```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
```

```
data = pd.read_csv("salary.csv")
```

```
data.isnull().sum()
```

```
YearsExperience    0
Salary             0
dtype: int64
```

```
print(data)
```

```
    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
```

```
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
```

```
data.describe()
```

	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

```
display(data)
```

	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

```
X = data.iloc[:,:-1].values
Y = data.iloc[:,1].values
```

```
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]]
```

```
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.]
```

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

```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y, test_size=1/3, random_state=0)
```

```
print(X_train,Y_train)
```

```
[[ 2.9]
 [ 5.1]
 [ 3.2]
 [ 4.5]
 [ 8.2]
 [ 6.8]
 [ 1.3]
 [10.5]
 [ 3. ]
 [ 2.2]
 [ 5.9]
 [ 6. ]
 [ 3.7]
 [ 3.2]
 [ 9. ]
 [ 2. ]
 [ 1.1]
 [ 7.1]
 [ 4.9]
 [ 4. ]] [ 56642.  66029.  64445.  61111. 113812.  91738.  46205. 121872.  60150.
  39891.  81363.  93940.  57189.  54445. 105582.  43525.  39343.  98273.
  67938.  56957.]
```

```
print(X_test)
```

```
[[ 1.5]
 [10.3]
 [ 4.1]
 [ 3.9]
 [ 9.5]
 [ 8.7]
 [ 9.6]
 [ 4. ]
 [ 5.3]
 [ 7.9]]
```

```
print(Y_test)
```

```
[ 37731. 122391.  57081.  63218. 116969. 109431. 112635.  55794.  83088.
 101302.]
```

```
print(X_test,Y_test)
```

```
[[ 1.5]
 [10.3]
 [ 4.1]
 [ 3.9]
 [ 9.5]
 [ 8.7]
 [ 9.6]
 [ 4. ]
 [ 5.3]
 [ 7.9]] [ 37731. 122391.  57081.  63218. 116969. 109431. 112635.  55794.  83088.
 101302.]
```

```
#step -3 fit simple linear Regression to training data
from sklearn.linear_model import LinearRegression
```

```
rg = LinearRegression()
```

```
rg.fit(X_train , Y_train)
```

```
LinearRegression()
```

```
#step 4: make prediction
Y_pred = rg.predict(X_test)
```

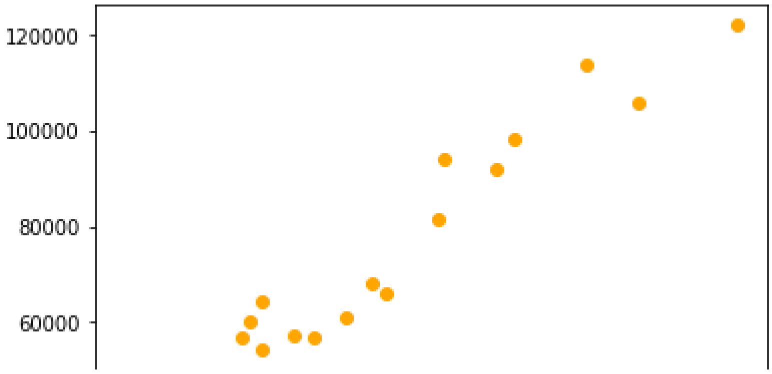
```
print(Y_test, Y_pred)
```

```
[ 37731. 122391.  57081.  63218. 116969. 109431. 112635.  55794.  83088.
 101302.] [ 40835.10590871 123079.39940819  65134.55626083  63265.36777221
 115602.64545369 108125.8914992  116537.23969801  64199.96201652
  76349.68719258 100649.1375447 ]
```

```
#step 5:visualize training set results
import matplotlib.pyplot as plt
# plot the actual data points of trainng set
```

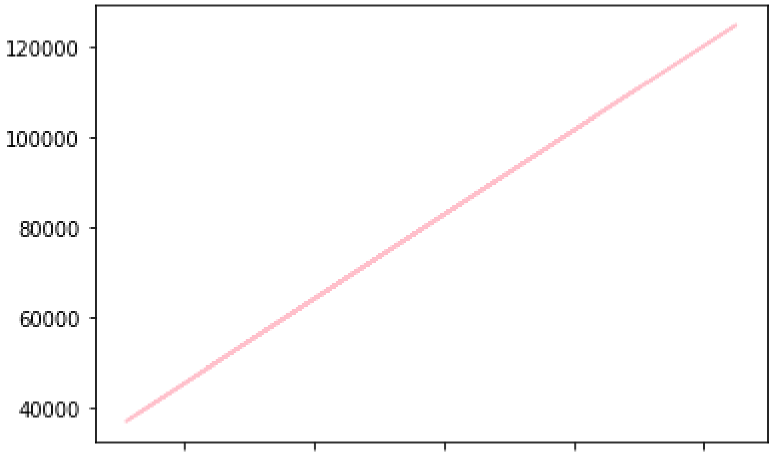
```
plt.scatter(X_train, Y_train, color = 'orange')
#first we will plot the actuAL
#plot the regression line
```

```
<matplotlib.collections.PathCollection at 0x7f80a635b810>
```



```
plt.plot(X_train, rg.predict(X_train),color = 'pink')
```

```
[<matplotlib.lines.Line2D at 0x7f80a62cca50>]
```



```
plt.title('salary vs Experience(Traing test)')
```

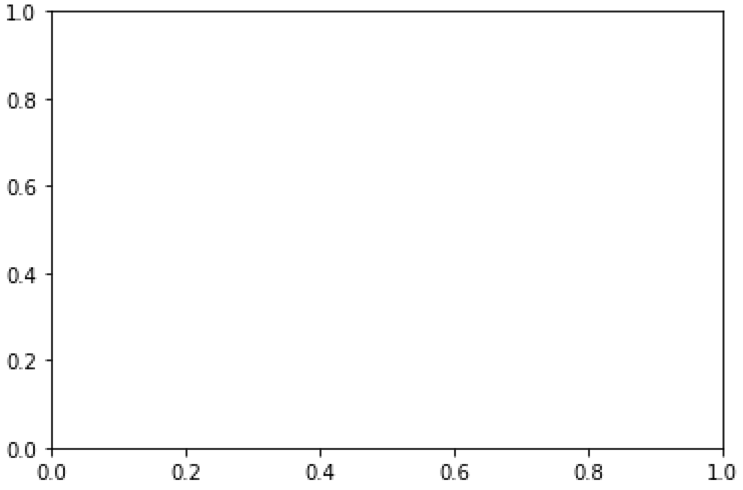
```
Text(0.5, 1.0, 'salary vs Experience(Traing test)')
```

```
plt.ylabel('Salary')
```

```
Text(0, 0.5, 'Salary')
```

```
plt.xlabel('Year of Expereince')
```

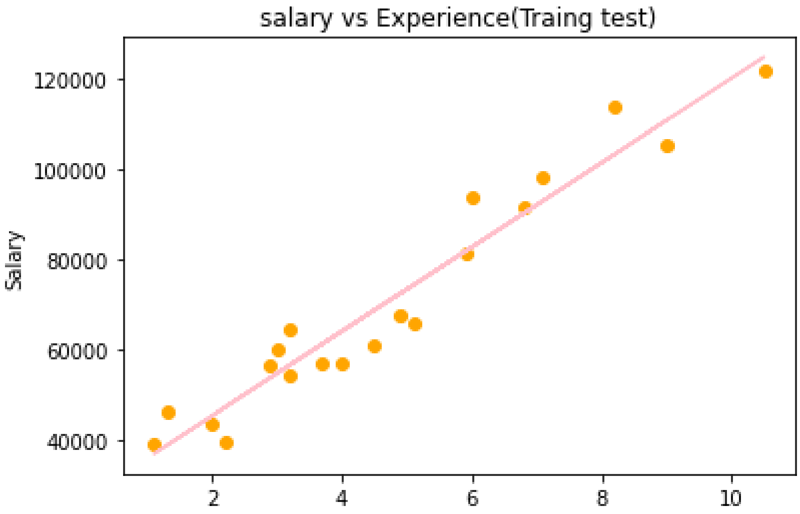
```
Text(0.5, 0, 'Year of Expereince')
```



```
plt.title('salary vs Experience(Traing test')
plt.xlabel('Year of Expereince')
plt.ylabel('Salary')
```

```
Text(0, 0.5, 'Salary')
```

```
plt.scatter(X_train, Y_train, color = 'orange')
plt.plot(X_train, rg.predict(X_train),color = 'pink')
plt.title('salary vs Experience(Traing test)')
plt.xlabel('Year of Expereince')
plt.ylabel('Salary')
plt.show()
```



```
#s 7: Make new prediction
new_salary_pred = rg.predict([[15]])
```

```
print('The predict salary of a person with in 10 years Expereince is :',new_salary_pred)
```

```
    The predict salary of a person with in 10 years Expereince is : [167005.32889087]
```

```
new_salary_pred = rg.predict([[30]])
```

```
print('ur salary will be :',new_salary_pred)
```

```
    ur salary will be : [307194.4655377]
```

```
print('R square =' ,metrics.r2_score(Y_test, Y_pred))
```

```
    R square = 0.9749154407708353
```

```
print('Mean square Error =' ,metrics.mean_absolute_error(Y_test,Y_pred))
```

```
    Mean square Error = 3426.4269374307078
```

```
print('Mean absolute Error =' ,metrics.mean_absolute_error(Y_test,Y_pred))
```

```
    Mean absolute Error = 3426.4269374307078
```