Training And Testing Available Data

We have a dataset containing prices of used BMW cars. We are going to analyze this dataset and build a prediction function that can predict a price by taking mileage and age of the car as input. We will use sklearn train_test_split method to split training and testing dataset

In [8]:
```
import pandas as pd
df = pd.read_csv("carprices.csv")
df.head()
```

Out[8]:

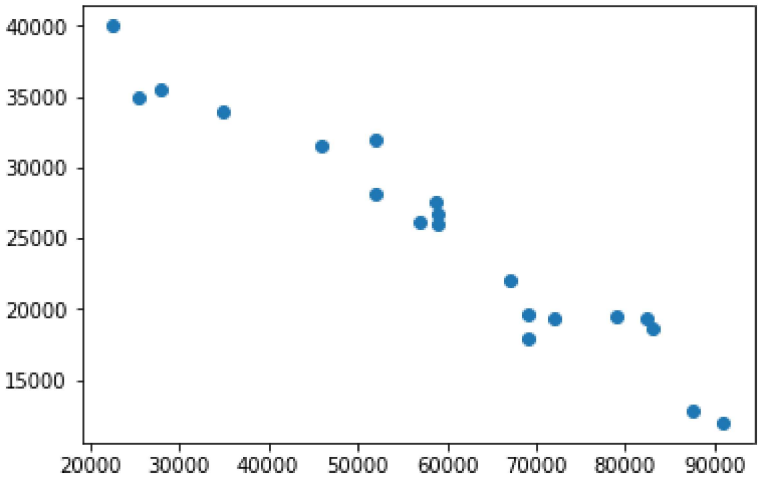
	Mileage	Age(yrs)	Sell Price($)
0	69000	6	18000
1	35000	3	34000
2	57000	5	26100
3	22500	2	40000
4	46000	4	31500

In [3]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

Car Mileage Vs Sell Price ($)

In [5]:
```
plt.scatter(df['Mileage'],df['Sell Price($)'])
```

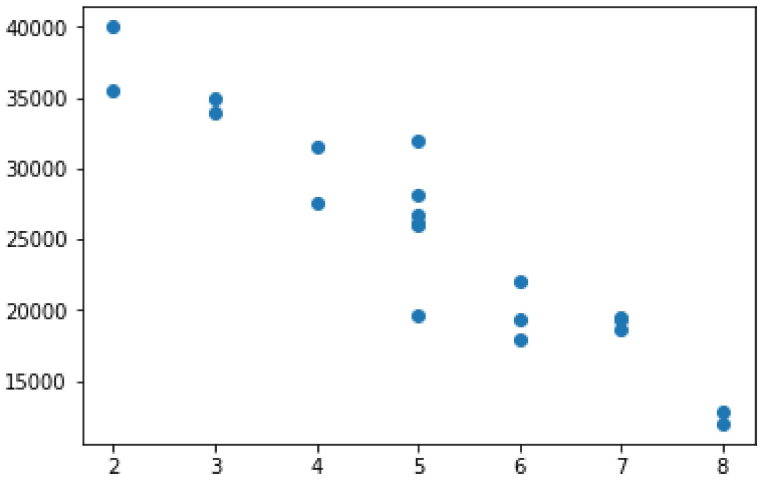
Out[5]: `<matplotlib.collections.PathCollection at 0x2882746dd30>`



Car Age Vs Sell Price ($)

In [6]:
```
plt.scatter(df['Age(yrs)'],df['Sell Price($)'])
```

Out[6]:
```
<matplotlib.collections.PathCollection at 0x28826e06240>
```



Looking at above two scatter plots, using linear regression model makes sense as we can clearly see a linear relationship between our dependant (i.e. Sell Price) and independant variables (i.e. car age and car mileage)

The approach we are going to use here is to split available data in two sets

1. **Training: We will train our model on this dataset**
2. **Testing: We will use this subset to make actual predictions using trained model**

The reason we don't use same training set for testing is because our model has seen those samples before, using same samples for making predictions might give us wrong impression about accuracy of our model. It is like you ask same questions in exam paper as you tought the students in the class.

In [12]:
```
X = df[['Mileage','Age(yrs)']]
```

In [14]:
```
y = df['Sell Price($)']
```

In [15]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3)
```

In [17]:
```
X_train
```

Out[17]:

	Mileage	Age(yrs)
11	79000	7
17	69000	5

	Mileage	Age(yrs)
10	83000	7
1	35000	3
0	69000	6
8	91000	8
7	72000	6
16	28000	2
6	52000	5
4	46000	4
19	52000	5
2	57000	5
5	59000	5
15	25400	3

In [18]:
```
X_test
```

Out[18]:

	Mileage	Age(yrs)
3	22500	2
12	59000	5
14	82450	7
13	58780	4
9	67000	6
18	87600	8

In [19]:
```
y_train
```

Out[19]:
```
11    19500
17    19700
10    18700
1     34000
0     18000
8     12000
7     19300
16    35500
6     32000
4     31500
19    28200
2     26100
5     26750
```

```
15    35000
Name: Sell Price($), dtype: int64
```

In [20]:
```
y_test
```

Out[20]:
```
3     40000
12    26000
14    19400
13    27500
9     22000
18    12800
Name: Sell Price($), dtype: int64
```

Lets run linear regression model now

In [22]:
```
from sklearn.linear_model import LinearRegression
clf = LinearRegression()
clf.fit(X_train, y_train)
```

Out[22]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [25]:
```
X_test
```

Out[25]:

	Mileage	Age(yrs)
3	22500	2
12	59000	5
14	82450	7
13	58780	4
9	67000	6
18	87600	8

In [26]:
```
clf.predict(X_test)
```

Out[26]:
```
array([ 38166.23426912,  25092.95646646,  16773.29470749,  24096.93956163,
        22602.44614295,  15559.98266172])
```

In [29]:
```
y_test
```

Out[29]:
```
3     40000
12    26000
14    19400
13    27500
9     22000
18    12800
Name: Sell Price($), dtype: int64
```

In [27]:
```
clf.score(X_test, y_test)
```

Out[27]:
```
0.92713129118963111
```

random_state argument

In [35]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=1
X_test
```

Out[35]:

	Mileage	Age(yrs)
7	72000	6
10	83000	7
5	59000	5
6	52000	5
3	22500	2
18	87600	8