Carregando uma base de dados de regressão que NÃO esteja nos toy datasets do scikit-learn;

A base de dados escolhida foi o conjunto de dados de habitações da Califórnia (para regressão), disponível no Real world datasets do Scikit-learn.

In [2]:
```
from sklearn.datasets import fetch_california_housing
dataset = fetch_california_housing()
```

In [3]:
```
print(dataset.DESCR)
```

```
.. _california_housing_dataset:

California Housing dataset
--------------------------

**Data Set Characteristics:**

    :Number of Instances: 20640

    :Number of Attributes: 8 numeric, predictive attributes and the target

    :Attribute Information:
        - MedInc        median income in block group
        - HouseAge      median house age in block group
        - AveRooms      average number of rooms per household
        - AveBedrms     average number of bedrooms per household
        - Population    block group population
        - AveOccup      average number of household members
        - Latitude      block group latitude
        - Longitude     block group longitude

    :Missing Attribute Values: None

This dataset was obtained from the StatLib repository.
https://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html

The target variable is the median house value for California districts,
expressed in hundreds of thousands of dollars ($100,000).

This dataset was derived from the 1990 U.S. census, using one row per census
block group. A block group is the smallest geographical unit for which the U.S.
Census Bureau publishes sample data (a block group typically has a population
of 600 to 3,000 people).

An household is a group of people residing within a home. Since the average
number of rooms and bedrooms in this dataset are provided per household, these
columns may take surpinsingly large values for block groups with few households
and many empty houses, such as vacation resorts.

It can be downloaded/loaded using the
:func:`sklearn.datasets.fetch_california_housing` function.

.. topic:: References

    - Pace, R. Kelley and Ronald Barry, Sparse Spatial Autoregressions,
      Statistics and Probability Letters, 33 (1997) 291-297
```

Features names

In [4]:
```
print(dataset.feature_names)
```

```
['MedInc', 'HouseAge', 'AveRooms', 'AveBedrms', 'Population', 'AveOccup', 'Latitud
e', 'Longitude']
```

Criando as matrizes de dados X e y

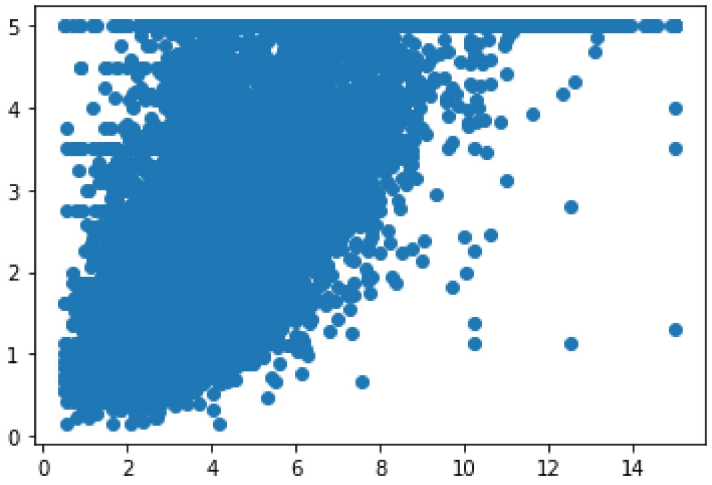
In [5]:
```
X, y = fetch_california_housing(return_X_y=True)
print(X.shape, y.shape)
```

```
(20640, 8) (20640,)
```

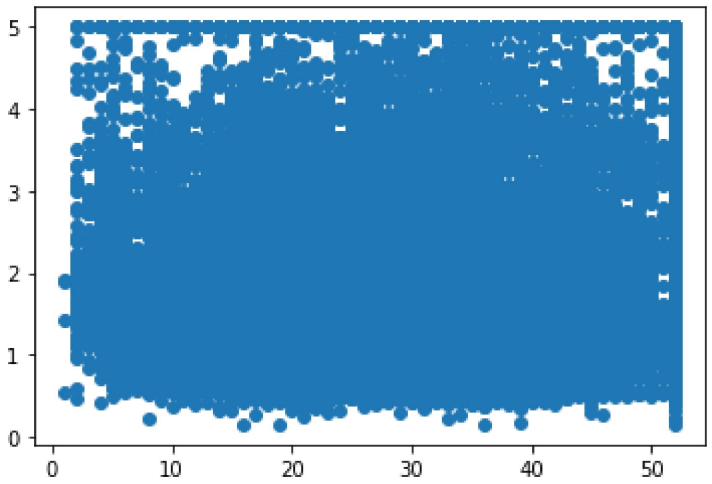
Base de dados que será trabalhada

In [6]:
```
import matplotlib.pylab as plt
for i in range(X.shape[1]):
    print(i)
    plt.scatter(X[:,i], y)
    plt.show()
```

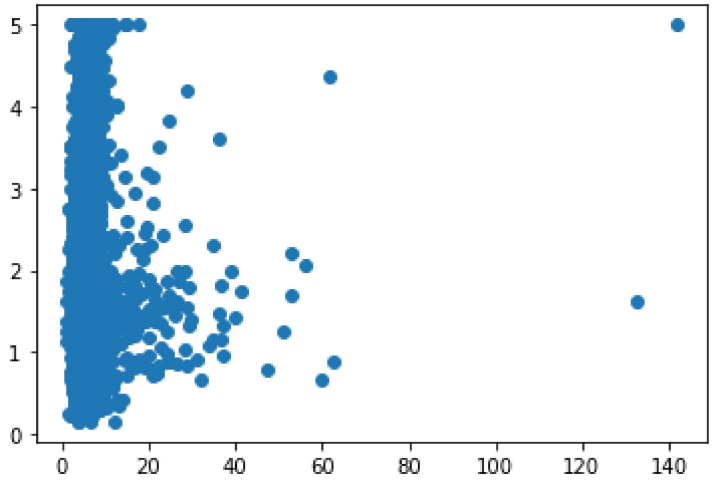
```
0
```



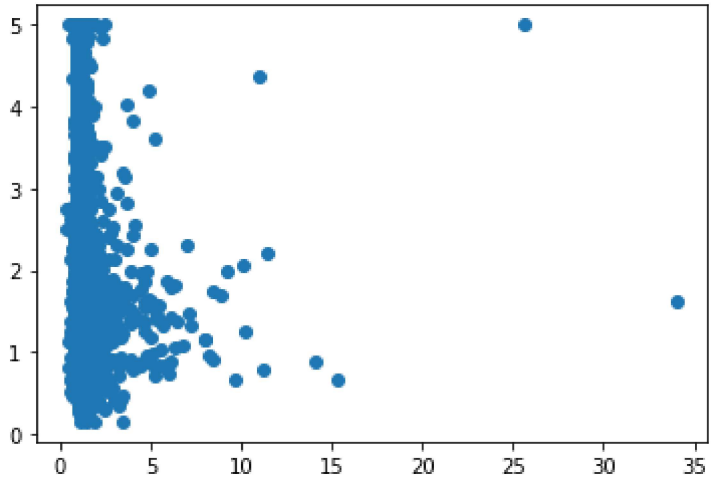

```
1
```



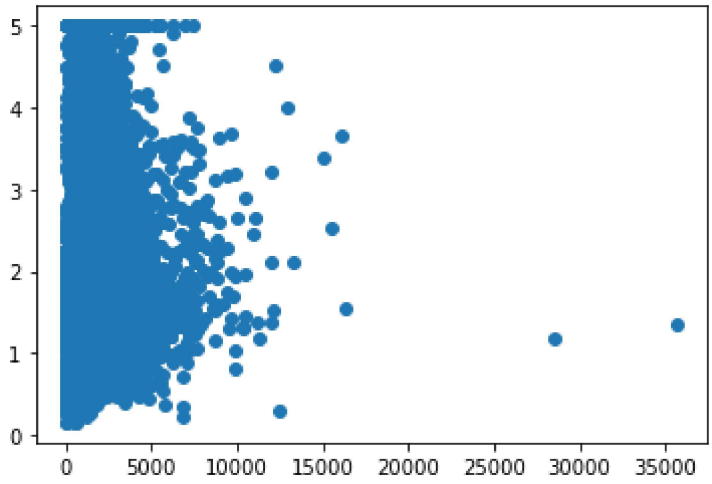

```
2
```


0
1
2
3
4
5
0
20
40
60
80
100
120
140

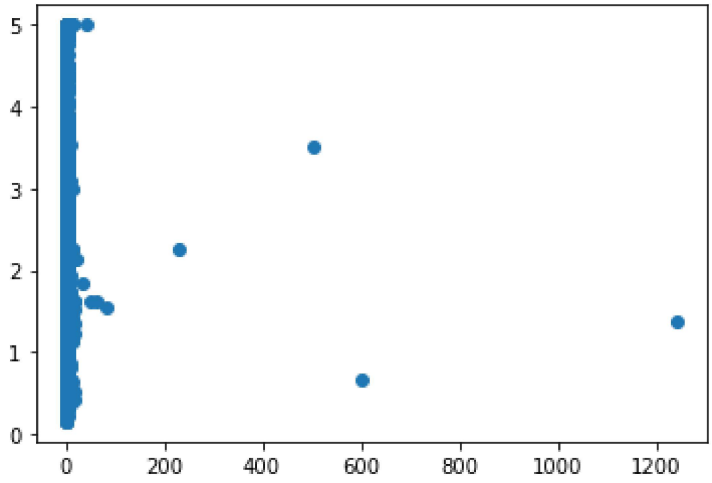
3


0
1
2
3
4
5
0
5
10
15
20
25
30
35

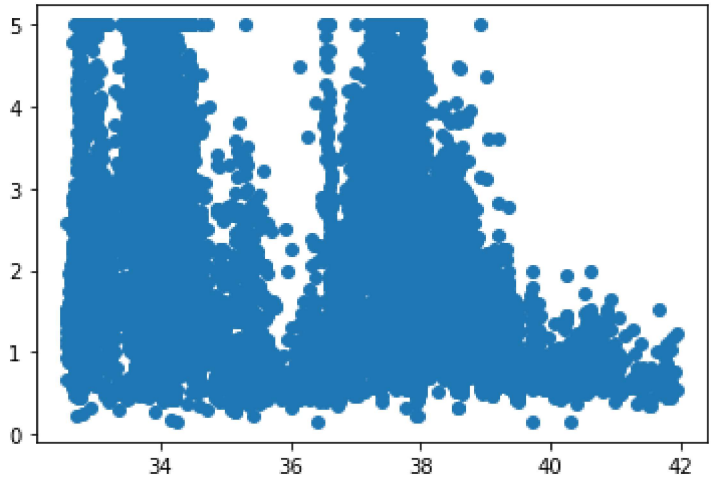
4


0
1
2
3
4
5
0
5000
10000
15000
20000
25000
30000
35000

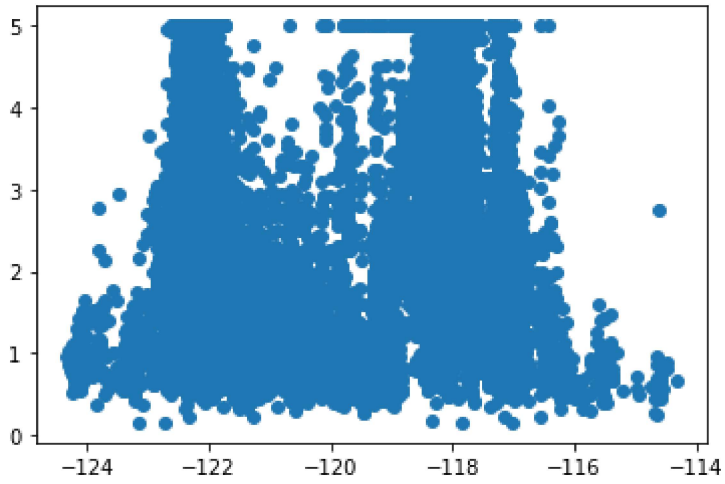
5



6



7



Modelo de Regressão Linear

Definindo o modelo de regressão

In [7]:
```
from sklearn import linear_model
modelo = linear_model.LinearRegression()
```

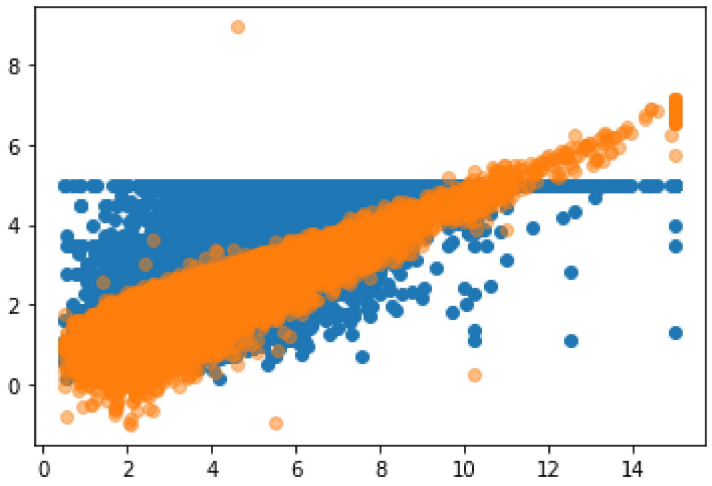
In [8]:
```
modelo.fit(X, y)
ypred = modelo.predict(X)
ypred.shape
```

Out[8]:
```
(20640,)
```

Visualização da reta de regressão linear que o modelo gerou, com os mesmos dados que criaram o modelo.

In [9]:
```
X0 = X[:,0]
plt.scatter(X0, y)
plt.scatter(X0, ypred, alpha=0.5)
```

Out[9]:
```
<matplotlib.collections.PathCollection at 0x197f5bffb80>
```



Calculando o MSE (erro quadrático médio)

In [10]:
```
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y, ypred)
print(mse)
```

```
0.5243209861846071
```

Calculando o MAE (Erro Absoluto Médio)

In [11]:
```
from sklearn.metrics import mean_absolute_error
mae = mean_absolute_error(y, ypred)
print(mae)
```

```
0.5311643817546478
```

Calculando o RMSE (Raiz Quadrada do Erro Médio)

In [12]:
```
rmse = mean_squared_error(y, ypred, squared=False)
print(rmse)
```

```
0.7241001216576387
```

Utilizando os atributos coef *e intercept* do modelo, sendo eles respectivamente o coeficiente angular e linear de nossa reta

In [13]:
```
print(modelo.intercept_,'\n', modelo.coef_)
```

```
-36.941920207184396
 [ 4.36693293e-01  9.43577803e-03 -1.07322041e-01  6.45065694e-01
 -3.97638942e-06 -3.78654265e-03 -4.21314378e-01 -4.34513755e-01]
```

Plotando um gráfico (2D) do MSE em função do atributo intercept *da classe LinearRegression. Mantendo os valores do atributo coef* fixos e variando o valor de intercept *entre intercept* - delta e intercept_ + delta.

In [14]:

```
import numpy as np

def plot(delta):
  erros = []
  intercept = modelo.intercept_
  ws = np.linspace(intercept-delta, intercept+delta, 100)
  for w in ws:
    modelo.intercept_= w
    ypred = modelo.predict(X)
    erro = mean_squared_error(y, ypred)
    erros.append(erro)
  modelo.intercept_ = intercept
  plt.plot(ws, erros)
  plt.show()

plot(1)
```