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 [7]:
```
import pandas as pd
df = pd.read_csv("C:/Users/prasa/Desktop/py codes/ds projects/ML/6 Training and Testing Data/carprices.csv")
df.head()
```

Out[7]:

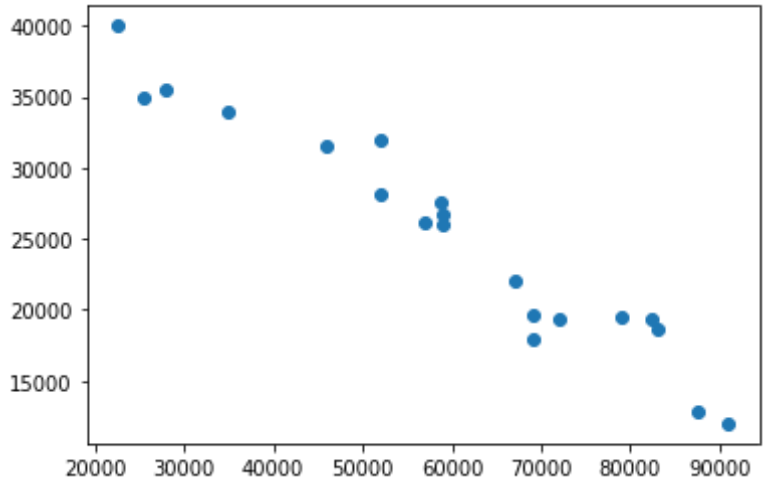
	Mileage	Age(yrs)	Sell Price($)
0	69000	6	18000
1	35000	3	34000
2	57000	5	26100
3	22500	2	40000
4	46000	4	31500

In [8]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

Car Mileage Vs Sell Price ($)

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

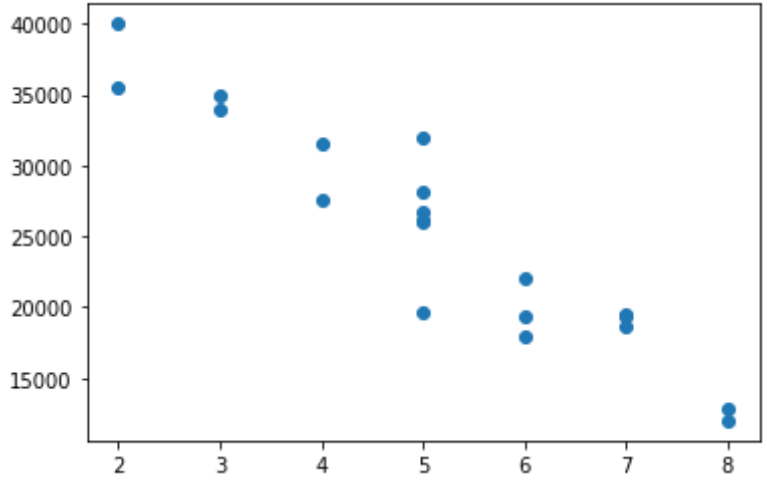
Out[10]:
```
<matplotlib.collections.PathCollection at 0x20fde0abf48>
```



Car Age Vs Sell Price ($)

In [11]:
```
plt.scatter(df['Age(yrs)'],df['Sell Price($)'])
```

Out[11]:
```
<matplotlib.collections.PathCollection at 0x20fde16f6c8>
```



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 u
sing 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)']]
y = df['Sell Price($)']
```

In [13]:
```
x
```

Out[13]:

	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

	Mileage	Age(yrs)
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

In [14]:
```
y
```

Out[14]:
```
0     18000
1     34000
2     26100
3     40000
4     31500
5     26750
6     32000
7     19300
8     12000
9     22000
10    18700
11    19500
12    26000
13    27500
14    19400
15    35000
16    35500
17    19700
18    12800
```

```
19    28200
Name: Sell Price($), dtype: int64
```

In [15]:
```
from sklearn.model_selection import train_test_split
```

In [25]:
```
X_train, X_test, y_train, y_test = train_test_split(x,y,test_size=0.2)
#random_state=10 won't change the sample
```

In [29]:
```
len(X_test)
```

Out[29]:
```
4
```

In [36]:
```
from sklearn.linear_model import LinearRegression
clf = LinearRegression()
```

In [37]:
```
clf.fit(X_train,y_train)
```

Out[37]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normaliz
e=False)
```

In [46]:
```
clf.predict(X_test)
```

Out[46]:
```
array([20474.0745775 , 16352.07892168, 25174.14834912, 27197.42175439])
```

In [47]:
```
y_test
```

Out[47]:
```
7     19300
10    18700
5     26750
6     32000
Name: Sell Price($), dtype: int64
```

In [49]:
```
clf.score(X_test, y_test)
```

Out[49]:
```
0.7332339593090138
```

random_state argument

In [51]:
```
X_train, X_test, y_train, y_test = train_test_split(x,y,test_size=0.3,r
andom_state=10)
X_test
```

Out[51]:

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