Modelo lineal

Este **notebook** contiene algunas pruebas de modelos lienales

1. Generar datos
2. Vamos a ajustar (varias) recatas
3. Vamos a hacer modelos aleatorios.

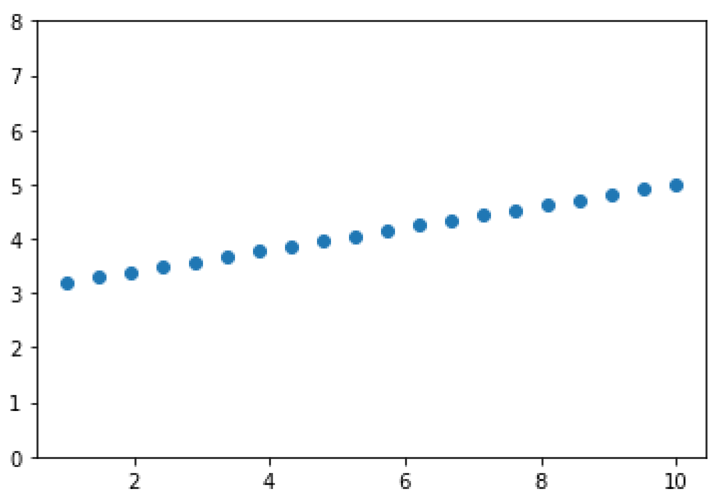
Preparar notebook

In [1]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
# crear
x = np.linspace(1, 10, 20)

# pendiente
m = 0.2
# ordenada al origen
b = 3

y_real = m*x + b
```

In [3]:
```
plt.scatter(x, y_real)
plt.ylim(0,8)
```

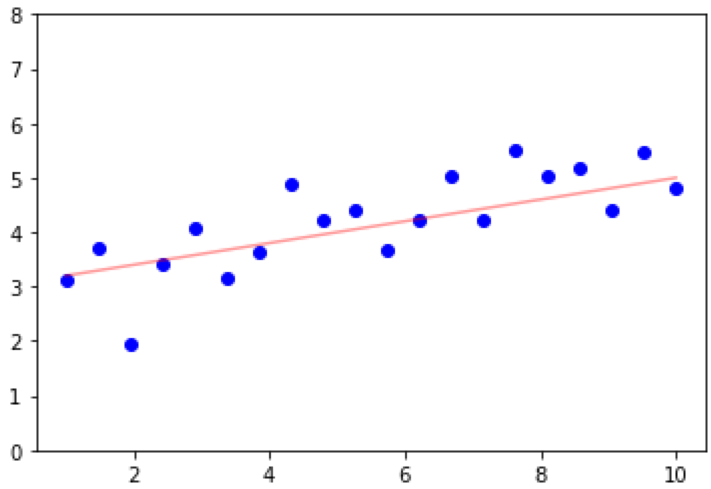
Out[3]: (0.0, 8.0)



In [4]:
```
y = y_real + np.random.normal(0, 0.8, size=len(y_real))
#media de cero y desviaación estandar de .8
```

In [5]:
```
plt.scatter(x, y, c='b')
plt.plot(x, y_real, c='r', alpha=0.4)
plt.ylim(0,8)
```

Out[5]:
```
(0.0, 8.0)
```



Definición de medidas de error

In [6]:
```
def msd(y, y_pred):
    """msd = mean signed deviation - promedio de la diferencia absoluta"""
    error = y_pred - y
    return error.mean()

def mse(y, y_pred):
    "Mean square error"
    error = (y_pred - y) **2
    return error.mean()

def sse(y, y_pred):
    "Sum square error"
    error = (y_pred - y) ** 2
    return error.sum()

def rmse(y, y_pred):
    error = (y_pred - y) ** 2
    mse = error.mean()
    return mse**0.5
```

In [7]:
```
print("Medidas de error contra el modelo inicial")
print(f"msd: {msd(y, y_real)}")
print(f'mse: {mse(y, y_real)}')
print(f"sse: {sse(y, y_real)}")
print(f"rmse: {rmse(y, y_real)}")
```

```
Medidas de error contra el modelo inicial
msd: -0.11051721510601782
mse: 0.33359218671499485
sse: 6.671843734299897
rmse: 0.5775743992898187
```

Crear modelos aleatorios.

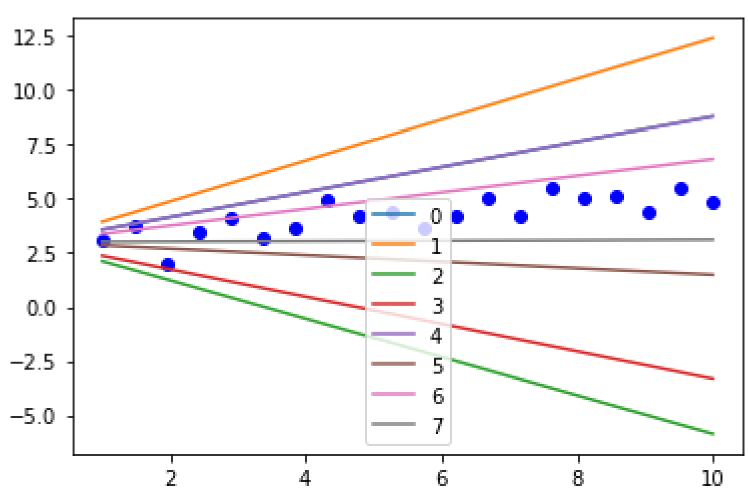
Mantener fijo el valor de la ordenada al origen b = 3

Crear valores aleatorios de pendinte.

In [8]:
```
b = 3

plt.scatter(x, y, c='b')

#listas de errores
listamsd = []
listamse = []

# lista de pendientes
ms = []

for i in range(8):
    mi = np.random.uniform(-1, 1)
    y_pred = mi * x + b

    meansign = msd(y, y_pred)
    sqrerror = mse(y, y_pred)

    # guardar en las lsitas
    listamsd.append(meansign)
    listamse.append(sqrerror)
    ms.append(mi)

    plt.plot(x, y_pred, label=i)
    print(f"m = {mi}, msd = {meansign}, mse = {sqrerror}")

plt.legend()
```

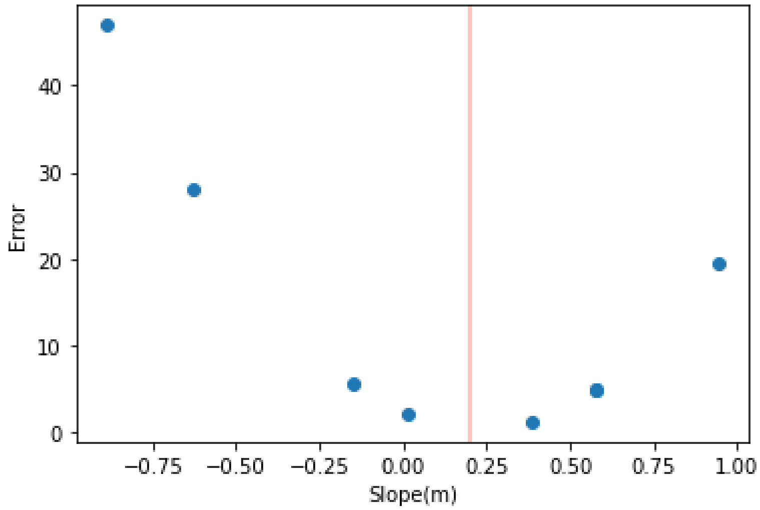
```
m = 0.5781731715267591, msd = 1.969435228291157, mse = 4.982960024227588
m = 0.9407122781605031, msd = 3.9634003147767487, mse = 19.566796175396913
m = -0.8860108201642862, msd = -6.083576726009591, mse = 46.94611746773654
m = -0.6315980841889863, msd = -4.684306678145442, mse = 28.0481186598776
m = 0.5789843677662405, msd = 1.9738968076083048, mse = 5.004526473303801
m = -0.15112525683260514, msd = -2.041706127685346, mse = 5.673485776475851
m = 0.3829313471146014, msd = 0.8956051940242895, mse = 1.23573836259683
m = 0.010970286543015684, msd = -1.1501806391194316, mse = 2.052862180315083
```

Out[8]:
```
<matplotlib.legend.Legend at 0x7f90d5b872b0>
```



In [9]:
```
plt.scatter(ms, listamse)
plt.xlabel('Slope(m)')
plt.ylabel('Error')
plt.axvline(0.2, color='r', alpha=0.3)
```

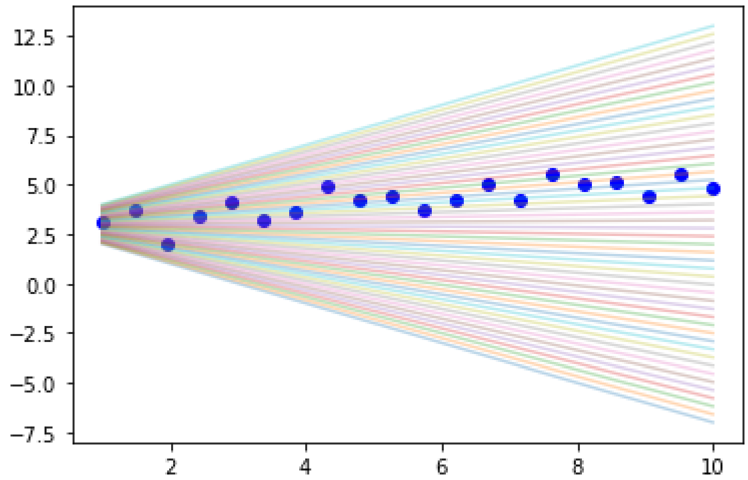
Out[9]:
```
<matplotlib.lines.Line2D at 0x7f90d5caceb0>
```



Búsqueda gradual de la pendiente.

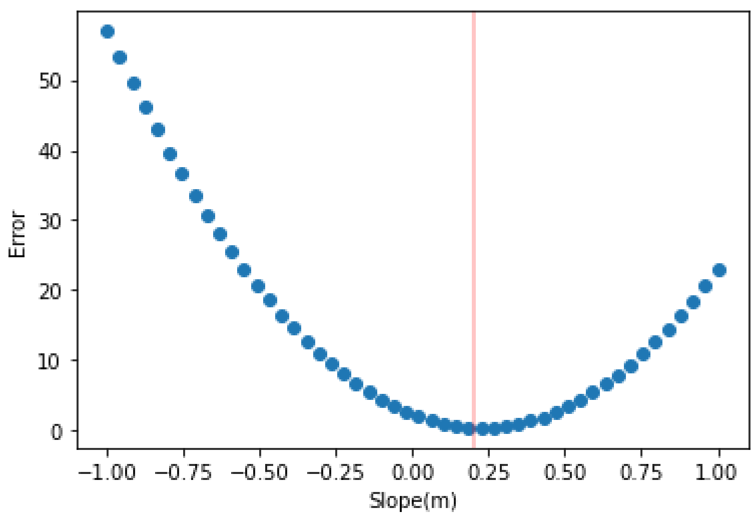
In [10]:
```
b = 3

plt.scatter(x, y, c='b')

#listas de errores
listamsd = []
listamse = []

# lista de pendientes
ms = []

for mi in np.linspace(-1, 1, 50):
    y_pred = mi * x + b

    meansign = msd(y, y_pred)
    sqrerror = mse(y, y_pred)

    # guardar en las lsitas
    listamsd.append(meansign)
    listamse.append(sqrerror)
    ms.append(mi)

    # Plot
    plt.plot(x, y_pred, alpha=0.3)
    print(f"m = {mi}, msd = {meansign}, mse = {sqrerror}")
```

```
m = -1.0, msd = -6.710517215106019, mse = 56.99694218138283
m = -0.9591836734693877, msd = -6.48602741918765, mse = 53.28539586497095
m = -0.9183673469387755, msd = -6.261537623269282, mse = 49.699498949028204
m = -0.8775510204081632, msd = -6.037047827350915, mse = 46.239251433554536
m = -0.8367346938775511, msd = -5.812558031432548, mse = 42.90465331854998
m = -0.7959183673469388, msd = -5.58806823551418, mse = 39.69570460401452
m = -0.7551020408163265, msd = -5.3635784395958135, mse = 36.612405289948164
m = -0.7142857142857143, msd = -5.139088643677446, mse = 33.65475537635093
m = -0.6734693877551021, msd = -4.914598847759079, mse = 30.82275486322278
m = -0.6326530612244898, msd = -4.690109051840713, mse = 28.116403750563734
m = -0.5918367346938775, msd = -4.465619255922343, mse = 25.535702038373792
m = -0.5510204081632654, msd = -4.241129460003977, mse = 23.080649726652958
m = -0.5102040816326531, msd = -4.01663966408561, mse = 20.75124681540122
m = -0.4693877551020409, msd = -3.7921498681672423, mse = 18.54749330461859
m = -0.4285714285714286, msd = -3.567660072248875, mse = 16.46938919430506
m = -0.3877551020408164, msd = -3.3431702763305085, mse = 14.51693448446063
m = -0.34693877551020413, msd = -3.1186804804121406, mse = 12.690129175085307
m = -0.30612244897959184, msd = -2.894190684493773, mse = 10.988973266179084
m = -0.26530612244897966, msd = -2.669700888575406, mse = 9.413466757741968
m = -0.22448979591836737, msd = -2.445211092657039, mse = 7.963609649773948
m = -0.1836734693877552, msd = -2.220721296738671, mse = 6.639401942275039
m = -0.1428571428571429, msd = -1.9962315008203035, mse = 5.440843635245227
m = -0.10204081632653073, msd = -1.7717417049019368, mse = 4.367934728684519
m = -0.061224489795918435, msd = -1.5472519089835692, mse = 3.4206752225929145
m = -0.020408163265306145, msd = -1.3227621130652016, mse = 2.599065116970411
m = 0.020408163265306145, msd = -1.0982723171468343, mse = 1.9031044118170108
m = 0.06122448979591821, msd = -0.8737825212284676, mse = 1.3327931071327164
m = 0.1020408163265305, msd = -0.6492927253101002, mse = 0.8881312029175218
m = 0.1428571428571428, msd = -0.4248029293917324, mse = 0.5691186991714302
m = 0.18367346938775508, msd = -0.20031313347336493, mse = 0.3757555958944415
m = 0.22448979591836715, msd = 0.024176662445001516, mse = 0.30804189308655594
m = 0.26530612244897944, msd = 0.24866645836336915, mse = 0.3659775907477729
m = 0.30612244897959173, msd = 0.47315625428173663, mse = 0.5495626888780928
m = 0.346938775510204, msd = 0.6976460502001041, mse = 0.858797187477516
m = 0.3877551020408163, msd = 0.9221358461184715, mse = 1.2936810865460413
m = 0.4285714285714284, msd = 1.1466256420368384, mse = 1.854214386083667
m = 0.46938775510204067, msd = 1.3711154379552055, mse = 2.5403970860903984
m = 0.510204081632653, msd = 1.5956052338735733, mse = 3.352229186566233
m = 0.5510204081632653, msd = 1.820095029791941, mse = 4.289710687511171
m = 0.5918367346938773, msd = 2.044584825710307, mse = 5.352841588925204
m = 0.6326530612244896, msd = 2.2690746216286746, mse = 6.541621890808349
m = 0.6734693877551019, msd = 2.4935644175470424, mse = 7.856051593160593
m = 0.7142857142857142, msd = 2.71805421346541, mse = 9.296130695981944
m = 0.7551020408163265, msd = 2.9425440093837776, mse = 10.861859199272397
m = 0.7959183673469385, msd = 3.1670338053021436, mse = 12.55323710303194
m = 0.8367346938775508, msd = 3.3915236012205114, mse = 14.370264407260597
m = 0.8775510204081631, msd = 3.6160133971388797, mse = 16.31294111195836
m = 0.9183673469387754, msd = 3.840503193057246, mse = 18.381267217125224
m = 0.9591836734693877, msd = 4.064992988975614, mse = 20.57524272276119
m = 1.0, msd = 4.289482784893982, mse = 22.89486762886626
```



In [11]:
```
plt.scatter(ms, listamse)
plt.xlabel('Slope(m)')
plt.ylabel('Error')
plt.axvline(0.2, color='r', alpha=0.3)
```

Out[11]: <matplotlib.lines.Line2D at 0x7f90d5f2bd00>



Búsqueda gradual de la pendiente y la ordenada al origen.

In [12]:
```
#
listamse = []
# pendientes y ordenadas
ms = []
bs = []

for mi in np.linspace(-1, 1, 50):
    for bi in np.linspace(0, 6, 50):
        ms.append(mi)
        bs.append(bi)
        y_pred = mi * x + bi
        error = mse(y, y_pred)
        listamse.append(error)
```

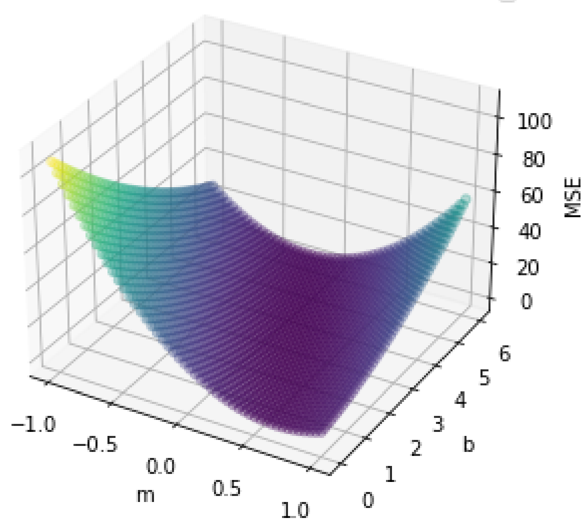
```
minindex = np.argmin(listamse)
minmse = listamse[minindex]
minm = ms[minindex]
minb = bs[minindex]

print(f"Error mínimo = {minmse}")
print(f"m = {minm, minb}")
```

```
Error mínimo = 0.30744377409048324
m = (0.26530612244897944, 2.693877551020408)
```

In [13]:

```
from mpl_toolkits import mplot3d
ax = plt.axes(projection='3d')
ax.scatter3D(ms, bs, listamse, c=listamse, alpha=0.3)
# ax.plot([0.2]*2, [3]*2, [0, 100], c='g', label='Real parameters')
# ax.plot([minm]*2, [minb]*2, [0, 100], c='r', label='Minimum parameters')

ax.legend()

ax.set_xlabel('m')
ax.set_ylabel('b')
ax.set_zlabel('MSE', rotation=90)

plt.tight_layout()
```

```
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argument.
```



Busquemos los valores de m usando la información del error.

In [14]:

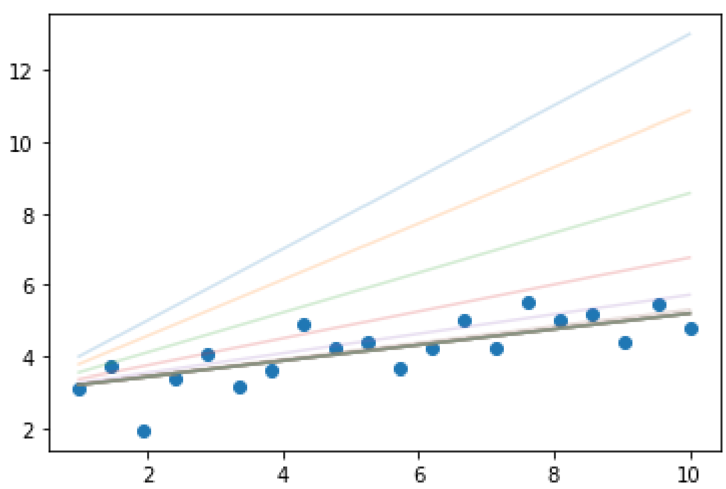
```
#Agregamos la constante de aprendizaje. Aquí se actualizará el valor de una sol
```

In [15]:

```
b = 3
eta = 0.05

# listas
listamsd = [] # errores absolutos
listamse = [] # error cuadratico medio
ms = []    # lista de pendientes
```

```
# valor inicial de m
mi = 1

plt.scatter(x, y)

for i in range(len(x)):
    # actualizar mi
    y_pred = mi * x + b
    error = msd(y, y_pred)
    serror = mse(y, y_pred)

    # Actualizar la pendiente
    mi = mi - error *x[i]*eta  #SE PARECE A LA REGLA DEL PERCEPTRON. Modificar

    listamsd.append(error)
    listamse.append(serror)
    ms.append(mi)


    plt.plot(x, y_pred, alpha=0.2)
    print(f"i={i}, msd={error}, mse={serror}")
```

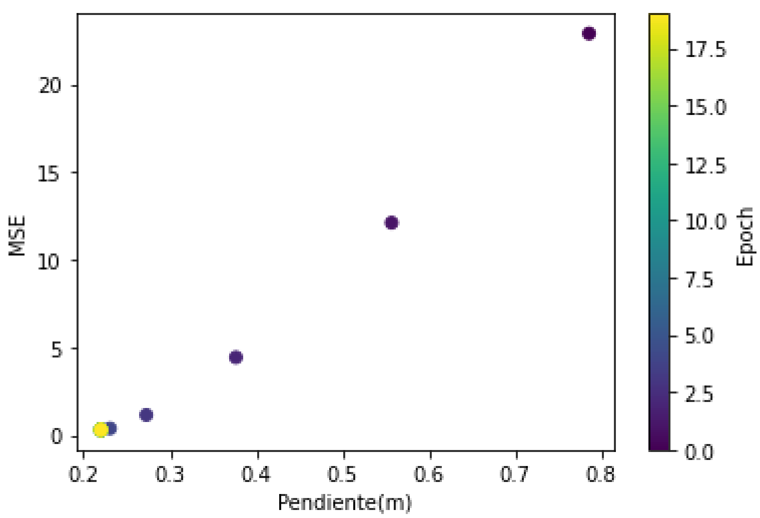
```
i=0, msd=4.289482784893982, mse=22.89486762886626
i=1, msd=3.109875019048137, mse=12.110661225939603
i=2, msd=1.849557248170734, mse=4.422073766315968
i=3, msd=0.8590706692161435, mse=1.1588195669727746
i=4, msd=0.2871104605011849, mse=0.3885003203464308
i=5, msd=0.05855542286537348, mse=0.3087665724134662
i=6, msd=0.004314610105869687, mse=0.30896625103543396
i=7, msd=-0.00024411609809531497, mse=0.30931719337602653
i=8, msd=4.561116569685719e-05, mse=0.3092933475469093
i=9, msd=-1.4463540701203926e-05, mse=0.3092982747608001
i=10, msd=6.470531366331578e-06, mse=0.30929655676631584
i=11, msd=-3.737583249852072e-06, mse=0.30929739437818327
i=12, msd=2.6458155110820057e-06, mse=0.30929687056732774
i=13, msd=-2.2176111585947567e-06, mse=0.3092972696426407
i=14, msd=2.147581332478943e-06, mse=0.3092969114479224
i=15, msd=-2.359513700711524e-06, mse=0.30929728128753886
i=16, msd=2.899718153170916e-06, mse=0.30929684973460914
i=17, msd=-3.941327437040343e-06, mse=0.3092974110987583
i=18, msd=5.870503498228707e-06, mse=0.3092966059935353
i=19, msd=-9.508670797764296e-06, mse=0.30929786803126535
```



De los 20 modelos lineales. Se modifico la pendiente con pasos grandes y luego con oasos más pequeños. Conforme nos acercamos a lo correcto.

In [16]:
```
plt.scatter(ms, listamse, c=range(len(listamse)))
plt.colorbar(label='Epoch')
plt.xlabel('Pendiente(m)')
plt.ylabel('MSE')
```

Out[16]:
```
Text(0, 0.5, 'MSE')
```



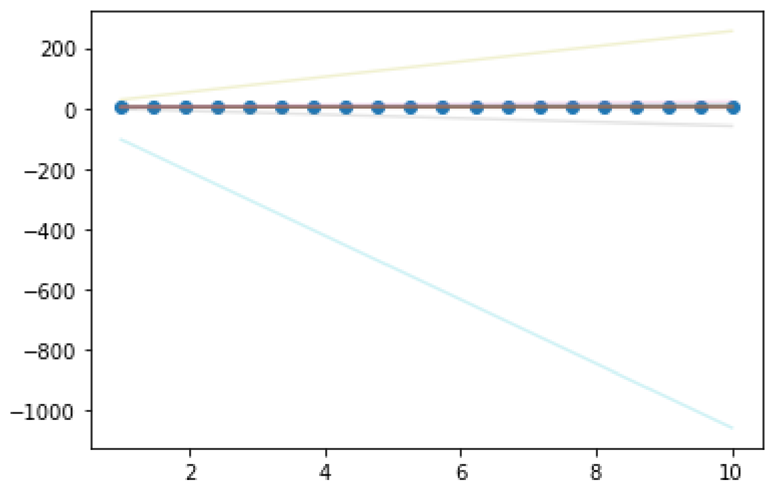
EERCICIO: Haz 3 gráficas como la anterior cambiando la constatne de aprendizaje (por ejemplo, *eta* = [0.1, 0.04, 0.001])

Ejercicio 1

In [17]:
```
b2 = 3
eta2 = 0.1

# listas
listamsd2 = [] # errores absolutos
listamse2 = [] # error cuadratico medio
ms2 = []   # lista de pendientes

# valor inicial de m
mi2 = 1

plt.scatter(x, y)

for i in range(len(x)):
    # actualizar mi
    y_pred2 = mi2 * x + b2
    error2 = msd(y, y_pred2)
    serror2 = mse(y, y_pred2)
```

```
    # Actualizar la pendiente
    mi2 = mi2 - error2 *x[i]*eta2  #SE PARECE A LA REGLA DEL PERCEPTRON. Modifi

    listamsd2.append(error2)
    listamse2.append(serror2)
    ms2.append(mi2)


    plt.plot(x, y_pred2, alpha=0.2)
    print(f"i={i}, msd={error2}, mse={serror2}")
```

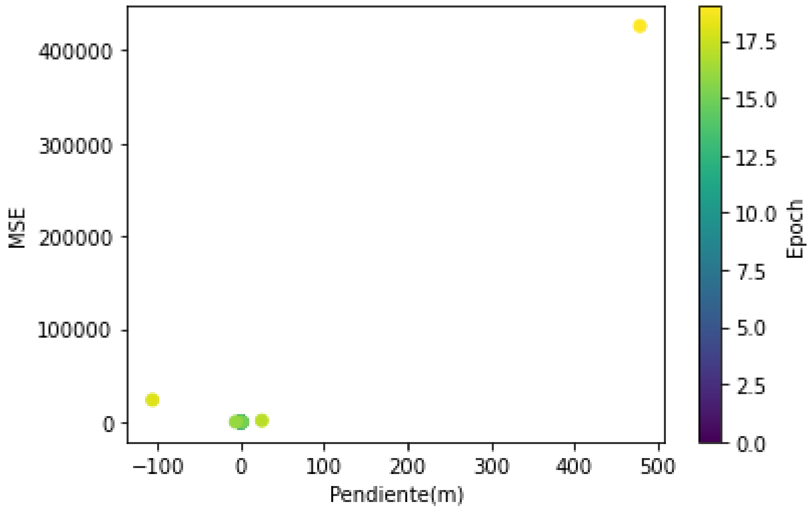
```
i=0, msd=4.289482784893982, mse=22.89486762886626
i=1, msd=1.930267253202291, mse=4.795759619224174
i=2, msd=0.3657348479751713, mse=0.4460375632978895
i=3, msd=-0.02598642340876207, mse=0.31227129536049536
i=4, msd=0.00861655091974738, mse=0.3086825957716112
i=5, msd=-0.005101905149850361, mse=0.30974818343878757
i=6, msd=0.004350045443556605, mse=0.3089637260787841
i=7, msd=-0.0048422874279591 8, mse=0.3097236615951971
i=8, msd=0.006651773782617609, mse=0.3088064219287233
i=9, msd=-0.01087039873422524, mse=0.31033638794115204
i=10, msd=0.02059654497011103, mse=0.3081358402464606
i=11, msd=-0.04439097455400 2496, mse=0.3153962306539514
i=12, msd=0.10723924905414288, mse=0.3148339160196827
i=13, msd=-0.28700609549490347, mse=0.43553587287891
i=14, msd=0.8428915857166114, mse=1.1258197616032029
i=15, msd=-2.6950349385412706, mse=9.584975540782036
i=16, msd=9.319147129587446, mse=107.80996317729804
i=17, msd=-34.65251287922909, mse=1500.1006647182012
i=18, msd=137.88052492998523, mse=23688.70738676178
i=19, msd=-584.5408570058057, mse=426006.4509714379
```



Eta 0.1

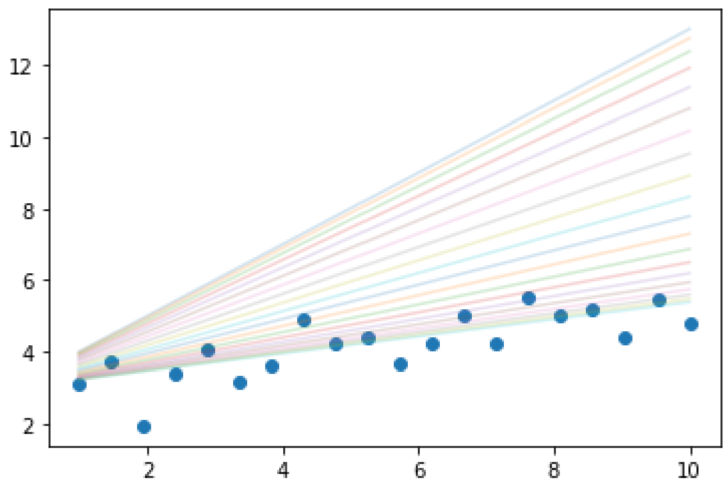
In [18]:
```
plt.scatter(ms2, listamse2, c=range(len(listamse2)))
plt.colorbar(label='Epoch')
plt.xlabel('Pendiente(m)')
plt.ylabel('MSE')
```

Out[18]:
```
Text(0, 0.5, 'MSE')
```



In [19]:
```
b3 = 3
eta3 = 0.006

# listas
listamsd3 = [] # errores absolutos
listamse3 = [] # error cuadratico medio
ms3 = []   # lista de pendientes

# valor inicial de m
mi3 = 1

plt.scatter(x, y)

for i in range(len(x)):
    # actualizar mi
    y_pred3 = mi3 * x + b3
    error3 = msd(y, y_pred3)
    serror3 = mse(y, y_pred3)

    # Actualizar la pendiente
    mi3 = mi3 - error3 *x[i]*eta3  #SE PARECE A LA REGLA DEL PERCEPTRON. Modifi

    listamsd3.append(error3)
    listamse3.append(serror3)
    ms3.append(mi3)


    plt.plot(x, y_pred3, alpha=0.2)
    print(f"i={i}, msd={error3}, mse={serror3}")
```

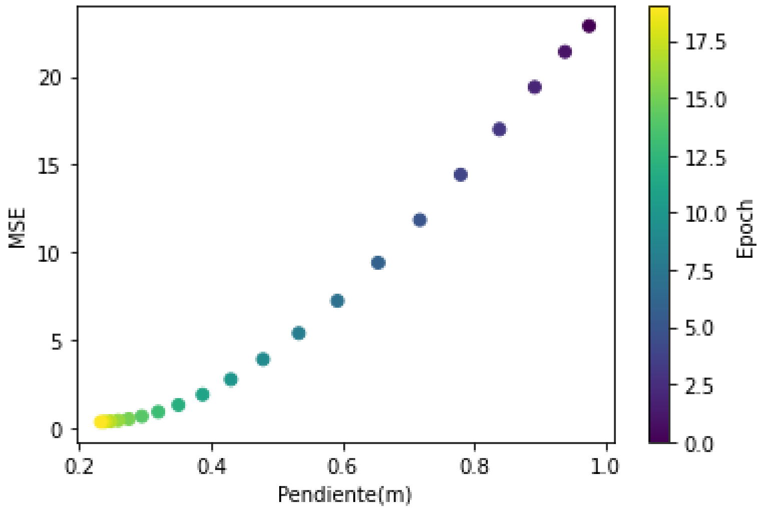
```
i=0, msd=4.289482784893982, mse=22.89486762886626
i=1, msd=4.1479298529924815, mse=21.41758356727511
i=2, msd=3.9462094748785304, mse=19.39869912729943
i=3, msd=3.6926135923087045, mse=17.004571106100084
i=4, msd=3.3975932010916194, mse=14.42113647173544
i=5, msd=3.073033640039973, mse=11.829718591765873
i=6, msd=2.7314416901576344, mse=9.38597997630022
i=7, msd=2.3851236358634385, mse=7.20542241888364З
i=8, msd=2.0454318169873087, mse=5.3570906699833944
i=9, msd=1.7221459355971565, mse=3.865218583452953
i=10, msd=1.4230363783618611, mse=2.716991297767856
i=11, msd=1.1536331229425127, mse=1.8737338101501997
i=12, msd=0.9171990502720844, mse=1.2827662691888086
i=13, msd=0.7148842492357523, mse=0.8877374388309505
i=14, msd=0.5460210644689075, mse=0.6361609878447395
i=15, msd=0.4085099700750273, mse=0.4838139452597378
i=16, msd=0.2992443033423279, mse=0.3963740910308575
i=17, msd=0.21452666609609716, mse=0.34906565849426086
i=18, msd=0.15043964731917908, mse=0.32516628648088475
i=19, msd=0.1031461729277339, mse=0.3140962797597864
```



Eta 0.006

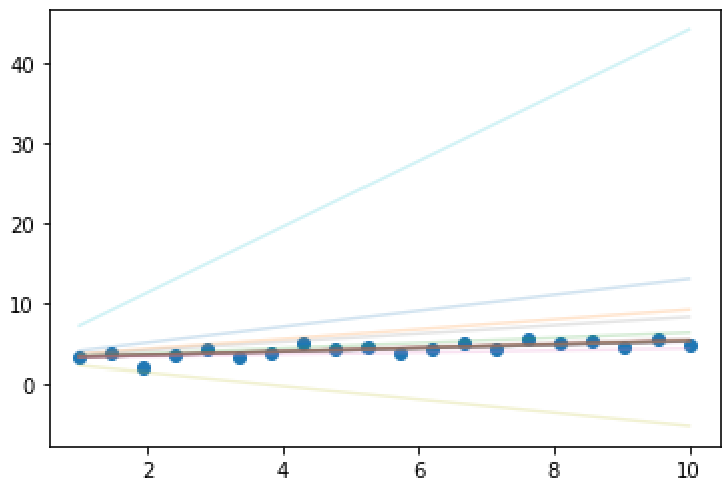
In [20]:
```
plt.scatter(ms3, listamse3, c=range(len(listamse3)))
plt.colorbar(label='Epoch')
plt.xlabel('Pendiente(m)')
plt.ylabel('MSE')
```

Out[20]:
```
Text(0, 0.5, 'MSE')
```



In [21]:
```
b4 = 3
eta4 = 0.09

# listas
listamsd4 = [] # errores absolutos
listamse4 = [] # error cuadratico medio
ms4 = []   # lista de pendientes

# valor inicial de m
mi4 = 1

plt.scatter(x, y)

for i in range(len(x)):
    # actualizar mi
    y_pred4 = mi4 * x + b4
    error4= msd(y, y_pred4)
    serror4 = mse(y, y_pred4)

    # Actualizar la pendiente
    mi4 = mi4 - error4 *x[i]*eta4  #SE PARECE A LA REGLA DEL PERCEPTRON. Modifi

    listamsd4.append(error4)
    listamse4.append(serror4)
    ms4.append(mi4)


    plt.plot(x, y_pred4, alpha=0.2)
    print(f"i={i}, msd={error4}, mse={serror4}")
```

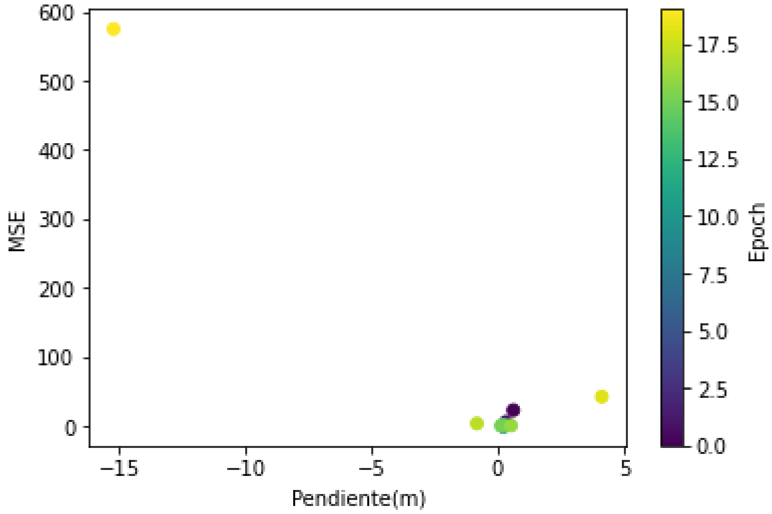
```
i=0, msd=4.289482784893982, mse=22.89486762886626
i=1, msd=2.1661888063714607, mse=5.981195556870366
i=2, msd=0.5860110770920692, mse=0.6893143978274187
i=3, msd=0.021127241463582403, mse=0.3081198966065982
i=4, msd=-0.004192089490405504, mse=0.309662985477544
i=5, msd=0.0018147334767676382, mse=0.3091522816608736
i=6, msd=-0.0012110958150218121, mse=0.3093982949793227
i=7, msd=0.0010922171994945718, mse=0.3092089508167719
i=8, msd=-0.0012411036493202633, mse=0.30940084906380994
i=9, msd=0.0017012918182391923, mse=0.30916109310111856
i=10, msd=-0.0027310210766647222, mse=0.30953048485046886
i=11, msd=0.005024360091273916, mse=0.3089162741967761
i=12, msd=-0.010421580589321456, mse=0.3102876462286849
i=13, msd=0.024060138555294097, mse=0.30804444770344086
i=14, msd=-0.061188731315358534, mse=0.31898550241038803
i=15, msd=0.16995975237726565, mse=0.33136123062218603
i=16, msd=-0.5119366646605743, mse=0.678020502590608
i=17, msd=1.662042797873027, mse=3.6165856799891274
i=18, msd=-5.785658455332736, mse=42.51351385606765
i=19, msd=21.49676624495865, mse=574.6262466589214
```



Eta 0.09

In [38]:
```python
plt.scatter(ms4, listamse4, c=range(len(listamse4)))
plt.colorbar(label='Epoch')
plt.xlabel('Pendiente(m)')
plt.ylabel('MSE')
```

Out[38]:
```
Text(0, 0.5, 'MSE')
```



RESPUESTA EJERCICIO 1

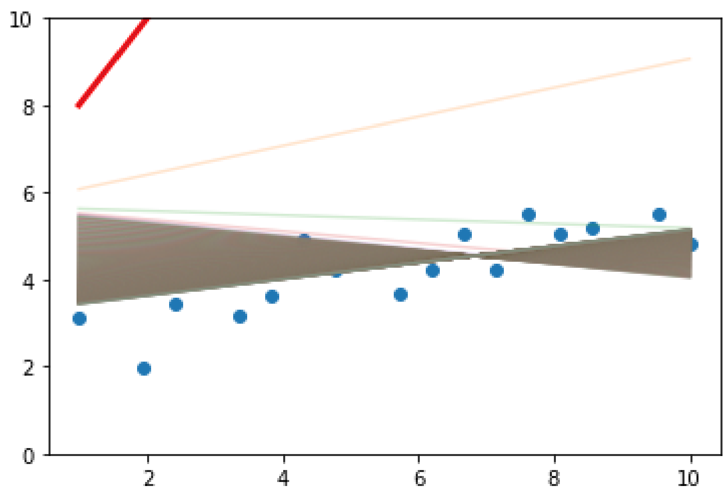
Al cambiar la constante de aprendizaje eta, vemos como se está comportanto el MSE al pasar las épocas. A valores pequeños de eta, tendremos una disminución de error conforme avanzan las épocas.

Encontrar los valores de la pendiente y la ordenada al origen usando Gradient Descent.

In [23]:
```
eta = 0.01

# listas
listamsd = [] # msd
listamse = [] # mse
ms = []
bs = []


# inicializar los valores
mi = 2
bi = 6

plt.scatter(x, y)
plt.plot(x, mi*x + bi, linewidth=3, c='r')

# numero de veces que actualizamos los pesos
epoch = 500

for i in range(epoch):
    # calculuar los errores
    y_pred = mi*x + bi
    error = msd(y, y_pred)
    serror = mse(y, y_pred)
    listamsd.append(error)
    listamse.append(serror)

    ms.append(mi)
```

```
    bs.append(bi)

    plt.plot(x, y_pred, alpha=0.2)

    # actualizar los pesos
    # errores
    part_error = (y_pred - y)
    # actualizar
    mi = mi - 2 * (part_error*x).sum()/len(x) * eta
    bi = bi - 2 * part_error.sum()/len(x) * eta



plt.ylim(0,10)
```

Out[23]:
```
(0.0, 10.0)
```

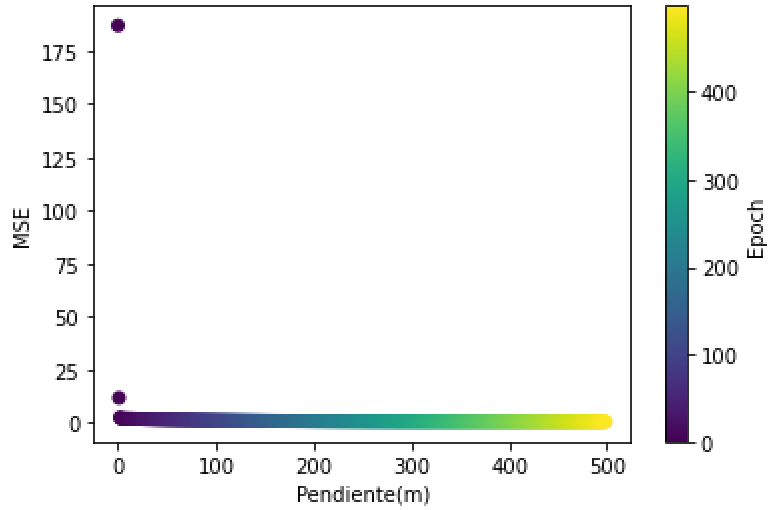


In [24]:
```
print('Pendiente = ', mi)
print('b = ', bi)
print('MSE = ', mse(y, mi*x + bi))
```

```
Pendiente =  0.19037108837652933
b =  3.2427107014037615
MSE =  0.3355805786804063
```

In [25]:
```
plt.scatter(range(epoch), listamse, c=range(len(listamse)))
plt.colorbar(label='Epoch')
plt.xlabel('Pendiente(m)')
plt.ylabel('MSE')
# plt.ylim(0, 3)
```

Out[25]:
```
Text(0, 0.5, 'MSE')
```



Ejercicio : ¿Qué pasa si corres nuevamente el algoritmo de gradient descent cambiando el parámetro epoch a los siguientes valores epoch=100 y epoch=800?

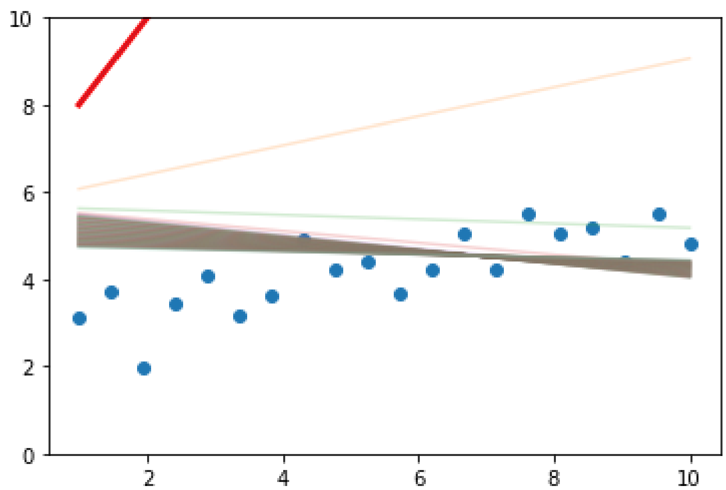
EJERCICIO 2

Epoch 100

In [29]:
```
eta = 0.01

# listas
listamsd = [] # msd
listamse = [] # mse
ms = []
bs = []


# inicializar los valores
mi = 2
bi = 6

plt.scatter(x, y)
plt.plot(x, mi*x + bi, linewidth=3, c='r')

# numero de veces que actualizamos los pesos
epoch = 100

for i in range(epoch):
    # calculuar los errores
    y_pred = mi*x + bi
    error = msd(y, y_pred)
    serror = mse(y, y_pred)
    listamsd.append(error)
    listamse.append(serror)

    ms.append(mi)
    bs.append(bi)
```

```
    plt.plot(x, y_pred, alpha=0.2)

    # actualizar los pesos
    # errores
    part_error = (y_pred - y)
    # actualizar
    mi = mi - 2 * (part_error*x).sum()/len(x) * eta
    bi = bi - 2 * part_error.sum()/len(x) * eta



plt.ylim(0,10)
```

Out[29]: (0.0, 10.0)

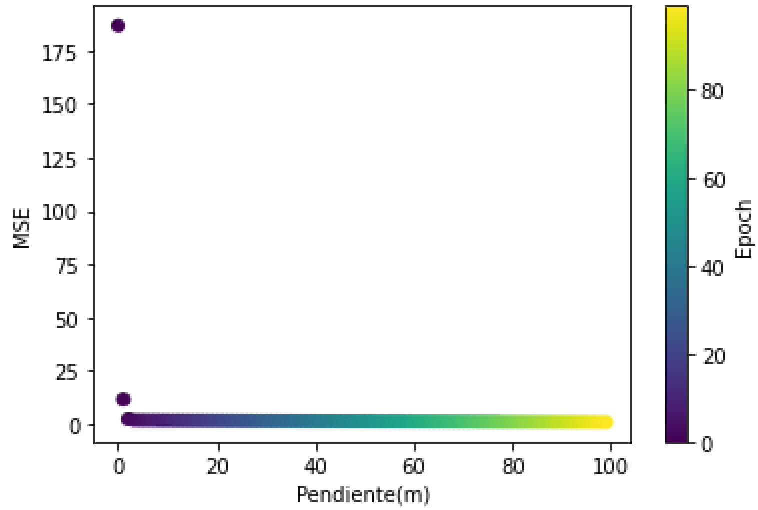


In [30]:
```
print('Pendiente = ', mi)
print('b = ', bi)
print('MSE = ', mse(y, mi*x + bi))
```

```
Pendiente =  -0.032931985229938385
b =  4.765915009448969
MSE =  1.0412095778753192
```

In [32]:
```
plt.scatter(range(epoch), listamse, c=range(len(listamse)))
plt.colorbar(label='Epoch')
plt.xlabel('Pendiente(m)')
plt.ylabel('MSE')
# plt.ylim(0, 3)
```

Out[32]: Text(0, 0.5, 'MSE')

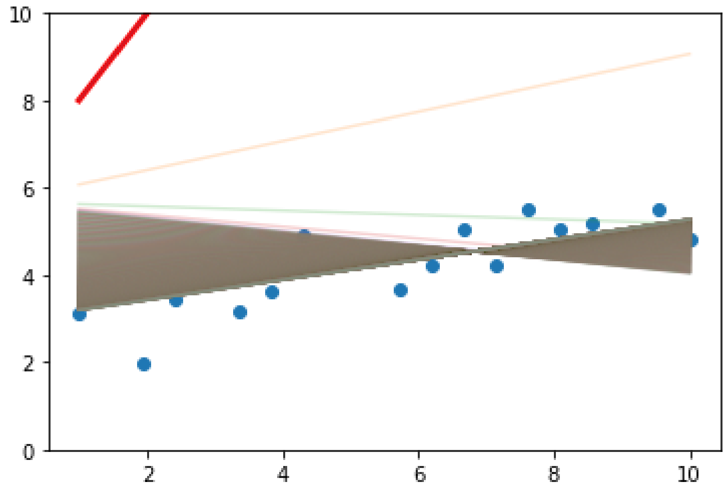


Epoch 800

In [34]:
```
eta = 0.01

# listas
listamsd = [] # msd
listamse = [] # mse
ms = []
bs = []


# inicializar los valores
mi = 2
bi = 6

plt.scatter(x, y)
plt.plot(x, mi*x + bi, linewidth=3, c='r')

# numero de veces que actualizamos los pesos
epoch = 800

for i in range(epoch):
    # calcular los errores
    y_pred = mi*x + bi
    error = msd(y, y_pred)
    serror = mse(y, y_pred)
    listamsd.append(error)
    listamse.append(serror)

    ms.append(mi)
    bs.append(bi)

    plt.plot(x, y_pred, alpha=0.2)

    # actualizar los pesos
    # errores
    part_error = (y_pred - y)
    # actualizar
    mi = mi - 2 * (part_error*x).sum()/len(x) * eta
    bi = bi - 2 * part_error.sum()/len(x) * eta
```

```python
plt.ylim(0,10)
```

Out[34]: (0.0, 10.0)

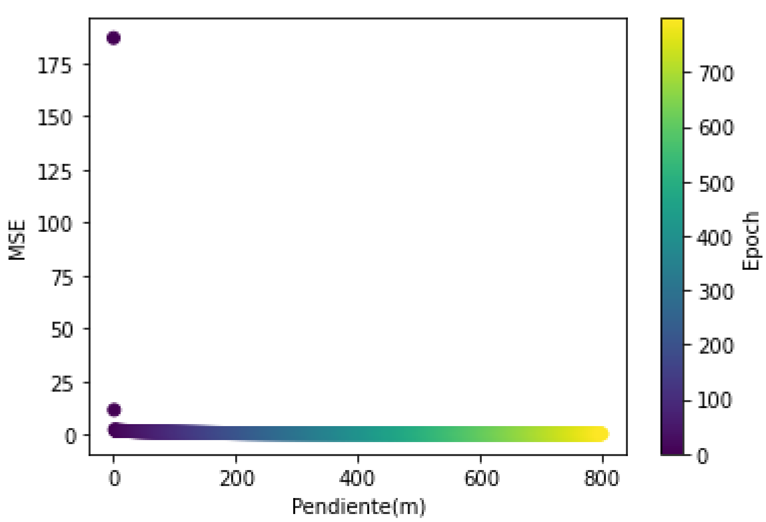


In [35]:
```python
print('Pendiente = ', mi)
print('b = ', bi)
print('MSE = ', mse(y, mi*x + bi))
```

```
Pendiente =  0.2316241741232314
b =  2.9613134196078383
MSE =  0.30569674967137395
```

In [39]:
```python
plt.scatter(range(epoch), listamse, c=range(len(listamse)))
plt.colorbar(label='Epoch')
plt.xlabel('Pendiente(m)')
plt.ylabel('MSE')
# plt.ylim(0, 3)
```

Out[39]: Text(0, 0.5, 'MSE')



Respuesta ejercicio 2

Al hacer las comparaciones de las tres épocas diferentes vemos que conforme avanzan las épocas la pendiente se va alejando del cero. MSE igual dismunuye al paso de las épocas.

Ecuación normal.

$(X^T X)^{-1} X^T y$

In [ ]:
```
xnorm = x.reshape(-1, 1)
ynorm = y.reshape(-1, 1)
```

In [ ]:
```
xnorm = np.c_[xnorm, np.ones(len(xnorm))]
```

In [ ]:
```
W =  np.linalg.inv(xnorm.T.dot(xnorm)).dot(xnorm.T.dot(y))
W
```

In [ ]:
```
mnorm = W[0]
bnorm = W[1]
```

In [ ]:
```
plt.scatter(x, y)
plt.plot(x, mnorm*x + bnorm, c='r')
```

In [ ]:
```
print('MSE ', mse(y, x*mnorm + bnorm))
print('GD MSE ', mse(y, x*mi + bi))
```

In [ ]:
```

```