Linear Regression

Steps

- Data collection
- data wrangling / pre-processing
- Model building
- Testing the model
- Evaluate the model performance
- Prediction with random input

In [1]:

```
# required libraries
import numpy as np
import pandas as pd
```

In [2]:

```
# Load the dataset into pandas dataframe
data = pd.read_csv('Salary_Data.csv')
data
```

Out[2]:

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0
5	2.9	56642.0
6	3.0	60150.0
7	3.2	54445.0
8	3.2	64445.0
9	3.7	57189.0
10	3.9	63218.0
11	4.0	55794.0
12	4.0	56957.0
13	4.1	57081.0
14	4.5	61111.0
15	4.9	67938.0
16	5.1	66029.0
17	5.3	83088.0
18	5.9	81363.0
19	6.0	93940.0
20	6.8	91738.0
21	7.1	98273.0
22	7.9	101302.0
23	8.2	113812.0
24	8.7	109431.0
25	9.0	105582.0
26	9.5	116969.0
27	9.6	112635.0
28	10.3	122391.0
29	10.5	121872.0

In [3]:

```
data.head()
```

Out[3]:

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

In [4]:

```
data.head(10)
```

Out[4]:

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0
5	2.9	56642.0
6	3.0	60150.0
7	3.2	54445.0
8	3.2	64445.0
9	3.7	57189.0

In [5]:

```
data.tail()
```

Out[5]:

	YearsExperience	Salary
25	9.0	105582.0
26	9.5	116969.0
27	9.6	112635.0
28	10.3	122391.0
29	10.5	121872.0

In [6]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     float64
dtypes: float64(2)
memory usage: 608.0 bytes
```

In [7]:

```
# yearsExperience(x) -> Independent variable
# Salary(y) -> Dependent variable
```

In [8]:

```
data.shape
```

Out[8]:

```
(30, 2)
```

In [9]:

```
data.describe()
```

Out[9]:

	YearsExperience	Salary
count	30.000000	30.000000
mean	5.313333	76003.000000
std	2.837888	27414.429785
min	1.100000	37731.000000
25%	3.200000	56720.750000
50%	4.700000	65237.000000
75%	7.700000	100544.750000
max	10.500000	122391.000000

Data Pre-processing

Step1: Handle Missing Data

In [10]:

```
data.isnull().any()
```

Out[10]:

```
YearsExperience    False
Salary             False
dtype: bool
```

Step2: Convert text column if any. to numeric column

Its not required as there are no non numeric columns

Step3: Perform Data Visualization

In [11]:

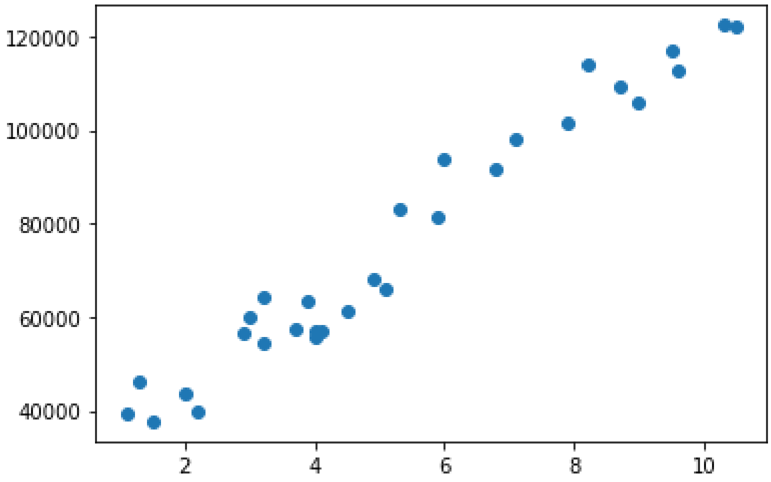
```
import matplotlib.pyplot as plt
```

In [12]:

```
plt.scatter(data.YearsExperience, data.Salary)
```

Out[12]:

```
<matplotlib.collections.PathCollection at 0x23cae99ddf0>
```



Step4: Split the data into dependent and independent variable

In [13]:

```
x = data.iloc[:,:1]
x
```

Out[13]:

	YearsExperience
0	1.1
1	1.3
2	1.5
3	2.0
4	2.2
5	2.9
6	3.0
7	3.2
8	3.2
9	3.7
10	3.9
11	4.0
12	4.0
13	4.1
14	4.5
15	4.9
16	5.1
17	5.3
18	5.9
19	6.0
20	6.8
21	7.1
22	7.9
23	8.2
24	8.7
25	9.0
26	9.5
27	9.6
28	10.3
29	10.5

In [14]:

```
y = data.iloc[:,1:2]
y
```

Out[14]:

```
      Salary
 0   39343.0
 1   46205.0
 2   37731.0
 3   43525.0
 4   39891.0
 5   56642.0
 6   60150.0
 7   54445.0
 8   64445.0
 9   57189.0
10   63218.0
11   55794.0
12   56957.0
13   57081.0
14   61111.0
15   67938.0
16   66029.0
17   83088.0
18   81363.0
19   93940.0
20   91738.0
21   98273.0
22  101302.0
23  113812.0
24  109431.0
25  105582.0
26  116969.0
27  112635.0
28  122391.0
29  121872.0
```

In [15]:

```
type(x)
```

Out[15]:

```
pandas.core.frame.DataFrame
```

In [16]:

```
type(y)
```

Out[16]:

```
pandas.core.frame.DataFrame
```

In [17]:

```
np.shape(x)
```

Out[17]:

```
(30, 1)
```

In [18]:

```
np.shape(y)
```

Out[18]:

```
(30, 1)
```

Step 5: Splitting the data into training and testing dataset

In [19]:

```
# Scikitlearn library has a train-test-fit function
```

In [20]:

```
from sklearn.model_selection import train_test_split
```

In [21]:

```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.2, random_state = 0)
```

In [22]:

```
print(x_train.shape) # Training input
```

```
(24, 1)
```

In [23]:

```
print(x_test.shape)  # testing input
```

```
(6, 1)
```

In [24]:

```
print(y_train.shape)  # training output
```

```
(24, 1)
```

In [25]:

```
print(y_test.shape)   # testing output
```

```
(6, 1)
```

Model Building - Linear Regression

In [26]:

```
from sklearn.linear_model import LinearRegression
```

Performing Linear regression by fitting the training data to the model

In [27]:

```
lr = LinearRegression()
lr.fit(x_train, y_train)
```

Out[27]:

```
LinearRegression()
```

Perform Testing

In [28]:

```
y_pred = lr.predict(x_test)
y_pred
```

Out[28]:

```
array([[ 40748.96184072],
       [122699.62295594],
       [ 64961.65717022],
       [ 63099.14214487],
       [115249.56285456],
       [107799.50275317]])
```

Compare Predicted value to actual value

In [29]:

```
y_pred  # predicted y
```

Out[29]:

```
array([[ 40748.96184072],
       [122699.62295594],
       [ 64961.65717022],
       [ 63099.14214487],
       [115249.56285456],
       [107799.50275317]])
```

In [30]:

```
y_test # actual y
```

Out[30]:

	Salary
2	37731.0
28	122391.0
13	57081.0
10	63218.0
26	116969.0
24	109431.0

Model Evaluation

In [31]:

```
# importing r2score metric
from sklearn.metrics import r2_score
```

In [32]:

```
# accuracy checking
acc = r2_score(y_pred, y_test)
acc
```

Out[32]:

```
0.986482673117654
```

Let us predict the value of y for some random input x

In [33]:

```
sal = lr.predict([[15]])
sal
```

Out[33]:

```
array([[166468.72605157]])
```

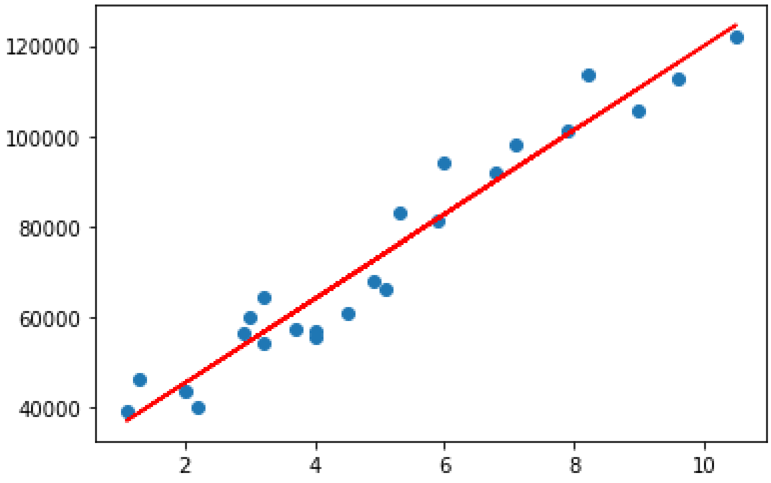
Draw the best-fit line

In [34]:

```
plt.scatter(x_train, y_train)
plt.plot(x_train, lr.predict(x_train),'r')
```

Out[34]:

```
[<matplotlib.lines.Line2D at 0x23cb6ab55b0>]
```



In [ ]:

```

```