**Regression**

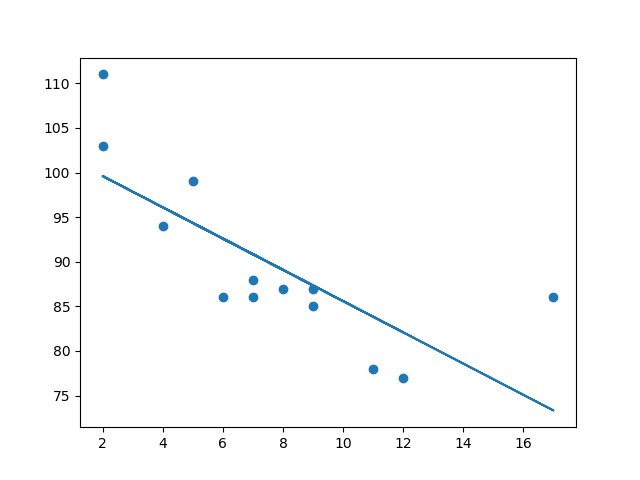
The term regression is used when you try to find the relationship between variables.

In Machine Learning, and in statistical modeling, that relationship is used to predict the outcome of future events.

**Linear Regression**

Linear regression uses the relationship between the data-points to draw a straight line through all them.

This line can be used to predict future values.



In Machine Learning, predicting the future is very important.

**Working of Linear Regression**

Python has methods for finding a relationship between data-points and to draw a line of linear regression. We will show you how to use these methods instead of going through the mathematic formula.

In the example below, the x-axis represents age, and the y-axis represents speed. We have registered the age and speed of 13 cars as they were passing a tollbooth. Let us see if the data we collected could be used in a linear regression:

import matplotlib.pyplot as plt  
  
x = [5,7,8,7,2,17,2,9,4,11,12,9,6]  
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]  
  
plt.scatter(x, y)  
plt.show()

Result:

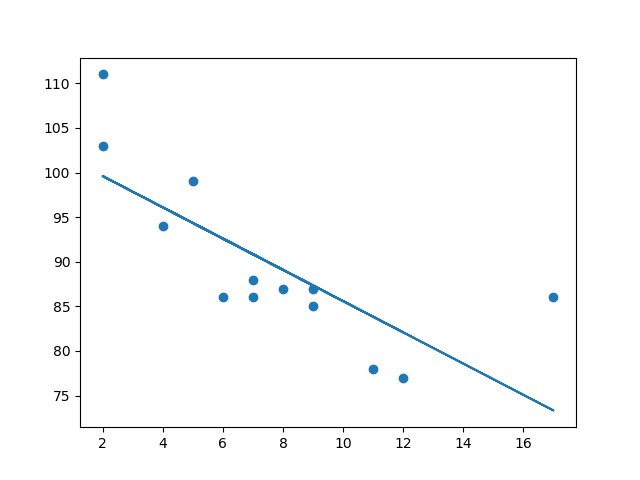


Example

Import scipy and draw the line of Linear Regression:

import matplotlib.pyplot as plt  
from scipy import stats  
  
x = [5,7,8,7,2,17,2,9,4,11,12,9,6]  
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]  
  
slope, intercept, r, p, std\_err = stats.linregress(x, y)  
  
def myfunc(x):  
  return slope \* x + intercept  
  
mymodel = list(map(myfunc, x))  
  
plt.scatter(x, y)  
plt.plot(x, mymodel)  
plt.show()

Result:



**Example Explained**

Import the modules you need.

import matplotlib.pyplot as plt  
from scipy import stats

Create the arrays that represent the values of the x and y axis:

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]  
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

Execute a method that returns some important key values of Linear Regression:

slope, intercept, r, p, std\_err = stats.linregress(x, y)

Create a function that uses the slope and intercept values to return a new value. This new value represents where on the y-axis the corresponding x value will be placed:

def myfunc(x):  
  return slope \* x + intercept

Run each value of the x array through the function. This will result in a new array with new values for the y-axis:

mymodel = list(map(myfunc, x))

Draw the original scatter plot:

plt.scatter(x, y)

Draw the line of linear regression:

plt.plot(x, mymodel)

Display the diagram:

plt.show()

**R for Relationship**

It is important to know how the relationship between the values of the x-axis and the values of the y-axis is, if there are no relationship the linear regression can not be used to predict anything.

This relationship - the coefficient of correlation - is called r.

The r value ranges from -1 to 1, where 0 means no relationship, and 1 (and -1) means 100% related.

Python and the Scipy module will compute this value for you, all you have to do is feed it with the x and y values.

Example

How well does my data fit in a linear regression?

from scipy import stats  
  
x = [5,7,8,7,2,17,2,9,4,11,12,9,6]  
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]  
  
slope, intercept, r, p, std\_err = stats.linregress(x, y)  
  
print(r)

**Note:** The result -0.76 shows that there is a relationship, not perfect, but it indicates that we could use linear regression in future predictions.

**Predict Future Values**

Now we can use the information we have gathered to predict future values.

Example: Let us try to predict the speed of a 10 years old car.

To do so, we need the same myfunc() function from the example above:

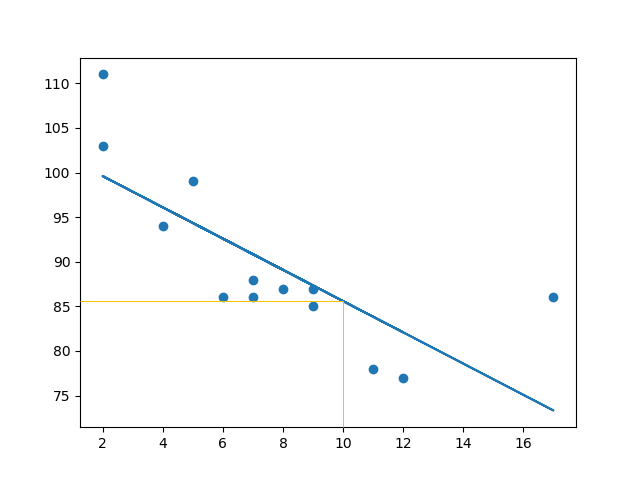
def myfunc(x):  
  return slope \* x + intercept

Example

Predict the speed of a 10 years old car:

from scipy import stats  
  
x = [5,7,8,7,2,17,2,9,4,11,12,9,6]  
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]  
  
slope, intercept, r, p, std\_err = stats.linregress(x, y)  
  
def myfunc(x):  
  return slope \* x + intercept  
  
speed = myfunc(10)  
  
print(speed)

The example predicted a speed at 85.6, which we also could read from the diagram:



**Bad Fit Line**

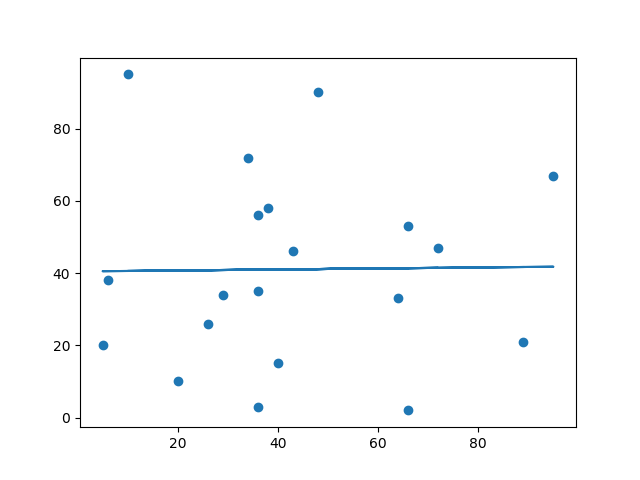
Let us create an example where linear regression would not be the best method to predict future values.

Example

These values for the x- and y-axis should result in a very bad fit for linear regression:

import matplotlib.pyplot as plt  
from scipy import stats  
  
x = [89,43,36,36,95,10,66,34,38,20,26,29,48,64,6,5,36,66,72,40]  
y = [21,46,3,35,67,95,53,72,58,10,26,34,90,33,38,20,56,2,47,15]  
  
slope, intercept, r, p, std\_err = stats.linregress(x, y)  
  
def myfunc(x):  
  return slope \* x + intercept  
  
mymodel = list(map(myfunc, x))  
  
plt.scatter(x, y)  
plt.plot(x, mymodel)  
plt.show()

Result:



And the r for relationship?

Example

You should get a very low r value.

import numpy  
from scipy import stats  
  
x = [89,43,36,36,95,10,66,34,38,20,26,29,48,64,6,5,36,66,72,40]  
y = [21,46,3,35,67,95,53,72,58,10,26,34,90,33,38,20,56,2,47,15]  
slope, intercept, r, p, std\_err = stats.linregress(x, y)  
print(r)

The result: 0.013 indicates a very bad relationship, and tells us that this data set is not suitable for linear regression.

**Multiple Regression**

Multiple regression is like linear regression, but with more than one independent value, meaning that we try to predict a value based on two or more variables.

Take a look at the data set below, it contains some information about cars.

| Car | Model | Volume | Weight | CO2 |
| --- | --- | --- | --- | --- |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Toyota | Aygo | 1000 | 790 | 99 |
| Mitsubishi | Space Star | 1200 | 1160 | 95 |
| Skoda | Citigo | 1000 | 929 | 95 |
| Fiat | 500 | 900 | 865 | 90 |
| Mini | Cooper | 1500 | 1140 | 105 |
| VW | Up! | 1000 | 929 | 105 |
| Skoda | Fabia | 1400 | 1109 | 90 |
| Mercedes | A-Class | 1500 | 1365 | 92 |
| Ford | Fiesta | 1500 | 1112 | 98 |
| Audi | A1 | 1600 | 1150 | 99 |
| Hyundai | I20 | 1100 | 980 | 99 |
| Suzuki | Swift | 1300 | 990 | 101 |
| Ford | Fiesta | 1000 | 1112 | 99 |
| Honda | Civic | 1600 | 1252 | 94 |
| Hundai | I30 | 1600 | 1326 | 97 |
| Opel | Astra | 1600 | 1330 | 97 |
| BMW | 1 | 1600 | 1365 | 99 |
| Mazda | 3 | 2200 | 1280 | 104 |
| Skoda | Rapid | 1600 | 1119 | 104 |
| Ford | Focus | 2000 | 1328 | 105 |
| Ford | Mondeo | 1600 | 1584 | 94 |
| Opel | Insignia | 2000 | 1428 | 99 |
| Mercedes | C-Class | 2100 | 1365 | 99 |
| Skoda | Octavia | 1600 | 1415 | 99 |
| Volvo | S60 | 2000 | 1415 | 99 |
| Mercedes | CLA | 1500 | 1465 | 102 |
| Audi | A4 | 2000 | 1490 | 104 |
| Audi | A6 | 2000 | 1725 | 114 |
| Volvo | V70 | 1600 | 1523 | 109 |
| BMW | 5 | 2000 | 1705 | 114 |
| Mercedes | E-Class | 2100 | 1605 | 115 |
| Volvo | XC70 | 2000 | 1746 | 117 |
| Ford | B-Max | 1600 | 1235 | 104 |
| BMW | 2 | 1600 | 1390 | 108 |
| Opel | Zafira | 1600 | 1405 | 109 |
| Mercedes | SLK | 2500 | 1395 | 120 |

We can predict the CO2 emission of a car based on the size of the engine, but with multiple regression we can throw in more variables, like the weight of the car, to make the prediction more accurate.

**Multiple Regression Working**

In Python we have modules that will do the work for us. Start by importing the Pandas module.

import pandas

The Pandas module allows us to read csv files and return a DataFrame object.

The file is meant for testing purposes only

df = pandas.read\_csv("data.csv")

Then make a list of the independent values and call this variable X.

Put the dependent values in a variable called y.

X = df[['Weight', 'Volume']]  
y = df['CO2']

We will use some methods from the sklearn module, so we will have to import that module as well:

from sklearn import linear\_model

From the sklearn module we will use the LinearRegression() method to create a linear regression object.

This object has a method called fit() that takes the independent and dependent values as parameters and fills the regression object with data that describes the relationship:

regr = linear\_model.LinearRegression()  
regr.fit(X, y)

Now we have a regression object that are ready to predict CO2 values based on a car's weight and volume:

#predict the CO2 emission of a car where the weight is 2300kg, and the volume is 1300cm3:  
predictedCO2 = regr.predict([[2300, 1300]])

Example

See the whole example in action:

import pandas  
from sklearn import linear\_model  
  
df = pandas.read\_csv("data.csv")  
  
X = df[['Weight', 'Volume']]  
y = df['CO2']  
  
regr = linear\_model.LinearRegression()  
regr.fit(X, y)  
  
#predict the CO2 emission of a car where the weight is 2300kg, and the volume is 1300cm3:  
predictedCO2 = regr.predict([[2300, 1300]])  
  
print(predictedCO2)

Result:

[107.2087328]

We have predicted that a car with 1.3 liter engine, and a weight of 2300 kg, will release approximately 107 grams of CO2 for every kilometer it drives.

**Coefficient**

The coefficient is a factor that describes the relationship with an unknown variable.

Example: if x is a variable, then 2x is x two times. x is the unknown variable, and the number 2 is the coefficient.

In this case, we can ask for the coefficient value of weight against CO2, and for volume against CO2. The answer(s) we get tells us what would happen if we increase, or decrease, one of the independent values.

**Example**

Print the coefficient values of the regression object:

import pandas  
from sklearn import linear\_model  
  
df = pandas.read\_csv("data.csv")  
  
X = df[['Weight', 'Volume']]  
y = df['CO2']  
  
regr = linear\_model.LinearRegression()  
regr.fit(X, y)  
  
print(regr.coef\_)

Result:

[0.00755095 0.00780526]

**Result Explained**

The result array represents the coefficient values of weight and volume.

Weight: 0.00755095  
Volume: 0.00780526

These values tell us that if the weight increase by 1kg, the CO2 emission increases by 0.00755095g.

And if the engine size (Volume) increases by 1 cm3, the CO2 emission increases by 0.00780526 g.

I think that is a fair guess, but let test it!

We have already predicted that if a car with a 1300cm3 engine weighs 2300kg, the CO2 emission will be approximately 107g.

What if we increase the weight with 1000kg?

Example

Copy the example from before, but change the weight from 2300 to 3300:

import pandas  
from sklearn import linear\_model  
  
df = pandas.read\_csv("data.csv")  
  
X = df[['Weight', 'Volume']]  
y = df['CO2']  
  
regr = linear\_model.LinearRegression()  
regr.fit(X, y)  
  
predictedCO2 = regr.predict([[3300, 1300]])  
  
print(predictedCO2)

Result:

[114.75968007]

We have predicted that a car with 1.3 liter engine, and a weight of 3300 kg, will release approximately 115 grams of CO2 for every kilometer it drives.

Which shows that the coefficient of 0.00755095 is correct:

107.2087328 + (1000 \* 0.00755095) = 114.75968