import pandas as pd

import numpy as np

from mpl\_toolkits.mplot3d import Axes3D

import matplotlib.pyplot as plt

# x: Number of floors of a building

# Y: Annual Electric bill (USD)

# assume as linear regrassion

x\_train=np.array([10,20,40,50,60,70])

y\_train=np.array([24,52,75,101,123,147])

x\_test=np.array([])

y\_test=np.array([])

print("x\_train :",x\_train)

print("y\_train :",y\_train)

import random

# Generate and store the random values in an array

w = [random.randint(-75, 75) for i in range(1000)]

b = [random.randint(-75, 75) for i in range(1000)]

#print(w,"/n",b)

plt.scatter(w, b,c='b')

plt.title("Parameters")

plt.ylabel('w')

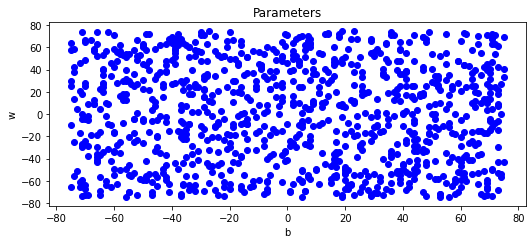
plt.xlabel('b')

#Output

**#x\_train : [10 20 40 50 60 70]**

**#y\_train : [ 24 52 75 101 123 147]**

**#Text(0.5, 0, 'b')**



# Plot the data points

plt.scatter(x\_train, y\_train,c='b')

plt.title("Parameters")

plt.ylabel('w')

plt.xlabel('b')

# defined Y^ (predcition value) for given w,b

def compute\_model\_prediction(x, w, b):

    m = x.shape[0]

    f\_wb = np.zeros(m)

    for i in range(m):

        f\_wb[i] = w \* x[i] + b

    return f\_wb

# define cost funtion

def compute\_cost(x, y, w, b):

    m = x.shape[0]

    cost\_sum = 0

    for i in range(m):

        f\_wb = w \* x[i] + b

        cost = (f\_wb - y[i]) \*\* 2

        cost\_sum = cost\_sum + cost

    total\_cost = (1 / (2 \* m)) \* cost\_sum

    return total\_cost

    # let b=0 (when line cutting the root) so the J(w,b) only depends on w

# value of W for each x

temp\_w=[]

for i in w:

  temp\_w.append(compute\_model\_prediction(x\_train, i, 0))

#print(temp\_w)

# Plot the data points

plt.scatter(x\_train, y\_train, marker='x', c='b',label='Actual Values')

plt.scatter(x\_train, y\_train,c='b')

plt.title("Electric charge")

plt.ylabel('Electric bill(USD)')

plt.xlabel('Number of floors')

# plot the prediction

for i in range(len(temp\_w)):

  plt.plot(x\_train, temp\_w[i], c='r',label='Prediction')

plt.show()

# cost funtion for each w

temp\_jwb=[]

for i in w:

  temp\_jwb.append(compute\_cost(x\_train, y\_train, i, 0))

#print(temp\_jwb)

#print(w)

#print(b)

#print(temp\_jwb)

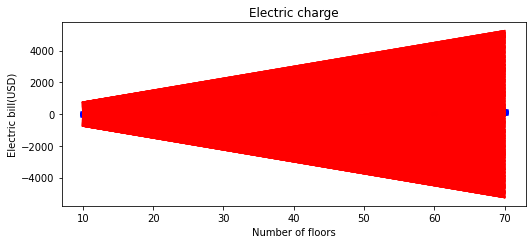
# Plot j(w) vs w

plt.plot(w,temp\_jwb,c='b')

plt.title("cost funtion")

plt.ylabel('j(w)')

plt.xlabel('w')



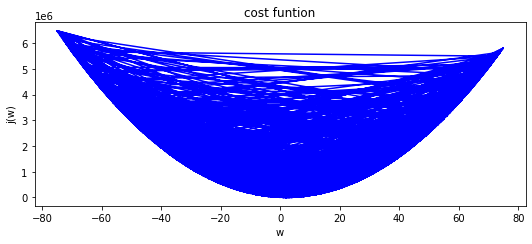


fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d')

ax.scatter(w, b, temp\_jwb , c='r', marker='o')

# Adding labels to the x, y, and z axis

ax.set\_xlabel('w')

ax.set\_ylabel('b')

ax.set\_zlabel('temp\_jwb')

plt.show()

import matplotlib.pyplot as plt

plt.rcParams["figure.figsize"] = [7.50, 3.50]

plt.rcParams["figure.autolayout"] = True

plt.plot(w, b, 'r\*')

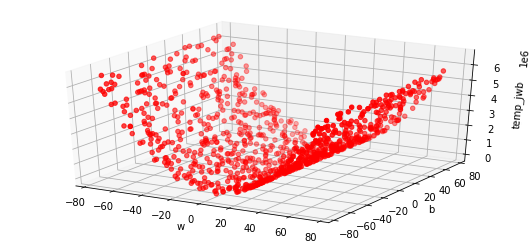
plt.axis([0, 6, 0, 20])

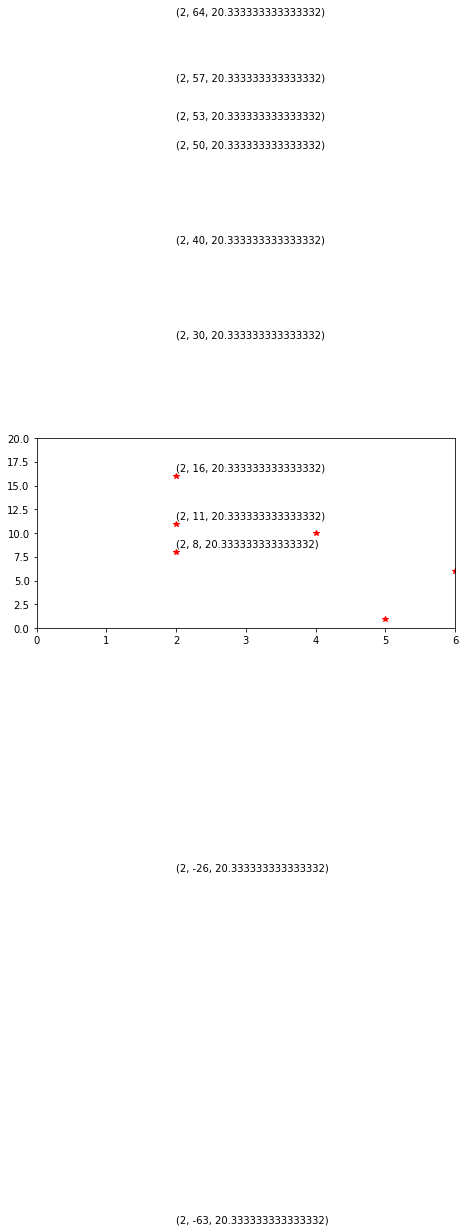
for i, j,k in zip(w, b,temp\_jwb):

  if k==min(temp\_jwb):

    plt.text(i, j+0.5, '({}, {}, {})'.format(i, j,k))

plt.show()





bOf\_fwb=[]

for i,j,k in zip(w, b,temp\_jwb):

  if k==min(temp\_jwb):

    bOf\_fwb.append(j)

print(bOf\_fwb)

#test accuracy for each b values

final\_w=2

cost=[]

for b in bOf\_fwb:

  cost.append(compute\_cost(x\_train, y\_train, final\_w, b))

print('cost:',cost)

#min cost => good accuracy

final\_b=bOf\_fwb[cost.index(min(cost))]

print("Perameter b :",final\_b)

#our model is

#Output

**#[-63, 16, -26, 64, 11, 57, 50, 53, 40, 8, 30]**

**#cost: [2235.833333333333, 89.66666666666666, 453.66666666666663, 1833.6666666666665, 40.5, 1435.8333333333333, 1087.0, 1230.5, 673.6666666666666, 23.0, 360.3333333333333]**

**#Perameter b : 8**

def model(x):

    y=final\_w\*x+final\_b

    #print("Y :",y)

    return y

# if x=80 then y=210

y=model(80)

print('Y:',y)

#Output

**#Y: 168**