Practical Example

Importing the relevant libraries

In [4]:
```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
import seaborn as sns
sns.set()
```

Loading the raw data

In [6]:
```
raw_data = pd.read_csv('/Users/saurabhsant/Downloads/1.04. Real-life example.csv')
raw_data.head()
```

Out[6]:

	Brand	Price	Body	Mileage	EngineV	Engine Type	Registration	Year	Model
0	BMW	4200.0	sedan	277	2.0	Petrol	yes	1991	320
1	Mercedes-Benz	7900.0	van	427	2.9	Diesel	yes	1999	Sprinter 212
2	Mercedes-Benz	13300.0	sedan	358	5.0	Gas	yes	2003	S 500
3	Audi	23000.0	crossover	240	4.2	Petrol	yes	2007	Q7
4	Toyota	18300.0	crossover	120	2.0	Petrol	yes	2011	Rav 4

Preprocessing

Exploring the descriptive statistics of the variables

In [8]:
```
raw_data.describe(include='all')
```

Out[8]:

	Brand	Price	Body	Mileage	EngineV	Engine Type	Registration	
count	4345	4173.000000	4345	4345.000000	4195.000000	4345	4345	4345.(
unique	7	NaN	6	NaN	NaN	4	2	
top	Volkswagen	NaN	sedan	NaN	NaN	Diesel	yes	
freq	936	NaN	1649	NaN	NaN	2019	3947	
mean	NaN	19418.746935	NaN	161.237284	2.790734	NaN	NaN	2006.5
std	NaN	25584.242620	NaN	105.705797	5.066437	NaN	NaN	6.7
min	NaN	600.000000	NaN	0.000000	0.600000	NaN	NaN	1969.(
25%	NaN	6999.000000	NaN	86.000000	1.800000	NaN	NaN	2003.(
50%	NaN	11500.000000	NaN	155.000000	2.200000	NaN	NaN	2008.(
75%	NaN	21700.000000	NaN	230.000000	3.000000	NaN	NaN	2012.(
max	NaN	300000.000000	NaN	980.000000	99.990000	NaN	NaN	2016.(

In [9]:
```
data = raw_data.drop(['Model'], axis =1)
data.describe(include='all')
```

Out[9]:

	Brand	Price	Body	Mileage	EngineV	Engine Type	Registration	
count	4345	4173.000000	4345	4345.000000	4195.000000	4345	4345	4345.(
unique	7	NaN	6	NaN	NaN	4	2	
top	Volkswagen	NaN	sedan	NaN	NaN	Diesel	yes	
freq	936	NaN	1649	NaN	NaN	2019	3947	
mean	NaN	19418.746935	NaN	161.237284	2.790734	NaN	NaN	2006.5
std	NaN	25584.242620	NaN	105.705797	5.066437	NaN	NaN	6.7
min	NaN	600.000000	NaN	0.000000	0.600000	NaN	NaN	1969.(
25%	NaN	6999.000000	NaN	86.000000	1.800000	NaN	NaN	2003.(
50%	NaN	11500.000000	NaN	155.000000	2.200000	NaN	NaN	2008.(
75%	NaN	21700.000000	NaN	230.000000	3.000000	NaN	NaN	2012.(
max	NaN	300000.000000	NaN	980.000000	99.990000	NaN	NaN	2016.(

Dealing with missing values

In [10]:
```
data.isnull().sum()
```

Out[10]:
```
Brand             0
Price           172
Body              0
Mileage           0
EngineV         150
Engine Type       0
Registration      0
Year              0
dtype: int64
```

In [11]:
```
data_no_mv = data.dropna(axis=0)
```

In [12]:
```
data_no_mv.describe(include='all')
```

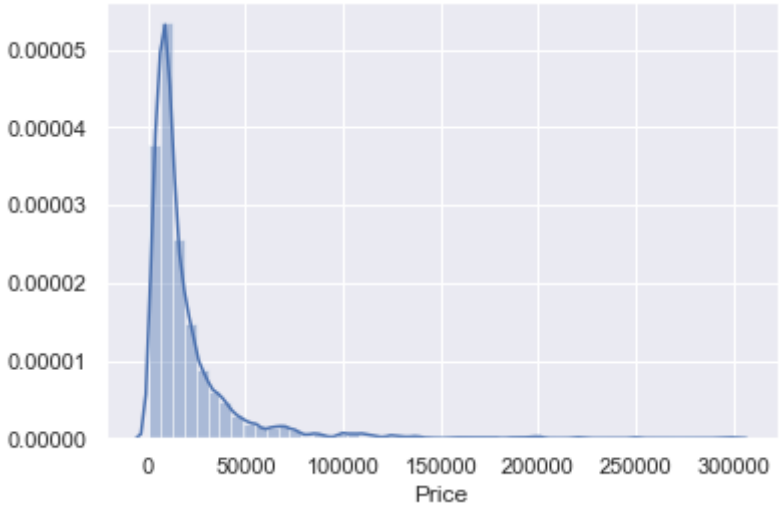
Out[12]:

	Brand	Price	Body	Mileage	EngineV	Engine Type	Registration	
count	4025	4025.000000	4025	4025.000000	4025.000000	4025	4025	4025.(
unique	7	NaN	6	NaN	NaN	4	2	
top	Volkswagen	NaN	sedan	NaN	NaN	Diesel	yes	
freq	880	NaN	1534	NaN	NaN	1861	3654	
mean	NaN	19552.308065	NaN	163.572174	2.764586	NaN	NaN	2006.3
std	NaN	25815.734988	NaN	103.394703	4.935941	NaN	NaN	6.(
min	NaN	600.000000	NaN	0.000000	0.600000	NaN	NaN	1969.(
25%	NaN	6999.000000	NaN	90.000000	1.800000	NaN	NaN	2003.(
50%	NaN	11500.000000	NaN	158.000000	2.200000	NaN	NaN	2007.(
75%	NaN	21900.000000	NaN	230.000000	3.000000	NaN	NaN	2012.(
max	NaN	300000.000000	NaN	980.000000	99.990000	NaN	NaN	2016.(

Exploring the PDFs

In [14]:
```
sns.distplot(data_no_mv['Price'])
```

Out[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1c1f5444e0>
```



Dealing with outliers

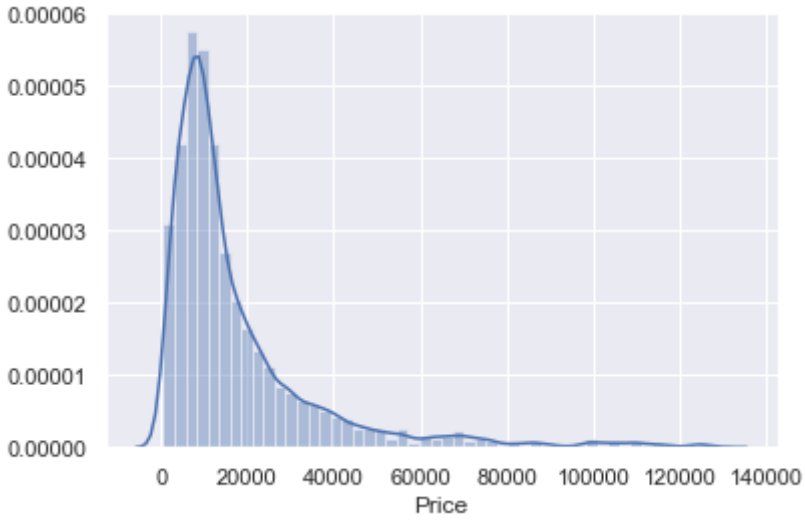
In [16]:
```
q = data_no_mv['Price'].quantile(0.99)
data_1 = data_no_mv[data_no_mv['Price']<q]
data_1.describe(include="all")
```

Out[16]:

	Brand	Price	Body	Mileage	EngineV	Engine Type	Registration	
count	3984	3984.000000	3984	3984.000000	3984.000000	3984	3984	3984.(
unique	7	NaN	6	NaN	NaN	4	2	
top	Volkswagen	NaN	sedan	NaN	NaN	Diesel	yes	
freq	880	NaN	1528	NaN	NaN	1853	3613	
mean	NaN	17837.117460	NaN	165.116466	2.743770	NaN	NaN	2006.2
std	NaN	18976.268315	NaN	102.766126	4.956057	NaN	NaN	6.(
min	NaN	600.000000	NaN	0.000000	0.600000	NaN	NaN	1969.(
25%	NaN	6980.000000	NaN	93.000000	1.800000	NaN	NaN	2002.7
50%	NaN	11400.000000	NaN	160.000000	2.200000	NaN	NaN	2007.(
75%	NaN	21000.000000	NaN	230.000000	3.000000	NaN	NaN	2011.(
max	NaN	129222.000000	NaN	980.000000	99.990000	NaN	NaN	2016.(

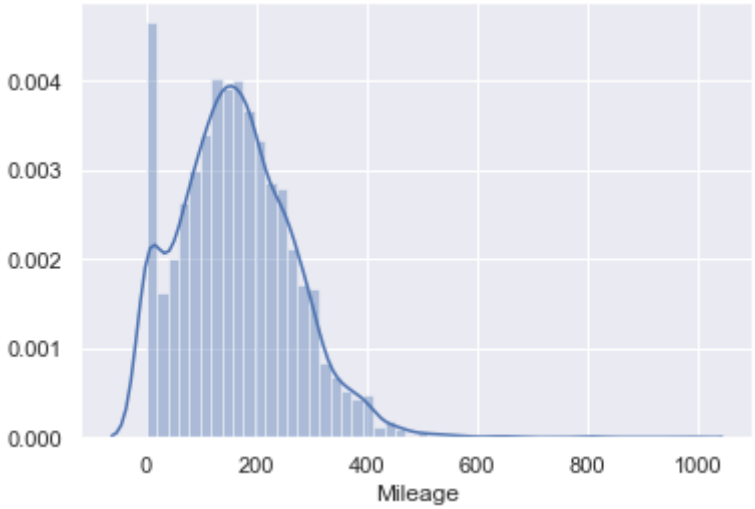
In [17]:
```
sns.distplot(data_1['Price'])
```

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x1c1fd2b630>



In [18]:
```
sns.distplot(data_no_mv['Mileage'])
```

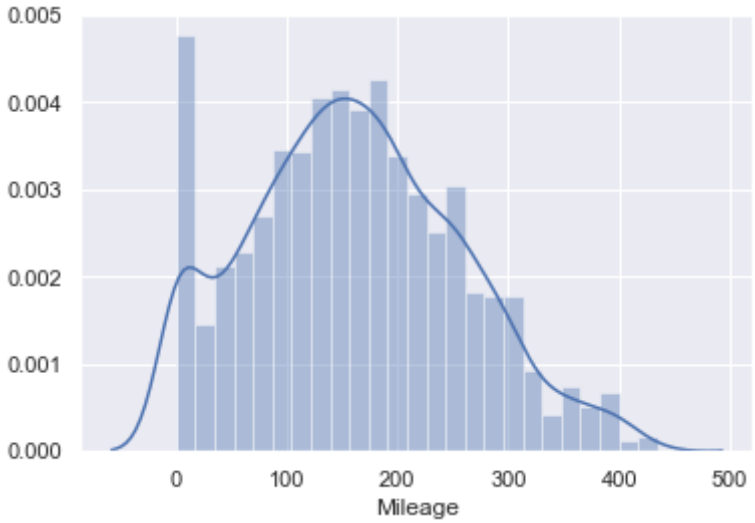
Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x1c1e95c828>



In [19]:
```
q = data_1['Mileage'].quantile(0.99)
data_2 = data_1[data_1['Mileage']<q]
```

In [20]:
```
sns.distplot(data_2['Mileage'])
```

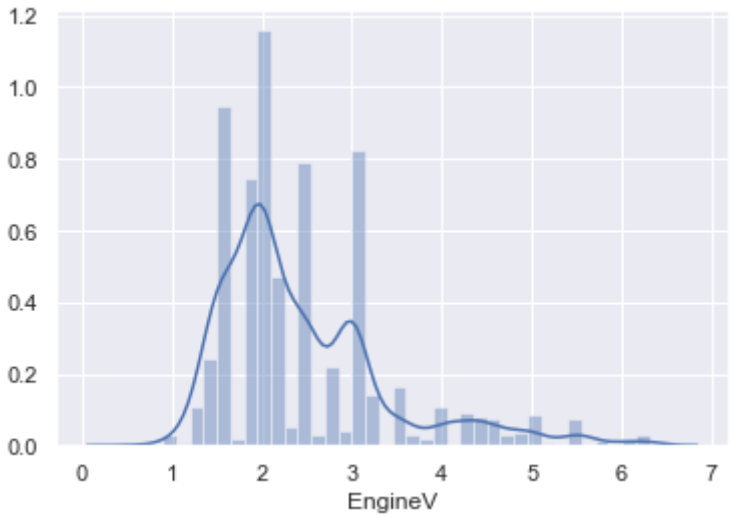
Out[20]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1c1ead6668>
```



