Salary Prediction With Simple Linear Regression

October 10, 2021

0.1 Simple linear Regression

0.1.1 Simple linear regression is a regression model that estimates the relationship between one independent variable and one dependent variable using a straight line. Both variables should be quantitative.

0.1.2 For example, the relationship between temperature and the expansion of mercury in a thermometer can be modeled using a straight line: as temperature increases, the mercury expands. This linear relationship is so certain that we can use mercury thermometers to measure temperature.

Simple linear regression formula The formula for a simple linear regression is: # y = B0 + B1X + e y is the predicted value of the dependent variable (y) for any given value of the independent variable (x).

B0 is the intercept, the predicted value of y when the x is 0.

B1 is the regression coefficient – how much we expect y to change as x increases.

x is the independent variable (the variable we expect is influencing y).

e is the error of the estimate, or how much variation there is in our estimate of the regression coefficient.

0.2 Importing the Libraries

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[2]:
```
from sklearn.model_selection import train_test_split
## Splitting the dataset into the Training set and test set
```

[3]:
```
from sklearn.linear_model import LinearRegression
## Training the simple linear Regression Model
```

[15]:
```
import statsmodels.api as sm
```

0.3 Importing the Dataset

[4]:
```
data_set = pd.read_csv('Salary_Data.csv')
```

[5]:
```
x = data_set.iloc[:, :-1].values
y = data_set.iloc[:, -1].values
```

0.4 Splitting the dataset into the Training set and test set

[16]:
```
data_set.describe()
```

[16]:

	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

[7]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y, test_size = 0.2,␣
↪random_state = 0)
```

0.5 Training the simple linear Regression Model on the Training set

[8]:
```
regressor = LinearRegression()
regressor.fit(x_train, y_train)
```

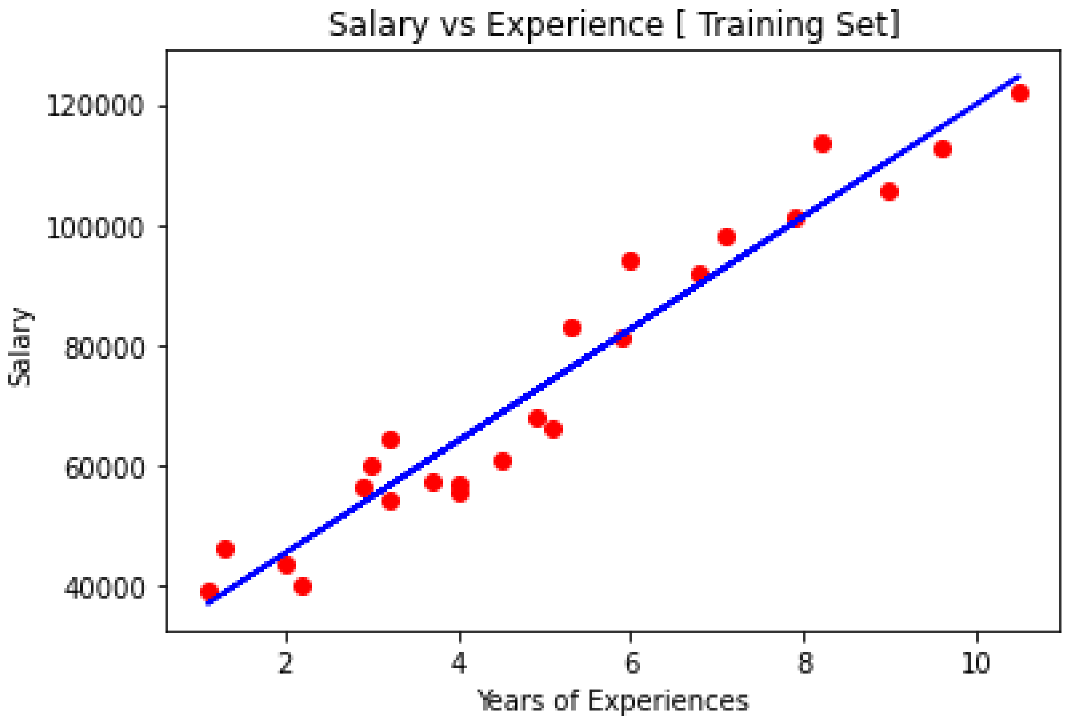
[8]: LinearRegression()

0.6 Predicting the Test set result

[11]:
```
y_predict = regressor.predict(x_test)
```

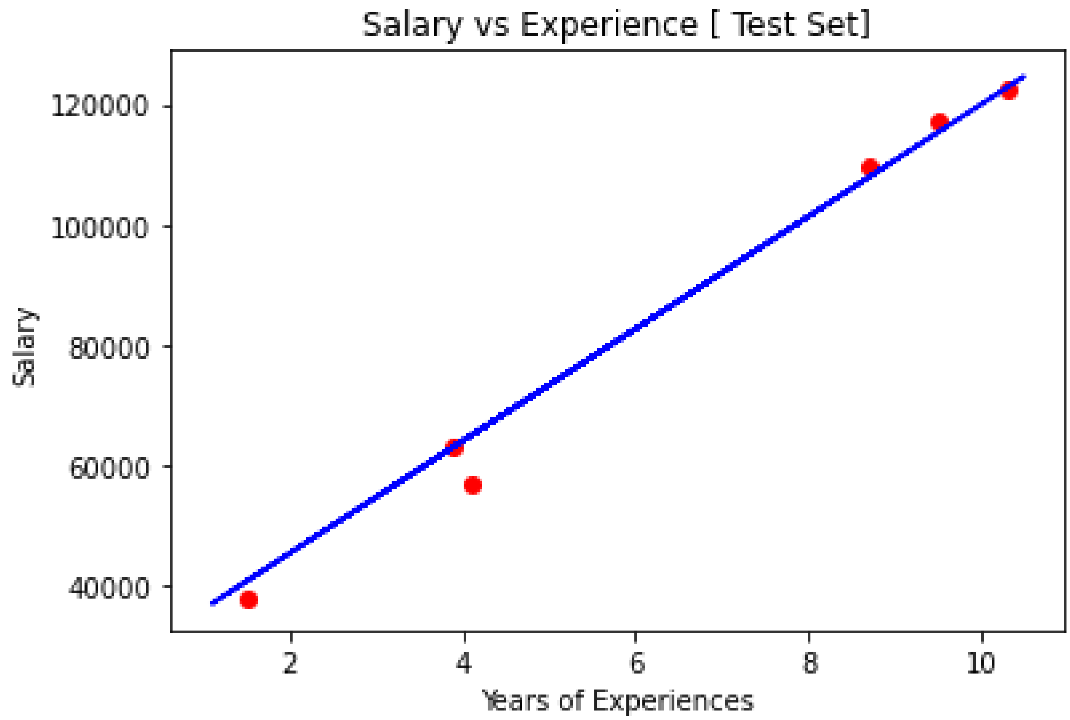
0.7 Visualising the Training set results

[12]:
```
plt.scatter(x_train, y_train, color = 'red')
plt.plot(x_train, regressor.predict(x_train), color ="blue")
plt.title("Salary vs Experience [ Training Set]")
plt.xlabel("Years of Experiences")
plt.ylabel("Salary")
plt.show()
```



0.8 Visualising the Test set results

[13]:
```
plt.scatter(x_test, y_test, color = 'red')
plt.plot(x_train, regressor.predict(x_train), color ="blue")
plt.title("Salary vs Experience [ Test Set]")
plt.xlabel("Years of Experiences")
plt.ylabel("Salary")
plt.show()
```



0.9 Regression Itself

[18]:
```
x_stats = sm.add_constant(x)
results = sm.OLS(y,x_stats).fit()
results.summary()
```

[18]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.957
Model:                            OLS   Adj. R-squared:                  0.955
Method:                 Least Squares   F-statistic:                     622.5
Date:                Sun, 10 Oct 2021   Prob (F-statistic):           1.14e-20
Time:                        17:09:50   Log-Likelihood:                -301.44
No. Observations:                  30   AIC:                             606.9
Df Residuals:                      28   BIC:                             609.7
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       2.579e+04   2273.053     11.347      0.000    2.11e+04    3.04e+04
```

```
x1          9449.9623    378.755     24.950      0.000    8674.119    1.02e+04
==============================================================================
Omnibus:                        2.140   Durbin-Watson:                   1.648
Prob(Omnibus):                  0.343   Jarque-Bera (JB):                1.569
Skew:                           0.363   Prob(JB):                        0.456
Kurtosis:                       2.147   Cond. No.                         13.2
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

[ ]: