ASSIGNMENT 7

September 15, 2021

1 ASSIGNMENT 7

2 SOUMYA GITE

3 SIRSS2276

[2]:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib as mpl
from sklearn.model_selection import train_test_split
```

[3]:

```python
data = pd.read_csv('Salary_Data.csv')
data.head(10)
```

[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
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
```

[4]:

```python
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
```

[5]:
```
data.shape
```

[5]:
```
(30, 2)
```

[6]:
```
data.columns
```

[6]:
```
Index(['YearsExperience', 'Salary'], dtype='object')
```

[7]:
```
data.describe()
```

[7]:
```
       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
```

[8]:
```
data.isnull().sum()
```

[8]:
```
YearsExperience    0
Salary             0
dtype: int64
```

[9]:
```
data.duplicated().sum()
```

[9]:
```
0
```

[10]:
```
X = data.iloc[:,:-1].values
Y = data.iloc[:,1].values
```

[11]:
```
class LinearRegression() :

    def __init__(self , learning_rate , iterations) :

        self.learning_rate = learning_rate

        self.iterations = iterations

    def fit(self , X , Y) :

        self.m, self.n = X.shape

        self.W = np.zeros(self.n)

        self.b = 0

        self.X = X

        self.Y = Y


        # gradient descent learning

        for i in range(self.iterations) :

            self.update_weights()

        return self

    def update_weights(self) :

        Y_pred = self.predict(self.X)

        # calculate gradients

        dW = -( 2 * (self.X.T).dot(self.Y - Y_pred )) / self.m

        db = - 2 * np.sum(self.Y - Y_pred ) / self.m

        # update weights

        self.W = self.W - self.learning_rate * dW

        self.b = self.b - self.learning_rate * db

        return self
    def predict(self , X) :

            return X.dot(self.W) + self.b
```

```
[12]: X_train, X_test, Y_train, Y_test = train_test_split( X, Y, test_size = 1/3,␣
       ↪random_state = 0 )
```

```
[13]: model = LinearRegression( iterations = 1000, learning_rate = 0.01 )
      model.fit( X_train, Y_train )
```

```
[13]: <__main__.LinearRegression at 0x1d30ea1f130>
```

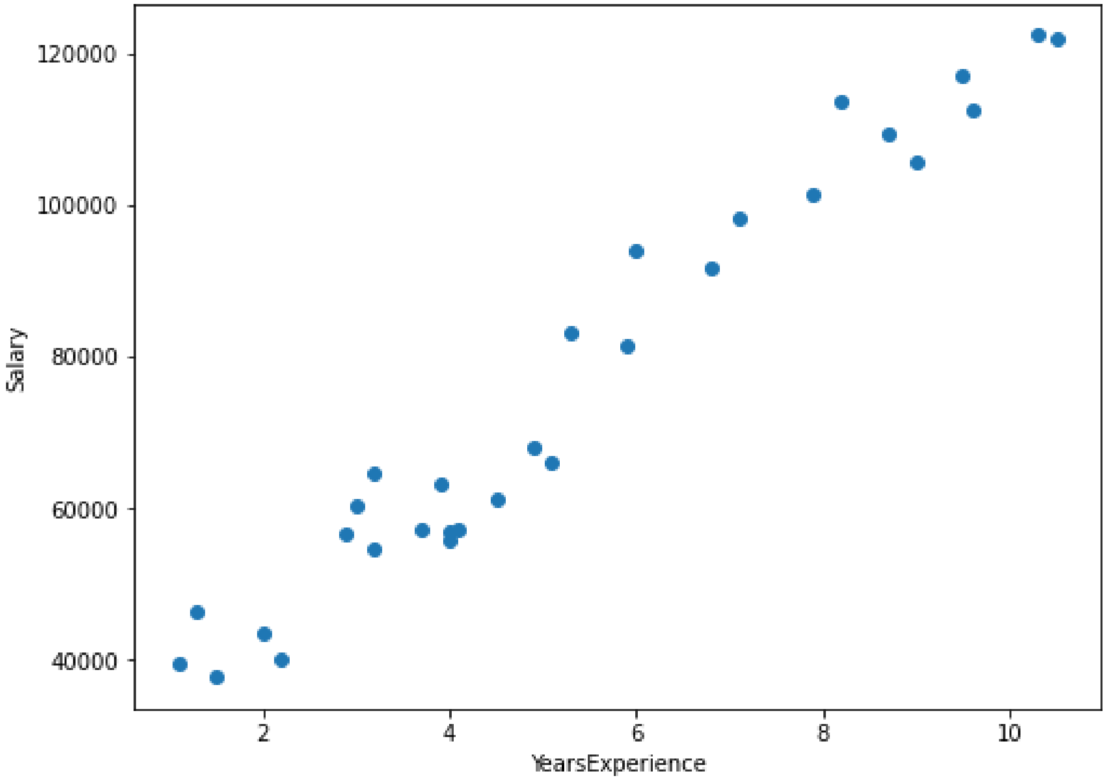
[14]:

```
Y_pred = model.predict( X_test )
print( "Predicted values ", np.round( Y_pred[:3], 2 ) )
print( "Real values      ", Y_test[:3] )
print( "Trained W        ", round( model.W[0], 2 ) )
print( "Trained b        ", round( model.b, 2 ) )
```

```
Predicted values  [ 40594.69 123305.18  65031.88]
Real values       [ 37731. 122391.  57081.]
Trained W         9398.92
Trained b         26496.31
```

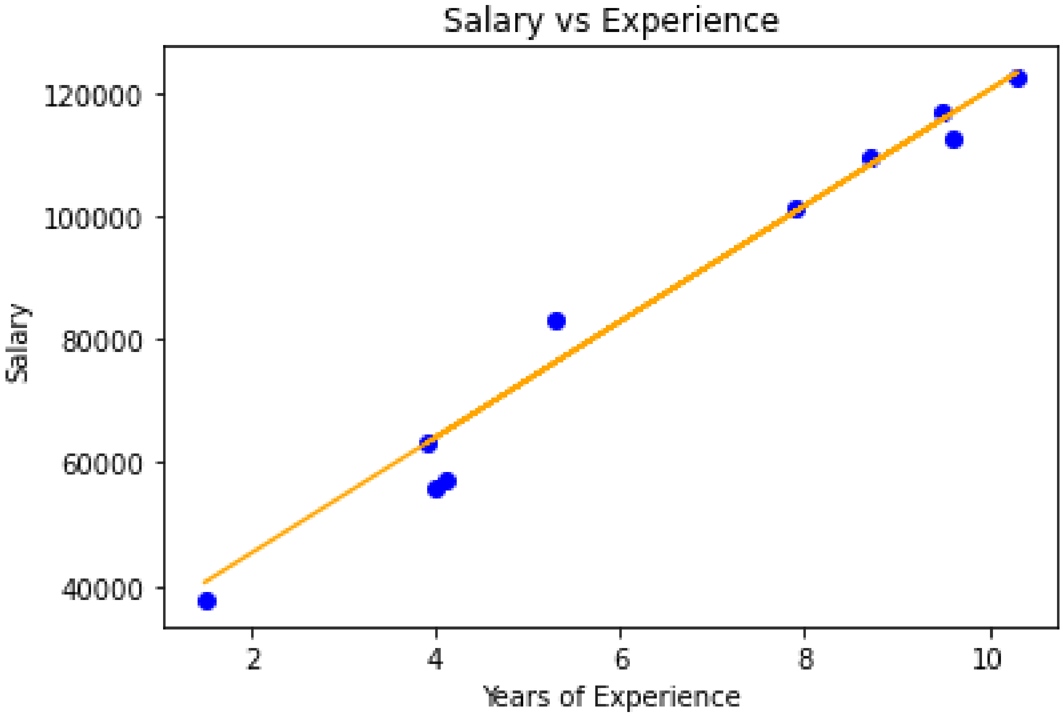
[15]:

```
plt.figure(figsize=(8,6))
plt.scatter(X, Y)
plt.xlabel('YearsExperience')
plt.ylabel('Salary')
plt.show()
```



[16]:

```
plt.scatter( X_test, Y_test, color = 'blue' )
plt.plot( X_test, Y_pred, color = 'orange' )
plt.title( 'Salary vs Experience' )
plt.xlabel( 'Years of Experience' )
plt.ylabel( 'Salary' )
plt.show()
```


Salary vs Experience
Salary
120000
100000
80000
60000
40000
2
4
6
8
10
Years of Experience

```
[ ]:
```