In [21]:
```
data_3 = data_2[data_2['EngineV']<6.5]
```

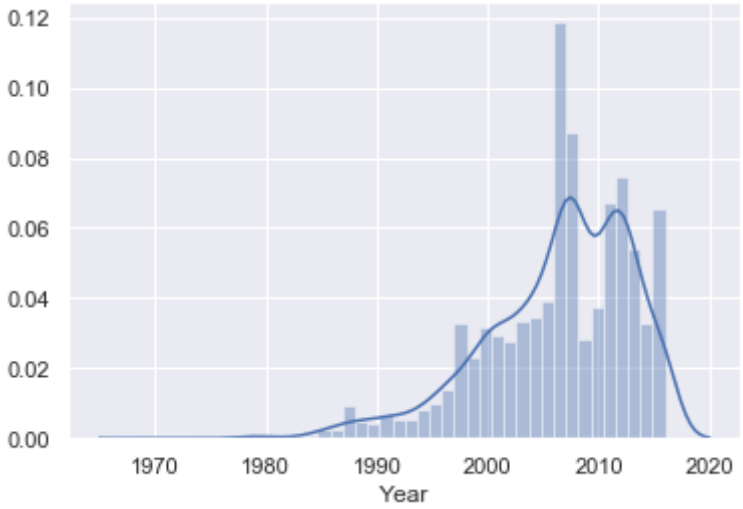
In [22]:
```
sns.distplot(data_3['EngineV'])
```

Out[22]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1c1ec0a4e0>
```



In [23]:
```
sns.distplot(data_no_mv['Year'])
```

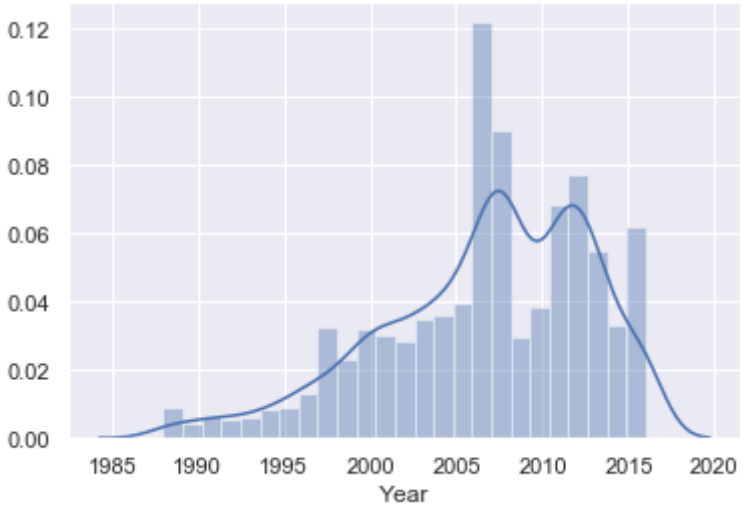
Out[23]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1c1edb22e8>
```



In [24]:
```
q = data_3['Year'].quantile(0.01)
data_4 = data_3[data_3['Year']>q]
```

In [25]:
```
sns.distplot(data_4['Year'])
```

Out[25]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1c1eecd0f0>
```



In [26]:
```
data_cleaned = data_4.reset_index(drop=True)
```

In [27]:
```
data_cleaned.describe(include='all')
```

Out[27]:

	Brand	Price	Body	Mileage	EngineV	Engine Type	Registration	
count	3867	3867.000000	3867	3867.000000	3867.000000	3867	3867	3867.(
unique	7	NaN	6	NaN	NaN	4	2	
top	Volkswagen	NaN	sedan	NaN	NaN	Diesel	yes	
freq	848	NaN	1467	NaN	NaN	1807	3505	
mean	NaN	18194.455679	NaN	160.542539	2.450440	NaN	NaN	2006.7
std	NaN	19085.855165	NaN	95.633291	0.949366	NaN	NaN	6.1
min	NaN	800.000000	NaN	0.000000	0.600000	NaN	NaN	1988.(
25%	NaN	7200.000000	NaN	91.000000	1.800000	NaN	NaN	2003.(
50%	NaN	11700.000000	NaN	157.000000	2.200000	NaN	NaN	2008.(
75%	NaN	21700.000000	NaN	225.000000	3.000000	NaN	NaN	2012.(
max	NaN	129222.000000	NaN	435.000000	6.300000	NaN	NaN	2016.(

Checking the OLS assumptions

In [ ]:
```

```

In [ ]:
```

```