

Supervised Machine Learning Algorithms



www.educba.com



ECOM SCHOOL

המכללה למקצועות הדיגיטל וההייטק

Last lecture reminder

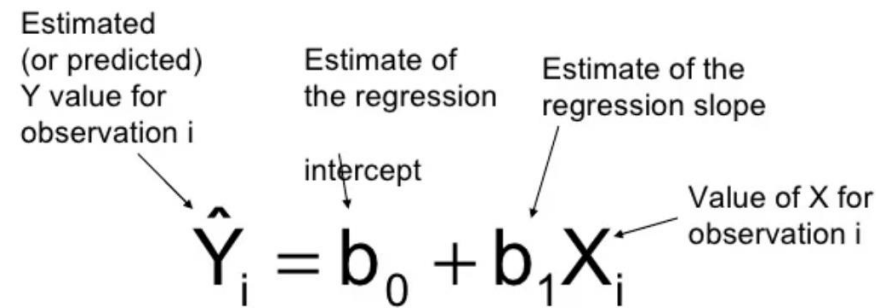


We learned about:

- Introduction to machine learning
- Introduction to supervised learning
- The supervised learning model process
- Introduction to linear regression
- Simple linear regression equation
- Multiple linear regression equation
- When can we use linear regression modeling

Solving Simple Linear Regression

As we learned a simple linear regression equation (single feature) looks as following:



Estimated (or predicted) Y value for observation i

Estimate of the regression intercept

Estimate of the regression slope

Value of X for observation i

$$\hat{Y}_i = b_0 + b_1 X_i$$

Solving a simple linear regression involves finding the best-fit line for the data by estimating the coefficients (b_0 Y-intercept and b_1 slope) that minimize the sum of the squared residuals.

So what we want to minimize is the following equation (for m observations):

$$\sum_{j=1}^m (y^j - \hat{y}^j)^2$$

Solving Simple Linear Regression

We can now put $\beta_0 + \beta_1 x$ instead of \hat{y} and we will get the following equation:

$$f(b_0, b_1) = \sum_{i=1}^n [y_i - (b_0 + b_1 x_i)]^2$$

Now in order to find the minimum of this function we will use derivatives for both β_0 and β_1 and equating them to 0.

Finding the β_0 and β_1 that are solving this equation will provide the values for β_0 and β_1 that minimize the square root of the error ($y - \hat{y}$).

$$\frac{\partial f(b_0, b_1)}{\partial b_0} = \sum 2(y_i - b_0 - b_1 x_i) (-1) = 0$$

$$\frac{\partial f(b_0, b_1)}{\partial b_1} = \sum 2(y_i - b_0 - b_1 x_i) (-x_i) = 0$$

Solving Simple Linear Regression

We got 2 equations with 2 unknowns (β_0 and β_1).

Now, after some mathematics manipulations we will get the following equations:

$$b_1 = \hat{\beta}_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

$$b_0 = \hat{\beta}_0 = \frac{\sum y_i - \hat{\beta}_1 \sum x_i}{n} = \bar{y} - \hat{\beta}_1 \bar{x}$$

$\bar{x} \rightarrow x$ mean ($\sum x_i / n$)

$\bar{y} \rightarrow y$ mean ($\sum y_i / n$)

Solving Simple Linear Regression - Example

Now that we have simple formulas for solving a simple linear regression let's take an example to show how it works.

A manager wants to find the relationship between the number of hours that a machine is operational in a week and weekly production. So in our example:

x → The hours of operation in a week (independent variable)

y → The weekly production volume (dependant variable)

