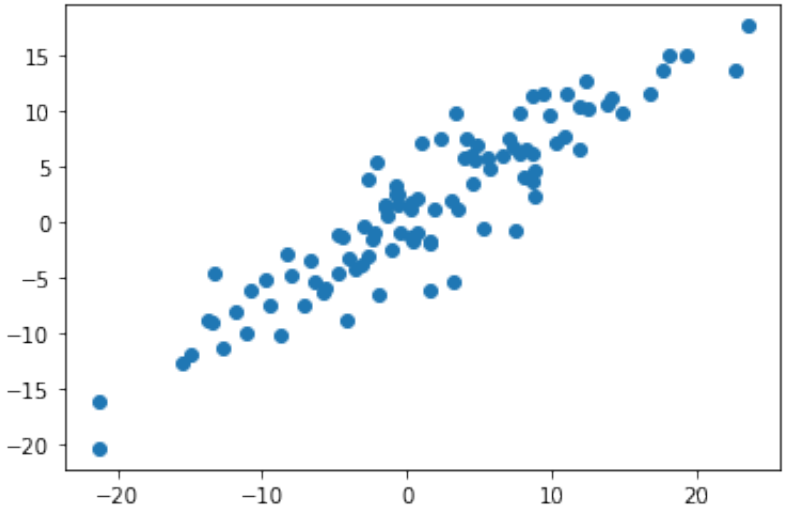
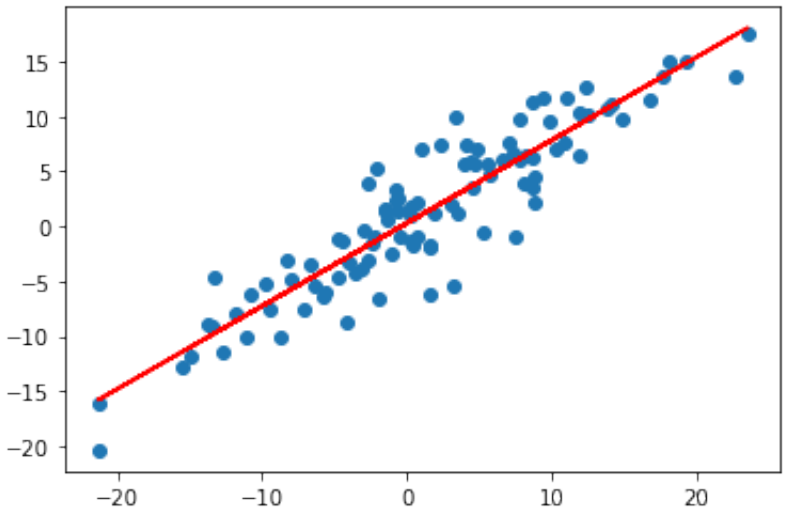
In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [ ]:
```
# Segunda parte
data_ej1: pd.DataFrame = pd.read_csv('ejercicio_1.csv', sep=',')
x_index: pd.Index = data_ej1.keys()[0]
y_index: pd.Index = data_ej1.keys()[1]

x: pd.DataFrame = data_ej1[x_index]
y: pd.DataFrame = data_ej1[y_index]
```

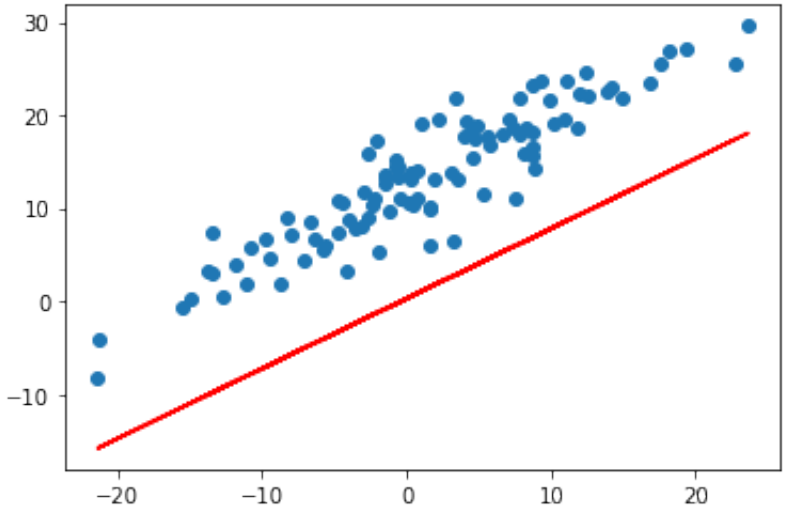
In [ ]:
```
# 1. a)
plt.scatter(x,y)
plt.show()
```



In [ ]:
```
# 1. b)
# Beta* = ((X^t.X)^-1).X^t.y
X = np.array([np.ones(len(x)),x]).T
Y = np.array([y]).T
Xt = np.transpose(X)
Xt_X = np.dot(np.transpose(X),X)
Xt_X_inv = np.linalg.inv(Xt_X)
B_est = np.dot(Xt_X_inv,np.dot(Xt,Y))
plt.scatter(x,y)
plt.plot(x, B_est[0] + B_est[1]*x, color = 'red')
plt.show()
```



In [ ]:
```
# 1. c)
plt.scatter(x,y+12)
plt.plot(x, B_est[0] + B_est[1]*x, color = 'red')
plt.show()
```

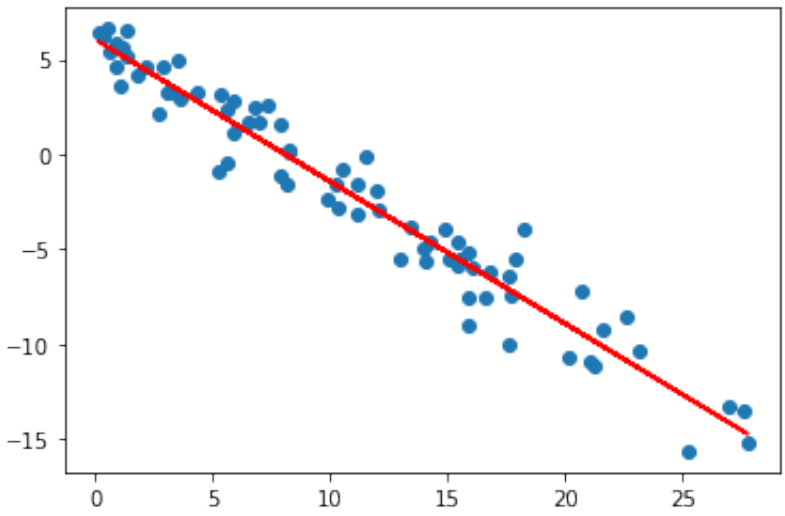


Si bien la pendiente de la recta es bastante buena, la aproximación no es buena dado que la ordenada al origen aleja a la recta de los puntos.

1. d) Para poder aproximar cualquier recta en el plano es necesario tener en cuenta las ordenadas al origen, es decir, tener un β_0. Para ello puede agregarse una columna de todos unos (1) a la matriz X

In [ ]:
```
# 2. a)
data_ej2: pd.DataFrame = pd.read_csv('ejercicio_2.csv', sep=',')
x2_index: pd.Index = data_ej2.keys()[0]
y2_index: pd.Index = data_ej2.keys()[1]

x2: pd.DataFrame = data_ej2[x2_index]
y2: pd.DataFrame = data_ej2[y2_index]
```

In [ ]:
```
X2 = np.array([np.ones(len(x2)),x2]).T
Y2 = np.array([y2]).T
Xt2 = np.transpose(X2)
Xt_X2 = np.dot(np.transpose(X2),X2)
Xt_X_inv2 = np.linalg.inv(Xt_X2)
B_est2 = np.dot(Xt_X_inv2,np.dot(Xt2,Y2))
plt.scatter(x2,y2)
plt.plot(x2, B_est2[0] + B_est2[1]*x2, color = 'red')
plt.show()
```



1. b) Si bien la aproximación realizada fue buena, si los datos en un futuro no siguen una tendencia lineal decreciente como la del modelo, las estimaciones pueden ser muy malas, pudiendo arrojar estimaciones sin sentido.

In [ ]:
```
# Tercera Parte
# 1. a)
data_ej3: pd.DataFrame = pd.read_csv('ejercicio_3.csv')
entrenamiento = data_ej3.loc[0:314]
test = data_ej3.loc[315:413]
completo = data_ej3.loc[0:413]

# X1:
x_1: pd.DataFrame = entrenamiento['X1 transaction date']
x_1 = np.array(x_1)

# X2:
x_2: pd.DataFrame = entrenamiento['X2 house age']
x_2 = np.array(x_2)

# X3:
x_3: pd.DataFrame = entrenamiento['X3 distance to the nearest MRT station']
x_3 = np.array(x_3)

# X4:
x_4: pd.DataFrame = entrenamiento['X4 number of convenience stores']
x_4 = np.array(x_4)

# X5:
x_5: pd.DataFrame = entrenamiento['X5 latitude']
x_5 = np.array(x_5)

# X6:
x_6: pd.DataFrame = entrenamiento['X6 longitude']
x_6 = np.array(x_6)

X = np.array([np.ones(len(x_1)),x_1,x_2,x_3,x_4,x_5,x_6]).T

y = np.array([entrenamiento['Y house price of unit area']]).T

X_t = np.transpose(X)
Xt_X = np.dot(X_t,X)
Xt_X_inv = np.linalg.inv(Xt_X)

Betas = np.dot(Xt_X_inv,np.dot(X_t,y)) # Fórmula de solución óptima al probl
print(Betas)
```

```
[[-1.48729504e+04]
 [ 4.96257526e+00]
 [-2.79853534e-01]
 [-4.29360373e-03]
 [ 1.11451659e+00]
 [ 2.63780784e+02]
 [-1.36639185e+01]]
```

In [ ]:
```
# 1. b)
y_sombrero = Betas[0] + x_1*Betas[1] + x_2*Betas[2] + x_3*Betas[3] + x_4*Bet
print(y_sombrero)
```

```
[47.441236   48.18652184 49.49007785 49.07818411 46.71288633 31.51438766
 38.93429447 47.16517526  9.39728843 34.98835541 33.66120364 53.23865749
 40.77286413 27.150665   46.61932964 38.43797225 51.1767929  37.5765122
 46.56141197 47.62066195 35.56875535 49.65984984 28.50344265 48.11429112
 34.86591836 32.40042985 46.82005969 41.62339725 42.91267347 46.05743047
 13.30283546 41.04083435 30.77241382 46.70763222 47.68642709 35.743442
 30.72997113 30.49969101 44.72414197 46.35830983 14.26448874 16.29638442
 36.11580951 40.79466697 47.98152101 39.56577583 46.65882245 37.57005518
 12.62385052  9.61139485 41.93778738 23.43382551 35.09654191 41.51269557
 45.96943886 22.69910063 43.25750882 45.50127676 13.49335186 40.27477756
 34.20587085 50.65001277 29.91959849 48.00950637 33.12853802 43.46926832
 47.03940814 46.96314004 43.4323655  45.99202923 54.39534525 35.20070896
 45.05864821 15.0372681  53.38281239 32.06827251 38.08949993 28.59194659
 33.02389489 27.69278766 45.02296011 37.54607893 45.71867692 24.77692992
 44.18676499 49.49753964 31.05647415 16.68029281 37.98485201 16.36619613
 43.11110021 39.58954632 28.33533811 22.98562806 34.62265259 44.20988543
 53.62752846 37.82725658 47.54298294 53.62752846 45.52523913 39.72573038
 47.42331653 43.53469249 35.18334184 49.10739902 48.13079305 31.26750779
 36.31999566 37.02646185 45.00568757 44.95709231 34.04756893 42.95800061
 43.71687536 39.09425231  1.52263189 13.05680885 33.86773503 47.89890598
 41.92955189 43.49813339 40.80815202 44.28391975 47.34647433 47.84017285
 35.50165797 49.5173436  36.74624308 43.61529589 45.10878888 36.61456811
 40.65246511 48.42148877 36.41839981 30.7466827  44.50992996 45.22959123
 39.03223005 39.56020567 46.77020358 40.36477085 41.91391872 43.49813339
 30.11082004 46.63291065 40.97388205 47.26105868 14.87027821 40.22043937
 40.11958951 55.06086928 31.3234785  42.96688777 16.26839907 15.0372681
 26.03723154 42.70723623 45.69460307 39.98196698 46.88157609 41.93926645
 12.7983358  46.96314004 41.3857758  35.80700486 52.00554297 45.35824132
 36.76312138 30.77002736 12.74165529 48.74456423 54.39534525 37.62848956
 52.21978755 35.92991003 13.23788469 44.83977558 46.7808373  39.80529602
  9.37678874 52.15000964 33.95595954 15.79427435 21.9238144  30.77049631
 30.13750532 22.48932933 42.3787994  13.54312981 44.02003127 39.59901208
 38.06375278 48.94638313 15.43423065 39.44549601 38.22018435 42.26675027
 39.62086737 46.34810602 33.5733797  48.72226107 33.57464073 44.04493432
 27.69278766 37.6286749  50.72560553 34.10158283 28.57022389 41.07980444
 49.96695332 44.35174403 38.48171518 52.02599903 35.7561722  46.61747368
 32.31044119 41.6407468  43.08623964 42.7622499  44.89712219 40.02942542
 49.22829981 39.58954632 42.02058306 48.2520666  13.45291349 43.70138717
 36.25048148 26.41754858 36.61456811 12.95367659 12.19082843 44.43807386
 34.2909937  43.40199112 55.48773704 38.61574722 38.25486161 34.56462221
 33.58363091 46.90003974 34.31157372 46.80767289 39.20808309 45.97324921
 43.51193891 31.44876243 31.1305911   8.34964261 39.58663582 23.09868261
 50.86931128 26.13503507 48.05638216  9.04136539 38.0117496  39.70895264
 52.00554297 32.60716974 37.67872989 29.84168609 45.20162207 36.23065971
 39.83895566 40.74978376 34.1875907  36.72815135 46.19681716 32.22326135
 42.6110112  44.00674516 39.94411406 42.48394757 42.65510881 47.62066195
 36.70152966 31.29419307 43.53469249 41.00988678 47.98276465 54.28808991
 31.41735976 33.7451066  43.39096285 36.6630454  51.2861676  36.67354431
 48.33216564 47.82576045 31.00787425 45.11240579 38.57953752 42.82196751
 45.45991636 30.85416867 43.22681797 33.9635852  15.04948035 47.86652821
 45.30579283 29.4292114  31.41026759 32.98560021 34.76278587 45.94145351
 42.55606269 21.25087384 47.54298294 24.49663539 31.68646709 40.77044945
 44.4343514  46.19036067 47.44941122]
```

In [ ]:
```
# 1. c)
y_somb_vec = np.array([y_sombrero]).T
ecm = np.sum((y - y_somb_vec)**2) / len(y)
print(ecm)
```

```
83.16551901820162
```

In [ ]:
```
# 2. a) Repetimos lo realizado en el entrenamiento pero con el conjunto de t
# X1:
x_1t: pd.DataFrame = test['X1 transaction date']
x_1t = np.array(x_1t)

# X2:
x_2t: pd.DataFrame = test['X2 house age']
x_2t = np.array(x_2t)

# X3:
x_3t: pd.DataFrame = test['X3 distance to the nearest MRT station']
x_3t = np.array(x_3t)

# X4:
x_4t: pd.DataFrame = test['X4 number of convenience stores']
x_4t = np.array(x_4t)

# X5:
x_5t: pd.DataFrame = test['X5 latitude']
x_5t = np.array(x_5t)

# X6:
x_6t: pd.DataFrame = test['X6 longitude']
x_6t = np.array(x_6t)

Xt = np.array([np.ones(len(x_1t)),x_1t,x_2t,x_3t,x_4t,x_5t,x_6t]).T

yt = np.array([test['Y house price of unit area']]).T

yt_sombrero = Betas[0] + x_1t*Betas[1] + x_2t*Betas[2] + x_3t*Betas[3] + x_4

yt_somb_vec = np.array([yt_sombrero]).T
ecm_test = np.sum((yt - yt_somb_vec)**2) / len(yt)
print(ecm_test)
```

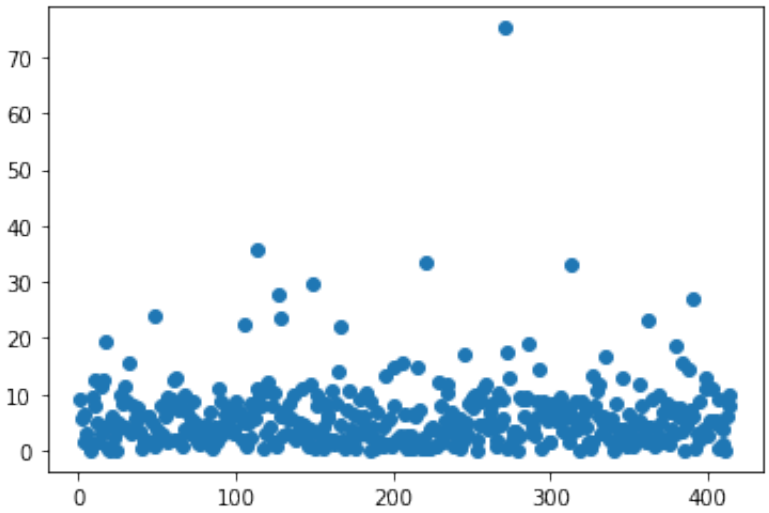
```
58.66453912169032
```

El Error Cuadrático Medio es menor por lo que la estimación es mejor. Esta discrepancia se debe a que la diferencia entre los datos de test y los datos de entrenamiento es menor.

In [ ]:
```
# 2. b) Repetimos lo realizado en el entrenamiento pero con el conjunto comp
# X1:
xc_1: pd.DataFrame = completo['X1 transaction date']
xc_1 = np.array(xc_1)

# X2:
xc_2: pd.DataFrame = completo['X2 house age']
xc_2 = np.array(xc_2)

# X3:
xc_3: pd.DataFrame = completo['X3 distance to the nearest MRT station']
xc_3 = np.array(xc_3)

# X4:
xc_4: pd.DataFrame = completo['X4 number of convenience stores']
xc_4 = np.array(xc_4)

# X5:
xc_5: pd.DataFrame = completo['X5 latitude']
xc_5 = np.array(xc_5)

# X6:
xc_6: pd.DataFrame = completo['X6 longitude']
xc_6 = np.array(xc_6)

Xc = np.array([np.ones(len(xc_1)),xc_1,xc_2,xc_3,xc_4,xc_5,xc_6]).T

yc = np.array([completo['Y house price of unit area']]).T

Xc_t = np.transpose(Xc)
Xct_Xc = np.dot(Xc_t,Xc)
Xct_Xc_inv = np.linalg.inv(Xct_Xc)

Betas_c = np.dot(Xct_Xc_inv,np.dot(Xc_t,yc))

yt_c_sombrero = Betas_c[0] + x_1t*Betas_c[1] + x_2t*Betas_c[2] + x_3t*Betas_

yt_c_somb_vec = np.array([yt_c_sombrero]).T
ecm_c = np.sum((yt - yt_c_somb_vec)**2) / len(yt)
print(ecm_c)
```

```
57.3918975956645
```

In [ ]:
```
# 3.
yc_sombrero = Betas_c[0] + xc_1*Betas_c[1] + xc_2*Betas_c[2] + xc_3*Betas_c[
yc_sombrero = yc_sombrero.reshape(-1, 1)
error_cometido = (yc - yc_sombrero)
n_casas = np.array([completo['No']]).T
plt.scatter(n_casas, abs(error_cometido))
plt.show()
```



1. Agregar una columna que informe el año en que cada casa fue construida no dismnuiría el ECM ya que no aporta información nueva. Esta información es la misma que la que provee la variable X2 house age, es decir, la edad de la casa.