UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE CIENCIAS FÍSICO

MATEMÁTICAS

Minería de Datos

Ejercicios 1

Profesor: Mayra Cristina Berrones Reyes

Alumnos:

Saúl Arath Hernández Hernández 1815642

Miguel Alejandro Noriega Ortega 1798528

Alison L. Roldán Sánchez 1806488

Aranza Sifuentes Soto 1887979

Valeria Solís Agundis 1815413

Tadeo Alejandro Suárez Martínez 1806069

Carrera: LA

Semestre: 7mo

Grupo: 002

29 de septiembre de 2020

REGRESIÓN LINEAL

La idea consiste en obtener la ecuación de la forma

$$y = mx + b$$

que mejor se ajuste a los datos que se tengan.

Tenemos que:

$$m = \frac{\sum x \sum y - n\sum(xy)}{(\sum x)^2 - n\sum x^2}$$

y

$$b = \bar{y} - m\bar{x}$$

El coeficiente de correlación se calcula como

$$R = \frac{\sigma_{xy}}{\sigma_x \sigma_y}$$

donde:

$$\sigma_x = \sqrt{\frac{\sum\left(x^2\right)}{n} - \bar{x}^2}, \quad \sigma_y = \sqrt{\frac{\sum\left(y^2\right)}{n} - \bar{y}^2}$$

$$\sigma_{xy} = \frac{\sum(xy)}{n} - \bar{x} \cdot \bar{y}$$

(desviaciones típicas y covarianza, respectivamente)

De la siguiente tabla, trazar una linea que se apegue lo más posible a los datos graficados:

In [4]:

```
#DATOS
#x=peso, y=altura

x=[68.78,74.11,71.73,69.88,67.25,68.78,68.34,67.01,63.45,71.19,
   67.19,65.80,64.30,67.97,71.18,65.27,66.09,67.51,70.10,68.25,
   67.89,68.14,69.08,72.80,67.42,68.49,68.61,74.03,71.52,69.18]

y=[162,212,220,206,152,183,167,175,156,186,
   183,163,163,172,194,168,161,164,188,187,
   162,192,184,206,175,154,187,212,195,205]
```

In [5]:

```
len(x)
```

Out[5]:

```
30
```

In [6]:

```
len(y)
```

Out[6]:

```
30
```

In [8]:

```
import numpy as np                    #trabajar con matrices
import matplotlib.pyplot as plt       #para hacer graficos
```

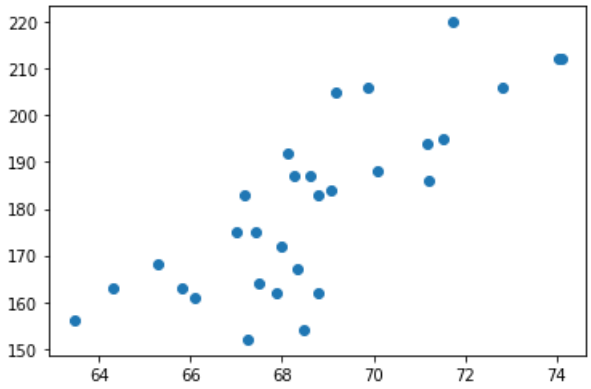
In [9]:

```
plt.scatter(x,y)
plt.show

n=len(x)        #longitud
n

x=np.array(x)    #manipular mejor los datos
y=np.array(y)    #arriba era una lista, aqui es ya un vector, las listas no se pueden multiplicar
x,y

sumx=sum(x)
sumy=sum(y)
sumx2=sum(x*x)
sumy2=sum(y*y)
sumxy=sum(x*y)
promx=sumx/n
promy=sumy/n
```



In [10]:

```
x,y
```

Out[10]:

```
(array([68.78, 74.11, 71.73, 69.88, 67.25, 68.78, 68.34, 67.01, 63.45,
        71.19, 67.19, 65.8 , 64.3 , 67.97, 71.18, 65.27, 66.09, 67.51,
        70.1 , 68.25, 67.89, 68.14, 69.08, 72.8 , 67.42, 68.49, 68.61,
        74.03, 71.52, 69.18]),
 array([162, 212, 220, 206, 152, 183, 167, 175, 156, 186, 183, 163, 163,
        172, 194, 168, 161, 164, 188, 187, 162, 192, 184, 206, 175, 154,
        187, 212, 195, 205]))
```

In [11]:

```
m=((sumx*sumy-n*sumxy)/((sumx**2)-n*sumx2))
b=promy-m*promx
m,b
```

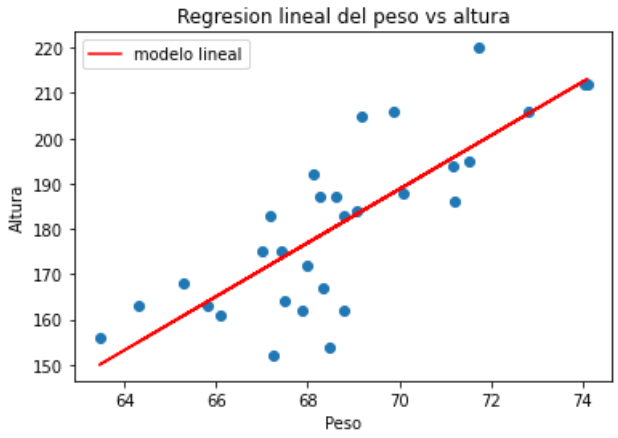
Out[11]:

```
(5.924381452685481, -225.93881545595633)
```

In [12]:

```
plt.scatter(x,y)
plt.plot(x,x*m+b,label="modelo lineal",color="red")
plt.xlabel("Peso")
plt.ylabel("Altura")
plt.title("Regresion lineal del peso vs altura")
plt.legend()
plt.show()
```

```
plt.show()
```



In [13]:

```
sigmax=np.sqrt((sumx2/n)-promx**2)
sigmay=np.sqrt((sumy2/n)-promy**2)
sigmaxy=(sumxy/n)-promx*promy
R2=(sigmaxy/(sigmax*sigmay))**2
R2
```

Out[13]:

```
0.6363206596436957
```

El coeficiente de determinación es de 63.63%, lo que nos dice que este porcentaje de la varianza de los datos está representado por el modelo lineal.

TÉCNICA DE ASOCIACIÓN

Utilizando el algoritmo a priori, y la técnica de asociación, realizar la tabla de relaciones y resuelve cuál es el nivel K de soporte más alto al que podemos llegar con estos datos teniendo un umbral de 0.5.

ID	TRANSACCIONES
1	ABCE
2	BE
3	CDE
4	ACD
5	AC

In [35]:

```
pip install mlxtend
```

```
Collecting mlxtendNote: you may need to restart the kernel to use updated packages.

  Downloading mlxtend-0.17.3-py2.py3-none-any.whl (1.3 MB)
Requirement already satisfied: setuptools in c:\users\saula\anaconda3\lib\site-packages (from
mlxtend) (49.2.0.post20200714)
Requirement already satisfied: scikit-learn>=0.20.3 in c:\users\saula\anaconda3\lib\site-packages
(from mlxtend) (0.23.1)
Requirement already satisfied: scipy>=1.2.1 in c:\users\saula\anaconda3\lib\site-packages (from
mlxtend) (1.5.0)
Requirement already satisfied: matplotlib>=3.0.0 in c:\users\saula\anaconda3\lib\site-packages
(from mlxtend) (3.2.2)
Requirement already satisfied: numpy>=1.16.2 in c:\users\saula\anaconda3\lib\site-packages (from
mlxtend) (1.18.5)
Requirement already satisfied: joblib>=0.13.2 in c:\users\saula\anaconda3\lib\site-packages (from
mlxtend) (0.16.0)
Requirement already satisfied: pandas>=0.24.2 in c:\users\saula\anaconda3\lib\site-packages (from
mlxtend) (1.0.5)
```

```
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\saula\anaconda3\lib\site-packages
(from scikit-learn>=0.20.3->mlxtend) (2.1.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
c:\users\saula\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (2.4.7)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\saula\anaconda3\lib\site-packages
(from matplotlib>=3.0.0->mlxtend) (2.8.1)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\saula\anaconda3\lib\site-packages
(from matplotlib>=3.0.0->mlxtend) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\saula\anaconda3\lib\site-packages (from
matplotlib>=3.0.0->mlxtend) (0.10.0)
Requirement already satisfied: pytz>=2017.2 in c:\users\saula\anaconda3\lib\site-packages (from
pandas>=0.24.2->mlxtend) (2020.1)
Requirement already satisfied: six>=1.5 in c:\users\saula\anaconda3\lib\site-packages (from
python-dateutil>=2.1->matplotlib>=3.0.0->mlxtend) (1.15.0)
Installing collected packages: mlxtend
Successfully installed mlxtend-0.17.3
```

In [37]:

```
import numpy as np
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules
```

In [62]:

```
file = pd.read_excel('ejercicio1_parte2.xlsx')
file.head()
```

Out[62]:

	A	B	C	D	E
0	1	1	1	0	1
1	0	1	0	0	1
2	0	0	1	1	1
3	1	0	1	1	0
4	1	0	1	0	1

In [63]:

```
frequent_itemsets = apriori(file, min_support=0.5, use_colnames=True)
frequent_itemsets
```

Out[63]:

	support	itemsets
0	0.6	(A)
1	0.8	(C)
2	0.8	(E)
3	0.6	(A, C)
4	0.6	(E, C)

In [ ]: