Linear Regression

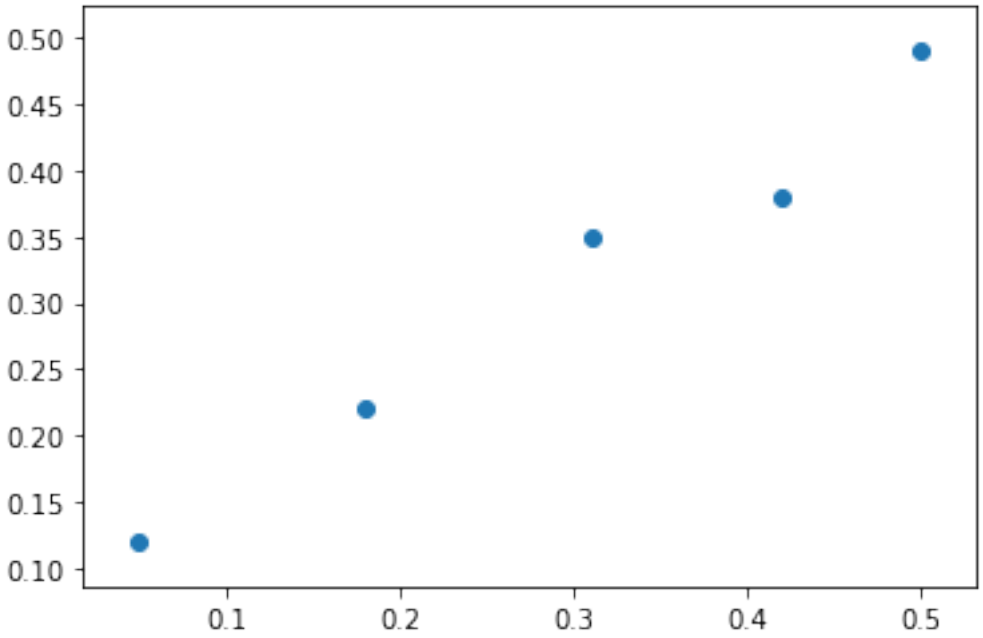
November 3, 2020

0.0.1 BASE DE DATOS

[2]:
```
# Base de datos para regresión lineal

from numpy import array
from matplotlib import pyplot
# Definiendo la base de datos
data = array([
    [.05, .12],
    [.18, .22],
    [.31, .35],
    [.42, .38],
    [.5, .49]
])
print(data)
# Dividir los datos en inputs y outputs
X, y = data[:, 0], data[:, 1]
X = X.reshape((len(X), 1))
# Scatter plot
pyplot.scatter(X, y)
pyplot.show()
```

```
[[0.05 0.12]
 [0.18 0.22]
 [0.31 0.35]
 [0.42 0.38]
 [0.5  0.49]]
```



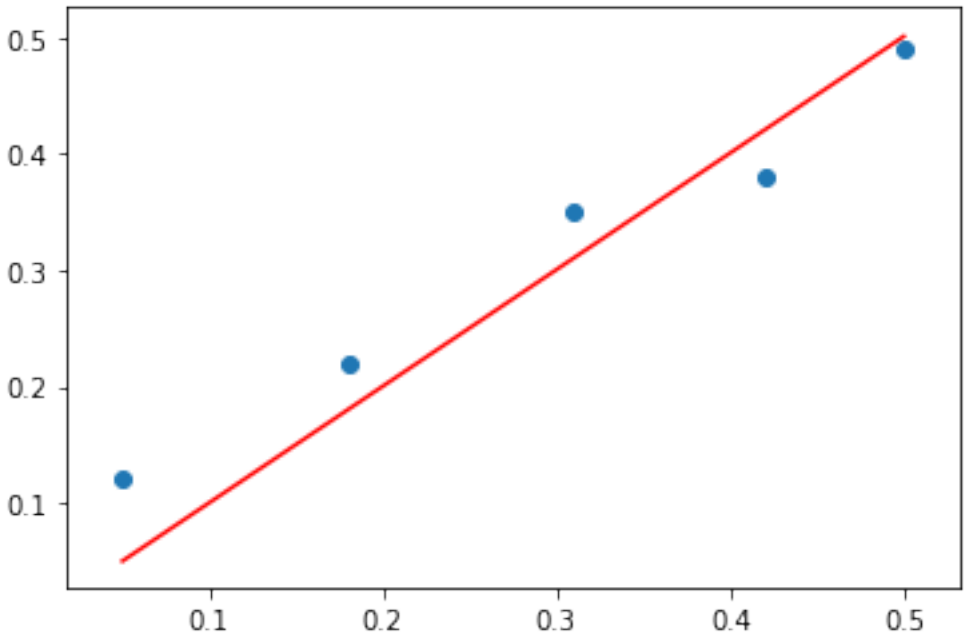
0.0.2 RESOLVIENDO MEDIANTE LA INVERSA

[3]:
```
# Solución directa

from numpy import array
from numpy.linalg import inv
from matplotlib import pyplot
# Definiendo la base de datos
data = array([
    [.05, .12],
    [.18, .22],
    [.31, .35],
    [.42, .38],
    [.5, .49]
])
print(data)
# Dividir los datos en inputs y outputs
X, y = data[:, 0], data[:, 1]
X = X.reshape((len(X), 1))
# Mínimos cuadrados
b = inv(X.T.dot(X)).dot(X.T).dot(y)
print(b)
# Prediciendo usando los coeficientes
y_gorro = X.dot(b)
# Gráfica de los datos y predicción
```

```
pyplot.scatter(X, y)
pyplot.plot(X, y_gorro, color = 'red')
pyplot.show()
```

```
[[0.05 0.12]
 [0.18 0.22]
 [0.31 0.35]
 [0.42 0.38]
 [0.5  0.49]]
[1.00233226]
```



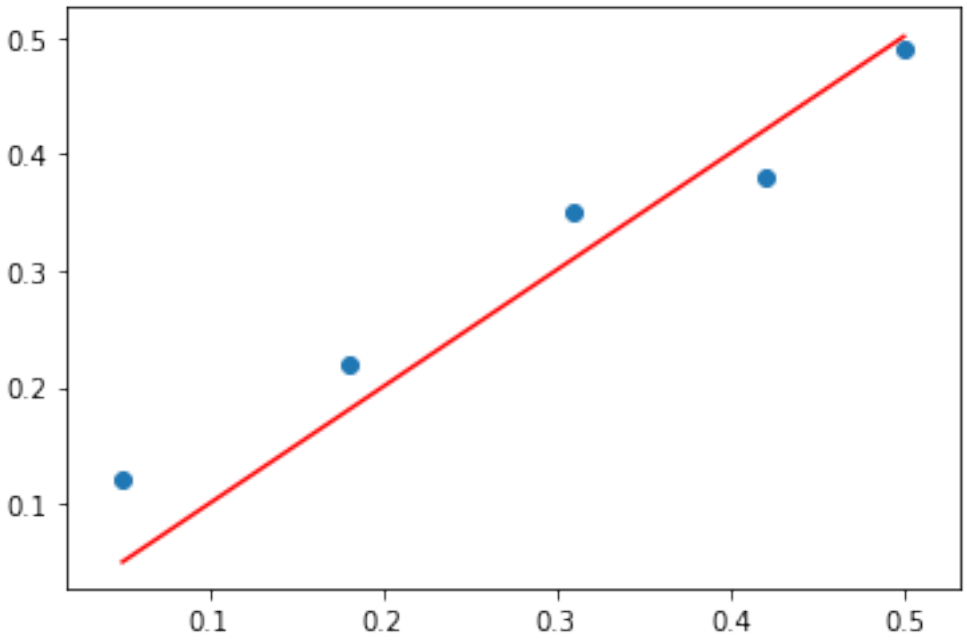
0.0.3 SOLUCIÓN CON DESCOMPOSICIÓN QR

[4]:
```
# Solución con descomposición QR

from numpy import array
from numpy.linalg import inv, qr
from matplotlib import pyplot
# Definiendo la base de datos
data = array([
    [.05, .12],
    [.18, .22],
    [.31, .35],
    [.42, .38],
    [.5, .49]
```

```
])
print(data)
# Dividir los datos en inputs y outputs
X, y = data[:, 0], data[:, 1]
X = X.reshape((len(X), 1))
# Factorizando
Q, R = qr(X)
b = inv(R).dot(Q.T).dot(y)
print(b)
# Prediciendo usando los coeficientes
y_gorro = X.dot(b)
# Gráfica de los datos y predicción
pyplot.scatter(X, y)
pyplot.plot(X, y_gorro, color = 'red')
pyplot.show()
```

```
[[0.05 0.12]
 [0.18 0.22]
 [0.31 0.35]
 [0.42 0.38]
 [0.5  0.49]]
[1.00233226]
```

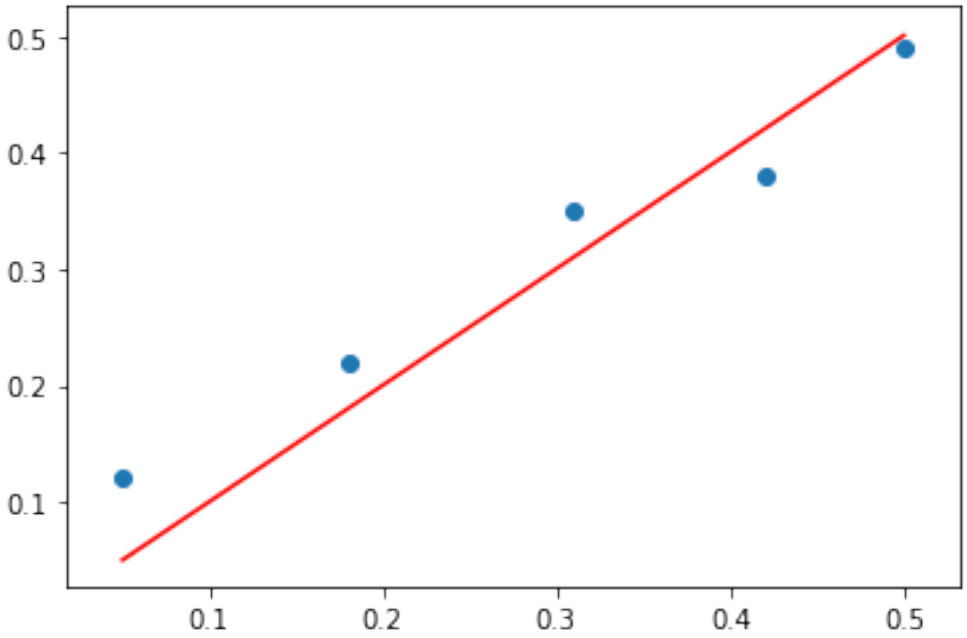


0.0.4 SOLUCIÓN CON SVD Y PSEUDOINVERSA

[7]:
```
# Solución con SVD y pseudoinversa

from numpy import array
from numpy.linalg import pinv
from matplotlib import pyplot
# Definiendo la base de datos
data = array([
    [.05, .12],
    [.18, .22],
    [.31, .35],
    [.42, .38],
    [.5, .49]
])
print(data)
# Dividir los datos en inputs y outputs
X, y = data[:, 0], data[:, 1]
X = X.reshape((len(X), 1))
# Calculando los coeficientes
b = pinv(X).dot(y)
print(b)
# Prediciendo usando los coeficientes
y_gorro = X.dot(b)
# Gráfica de los datos y predicción
pyplot.scatter(X, y)
pyplot.plot(X, y_gorro, color = 'red')
pyplot.show()
```

```
[[0.05 0.12]
 [0.18 0.22]
 [0.31 0.35]
 [0.42 0.38]
 [0.5  0.49]]
[1.00233226]
```



0.0.5 SOLUCIÓN CON FUNCIÓN DE CONVENIENCIA

Usamos la función `lstsq()`, la cual usa SVD.

[8]:
```
# Solución con lstsq()

from numpy import array
from numpy.linalg import lstsq
from matplotlib import pyplot
# Definiendo la base de datos
data = array([
    [.05, .12],
    [.18, .22],
    [.31, .35],
    [.42, .38],
    [.5, .49]
])
print(data)
# Dividir los datos en inputs y outputs
X, y = data[:, 0], data[:, 1]
X = X.reshape((len(X), 1))
# Calculando los coeficientes
b, residuals, rank, s = lstsq(X, y)
print(b)
# Prediciendo usando los coeficientes
y_gorro = X.dot(b)
```

```
# Gráfica de los datos y predicción
pyplot.scatter(X, y)
pyplot.plot(X, y_gorro, color = 'red')
pyplot.show()
```

```
[[0.05 0.12]
 [0.18 0.22]
 [0.31 0.35]
 [0.42 0.38]
 [0.5  0.49]]
[1.00233226]
```

```
D:\Anaconda3\lib\site-packages\ipykernel_launcher.py:19: FutureWarning: `rcond`
parameter will change to the default of machine precision times ``max(M, N)``
where M and N are the input matrix dimensions.
To use the future default and silence this warning we advise to pass
`rcond=None`, to keep using the old, explicitly pass `rcond=-1`.
```