ALBUKHARY INTERNATIONAL UNIVERSITY

Artificial Intelligence

LAB EIGHT (a)

Simple Linear Regression

Simple Linear regression algorithm has mainly two objectives:

- Model the relationship between the two variables. Such as the relationship between Income and expenditure, experience and Salary, etc.
- Forecasting new observations. Such as Weather forecasting according to temperature, Revenue of a company according to the investments in a year, etc.

To implement the Simple Linear regression model in machine learning using Python, we need to follow the below steps:

- Data Pre-processing
- Fitting the Simple Linear Regression to the Training Set
- Prediction of test set result
- Visualizing the Training set results
- Visualizing the Test set results

1. Data Pre-processing

Importing the libraries:

In [2]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Loading the dataset:

In [3]:

```
data= pd.read_csv('Salary_Data.csv')
```

In [4]:

```
data.head()
```

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

In [5]:

```
data.shape
```

Out[5]:

```
(30, 2)
```

In [ ]:

```
data.describe()
```

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 [ ]:

```
data
```

Extracting the dependent and independent variables:

In [7]:

```
x= data.iloc[:, :-1].values
y= data.iloc[:, 1].values
```

In [8]:

```
x
```

Out[8]:

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

In [9]:

```
y
```

Out[9]:

```
array([ 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.])
```

splitting the variables into test set and training set:

We have 30 observations, so we will take 20 observations for the training set and 10 observations for the test set

In [10]:

```
# Splitting the dataset into training and test set.
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)
```

In [11]:

```
print(x_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. ]]
```

In [12]:

```
print(x_test)
```

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

In [13]:

```
print(y_train)
```

```
[ 56642.  66029.  64445.  61111. 113812.  91738.  46205. 121872.  6015
0.
  39891.  81363.  93940.  57189.  54445. 105582.  43525.  39343.  9827
3.
  67938.  56957.]
```

In [14]:

```
print(y_test)
```

```
[ 37731. 122391.  57081.  63218. 116969. 109431. 112635.  55794.  8308
8.
 101302.]
```

2. Fitting the Simple Linear Regression to the Training Set

Now the second step is to fit our model to the training dataset. To do so, we will import the LinearRegression class of the *linear_model* library from the *scikit learn*. After importing the class, we are going to create an object of the class named as a $regressor$.

In [15]:

```
#Fitting the Simple Linear Regression model to the training dataset
from sklearn.linear_model import LinearRegression
regressor= LinearRegression()
regressor.fit(x_train, y_train)
```

Out[15]:

```
LinearRegression()
```

In the above code, we have used a $fit()$ method to fit our Simple Linear Regression object to the training set. In the $fit()$ function, we have passed the *x_train* and *y_train*, which is our training dataset for the dependent and an independent variable. We have fitted our regressor object to the training set so that the model can easily learn the correlations between the predictor and target variables.

3. Prediction of test set result:

Dependent (salary) and an independent variable (Experience). So, now, our model is ready to predict the output for the new observations. In this step, we will provide the test dataset (new observations) to the model to check whether it can predict the correct output or not.

We will create a prediction vector *y_pred*, and *x_pred*, which will contain predictions of test dataset, and prediction of training set respectively.

In [16]:

```
#Prediction of Test and Training set result
y_pred= regressor.predict(x_test)
x_pred= regressor.predict(x_train)
```

In [17]:

```
print(y_pred)
```

```
[ 40835.10590871 123079.39940819  65134.55626083  63265.36777221
 115602.64545369 108125.8914992  116537.23969801  64199.96201652
  76349.68719258 100649.1375447 ]
```

In [18]:

```
print(x_pred)
```

```
[ 53919.42532909  74480.49870396  56723.20806202  68872.93323808
 103452.92027763  90368.60085726  38965.91742009 124948.58789682
  54854.0195734   47377.2656189   81957.25265845  82891.84690277
  61396.17928358  56723.20806202 110929.67423213  45508.07713028
  37096.72893147  93172.3835902   72611.31021533  64199.96201652]
```

In [ ]:

```

```

4. Visualizing the Training set results

Now in this step, we will visualize the training set result. To do so, we will use the $scatter()$ function of the $pyplot$ library, which we have already imported in the pre-processing step. The $scatter()$ function will create a scatter plot of observations.

In the *x-axis*, we will plot the Years of Experience of employees and on the *y-axis*, salary of employees. In the function, we will pass the real values of training set, which means a year of experience *x_train*, training set of Salaries *y_train*, and color of the observations.

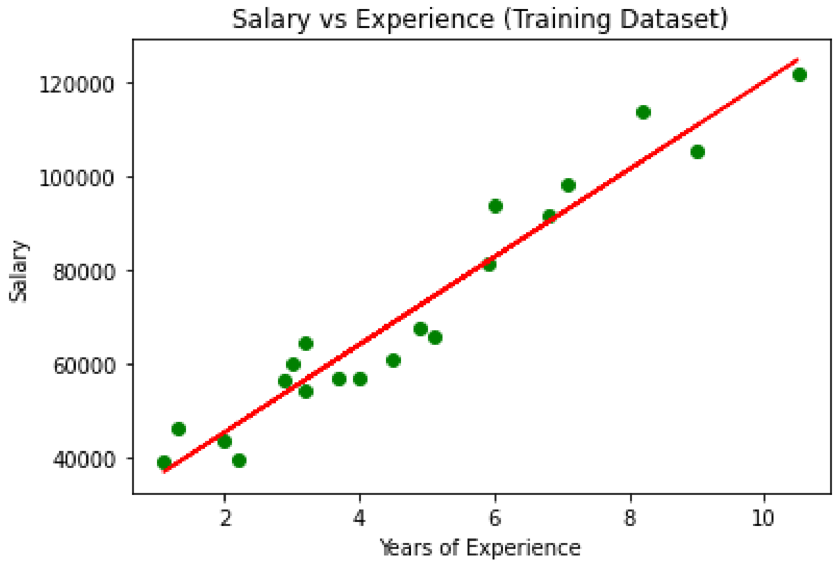
Now, we need to plot the regression line, so for this, we will use the $plot()$ function of the $pyplotlibrary$. In this function, we will pass the years of experience for training set, predicted salary for training set *x_pred*, and color of the line.

Next, we will give the title for the plot. So here, we will use the $title()$ function of the pyplot library and pass the name ("Salary vs Experience (Training Dataset)".

After that, we will assign labels for *x-axis* and *y-axis* using $xlabel()$ and $ylabel()$ function.

In [19]:

```
plt.scatter(x_train, y_train, color="green")
plt.plot(x_train, x_pred, color="red")
plt.title("Salary vs Experience (Training Dataset)")
plt.xlabel("Years of Experience")
plt.ylabel("Salary")
plt.show()
```



The good fit of the line can be observed by calculating the difference between actual values and predicted values. But as we can see in the above plot, most of the observations are close to the regression line, hence our model is good for the training set.

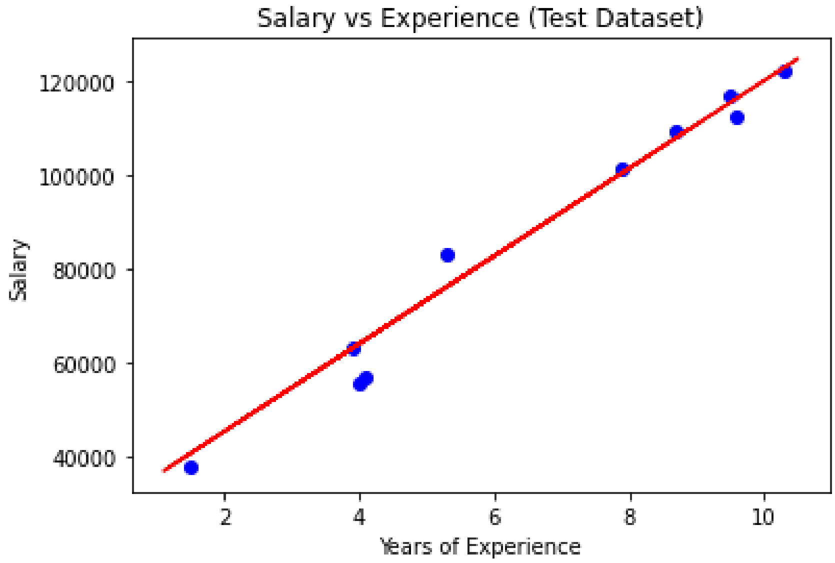
5. Visualizing the Test set results

In the previous step, we have visualized the performance of our model on the training set. Now, we will do the same for the Test set. The complete code will remain the same as the above code, except in this, we will use *x_test*, and *y_test* instead of *x_train* and *y_train*.

Here we are also changing the color of observations and regression line to differentiate between the two plots, but it is optional.

In [20]:

```
#visualizing the Test set results
plt.scatter(x_test, y_test, color="blue")
plt.plot(x_train, x_pred, color="red")
plt.title("Salary vs Experience (Test Dataset)")
plt.xlabel("Years of Experience")
plt.ylabel("Salary")
plt.show()
```



In the above plot, there are observations given by the blue color, and prediction is given by the red regression line. As we can see, most of the observations are close to the regression line, hence we can say our Simple Linear Regression is a good model and able to make good predictions.

In [21]:

```
import sklearn.metrics as sm

# Compute performance metrics
print("Linear regressor performance:")
print("R2 score =", round(sm.r2_score(y_test, y_pred), 2))
```

```
Linear regressor performance:
R2 score = 0.97
```

In [ ]: