk'].head()
```

```
C:\Users\aduzo\Anaconda3\lib\site-packages\ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

[ ]:
```
71    Tuesday
72    Tuesday
73    Tuesday
74    Tuesday
75    Tuesday
Name: day_of_wk, dtype: object
```

[ ]:
```
grouped= ire_Fran.groupby(['Country','day_of_wk']).agg(total_quantity=␣
↪('Quantity',np.sum)).reset_index()
grouped=grouped[grouped.total_quantity >=0]
```

[ ]:
```
grouped
```

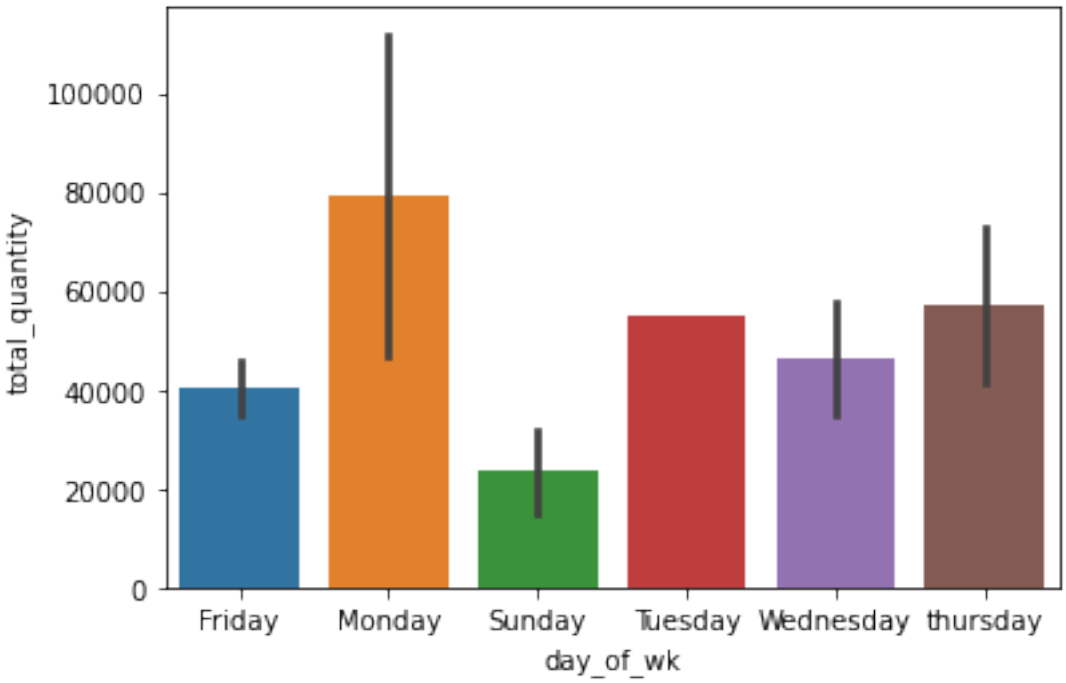
[ ]:

	Country	day_of_wk	total_quantity
0	EIRE	Friday	45835
1	EIRE	Monday	47061
2	EIRE	Sunday	31796
3	EIRE	Tuesday	54930
4	EIRE	Wednesday	57443
5	EIRE	thursday	72652
6	France	Friday	34814
7	France	Monday	111764
8	France	Sunday	15321
10	France	Wednesday	35035

```
11  France   thursday          41462
```

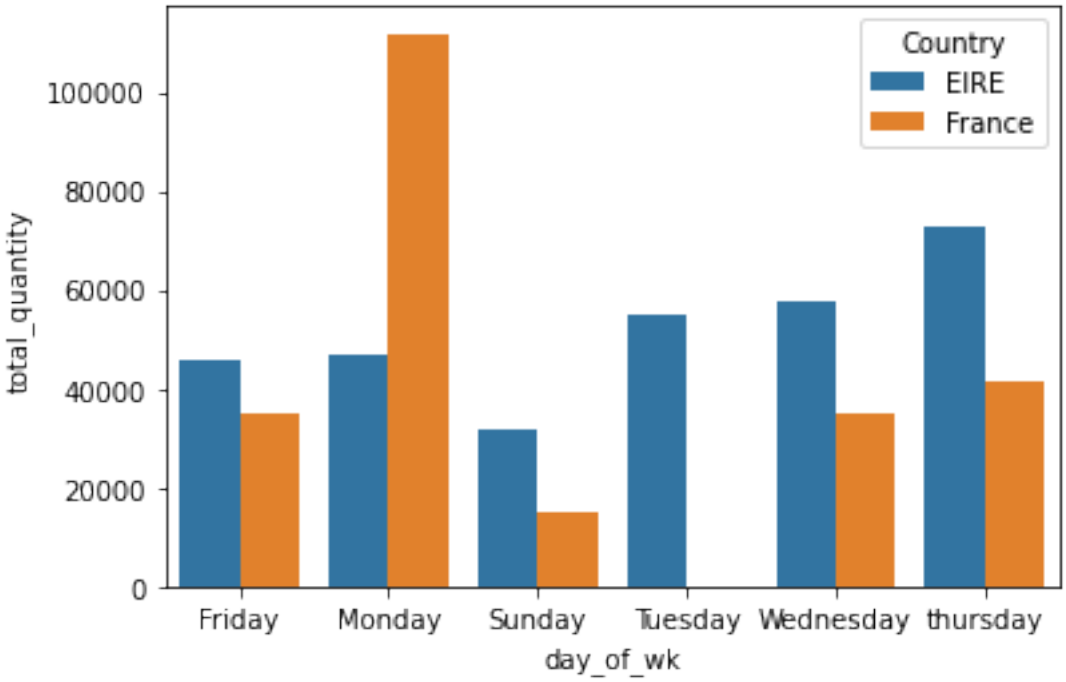
[ ]:
```
sns.barplot(x='day_of_wk',y='total_quantity',data=grouped)
```

[ ]:
```
<AxesSubplot:xlabel='day_of_wk', ylabel='total_quantity'>
```



[ ]:
```
sns.barplot(x='day_of_wk',y='total_quantity',data=grouped,hue='Country')
```

[ ]:
```
<AxesSubplot:xlabel='day_of_wk', ylabel='total_quantity'>
```



[ ]:
```
ire_Fran.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 29911 entries, 71 to 1067370
Data columns (total 10 columns):
 #   Column       Non-Null Count  Dtype         
---  ------       --------------  -----         
 0   Invoice      29911 non-null  object        
 1   StockCode    29911 non-null  object        
 2   Description  29911 non-null  object        
 3   Quantity     29911 non-null  int64         
 4   InvoiceDate  29911 non-null  datetime64[ns]
 5   Price        29911 non-null  float64       
 6   Customer ID  29911 non-null  float64       
 7   Country      29911 non-null  object        
 8   dayofweek    29911 non-null  int64         
 9   day_of_wk    29911 non-null  object        
dtypes: datetime64[ns](1), float64(2), int64(2), object(5)
memory usage: 3.5+ MB
```

0.5 Distribution Plot

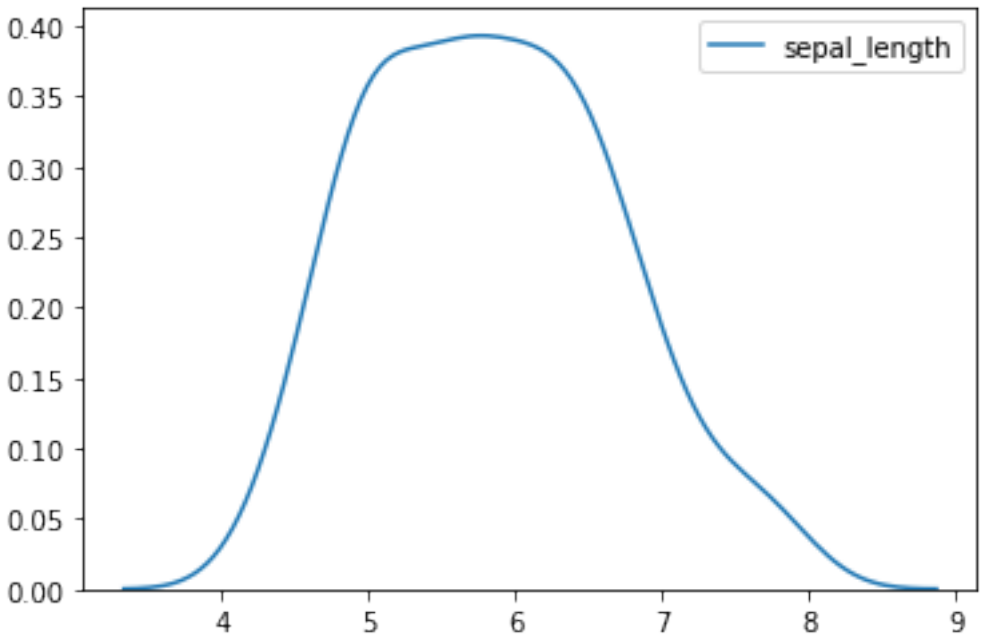
[ ]:
```
iris= pd.read_csv('iris.csv')
iris.head()
```

[ ]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

[ ]:
```
sns.kdeplot(iris.sepal_length)
```

[ ]:
```
<AxesSubplot:>
```



Sepal length pair each type of flower

[ ]:
```
iris.species.value_counts()
```

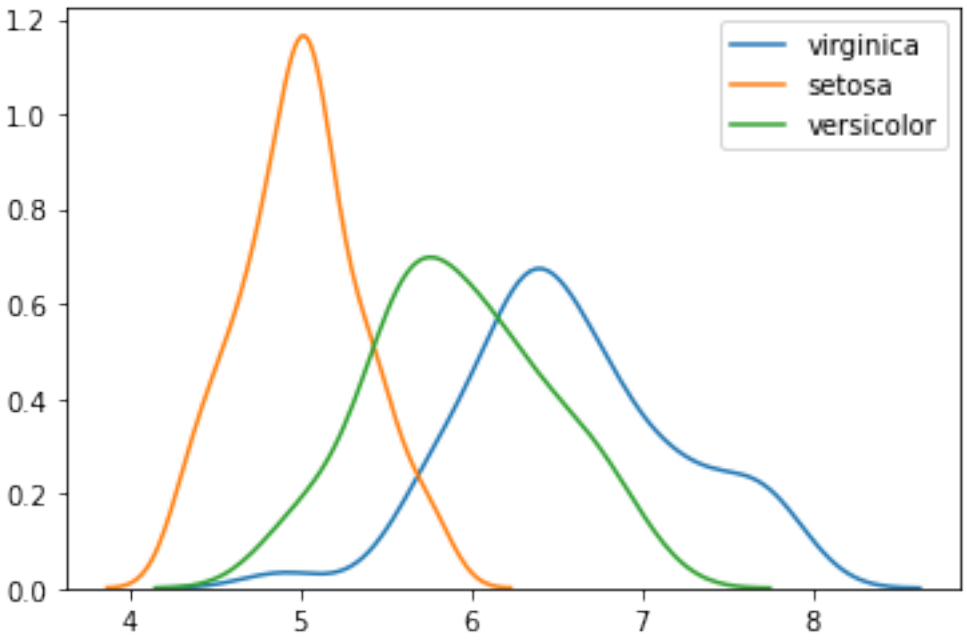
[ ]:
```
versicolor    50
virginica     50
setosa        50
Name: species, dtype: int64
```

[ ]:
```
iris.sepal_length[iris.species=='virginica'].count()
```

[ ]: 50

[ ]:
```
sns.kdeplot(iris.sepal_length[iris.species=='virginica'],label='virginica')
sns.kdeplot(iris.sepal_length[iris.species=='setosa'],label='setosa')
sns.kdeplot(iris.sepal_length[iris.species=='versicolor'],label='versicolor')
```

[ ]: <AxesSubplot:>



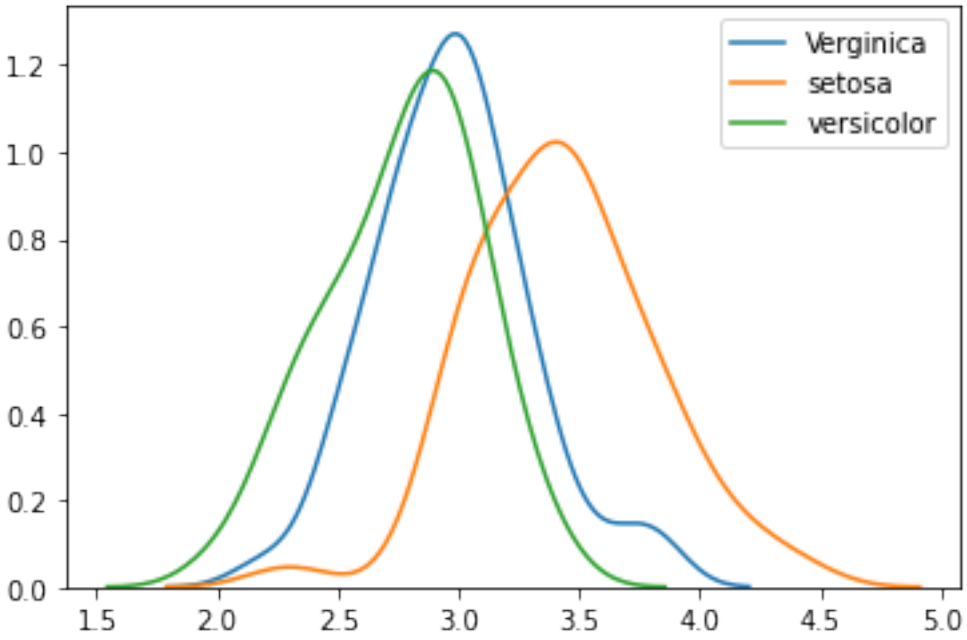
in general, The sepal length of the setosa is lower than the sepal length of versicolor and sepal length of versicolor is lower virginica

[ ]:
```
iris.sepal_length.describe()
```

[ ]:
```
count    150.000000
mean       5.843333
std        0.828066
min        4.300000
25%        5.100000
50%        5.800000
75%        6.400000
max        7.900000
Name: sepal_length, dtype: float64
```

[ ]:
```
sns.kdeplot(iris.sepal_width[iris.species=='virginica'],label='Verginica')
sns.kdeplot(iris.sepal_width[iris.species=='setosa'],label='setosa')
sns.kdeplot(iris.sepal_width[iris.species=='versicolor'],label='versicolor')
```

[ ]: <AxesSubplot:>



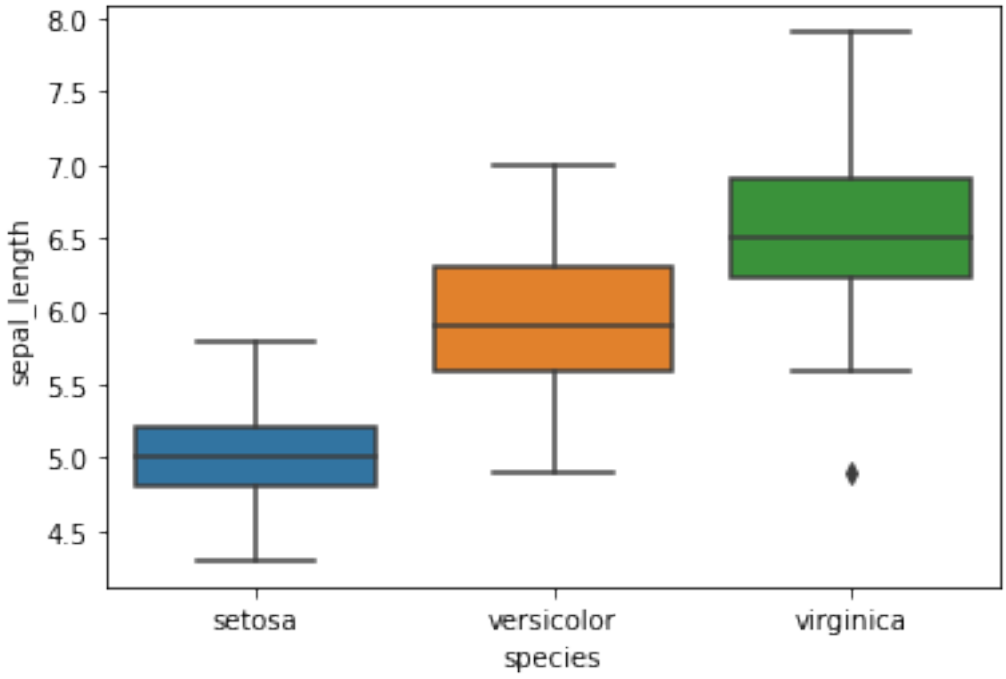
The setosa is more higher in width than the other 2.

0.6 Boxplot

we will see the outliers and percentiles using the Boxplot

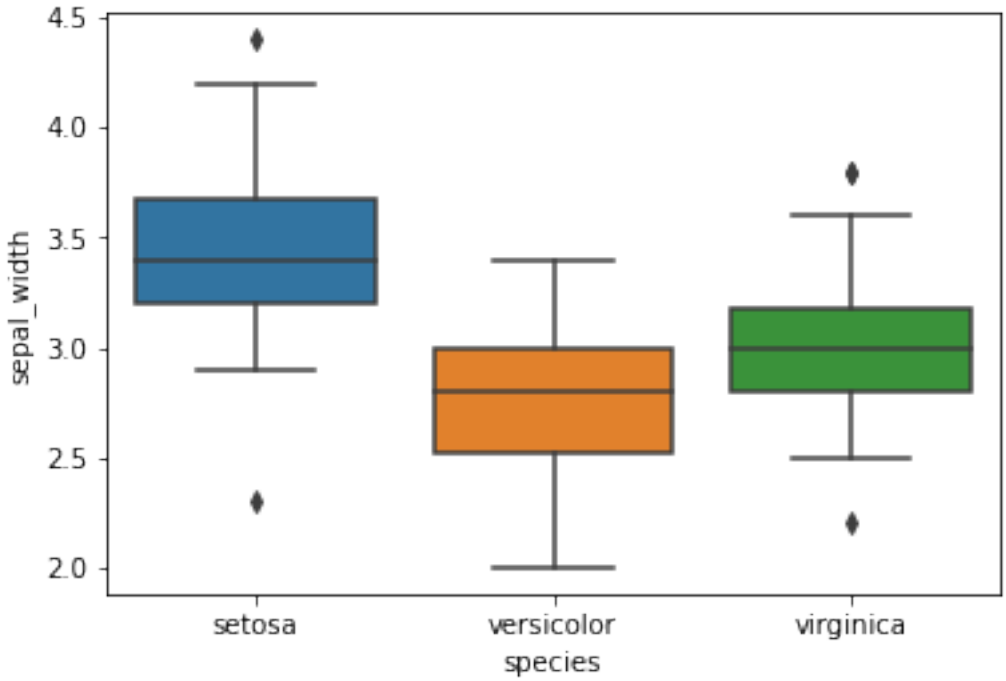
[ ]:
```
sns.boxplot(x ="species", y="sepal_length",data=iris)
```

[ ]: <AxesSubplot:xlabel='species', ylabel='sepal_length'>



[ ]:
```
sns.boxplot(x= 'species',y='sepal_width',data=iris)
```

[ ]:
```
<AxesSubplot:xlabel='species', ylabel='sepal_width'>
```



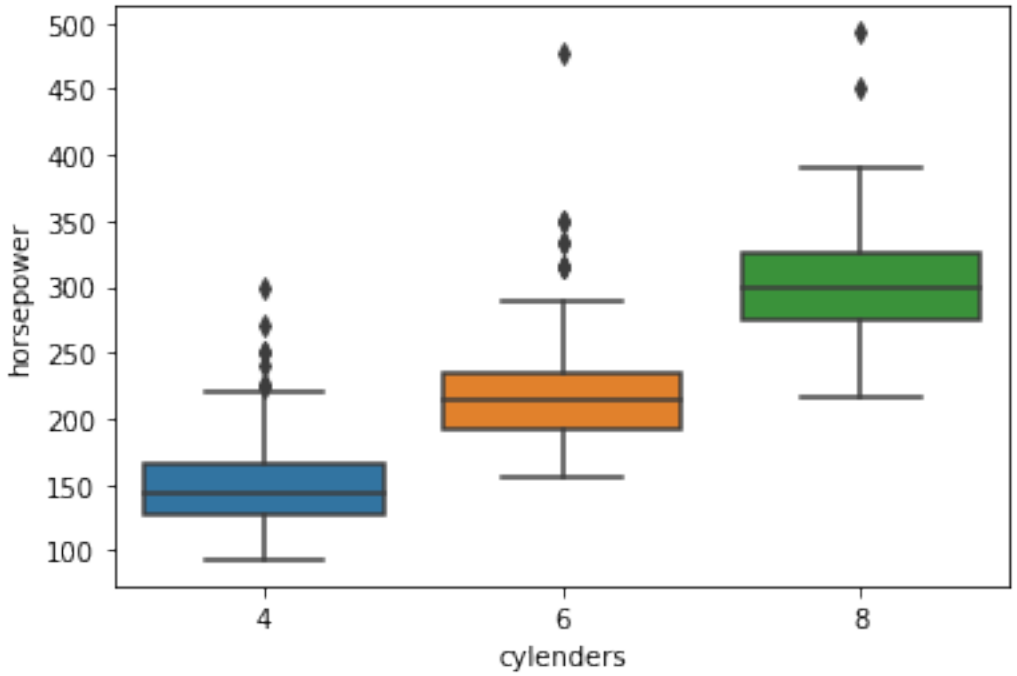
[ ]:
```
cars_common
```

[ ]:
```
                              name  sports_car    suv  wagon  minivan  \
0                 Chevrolet Aveo 4dr       False  False  False    False
1          Chevrolet Aveo LS 4dr hatch       False  False  False    False
2               Chevrolet Cavalier 2dr       False  False  False    False
3               Chevrolet Cavalier 4dr       False  False  False    False
4            Chevrolet Cavalier LS 2dr       False  False  False    False
..                                 …         …    …    …      …
423          Nissan Titan King Cab XE       False  False  False    False
424                       Subaru Baja       False  False  False    False
425                    Toyota Tacoma       False  False  False    False
426      Toyota Tundra Regular Cab V6       False  False  False    False
427  Toyota Tundra Access Cab V6 SR5       False  False  False    False

     pickup  all_wheel  rear_wheel  Price  Dealer_Cost  Engine_size cylenders  \
0     False      False       False  11690        10965          1.6         4
1     False      False       False  12585        11802          1.6         4
2     False      False       False  14610        13697          2.2         4
3     False      False       False  14810        13884          2.2         4
4     False      False       False  16385        15357          2.2         4
..      …        …          …    …          …           …       …
423    True       True       False  26650        24926          5.6         8
424    True       True       False  24520        22304          2.5         4
425    True      False        True  12800        11879          2.4         4
426    True      False        True  16495        14978          3.4         6
427    True       True       False  25935        23520          3.4         6

     horsepower  city_miles_per_galloon  highway_miles_per_Gallon  weight  \
0           103                    28.0                      34.0  2370.0
1           103                    28.0                      34.0  2348.0
2           140                    26.0                      37.0  2617.0
3           140                    26.0                      37.0  2676.0
4           140                    26.0                      37.0  2617.0
..          …                      …                         …     …
423         305                    14.0                      18.0  5287.0
424         165                    21.0                      28.0  3485.0
425         142                    22.0                      27.0  2750.0
426         190                    16.0                      20.0  3925.0
427         190                    14.0                      17.0  4435.0

     base_wheeel  length  width
0           98.0   167.0   66.0
1           98.0   153.0   66.0
2          104.0   183.0   69.0
```

```
3           104.0   183.0   68.0
4           104.0   183.0   69.0
..            …       …     …
423         140.0     NaN    NaN
424         104.0     NaN    NaN
425         103.0     NaN    NaN
426         128.0     NaN    NaN
427         128.0     NaN    NaN

[413 rows x 19 columns]
```

[ ]:
```
sns.boxplot(x='cylenders', y='horsepower', data=cars_common)
```

[ ]:
```
<AxesSubplot:xlabel='cylenders', ylabel='horsepower'>
```



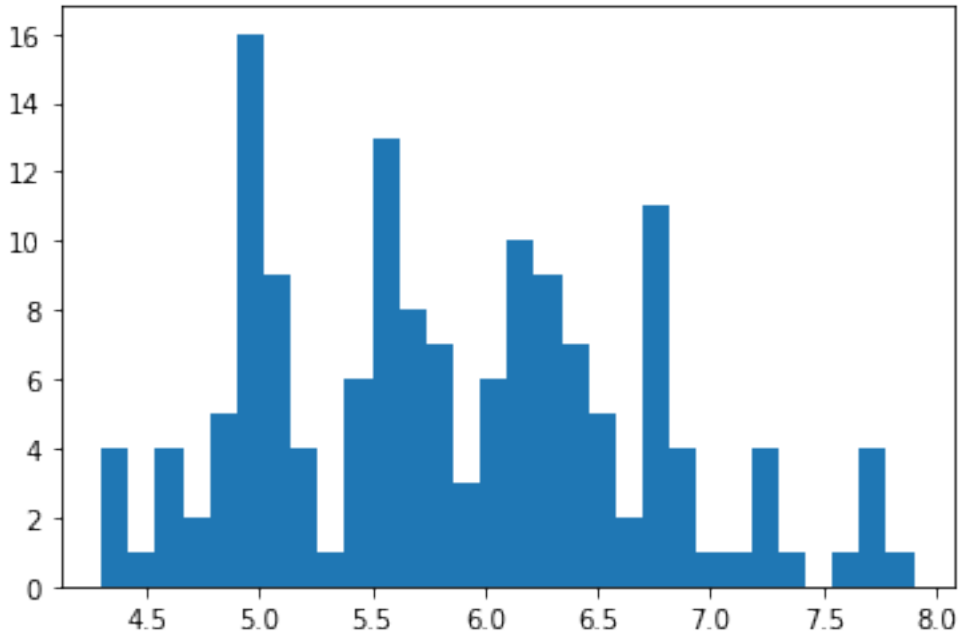
0.7 Histograms

In Histogram, x is always continues and we are doing frequency count of the continues variable.

[ ]:
```
plt.hist(x= iris.sepal_length, bins=30)
```

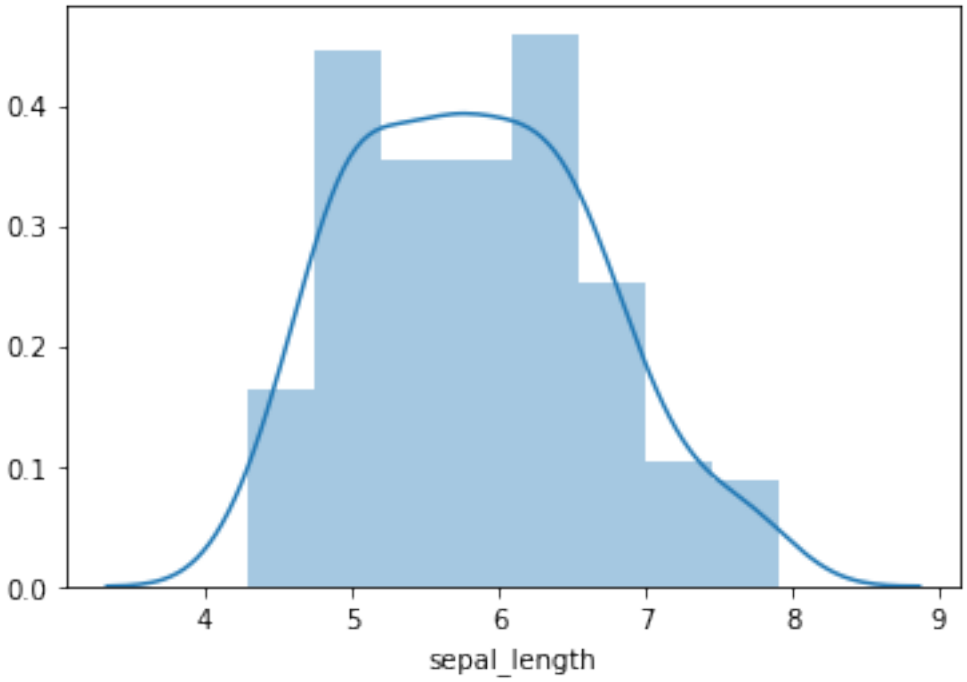
[ ]:
```
(array([ 4.,  1.,  4.,  2.,  5., 16.,  9.,  4.,  1.,  6., 13.,  8.,  7.,
         3.,  6., 10.,  9.,  7.,  5.,  2., 11.,  4.,  1.,  1.,  4.,  1.,
         0.,  1.,  4.,  1.]),
 array([4.3 , 4.42, 4.54, 4.66, 4.78, 4.9 , 5.02, 5.14, 5.26, 5.38, 5.5 ,
```

```
       5.62, 5.74, 5.86, 5.98, 6.1 , 6.22, 6.34, 6.46, 6.58, 6.7 , 6.82,
       6.94, 7.06, 7.18, 7.3 , 7.42, 7.54, 7.66, 7.78, 7.9 ]),
<BarContainer object of 30 artists>)
```



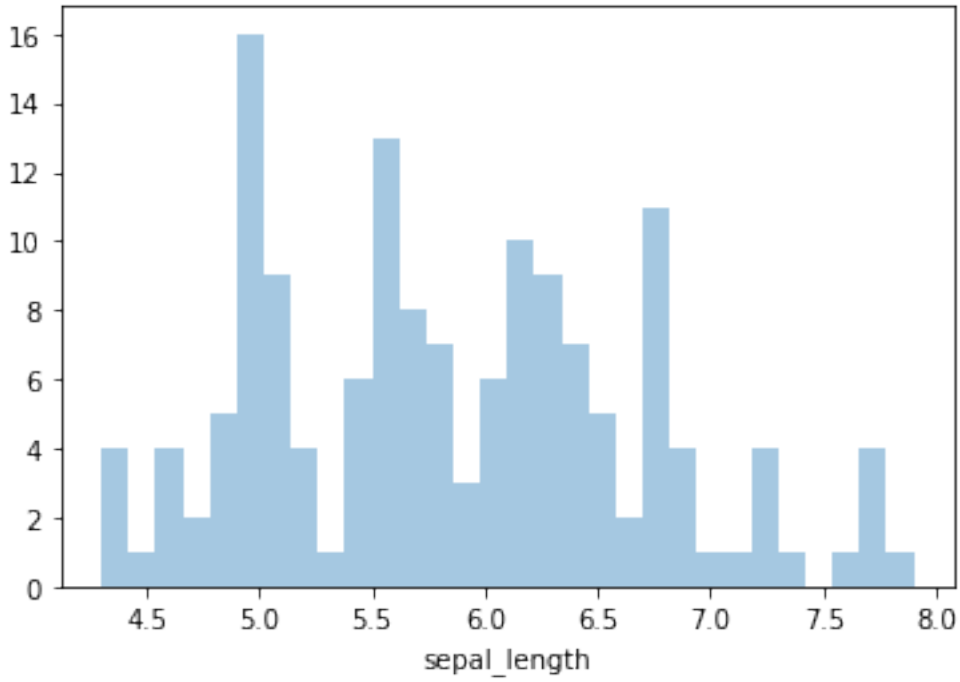
[ ]:
```
sns.distplot(iris.sepal_length)
```

[ ]:
```
<AxesSubplot:xlabel='sepal_length'>
```



[ ]:
```
sns.distplot(iris.sepal_length,bins=30,kde=False)
```

[ ]:
```
<AxesSubplot:xlabel='sepal_length'>
```



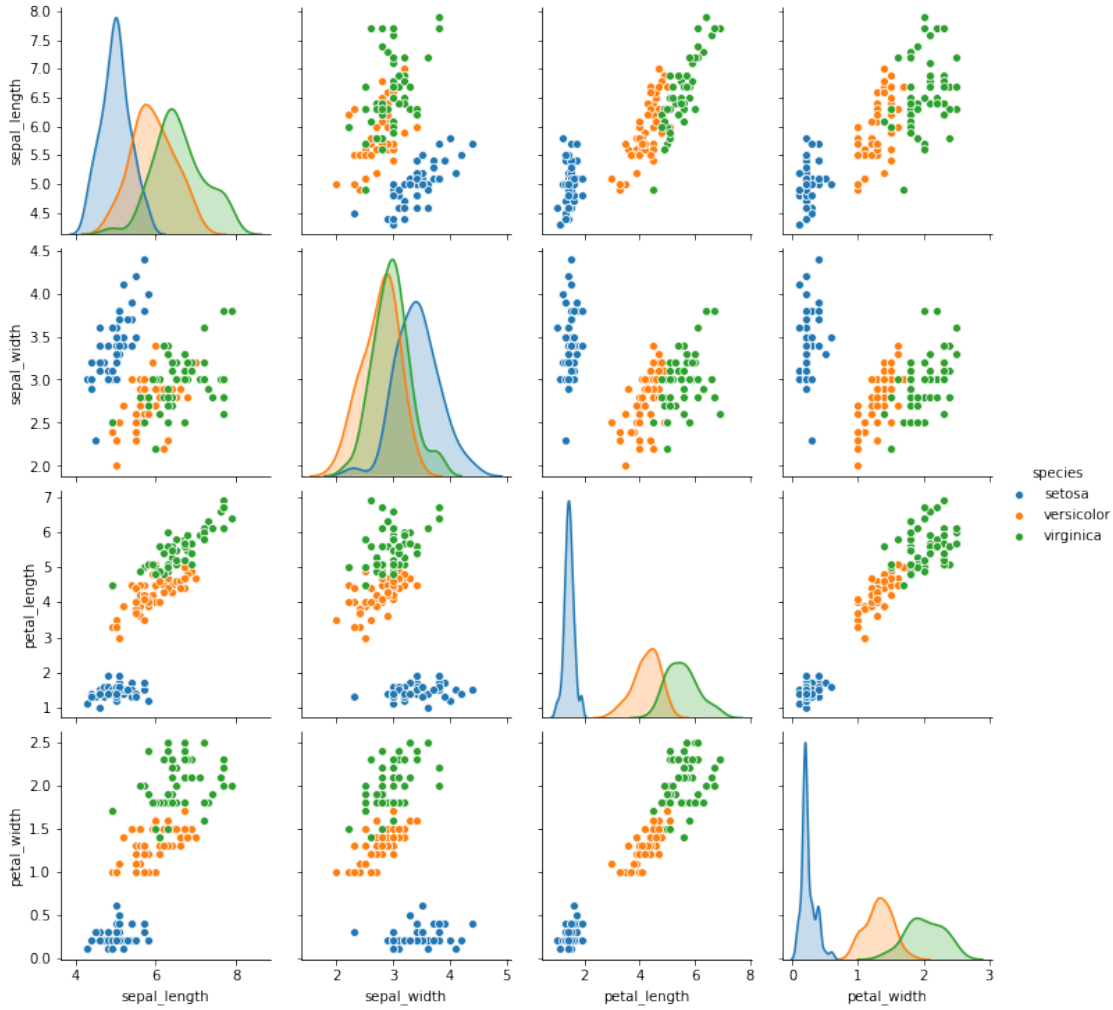
0.8 Pairplots

It does multiple comparisons between many variables as long as we specify which variables.

E.g Compare Setosa, verginica and versicula

[ ]:
```
sns.pairplot(iris,hue='species')
```

[ ]:
```
<seaborn.axisgrid.PairGrid at 0x1de93994bc8>
```



As seen petal lengths of setosa is very low, if a flower has petal length of 2, most definitely, it could be a setosa.

if a flower has petal width from 0 to 1, most definitly it migth be a setosa.

petal widths more than 2.0, most defeniltly it is a verginica.

```
[ ]: cars_common.columns
```

```
[ ]: Index(['name', 'sports_car', 'suv', 'wagon', 'minivan', 'pickup', 'all_wheel',
       'rear_wheel', 'Price', 'Dealer_Cost', 'Engine_size', 'cylenders',
       'horsepower', 'city_miles_per_galloon', 'highway_miles_per_Gallon',
       'weight', 'base_wheeel', 'length', 'width'],
      dtype='object')
```

```
[ ]: cars_common.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 413 entries, 0 to 427
Data columns (total 19 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   name                      413 non-null    object
 1   sports_car                413 non-null    bool
 2   suv                       413 non-null    bool
 3   wagon                     413 non-null    bool
 4   minivan                   413 non-null    bool
 5   pickup                    413 non-null    bool
 6   all_wheel                 413 non-null    bool
 7   rear_wheel                413 non-null    bool
 8   Price                     413 non-null    int64
 9   Dealer_Cost               413 non-null    int64
 10  Engine_size               413 non-null    float64
 11  cylenders                 413 non-null    category
 12  horsepower                413 non-null    int64
 13  city_miles_per_galloon    402 non-null    float64
 14  highway_miles_per_Gallon  402 non-null    float64
 15  weight                    411 non-null    float64
 16  base_wheeel               411 non-null    float64
 17  length                    387 non-null    float64
 18  width                     387 non-null    float64
dtypes: bool(7), category(1), float64(7), int64(3), object(1)
memory usage: 58.2+ KB
```

```
[ ]: cars_common.iloc[0:2]
```

```
[ ]:                         name  sports_car    suv  wagon  minivan  pickup  \
     0          Chevrolet Aveo 4dr       False  False  False    False   False
     1  Chevrolet Aveo LS 4dr hatch       False  False  False    False   False

        all_wheel  rear_wheel  Price  Dealer_Cost  Engine_size cylenders  \
     0      False       False  11690        10965          1.6         4
     1      False       False  12585        11802          1.6         4

        horsepower  city_miles_per_galloon  highway_miles_per_Gallon  weight  \
```

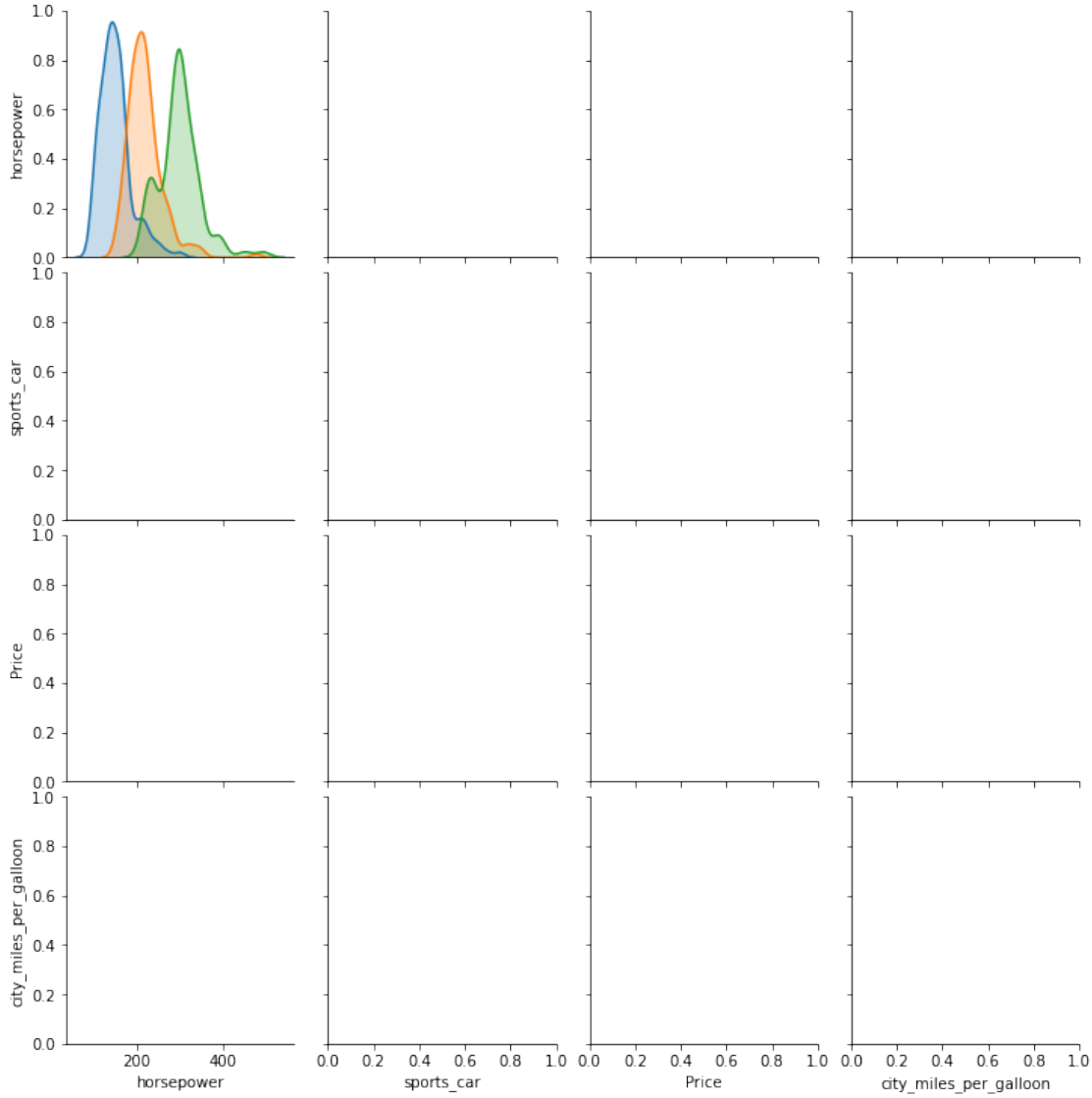
```
0         103                    28.0                     34.0  2370.0
1         103                    28.0                     34.0  2348.0

   base_wheeel  length  width
0         98.0   167.0   66.0
1         98.0   153.0   66.0
```

[ ]:
```
sns.
↪pairplot(cars_common[['horsepower','sports_car','Price','city_miles_per_galloon','cylenders
```

```
      ␣
↪--------------------------------------------------------------------------

       ValueError                                Traceback (most recent call␣
↪last)

       ~\Anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py in␣
↪kdensityfft(X, kernel, bw, weights, gridsize, adjust, clip, cut, retgrid)
       450     try:
   --> 451         bw = float(bw)
       452     except:


       ValueError: could not convert string to float: 'scott'


   During handling of the above exception, another exception occurred:


       RuntimeError                              Traceback (most recent call␣
↪last)

       <ipython-input-69-11e2a3f1cfe8> in <module>
   ----> 1 sns.
↪pairplot(cars_common[['horsepower','sports_car','Price','city_miles_per_galloon','cylenders'


       ~\Anaconda3\lib\site-packages\seaborn\axisgrid.py in pairplot(data, hue,␣
↪hue_order, palette, vars, x_vars, y_vars, kind, diag_kind, markers, height,␣
↪aspect, corner, dropna, plot_kws, diag_kws, grid_kws, size)
      2119             diag_kws.setdefault("shade", True)
      2120             diag_kws["legend"] = False
   -> 2121             grid.map_diag(kdeplot, **diag_kws)
      2122
      2123     # Maybe plot on the off-diagonals
```

```
      ~\Anaconda3\lib\site-packages\seaborn\axisgrid.py in map_diag(self,␣
↪func, **kwargs)
     1488                     data_k = utils.remove_na(data_k)
     1489
  -> 1490                 func(data_k, label=label_k, color=color, **kwargs)
     1491
     1492             self._clean_axis(ax)


      ~\Anaconda3\lib\site-packages\seaborn\distributions.py in kdeplot(data,␣
↪data2, shade, vertical, kernel, bw, gridsize, cut, clip, legend, cumulative,␣
↪shade_lowest, cbar, cbar_ax, cbar_kws, ax, **kwargs)
      703         ax = _univariate_kdeplot(data, shade, vertical, kernel, bw,
      704                                  gridsize, cut, clip, legend, ax,
  --> 705                                  cumulative=cumulative, **kwargs)
      706
      707     return ax


      ~\Anaconda3\lib\site-packages\seaborn\distributions.py in␣
↪_univariate_kdeplot(data, shade, vertical, kernel, bw, gridsize, cut, clip,␣
↪legend, ax, cumulative, **kwargs)
      293         x, y = _statsmodels_univariate_kde(data, kernel, bw,
      294                                            gridsize, cut, clip,
  --> 295                                            cumulative=cumulative)
      296     else:
      297         # Fall back to scipy if missing statsmodels


      ~\Anaconda3\lib\site-packages\seaborn\distributions.py in␣
↪_statsmodels_univariate_kde(data, kernel, bw, gridsize, cut, clip, cumulative)
      365     fft = kernel == "gau"
      366     kde = smnp.KDEUnivariate(data)
  --> 367     kde.fit(kernel, bw, fft, gridsize=gridsize, cut=cut, clip=clip)
      368     if cumulative:
      369         grid, y = kde.support, kde.cdf


      ~\Anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py in␣
↪fit(self, kernel, bw, fft, weights, gridsize, adjust, cut, clip)
      138             density, grid, bw = kdensityfft(endog, kernel=kernel,␣
↪bw=bw,
      139                     adjust=adjust, weights=weights,␣
↪gridsize=gridsize,
  --> 140                     clip=clip, cut=cut)
      141         else:
```

```
      142             density, grid, bw = kdensity(endog, kernel=kernel, bw=bw,


      ~\Anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py in␣
↪kdensityfft(X, kernel, bw, weights, gridsize, adjust, clip, cut, retgrid)
      451         bw = float(bw)
      452     except:
  --> 453         bw = bandwidths.select_bandwidth(X, bw, kern) # will␣
↪cross-val fit this pattern?
      454     bw *= adjust
      455


      ~\Anaconda3\lib\site-packages\statsmodels\nonparametric\bandwidths.py in␣
↪select_bandwidth(x, bw, kernel)
      172         # eventually this can fall back on another selection␣
↪criterion.
      173         err = "Selected KDE bandwidth is 0. Cannot estiamte density."
  --> 174         raise RuntimeError(err)
      175     else:
      176         return bandwidth


      RuntimeError: Selected KDE bandwidth is 0. Cannot estiamte density.
```


horsepower
sports_car
Price
city_miles_per_galloon
0.0
0.2
0.4
0.6
0.8
1.0
200
400

[ ]:

[ ]: