Regression Analysis

Avijit Bose
Assistant Professor
Department of Computer Science& Engineering, MCKV Institute of Engineering,Liluah,Howrah-711204

What is Regression?

- Regression searches for relationship among variables.
- For example, we can observe several employees of some company and try to understand how their salaries depend on the **features**, such as experience, level of education, role, city they work in, and so on.

Regression

- This is a regression problem where data related to each employee represent one **observation**. The presumption is that the experience, education, role, and city are the independent features, while the salary depends on them
- Generally, in regression analysis, you usually consider some phenomenon of interest and have a number of observations. Each observation has two or more features. Following the assumption that (at least) one of the features depends on the others, you try to establish a relation among them.

Regression

- In other words, **you need to find a function that maps some features or variables to others sufficiently well**.
- The dependent features are called the **dependent variables**, **outputs**, or **responses**.
- The independent features are called the **independent variables**, **inputs**, or **predictors**.

Regression

- Regression problems usually have one continuous and unbounded dependent variable. The inputs, however, can be continuous, discrete, or even categorical data such as gender, nationality, brand, and so on.
- It is a common practice to denote the outputs with y and inputs with x. If there are two or more independent variables, they can be represented as the vector $\mathbf{x} = (x_1, \ldots, x_r)$, where r is the number of inputs.

Regression When is Required?

- Typically, you need regression to answer whether and how some phenomenon influences the other or **how several variables are related**. For example, you can use it to determine *if* and *to what extent* the experience or gender impact salaries.
- Regression is also useful when you want **to forecast a response** using a new set of predictors. For example, you could try to predict electricity consumption of a household for the next hour given the outdoor temperature, time of day, and number of residents in that household.
- Regression is used in many different fields: economy, computer science, social sciences, and so on. Its importance rises every day with the availability of large amounts of data and increased awareness of the practical value of data.

Regression-Linear

- Linear regression is probably one of the most important and widely used regression techniques. It's among the simplest regression methods. One of its main advantages is the ease of interpreting results.

Regression

- When implementing linear regression of some dependent variable y on the set of independent variables $\mathbf{x} = (x_1, \ldots, x_r)$, where r is the number of predictors, you assume a linear relationship between y and $\mathbf{x}$: $y = \beta_0 + \beta_1 x_1 + \cdots + \beta_r x_r + \varepsilon$. This equation is the **regression equation**. $\beta_0, \beta_1, \ldots, \beta_r$ are the **regression coefficients**, and ε is the **random error**.

Regression

- Linear regression calculates the **estimators** of the regression coefficients or simply the **predicted weights**, denoted with b_0, b_1, …, b_r. They define the **estimated regression function** $f(\mathbf{x}) = b_0 + b_1x_1 + \cdots + b_rx_r$. This function should capture the dependencies between the inputs and output sufficiently well.

Regression

- The **estimated** or **predicted response**, $f(\mathbf{x}_i)$, for each observation $i = 1, \ldots, n$, should be as close as possible to the corresponding **actual response** y_i. The differences $y_i - f(\mathbf{x}_i)$ for all observations $i = 1, \ldots, n$, are called the **residuals**. Regression is about determining the **best predicted weights**, that is the weights corresponding to the smallest residuals.

Regression

- To get the best weights, you usually **minimize the sum of squared residuals** (SSR) for all observations $i = 1, \ldots, n$: SSR = $\Sigma_i(y_i - f(\mathbf{x}_i))^2$. This approach is called the **method of ordinary least squares**.

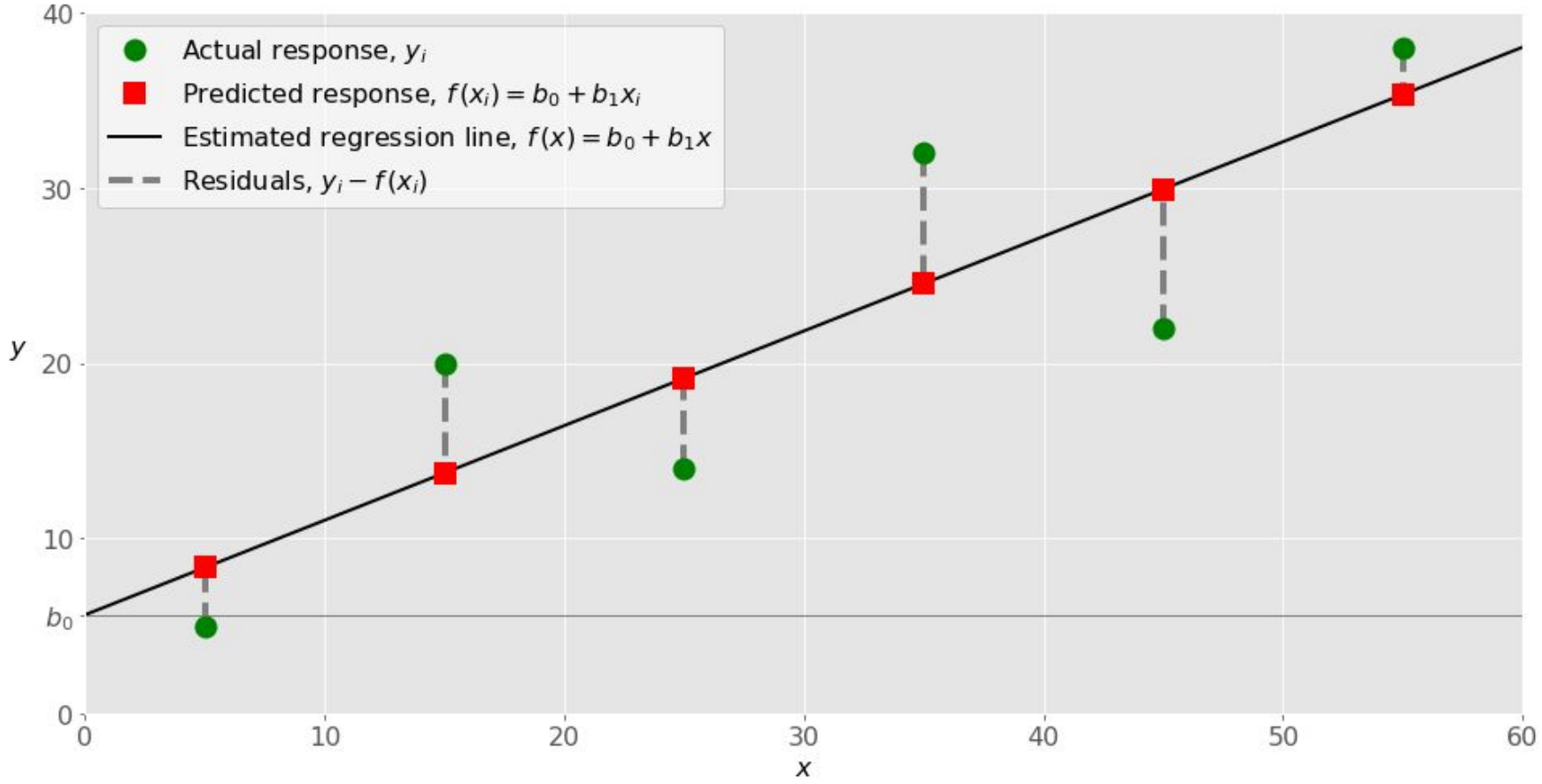
Regression Performance

- The variation of actual responses y_i, $i = 1, \ldots, n$, occurs partly due to the dependence on the predictors $\mathbf{x}_i$. However, there is also an additional inherent variance of the output.
- The **coefficient of determination**, denoted as R^2, tells you which amount of variation in y can be explained by the dependence on $\mathbf{x}$ using the particular regression model. Larger R^2 indicates a better fit and means that the model can better explain the variation of the output with different inputs.
- The value $R^2 = 1$ corresponds to SSR = 0, that is to the **perfect fit** since the values of predicted and actual responses fit completely to each other.

Regression(Simple Linear)

- Simple or single-variate linear regression is the simplest case of linear regression with a single independent variable, $\mathbf{x} = x$.

Regression


Actual response, y_i
Predicted response, $f(x_i) = b_0 + b_1 x_i$
Estimated regression line, $f(x) = b_0 + b_1 x$
Residuals, $y_i - f(x_i)$
y
x
b_0
0
10
20
30
40
50
60

Regression

- When implementing simple linear regression, you typically start with a given set of input-output (x-y) pairs (green circles). These pairs are your observations. For example, the leftmost observation (green circle) has the input x = 5 and the actual output (response) y = 5. The next one has x = 15 and y = 20, and so on.

Regression

- The estimated regression function (black line) has the equation $f(x) = b_0 + b_1x$. Your goal is to calculate the optimal values of the predicted weights b_0 and b_1 that minimize SSR and determine the estimated regression function. The value of b_0, also called the **intercept**, shows the point where the estimated regression line crosses the y axis. It is the value of the estimated response $f(x)$ for $x = 0$. The value of b_1 determines the **slope** of the estimated regression line.

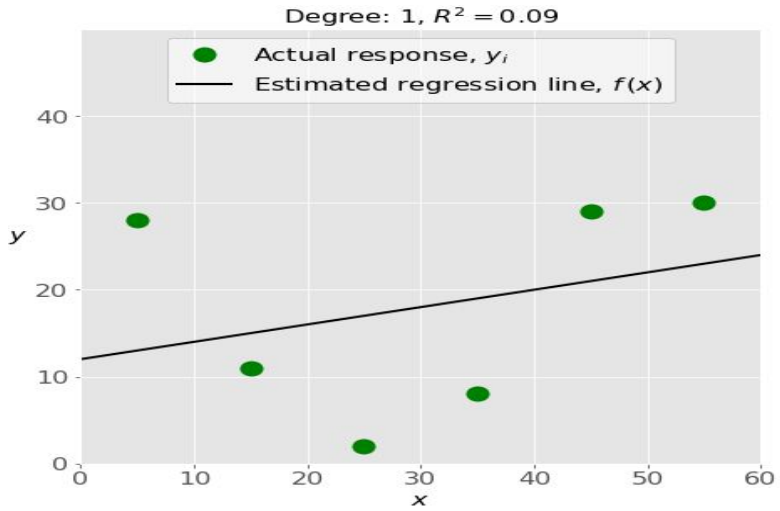
Regression

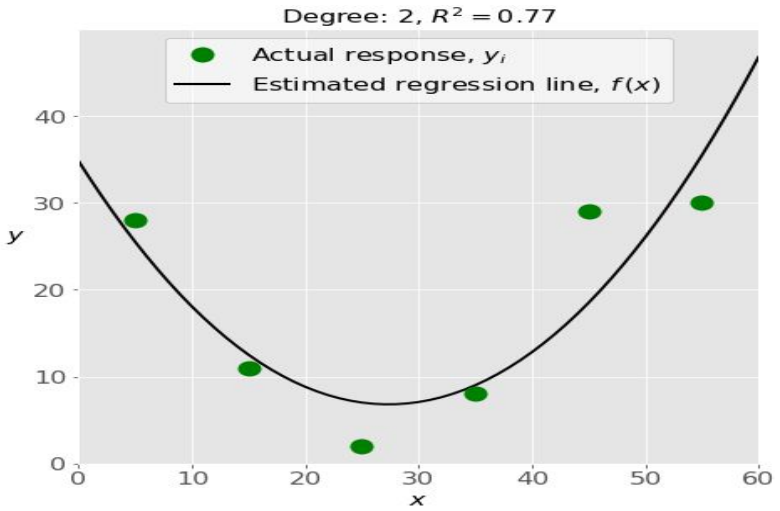
- The predicted responses (red squares) are the points on the regression line that correspond to the input values. For example, for the input x = 5, the predicted response is $f(5)$ = 8.33 (represented with the leftmost red square).
- The residuals (vertical dashed gray lines) can be calculated as $y_i - f(\mathbf{x}_i) = y_i - b_0 - b_1 x_i$ for $i = 1, \ldots, n$. They are the distances between the green circles and red squares. When you implement linear regression, you are actually trying to minimize these distances and make the red squares as close to the predefined green circles as possible.

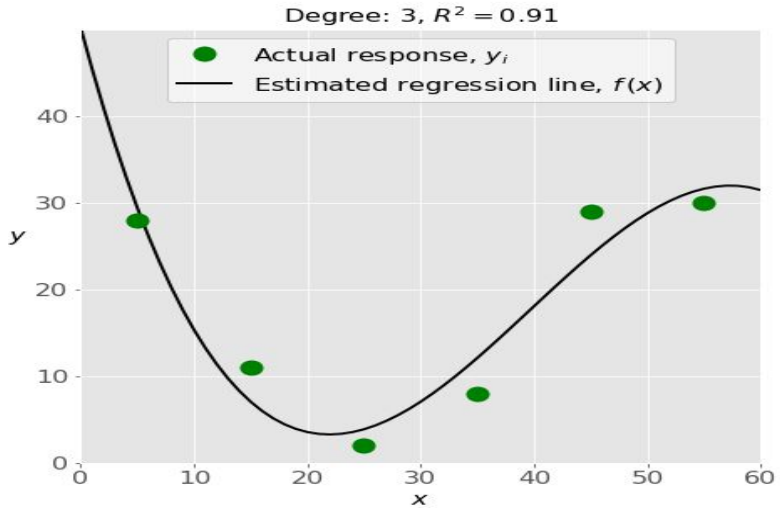
Fitting Values

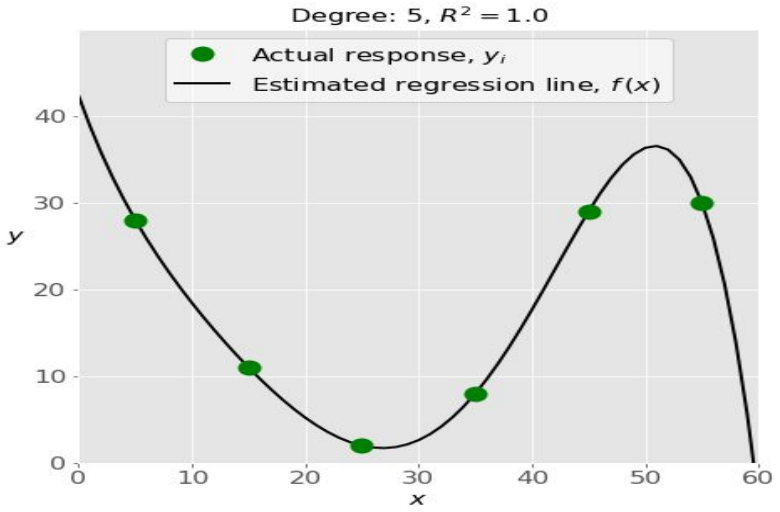
- **Underfitting** occurs when a model can't accurately capture the dependencies among data, usually as a consequence of its own simplicity. It often yields a low R^2 with known data and bad generalization capabilities when applied with new data.
- **Overfitting** happens when a model learns both dependencies among data and random fluctuations. In other words, a model learns the existing data too well. Complex models, which have many features or terms, are often prone to overfitting. When applied to known data, such models usually yield high R^2. However, they often don't generalize well and have significantly lower R^2 when used with new data.

Fitting Values









Fitting Data

- The top left plot shows a linear regression line that has a low R^2. It might also be important that a straight line can't take into account the fact that the actual response increases as x moves away from 25 towards zero. This is likely an example of under fitting.
- The top right plot illustrates polynomial regression with the degree equal to 2. In this instance, this might be the optimal degree for modeling this data. The model has a value of R^2 that is satisfactory in many cases and shows trends nicely.

Fitting Data

- The bottom left plot presents polynomial regression with the degree equal to 3. The value of R^2 is higher than in the preceding cases. This model behaves better with known data than the previous ones. However, it shows some signs of overfitting, especially for the input values close to 60 where the line starts decreasing, although actual data don’t show that.

Fitting Data

- Finally, on the bottom right plot, you can see the perfect fit: six points and the polynomial line of the degree 5 (or higher) yield $R^2 = 1$. Each actual response equals its corresponding prediction.
- In some situations, this might be exactly what you're looking for. In many cases, however, this is an overfitted model. It is likely to have poor behavior with unseen data, especially with the inputs larger than 50.
- For example, it assumes, without any evidence, that there is a significant drop in responses for $x > 50$ and that y reaches zero for x near 60. Such behavior is the consequence of excessive effort to learn and fit the existing data.

Steps

1. The first step is to import the package numpy and the class LinearRegression from sklearn.linear_model :
2. import numpy as np from sklearn.linear_model import LinearRegression
3. **Provide data**

x = np.array([5, 15, 25, 35, 45, 55]).reshape((-1, 1)) y = np.array([5, 20, 14, 32, 22, 38])

Steps

- >>> print(x)
- [[5]
- [15]
- [25]
- [35]
- [45]
- [55]]
- >>> print(y)
- [5 20 14 32 22 38]

Steps

- 3. Create a model and fit in
- The next step is to create a linear regression model and fit it using the existing data.
- Let’s create an instance of the class LinearRegression, which will represent the regression model:
- model = LinearRegression()

Steps

- This statement creates the variable model as the instance of LinearRegression. You can provide several optional parameters to LinearRegression:
- **fit_intercept** is a Boolean (True by default) that decides whether to calculate the intercept b_0 (True) or consider it equal to zero (False).
- **normalize** is a Boolean (False by default) that decides whether to normalize the input variables (True) or not (False).
- **copy_X** is a Boolean (True by default) that decides whether to copy (True) or overwrite the input variables (False).
- **n_jobs** is an integer or None (default) and represents the number of jobs used in parallel computation. None usually means one job and -1 to use all processors.
- This example uses the default values of all parameters.
- It's time to start using the model. First, you need to call .fit() on model:

Steps

- model.fit(x, y)
- With .fit(), you calculate the optimal values of the weights b_0 and b_1, using the existing input and output (x and y) as the arguments. In other words, .fit() **fits the model**. It returns self, which is the variable model itself. That's why you can replace the last two statements with this one:
- model = LinearRegression().fit(x, y)

Steps

- Once you have your model fitted, you can get the results to check whether the model works satisfactorily and interpret it.
- You can obtain the coefficient of determination (R^2) with .score() called on model:
- >>> r_sq = model.score(x, y)
- >>> print('coefficient of determination:', r_sq) coefficient of determination: 0.715875613747954

Steps

- When you're applying .score(), the arguments are also the predictor x and regressor y, and the return value is R^2.
- The attributes of model are .intercept_, which represents the coefficient, b_0 and .coef_, which represents b_1:
- >>> print('intercept:', model.intercept_) intercept: 5.633333333333329
- >>> print('slope:', model.coef_)
- slope: [0.54]

Steps

- The value b_0 = 5.63 (approximately) illustrates that your model predicts the response 5.63 when x is zero. The value b_1 = 0.54 means that the predicted response rises by 0.54 when x is increased by one.
- Step-5 Predict Response
- Once there is a satisfactory model, you can use it for predictions with either existing or new data.
- To obtain the predicted response, use .predict():

Steps

- >>> y_pred = model.predict(x)
- >>> print('predicted response:', y_pred, sep='\n')
- predicted response:
- [8.33333333 13.73333333 19.13333333 24.53333333 29.93333333 35.33333333]
- >>> y_pred = model.intercept_ + model.coef_ * x >>> print('predicted response:', y_pred, sep='\n') predicted response:
- [[8.33333333]
- [13.73333333]
- [19.13333333]
- [24.53333333]
- [29.93333333]
- [35.33333333]]

Multiple Linear Regression

- Multiple or multivariate linear regression is a case of linear regression with two or more independent variables.
- If there are just two independent variables, the estimated regression function is $f(x_1, x_2) = b_0 + b_1x_1 + b_2x_2$. It represents a regression plane in a three-dimensional space. The goal of regression is to determine the values of the weights b_0, b_1, and b_2 such that this plane is as close as possible to the actual responses and yield the minimal SSR.
- The case of more than two independent variables is similar, but more general. The estimated regression function is $f(x_1, \ldots, x_r) = b_0 + b_1x_1 + \cdots + b_rx_r$, and there are $r + 1$ weights to be determined when the number of inputs is r.

Multiple Linear Regression

- **Steps 1 and 2: Import packages and classes, and provide data**

import numpy and sklearn.linear_model.LinearRegression

and provide known inputs and output:

```
import numpy as np
from sklearn.linear_model import LinearRegression
x = [[0, 1], [5, 1], [15, 2], [25, 5], [35, 11], [45, 15], [55, 34], [60, 35]]
y = [4, 5, 20, 14, 32, 22, 38, 43]
x, y = np.array(x), np.array(y)
```

Multiple Linear Regression

- In multiple linear regression, x is a two-dimensional array with at least two columns, while y is usually a one-dimensional array. This is a simple example of multiple linear regression, and x has exactly two columns.

Step-3 Create a model and fit it

The next step is to create the regression model as an instance of Linear Regression and fit it with .fit():

model = Linear Regression().fit(x, y)

Multiple Linear Regression

- **Step 4: Get results**
- You can obtain the properties of the model the same way as in the case of simple linear regression:
- >>> r_sq = model.score(x, y)
- >>> print('coefficient of determination:', r_sq) coefficient of determination: 0.861593925875677 6
- >>> print('intercept:', model.intercept_) intercept: 5.52257927519819
- >>> print('slope:', model.coef_)
- slope: [0.44706965 0.25502548]

Multiple Linear Regression

- You obtain the value of R^2 using .score() and the values of the estimators of regression coefficients with .intercept_ and .coef_. Again, .intercept_ holds the bias b_0, while now .coef_ is an array containing b_1 and b_2 respectively.

Step-5 predict response

```
>>> y_pred = model.predict(x)
>>> print('predicted response:', y_pred, sep='\n')
predicted response: [ 5.77760476 8.012953 12.73867497
17.9744479 23.97529728 29.4660957 38.78227633
41.27265006]
```

Multiple linear regression

- >>> y_pred = model.intercept_ + np.sum(model.coef_ * x, axis=1)
- >>> print('predicted response:', y_pred, sep='\n')
- predicted response: [5.77760476 8.012953 12.73867497 17.9744479 23.97529728 29.4660957 38.78227633 41.27265006]

Logistic Regression

- As the amount of available data, the strength of computing power, and the number of algorithmic improvements continue to rise, so does the importance of data science and machine learning. **Classification** is among the most important areas of machine learning, and **logistic regression** is one of its basic methods.

Logistic Regression

- Classification is a very important area of supervised machine learning. A large number of important machine learning problems fall within this area. There are many classification methods, and logistic regression is one of them.

Logistic Regression

- Supervised machine learning algorithms define models that capture relationships among data. **Classification** is an area of supervised machine learning that tries to predict which class or category some entity belongs to, based on its features.
- For example, you might analyze the employees of some company and try to establish a dependence on the **features** or **variables**, such as the level of education, number of years in a current position, age, salary, odds for being promoted, and so on. The set of data related to a single employee is one **observation**.

Logistic Regression

- **Independent variables**, also called inputs or predictors, don't depend on other features of interest (or at least you assume so for the purpose of the analysis).
- **Dependent variables**, also called outputs or responses, depend on the independent variables.
- In the above example where you're analyzing employees, you might presume the level of education, time in a current position, and age as being mutually independent, and consider them as the inputs. The salary and the odds for promotion could be the outputs that depend on the inputs.

Logistic Regression

- The nature of the dependent variables differentiates regression and classification problems. **Regression** problems have continuous and usually unbounded outputs. An example is when you're estimating the salary as a function of experience and education level. On the other hand, **classification** problems have discrete and finite outputs called **classes** or **categories**. For example, predicting if an employee is going to be promoted or not (true or false) is a classification problem.

Logistic Regression

- There are two main types of classification problems:
- **Binary** or **binomial classification:** exactly two classes to choose between (usually 0 and 1, true and false, or positive and negative)
- **Multiclass** or **multinomial classification:** three or more classes of the outputs to choose from
- If there's only one input variable, then it's usually denoted with x. For more than one input, you'll commonly see the vector notation $\mathbf{x} = (x_1, \ldots, x_r)$, where r is the number of the predictors (or independent features). The output variable is often denoted with y and takes the values 0 or 1.