

# ion-using-simple-linear-regression

October 23, 2023

## 1 Salary Prediction using Simple Linear Regression

Aim: Salary Prediction using Simple Linear Regression

Experiment no.: 7

```
[1]: #Name: Krutika Nemade  
      #Roll no.: 55  
      #Class: 3rd Year  
      #Sec:A
```

```
[2]: import pandas as pd  
      import matplotlib.pyplot as plt  
      import seaborn as sns  
      import numpy as np
```

```
[3]: import os
```

```
[4]: os.getcwd()
```

```
[4]: 'C:\\Users\\hp\\Desktop\\DSS Practicals'
```

```
[5]: os.chdir("C:\\Users\\hp\\Desktop")
```

```
[6]: df=pd.read_csv("Salary_Data.csv")
```

```
[7]: df.head()
```

```
[7]:   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
```

```
[8]: df.tail()
```

```
[8]:      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
```

```
[9]: df.head(30)
```

```
[9]:      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
```

```
[10]: df[5:15]
```

```
[10]:      YearsExperience      Salary
      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

```
[11]: df.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: 612.0 bytes
```

```
[12]: df.describe()
```

```
[12]:
```

	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

```
[13]: df.shape
```

```
[13]: (30, 2)
```

```
[14]: df.size
```

```
[14]: 60
```

```
[15]: df.ndim
```

```
[15]: 2
```

```
[16]: df.columns
```

```
[16]: Index(['YearsExperience', 'Salary'], dtype='object')
```

```
[17]: df.isnull()
```

```
[17]:   YearsExperience  Salary
0             False   False
1             False   False
2             False   False
3             False   False
4             False   False
5             False   False
6             False   False
7             False   False
8             False   False
9             False   False
10            False   False
11            False   False
12            False   False
13            False   False
14            False   False
15            False   False
16            False   False
17            False   False
18            False   False
19            False   False
20            False   False
21            False   False
22            False   False
23            False   False
24            False   False
25            False   False
26            False   False
27            False   False
28            False   False
29            False   False
```

```
[18]: df.isnull().sum()
```

```
[18]: YearsExperience    0
Salary                0
dtype: int64
```

```
[19]: #Assigning values in X & Y
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values

#X = df['YearsExperience']
```

```
#y = df['Salary']
```

```
[20]: print(X)
```

```
[[ 1.1]
 [ 1.3]
 [ 1.5]
 [ 2. ]
 [ 2.2]
 [ 2.9]
 [ 3. ]
 [ 3.2]
 [ 3.2]
 [ 3.7]
 [ 3.9]
 [ 4. ]
 [ 4. ]
 [ 4.1]
 [ 4.5]
 [ 4.9]
 [ 5.1]
 [ 5.3]
 [ 5.9]
 [ 6. ]
 [ 6.8]
 [ 7.1]
 [ 7.9]
 [ 8.2]
 [ 8.7]
 [ 9. ]
 [ 9.5]
 [ 9.6]
 [10.3]
 [10.5]]
```

```
[21]: print(y)
```

```
[ 39343.  46205.  37731.  43525.  39891.  56642.  60150.  54445.  64445.
  57189.  63218.  55794.  56957.  57081.  61111.  67938.  66029.  83088.
  81363.  93940.  91738.  98273. 101302. 113812. 109431. 105582. 116969.
 112635. 122391. 121872.]
```

```
[22]: #Splitting testdata into X_train,X_test,y_train,y_test
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=.
↪3,random_state=42)
```

```
[23]: print(X_train)
```

```
[[ 1.1]
 [ 2.2]
 [ 5.1]
 [ 2.9]
 [ 4.1]
 [ 4. ]
 [ 7.9]
 [ 1.3]
 [ 1.5]
 [ 9. ]
 [ 2. ]
 [ 7.1]
 [ 9.5]
 [ 5.9]
 [10.5]
 [ 6.8]
 [ 3.2]
 [ 3.9]
 [ 4.5]
 [ 6. ]
 [ 3. ]]
```

```
[24]: print(X_test)
```

```
[[ 9.6]
 [ 4.9]
 [ 8.2]
 [ 5.3]
 [ 3.2]
 [ 3.7]
 [10.3]
 [ 8.7]
 [ 4. ]]
```

```
[25]: print(y_train)
```

```
[ 39343.  39891.  66029.  56642.  57081.  55794. 101302.  46205.  37731.
 105582.  43525.  98273. 116969.  81363. 121872.  91738.  54445.  63218.
  61111.  93940.  60150.]
```

```
[26]: print(y_test)
```

```
[112635.  67938. 113812.  83088.  64445.  57189. 122391. 109431.  56957.]
```

```
[27]: from sklearn.linear_model import LinearRegression  
lr = LinearRegression()  
lr.fit(X_train, y_train)
```

```
[27]: LinearRegression()
```

```
[28]: #Assigning Coefficient (slope) to m  
m = lr.coef_
```

```
[29]: print("Coefficient :", m)
```

```
Coefficient : [9339.08172382]
```

```
[30]: #Assigning Y-intercept to a  
c = lr.intercept_
```

```
[31]: print("Intercept :", c)
```

```
Intercept : 25918.438334893202
```

```
[32]: lr.score(X_test, y_test)*100
```

```
[32]: 94.14466227178214
```