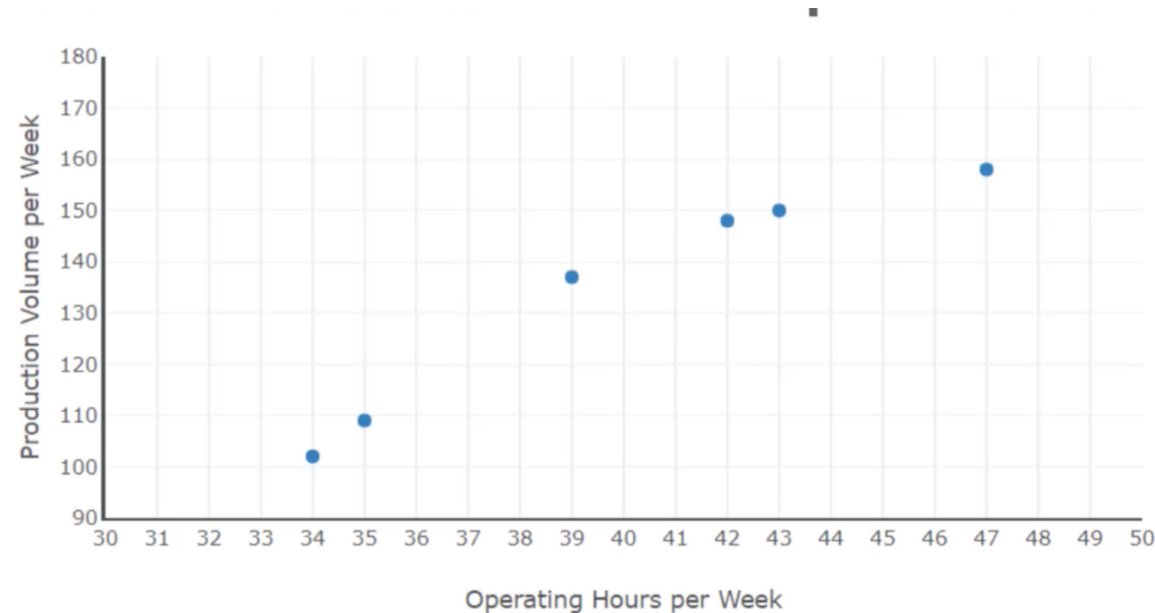
The manager provide the following historical data:

Production Hours (x)	Production Volume (y)
34	102
35	109
39	137
42	148
43	150
47	158



Solving Simple Linear Regression - Example

If we will plot the data to see the correlation between the production hours and the production volume according to the observations we will see that there is a linear pattern between them.



Solving Simple Linear Regression - Example

So now we will try to formulate according to simple linear regression a best fit line for this data.

We will use the formulas we found for \hat{y} , β_0 , β_1

$$\begin{aligned}\hat{y} &= b_0 + b_1x \\ b_1 &= \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2} \\ b_0 &= \bar{y} - b_1\bar{x}\end{aligned}$$

	Production Hours (x)	Production Volume (y)	$(x - \bar{x})$	$(y - \bar{y})$	$(x - \bar{x})(y - \bar{y})$	$(x - \bar{x})^2$
	34	102	-6	-32	192	36
	35	109	-5	-25	125	25
	39	137	-1	3	-3	1
	42	148	2	14	28	4
	43	150	3	16	48	9
	47	158	7	24	168	49
\bar{x}, \bar{y}	40	134		Sum:	558	124

Solving Simple Linear Regression - Example

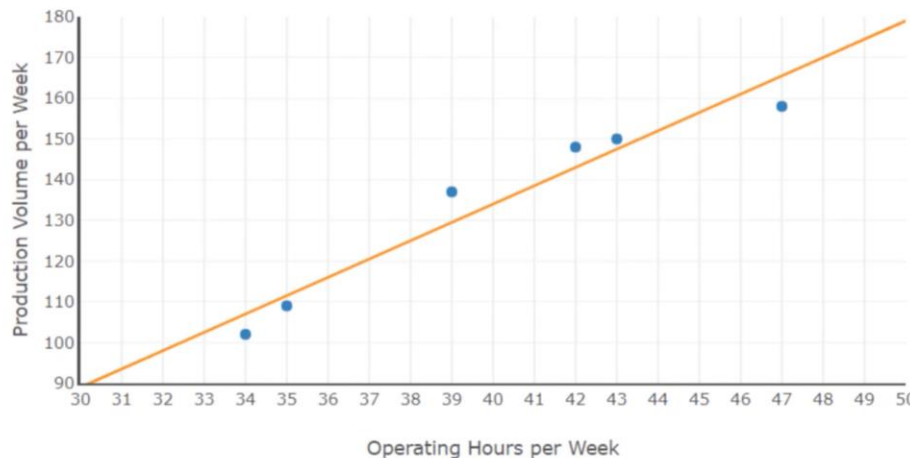
Let's now find the values for \hat{y} , β_0 , β_1 :

$$b_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2} = \frac{558}{124} = 4.5$$

$$b_0 = \bar{y} - b_1\bar{x} = 134 - (4.5 \times 40) = -46$$

$$\hat{y} = -46 + 4.5x$$

So now we have the equation of the best fit line according to the provided observations:



Solving Simple Linear Regression - Example

With this equation we can predict how many production volume will be for a given production hours or how many production hours we will need to get a specific production volume.

