In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
```

In [2]:
```
data = pd.read_csv('bmwcarprices.csv')
data
```

Out[2]:

	Mileage	Age(yrs)	Sell Price($)
0	69000	6	18000
1	35000	3	34000
2	57000	5	26100
3	22500	2	40000
4	46000	4	31500
5	59000	5	26750
6	52000	5	32000
7	72000	6	19300
8	91000	8	12000
9	67000	6	22000
10	83000	7	18700
11	79000	7	19500
12	59000	5	26000
13	58780	4	27500
14	82450	7	19400
15	25400	3	35000
16	28000	2	35500
17	69000	5	19700
18	87600	8	12800
19	52000	5	28200

Cleaning and Preprocessing

Checking if there is any duplicate or empty data cells

In [3]:
```
data.duplicated()
```

Out[3]:
```
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9     False
10    False
11    False
12    False
13    False
14    False
15    False
16    False
17    False
18    False
19    False
dtype: bool
```

In [4]:
```
data.isnull()
```

Out[4]:

	Mileage	Age(yrs)	Sell Price($)
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False
4	False	False	False
5	False	False	False
6	False	False	False
7	False	False	False
8	False	False	False
9	False	False	False
10	False	False	False
11	False	False	False
12	False	False	False
13	False	False	False
14	False	False	False
15	False	False	False
16	False	False	False
17	False	False	False
18	False	False	False
19	False	False	False

Analyzing data

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Mileage        20 non-null     int64
 1   Age(yrs)       20 non-null     int64
 2   Sell Price($)  20 non-null     int64
dtypes: int64(3)
memory usage: 608.0 bytes
```

In [6]:
```
data.describe()
```

Out[6]:

	Mileage	Age(yrs)	Sell Price($)
count	20.000000	20.000000	20.000000
mean	59736.500000	5.150000	25197.500000
std	20595.441825	1.785173	7834.479713
min	22500.000000	2.000000	12000.000000
25%	50500.000000	4.000000	19375.000000
50%	59000.000000	5.000000	26050.000000
75%	73750.000000	6.250000	31625.000000
max	91000.000000	8.000000	40000.000000

In [7]:
```
data['Sell Price($)'].mean()
```

Out[7]:
```
25197.5
```

In [8]:
```
data.mean()
```

Out[8]:
```
Mileage          59736.50
Age(yrs)             5.15
Sell Price($)    25197.50
dtype: float64
```

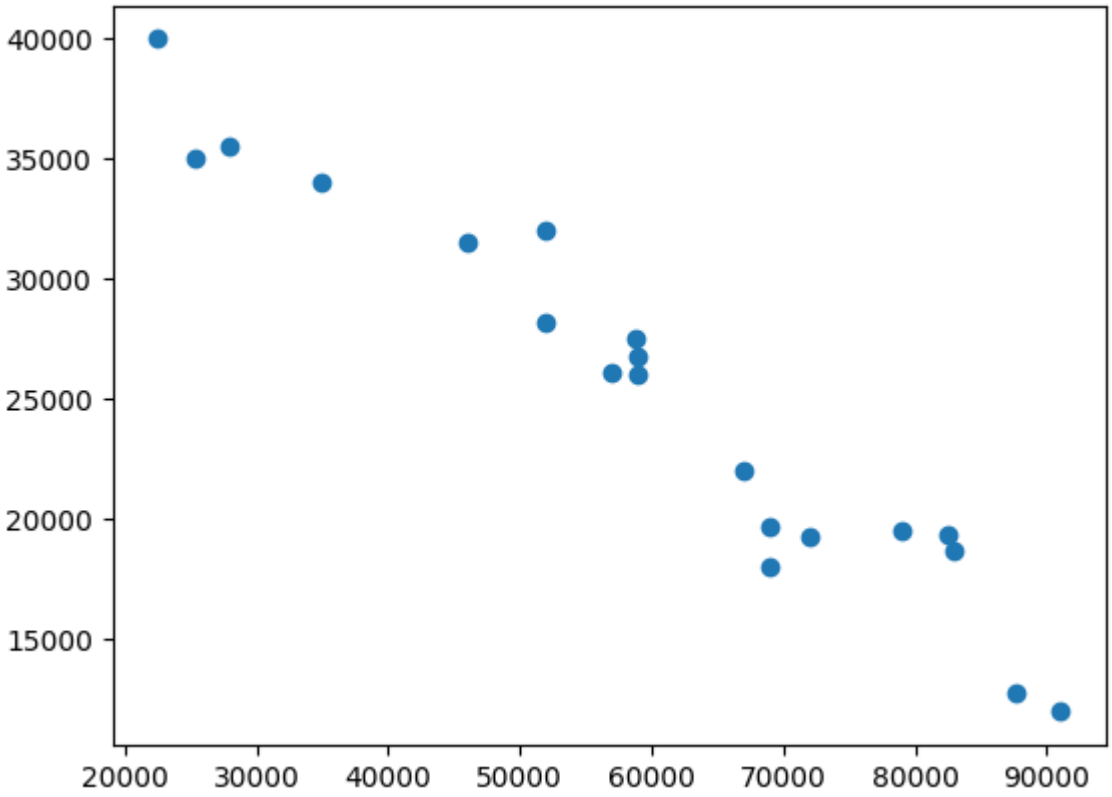
In [9]:
```
txt = "Oldest BMW on sell is {} years old and newest one is {} years old"
txt.format(data['Age(yrs)'].max(), data['Age(yrs)'].min())
```

Out[9]:
```
'Oldest BMW on sell is 8 years old and newest one is 2 years old'
```

Visualizing the data

In [10]:
```
plt.scatter(data['Mileage'], data['Sell Price($)'])
```

Out[10]:
```
<matplotlib.collections.PathCollection at 0x206a3322a90>
```



Machine Learning

Using Linear Regression

In [11]:
```
model = LinearRegression()
```

Splitting the data into X and y

In [24]:
```
X = data.drop(['Sell Price($)'], axis=1)
y = data['Sell Price($)']
X
```

Out[24]:

	Mileage	Age(yrs)
0	69000	6
1	35000	3
2	57000	5
3	22500	2
4	46000	4
5	59000	5
6	52000	5
7	72000	6
8	91000	8
9	67000	6
10	83000	7
11	79000	7
12	59000	5
13	58780	4
14	82450	7
15	25400	3
16	28000	2
17	69000	5
18	87600	8
19	52000	5

Splitting the data into Training and Testing sets

In [13]:
```
from sklearn.model_selection import train_test_split
```

In [14]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

Training the model

In [15]:
```
model.fit(X_train, y_train)
```

Out[15]:
```
▼ LinearRegression
LinearRegression()
```

Predicting using Testing data set

In [16]:
```
predict = model.predict(X_test)
```

In [17]:
```
predict.astype(int)
```

Out[17]:
```
array([27954, 37986, 38414, 16123])
```

In [18]:
```
y_test
```

Out[18]:
```
6     32000
16    35500
15    35000
14    19400
Name: Sell Price($), dtype: int64
```

Displaying Predicted and Actual value in a DataFrame

In [19]:
```
value_data = {
    "Actual values" : y_test,
    "Predicted Values" : predict.astype(int)
}

resultdf = pd.DataFrame(value_data)
resultdf
```

Out[19]:

	Actual values	Predicted Values
6	32000	27954
16	35500	37986
15	35000	38414
14	19400	16123

In [20]:
```
model.score(X_test, y_test)
```

Out[20]:
```
0.7366857366135322
```