Week 1 hello world

September 9, 2018

In [1]:
```
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pylab as plt
import os
```

In [2]:
```
data = pd.DataFrame({'weight':[150, 100, 110, 122, 80, 90, 100, 180, 200, 100, 200, 120,
                               'age'   :[ 17,   9,  13,  12,  6,  7,  12,  16,  17,  10,  20,  15,
data
```

Out[2]:

	weight	age
0	150	17
1	100	9
2	110	13
3	122	12
4	80	6
5	90	7
6	100	12
7	180	16
8	200	17
9	100	10
10	200	20
11	120	15
12	80	8
13	10	2
14	200	20

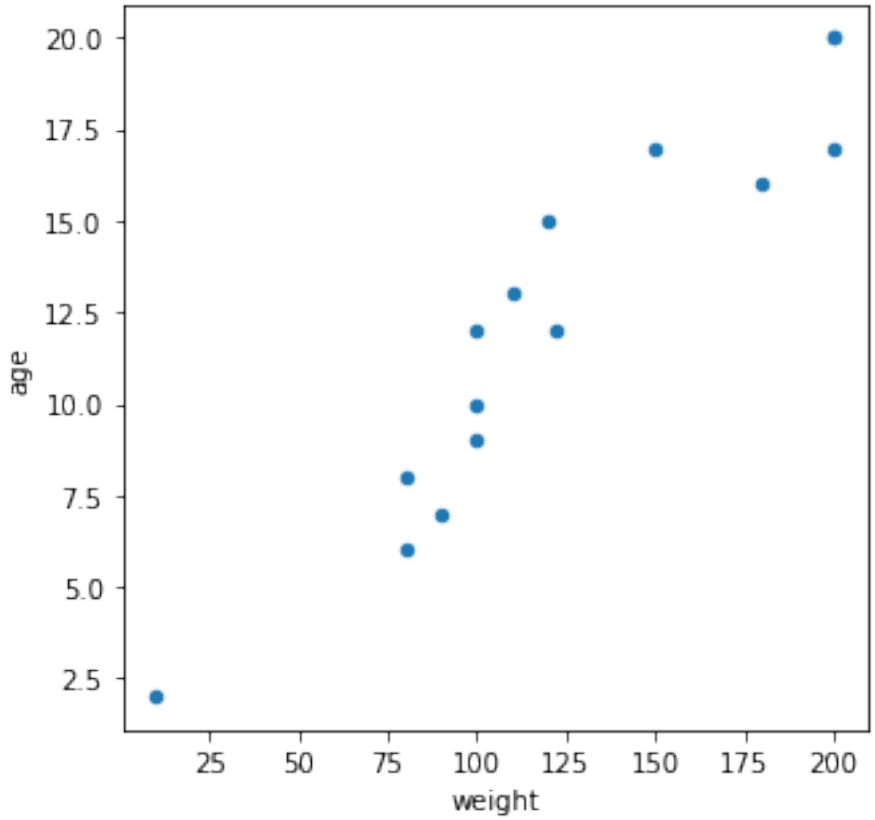
In [3]:
```
data.describe()
```

Out[3]:

	weight	age
count	15.000000	15.000000
mean	122.800000	12.266667
std	54.173267	5.297798
min	10.000000	2.000000
25%	95.000000	8.500000
50%	110.000000	12.000000
75%	165.000000	16.500000
max	200.000000	20.000000

```
In [4]: data.plot(kind='scatter', x='weight', y='age', figsize=(5,5))

Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0x115231048>
```




```
In [5]: def computeCost(X, y, theta):
            inner = np.power(((X * theta.T) - y), 2)
            return np.sum(inner) / (2 * len(X))

In [6]: data.insert(0, 'theOnes', 1)

In [7]: cols = data.shape[1]
        X = data.iloc[:,0:cols-1]
        y = data.iloc[:,cols-1:cols]

In [8]: print(X.head())
        print(y.head())

   theOnes  weight
0        1     150
1        1     100
2        1     110
```

```
3        1     122
4        1      80
   age
0   17
1    9
2   13
3   12
4    6
```

```
In [9]: X = np.matrix(X.values)
        y = np.matrix(y.values)
        theta = np.matrix(np.array([0,0]))
        theta
```

```
Out[9]: matrix([[0, 0]])
```

```
In [10]: X.shape, theta.shape, y.shape
```

```
Out[10]: ((15, 2), (1, 2), (15, 1))
```

```
In [11]: computeCost(X, y, theta)
```

```
Out[11]: 88.33333333333333
```

```
In [12]: def gradientDescent(X, y, theta, alpha, iters):
             temp = np.matrix(np.zeros(theta.shape))
             parameters = int(theta.ravel().shape[1])
             jhuAAP = np.zeros(iters)

             for i in range(iters):
                 error = (X * theta.T) - y

                 for j in range(parameters):
                     term = np.multiply(error, X[:,j])
                     temp[0,j] = theta[0,j] - ((alpha / len(X)) * np.sum(term))

                 theta = temp
                 jhuAAP[i] = computeCost(X, y, theta)

             return theta, jhuAAP
```

```
In [35]: alpha = 0.01
         iters = 67
         # Jim to Professor: Adding more iterations led to an error. Stackoverflow tells me this
```

```
In [36]: g, jhuAAP = gradientDescent(X, y, theta, alpha, iters)
         g
```

```
Out[36]: matrix([[3.01675102e+147, 4.37742847e+149]])
```

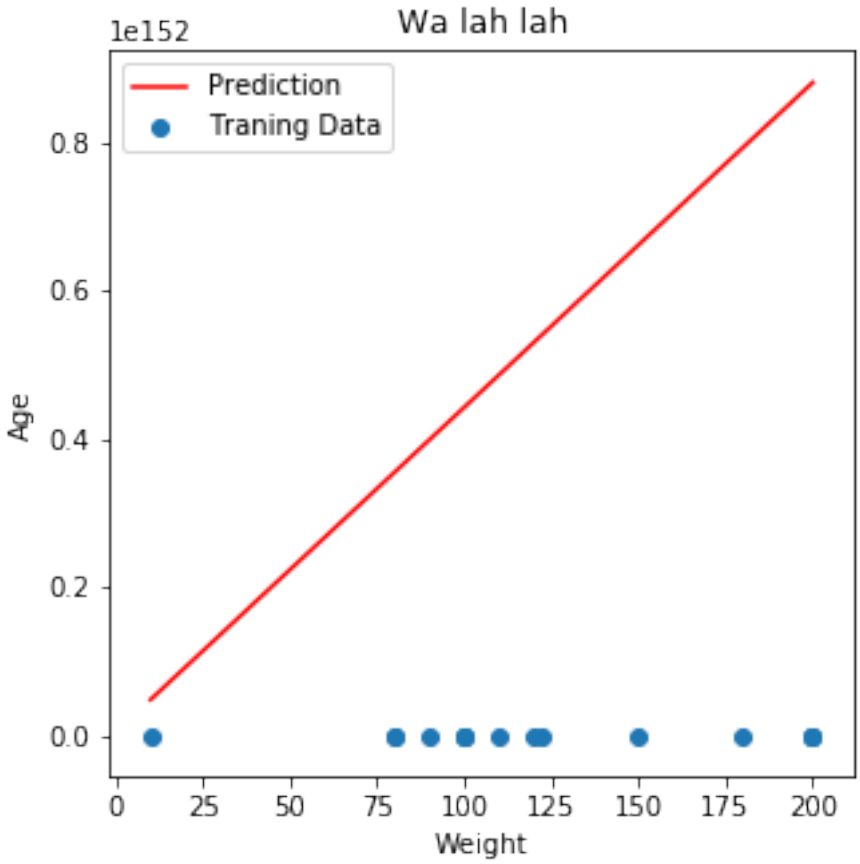
In [37]:
```
computeCost(X, y, g)
```

Out[37]:
```
1.7073834821157e+303
```

In [47]:
```
x = np.linspace(data.weight.min(), data.weight.max(), 80)
f = g[0, 1] + (g[0, 1] * x)

fig, ax = plt.subplots(figsize=(5,5))
ax.plot(x, f, 'r', label='Prediction')
ax.scatter(data.weight, data.age, label='Traning Data')
ax.legend(loc=2)
ax.set_xlabel('Weight')
ax.set_ylabel('Age')
ax.set_title('Wa lah lah')
```

Out[47]:
```
Text(0.5,1,'Wa lah lah')
```

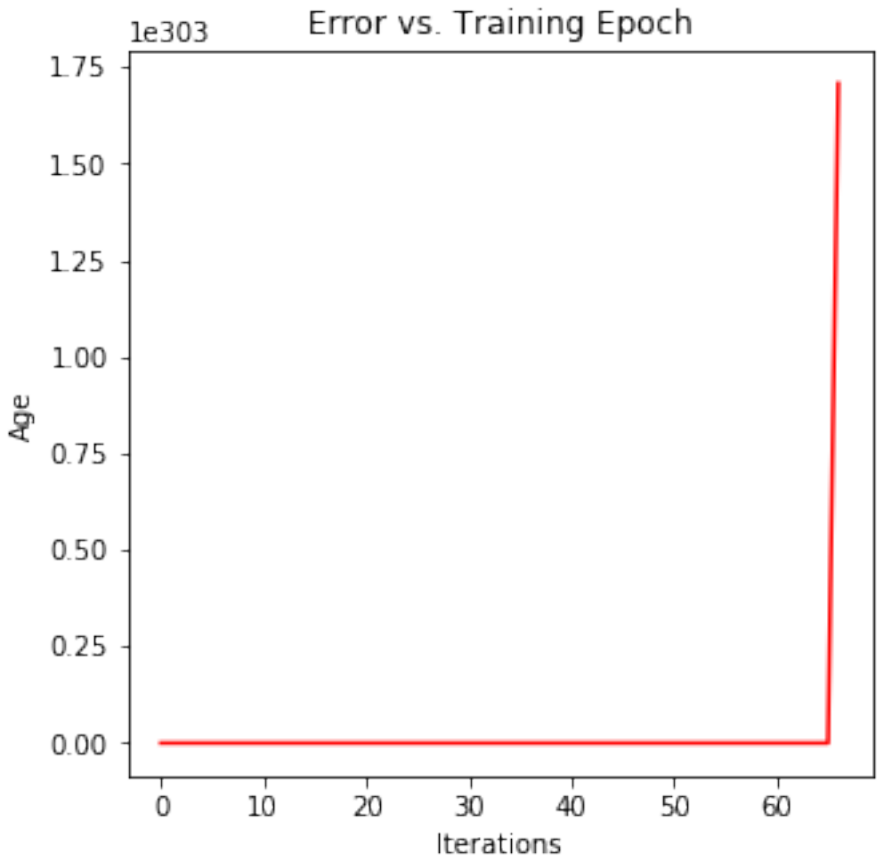


In [39]:
```
# Well, that's a fail.
```

In [48]:
```
fig, ax = plt.subplots(figsize=(5,5))
ax.plot(np.arange(iters), jhuAAP, 'r')
```

```
        ax.set_xlabel('Iterations')
        ax.set_ylabel('Age')
        ax.set_title('Error vs. Training Epoch')
```

Out[48]: Text(0.5,1,'Error vs. Training Epoch')



In [ ]: *# Okay, I don't know what I'm doing.*