For example, if the manager wants to produce 125 units per week (production volume = 125) the machine should run for:

$$\hat{y} = b_0 + b_1x$$

$$125 = -46 + 4.5x$$

$$x = \frac{171}{4.5} = \mathbf{38 \text{ hours per week}}$$



Class Exercise - Simple Linear Regression

Instructions:

Given the following simple data set that shows the number of hours spent calling customers and the associated money earned:

Hours of calling customers: [2, 3, 4, 5, 6]

Money earned: [50K\$, 70K\$, 90K\$, 100K\$, 110K\$]

- Decide which is the independent variable and which is the dependant variable (x, y)
- Plot the dataset to see if there is a linear relationship between the variables
- Perform simple linear regression modeling and find the best fit line equation (\hat{y})
- Find how much hours should employee spend on calling customers in order to earn 150K\$

Class Exercise Solution - Simple Linear Regression

Solving Multiple Linear Regression

When dealing with multiple linear regression (multiple features) we still want to minimize the sum of squared errors:

$$\sum_{j=1}^m \left(y^j - \hat{y}^j \right)^2$$

but now the \hat{y} value is much more complex:

$$\hat{y} = \beta_0 x_0 + \dots + \beta_n x_n$$

Gradient Descent → Gradient descent is an optimization algorithm that's used to minimize some function by iteratively moving in the direction of steepest descent. In the case of linear regression, we're trying to minimize the cost function (mean squared error), and we use gradient descent to find the combination of parameters ($\beta_0, \beta_1, \beta_2, \dots, \beta_n$) that gives the lowest value for the cost function.



Solving Linear Regression - Python Example

In this example we will use the “Advertising.csv” file:

```
In [4]: df = pd.read_csv('/Users/ben.meir/Downloads/UNZIP_FOR_NOTEBOOKS_FINAL/DATA/Advertising.csv')
df
```

Out[4]:

	TV	radio	newspaper	sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9
...
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	9.7
197	177.0	9.3	6.4	12.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	13.4

200 rows x 4 columns

This csv contain pass data about campaign expenses in TV, Radio and Newspaper and what were the actual sales for each campaign.

We will want to explore if there is a linear dependency between the expenses amount and the sales amount.

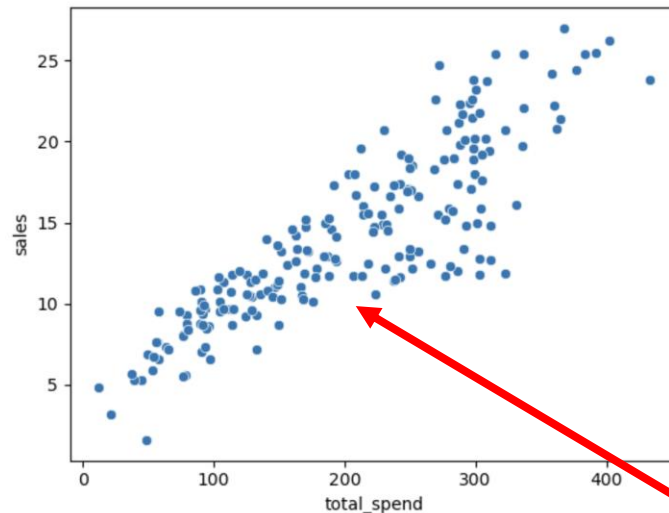


Solving Linear Regression - Python Example

First, let's perform a scatter plot that will allow us to better understand the dependency between the amount of marketing expenses and the sales this campaign got.

For that we will combine all the expenses in each campaign into one 'total_spend' column and create a scatter plot between the 'total_spend' column and the 'sales' column.

```
In [11]: sns.scatterplot(data=df_copy, x='total_spend', y='sales')
plt.show()
```



```
In [8]: df_copy = df.copy()
df_copy['total_spend'] = df.drop('sales', axis=1).sum(axis=1)
df_copy.head(5)
```

Out[8]:

	TV	radio	newspaper	sales	total_spend
0	230.1	37.8	69.2	22.1	337.1
1	44.5	39.3	45.1	10.4	128.9
2	17.2	45.9	69.3	9.3	132.4
3	151.5	41.3	58.5	18.5	251.3
4	180.8	10.8	58.4	12.9	250.0

We can see that increasing the total spend increase the sales amount linearly

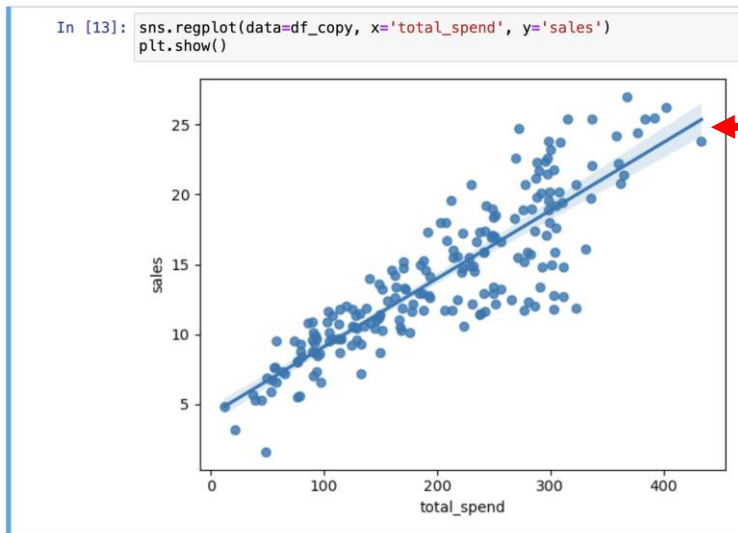


ECOM SCHOOL

המכללה למקצועות הדיגיטל וההייטק

Solving Linear Regression - Python Example

sns.regplot() → Allow us to get a scatter plot with the linear regression line, this can be useful when we want to visualize this line across the scatter plot.



The linear regression line

Now that we validated that there is a linear dependency between the 'total_spend' and 'sales' we can perform basic linear regression calculation to get the linear regression line formula.



Solving Linear Regression - Python Example

np.polyfit() → Allow us to find the β values of the linear regression line according to x (feature), y (label) and a specified polynomial degree.

For degree = 1 we will get array with the β_0 and β_1 values, for polynomial of larger degree we will get an array with more β values.

```
In [16]: x = df_copy['total_spend']
         y = df_copy['sales']

         np.polyfit(x, y, deg=1)

Out[16]: array([0.04868788, 4.24302822])
```

The first value in the array is β_1
and the second is β_0

Now that we calculated the β_0 and β_1 values of the linear regression line we can find how much sales we expect to get on any given total campaign expenses.

```
In [20]: betas = np.polyfit(x, y, deg=1)
         expenses = 200
         predicted_sales = expenses*betas[0] + betas[1]
         print(predicted_sales)

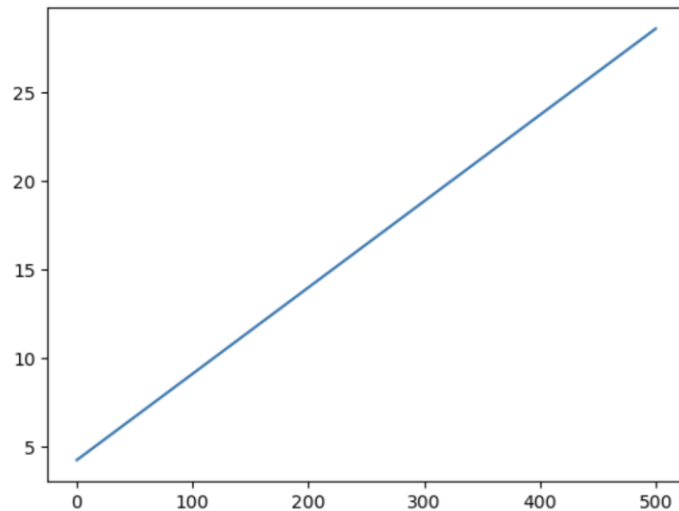
13.98060407984596
```

If we be 200\$ our predicted sales will be 13.98\$.

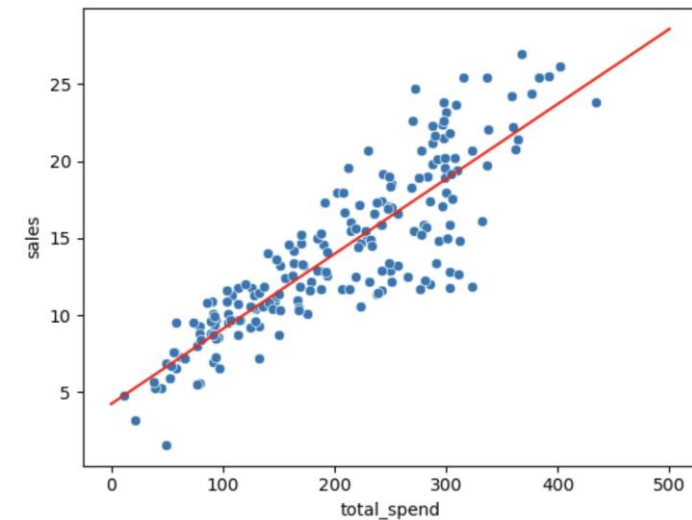
Solving Linear Regression - Python Example

We can also use `plt.plot()` to plot the real linear regression line and combine it with the scatter plot from earlier:

```
In [21]: potential_spends = np.linspace(0, 500, 100)  
predicted_sales = betas[0] * potential_spends + betas[1]  
plt.plot(potential_spends, predicted_sales)  
plt.show()
```



```
In [25]: sns.scatterplot(data=df_copy, x='total_spend', y='sales')  
plt.plot(potential_spends, predicted_sales, color='red')  
plt.show()
```



Polynomial Regression

In linear regression, sometimes a simple linear model (degree 1 polynomial) ($y = \beta_0 + \beta_1 * x$) does not always provide a satisfactory fit to the data. This might be because the underlying relationship between the variables isn't just linear, but more complex.

Polynomial regression → To solve use cases like this we can use polynomial regression, which involves fitting a polynomial of degree higher than 1, allows us to capture these more complex relationships.

For example → with a 2nd degree polynomial ($y = \beta_0 + \beta_0 + \beta_1 * x + \beta_2 * (x^2)$), the relationship between x and y is modeled as an upward or downward curve, rather than a straight line.

Polynomial Regression

In polynomial regression, the degree of the polynomial chosen determines the complexity of the model.

While we can select as high a degree as desired, beyond a certain threshold, the higher degrees may no longer provide meaningful or valuable information about the data.

One way to assess the relevance of the degree is through the beta or coefficient values. A beta value near 1 indicates a strong linear relationship between the predictor and the response variable, implying that the predictor is significant in understanding the response. On the other hand, a beta value close to 0 signifies a weak or negligible dependency, suggesting that the predictor may not be as useful in predicting the outcome variable.

Polynomial Regression

In our example we can check for higher polynomial degree and see the beta values that we are getting.

For example → Let's check for 3 degree polynomial ($y = \beta_0 + \beta_1*x + \beta_2*(x^2) + \beta_3*(x^3)$)

```
In [28]: x = df_copy['total_spend']  
         y = df_copy['sales']  
         np.polyfit(x, y, deg=3)
```

```
Out[28]: array([ 3.07615033e-07, -1.89392449e-04,  8.20886302e-02,  2.70495053e+00])
```

We got very small beta values for the higher degree polynomial indicating that there is no strong correlation between the predictor and the outcome, so we can ignore them and not count them in our linear regression calculation.

Class Exercise - Linear Regression In Python

Instructions:

For this exercise use the '**tips.csv**' file.

Explore the data and analyze the relationship between "total_bill" and "tip".

1. Create a scatter plot that show if there is a linear relationship between them.
2. In case you found that there is a linear relationship calculate using Python the linear regression line and find the beta coefficients.
3. Generate the linear regression line on the scatter plot from the previous exercise
4. Calculate what is the predicted tip for total bill of 150\$.
5. Find the beta coefficient of a 3 degree polynomial and determine if those coefficients are meaningful or should be ignored.

Class Exercise Solution - Linear Regression In Python

