GETTING STARTED

How to Estimate Salary with Linear Regression

A beginner-friendly guide to linear regression using Python.

Bee Guan Teo Sep 5, 2021 · 6 min read ★

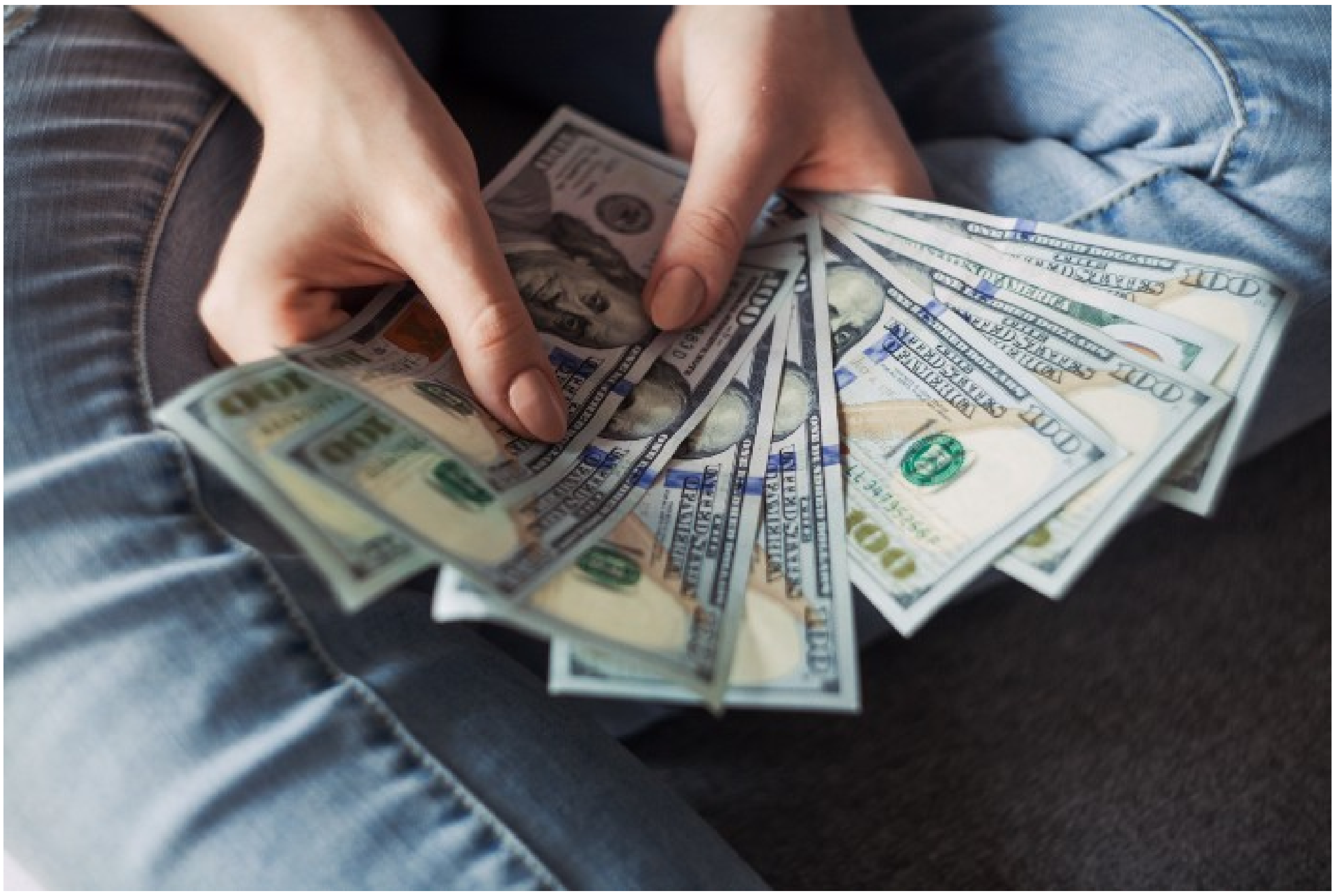


Photo by Alexander Mils on Unsplash

data-driven approach to verify the fact. We will use **linear regression** to model the relationship between the amount of salary with the years of working experience.

