In [1]:
```
pip install wget
```

```
Requirement already satisfied: wget in c:\users\lenovo\anaconda3\lib\site-packages (3.2)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
!python -m wget http://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data
```

```
Saved under auto-mpg (7).data
```

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:
```
# defining the column names
cols = ['MPG','Cylinders','Displacement','Horsepower','Weight',
                'Acceleration', 'Model Year', 'Origin']
# reading the .data file using pandas
df = pd.read_csv('./auto-mpg.data', names=cols, na_values = "?",
                comment = '\t',
                sep= " ",
                skipinitialspace=True)
#making a copy of the dataframe
data = df.copy()
```

In [5]:
```
##checking the data info
data.info()
```

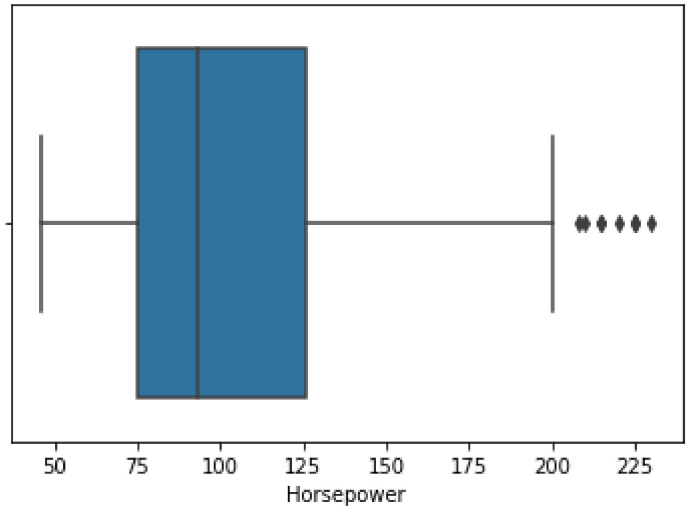
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   MPG           398 non-null    float64
 1   Cylinders     398 non-null    int64  
 2   Displacement  398 non-null    float64
 3   Horsepower    392 non-null    float64
 4   Weight        398 non-null    float64
 5   Acceleration  398 non-null    float64
 6   Model Year    398 non-null    int64  
 7   Origin        398 non-null    int64  
dtypes: float64(5), int64(3)
memory usage: 25.0 KB
```

In [6]:
```
##checking for all the null values
data.isnull().sum()
```

Out[6]:
```
MPG             0
Cylinders       0
Displacement    0
Horsepower      6
Weight          0
Acceleration    0
Model Year      0
Origin          0
dtype: int64
```

In [7]:
```
##summary statistics of quantitative variables
data.describe()

##looking at horsepower box plot
sns.boxplot(x=data['Horsepower'])
```

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x19eb07ea388>



In [8]:
```
##imputing the values with median
median = data['Horsepower'].median()
data['Horsepower'] = data['Horsepower'].fillna(median)
data.info()
print(data.head())
```

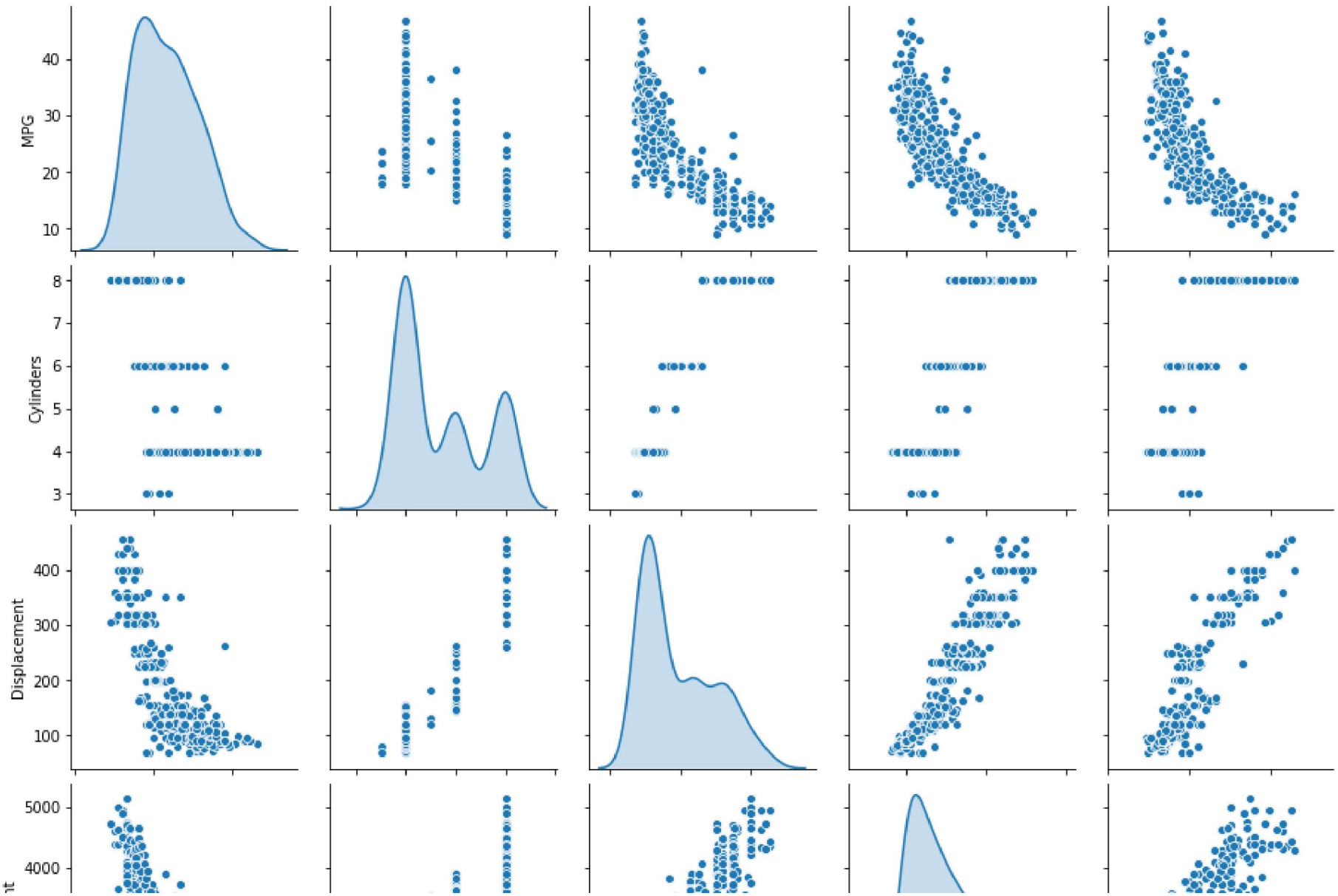
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   MPG           398 non-null    float64
 1   Cylinders     398 non-null    int64  
 2   Displacement  398 non-null    float64
 3   Horsepower    398 non-null    float64
 4   Weight        398 non-null    float64
 5   Acceleration  398 non-null    float64
 6   Model Year    398 non-null    int64  
 7   Origin        398 non-null    int64  
dtypes: float64(5), int64(3)
memory usage: 25.0 KB
    MPG  Cylinders  Displacement  Horsepower  Weight  Acceleration  \
0  18.0          8         307.0       130.0  3504.0          12.0   
1  15.0          8         350.0       165.0  3693.0          11.5   
2  18.0          8         318.0       150.0  3436.0          11.0   
3  16.0          8         304.0       150.0  3433.0          12.0   
4  17.0          8         302.0       140.0  3449.0          10.5   

   Model Year  Origin  
0          70       1  
1          70       1  
2          70       1  
3          70       1  
4          70       1  
```

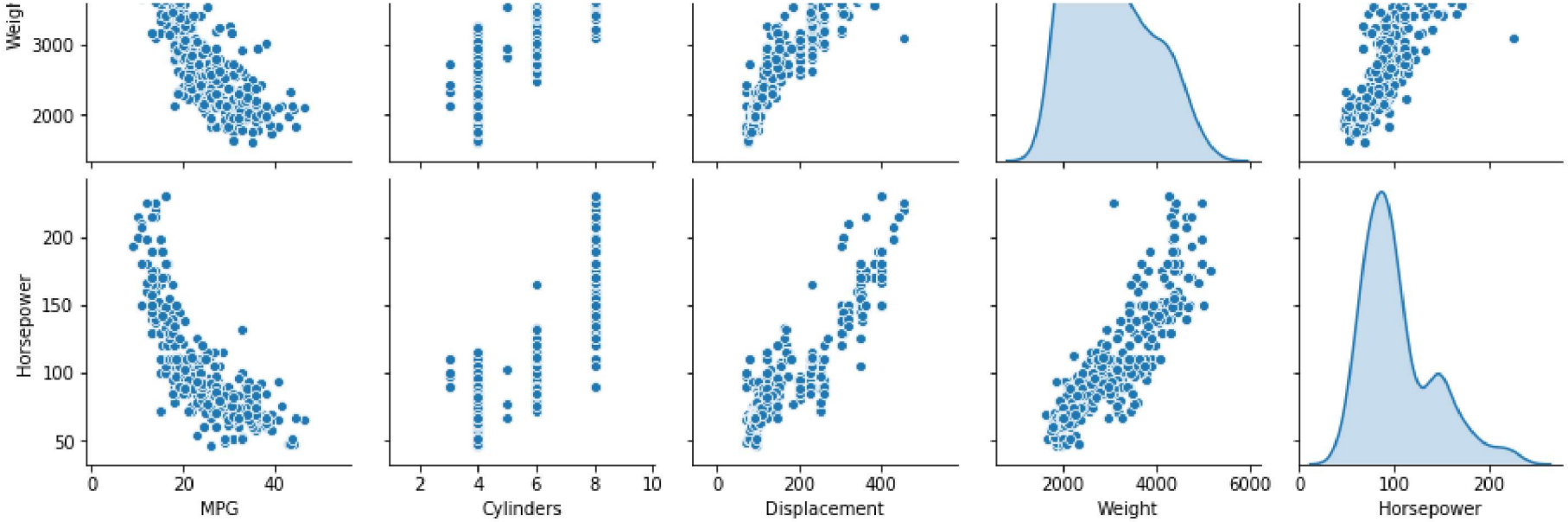
In [9]:
```
##category distribution

data["Cylinders"].value_counts() / len(data)
data['Origin'].value_counts()
```

Out[9]:
```
1    249
3     79
2     70
Name: Origin, dtype: int64
```

In [10]:
```
##pairplots to get an intuition of potential correlations

sns.pairplot(data[["MPG", "Cylinders", "Displacement", "Weight", "Horsepower"]], diag_kind="kde")
```

Out[10]:
```
<seaborn.axisgrid.PairGrid at 0x19eb0fb1188>
```





In [11]:
```
from sklearn.model_selection import StratifiedShuffleSplit
split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)
for train_index, test_index in split.split(data, data["Cylinders"]):
    strat_train_set = data.loc[train_index]
    strat_test_set = data.loc[test_index]
```

In [12]:
```
##checking for cylinder category distribution in training set

strat_train_set['Cylinders'].value_counts() / len(strat_train_set)
```

Out[12]:
```
4    0.512579
8    0.257862
6    0.210692
5    0.009434
3    0.009434
Name: Cylinders, dtype: float64
```

In [13]:
```
strat_test_set["Cylinders"].value_counts() / len(strat_test_set)
```

Out[13]:
```
4    0.5125
8    0.2625
6    0.2125
3    0.0125
Name: Cylinders, dtype: float64
```

In [14]:
```
data=strat_train_set.drop("MPG",axis=1)
data_labels=strat_train_set["MPG"].copy()
data
```

Out[14]:

	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year	Origin
145	4	83.0	61.0	2003.0	19.0	74	3
151	4	79.0	67.0	2000.0	16.0	74	2
388	4	156.0	92.0	2585.0	14.5	82	1
48	6	250.0	88.0	3139.0	14.5	71	1
114	4	98.0	90.0	2265.0	15.5	73	2
...	...	...	...	...	...	...	...
147	4	90.0	75.0	2108.0	15.5	74	2
156	8	400.0	170.0	4668.0	11.5	75	1
395	4	135.0	84.0	2295.0	11.6	82	1
14	4	113.0	95.0	2372.0	15.0	70	3
362	6	146.0	120.0	2930.0	13.8	81	3

318 rows × 7 columns

In [15]:
```
def preprocess_origin_cols(df):
    df["Origin"]=df['Origin'].map({1: 'India', 2: 'USA', 3 : 'Germany'})
    return df
data_tr=preprocess_origin_cols(data)
data_tr.head()
```

Out[15]:

	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year	Origin
145	4	83.0	61.0	2003.0	19.0	74	Germany
151	4	79.0	67.0	2000.0	16.0	74	USA
388	4	156.0	92.0	2585.0	14.5	82	India
48	6	250.0	88.0	3139.0	14.5	71	India
114	4	98.0	90.0	2265.0	15.5	73	USA

In [16]:
```
data_tr.info()
data_cat=data_tr[["Origin"]]
data_cat.head()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 318 entries, 145 to 362
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Cylinders     318 non-null    int64
 1   Displacement  318 non-null    float64
 2   Horsepower    318 non-null    float64
 3   Weight        318 non-null    float64
 4   Acceleration  318 non-null    float64
 5   Model Year    318 non-null    int64
 6   Origin        318 non-null    object
dtypes: float64(4), int64(2), object(1)
memory usage: 19.9+ KB
```

Out[16]:

	Origin
145	Germany
151	USA
388	India
48	India
114	USA

In [17]:
```
##onehotencoding the categorical values
from sklearn.preprocessing import OneHotEncoder
cat_encoder = OneHotEncoder()
data_cat_1hot = cat_encoder.fit_transform(data_cat)
data_cat_1hot   # returns a sparse matrix

data_cat_1hot.toarray()[:5]
```

Out[17]:
```
array([[1., 0., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

In [18]:
```
cat_encoder.categories_
```

Out[18]:
```
[array(['Germany', 'India', 'USA'], dtype=object)]
```

In [19]:
```
num_data = data.iloc[:, :-1]
num_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 318 entries, 145 to 362
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Cylinders     318 non-null    int64  
 1   Displacement  318 non-null    float64
 2   Horsepower    318 non-null    float64
 3   Weight        318 non-null    float64
 4   Acceleration  318 non-null    float64
 5   Model Year    318 non-null    int64  
dtypes: float64(4), int64(2)
memory usage: 17.4 KB
```

In [20]:
```
##handling missing values
from sklearn.impute import SimpleImputer

imputer = SimpleImputer(strategy="median")
imputer.fit(num_data)
```

Out[20]:
```
SimpleImputer(add_indicator=False, copy=True, fill_value=None,
              missing_values=nan, strategy='median', verbose=0)
```

In [21]:
```
##median of all the columns from imputer
imputer.statistics_
```

Out[21]:
```
array([   4. ,  146. ,   93.5, 2844. ,   15.5,   76. ])
```

In [22]:
```
##median from pandas dataframe - same
data.median().values
```

Out[22]:
```
array([   4. ,  146. ,   93.5, 2844. ,   15.5,   76. ])
```

In [23]:
```
##imputing the missing values by transforming the dataframe
X = imputer.transform(num_data)
X
```

Out[23]:
```
array([[   4. ,   83. ,   61. , 2003. ,   19. ,   74. ],
       [   4. ,   79. ,   67. , 2000. ,   16. ,   74. ],
       [   4. ,  156. ,   92. , 2585. ,   14.5,   82. ],
       ...,
       [   4. ,  135. ,   84. , 2295. ,   11.6,   82. ],
       [   4. ,  113. ,   95. , 2372. ,   15. ,   70. ],
       [   6. ,  146. ,  120. , 2930. ,   13.8,   81. ]])
```

In [24]:
```
##converting the 2D array back into a dataframe
data_tr = pd.DataFrame(X, columns=num_data.columns,index=num_data.index)
data_tr.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 318 entries, 145 to 362
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Cylinders     318 non-null    float64
 1   Displacement  318 non-null    float64
 2   Horsepower    318 non-null    float64
 3   Weight        318 non-null    float64
 4   Acceleration  318 non-null    float64
 5   Model Year    318 non-null    float64
dtypes: float64(6)
memory usage: 17.4 KB
```