##Importing the SFramme and reading the data

In [1]:
```
import pandas
import turicreate
from turicreate import SFrame
df = pandas.DataFrame()
sf = SFrame(data=df)
```

In [2]:
```
sf = SFrame(data='mirpur.csv')
```

```
Finished parsing file /home/muntasir/mirpur.csv

Parsing completed. Parsed 100 lines in 0.131341 secs.

------------------------------------------------------
Inferred types from first 100 line(s) of file as
column_type_hints=[int,int,float,int,int,float,int,str,int,int,int,str,int,int,int,str,int,int,float]
If parsing fails due to incorrect types, you can correct
the inferred type list above and pass it to read_csv in
the column_type_hints argument
------------------------------------------------------

Finished parsing file /home/muntasir/mirpur.csv

Parsing completed. Parsed 164 lines in 0.08487 secs.
```

##Output of the data

In [3]:
```
sf
```

Out[3]:

SN	Altitude	Temperature	Humidity	CO2	CO	Air	X8	AltTest	Tfixed	Hfixed	.1	Afixed	TempTest
299	20	19.2	91	51	3.83	60		1	25	75		10	15
300	20	19.2	92	51	3.83	60		2	25	75		10	16
301	20	19.1	92	52	3.83	60		3	25	75		10	17
302	20	19.0	92	51	3.83	60		4	25	75		10	18
303	20	19.0	92	52	3.84	61		5	25	75		10	19
304	20	19.0	92	51	3.83	61		6	25	75		10	20
305	20	19.1	92	51	3.84	61		7	25	75		10	21
306	20	19.0	92	51	3.83	61		8	25	75		10	22
307	20	19.0	92	50	3.84	61		9	25	75		10	23
308	20	19.2	93	50	3.83	60		10	25	75		10	24

Hfixed2	.2	Afixed2	Tfixed2	Humtest
75		10	25	90.0
75		10	25	89.5
75		10	25	89.0
75		10	25	88.5
75		10	25	88.0
75		10	25	87.5
75		10	25	87.0
75		10	25	86.5
75		10	25	86.0
75		10	25	85.5

[164 rows x 19 columns]
Note: Only the head of the SFrame is printed.
You can use print_rows(num_rows=m, num_columns=n) to print more rows and columns.

##Creating a scatter plot for humidity vs co

In [5]:
```
turicreate.visualization.set_target(target='auto')
```

In [6]:
```
turicreate.visualization.scatter(sf["Humidity"],sf["CO"])
```

Out[6]:
```
<turicreate.visualization._plot.Plot at 0x7f05302ce080>
```

##spliting data for training and testing

0.2 for testing and 0.8 for trainning

In [7]:
```
testing,training=sf.random_split(0.8,seed=0)
```

In [11]:
```
our_model = turicreate.linear_regression.create(training, target='CO', features=['Humidity'])
```

```
Linear regression:

--------------------------------------------------------

Number of examples          : 23

Number of features          : 1

Number of unpacked features : 1

Number of coefficients    : 2

Starting Newton Method

--------------------------------------------------------

+-----------+----------+--------------+--------------------+---------------------------------+

| Iteration | Passes   | Elapsed Time | Training Max Error | Training Root-Mean-Square Error |

+-----------+----------+--------------+--------------------+---------------------------------+

| 1         | 2        | 0.001448     | 0.839591           | 0.227615                        |

+-----------+----------+--------------+--------------------+---------------------------------+

SUCCESS: Optimal solution found.
```

##Getting the mean and error from test data

In [12]:
```
print(testing['CO'].mean())
```

```
3.8612056737588647
```

In [13]:
```
print(our_model.evaluate(testing))
```

```
{'max_error': 0.8695905185427226, 'rmse': 0.21351620343699856}
```

##Plotting and observing data with matpolotlib

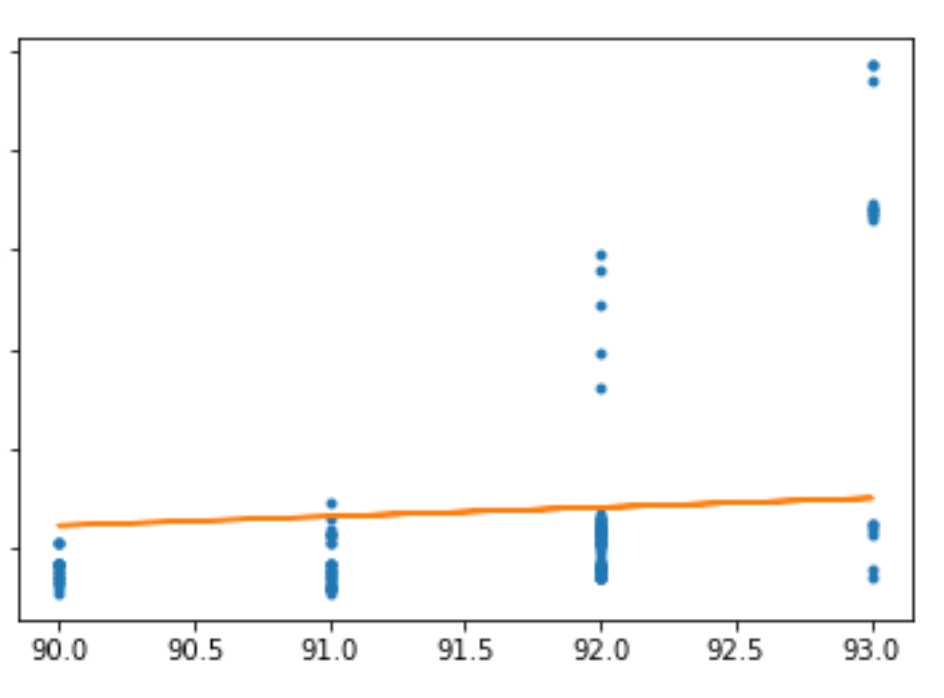
In [15]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [16]:
```
import numpy as np
import random
import matplotlib.pyplot as plt
%matplotlib inline
```

##regression line from the testing data

In [17]:
```
plt.plot(testing['Humidity'],testing['CO'],'.',testing['Humidity'],our_model.predict(testing),'-')
```

Out[17]:
```
[<matplotlib.lines.Line2D at 0x7f050e8d4748>,
 <matplotlib.lines.Line2D at 0x7f050e8d49e8>]
```



##cofficients for the predicted line

In [18]:
```
coefficients = sqft_model.coefficients
```

In [19]:
```
print(coefficients)
```

```
+-------------+-------+----------------------+---------------------+
|     name    | index |        value         |        stderr       |
+-------------+-------+----------------------+---------------------+
| (intercept) |  None |  2.1972134535534096  |  6.088871746201955  |
|   Humidity  |  None | 0.018313935783912553 | 0.06640084899547898 |
+-------------+-------+----------------------+---------------------+
[2 rows x 4 columns]
```

##multiple variables for multiple regression</h2>

In [20]:
```
multiple_features=['CO2','Air','Altitude']
```

##output of the data

In [22]:
```
sf[multiple_features].show()
```

```
Materializing SFrame
```

##creating regression with multiple variables

In [23]:
```
multiple_regression_model = turicreate.linear_regression.create(training, target='CO', features=multiple_features)
```

```
Linear regression:

--------------------------------------------------------

Number of examples          : 23

Number of features          : 3

Number of unpacked features : 3

Number of coefficients    : 4

Starting Newton Method

--------------------------------------------------------

+-----------+----------+--------------+--------------------+---------------------------------+

| Iteration | Passes   | Elapsed Time | Training Max Error | Training Root-Mean-Square Error |

+-----------+----------+--------------+--------------------+---------------------------------+

| 1         | 2        | 0.000540     | 0.157295           | 0.079147                        |

+-----------+----------+--------------+--------------------+---------------------------------+

SUCCESS: Optimal solution found.
```

##RMSE error evaluation

In [24]:
```
print(our_model.evaluate(testing))
```

```
{'max_error': 0.8695905185427226, 'rmse': 0.21351620343699856}
```

In [25]:
```
print(multiple_regression_model.evaluate(testing))
```

```
{'max_error': 0.19176832982167857, 'rmse': 0.08085509507326105}
```

##prediction from the tseting data

random row with a CO of 3.84 and predicting with two models(single and multiple regression)

In [27]:
```
test1=sf[sf['SN']]
```

In [30]:
```
random_data1=test1[8]
```

In [31]:
```
print(random_data1)
```

```
{'SN': 307, 'Altitude': 20, 'Temperature': 19.0, 'Humidity': 92, 'CO2': 50, 'CO': 3.84, 'Air': 61, 'X8': '', 'AltTest': 9, 'Tfi
xed': 25, 'Hfixed': 75, '.1': '', 'Afixed': 10, 'TempTest': 23, 'Hfixed2': 75, '.2': '', 'Afixed2': 10, 'Tfixed2': 25, 'Humtes
t': 86.0}
```

In [32]:
```
print(our_model.predict(random_data1))
```

```
[3.8820955456733643]
```

In [33]:
```
print(multiple_regression_model.predict(random_data1))
```

```
[3.754597483612408]
```

(It is clearly shown that multiple regression is bad at predicting from our dataset)

In [ ]: