Chapter 6: Regression

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

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

1- 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.

For example: We have registered the age (x) and speed (y) of 13 cars as they were passing a radar.

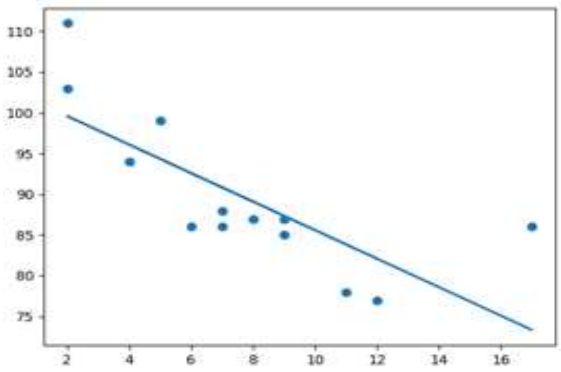
```
import matplotlib.pyplot as plt
from scipy import stats

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 getY(x):
  return slope * x + intercept

line = list(map(getY, x))

print(r)
plt.scatter(x, y)
plt.plot(x, line)
plt.show()
```



a- R for Relationship:

The relationship between x and y is called R (the coefficient of correlation). The R value ranges from -1 to 1, where 0 means no relationship, and 1 (and -1) means 100% related.

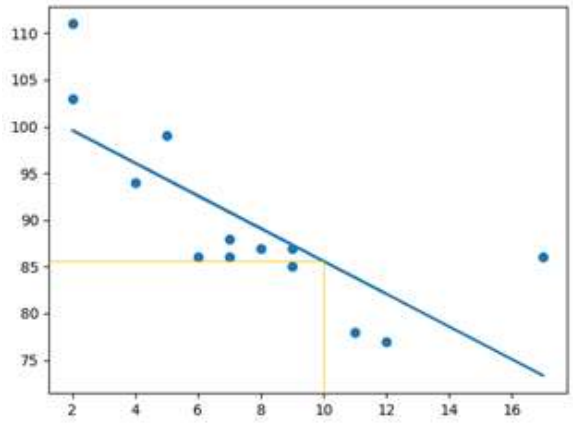
Note: The result -0.76 shows that there is a relationship, not perfect, but it indicates that we could use linear regression in future predictions.

b- Predict Future Values:

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

Example: Try to 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 getSpeed(x):
  return slope * x + intercept
print(getSpeed(10))
```

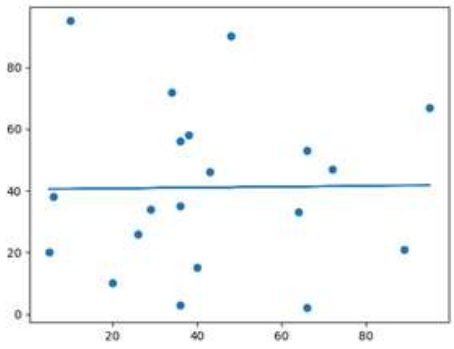


The example predict a speed of 85.6, which we also could read from the diagram

c- Bad Fit?

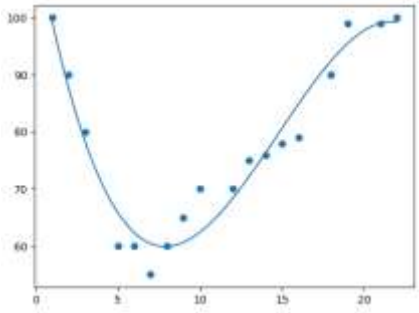
Let us create an example where linear regression would not be the best method to predict future values.

```
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 getSpeed(x):
  return slope * x + intercept
Line = list(map(getSpeed, x))
plt.scatter(x, y)
plt.plot(x, Line)
plt.show()
print(r)
```



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

2- Polynomial Regression



- If your data points clearly will not fit a straight linear regression, it might be ideal for polynomial regression.

- Polynomial regression, like linear regression, uses the relationship between the variables x and y to find the best way to draw a line through the data points.

In the example below, we have registered 18 cars. We have registered the car's speed (y), and the time of day in hour (x) the passing occurred.

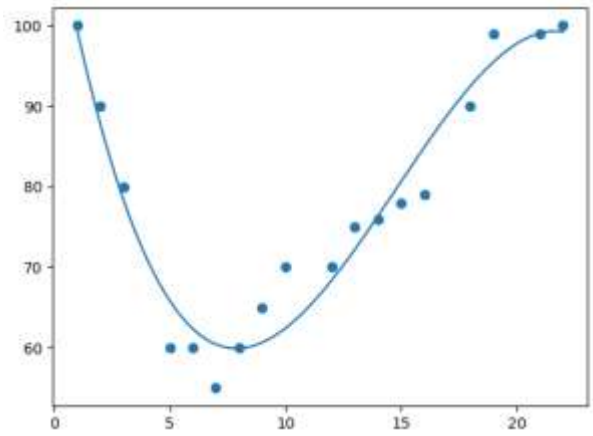
```
import numpy as np
import matplotlib.pyplot as plt

x = [1,2,3,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22]
y = [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]

mymodel = np.poly1d(np.polyfit
          (x, y, 3))

myline = np.linspace(1, 22, 100)

plt.scatter(x, y)
plt.plot(myline, mymodel(myline))
plt.show()
```



a- R-Squared

It is important to know how well the relationship between the values of the x and y is,

- If there are no relationship the polynomial regression cannot be used to predict anything.
- The relationship is measured with a value called the r-squared.
- The r-squared value ranges from 0 to 1, where 0 means no relationship, and 1 means 100% related.

```
import numpy
from sklearn.metrics import r2_score
x = [1,2,3,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22]
y = [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]
```

```
mymodel = numpy.poly1d(numpy.polyfit(x, y, 3))
print(r2_score(y, mymodel(x)))
```

Note: The result 0.94 shows that there is a very good relationship, and we can use polynomial regression in future predictions.

b- 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 car that passes around the time 17:00

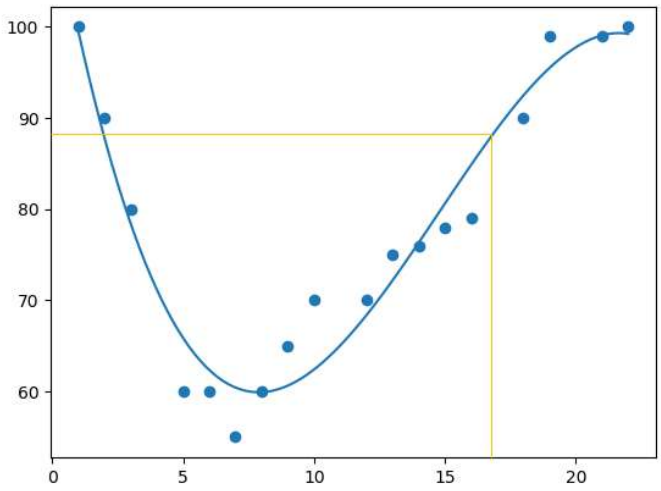
```
import numpy
from sklearn.metrics import r2_score

x = [1,2,3,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22]
y = [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]

mymodel = numpy.poly1d(numpy.polyfit(x, y, 3))

speed = mymodel(17)
print(speed)
```

The example predicted a speed to be 88.87, which we also could read from the diagram:



c- Bad Fit?

Example where polynomial regression would not be the best method to predict future values.

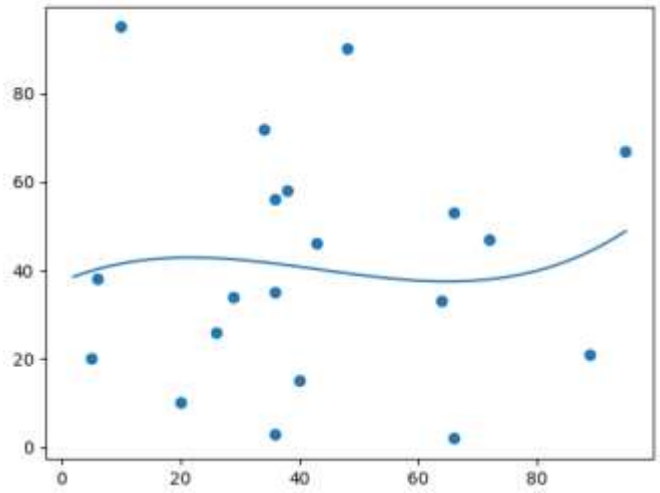
```
import numpy
import matplotlib.pyplot as plt

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]

mymodel = numpy.poly1d(numpy.polyfit(x, y, 3))

myline = numpy.linspace(2, 95, 100)

plt.scatter(x, y)
plt.plot(myline, mymodel(myline))
plt.show()
```



And the r-squared value is:

```
import numpy
from sklearn.metrics import r2_score

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]

mymodel = numpy.poly1d(numpy.polyfit(x, y, 3))

print(r2_score(y, mymodel(x)))
```

The result: **0.00995** indicates a very bad relationship, and tells us that this data set is not suitable for polynomial regression.

3- 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.

Car	Model	Volume (cm^3)	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.

In Python we have modules that will do the work using the Pandas module that allows to read csv files and return a DataFrame object.

Then make a list of the independent values in X and the dependent values in y.

```
x = df[['Weight', 'Volume']]
y = df['CO2']
```

From the sklearn module we will use the `LinearRegression()` method to create a linear regression object called `fit()` that takes the independent and dependent values as parameters and fills the regression object with data

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

```
import pandas as pd
import numpy as np
from sklearn import linear_model

df = pd.read_csv("cars_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:
new_data = pd.DataFrame([[2300, 1300]], columns=['Weight',
'Volume'])

predictedCO2 = regr.predict(new_data)

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.
- 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 as pd
from sklearn import linear_model as lm

df = pd.read_csv("cars_data.csv")

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

regr = lm.LinearRegression().fit(X, y)

print(regr.coef_)
```

Result: [0.00755095 0.00780526]

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

- Weight: 0.00755095 and 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 cm^3, the CO2 emission increases by 0.00780526 g.