Linear regression is a model that assumes a linear relationship between an **explanatory variable (X) and a response variable (y)**. We can predict the value of y based on the value of X. In our context here, the estimated salary will be our response variable (y) since it is our target predicted value and the years of working experience will be our explanatory variable (X).

Once we have defined the explanatory variable and response variable in our case, we will use **Python** to build a linear regression model to address our question.

Prerequisite Python Libraries

Pandas — https://pandas.pydata.org/

Numpy — https://numpy.org/

Matplotlib — https://matplotlib.org/

scikit-learn — https://scikit-learn.org/stable/

Github

The original full source codes presented in this article are available on my Github Repo. Feel free to download it (*LinearRegression.py*) if you wish to use it to follow my article.

Acquisition of data

The data of salary (*Salary_Data.csv*) that we will use for linear regression can be obtained from Kaggle.

Linear Regression

1. Loading data

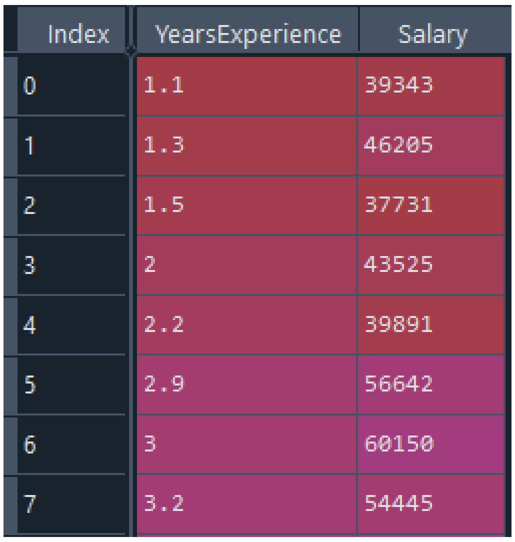
Firstly, we will use the *Python Pandas* library to read our CSV data.

```
1   import pandas as pd
2   import numpy as np
3   from sklearn.model_selection import train_test_split
4   from sklearn import linear_model
5   import sklearn.metrics as sm
6   import matplotlib.pyplot as plt
7
8   data = pd.read_csv("Data/Salary_Data.csv")
9   X = data['YearsExperience']
10  y = data['Salary']
```

Linear_Regression_Part1.py hosted with ❤ by GitHub view raw

Line 1–6: Import all the required libraries.

Line 8: Use the *Pandas read_csv* function to read the CSV file. This function will return the data in a dataframe format.



Index	YearsExperience	Salary
0	1.1	39343
1	1.3	46205
2	1.5	37731
3	2	43525
4	2.2	39891
5	2.9	56642
6	3	60150
7	3.2	54445

Image Prepared by the Author

Line 9–10: Extract the column of *YearsExperience* and *Salary* and assign them to the variables X and y, respectively.

2. Splitting data into a training set and a test set

train the model whereas the test set will be used to assess the performance of the trained model in predicting the result from unseen data.

We can use the *Python scikit-learn train_test_split* function to randomly split our data into a training and test set.

```
1   X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, random_state=42)
```

Linear_Regression_Part2.py hosted with ❤ by GitHub view raw

Line 1: Set apart 30% of the entire dataset as the test set and assign the training set and test set into four variables, respectively.

3. Data Transformation

Python scikit-learn only accepts the training and test data in a 2-dimensional array format. We have to perform data transformation on our training set and test set.

```
1   X_train = np.array(X_train).reshape((len(X_train),1))
2   y_train = np.array(y_train).reshape((len(y_train),1))
3
4   X_test = np.array(X_test).reshape(len(X_test), 1)
5   y_test = np.array(y_test).reshape(len(y_test), 1)
```

Linear_Regression_Part3.py hosted with ❤ by GitHub view raw

Line 1–2: Use *Numpy reshape function* to transform the training set from 1-dimensional series to a 2-dimensional array.

Line 3–4: Use *Numpy reshape function* to transform the test set from 1-dimensional series to a 2-dimensional array.

4. Training Model

Now we are ready to train our linear model.

```
1   model = linear_model.LinearRegression()
2   model.fit(X_train, y_train)
```

Linear_Regression_Part4.py hosted with ❤ by GitHub view raw

Line 1: Use the *Scikit-Learn LinearRegression* function to create a model object.

At this stage, we have trained a linear model and we first use it to predict the salary on our training set to see how well it fit on the data.

```
1   y_train_pred = model.predict(X_train)
2
3   plt.figure()
4   plt.scatter(X_train, y_train, color='blue', label="True Value")
5   plt.plot(X_train, y_train_pred, color='black', linewidth=2, label="Prediction")
6   plt.xlabel("Years of Experiences")
7   plt.ylabel("Salary")
8   plt.title('Prediction Result of Training Data')
9   plt.legend()
10  plt.show()
```

Linear_Regression_Part5.py hosted with ❤ by GitHub view raw

Line 1: Use the linear model to predict the salary based on the training set.

Line 3–10: Use the Matplotlib to create a plot to visualize the predicted results. The "true values" are plotted as the blue dots on the chart and the predicted values are plotted as a black color straight line.

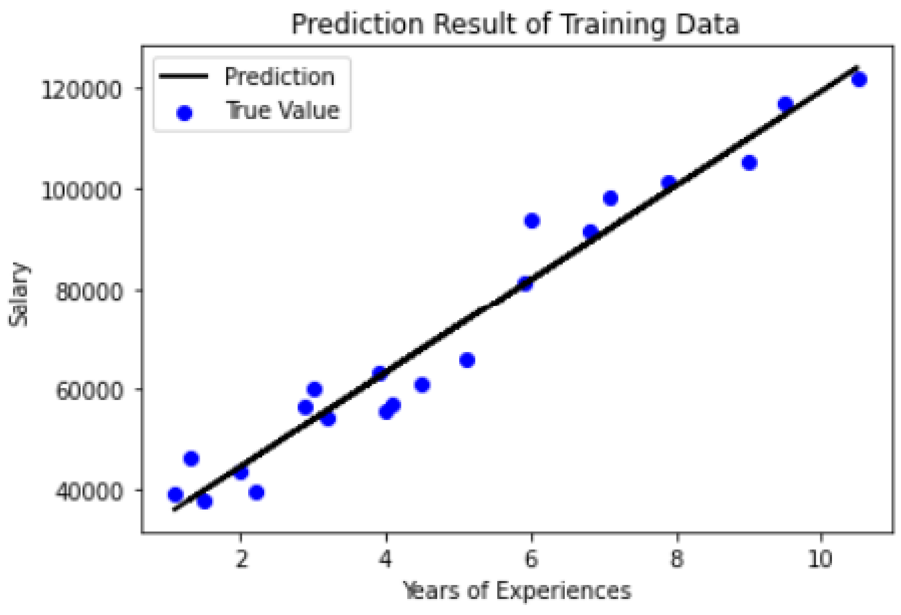


Image Prepared by the Author

In general, the linear model fits well on the training data. This shows a linear relationship between the salary and years of experience.

```
1   y_test_pred = model.predict(X_test)
2
3   plt.figure()
4   plt.scatter(X_test, y_test, color='green', label='True Value')
5   plt.plot(X_test, y_test_pred, color='black', linewidth=2, label='Prediction')
6   plt.xlabel("Years of Experiences")
7   plt.ylabel("Salary")
8   plt.title('Prediction Result of Test data')
9   plt.legend()
10  plt.show()
```

Line 1: Use the linear model to predict the salary based on the test set.

Line 3–10: Use the Matplotlib to create a plot to visualize the predicted results. The “true values” are plotted as the green dots on the chart and the predicted values are plotted as a black color straight line.

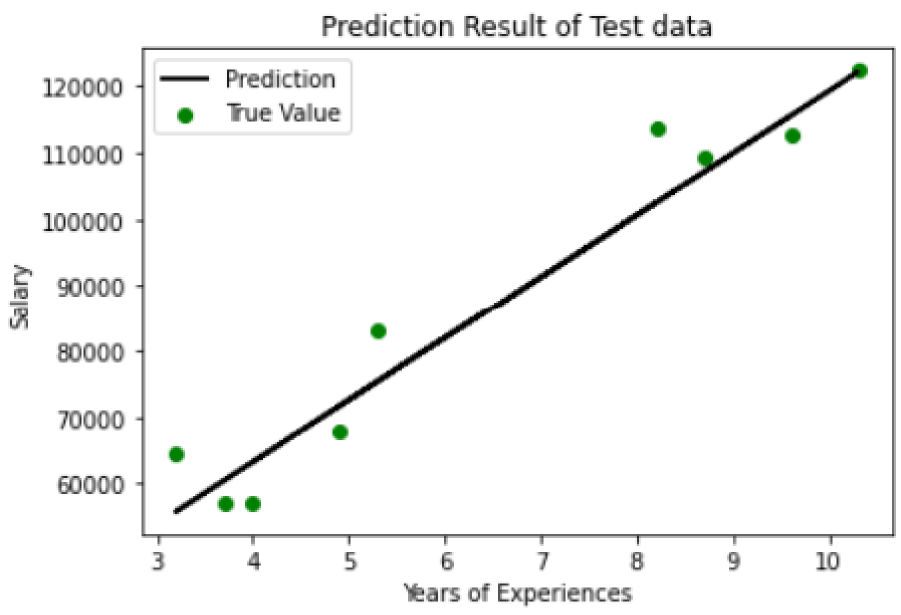


Image Prepared by the Author

The graph shows that our linear model can fit quite well on the test set. We can observe a linear pattern of how the amount of salary is increased by the years of experience.

6. Model Evaluation

use some quantitative methods to obtain a more precise performance evaluation of our linear model.

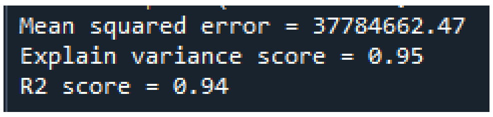
We will use three types of quantitative metrics:

- **Mean Square Error** — The average of the squares of the difference between the true values and the predicted values. The lower the difference the better the performance of the model. This is a common metric used for regression analysis.
- **Explained Variance Score** — A measurement to examine how well a model can handle the variation of values in the dataset. A score of 1.0 is the perfect score.
- **R2 Score** — A measurement to examine how well our model can predict values based on the test set (unknown samples). The perfect score is 1.0.

```
1  print("Mean squared error =", round(sm.mean_squared_error(y_test, y_test_pred), 2))
2  print("Explain variance score =", round(sm.explained_variance_score(y_test, y_test_pred), 2))
3  print("R2 score =", round(sm.r2_score(y_test, y_test_pred), 2))
```

Linear_Regression_Part7.py hosted with ❤ by GitHub view raw

Line 1–3: Use the *Scikit-learn Metrics* functionalities to calculate the mean square error, explained variance, and R2 score for our linear model. We feed the functions with the true values (test set) and the predicted values.



```
Mean squared error = 37784662.47
Explain variance score = 0.95
R2 score = 0.94
```

Image Prepared by the Author

If we perform a root square calculation on our mean squared error, we will gain an average discrepancy of around $6146.92 which is quite a low error. Besides, the explain variance score and the R2 score hit 0.9 above. This shows our linear model is not overfitted and can work nicely to predict the salary based on new data.

Conclusions

grown with our years of working experience and there is a linear relationship between them. We can use our linear model to predict the salary by giving input of years of experience.

Although this is a little too naive to presume the salary is only dependent on the years of working experience, this is one of the great examples to demonstrate how we can develop a simple linear regression model to show a relationship between two variables. In fact, most phenomena in real life cannot be easily explained by a linear model. However, the understanding of building a linear model is the foundation to build a complex model.

I hope you enjoyed reading this article.

If you like my article and would like to read more similar articles, feel free to subscribe to Medium. You will be able to read unlimited articles from me and other authors on Medium. Thanks for your support.

References

1. http://www.stat.yale.edu/Courses/1997-98/101/linreg.htm
2. https://machinelearningmastery.com/regression-metrics-for-machine-learning/
3. https://www.statisticshowto.com/explained-variance-variation/
4. https://en.wikipedia.org/wiki/Coefficient_of_determination